Κεφαλαιο 6

Δομες Δεδομενων και Αλγοριθμοι Δεικτοδοτησης

ΠΑΤΡΑ
24 Αυγούστου 2005

1

Εισαγωγή

Οι συμβολοσειρές, δηλαδή οι ακολουθίες αλφαριθμητικών χαρακτήρων είναι ένας τύπος δεδομένων ο οποίος συναντάται συχνά στην επιστήμη της πληροφορικής σε πολλές, αν όχι σε όλες, τις εφαρμογές της. Προσπαθώντας να ορίσουμε τι είναι συμβολοσειρά θα λέγαμε ότι :

Ορισμός 1.1. *Συμβολοσειρά είναι μια ακολουθία συμβόλων, διαλεγμένων από ένα ή περισσότερα αλφάβητα, τα οποία παραθέτονται από τα αριστερά προς τα δεξιά.*

Με βάση τον ορισμό αυτό τα "hello world", "καλησπέρα", και "acgtttcatccctaag" είναι συμβολοσειρές που έχουν σχηματισθεί από το λατινικό, το ελληνικό και το αλφάβητο του DNA αντίστοιχα. Η συμβολοσειρά "το σύνολο της χρέωσης είναι 12.456 ευρώ" έχει σχηματισθεί από το ελληνικό και το αλφάβητο του δεκαδικού συστήματος αρίθμησης ($\{0, 1, \cdots, 9\}$).

Η διαχείριση και η επεξεργασία συμβολοσειρών απαντάται σε πολλές εφαρμογές όπως οι ψηφιακές βιβλιοθήκες, οι κατάλογοι προϊόντων, οι καταχωρήσεις των τηλεφώνων του χρυσού οδηγού, τα πελατολόγια διαφόρων εταιριών, τα δημοτολόγια, οι βάσεις δεδομένων γονιδιωμάτων ή πρωτεϊνών (DNA sequences, protein databases), τα αρχεία λειτουργίας (log files) ενός υπολογιστικού συστήματος καθώς και το πιο χαρακτηριστικό παράδειγμα ο παγκόσμιος ιστός (world wide web). Σε κάθε μία από αυτές τις εφαρμογές υπάρχει ήδη ένας μεγάλος αριθμός εγγράφων[1], όπου κάθε έγγραφο μπορεί να διαθέτει αυθαίρετα μεγάλο μήκος, και καθημερινά προστίθενται νέα ή μεταβάλλονται τα ήδη υπάρχοντα. Το ερώτημα που προκύπτει είναι μέσα σε τόσο μεγάλες συλλογές κειμένου πως θα ψάξω γρήγορα για σημεία που με ενδιαφέρουν, π.χ. για τις σελίδες του παγκόσμιου ιστού που αναφέρουν τον όρο *αλγόριθμος* ή τα σημεία του ανθρώπινου γονιδιώματος στα οποία εμφανίζεται η συμβολοσειρά "acatcct" η οποία ενδεχομένως να έχει βιολογικό ενδιαφέρον. Συνεπώς το πρόβλημα το οποίο ανακύπτει είναι το εξής :

Πρόβλημα 1.2. Σε ένα σύνολο k συμβολοσειρών $S = \{T_1, T_2, \cdots, T_k\}$ αναζητούνται οι θέσεις που εμφανίζεται μια συμβολοσειρά P σε κάθε μία από τις

[1] Για παράδειγμα ο παγκόσμιος ιστός υπολογίζεται σε τουλάχιστον 203 εκατομμύρια σελίδες [BKM+00], που προσεγγιστικά αυτό ισοδυναμεί με τουλάχιστον ένα κείμενο μήκους 10^{12} χαρακτήρων.

συμβολοσειρές. Η P συνήθως καλείται κλειδί ερωτήματος (pattern) και το πρόβλημα ταίριασμα κλειδιού ερωτήματος (pattern matching)

Η απλοϊκή αναζήτηση, η οποία για κάθε θέση μιας συμβολοσειράς από τα αριστερά προς τα δεξιά ελέγχει, έναν προς έναν τους χαρακτήρες του κλειδιού ερωτήματος και της συμβολοσειράς για να διαπιστώσει αν υπάρχει εμφάνιση, σε καμία περίπτωση δεν μπορεί να εφαρμοσθεί εξαιτίας της μεγάλης χρονικής πολυπλοκότητας $O(nm)$ όπου n το μήκος της συμβολοσειράς και m το μήκος του κλειδιού. Επίσης οι αλγόριθμοι αναζήτησης όπως του Knuth-Morris-Pratt [KMP77] ή του Boyer-Moore [BM77], οι οποίοι βελτιώνουν τον παραπάνω απλοϊκό αλγόριθμο μετακινώντας το κλειδί παραπάνω από μία θέση σε κάθε επανάληψη, παρόλο που στην χειρότερη και στη μέση αντίστοιχα είναι γραμμικοί με χρόνο $O(n+m)$ δεν ενδείκνυται να χρησιμοποιηθούν παρά σε μικρό εύρος εφαρμογών. Το ότι η χρονική τους πολυπλοκότητα εξαρτάται γραμμικά από το μήκος της συμβολοσειράς μέσα στην οποία ψάχνουμε τους κάνει πολύ αργούς για χρήση σε συστήματα πραγματικού χρόνου. Έτσι λοιπόν στόχος είναι η μείωση του χρόνου αναζήτησης με τίμημα κάποιο αρχικό χρόνο προεπεξεργασίας και χώρο για την αποθήκευση της ενδιάμεσης πληροφορίας που θα βοηθήσει στην γρηγορότερη απάντηση. Στόχος λοιπόν είναι η ανάπτυξη δομών δεδομένων δεικτοδότησης ή αλλιώς καταλόγων (indexes).

Ως δομή δεδομένων δεικτοδότησης ή αλλιώς κατάλογος θα μπορούσε να χαρακτηρισθεί οτιδήποτε μας βοηθάει να βρούμε την απάντηση μας γρηγορότερα από το να διαβάσουμε όλα τα δεδομένα. Χαρακτηριστικό παράδειγμα καταλόγου είναι η οργάνωση των λεξικών κατά αλφαβητική σειρά η οποία μας βοηθάει να βρούμε το λήμμα που αναζητούμε χωρίς να χρειαστεί να διαβάσουμε όλα τα λήμματα από το Α μέχρι το Ω. Σύμφωνα με τον Ferragina [Fer02] ένας κατάλογος είναι μία δομή δεδομένων η οποία μας επιτρέπει κατά το χρόνο της ερώτησης να επικεντρώσουμε την αναζήτηση για ένα κλειδί ερωτήματος σε ένα μικρό μέρος των δεδομένων και πιο συγκεκριμένα κοντά στις θέσεις που βρίσκεται αυτό το κλειδί.

Στην παρούσα εργασία γίνεται προσπάθεια να παρουσιασθούν οι βασικές δομές δεικτοδότησης για συμβολοσειρές, να παρατεθούν τα πλεονεκτήματα και τα μειονεκτήματα της κάθε μίας καθώς και οι συνθήκες στις οποίες κάθε τέτοια δομή ενδείκνυται να χρησιμοποιηθεί.

1.1 Πλαίσιο Μελέτης

Οι δομές δεικτοδότησης συμβολοσειρών που θα παρουσιασθούν εξετάζονται ως προς τρεις βασικούς άξονες:

→ **Τον τύπο των συμβολοσειρών που μπορούν να δεικτοδοτήσουν και τα ερωτήματα που μπορούν να απαντήσουν αποδοτικά.** Στην βιβλιογραφία οι δομές δεικτοδότησης συμβολοσειρών χωρίζονται σε δύο μεγάλες κατηγορίες: τις δομές που δεικτοδοτούν συμβολοσειρές/κείμενα φυσικού λόγου (linguistic texts) και τις δομές πλήρου ς δεικτοδότησης (full-text indexes). Οι δομές της πρώτης κατηγορίας, οι οποίοι συνήθως

καλούνται κατάλογοι βασισμένοι σε λέξεις (word based indexes) με κύριους αντιπρόσωπους τα αντεστραμμένα αρχεία (inverted files), τα αρχεία υπογραφών (signature-files) και τα bitmaps, εκμεταλλεύονται κυρίως διάφορα χαρακτηριστικά που έχουν τα κείμενα φυσικής γλώσσας με κεντρικό το γεγονός ότι τα κείμενα αυτά μπορούν να διασπαστούν σε στοιχειώδεις συμβολοσειρές, τις λέξεις. Οι δομές αυτές απαντούν γρήγορα ερωτήματα λέξεων σε τέτοιου είδους κείμενα και συνήθως δεν απαιτούν υπερβολικά επιπρόσθετο χώρο, όμως σε καμιά περίπτωση δεν μπορούν να εφαρμοσθούν σε τυχαία κείμενα όπως βιολογικές ή μουσικές ακολουθίες ή για αναζήτηση κλειδιών που δεν είναι στοιχειώδεις λέξεις. Τις περιπτώσεις που αποτυγχάνουν να εξυπηρετήσουν οι δομές της πρώτης κατηγορίας έρχονται να καλύψουν οι δομές πλήρους δεικτοδότησης, με κύριους αντιπρόσωπους τα pat trees, τα δένδρα επιθεμάτων (suffix trees) και τους πίνακες επιθεμάτων (suffix arrays). Οι δομές αυτές μπορούν να δεικτοδοτήσουν αυθαίρετες συμβολοσειρές και να απαντήσουν πιο γενικά ερωτήματα με τίμημα ότι συνήθως χρησιμοποιούν παραπάνω μνήμη από ότι οι δομές τις πρώτης κατηγορίας. Στην εργασία αυτή ασχολούμαστε με την κατηγορία δομών δεδομένων πλήρους δεικτοδότησης κειμένων εξαιτίας της δυνατότητας να απαντούν σε πιο σύνθετα και ευρεία ερωτήματα και να επιλύουν προβλήματα ιδιαίτερα από τον χώρο της μοριακής βιολογίας [Gus97]. Στην επόμενη ενότητα της εισαγωγής παρουσιάζονται συνοπτικά οι κατάλογοι βασισμένοι σε λέξεις καθώς και οι τύποι ερωτημάτων που μπορούν να απαντήσουν.

→ **Την συμπεριφορά τους στη δευτερεύουσα μνήμη.** Σε πολλές εφαρμογές το μέγεθος των δεδομένων είναι τόσο μεγάλο έτσι ώστε να επιβάλλεται η χρησιμοποίηση της δευτερεύουσας μνήμης. Σε αυτές τις περιπτώσεις τόσο τα δεδομένα όσο και οι δομές δεικτοδότησης αποθηκεύονται σε δίσκους μαγνητικούς ή οπτικούς ή σε μαγνητικές ταινίες. Κύριο χαρακτηριστικό αυτών των μέσων είναι οι μεγάλοι χρόνοι προσπέλασης ενός στοιχείου οι οποίοι μπορεί και να είναι έως και 10^5 φορές πιο μεγάλοι από τον χρόνο προσπέλασης ενός στοιχείου στην κύρια μνήμη. Επειδή από την δευτερεύουσα μνήμη τα στοιχεία προσκομίζονται υπό μορφή σελίδων (κάθε σελίδα περιέχει έναν σταθερό αριθμό στοιχείων B) αναγκαίο είναι η πληροφορία και οι κατάλογοι να διαταχθούν κατάλληλα σε αυτά τα μέσα έτσι ώστε να ελαχιστοποιείται ο αριθμός προσπελάσεων. Σε αυτό το περιβάλλον λειτουργίας οι δομές δεικτοδότησης συγκρίνονται κατά κύριο λόγο με τον αριθμό προσπελάσεων στην δευτερεύουσα μνήμη για κάθε πράξη που υποστηρίζουν, με την χρονική τους πολυπλοκότητα (cpu time) καθώς και με τις σελίδες που καταλαμβάνουν επιπρόσθετα από τα δεδομένα

1.2 Λεξική Δεικτοδότηση (word based indexes)

Σε αυτή την ενότητα θα παρουσιασθούν συνοπτικά οι κατάλογοι κειμένων φυσικής γλώσσας, οι οποίοι στηρίζονται στο γεγονός ότι τα κείμενα αποτελούνται από στοιχειώδεις υποσυμβολοσειρές τις λέξεις (word based indexes). Οι τρεις πιο δημοφιλείς κατάλογοι της κατηγορίας αυτής είναι τα αντεστραμμένα αρχεία (inverted files), τα αρχεία υπογραφών (signature-files) και τα bitmap-

s (για εκτενή παρουσίαση δείτε το [WMB99]) με τα αντεστραμμένα αρχεία να επικρατούν. Μετά από εκτενή πειραματικά αποτελέσματα [ZMK98] διαπιστώθηκε ότι είναι πιο αποδοτικά όσον αφορά το χρόνο απάντησης ερωτημάτων αλλά και τον επιπρόσθετο χώρο που καταλαμβάνουν.

Ορισμός 1.3. *Ένα **αντεστραμμένο αρχείο** αποτελείται από δύο βασικά συστατικά: το λεξιλόγιο ή λεξικό (vocabulary ή lexicon), το οποίο είναι το σύνολο των λέξεων που συναντάμε στα κείμενα που θέλουμε να δεικτοδοτήσουμε καθώς, και τις αντεστραμμένες λίστες, μία για κάθε όρο του λεξιλογίου. Η αντεστραμμένη λίστα μιας λέξης t του λεξιλογίου είναι μια λίστα όπου σημειώνονται με αύξουσα σειρά τα έγγραφα στα οποία αυτή εμφανίζεται.*

Παράδειγμα 1.4. Έστω ότι έχουμε ένα σύνολο από τρία κείμενα:

$t_1 =$"παγωτό βανίλια λεμόνι"
$t_2 =$"παγωτό βανίλια σοκολάτα"
$t_3 =$"παγωτό σοκολάτα σοκολάτα παρφέ"

τότε το ανεστραμμένο αρχείο για αυτά θα είναι:

Λεξιλόγιο	Συχνότητα Όρου	Ανεστραμμένες Λίστες
βανίλια	2	$\{1, 2\}$
λεμόνι	1	$\{1\}$
παγωτό	3	$\{1, 2, 3\}$
σοκολάτα	3	$\{2, 3\}$
παρφέ	1	$\{3\}$

Αν τώρα το ερώτημα μας είναι ο όρος "σοκολάτα" τότε από το αντεστραμμένο αρχείο αμέσως μπορούμε να δούμε, από την αντίστοιχη αντεστραμμένη λίστα, ότι βρίσκεται στα κείμενα t_2, t_3.

Τα αντεστραμμένα αρχεία μπορούν επίσης να απαντήσουν και σε πιο σύνθετα ερωτήματα όπως π.χ. βρες τα κείμενα που περιέχουν τον όρο t και t' οπότε και λαμβάνεται η τομή των δύο αντεστραμμένων λιστών για αυτούς τους όρους. Αν αναζητούμε τα κείμενα που δεν περιέχουν έναν όρο τότε απαντώνται όλες οι λίστες πλην αυτή του συγκεκριμένου όρου. Τέλος μπορεί να αναζητούνται τα κείμενα που είτε εμφανίζεται ένας όρος είτε ένας άλλος οπότε λαμβάνεται η ένωση των ανεστραμμένων λιστών τους.

Σε εφαρμογές που μας ενδιαφέρει και η θέση του κάθε όρου μέσα σε κάθε κείμενο τότε στις αντεστραμμένες λίστες δίπλα από κάθε εγγραφή κειμένου σημειώνονται και οι θέσεις που εμφανίζονται μέσα σε αυτό. Έτσι π.χ. για τον όρο "σοκολάτα" η ανεστραμμένη λίστα θα είναι $\{2@[16], 3@[8, 16]\}$. Με αυτό το πιο εκλεπτυσμένο σχήμα το μέγεθος του καταλόγου τουλάχιστον διπλασιάζεται αφού σημειώνεται και τουλάχιστον μία θέση δίπλα από κάθε κείμενο.

Αξίζει να παρατηρήσουμε ότι το μέγεθος ενός ανεστραμμένου αρχείου είναι υπο-γραμμικό δηλαδή καταλαμβάνει χώρο το πολύ όσο και το μέγεθος το κειμένων που δεικτοδοτεί. Αν ένας όρος του λεξιλογίου έχει μέγεθος z χαρακτήρες και εμφανίζεται x φορές μέσα στα κείμενα τότε στο ανεστραμμένο αρχείο αποθηκεύεται μία φορά καταλαμβάνοντας χώρο z και όχι zx χαρακτήρων.

Τώρα όταν χρησιμοποιούνται οι μη-εκλεπτυσμένες ανεστραμμένες λίστες καταλαμβάνονται επιπρόσθετα και $x \log k$ $bits$ όπου k είναι το πλήθος των κειμένων. Έτσι ο συνολικός χώρος που καταλαμβάνει ένας όρος είναι $8z + x \log k$ $bits$ το οποίο εν δυνάμει μπορεί να είναι λιγότερο από $8zx$ $bits$ που καταλαμβάνει συνολικά μέσα στα δεδομένα. Σύμφωνα με το [WMB99] για τυπικά αγγλικά κείμενα ένα ανεστραμμένο αρχείο αυτού του τύπου καταλαμβάνει χώρο ίσο με το 50% μέχρι 100% του χώρου της πληροφορίας. Για την εξοικονόμηση χώρου ο δείκτης που υποδεικνύει την θέση ενός όρου μέσα σε ένα κείμενο της συλλογής μπορεί να μην είναι η θέση του χαρακτήρα που ξεκινά αλλά η σειρά της λέξης. Με αυτό τον τρόπο χρησιμοποιούνται μικρότεροι σε $bits$ δείκτες. Ένας επιπρόσθετος τρόπος εξοικονόμησης χώρου είναι λεκτικοί όροι όπως τα άρθρα ή οι σύνδεσμοι να μην ενσωματώνονται στο λεξιλόγιο. Παραδείγματος χάρη η αναζήτηση για το άρθρο "το" θα επιστρέψει μεγάλο αριθμό απαντήσεων, ίσως και ολόκληρη την συλλογή, χωρίς καμία ιδιαίτερη αξία. Τέλος τόσο για την εξοικονόμηση χώρου αλλά και την βελτίωση των απαντήσεων όλοι οι όροι του λεξιλογίου γράφονται με μικρά (έτσι οι όροι "Σοκολάτα" και "σοκολάτα" δεν υπάρχουν ταυτόχρονα) και όπως επίσης γίνεται προσπάθεια να αποθηκεύονται σαν όροι οι ρίζες μιας λέξης. Αυτό γίνεται για να μην υπάρχουν λέξεις μέσα στην δομή με διαφορετικές καταλήξεις.

Ο χώρος που καταλαμβάνουν τα ανεστραμμένα αρχεία μπορεί να μειωθεί ακόμη περισσότερο συμπιέζοντας με διάφορες μεθόδους τις ανεστραμμένες λίστες. Όπως είδαμε προηγουμένως μια ανεστραμμένη λίστα είναι μία λίστα από αριθμούς $\{t_1, t_2, \cdots, t_x\}$. Αν αποθηκευόταν η λίστα σημειώνοντας τον αριθμό του πρώτου κειμένου και μετά τις διαφορές των γειτονικών αριθμών θα έπαιρνε την μορφή $\{t_1, t_2 - t_1, \cdots, t_x - t_{x-1}\}$ η οποία είναι ακριβώς αντίστοιχη της προηγούμενης αφού τα στοιχεία προσπελάζονται από τα αριστερά προς τα δεξιά. Οι διαφορές έχουν το πολύ μέγεθος k, όσο και το πλήθος των κειμένων, έτσι το πολύ να χρειάζονται $\log k$ $bits$. Σε εφαρμογές όπου οι διαφορές είναι μικρές υπάρχει μεγάλη εξοικονόμηση σε χώρο. Υπάρχει μεγάλος αριθμός τρόπων κωδικοποίησης των διαφορών αυτών (ανατρέξτε στο [WMB99] για αναλυτική περιγραφή) που προσπαθούν να δώσουν μικρές αναπαραστάσεις σε $bits$ μικρών διαφορών που εμφανίζονται συχνά και μεγαλύτερες αναπαραστάσεις στις μεγάλες και όχι τόσο συχνές διαφορές.

Η πιο διαδεδομένη κωδικοποίηση είναι η Golomb η οποία είναι κατάλληλη για ακέραιους που είναι κατανεμημένοι σύμφωνα με γεωμετρική πρόοδο, κατά την οποία μια διαφορά x εμφανίζεται με πιθανότητα $Pr(x) = (1-p)^{x-1}p$ για δεδομένη πιθανότητα p. Μια διαφορά $x \geq 1$ κωδικοποιείται από ένα μοναδικό κωδικό $code(x) = qr$ που σχηματίζεται από το πηλίκο $q = (x-1)div\ b$ (αυτό είναι $q-1$ $bits$ ακολουθούμενα από ένα μηδέν) ακολουθούμενο από υπόλοιπο $r = (x-1)mod\ b$ σε δυαδική αναπαράσταση. Το b συνήθως είναι δύναμη του δύο. Επομένως για διαφορές 2,3,20 και με $b = 4$ θα έχουμε τους κωδικούς "0 01","0 10","11110 11" αντίστοιχα ενώ για $b = 8$ "0 001","0 010","110 011". Όπως παρατηρούμε για μικρές διαφορές δίνονται μικροί κωδικοί ενώ για μεγάλες κωδικοί με περισσότερα $bits$. Αν το $b = \frac{\log 2}{p} = 0.69\frac{N}{f_t}$, όπου N είναι ο αριθμός των κειμένων της συλλογής και f_t είναι ο αριθμός αυτών που περιέχουν τον όρο t. Με πειράματα [WMB99] διαπιστώθηκε ότι κάθε διαφορά καταλάμβανε περίπου 6 $bits$.

Η κατασκευή ενός ανεστραμμένου αρχείου είναι μια διαδικασία αρκετά σύνθετη επειδή δεν μπορεί να γίνει εξολοκλήρου στην κύρια μνήμη ενός υπολογιστικού συστήματος. Φανταστείτε ότι χτίζεται ένας κατάλογος πάνω στην Βίβλο, όπου κάθε ενότητα της λαμβάνεται ως ξεχωριστό κείμενο , επομένως υπάρχουν 31.101 κείμενα, και εμφανίζει 8.965 όρους. Ο πίνακας του ανεστραμμένου αρχείου σε αυτή την περίπτωση θα έχει μέγεθος τουλάχιστον $31.101 \times 8.965\ bytes \approx 4.7 \times 10^9\ bytes = 4.7\ Gbytes$ με αποτέλεσμα δύσκολα να χωράει στην κύρια μνήμη κάποιου συστήματος. Σειρά αλγορίθμων (ανατρέξτε στο [WMB99]) χτίζουν ένα αρχείο τμηματικά χρησιμοποιώντας την κύρια μνήμη και την δευτερεύουσα ταυτόχρονα.

Ο δεύτερος τύπος καταλόγων αυτής της κατηγορίας είναι τα αρχεία υπογραφών (signature-files) .

Ορισμός 1.5. *Σε ένα* ***αρχείο υπογραφών*** *κάθε κείμενο της συλλογής λαμβάνει μια υπογραφή ή οποία είναι μια συμβολοσειρά από bits που περιγράφουν κατά κάποιο τρόπο το περιεχόμενο του κειμένου. Έστω ότι χρησιμοποιείται υπογραφή μήκους d bits, για κάθε όρο του κειμένου λαμβάνονται μερικές τιμές, έστω x, από αντίστοιχο αριθμό διαφορετικών συναρτήσεων κατακερματισμού. Οι τιμές που απαντούν αυτές οι συναρτήσεις είναι οι θέσεις στην υπογραφή που το bit γίνεται* 1.

Έτσι αν έχει υπολογιστεί η υπογραφή ενός κειμένου και αναζητούμε τον όρο t χρησιμοποιώντας τις ίδιες συναρτήσεις κατακερματισμού βλέπουμε ποιες θέσεις για αυτόν τον όρο πρέπει να εμφανίζουν 1. Έστω και μια θέση από αυτές να είναι 0 τότε δεν υπάρχει ο όρος μέσα σε αυτό το κείμενο. Αν όλες οι θέσεις εμφανίζουν 1 τότε είτε υπάρχει ο όρος στο κείμενο είτε οι θέσεις αυτές ενεργοποιήθηκαν από άλλους όρους οπότε και πρέπει να ανακτηθεί το κείμενο και να αναζητηθεί αν όντως βρίσκεται μέσα σε αυτό ο όρος. Αυτό είναι ένα γεγονός που κάνει τα ερωτήματα πιο αργά σχετικά με τα ανεστραμμένα αρχεία. Με την κατάλληλη επιλογή του μήκους της υπογραφής καθώς και των συναρτήσεων κατακερματισμού η πιθανότητα να έχουμε εμφάνιση ενός όρου κατά την υπογραφή αλλά όχι μέσα στο κείμενο γίνεται πολύ μικρή (ανατρέξτε στα [WMB99],[Fal85]) .

Τέλος υπάρχουν και οι κατάλογοι bitmap οι οποίοι συνδυάζουν στοιχεία και από τους προηγούμενους δύο τύπους.

Ορισμός 1.6. *Ένα* ***bitmap*** *έχει δομή παρόμοια με τα ανεστραμμένα αρχεία. Στην θέση όμως των ανεστραμμένων λιστών υπάρχει μια συμβολοσειρά από k bits, όπου k το πλήθος των κειμένων. Το i−οστό bit γίνεται 1 αν ο όρος εμφανίζεται και στο i−οστό κείμενο.*

Για αναζήτηση ενός όρου t ελέγχοντας ποια *bit* είναι 1 διαπιστώνεται σε ποια κείμενα περιέχεται ο όρος. Για την εύρεση της θέσης του όρου μέσα σε κάθε κείμενο της απάντησης αναγκαία είναι η αναζήτηση, γεγονός που κάνει τα bitmaps πιο αργά σε σχέση με τα ανεστραμμένα αρχεία. Τέλος για μια συλλογή k κειμένων με N όρους ο χώρος που απαιτεί ένα bitmap είναι $k \times N\ bits$. Επειδή το k μπορεί να είναι αρκετά μεγάλο και επειδή αποθηκεύεται ένα *bit* για κάθε κείμενο (ακόμη και αν δεν υπάρχει ο όρος μέσα σε αυτό, παρατηρήστε ότι τα

ανεστραμμένα αρχεία αποθηκεύουν περίπου 1 με 2 *byte* μόνο για τα κείμενα που περιέχουν τον όρο) ο χώρος που απαιτείται για τον κατάλογο μπορεί να είναι πολύ μεγαλύτερος και από την πληροφορία αυτή καθαυτή.

1.3 Συμβολισμοί

Στην ενότητα αυτή παρατίθενται οι συμβολισμοί που θα χρησιμοποιηθούν στην συνέχεια:

Συμβολισμός	Περιγραφή
Σ	το αλφάβητο από το οποίο σχηματίζονται οι συμβολοσειρές
$\|.\|$	μέγεθος συνόλου ή μήκος συμβολοσειράς
S	το σύνολο k συμβολοσειρών που θέλουμε να δεικτοδοτήσουμε ($\|S\| = k$)
$T_{1\ldots n}$	συμβολοσειρά n χαρακτήρων ($\|T\| = n$)
T_i	ο i-οστος χαρακτήρας της συμβολοσειράς $T_{1\ldots n}$
$T_{i\ldots j}$	η υποσυμβολοσειρά της $T_{1\ldots n}$ που ξεκινά από την θέση i και καταλήγει στην j ($T_{i\ldots j} = T_i T_{i+1} \cdots T_j$)
$T_{i\ldots n}$	το i-οστο επίθεμα $suffix$) της $T_{1\ldots n}$ ($T_{i\ldots n} = T_i T_{i+1} \cdots T_n$)
$T_{1\ldots i}$	το i-οστο πρόθεμα ($prefix$) (της $T_{1\ldots n}$ ($T_{1\ldots i} = T_1 T_2 \cdots T_i$)
$P_{1\ldots m}$	κλειδί ερωτήματος ($pattern$) m χαρακτήρων ($\|P\| = m$)
I	δομή δεικτοδότησης πάνω στο S
$T = T'$	η ισότητα δυο συμβολοσειρών T,T΄ ισχύει ανν $\|T\| = \|T'\|$ και $T_i = T'_i, \forall i$
$<_L$	τελεστής λεξικογραφικής σειράς μεταξύ δύο συμβολοσειρών π.χ. "θάλασσα" $<_L$ "πανεπιστήμιο"
$LCP(T, T')$	μέγιστο κοινό πρόθεμα δύο συμβολοσειρών $LCP(T, T') = \{max\ i : T_{1\ldots i} = T'_{1\ldots i}\}$
$LCA(node_1, node_2)$	βαθύτερος κοινός πρόγονος δύο κόμβων σε μία δενδρική δομή
$Prefixes(T)$	το σύνολο των προθεμάτων μιας συμβολοσειράς
$Suffixes(T)$	το σύνολο των επιθεμάτων μιας συμβολοσειράς
ST_T	το δένδρο επιθεμάτων μιας συμβολοσειράς T
SA_T	ο πίνακας επιθεμάτων μιας συμβολοσειράς T
$T = T'T''$	συνένωση δύο συμβολοσειρών T' και T'' δηλ. $T = T'_1 T'_2 \cdots T'_{\|T'\|} T''_1 T''_2 \cdots T''_{\|T''\|}$
$i = 1 :_x n$	η μεταβλητή i λαμβάνει τιμές από το 1 μέχρι το n με βήμα x για $x = 1$ το x παραλείπεται.
$period(T) = \pi$	περίοδος συμβολοσειράς $T = \pi^t \pi'$ με π' πρόθεμα της π και $t \geq 0$

2

Πλήρης Δεικτοδότηση Συμβολοσειρών

Στην ενότητα αυτή θα εξετάσουμε αναλυτικά τις τρεις κύριες δομές πλήρους δεικτοδότησης συμβολοσειρών, τα δένδρα επιθεμάτων, τους πίνακες επιθεμάτων και τους άκυκλους κατευθυνόμενους γράφους λέξεων, με πρόθεση να καταγράψουμε όλα τα ιδιαίτερα χαρακτηριστικά τους και να αναδείξουμε τις δυνατότητές τους.

2.1 Δένδρα Επιθεμάτων (Suffix Trees)

Το δένδρο επιθεμάτων είναι η καλύτερη γνωστή δομή δεδομένων πλήρους δεικτοδότησης συμβολοσειρών η οποία παρουσιάσθηκε πρώτη φορά από τον Weiner το 1973 [Wei73]. Πρόκειται για την πιο παλιά και πιο μελετημένη δομή σε αυτό το πεδίο η οποία πέρα από το ότι απαντάει αποδοτικά στο *Πρόβλημα 1.2* έχει την δυνατότητα να απαντάει αποδοτικά πολύ πιο σύνθετα ερωτήματα. Στην ενότητα 2.1.4 θα δούμε ορισμένες από τις βασικές εφαρμογές του δένδρου επιθεμάτων ενώ για μια ευρεία παρουσίαση ανατρέξτε στα [Gus97, Nav01, GI93]. Βέβαια αυτή η καλή συμπεριφορά, όπως θα δούμε και στη συνέχεια, υπάρχει με τίμημα σημαντική επιπρόσθετη χρησιμοποίηση χώρου η οποία κυμαίνεται από 10 μέχρι 20 φορές παραπάνω από τα δεδομένα που δεικτοδοτούνται.

2.1.1 Ορισμός - Περιγραφή

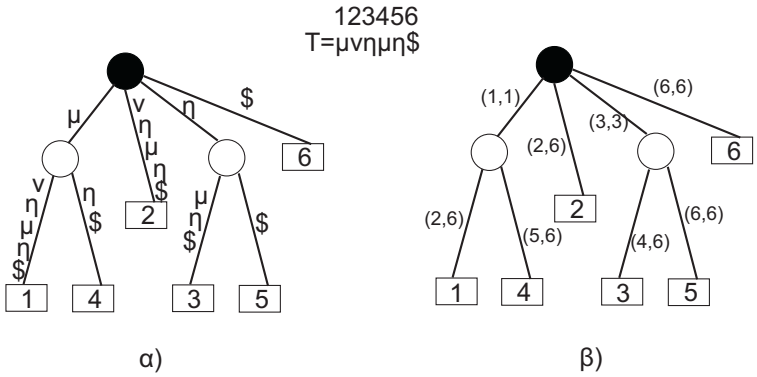
Ορισμός 2.1. *Το δένδρο επιθεμάτων ST_T μιας συμβολοσειράς T είναι ένα συμπαγές ψηφιακό δένδρο αναζήτησης (compacted digital search tree ή trie) που περιέχει ως κλειδιά όλα τα επιθέματα της $T\$$ (Suffixes(T)). Ο χαρακτήρας $\$$, ο οποίος είναι εκτός του αλφάβητου Σ από το οποίο κατασκευάστηκε η T, χρησιμοποιείται για να έχουν διαφορετική αναπαράσταση, σε ξεχωριστό φύλλο του δένδρου, δύο επιθέματα $T_{i\ldots n}$ και $T_{j\ldots n}$ με $i \neq j$ όταν το ένα τυγχάνει να είναι πρόθεμα του άλλου* [1]. *Στο φύλλο του δένδρου που αντιστοιχεί για το επίθεμα $T_{i\ldots n}$ αποθηκεύεται ο ακέραιος i.*

Παράδειγμα 2.2. Έστω ότι έχουμε τη συμβολοσειρά T =μνημη$ τα επιθέματα της είναι τα εξής:

[1] Από εδώ και στο εξής γίνεται η παραδοχή ότι κάθε συμβολοσειρά T, για την οποία κατασκευάζεται ένα δένδρο επιθεμάτων, έχει ως τερματικό χαρακτήρα τον $.

$$T_{1...6} = \mu\nu\eta\mu\eta\$$$
$$T_{2...6} = \nu\eta\mu\eta\$$$
$$T_{3...6} = \eta\mu\eta\$$$
$$T_{4...6} = \mu\eta\$$$
$$T_{5...6} = \eta\$$$
$$T_{6...6} = \$$$

το δένδρο επιθεμάτων της T φαίνεται στο σχήμα 2.1α) :



Σχήμα 2.1. Δένδρο Επιθεμάτων συμβολοσειράς "μνημη$"

Το δένδρο επιθεμάτων, όπως φαίνεται και στο παραπάνω παράδειγμα, για μια συμβολοσειρά μήκους n αποτελείται από n φύλλα. Όντας ένα k-δικο συμπαγές trie, όπου $k = |\Sigma| + 1$, κάθε εσωτερικός κόμβος θα έχει τουλάχιστον 2 και το πολύ k παιδιά. Συνεπώς οι εσωτερικοί κόμβοι και οι ακμές θα είναι το πολύ $n-1$ και $2n-2$ αντίστοιχα. Κάθε ακμή του δένδρου επιθεμάτων αναπαριστά και μια υποσυμβολοσειρά $T_{i...j}$ συνεπώς για εξοικονόμηση χώρου δεν είναι αναγκαίο να αποθηκεύονται πάνω σε αυτή όλοι οι χαρακτήρες παρά μόνα τα όρια (i, j). Έτσι προκύπτει η συμπαγής αναπαράσταση του σχήματος 2.1β. Αφού το δένδρο επιθεμάτων αποτελείται το πολύ από $2n-1$ κόμβους και $2n-2$ ακμές και κάθε ένα από τα στοιχεία αυτά έχει σταθερό μέγεθος, απαιτεί τελικά γραμμικό χώρο αποθήκευσης $O(n)$.

Το δένδρο επιθεμάτων διαθέτει τα εξής δομικά χαρακτηριστικά που απορρέουν από τον ορισμό του.

Ιδιότητα 2.3. Υπάρχουν n φύλλα, ένα φύλλο για κάθε επίθεμα της T. Η συνένωση των υποσυμβολοσειρών των πλευρών που συναντώνται κινούμενοι από την ρίζα προς ένα φύλλο i σχηματίζουν το επίθεμα $T_{i...n}$.

Ιδιότητα 2.4. Έστω δύο διαφορετικά επιθέματα $T_{i...n} = xa$ και $T_{j...n} = xb$, τα οποία μοιράζονται ένα κοινό πρόθεμα x. Στο δένδρο επιθεμάτων τα δύο φύλλα που αντιστοιχούν στα δύο αυτά επιθέματα έχουν ένα κοινό πρόγονο u για τον όποιο η συνένωση των υποσυμβολοσειρών των πλευρών που συναντώνται κινούμενοι από την ρίζα προς τον u σχηματίζει το κοινό πρόθεμα x. Αυτό μπορεί να διατυπωθεί και αντίστροφα. Για κάθε εσωτερικό κόμβο u, ενός δένδρου επιθεμάτων, τα επιθέματα που αντιστοιχούν σε φύλλα του υποδένδρου του

μοιράζονται ένα κοινό πρόθεμα x το οποίο αναπαρίσταται από τις πλευρές του μονοπατιού από την ρίζα προς τον κόμβο. Η υποσυμβολοσειρά που προκύπτει από τη συνένωση των υποσυμβολοσειρών των πλευρών που συναντώνται κινούμενοι από την ρίζα προς τον u καλείται *ετικέτα μονοπατιού* (path label) του κόμβου u.

Ας δούμε τώρα πως αντιμετωπίζει το δένδρο επιθεμάτων το κύριο πρόβλημα του ταιριάσματος (*Πρόβλημα 1.2*). Έστω ότι έχουμε ένα κείμενο T για το οποίο έχουμε κατασκευάσει το δένδρο επιθεμάτων ST_T και αναζητούμε μέσα σε αυτό ένα πρότυπο P. Ξεκινώντας από την ρίζα του ST_T ακολουθείται το μονοπάτι που υπαγορεύεται από το P. Αν αυτή η διαδικασία είναι σε εσωτερικό κόμβο και μέχρι στιγμής έχουν ταιριάξει οι i πιο αριστεροί χαρακτήρες του P ακολουθείται η εξερχόμενη ακμή που ξεκινά με τον χαρακτήρα P_{i+1} ενώ αν είναι η διαδικασία πάνω σε ακμή ελέγχεται αν ο επόμενος χαρακτήρας της ακμής είναι ίδιος με τον P_{i+1}. Αν αυτή η κάθοδος από την ρίζα προς τα φύλλα εξαντληθεί έχοντας ταιριάζει επιτυχώς και τους $|P|$ χαρακτήρες του προτύπου, τότε με βάση την *ιδιότητα 2.4* τα επιθέματα που αντιστοιχούν στο υποδένδρο κάτω από το σημείο όπου τερμάτισε η διαδικασία του ταιριάσματος μοιράζονται το ίδιο κοινό πρόθεμα P. Συνεπώς το πρότυπο που αναζητείται εμφανίζεται στις θέσεις υποδεικνύονται από τα κλειδιά στο εν λόγω υποδένδρο. Αν η διαδικασία του ταιριάσματος από την ρίζα προς τα φύλλα τερματίσει πριν εξαντληθούν όλοι οι χαρακτήρες του προτύπου κανένα από τα επιθέματα της T δεν έχει ως πρόθεμα το P επομένως το πρότυπο δεν εμφανίζεται πουθενά μέσα στο κείμενο T. Εν κατακλείδι βλέπουμε ότι το με δένδρο επιθεμάτων το πρόβλημα ταιριάσματος προτύπων μπορεί να επιλυθεί αποδοτικά με την εκτέλεση το πολύ $|P|$ συγκρίσεις και άρα συνολικά σε χρόνο $O(|P| + \alpha)$, όπου α το μέγεθος της απάντησης και με την προϋπόθεση ότι σε κάθε εσωτερικό κόμβο η ακμή που θα ακολουθηθεί επιλέγεται σε σταθερό χρόνο. Έτσι σε περιβάλλοντα όπου συνεχώς υποβάλλονται συνεχώς νέες ερωτήσεις πάνω σε συγκεκριμένες συμβολοσειρές ενδείκνυται η δεικτοδότηση αυτών με δένδρα επιθεμάτων.

2.1.2 Αλγόριθμοι Κατασκευής

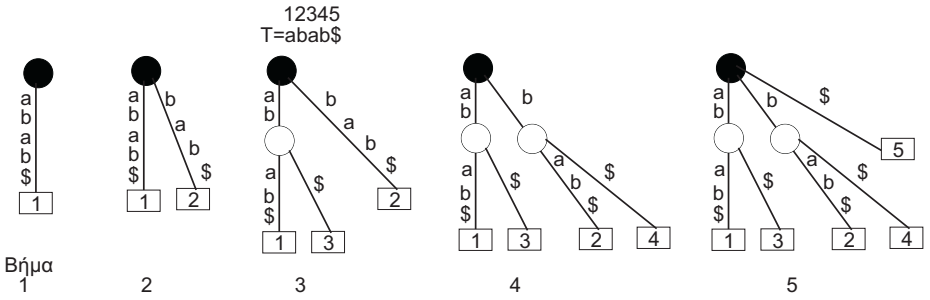
Υπάρχουν αρκετοί σειριακοί αλγόριθμοι για την κατασκευή ενός δένδρου επιθεμάτων σε γραμμικό χρόνο. Οι [Wei73, McC76, Ukk95] χρησιμοποιούν την παραδοχή ότι οι συμβολοσειρές έχουν σχηματισθεί από αλφάβητα σταθερού μεγέθους και κατά βάση στηρίζονται σε παρόμοιες κεντρικές ιδέες (για την συσχέτιση τους ανατρέξτε στο [GK97]) ενώ στην [Far97] δίνεται γραμμικός αλγόριθμός κατασκευής για μεγάλα αλφάβητα[2] (της τάξης $O(n)$). Τέλος στις [AIL+88, Har94, SV94] παρουσιάζονται παράλληλοι αλγόριθμοι για το μοντέλο ταυτόχρονης ανάγνωσης/ταυτόχρονης εγγραφής (CRCW PRAM), το οποίο δεν εξετάζεται όμως στην παρούσα εργασία.

Πριν δούμε τον αλγόριθμο του McCreight [McC76], θα παρουσιασθεί ο απλοϊκός τρόπος κατασκευής του δένδρου επιθεμάτων σε χρόνο $O(n^2)$, μιας και στηρίζεται πάνω σε αυτόν. Στη συνέχεια κάνοντας ορισμένες παρατηρήσεις πάνω στο δένδρο επιθεμάτων επιτυγχάνεται γραμμικός χρόνος κατασκευής.

[2] Παρατηρήστε ότι το μέγεθος του αλφάβητου είναι ίσο με τον μέγιστο αριθμό παιδιών που μπορεί να έχει ένας εσωτερικός κόμβος.

Ο απλοϊκός (naive) αλγόριθμός κατασκευής του δένδρου επιθεμάτων για μια συμβολοσειρά T ($|T| = n$) αρχικά θεωρεί ότι το δένδρο αποτελείται μόνο από την ρίζα και εισάγει σε αυτό διαδοχικά όλα τα επιθέματα $T_{i...n}$, το i να κινείται από το 1 μέχρι και το n. Αρχικά εισάγεται στο συμπαγές trie το επίθεμα $T_{1...n}$ (ολόκληρη η συμβολοσειρά δηλαδή) κάτω από την ρίζα εισάγοντας μία πλευρά και ένα φύλλο. Στο i-στο βήμα, όπου εισάγουμε το επίθεμα $T_{i...n}$, αρχίζουμε να κατεβαίνουμε από την ρίζα προς τα φύλλα ακολουθώντας το μοναδικό μονοπάτι που ορίζεται από τους χαρακτήρες του επιθέματος. Πιο συγκεκριμένα ξεκινώντας από την ρίζα ακολουθούμε την πλευρά που η ετικέτα της αρχίζει με τον χαρακτήρα T_i και διαδοχικά συγκρίνουμε τους χαρακτήρες του επιθέματος με τους χαρακτήρες των ετικετών των πλευρών που συναντάμε μέχρι κάποιος χαρακτήρας να μην ταιριάζει. Έστω ότι δεν ταίριασε ο χαρακτήρας $T_k, k > i$, σε αυτό το σημείο υπάρχουν δύο ενδεχόμενα: είτε βρισκόμαστε σε κάποιο κόμβο (ο οποίος θα έχει και ετικέτα μονοπατιού ίση με $T_{i...k-1}$) είτε στο μέσο της ετικέτας μιας πλευράς. Αν ισχύει η πρώτη περίπτωση εισάγουμε το υπόλοιπο του επιθέματος ($T_{k...n}$) τοποθετώντας κάτω από τον κόμβο που βρίσκεται η διεργασία μία πλευρά και ένα φύλλο. Σε αυτό το φύλλο θα αποθηκευτεί ο αριθμός i ο οποίος και αντιστοιχεί σ' αυτό το επίθεμα. Εάν ισχύει η δεύτερη περίπτωση σπάμε στην πλευρά αυτή δημιουργώντας ένα εσωτερικό κόμβο με ετικέτα μονοπατιού $T_{i...k-1}$ και εισάγουμε το υπόλοιπο του επιθέματος ($T_{k...n}$) δημιουργώντας μία πλευρά και ένα φύλλο κάτω από αυτόν τον νεοσύστατο κόμβο.

Παράδειγμα 2.5. Για την συμβολοσειρά abab$ τα βήματα του απλοϊκού αλγορίθμου φαίνονται στο σχήμα 2.2.



Σχήμα 2.2. Απλοϊκή Κατασκευή του Δένδρου Επιθεμάτων της "abab$"

Αν υποθέσουμε ότι το αλφάβητο έχει σταθερό μέγεθος και άρα ο αλγόριθμος σε κάθε εσωτερικό κόμβο επιλέγει σε σταθερό χρόνο την κατάλληλη εξερχόμενη ακμή τότε ο απλοϊκός αυτός αλγόριθμός κατασκευής του δένδρου επιθεμάτων εκτελεί στην χειρότερη περίπτωση $(n-1) + \cdots + 2 + 1 = O(n^2)$ συγκρίσεις. Σε περίπτωση που το αλφάβητο δεν θεωρείται σταθερό η χρονική πολυπλοκότητα ανέρχεται σε $O(|\Sigma|n^2)$.

Ο αλγόριθμος του McCreight μοιάζει σε μεγάλο βαθμό με τον απλοϊκό αλγόριθμο που δόθηκε παραπάνω. Εισάγει και αυτός στο δένδρο τα επιθέματα με φθίνουσα σειρά. Ο απλοϊκός αλγόριθμος σε κάθε βήμα ξεκινάει πάντα από

την ρίζα για να βρει ποιο τμήμα (για την ακρίβεια το μέγιστο πρόθεμα) του προς εισαγωγή επιθέματος υπάρχει ήδη μέσα στο δένδρο. Κατεβαίνει από την ρίζα όσο βρίσκει ταιριάσματα και στο σημείο που τερματίζει προσθέτει το υπόλοιπο τμήμα του επιθέματος. Εν αντιθέσει ο αλγόριθμος του McCreight διατηρεί κάποιους δείκτες μεταξύ των εσωτερικών κόμβων, οι οποίοι καλούνται suffix links, που του επιτρέπουν να μεταβαίνει στο σημείο αυτό πιο γρήγορα χωρίς να ξεκινάει πάντα από την ρίζα αλλά από την θέση στην οποία είχε ενθέσει το αμέσως προηγούμενο επίθεμα. Όπως θα δούμε παρακάτω με αυτή της διαφορά η κατασκευή του δένδρου επιθεμάτων ολοκληρώνεται σε γραμμικό χρόνο.

Για την περιγραφή του αλγορίθμου απαραίτητοι είναι οι ακόλουθοι ορισμοί:

Ορισμός 2.6. *Ετικέτα Μονοπατιού (path label) ενός εσωτερικού κόμβου u στο δένδρο επιθεμάτων είναι η συμβολοσειρά που προκύπτει από την συνένωση των υποσυμβολοσειρών των πλευρών που συναντώνται κινούμενοι από την ρίζα προς τον u. Συνεπώς κάθε κόμβος στο δένδρο επιθεμάτων διαθέτει μοναδική ετικέτα μονοπατιού. Η ετικέτα μονοπατιού ενός κόμβου u συμβολίζεται ως $L(u)$.*

Ορισμός 2.7. Θέση *(locus) μιας συμβολοσειράς x μέσα στο δένδρο επιθεμάτων είναι ο κόμβος (αν υπάρχει) που έχει ως ετικέτα μονοπατιού την x.*

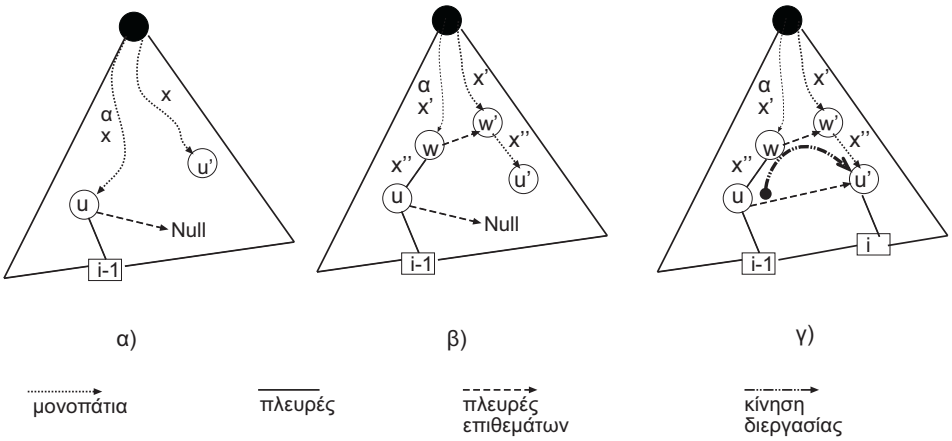
Ορισμός 2.8. Εγγυημένη Θέση *(contracted locus) μιας συμβολοσειράς x μέσα στο δένδρο επιθεμάτων είναι η θέση του μεγαλύτερου προθέματος της το οποίο ορίζεται στο δένδρο επιθεμάτων (αν ένα σε ένα σύνολο εσωτερικών κόμβων κάθε κόμβος έχει ως ετικέτα μονοπατιού ένα πρόθεμα της x τότε η εγγυημένη θέση είναι ο κόμβος που αντιστοιχεί στο μεγαλύτερο από αυτά προθέματα».*

Γυρνώντας πάλι πίσω στους αλγόριθμους κατασκευής διαπιστώνουμε ότι σε κάθε βήμα i όπου εισάγεται το επίθεμα $T_{i\ldots n}$ αναζητείται η εγγυημένη θέση του επιθέματος αυτού και στην συνέχεια προστίθεται στην δομή το υπόλοιπο του επιθέματος. ο απλοϊκός αλγόριθμός αναζητεί αυτή την εγγυημένη θέση ξεκινώντας πάντα από την ρίζα με αποτέλεσμα να έχει τετραγωνικό χρόνο εκτέλεσης. Έστω ότι κάθε εσωτερικός κόμβος u διατηρεί έναν επιπλέον δείκτη προς έναν άλλο εσωτερικό κόμβο. Ο δείκτης αυτός καλείται δείκτης επιθέματος (suffix link) και ορίζεται ως εξής:

Ορισμός 2.9. *Ο* δείκτης επιθέματος *(suffix link) κάθε εσωτερικού κόμβου με ετικέτα μονοπατιού ax με $|a| = 1, |x| \geq 1$ υποδεικνύει τον εσωτερικό κόμβο με ετικέτα μονοπατιού x.*

Ας δούμε πως αρχικοποιούνται οι δείκτες επιθέματος και πως χρησιμοποιούνται από τον αλγόριθμο του McCreight. Ας θεωρήσουμε μόλις έχει ολοκληρωθεί το $(i-1)$-στο βήμα όπου εισάγεται το επίθεμα $T_{i-1\ldots n}$ και ας υποθέσουμε ότι όλοι οι κόμβοι που έχουν δημιουργηθεί σε προηγούμενα βήματα ($< i-1$) έχουν αρχικοποιημένο τον δείκτη επιθέματος τους. Η διαδικασία βρίσκεται στην *εγγυημένη θέση* της $T_{i-1\ldots n}$ όπου έχει δημιουργήσει ένα φύλλο και μια πλευρά για την αναπαράσταση του υπόλοιπου του επιθέματος. Εκεί είτε έχει βρει κάποιο ήδη υπάρχον εσωτερικό κόμβο είτε έχει δημιουργήσει έναν καινούργιο. Ας υποθέσουμε ότι ισχύει το δεύτερο ενδεχόμενο και μόλις έχει δημιουργηθεί ένας

νέος κόμβος u, με ετικέτα μονοπατιού ax, $|a| = 1, |x| \geq 1$, του οποίου ο δείκτης επιθέματος δεν έχει αρχικοποιηθεί ακόμη (δείτε το σχήμα 2.3 α.). Από τον u κινούμαστε προς τον πατέρα του w. Έστω ότι η πλευρά που διασχίσαμε έχει σαν ετικέτα την υποσυμβολοσειρά x'' η οποία είναι επίθεμα της ax οπότε και ισχύει $ax = ax'x''$ (βλ. σχήμα 2.3 β.). Ο κόμβος w εκ υποθέσεως έχει αρχικοποιήσει τον δείκτη επιθέματος του σε προηγούμενο βήμα, οπότε και δείχνει τον κόμβο w'. Ακολουθώντας τον δείκτη αυτόν η διεργασία μεταφέρεται στον w' και κατεβαίνει πάνω στο μονοπάτι που επιτάσσει η υποσυμβολοσειρά q''. Έτσι καταλήγει στην εγγυημένη θέση του προς εισαγωγή επιθέματος σε αυτό το (i)-στο βήμα που είναι το $x = x'x''$. Αυτή η θέση είναι ο κόμβος u' ο οποίος υπάρχει ή μόλις δημιουργείται. Μόλις βρεθεί ο u' τότε αρχικοποιείται και ο δείκτης επιθέματος του u ώστε να τον δείχνει (βλ. σχήμα 2.3 β.). Αν ο u' δημιουργείται τότε το υπόλοιπο του (i)-στου επιθέματος προστίθεται κάτω από αυτόν με μία πλευρά και ένα φύλλο. Παρατηρήστε ότι μόλις δημιουργείται ένας νέος κόμβος αμέσως στο επόμενο βήμα αρχικοποιείται ο δείκτης επιθέματος τους οπότε έτσι εγγυάται ότι στην αρχή ενός βήματος υπάρχουν όλοι οι δείκτες επιθέματος πλην ενός. Αν ο u' υπάρχει τότε μπορεί να υπάρχει κάποιο μέρος του υπολοίπου του επιθέματος κάτω από αυτόν οπότε αυτή ελέγχεται κατεβαίνοντας από αυτό το σημείο χαρακτήρα προς χαρακτήρα. Στο σημείο που τερματίζει αυτή η κάθοδος τότε τότε δημιουργείται ένας νέος κόμβος και προστίθεται κάτω από αυτόν το υπόλοιπο του επιθέματος με μία πλευρά και ένα φύλλο. Για το επόμενο το $(i+1)$-στο ο u' ή ο κόμβος που μόλις έχει δημιουργηθεί χρησιμοποιείται ως ο u.



Σχήμα 2.3. i-στο βήμα του αλγ. McCreight

Η συνολική πολυπλοκότητα του i-στου βήματος του αλγόριθμου αποτελείται από δύο επιμέρους χρονικές πολυπλοκότητες:

1. τις πράξεις μέχρι να κατέλθει η διεργασία από τον κόμβο w' κατά την συμβολοσειρά x''. Επειδή γνωρίζουμε ότι σίγουρα υπάρχει αυτή η υποσυμβολοσειρά κάτω από τον w' δεν είναι αναγκαία η σύγκριση χαρακτήρα προς χαρακτήρα πάνω στις πλευρές που ακολουθούνται αλλά μόνο μία σύγκριση σε κάθε κόμβο για την επιλογή της κατάλληλης πλευράς και στην συνέχεια μετάβαση στον κόμβο που δείχνει παραλείποντας από την x'' τόσους

χαρακτήρες όσους διαθέτει η πλευρά. Έτσι το συνολικό κόστος θα είναι το πλήθος των κόμβων που συναντώνται, το οποίο ας το συμβολίσουμε με int_i για το βήμα i. Επίσης παρατηρήστε ότι το $int_i \leq |x''|$ επομένως για την υποσυμβολοσειρά x'' του επόμενου βήματος θα ισχύει $|x''_{i+1}| \leq |x''_i| - int_i$ ή αλλιώς $int_i \leq |x''_i| - |x''_{i+1}|$. Αθροίζοντας αυτό το κόστος και για τα n βήματα θα έχουμε $\sum_{i=1}^{n} int_i \leq \sum_{i=1}^{n} |x''_i| - |x''_{i+1}| \leq n - 1$

2. τις συγκρίσεις που εκτελούνται χαρακτήρα με χαρακτήρα αφού έχουμε βρει τον u για να εντοπισθεί η θέση που θα τοποθετηθεί το υπόλοιπο του επιθέματος. Στο $i-1$-στο βήμα ξεκινήσαμε από θέση με ετικέτα μονοπατιού $h_{i-1} = ax'x''$ και η διεργασία καταλήγει στην θέση με ετικέτα μονοπατιού $h_i = x'x''v$ τότε εκτελούνται $|h_i| - |h_{i-1}| + 1$ συγκρίσεις. Αθροίζοντας αυτό το κόστος και για τα n βήματα θα έχουμε $\sum_{i=1}^{n}(|h_i| - |h_{i-1}| + 1) = \sum_{i=1}^{n}(|h_i| - |h_{i-1}|) + n = |h_n| - |h_1| + n = n$

Με βάση τα παραπάνω προκύπτει ότι :

Θεώρημα 2.10. *Η κατασκευή του δένδρου επιθεμάτων μιας συμβολοσειράς T, η οποία σχηματίζεται από αλφάβητο Σ σταθερού μεγέθους και έχει μήκος n χαρακτήρες, κατασκευάζεται σε χρόνο $O(n)$ και καταλαμβάνει χώρο επίσης $O(n)$.*

Στην [Far97] δίνεται ένας γραμμικός αλγόριθμος κατασκευής του δένδρου επιθεμάτων ακόμη και όταν το αλφάβητο έχει μέγεθος της τάξης του μήκους της συμβολοσειράς, δηλαδή όταν $\Sigma = \{1, 2, 3, \cdots, n\}$. Ο αλγόριθμος του Farach, για μια συμβολοσειρά T, κατασκευάζει με αναδρομικό τρόπο ένα δένδρο επιθεμάτων ST_T^{odd} για τα περιττά επιθέματα $(T_{2i-1\cdots n}, i = 1 : \frac{n}{2})$ και ένα δένδρο επιθεμάτων (ST_T^{even}) για τα άρτια επιθέματα $(T_{2i\cdots n}, i = 1 : \frac{n}{2})$ σε γραμμικό χρόνο. Στην συνέχεια ενώνει κατάλληλα τα δύο αυτά δένδρα σε γραμμικό χρόνο κατασκευάζοντας το επιθυμητό ST_T.

Για την κατασκευή του ST_T^{odd} ακολουθείται η εξής διαδικασία. Αρχικά σχηματίζονται τα ζεύγη (T_{2i-1}, T_{2i}), για $i = 1 : \frac{n}{2}$ [3]. Τα ζεύγη αυτά διατάσσονται σε χρόνο $2\frac{n}{2}$ αρχικά με βάση το λιγότερο σημαντικό ψηφίο και μετά με βάση το σημαντικότερο (αλγόριθμος radix sort) και στη συνέχεια απομακρύνονται τα διπλότυπα ζευγάρια. Από αυτή την διατεταγμένη ακολουθία ζευγαριών σχηματίζεται η συμβολοσειρά T' σημειώνοντας σε κάθε θέση της $T'_i (i = 1 : \frac{n}{2})$ την θέση του ζευγαριού (T_{2i-1}, T_{2i}) μέσα στη διατεταγμένη ακολουθία. Έτσι η αρχική συμβολοσειρά T κωδικοποιείται σε μία T' με υποδιπλάσιο μήκος. Επίσης μεταξύ των δύο αυτών συμβολοσειρών υπάρχουν οι εξής ιδιότητες που επιτρέπουν την αναδρομική κατασκευή του ST_T^{odd}:

Ιδιότητα 2.11. Ένα περιττό επίθεμα $T_{2i-1\cdots n}$ στην T αντιστοιχεί στο επίθεμα $T'_{i\cdots\frac{n}{2}}$ της T'. Έτσι κάθε φύλλο στο δένδρο επιθεμάτων $ST_{T'}$ της T' με ετικέτα i γίνεται φύλλο με ετικέτα $2i-1$ στο περιττό δένδρο επιθεμάτων ST_T^{odd} της T. Επίσης κάθε εσωτερικός κόμβος του $ST_{T'}$ με ετικέτα μονοπατιού μήκους i γίνεται εσωτερικός κόμβος με ετικέτα μονοπατιού μήκους $2i$ στο ST_T^{odd}. Συνεπώς παρατηρείται ότι τα δύο αυτά δένδρα διαφέρουν μόνο στις ταμπέλες τους.

[3] Παρατηρείστε ότι τα ζευγάρια αυτά αντιμετωπίζονται ως διψήφιοι αριθμοί d_2d_1 με $d_2 = T_{2i-1}$ και $d_1 = T_{2i}$.

Ιδιότητα 2.12. Ενώ το $ST_{T'}$ περιέχει όλα τα περιττά επιθέματα της T μπορεί να μην έχει συμπαγή αναπαράσταση και αυτό επειδή ένας χαρακτήρας στην T' κωδικοποιεί δύο χαρακτήρες της T. Έτσι μπορεί να υπάρχει το εξής ενδεχόμενο: από έναν εσωτερικό κόμβο του $ST_{T'}$ να υπάρχουν εξερχόμενες πλευρές που ξεκινούν με διαφορετικό χαρακτήρα της T' που όμως κωδικοποιούν ζεύγος χαρακτήρων της T που έχει ίδιο το σημαντικότερο ψηφίο του . Για παράδειγμα οι δύο αυτοί διαφορετικοί χαρακτήρες μπορεί να κωδικοποιούν τα ζεύγη (α,β), (α,γ). Για όλες αυτές τις περιπτώσεις πρέπει μια διεργασία να διατρέξει όλους τους εσωτερικούς κόμβους του $ST_{T'}$ και να διορθώσει τις πλευρές που εξέρχονται με χαρακτήρες που αντιστοιχούν σε ζεύγη που έχουν το σημαντικότερο ψηφίο ίδιο, προσθέτοντας έναν εσωτερικό κόμβο που αναπαριστά αυτό το κοινό πρόθεμα των πλευρών μονάχα μια φορά.

Για να σχηματίσουμε το ST_T^{odd} από το $ST_{T'}$ χρειάζεται καταρχάς χρόνος $O(n)$ για την κωδικοποίηση της συμβολοσειράς και εν συνεχεία χρόνος $O(\frac{n}{2})$ για να μετασχηματισθεί το $ST_{T'}$ στο ST_T^{odd}. Έτσι αν εφαρμοσθεί αναδρομικά η παραπάνω διεργασία στην T' και στο $ST_{T'}$ προκύπτει ότι τελικά ο χρόνος κατασκευής C_n του ST_T^{odd} της συμβολοσειράς T δίνεται από την αναδρομική σχέση $C_n = C_{\frac{n}{2}} + O(n)$.

Λήμμα 2.13. *Ο χρόνος κατασκευής του ST_T^{odd} της συμβολοσειράς T είναι γραμμικός.*

Απόδειξη. Θα αποδείξουμε με επαγωγή ότι ισχύει $C_n = C_{\frac{n}{2}} + cn = c(n + (n - 1))$.

→ Για $k = 1$ έχουμε $C_1 = c$
→ Για $k = \frac{n}{2}$ έστω ότι ισχύει, οπότε έχουμε $C_{\frac{n}{2}} = c(\frac{n}{2} + \frac{n}{2} - 1) = c(n - 1)$
→ Για $k = n$ έχουμε $C_n = C_{\frac{n}{2}} + cn = c(n - 1) + cn = c(n + (n - 1))$ οπότε και ισχύει

Στη συνέχεια από το δένδρο περιττών επιθεμάτων ST_T^{odd} κατασκευάζεται το δένδρο των άρτιων επιθεμάτων ST_T^{even}. Ένα άρτιο επίθεμα $T_{2i\cdots n}, i = 1 : \frac{n}{2}$ στην ουσία αποτελείται από τον χαρακτήρα T_{2i} ακολουθούμενο από το περιττό επίθεμα $T_{2i+1\cdots n}$. Με χρήση του ST_T^{odd} τα άρτια επιθέματα μπορούν σε γραμμικό χρόνο να διαταχθούν λεξικογραφικά χρησιμοποιώντας το σχήμα radix sort. Τα περιττά επιθέματα εκ κατασκευής του ST_T^{odd} βρίσκονται λεξικογραφικά διατεταγμένα από τα αριστερότερο φύλλο και προς το δεξί. Έτσι σε χρόνο $O(n)$ είναι εύκολο να ληφθεί η λεξικογραφική σειρά των άρτιων επιθεμάτων χρησιμοποιώντας μονάχα τον πρώτο τους χαρακτήρα ως το πιο σημαντικό ψηφίο διάταξης για το radix sort σχήμα. Επίσης σε γραμμικό χρόνο μπορούν να υπολογιστούν με την βοήθεια πάλι του ST_T^{odd} τα μέγιστα κοινά προθέματα για κάθε ζεύγος άρτιων επιθεμάτων τα οποία βρίσκονται σε γειτονικές θέσεις στην λεξικογραφική τους διάταξη. Με γραμμική προεπεξεργασία του ST_T^{odd} μπορεί να υπολογιστεί σε σταθερό χρόνο ο βαθύτερος κοινός πρόγονος δύο οποιοδήποτε φύλλων [HT84] και άρα το μέγιστο κοινό πρόθεμα οποιονδήποτε περιττών επιθεμάτων. Με βάση αυτήν την πληροφορία μπορεί να υπολογιστεί και το μέγιστο κοινό πρόθεμα κάθε ζεύγους άρτιων επιθεμάτων τα οποία βρίσκονται σε γειτονικές θέσεις στην λεξικογραφική τους διάταξη. Αυτό το μέγιστο

κοινό πρόθεμα υπολογίζεται με βάση τον πρώτο χαρακτήρα των επιθεμάτων. Αν οι πρώτοι χαρακτήρες είναι διαφορετικοί τότε το μέγιστο κοινό πρόθεμα είναι μηδενικό. Αν όμως είναι ίδιοι τότε το μέγιστο κοινό πρόθεμα ισούται με τον πρώτο χαρακτήρα συν το μέγιστο κοινό πρόθεμα των περιττών επιθεμάτων που ξεκινούν από την αμέσως δεξιότερη θέση.

Χρησιμοποιώντας την λεξικογραφική διάταξη των άρτιων επιθεμάτων και την πληροφορία των μέγιστων κοινών προθεμάτων κάθε γειτονικού ζεύγους το ST_T^{even} μπορεί να κατασκευασθεί σε γραμμικό χρόνο. Ξεκινώντας από το επίθεμα με την μικρότερη λεξικογραφικά θέση προσθέτονται τα επιθέματα ένα προς ένα κινούμενοι σε μεγαλύτερες θέσεις. Στην γενική περίπτωση για μια άρτια θέση i θα θέλουμε να υπολογίσουμε το $lcp(T_i, T_{i+2}$. Έχοντας προϋπολογίσει τα $lcp(T_{i+1}, T_{i+3}$ η ζητούμενη μέγιστη προέκταση υπολογίζεται από την σχέση:

$$lcp(T_i, T_{i+2}) = \begin{cases} 0 & , T_i \neq T_{i+2} \\ 1 + lcp(T_{i+1}, T_{i+3}) & , T_i = T_{i+2} \end{cases}$$

Στη συνέχεια έχοντας τα μέγιστα κοινά προθέματα για δύο διαδοχικά ζυγά επιθέματα μπορεί να κατασκευαστεί το ST_T^{even} σε γραμμικό χρόνο χρησιμοποιώντας μια στοίβα. Η διαδικασία αυτή περιγράφεται στην παράγραφο 2.2.3.

Με βάση τα παραπάνω προκύπτει το ακόλουθο λήμμα:

Λήμμα 2.14. *Ο χρόνος κατασκευής του ST_T^{even} της συμβολοσειράς T είναι γραμμικός.*

Στη συνέχεια τα ST_T^{odd} καιST_T^{even} συγχωνεύονται κατάλληλα ώστε προκύψει το επιθυμητό ST_T. Ξεκινώντας από τις ρίζες των δύο δένδρων ξεκινάει μια ταυτόχρονη διάβαση κατά βάθος. Η διεργασία αρχικά βρίσκεται στις ρίζες των δύο δένδρων και ακολουθεί ταυτόχρονα τις πλευρές κάτω από την ρίζα που αντιστοιχούν στο χαμηλότερο λεξικογραφικά γράμμα του αλφάβητου. Αν ένα από τα δύο δένδρα δεν διαθέτουν μια τέτοια πλευρά τότε αυτή μαζί με τον κόμβο που δείχνει και ολόκληρο το υποδένδρο του εισάγεται στο ST_T. Αν και τα δύο δένδρα διαθέτουν όμως μια τέτοια πλευρά οι δύο πλευρές συγχωνεύονται σε μία. Όταν οι δύο πλευρές δεν έχουν το ίδιο μήκος η μεγαλύτερη από αυτές σπάει σε δύο μέρη έτσι ώστε το πρώτο μέρος να έχει μήκος ίσο με της άλλης πλευράς. Στην περίπτωση της συγχώνευσης πλευρών στη συνέχεια τοποθετούνται τα δύο υποδένδρα. Με αυτή την ενέργεια υπάρχει το ενδεχόμενο να συγχωνευθούν δύο πλευρές που παρόλο που ξεκινούν με τον ίδιο χαρακτήρα οι υπόλοιποι ή ένα μέρος αυτών είναι διαφορετικό. Κατά την συγχώνευση των πλευρών ένας έλεγχος χαρακτήρα προς χαρακτήρα δεν είναι δυνατός μιας και οδηγεί σε συνολική χρονική πολυπλοκότητα $O(n^2)$. Η παραπάνω διεργασία διατρέχει κατά βάθος και τα δύο δένδρα σε χρόνο $O(n)$. Οι πλευρές που λανθασμένα έχουν συγχωνευθεί είναι δυνατόν να διορθωθούν παραμένοντας γραμμική η συνολική πολυπλοκότητα.

Με τον τερματισμό της παραπάνω διεργασίας έχει σχηματισθεί το ST_T στο οποίο αναγκαίο είναι να γίνουν ορισμένες διορθώσεις. Κάθε κόμβος του ST_T μπορεί να είναι είτε άρτιος είτε περιττός αναλόγως από το αν προέρχεται από το ST_T^{even} ή το ST_T^{odd} ή και τα δύο (όπως π.χ. η ρίζα). Έστω ότι εξετάζουμε

έναν εσωτερικό κόμβο u ο οποίος είναι ο βαθύτερος κοινός πρόγονος ενός περιττού $T_{2j-1\cdots n}$ και ενός άρτιου $T_{2i\cdots n}$ επιθέματος. Παρατηρήστε ότι πάντα τέτοια ζεύγη θα βρίσκονται σε υποδένδρα κόμβων, όπως ο u, που η εισερχόμενη σε αυτά πλευρά έχει συγχωνευθεί. Έτσι εξετάζοντας αν η ετικέτα μονοπατιού $L(u)$ του u είναι ίση με το μεγαλύτερο κοινό πρόθεμα των δύο επιθεμάτων ($L'(u) = LCP(T_{2j-1\cdots n}, T_{2i\cdots n})$ που είναι και η σωστή ετικέτα μονοπατιού στο τελικό δένδρο) μπορούμε να αποφανθούμε αν όντως η συγχώνευση της πλευράς που οδηγεί στον u είναι σωστή. Σε αντίθετη περίπτωση που το το $L'(u)$ είναι μικρότερο από την ταμπέλα μονοπατιού σημαίνει ότι έχουμε ενώσει πάνω από τον u δύο πλευρές που δεν είναι ίδιες αλλά μοιράζονται ένα κοινό πρόθεμα.

Το $L'(u) = LCP(T_{2j-1\cdots n}, T_{2i\cdots n})$ δεν μπορεί να υπολογισθεί άμεσα αφού τα επιθέματα αντιστοιχούν σε φύλλα που βρίσκονται σε ξεχωριστά δένδρα (στο άρτιο και στο περιττό) οπότε και δεν μπορούν να εφαρμοστούν ερωτήματα βαθύτερου κοινού προγόνου σε σταθερό χρόνο. Το $LCP(T_{2j-1\cdots n}, T_{2i\cdots n})$ ισούται με $1 + LCP(T_{2j\cdots n}, T_{2i+1\cdots n})$ αν $T_{2i} = T_{2j-1}$ αλλιώς ισούται με 0. Αν ο κόμβος x είναι ο βαθύτερος κοινός πρόγονος των φύλλων $T_{2j\cdots n}$ και $T_{2i+1\cdots n}$ στο ST_T δείχνεται ότι ισχύει $L'(u) = 1 + L'(x)$ και ότι αν ορισθεί η συνάρτηση $d(u) = x$, για κάθε κόμβο u που είναι συγχωνευμένος, σχηματίζεται ένα δένδρο πάνω στους κόμβους του ST_T. Όπως είπαμε και προηγουμένως $L'(u) = LCP(T_{2j-1\cdots n}, T_{2i\cdots n}) = 1 + LCP(T_{2j\cdots n}, T_{2i+1\cdots n}) = 1 + L'(x)$ όπου $x = LCA(T_{2j\cdots n}, T_{2i+1\cdots n})$ και το $L'(u)$ αντιστοιχεί στο βάθος του u σε αυτό αφού $L'(\text{ρίζας}) = 0$. Έτσι με μια διαπέραση αυτού του δένδρου κατά βάθος μπορούν σε γραμμικό χρόνο να υπολογιστούν τα $L'(u)$ για κάθε u. Έτσι κατά συνέπεια όλοι οι κόμβοι που βρίσκονται κάτω από συγχωνευμένες πλευρές μπορούν να ελεγχθούν και να διορθωθούν αν αυτό χρειάζεται σπάζοντας την πλευρά και προσθέτοντας έναν κόμβο ως πρόγονο.

Και τα τρία βασικά μέρη του αλγορίθμου του Farach εκτελούνται σε γραμμικό χρόνο συνεπως προκύπτει το ακόλουθο θεώρημα:

Θεώρημα 2.15. *Το δένδρο επιθεμάτων μιας συμβολοσειράς T σχηματισμένης από το αλφάβητο $\Sigma = \{1, 2, \cdots, |T|\}$ κατασκευάζεται σε $O(|T|)$ χώρο και χρόνο.*

2.1.3 Υλοποίηση

Όπως είδαμε και προηγουμένως για συμβολοσειρά T, $|T| = n$ χαρακτήρων, το δένδρο επιθεμάτων της ST_T θα έχει το πολύ $2n - 1$ κόμβους και $2n - 2$ πλευρές. Όπως διαφάνηκε και στις προηγούμενες ενότητες ο τρόπος υλοποίησης των δομικών στοιχείων του δένδρου επηρεάζει τον χώρο που καταλαμβάνει η δομή δεικτοδότησης αλλά και τον χρόνο απάντησης των ερωτημάτων. Πιο συγκεκριμένα πυρήνας αυτού του θέματος είναι το πως επιλέγουμε μια πλευρά, με βάση των πρώτο χαρακτήρα της ταμπέλας της, όταν βρισκόμαστε σε ένα κόμβο και θέλουμε να εξέλθουμε από αυτόν με βάση έναν συγκεκριμένο χαρακτήρα. Οι δύο ακραίες υλοποιήσεις είναι να φυλάμε όλες τις πλευρές ενός κόμβου σε μια γραμμική λίστα ενώ η άλλη είναι να γίνει χρήση ενός πίνακα μεγέθους $|\Sigma|$ από δείκτες. Στην πρώτη περίπτωση για να εξέλθουμε από έναν κόμβο σαρώνεται η γραμμική λίστα για να βρεθεί η κατάλληλη πλευρά σε χρόνο $O(|\Sigma|)$ ενώ στην

δεύτερη σε $O(1)$ χρησιμοποιώντας περισσότερη μνήμη αφού από κάθε κόμβο μπορεί να μην υπάρχουν πλευρές για κάθε γράμμα του αλφαβήτου. Συνεπώς το πρόβλημα ταιριάσματος στην πρώτη περίπτωση θα λύνεται σε $O(|\Sigma||P|)$ ενώ στην δεύτερη σε $O(|P|)$. Μεταξύ αυτών των δύο ακραίων αντιμετωπίσεων υπάρχουν και άλλες λύσεις όπως η χρήση δένδρων εύρεσης ή σχημάτων κατακερματισμού.

Ας δούμε σε πρώτη φάση πόσο μπορεί να εκταθεί το δένδρο επιθεμάτων συγκριτικά με τη συμβολοσειρά που δεικτοδοτεί και πως με την χρήση κάποιων παρατηρήσεων μπορεί ο χώρος που καταλαμβάνεται να περιορισθεί.

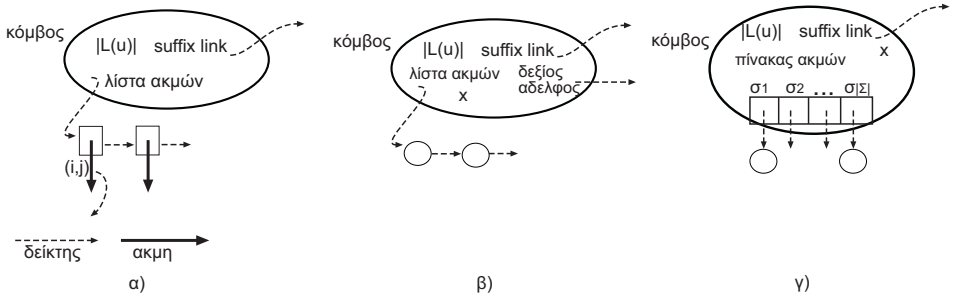
Έστω ότι σε κάθε κόμβο u οι ακμές φυλάσσονται σε μια γραμμική λίστα, τότε ένας κόμβος και μια πλευρά θα αποτελούνται από (βλ. σχήμα 2.4α.):

Κόμβος:

→ έναν ακέραιο ο οποίος φυλάει το μήκος της ετικέτας μονοπατιού $|L(u)|$.
→ την γραμμική λίστα πλευρών, δηλαδή από έναν δείκτη στην πρώτη πλευρά
→ το δείκτη επιθέματος suffix link

Ακμή:

→ δύο ακέραιους (i, j) για αναπαράσταση της ταμπέλας της πλευράς.
→ έναν δείκτη στην επόμενη πλευρά της γραμμικής λίστας των πλευρών
→ ένας δείκτης προς τον κόμβο τον οποίο δείχνει



Σχήμα 2.4. Κόμβοι και Ακμές του Δένδρου Επιθεμάτων

Κάθε κόμβος αποτελείται από 3 στοιχεία ενώ κάθε ακμή από 4 συνεπώς το δένδρο επιθεμάτων θα καταλαμβάνει χώρο $3(2n-1)+4(2n-2) = 14n-11$ στοιχείων [4].

Παρατήρηση 2.16. Τα n φύλλα του δένδρου επιθεμάτων δεν είναι αναγκαίο να αποτελούνται από 3 στοιχεία παρά μόνο να αποθηκεύουν τη θέση έναρξης του αντίστοιχου επιθέματος. Συνεπώς το ST_T θα καταλαμβάνει χώρο $n + 3(n-1) + 4(2n-2) = 12n-11$ στοιχείων.

[4] Στα συνήθη συστήματα κάθε τέτοιο στοιχείο καταλαμβάνει μια λέξη της μηχανής, δηλαδή 4bytes. Συνεπώς χρειάζονται περίπου 54 φορές περισσότερα bytes για να δεικτοδοτήσουμε μια συμβολοσειρά από n χαρακτήρες -bytes

Παρατήρηση 2.17. Η αναπαράσταση της ετικέτας μιας πλευράς μπορεί να γίνει έμμεσα, χωρίς την αποθήκευση του ζεύγους (i, j) ως εξής: αν σε κάθε κόμβο αποθηκευθεί η θέση εμφάνισης x της ετικέτας μονοπατιού του κόμβου u (παρατηρήστε υπάρχουν τουλάχιστον δύο εμφανίσεις, κατά τον αλγόριθμο του McCreight αποθηκεύεται το επίθεμα που εισάγεται την στιγμή που δημιουργείται ο κόμβος) μπορεί η ετικέτα της εισερχόμενης στον u πλευράς να βρεθεί από τη διαφορά $x - |L(\pi(u))|$, όπου $\pi(u)$ ο πατέρας του u. Έτσι σε κάθε κόμβο αποθηκεύεται ένα ακόμη στοιχείο, άρα ο κόμβος αποτελείται από 4 στοιχεία, ενώ η πλευρά θα αποτελείται από δύο. Συνεπώς το ST_T μαζί και με την προηγούμενη παρατήρηση θα καταλαμβάνει χώρο $n+4(n-1)+2(2n-2) = 9n-8$ στοιχείων.

Παρατήρηση 2.18. Η γραμμική λίστα των πλευρών μπορεί να υλοποιηθεί έμμεσα ως εξής : κάθε κόμβος έχει ένα δείκτη στο αριστερότερο παιδί του (αρχή της λίστας) και επίσης έχει ένα δείκτη στο αμέσως δεξιότερο αδελφό του. Με αυτό τον τρόπο σε κάθε κόμβο αποθηκεύεται ένα ακόμη στοιχείο, άρα ο κόμβος αποτελείται από 5 στοιχεία, ενώ οι πλευρές αναπαριστώνται έμμεσα με δείκτες προς κόμβους (βλ. σχήμα 2.4β.). Συνεπώς το ST_T μαζί και με τις δύο προηγούμενες παρατηρήσεις θα καταλαμβάνει χώρο $n + 5(n - 1) = 6n - 5$ στοιχείων.

Σε περίπτωση που σε κάθε κόμβο οι πλευρές αναπαρασταθούν ως ένας πίνακας (βλ. σχήμα 2.4γ.) επιτυγχάνουμε επιλογή της πλευράς με μόνο μία σύγκριση αλλά χρησιμοποιείται πολύ περισσότερος χώρος. Το ST_T σε αυτή την περίπτωση θα καταλαμβάνει χώρο $n + (|\Sigma| + 4)(n - 1) = (|\Sigma| + 5)n - (|\Sigma| + 3)$ στοιχείων. Για το αλφάβητο των ακολουθιών DNA αποθηκεύονται $(5 + 5)n - (5 + 3) = 10n - 8$ ενώ για το λατινικό $30n - 28$. Παρατηρούμε ότι αυτή η δεύτερη λύση, όταν απαιτούνται μικροί χρόνοι απάντησης (π.χ. on-line συστήματα), μπορεί να εφαρμοσθεί σε περιπτώσεις που υπάρχει μεγάλος αποθηκευτικός χώρος σε συνδυασμό με μικρά σχετικά αλφάβητα ή με συμβολοσειρές σχετικά μικρού μήκους.

Μεταξύ αυτών των δύο ακραίων λύσεων θα μπορούσαν να χρησιμοποιηθούν και άλλοι τρόποι οργάνωσης που να δίνουν μια διαφορετική αναλογία στην συναλλαγή χώρου-χρόνου απάντησης. Παραδείγματος χάρη οι πλευρές σε κάθε κόμβο μπορούν να οργανωθούν σε ένα δένδρο εύρεσης με αποτέλεσμα το πρόβλημα του ταιριάσματος να απαντάται σε χρόνο $O(\log|\Sigma||P|)$ αλλά χωρίς να διαφαίνεται καθαρά υπάρχει μεγάλο κέρδος σε χώρο ειδικά όταν οι κόμβοι αρχίζουν να είναι μισογεμάτοι (δηλαδή $\approx \frac{|\Sigma|}{2}$ παιδιά) τότε ο χώρος είναι ίδιος με την λύση χρησιμοποίησης του πίνακα.

Μια πιο αποδοτική προσέγγιση, κυρίως στη μέση περίπτωση, είναι να οργανωθούν οι ακμές σε πίνακα κατακερματισμού. Όπως αναφέρει και ο McCreight [McC76] προτείνει την χρήση του σχήματος κατακερματισμού του Lampson [Knu98] το οποία ανήκει στην οικογένεια του κατακερματισμού με αλυσίδες και υλοποιεί την συνάρτηση $f(u, char) = u'$ όπου u, u' εσωτερικοί κόμβοι. Στην [Kur99] προτείνουν ένα σχήμα ανοιχτού κατακερματισμού χρησιμοποιώντας ένα πίνακα μεγέθους $2n-2$, όσες και οι πλευρές δηλαδή, έτσι συνολικά το ST_T καταλαμβάνει χώρο $n + 3(n - 1) + 2n - 2 = 6n - 5$ στοιχείων.

Επίσης στην [Kur99] προτείνουν επίσης άλλη μια σχεδιαστική προσέγγιση η οποία πειραματικά έχει πολύ καλά αποτελέσματα. Προτείνουν ένα σχήμα στο οποίο γίνεται χρήση δύο τύπων εσωτερικών κόμβων ενός μεγάλου και ενός μικρού. Όπως είπαμε και προηγουμένως στην παρατήρηση 2.17 κάθε εσωτερικός κόμβος με ετικέτα μονοπατιού $L(u)$ αποθηκεύει την δεύτερη από αριστερά εμφάνιση της μέσα στην συμβολοσειρά T, ας την καλέσουμε $head(u)$, μιας και κατά τη διάρκεια αυτής διακλαδίζεται μια πλευρά και δημιουργείται ο νέος κόμβος. Δεξιότερες εμφανίσεις του $L(u)$ θα διακλαδιστούν πάνω στον ήδη υπάρχοντα κόμβο. Αναφέρθηκε ότι σε κάθε βήμα ο αλγόριθμος του McCreight δημιουργεί το πολύ ένα κόμβο συνεπώς κάθε εσωτερικός κόμβος θα έχει διαφορετικό $head$. Αυτό οδηγεί στην εξής σχέση, για έναν κόμβος με ετικέτα μονοπατιού $ax, |a| = 1$, και τον κόμβο με ετικέτα μονοπατιού x θα ισχύει $head(ax) \geq head(x) - 1$. Αυτή η σχέση στην ουσία σημαίνει ότι ή ο κόμβος με την ετικέτα μονοπατιού x έχει σχηματισθεί στο επόμενο βήμα από αυτό που σχηματίστηκε ο ax, οπότε και ισχύει η ισότητα η το ax βρίσκεται δεξιότερα από την $head(x)$. Με βάση την παραπάνω ανισότητα οι κόμβοι μπορούν να χωρισθούν σε αλυσίδες ως εξής: μια αλυσίδα απαρτίζεται από μηδέν ή περισσότερους κόμβους για τους οποίους ισχύει η ισότητα (άρα σχηματίστηκαν σε διαδοχικά βήματα) που ακολουθούνται από έναν κόμβο για το οποίο ισχύει μόνο η ανισότητα με τον προτελευταίο κόμβο.

Έστω ότι έχουμε μια τέτοια αλυσίδα κόμβων $n_1, n_2, \cdots n_i \cdots n_k$. Τότε για κάθε στοιχείο $n_i, i < k$ ισχύει:

→ $L(n_i) = L(n_k) + (k - i)$ αφού όλα τα στοιχεία δημιουργήθηκαν σε διαδοχικά βήματα θα έχουν συνεχώς μειούμενη κατά ένα ετικέτα μονοπατιού
→ $head(n_i) = head(n_k) + (k - i)$
→ Ο δείκτης επιθέματος του n_i δείχνει τον n_{i+1}.

Με βάση τα παραπάνω σε κάθε τέτοια αλυσίδα οι κόμβοι $n_i, i < k$ προτείνεται να υλοποιηθούν με μικρούς κόμβους που δεν περιέχουν τις παραπάνω τιμές οι οποίες μπορούν σε σταθερό χρόνο να βρεθούν από το τελευταίο στοιχείο της αλυσίδας το οποίο τις σημειώνει όλες σε έναν μεγαλύτερο κόμβο. Με αυτό τον τρόπο είναι δυνατόν να εξοικονομηθεί αποθηκευτικός χώρος. Με βάση αυτές τις παρατηρήσεις στην [Kur99] το σχήμα με τις λίστες (βλ. παρατήρηση 2.18) καταλαμβάνει χώρο $n + 3s + 5l$, όπου s, l ο αριθμός των μικρών και μεγάλων κόμβων αντίστοιχα. Επίσης δίνουν και ένα σχήμα κατακερματισμού με χώρο $4n + 3l$ όπου l ο αριθμός των μεγάλων κόμβων. Το μόνο αρνητικό στοιχείο των αποτελεσμάτων αυτών είναι ότι δεν υπάρχει ένα θεωρητικό άνω όριο αλλά ισχύουν για $n \leq 2^{27}$.

Εν κατακλείδι θα μπορούσαμε να πούμε ότι όταν το δένδρο επιθεμάτων απαντάει ερωτήματα όπως το ταίριασμα ενός προτύπου, όπου υπάρχει κίνηση από την ρίζα προς τα φύλλα, προτιμότερη είναι μία υλοποίηση τύπου κατακερματισμού, ενώ σε εφαρμογές εύρεσης επαναλήψεων ή αλληλοσυσχέτισης συμβολοσειρών, όπου όπως θα δούμε και στην επόμενη ενότητα διατρέχονται όλοι οι εσωτερικοί κόμβοι και σε καθένα από αυτούς εξετάζονται όλα τα παιδιά του, η υλοποίηση τύπου λίστας είναι προτιμότερη.

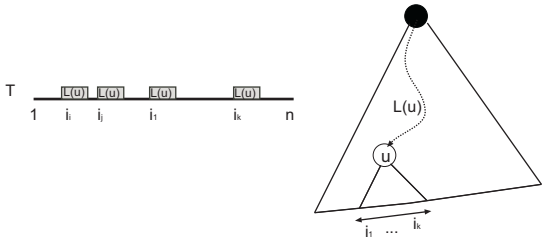
Στην [CM96] δίνεται η καλύτερη δυνατή υλοποίηση του δένδρου επιθεμάτων όσον αφορά το χώρο που καταλαμβάνει. Η υλοποίηση αυτή απαιτεί $4n$ στοιχεία

αλλά δεν μπορεί να κατασκευαστεί άμεσα από κάποιον γραμμικό αλγόριθμο αφού δεν διαθέτει τους δείκτες επιθεμάτων (suffix links). Αυτή την έμμεση υλοποίηση θα την δούμε συνοπτικά στην Ενότητα 3.4 όπου χρησιμοποιείται για την επέκταση του δένδρου επιθεμάτων στην δευτερεύουσα μνήμη.

2.1.4 Εφαρμογές

Το δένδρο επιθεμάτων μπορεί να απαντήσει σε ένα μεγάλο αριθμό ερωτημάτων με βέλτιστο τρόπο πέρα από το ταίριασμα προτύπου. Η βασική ιδιότητα του δένδρου επιθεμάτων που χρησιμοποιείται είναι το γεγονός η ετικέτα μονοπατιού $L(u)$ ενός εσωτερικού κόμβου εμφανίζεται στις θέσεις που υποδεικνύουν τα φύλλα στο υποδένδρο του (βλ. σχήμα 2.5). Μερικές χαρακτηριστικές εφαρμογές του δένδρου επιθεμάτων είναι οι εξής:

→ **Μέγιστη Επαναλαμβανόμενη Υποσυμβολοσειρά (Longest Repeated Substring):** Ψάχνεται να βρεθεί η μέγιστη υπό-συμβολοσειρά u μιας συμβολοσειράς T, $|T| = n$, που εμπεριέχεται παραπάνω από μία φορά. Αφού έχουμε κατασκευάσει το δένδρο επιθεμάτων ST_T σε χρόνο $O(n)$ τότε μπορεί το παραπάνω ερώτημα να απαντηθεί χρόνο ακόμη $O(n)$. Διατρέχονται όλοι οι κόμβοι του δένδρου επιθεμάτων (το πλήθος τους είναι $O(n)$) και υπολογ΄-ιζοντας για κάθε κόμβο u το πλήθος των χαρακτήρων της συμβολοσειράς από την ρίζα μέχρι και αυτόν (δηλαδή βρίσκεται ο u με το μέγιστο $|L(u)|$) και επιλέγεται ο *΄βαθύτερος΄* εσωτερικός κόμβος. Βαθύτερος με την έννοια ότι το μονοπάτι από την ρίζα μέχρι και αυτόν έχει το μεγαλύτερο πλήθος χαρακτήρων. Τότε οι θέσεις που υποδεικνύονται στα φύλλα του υποδένδρου κάτω από αυτόν είναι οι θέσεις που ξεκινάει η μέγιστη επαναλαμβανόμενη υποσυμβολοσειρά u (βλ. Σχήμα 2.5).



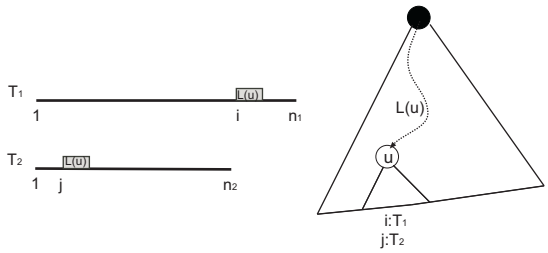
Σχήμα 2.5. Εύρεση (Μέγιστων) Επαναλαμβανόμενων Υποσυμβολοσειρών

→ **Μέγιστη Κοινή Υποσυμβολοσειρά (Longest Common Substring):** Ψάχνεται να βρεθεί η μέγιστη κοινή υπό-συμβολοσειρά δύο συμβολοσειρών T_1 και T_2. Χτίζουμε ένα γενικευμένο δένδρο επιθεμάτων πάνω στα T_1 και T_2.

Ορισμός 2.19. *Ένα γενικευμένο δένδρο επιθεμάτων $gST(\{T_1, T_2, \cdots T_k\})$ ενός συνόλου συμβολοσειρών $\{T_1, T_2, \cdots T_k\}$ είναι το συμπαγές trie που περιέχει για κλειδιά όλα τα επιθέματα των συμβολοσειρών αυτών. Για την κατασκευή ενός γενικευμένου δένδρου επιθεμάτων μπορούν να χρησιμοποιηθούν οι γνωστοί αλγόριθμοι κατασκευής των δένδρων επιθεμάτων που υπερθέτοντας τα επιθέματα των συμβολοσειρών στην ίδια δομή. Ολοκληρώνοντας*

την εισαγωγή των επιθεμάτων για μια συμβολοσειρά η διαδικασία συνεχίζει για την επόμενη από την ρίζα. Ένα γενικευμένο δένδρο επιθεμάτων gST καταλαμβάνει χώρο $O(|T_1| + |T_2| + \cdots + |T_k|)$ και ολοκληρώνεται σε χρόνο $O(|T_1| + |T_2| + \cdots + |T_k|)$

Αφού κατασκευαστεί το $gST(\{T_1, T_2\})$σε χρόνο $O(|T_1| + |T_2|)$, τότε μπορεί το παραπάνω ερώτημα να απαντηθεί χρόνο ακόμη $O(|T_1| + |T_2|)$. Σε αυτό το γενικευμένο δένδρο κάποια φύλλα θα έχουν δείκτες για το ένα κείμενο κάποια για το άλλο και κάποια και για τα δύο. Διατρέχονται όλοι οι κόμβοι του δένδρου και υπολογίζεται για κάθε κόμβο το πλήθος των χαρακτήρων του κειμένου από την ρίζα μέχρι και αυτόν. Επιλέγεται ο '*βαθύτερος*' εσωτερικός κόμβος (με το μέγιστη ταμπέλα μονοπατιού) που στο υποδένδρο του έχει φύλλα που απευθύνονται και στα δύο κείμενα. Τότε οι θέσεις που υποδεικνύονται στα φύλλα του υποδένδρου κάτω από αυτόν, έστω του u, είναι οι θέσεις που ξεκινάει η μέγιστη κοινή υποσυμβολοσειρά $L(u)$ (βλ. Σχήμα 2.6).



Σχήμα 2.6. Εύρεση Μεγίστης Κοινής Υποσυμβολοσειράς

Με αντίστοιχο γραμμικό τρόπο μπορεί να βρεθεί και η μέγιστη κοινή υπόσυμβολοσειρά ενός συνόλου συμβολοσειρών χρησιμοποιώντας το γενικευμένο δένδρο επιθεμάτων τους.

2.1.5 Σχολιασμός

Κλείνοντας αυτή την ενότητα και έχοντας ολοκληρώσει την παρουσίαση των δένδρων επιθεμάτων μπορούμε να συνοψίσουμε στο ότι είναι ο κυριότερος και ο καλύτερος αντιπρόσωπος, όσον αφορά το εύρος ερωτήσεων που απαντά, των δομών πλήρους δεικτοδότησης. Το κόστος για αυτή την πληρότητα είναι η έκταση του χώρου μνήμης που αυτή καταλαμβάνει. Όπως είδαμε και προηγουμένως ο χώρος που απαιτείται μπορεί να κυμανθεί ορισμένες φορές και 25 φορές παραπάνω από τα δεδομένα που δεικτοδοτούνται. Αυτό το γεγονός όπως και φτωχή συμπεριφορά στην δευτερεύουσα μνήμη επιβάλει την χρήση των δένδρων επιθεμάτων σε εφαρμογές που μπορούν να περιορισθούν στην κύρια μνήμη ενός υπολογιστικού συστήματος.

2.2 Πίνακες Επιθεμάτων (Suffix Arrays)

Οι πίνακες επιθεμάτων που παρουσιάστηκαν στην [MM93] είναι η κύρια εναλλακτική δομή δεδομένων πλήρους δεικτοδότησης έναντι των δένδρων επιθεμάτων. Τα κύρια πλεονεκτήματα των πινάκων επιθεμάτων είναι οι μικρές απαιτήσεις σε μνήμη, η απλότητα και το ότι η χρονικές πολυπλοκότητες κατασκευής και απάντησης ερωτημάτων δεν εξαρτώνται από το αλφάβητο της συμβολοσειράς. Όλα αυτά βέβαια με τίμημα μεγαλύτερους χρόνους απαντήσεων από τους χρόνους που απαντάει τα αντίστοιχα ερωτήματα το δένδρο επιθεμάτων.

2.2.1 Ορισμός - Περιγραφή

Ορισμός 2.20. *Ένας πίνακας επιθεμάτων SA_T μιας συμβολοσειράς T, n χαρακτήρων, είναι ένας πίνακας που υποδεικνύει την λεξικογραφική διάταξη των επιθεμάτων της T. Δηλαδή σε κάθε θέση i του πίνακα επιθεμάτων αποθηκεύεται η θέση έναρξης j του επιθέματος $T_{j\cdots n}$ ($SA_T[i] = j$) έτσι ώστε τα επιθέματα που είναι λεξικογραφικά μικρότερα να βρίσκονται σε αριστερότερες θέσεις και αυτά που είναι μεγαλύτερα σε δεξιότερες. Συνεπώς ισχύει $T_{SA_T[1]\cdots n} <_L T_{SA_T[2]\cdots n} <_L \cdots <_L T_{SA_T[n]\cdots n}$. Ο ειδικός τερματικός χαρακτήρας \$ θεωρείται μικρότερος λεξικογραφικά από κάθε άλλο γράμμα του αλφάβητου.*

Παράδειγμα 2.21. Όπως και στο Παράδειγμα 2.2, έστω ότι έχουμε τη συμβολοσειρά T =μνημη\$. ο πίνακας επιθεμάτων της T φαίνεται στο σχήμα 2.7 :

		1	2	3	4	5	6
T	=	μ	ν	η	μ	η	$

SA_T	=	6	5	3	4	1	2
		$	η $	η μ η $	μ η $	μ ν η μ η $	ν η μ η $

Σχήμα 2.7. Πίνακας Επιθεμάτων συμβολοσειράς "μνημη\$"

Οι πίνακες επιθεμάτων, όπως και όλες οι διατεταγμένες λεξικογραφικά συμβολοσειρές, διαθέτουν την εξής βασική ιδιότητα:

Ιδιότητα 2.22. Έστω τα επιθέματα που στις θέσεις i, j με $i < j$ διαθέτουν ένα κοινό πρόθεμα x. Δηλαδή $LCP(T_{SA_T[i]\cdots n}, T_{SA_T[j]\cdots n}) = x$. Τότε και όλα τα επιθέματα $T_{SA_T[w]\cdots n}$ που βρίσκονται σε ενδιάμεσες θέσεις $i < w < j$ διαθέτουν το x ως πρόθεμα. Σε περίπτωση που δεν ισχύει κάτι τέτοιο τότε τα $LCP(T_{SA_T[i]\cdots n}, T_{SA_T[j]\cdots n}) = x' < x$ και η υπόθεση μας είναι εσφαλμένη (εις άτοπο απαγωγή).

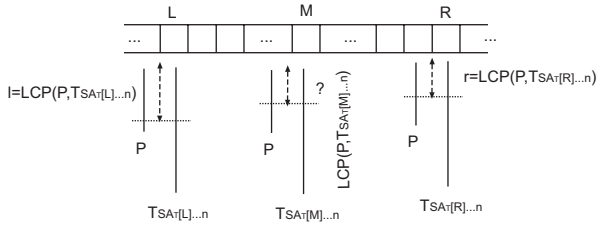
Όπως βλέπουμε ένας πίνακας επιθεμάτων αποτελείται μόνο από n στοιχεία, πολύ μικρότερο από το $6n$ που χαρακτηρίζει τα δένδρα επιθεμάτων. Ας δούμε όμως πως μπορεί να λυθεί το πρόβλημα του ταιριάσματος προτύπων (*Πρόβλημα 1.2*) έχοντας στην διάθεση μας τον πίνακα των επιθεμάτων της T και ένα πρότυπο P. Με βάση την Ιδιότητα 2.22 αρκεί να βρεθούν δύο θέσεις i, j με $i \leq j$ για τις οποίες θα ισχύει το εξής: οι πρώτοι $|P|$ χαρακτήρες του επιθέματος της θέσης i να είναι μικρότεροι ή ίσοι λεξικογραφικά από το πρότυπο (δηλαδή $T_{SA_T[i]\cdots SA_T[i]+|P|} \leq_L P$) ενώ οι πρώτοι $|P|$ χαρακτήρες του επιθέματος της θέσης j να είναι μεγαλύτεροι ή ίσοι λεξικογραφικά από το πρότυπο (δηλαδή $P \leq_L T_{SA_T[j]\cdots SA_T[j]+|P|}$). Σύμφωνα με αυτό τα επιθέματα που αντιστοιχούν στις θέσεις i, j αλλά και σε όλες τις ενδιάμεσες θέσεις έχουν ως πρόθεμα το P. Συνεπώς οι θέσεις εμφάνισης του P μέσα στο T μπορούν να βρεθούν εντοπίζοντας τις δύο ακραίες θέσεις i, j στον πίνακα επιθεμάτων και εν συνεχεία σαρώνοντας και τις ενδιάμεσες θέσεις. Για την εύρεση των ακραίων θέσεων εκτελείται μια δυαδική αναζήτηση πάνω στον πίνακα επιθεμάτων όπου σε κάθε βήμα της αναζήτησης εκτελούνται $|P|$ συγκρίσεις ενώ στη συνέχεια η διαδικασία κινείται αριστερά ή δεξιά. Έτσι με $2|P| \log n = O(|P| \log n)$ συγκρίσεις βρίσκονται οι ακραίες θέσεις και μετά ανακτώνται οι ενδιάμεσες. Συνεπώς το πρόβλημα του ταιριάσματος προτύπων με χρήση των πινάκων επιθεμάτων λύνεται σε χρόνο $O(|P| \log n + \alpha)$, όπου α το μέγεθος της απάντησης.

Ο χρόνος αυτός μπορεί να βελτιωθεί δραματικά σε $O(|P| + \log n + \alpha)$ αν χρησιμοποιηθούν δύο ακόμη πίνακες των $n-2$ στοιχείων με την βοήθεια των οποίων σε κάθε επανάληψη της δυαδικής αναζήτησης δεν είναι αναγκαίο να εκτελεσθούν και οι $|P|$ συγκρίσεις στο επίθεμα που αντιστοιχεί στο μέσο του ενεργού τμήματος. Έστω ότι η δυαδική αναζήτηση βρίσκεται σε ένα διάστημα $[L, R]$ του πίνακα επιθεμάτων και ζητούμενο είναι να υπολογιστεί η τιμή $m = LCP(P, T_{SA_T[M]\cdots n})$ για το μέσο M του διαστήματος. Υποθέτουμε πως οι τιμές $l = LCP(P, T_{SA_T[L]\cdots n})$ και $r = LCP(P, T_{SA_T[R]\cdots n})$ έχουν υπολογιστεί σε προηγούμενη επανάληψη της δυαδικής αναζήτησης (βλ. σχήμα 2.8). Η πρώτη παρατήρηση που μπορεί να γίνει είναι ότι δεν χρειάζεται να γίνουν και οι $|P|$ συγκρίσεις από την αρχή αφού εξαιτίας της Ιδιότητας 2.22 $m \geq min\{l, r\}$, επομένως οι συγκρίσεις συνεχίζουν από τη θέση $m+1$ και έτσι εξοικονομούνται $min\{l, r\}$ συγκρίσεις. Παρόλη αυτή την παρατήρηση η συνολική πολυπλοκότητα δεν αλλάζει τάξη.

Έστω ότι διαθέταμε ακόμη την εξής πληροφορία το μέγιστο κοινό πρόθεμα του επιθέματος του αριστερού άκρου L με το επίθεμα της μέσης και το μέγιστο κοινό πρόθεμα του επιθέματος του δεξιού άκρου R με το επίθεμα της μέσης. Ας τα συμβολίσουμε με $x = LCP(T_{SA_T[L]\cdots n}, T_{SA_T[M]\cdots n})$ και $x' = LCP(T_{SA_T[M]\cdots n}, T_{SA_T[R]\cdots n})$ αντίστοιχα.

Ας υποθέσουμε ότι $l \geq r$ οπότε ισχύουν τα εξής αναλόγως με το αν το x, δηλαδή το μέγιστο κοινό πρόθεμα του αριστερού άκρου με το μεσαίο, είναι μεγαλύτερο, μικρότερο ή ίσο με το l :

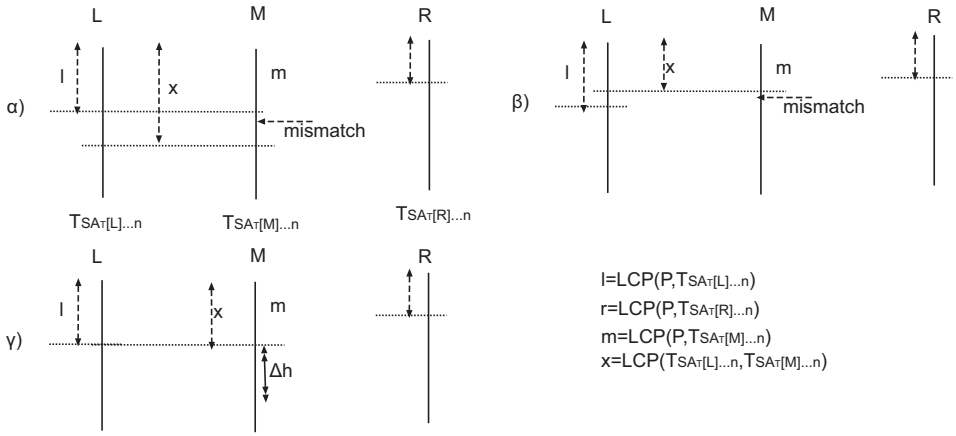
→ Αν $x > l$ τότε αφού το αριστερό άκρο έχει τους πρώτους l χαρακτήρες ίδιους με το P και τους πρώτους x με το επίθεμα M τότε επειδή $x > l$ ο $l+1$ χαρακτήρας του επιθέματος στη θέση M δεν θα ταιριάζει με τον $l+1$ χαρακτήρα του P. Συνεπώς, σύμφωνα με την Ιδιότητα 2.22 δεν υπάρχει ένα



Σχήμα 2.8. Δυαδική αναζήτηση στον πίνακα επιθεμάτων

κοινό πρόθεμα με το P αριστερότερα από το M. Οπότε επιλέγεται ως νέο διάστημα το $[M, R]$ (βλ. σχήμα 2.9α.).

→ Αν $x < l$ τότε, αφού το P έχει ταιριάξει με τους l χαρακτήρες του αριστερού άκρου και με τους x του μεσαίου επιθέματος M, θα έχουμε μη ταίριασμα του μεσαίου επιθέματος στη θέση $x + 1$. Συνεπώς μεγαλύτερο πρόθεμα του P θα βρίσκεται μόνο στο αριστερό διάστημα, οπότε ως νέο διάστημα αναζήτησης επιλέγεται το $[L, M]$(βλ. σχήμα 2.9β.).



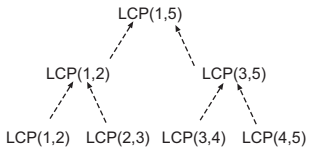
Σχήμα 2.9. Βήμα δυαδικής αναζήτησης στον πίνακα επιθεμάτων

→ Αν $x = l$ τότε δεν μπορεί να αποφανθεί άμεσα αν ένα μεγαλύτερο κοινό πρόθεμα με το P βρίσκεται αριστερά ή δεξιά. Έτσι με συγκρίσεις χαρακτήρα προς χαρακτήρα επεκτείνεται το κοινό πρόθεμα (αν επεκτείνεται) πέρα από την θέση l. Έστω ότι κάναμε Δh επιτυχείς συγκρίσεις. Οπότε το κοινό πρόθεμα του επιθέματος της M με το P έχει μήκος $l + \Delta h$(βλ. σχήμα 2.9γ.). Η αποτυχία στο ταίριασμα στη θέση $l + \Delta h + 1$ μας οδηγεί αριστερά η δεξιά ανάλογα με το αν ο χαρακτήρας της αντίστοιχης θέσης του P είναι λεξικογραφικά μικρότερος ή μεγαλύτερος από την αντίστοιχη θέση του μεσαίου επιθέματος.

Κάθε ένα από τα $O(\log n)$ βήματα της δυαδικής αναζήτησης λοιπόν είτε κάνει σταθερό αριθμό πράξεων (περιπτώσεις $x > l$ ή $x < l$) ή κάνει Δh συγκρίσεις (περίπτωση $x = l$). Το άθροισμα των συγκρίσεων αυτής της τελευταίας περίπτωσης δεν ξεπερνάει τις P αφού πάντα το μέσο θα παίζει το μεγαλύτερο

άκρο στην επόμενη επανάληψη (δηλαδή συνεχώς αυξάνει). Συνεπώς το πρόβλημα του ταιριάσματος προτύπων επιλύεται σε χρόνο $O(P + \log n)$.

Αυτός ο χρόνος όπως είδαμε επιτυγχάνεται χρησιμοποιώντας προ υπολογισμένη πληροφορία, τα $LCP(T_{SA_T[L]\cdots n}, T_{SA_T[M]\cdots n})$ και $LCP(T_{SA_T[M]\cdots n}, T_{SA_T[R]\cdots n})$ για κάθε δυνατό διάστημα που μπορεί να υπάρχει κατά την δυαδική αναζήτησης. Το πλήθος των δυνατών διαστημάτων είναι $n-2$ αφού τον ρόλο μέσων μπορούν να τον έχουν όλα τα στοιχεία πλην των δύο άκρων. Έτσι ο ένας πίνακας αποθηκεύει τις τιμές με το αριστερό άκρο για κάθε δυνατό μέσο ενώ ο άλλος για τα δεξιά. Όπως θα δούμε και στην επόμενη ενότητα με τους αλγόριθμους κατασκευής πινάκων επιθεμάτων αυτοί υπολογίζουν άμεσα μέγιστο κοινό πρόθεμα για γειτονικά επιθέματα στον πίνακα επιθεμάτων, δηλαδή υπολογίζουν τον πίνακα $LCP[i] = LCP(T_{SA_T[i]\cdots n}, T_{SA_T[i+1]\cdots n})$ για $i = 1 : n$. Χρησιμοποιώντας την σχέση $LCP(L, R) = min\{LCP(L, M), LCP(M, R)\}$ από τον παραπάνω πίνακα μπορούν να κατασκευαστούν οι ζητούμενες LCP τιμές για όλα τα δυνατά διαστήματα της δυαδικής αναζήτησης (βλ. σχήμα 2.10γ.).



Σχήμα 2.10. Υπολογισμός πινάκων LCP

Τελειώνοντας βλέπουμε ότι αυτός ο εμπλουτισμένος πίνακας επιθεμάτων καταλαμβάνει χώρο $n + 4n - 4 = 5n - 4$. Παρατηρούμε ότι ακόμη εξακολουθεί να καταλαμβάνει χώρο μικρότερο από τα δένδρα επιθεμάτων ($\geq 6n$ στοιχεία). Ο χρόνος επίλυσης του προβλήματος ταιριάσματος προτύπων $O(P + \log n)$ σε σύγκριση με τον χρόνο που απαιτεί ένα δένδρο επιθεμάτων $O(|\Sigma|P)$ (στην αποδοτική σε χώρο υλοποίηση) ή $O(P)$ (στην αποδοτική σε χρόνο υλοποίηση) μπορεί να είναι καλύτερος . Από την δεύτερη περίπτωση υπερέχουν ξεκάθαρα όσον αφορά το χώρο που χρησιμοποιούν ενώ από την πρώτη υπερέχουν όσον αφορά το χρόνο σε περίπτωση που το αλφάβητο είναι μεγάλο της τάξης του μήκους της συμβολοσειράς.

2.2.2 Αλγόριθμοι Κατασκευής

Στην [MM93] ο αλγόριθμος που παρουσιάστηκε για την κατασκευή ενός πίνακα επιθεμάτων δεν ήταν γραμμικός αλλά απαιτούσε χρόνο $O(n \log n)$. Οι απλοϊκοί τρόποι κατασκευής ενός πίνακα επιθεμάτων σε γραμμικό χρόνο ήταν η κατασκευή ενός δένδρου επιθεμάτων και στη συνέχεια η κατασκευή από αυτό του πίνακα επιθεμάτων. Αυτό μπορεί να επιτευχθεί με λεξικογραφική διαπέραση του δένδρου επιθεμάτων σε γραμμικό χρόνο και ο υπολογισμός των πινάκων LCP με χρήση ερωτημάτων βαθύτερων προγόνων. Αυτή η διαδικασία ναι μεν είναι γραμμική αλλά ακυρώνει το βασικό προτέρημα ενός πίνακα επιθεμάτων, την μικρή κατανάλωση σε χώρο.

Μόλις πρόσφατα, το 2003, παρουσιάστηκαν τρεις εργασίες (ανεξάρτητες μεταξύ τους) που παρουσιάζουν απευθείας γραμμικό σειριακό αλγόριθμο κατασκευής ενός δένδρου επιθεμάτων [KS03, PS03, DJHK03]. Στην συνέχεια θα εξετάσουμε την λύση που προτείνουν στην [KS03] μιας και έχει πιο απλή υλοποίηση έναντι των δύο άλλων.

Ο αλγόριθμός αυτός όπως και ο αλγόριθμος του Farach [Far97], που είδαμε στην προηγούμενη ενότητα για την κατασκευή του δένδρου επιθεμάτων, κατασκευάζει τη δομή για ένα υποσύνολο επιθεμάτων, στη συνέχεια κατασκευάζει την δομή για τα υπόλοιπα επιθέματα από την άλλη δομή και στο τέλος ενώνει τις δύο δομές κατασκευάζοντας την τελική.

Πιο συγκεκριμένα ο αλγόριθμός των [KS03] εκτελεί τα εξής αδρά βήματα:

→ Αναδρομική κατασκευή του πίνακα επιθεμάτων SA_T^R από τα επιθέματα που ξεκινούν από τις θέσεις $i \; mod \; 3 \neq \; 0, i = 1 : n$. Αυτό υποβαθμίζει το μήκος της συμβολοσειράς στα $\frac{2}{3}$.
→ Σε κάθε αναδρομική κλήση, κατασκευή του πίνακα επιθεμάτων SA_T^L από τα υπόλοιπα επιθέματα και το SA_T^T.
→ Συγχώνευση των SA_T^L και SA_T^R στο τελικό SA_T

Στο πρώτο βήμα λαμβάνονται τα επιθέματα $T_{i\ldots n}$ με $i \; mod \; 3 \neq \; 0, i = 1 : n$. Για κάθε τέτοιο επίθεμα σχηματίζεται η τριάδα χαρακτήρων $T_{i\ldots i+2}$. Οι τριάδες που σχηματίζονται λαμβάνουν μια ταμπέλα l_i για τις οποίες ισχύει για κάθε ζεύγος τριάδων $T_{i\ldots i+2} \leq_L T_{j\ldots j+2}$ ισχύει $l_i \leq l_j$. Συνεπώς οι ταμπέλες θα είναι από 1 μέχρι και $\frac{2n}{3}$ ενώ για όμοιες τριάδες δεν δίνουμε κάποιο καινούργια ταμπέλα. Αυτό μπορεί να επιτευχθεί χρησιμοποιώντας το radix sort σχήμα σε χρόνο $3\frac{2n}{3} = O(n)$. Με αυτό τον τρόπο σχηματίζεται μια νέα συμβολοσειρά $T' = l_1 l_2 \cdots l_{\frac{2n}{3}}$ μήκους το πολύ $\frac{2n}{3}$. Σε περίπτωση που όλες οι ταμπέλες είναι διαφορετικές μεταξύ τους σημαίνει ότι κανένα από αυτά τα επιθέματα δεν έχουν ίδιους τους τρεις πρώτους χαρακτήρες τους, συνεπώς με τη διαδικασία λεξικογραφικής διάταξης τους έχει ολοκληρωθεί και η ζητούμενη κατασκευή του SA_T^R του πρώτου βήματος. Σε αντίθετη περίπτωση εφαρμόζεται το ίδιο αναδρομικά στην συμβολοσειρά $T'' = l_i \cdots l_x \$ l_i \cdots l_y$ με το πρώτο μισό να περιέχει τα l_i για τα οποία $i \; mod \; 3 = 1$ ενώ το δεύτερο μισό περιέχει τα l_i για τα οποία $i \; mod \; 3 = 2$. Με αυτό τον τρόπο στο πρώτο μισό διατάσσονται αυτά που αντιστοιχούν στο $i \; mod \; 3 = 1$ ενώ με το δεύτερο μισό θα διαταχθούν αυτά που αντιστοιχούν στο $i \; mod \; 3 = 2$.Αυτό ισχύει διότι $T''_{i\ldots\frac{n}{3}}$ αντιστοιχεί στο $T''_{i\ldots n}$ με $i \; mod \; 3 = 1$ ενώ το $T''_{\frac{n+i-1}{3}\ldots\frac{2n}{3}}$ αντιστοιχεί στο $T''_{i\ldots n}$ για $i \; mod \; 3 = 2$

Στο δεύτερο βήμα και αφού έχει υπολογιστεί το SA_T^R εύκολα μπορούν να διαταχθούν με n πράξεις τα επιθέματα $T_{i\ldots n}$ με $i \; mod \; 3 = \; 0$ αφού στην ουσία μπορούν να ειδωθούν ως $T_i T_{i+1\ldots n}$. Συνεπώς είναι σαν να προστίθεται ένα ακόμη ψηφίο και άρα ένα ακόμη πέρασμα στον radix sort αλγόριθμο.

Από την στιγμή που έχουν βρεθεί οι SA_T^L και SA_T^R συγχωνεύονται σχηματίζοντας τον τελικό SA_T. Στον πρώτο πίνακα έχουμε τα επιθέματα που αντιστοιχούν σε $i \; mod \; 3 = \; 0$ και αυτά μπορούν να ειδωθούν ως $T_i T_{i+1\ldots n}$. Τα επιθέματα που αντιστοιχούν σε $j \; mod \; 3 = \; 1$ και αυτά μπορούν να ειδωθούν ως $T_j T_{j+1\ldots n}$. Τα $T_{i+1\ldots n}, T_{j+1\ldots n}$ βρίσκονται διατεταγμένα στον πίνακα SA_T^R συνεπώς μπορεί να βρεθεί η λεξικογραφική τους σχέση στα σταθερό χρόνο.

Επίσης τα επιθέματα που αντιστοιχούν σε $i \ mod\ 3 = 2$ και αυτά μπορούν να ειδωθούν ως $T_{j'}T_{j'+1}T_{i+1\cdots n}$. Βλέπουμε ότι μόνο για τα δύο σημαντικότερα ψηφία δεν έχει ολοκληρωθεί το radix sort, οπότε αυτό ολοκληρώνεται μετά από δύο περάσματα όπου συνολικά εκτελούνται $2n$ πράξεις.

Από τα παραπάνω προκύπτει ότι ο αλγόριθμος ακολουθεί την αναδρομή $T(n) = O(n) + T(\frac{2}{3}n)$ συνεπώς προκύπτει το ακόλουθο θεώρημα:

Θεώρημα 2.23. *Ένας πίνακας επιθεμάτων μιας συμβολοσειράς T κατασκευάζεται σε $O(|T|)$.*

Η εύρεση του πίνακα LCP με $LCP[i] = LCP(T_{SA_T[i]\cdots n}, T_{SA_T[i+1]\cdots n})$ για την δημιουργία του εμπλουτισμένου πίνακα επιθεμάτων μπορεί να γίνει σε γραμμικό χώρο. Παρόλο που στην [KS03] δίνουν γραμμικό αλγόριθμο αυτός χρησιμοποιεί τεχνικές υπολογισμού βαθύτερου προγόνου σε σταθερό χρόνο που η υλοποίηση τους είναι αρκετά σύνθετη. Εδώ θα δούμε τον αλγόριθμο που προτείνεται στην [KLA+01] όπου είναι πιο απλή.

Ας δούμε πρώτα κάποια χαρακτηριστικά που έχει αυτός ο πίνακας.

Ορισμός 2.24. *Για κάθε επίθεμα i, $i = 1 : n$, ορίζουμε ως $RANK[i] = j$ όπου j η σχετική θέση του επιθέματος στον πίνακα επιθεμάτων.*

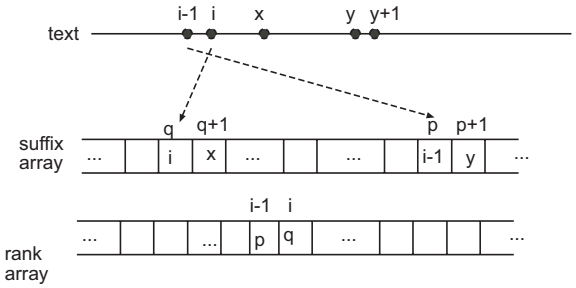
Ιδιότητα 2.25. Για κάθε θέσεις i, j του δένδρου επιθεμάτων με $i < j$ τότε ισχύει ότι: $LCP(T_{SA_T[i]\cdots n}, T_{SA_T[j]\cdots n}) = min_{i\leq x<j}\{LCP(T_{SA_T[x]\cdots n}, T_{SA_T[x+1]\cdots n})\}$ που προκύπτει άμεσα από την Ιδιότητα 2.22. Αυτό μπορεί να διατυπωθεί και με διαφορετικό τρόπο: Το μέγιστο κοινό πρόθεμα γειτονικών επιθεμάτων στον πίνακα επιθεμάτων θα είναι ίσο ή μεγαλύτερο από το μέγιστο κοινό πρόθεμα δύο επιθεμάτων που βρίσκονται εκατέρωθεν (το ένα σε πιο αριστερή και το άλλο σε πιο δεξιά θέση) από αυτά. Δηλαδή $LCP(T_{SA_T[i]\cdots n}, T_{SA_T[i+1]\cdots n}) \geq LCP(T_{SA_T[x]\cdots n}, T_{SA_T[y]\cdots n})$ με $x \leq i$ και $y > i+1$.

Ιδιότητα 2.26. Όταν ένας αριθμός από γειτονικά επιθέματα στον πίνακα επιθεμάτων μοιράζονται ένα κοινό πρόθεμα x με $|x| > 1$ τότε η λεξικογραφική τους διάταξη διατηρείται όταν διαγράφεται ο πρώτος χαρακτήρας. Όλα αυτά τα επιθέματα διαφοροποιούνται στη θέση $|x| + 1$ οπότε η διαγραφή του πρώτου χαρακτήρα δεν διαταράσσει την διάταξη. Αυτό έχει ως αποτέλεσμα ότι αν $LCP(T_{SA_T[i]\cdots n}, T_{SA_T[i+1]\cdots n}) > 1$ τότε $RANK[T_{SA_T[i]+1\cdots n}] < RANK[T_{SA_T[i+1]+1\cdots n}]$.

Ιδιότητα 2.27. Όταν δύο γειτονικά επιθέματα στον πίνακα επιθεμάτωνi, j μοιράζονται ένα μέγιστο κοινό πρόθεμα x χαρακτήρων, τότε τα αμέσως επόμενα από αυτά επιθέματα, τα $i+1, j+1$, θα έχουν ως μέγιστο κοινό πρόθεμα τους πρώτους $x-1$ χαρακτήρες

Έστω ότι έχουμε υπολογίσει το μέγιστο κοινό πρόθεμα του επιθέματος $T_{i-1\cdots n}$, το οποίο έχει $RANK[i-1] = p$, με επόμενο επίθεμα στον πίνακα επιθεμάτων (βλ. σχήμα 2.11). Έστω ότι αυτό είναι το επίθεμα $T_{y\cdots n}$, συνεπώς γνωρίζουμε την τιμή $LCP(T_{i-1\cdots n}, T_{y\cdots n}) = h \geq 0$. Θέλουμε να δούμε πως από αυτό το δεδομένο μπορούμε να υπολογίσουμε το μέγιστο κοινό πρόθεμα του επιθέματος $T_{i\cdots n}$, το οποίο έχει $RANK[i-1] = q$, με το επόμενο επίθεμα $T_{x\cdots n}$στον πίνακα επιθεμάτων.

Εφόσον γνωρίζουμε ότι $LCP(T_{i-1\cdots n}, T_{y\cdots n}) = h \geq 0 \Rightarrow RANK[y] = RANK[i-1]+1$ αυτό σημαίνει ότι $RANK[y+1] < RANK[i]$ από την Ιδιότητα 2.26. Συνεπώς $LCP(T_{i\cdots n}, T_{x\cdots n}) \geq LCP(T_{i\cdots n}, T_{y+1\cdots n}) = LCP(T_{i-1\cdots n}, T_{y\cdots n})-1$ από την Ιδιότητα 2.25.



Σχήμα 2.11. Βήμα Υπολογισμού $LCP[i]$

Με βάση αυτή την τελευταία σχέση:

$$LCP(T_{i\cdots n}, T_{x\cdots n}) \geq LCP(T_{i-1\cdots n}, T_{y\cdots n}) - 1$$

για τα επιθέματα $T_{i\cdots n}$, με το i να κινείται από 1 μέχρι $n-1$. Έχοντας υπολογίσει την τιμή $LCP(T_{i-1\cdots n}, T_{y\cdots n}) = h_{i-1} \geq 0$ από την προηγούμενη επανάληψη μπορούμε με βάση αυτή την ανισότητα μπορεί να υπολογιστεί το $LCP(T_{i\cdots n}, T_{x\cdots n})$ αρχίζοντας τις συγκρίσεις από την θέση h_i των δύο επιθεμάτων. Με αυτό τον τρόπο εξοικονομούνται $h_{i-1} - 1$ συγκρίσεις σε κάθε επανάληψη. Σε κάθε επανάληψη ισχύει $h_i = h_{i-1}+\#m_i+1 \Rightarrow \#m_i = h_i - h_{i-1}$ όπου $\#m$ ο αριθμός των επιτυχών συγκρίσεων για την επέκταση του μέγιστου κοινού προθέματος. Ο αριθμός συγκρίσεων σε κάθε βήμα είναι $C_i = \#m_i + 1$. Συνεπώς συνολικά εκτελούνται $\sum \#C_i = n + \sum h_i - h_{i-1} = h_{n-1} - h_0 = n + h_{n-1}$ συγκρίσεις το οι οποίες είναι λιγότερες ή ίσες με $2n$.

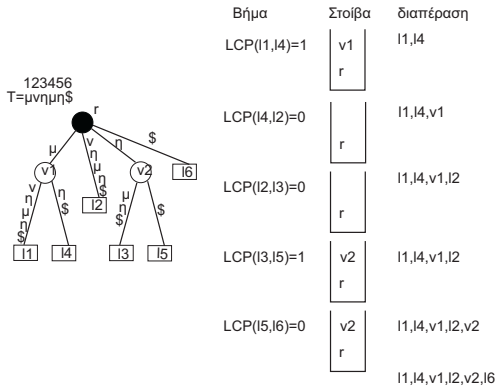
Με βάση τα παραπάνω προκύπτει ότι :

Θεώρημα 2.28. *Ο πίνακας LCP ενός πίνακα επιθεμάτων μπορεί να υπολογιστεί σε γραμμικό χρόνο.*

2.2.3 Εφαρμογές

Κύρια αδυναμία των πινάκων επιθεμάτων, όσον αφορά τις εφαρμογές, είναι ότι δεν έχει μεγάλο εύρος ερωτήσεων στις οποίες μπορεί άμεσα να απαντήσει όπως το δένδρο επιθεμάτων. Στην εργασία [KLA+01] δίνεται μια μέθοδος με την οποία εξομοιώνεται μια διαπέραση μεταδιάταξης (post-order traversal) του δένδρου επιθεμάτων μιας συμβολοσειράς T έχοντας στην διάθεση μας έναν πίνακα επιθεμάτων μαζί με τον πίνακα LCP. Αυτό δίνει την δυνατότητα μερικές από τις εφαρμογές του δένδρου επιθεμάτων που εκτελούν υπολογισμούς από τα φύλλα προς την ρίζα (bottom-up) να μπορούν να εκτελεσθούν με ορισμένες επιπλέον αλλαγές στους πίνακες επιθεμάτων.

Ο πίνακας επιθεμάτων διατρέχεται από τα αριστερά προς τα δεξιά, οπότε $i = 1 : n$ και κρατιέται και μια στοίβα S. Αρχικά η στοίβα θα περιέχει την



Σχήμα 2.12. Παράδειγμα μεταδιάταξης δένδρου επιθεμάτων με χρήση πίνακα επιθεμάτων.

ρίζα. Αρχικά $LCP[1] = LCP(T_{SA_T[1]\cdots n}, T_{SA_T[2]\cdots n}) \geq 0$. Αν είναι ίσο με το μηδέν τότε τα δύο αριστερά φύλλα-επιθέματα έχουν ελάχιστο κοινό πρόγονο την ρίζα άρα κατά την διαπέραση αγγίζεται το πρώτο και στην συνέχεια το δεύτερο. Αν η τιμή είναι μεγαλύτερη από μηδέν τότε υπάρχει και κάποιος εσωτερικός κόμβος ο οποίες ενθέτεται στην στοίβα S. Γενικότερα στο βήμα i: αν η $LCP[i] = (T_{SA_T[i]\cdots n}, T_{SA_T[i+1]\cdots n})$ είναι μεγαλύτερη από το βάθος του κόμβου u της κορυφής της στοίβας (δηλαδή το μήκος της ετικέτας μονοπατιού $L(u)$), τότε μεταξύ του i-στου φύλλου/επιθέματος και του αμέσως επόμενου θα υπάρχει ένας βαθύτερος κόμβος ο οποίος και εισάγεται στην S, αν τώρα η τιμή $LCP[i]$ είναι μικρότερη από τη βάθος του κόμβου u τότε ο ελάχιστος κοινός πρόγονος βρίσκεται πιο ψηλά στο μονοπάτι από τον u προς την ρίζα. Σε αυτή την περίπτωση αδειάζει η στοίβα μέχρι να βρεθεί κόμβος με μικρότερο βάθος και εφαρμόζεται τότε ότι και στο πρώτο ενδεχόμενο. Με βάση αυτή τη διαδικασία ένας κόμβος εισάγεται στην στοίβα όταν συναντάται καθώς κατεβαίνει η διαδικασία από πάνω προς τα κάτω ενώ πετάγεται έξω από την στοίβα όταν συναντάται από κάτω προς τα πάνω για τελευταία φορά (βλ παράδειγμα σχήματος 2.12).

Σε κάθε βήμα της μεθόδου είτε προστίθεται ένας κόμβος στην στοίβα είτε έχουμε έναν αριθμό αφαιρέσεων από την στοίβα. Επειδή κάθε κόμβος προστίθεται στην στοίβα και αφαιρείται μόνο μία φορά τότε συνολικά όλη η διαδικασία χρειάζεται χρόνο $O(n)$.

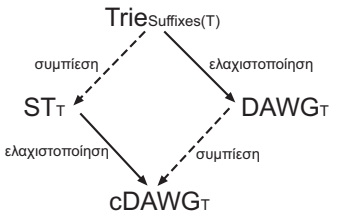
2.2.4 Σχολιασμός

Κλείνοντας αυτή την ενότητα και έχοντας ολοκληρώσει την παρουσίαση των πινάκων επιθεμάτων μπορούμε να συνοψίσουμε στο ότι είναι μια καλή εναλλακτική λύση των δένδρων επιθεμάτων ειδικά σε περιπτώσεις που είτε υπάρχει περιορισμός της κύριας μνήμης είτε το αλφάβητο είναι μεγάλο.

2.3 Άκυκλοι Κατευθυνόμενοι Γράφοι Λέξεων (DAWG & cDAWG)

Σε αυτή την ενότητα θα δούμε τους άκυκλους κατευθυνόμενους γράφους λέξεων (DAWG, [Cro85]) και τους συμπιεσμένους άκυκλους κατευθυνόμενους

γράφους λέξεων (cDAWG,[BBH+87]). Και οι δύο αυτές δομές πρόκειται για δομές που προέρχονται από ένα ψηφιακό δένδρο (trie) που περιέχει όλα τα επιθέματα μιας συμβολοσειράς, όπως και το δένδρο επιθεμάτων. Για αυτό το λόγο και είναι ταυτόσημες δομές δεικτοδότησης και επιλύουν τα ίδια προβλήματα. Ο τρόπος με τον οποίο σχετίζονται οι δομές αυτές υποδείχθηκε στην εργασία [CS85] και περιγράφεται στο σχήμα 2.13.

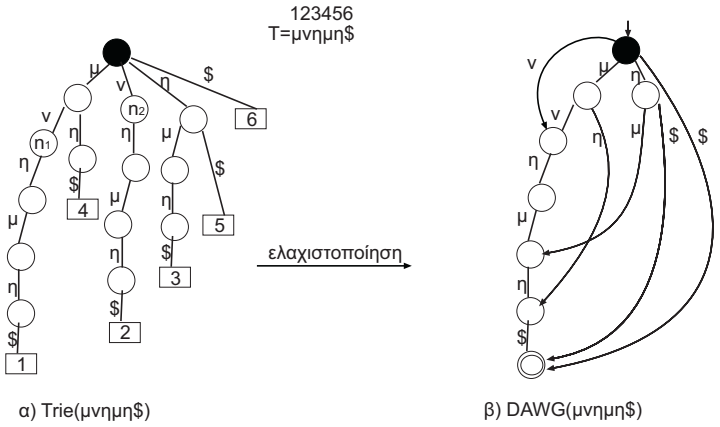


Σχήμα 2.13. Σχέση Δομών Δεικτοδότησης.

2.3.1 Ορισμός - Περιγραφή

Ορισμός 2.29. *Ένας άκυκλος κατευθυνόμενος γράφος $DAWG_T$ μιας συμβολοσειράς T είναι το ελάχιστο ντετερμινιστικό αυτόματο που δέχεται όλα τα επιθέματα της T.*

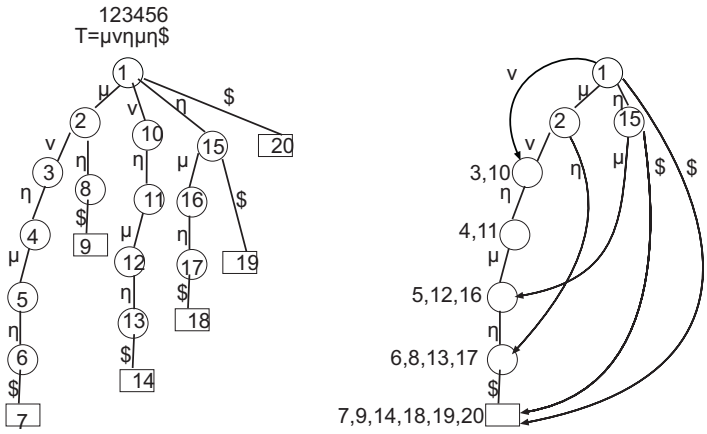
Παράδειγμα 2.30. Όπως και στο Παράδειγμα 2.2, έστω ότι έχουμε τη συμβολοσειρά T =μνημη$. ο άκυκλος κατευθυνόμενος γράφος λέξεων της T φαίνεται στο σχήμα 2.14β. :



Σχήμα 2.14. DAWG συμβολοσειράς "μνημη$"

Αν για μια υποσυμβολοσειρά x της συμβολοσειράς T ορίσουμε ως $end\_pos(x)$ το σύνολο των θέσεων όπου τερματίζουν όλες οι εμφανίσεις της x μέσα στην T. Δηλαδή $end\_pos(x) = \{i : T_{i-|x|...i} = x\}$. Έστω τώρα ότι για τουλάχιστον δύο διαφορετικές υποσυμβολοσειρές x, y ότι ισχύει $end\_pos(x) = end\_pos(y)$ τότε

η μικρότερη θα είναι επίθεμα της μεγαλύτερης. Έστω ότι $|x| < |y|$ τότε αφού για της θέσεις τερματισμού $i \in end\_pos(x) = end\_pos(y)$ έχουμε $T_{i-|y|\cdots i} = y$ και $T_{i-|x|\cdots i} = x$ τότε $y_{|y|-|x|\cdots|y|} = x$. Αυτές οι υποσυμβολοσειρές καλούνται ισομορφικές . Θεωρήστε για δύο ισομορφικές υποσυμβολοσειρές, έστω x, y και τους κόμβους n_1, n_2 (καλούνται και αυτοί ισομορφικοί)που έχουν ως ετικέτες μονοπατιού τα x, y αντίστοιχα στο trie με τα επιθέματα. Για το παράδειγμα που είδαμε προηγουμένως (βλ. Παράδειγμα 2.14) έστω ότι y =μν και x =ν που είναι ισομορφικά αφού $end\_pos(x) = end\_pos(y) = 2$. Τα υποδένδρα κάτω από αυτούς τους κόμβους είναι πανομοιότυπα αφού από τις θέσεις $end\_pos(x) = end\_pos(y)$ και δεξιότερα υπάρχουν οι ίδιες ακολουθίες χαρακτήρων. Αυτά τα πανομοιότυπα (ή αλλιώς ισομορφικά υποδένδρα) δεν χρειάζεται να αποθηκεύονται παρά μόνο μία φορά στο trie των επιθεμάτων ταυτίζοντας τους ισομορφικούς κόμβους. Εφαρμόζοντας αυτήν την αρχή για κάθε ισομορφική υποσυμβολοσειρά της T, ελαχιστοποιείται το trie των επιθεμάτων και λαμβάνεται το ελάχιστο ντετερμινιστικό αυτόματο που δέχεται όλα τα επιθέματα της (βλ. Παράδειγμα 2.15).



Φ={
end_pos(1)={1,2,3,4,5,6}
end_pos(2)={1,4}
end_pos(3)=end_pos(10)={2}
end_pos(4)=end_pos(11)={3}
end_pos(5)=end_pos(12)=end_pos(16)={4}
end_pos(6)=end_pos(8)=end_pos(13)=end_pos(17)={5}
end_pos(7)=end_pos(9)=end_pos(14)=end_pos(18)=end_pos(19)=end_pos(20)={6}
end_pos(15)={3,5}
}

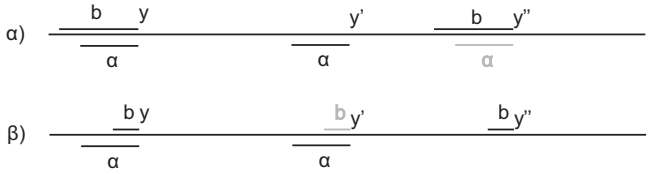
Σχήμα 2.15. DAWG συμβολοσειράς "μνημη$" και οι *end pos*(.)

Ας ορίσουμε ως Φ την οικογένεια όλων των συνόλων $end\_pos(x)$ για κάθε εσωτερικό κόμβο x του *dawg*. Παρατηρούμε ότι:

→ κάθε κόμβος x έχει μοναδικό σύνολο $end\_pos(x)$ (βλ. Σχήμα 2.15), αυτό είναι λογικό αφού κόμβοι με ίδια $end\_pos(.)$ στο trie επιθεμάτων έχουν γίνει ένας στο *dawg*

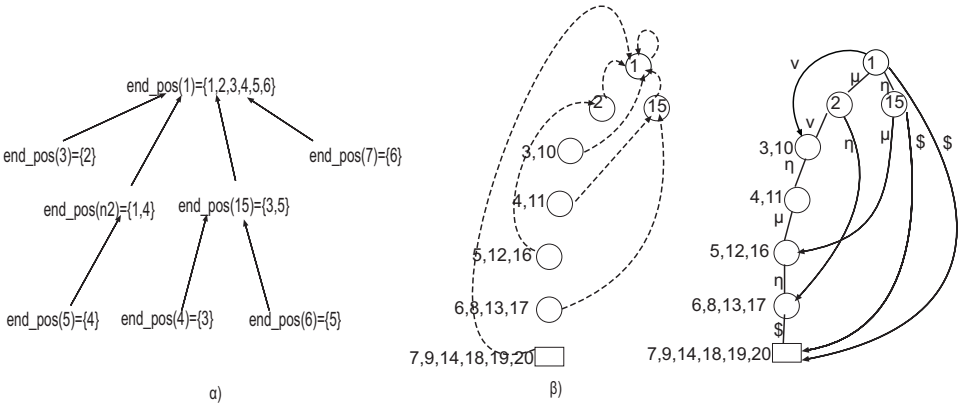
→ κάθε ζεύγος από την Φ (έστω $end\_pos(n_i)$ και $end\_pos(n_j)$) είναι είτε ξένα σύνολα είτε το ένα εμπεριέχεται εξολοκλήρου στο άλλο. Το να είναι

μερικώς επικαλυπτόμενα είναι αδύνατον γιατί έστω ότι $y, y' \in end\_pos(n_i)$ που αντιστοιχούν στην ετικέτα μονοπατιού a του n_i και $y, y'' \in end\_pos(n_j)$ που αντιστοιχούν στην ετικέτα μονοπατιού b του n_j. Αφού το το y ανήκει και στα δύο σύνολα τότε ή το a είναι επίθεμα του b ή το αντίθετο.Στην πρώτη περίπτωση το a αναγκαστικά θα τερματίζει και στην θέση y'' οπότε και $y'' \in end\_pos(n_i)$ (βλ. Σχήμα 2.16α.). Αν ισχύει η δεύτερη περίπτωση τότε το b θα τερματίζει και στην θέση y' οπότε και $y' \in end\_pos(n_j)$ (βλ. Σχήμα 2.16β.) Συνεπώς είτε δύο σύνολα είναι ξένα μεταξύ τους είτε το ένα εμπεριέχεται εξολοκλήρου στο άλλο.



Σχήμα 2.16. Περιπτώσεις επικάλυψης συνόλων $end\_pos(.)$

Εξαιτίας της δεύτερης παρατήρησης παρατηρούμε ότι η Φ σχηματίζει μια δενδρική δομή (βλ. Σχήμα 2.17). Τα φύλλα είναι ανά δύο ξένα μεταξύ τους ενώ οι εσωτερικοί κόμβοι είναι υπερσύνολα των απόγονων τους. Παρόλο που το πολύ να είναι n δεν μπορούμε άμεσα να βγάλουμε συμπέρασμα για τον συνολικό αριθμό των κόμβων αφού υπάρχουν και κόμβοι με μόνο ένα παιδί. Θεωρούμε ότι χωρίζονται οι κόμβοι της παραπάνω δεντρικής δομής σε δύο σύνολα με βάση με το αν η ετικέτα μονοπατιού τους αντιστοιχεί σε πρόθεμα ή όχι της αρχικής συμβολοσειράς T. Το πρώτο σύνολο γνωρίζουμε ότι έχει μέγεθος το πολύ $n+1$ όσα δηλαδή και όλα τα δυνατά προθέματα μαζί με το μηδενικό που αντιστοιχεί στην ρίζα. Παρατηρήστε ότι στο τελικό αυτόματο αυτοί οι κόμβοι αποτελούν τον κύριο κορμό του (βλ. Σχήμα 2.15α.). Στόχος είναι λοιπόν να προσδιοριστεί και το μέγεθος του άλλου συνόλου. Οι κόμβοι του συνόλου αυτού θα έχουν $|end\_pos(.)| \geq 2$ γιατί είναι σίγουρα επίθεμα ενός προθέματος που αντιστοιχεί σε μια ετικέτα μονοπατιού του κύριου κορμού.



Σχήμα 2.17. Δενδρική Διασύνδεση των συνόλων $end\_pos(.)$

Ας θεωρήσουμε τέλος για κάθε κόμβο u και έναν δείκτη suf_u οποίος να δείχνει τον κόμβο w με ετικέτα μονοπατιού που αντιστοιχεί στο μεγαλύτερο επίθεμα της ετικέτας μονοπατιού του u (βλ. Σχήμα 2.15β.). Παρατηρήστε την αναλογία με τα δένδρα επιθεμάτων. Για τους εσωτερικούς κόμβους του δεύτερου συνόλου που η ετικέτα μονοπατιού τους παρουσιάζει τουλάχιστον δύο εμφανίσεις θα υπάρχουν τουλάχιστον δύο κόμβοι που θα τους υποδεικνύουν με τους suf_u δείκτες τους. Παρατηρώντας το δένδρο που σχηματίζεται από αυτούς τους δείκτες βλέπουμε ότι έχει σαν φύλλα τους κόμβους του πρώτου συνόλου επομένως οι κόμβοι του δεύτερου συνόλου που σε αυτό το δένδρο έχουν τουλάχιστον δύο παιδιά θα είναι το πολύ $n-1$. Συνεπώς οι κόμβοι και των δύο συνόλων και άρα και του $dawg$ θα είναι το πολύ $2n$. Οι πλευρές/ακμές μετάβασης του $dawg$ θα είναι το πολύ $2n-1$ αφού σε κάθε κόμβο υπάρχει τουλάχιστον μια εισερχόμενη πλευρά που υπήρχε και στο trie των επιθεμάτων. Από την ένωση κόμβων μπορεί να υπάρχει το εξής: να υπάρχουν και επιπρόσθετες πλευρές οι οποίες αντιστοιχούν στα επιθέματα xay με $|a|=1$ την ετικέτα αυτής της πλευράς ενώ τα x, y είναι τα υπόλοιπα πρόθεμα, επίθεμα που σχηματίζουν το ολικό επίθεμα. Τα x, y θα υπάρχουν είδη στο μέσα στο trie. Συνεπώς αφού τα επιθέματα πλην του μεγαλυτέρου που αποτελεί το βασικό κορμό είναι $n-1$ συνεπώς και οι επιπρόσθετες αυτές πλευρές είναι $n-1$. Συνεπώς ισχύει το ακόλουθο θεώρημα:

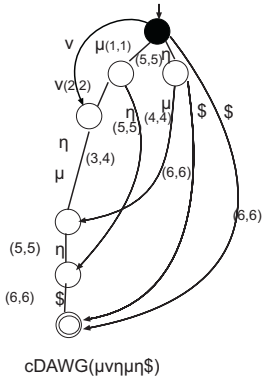
Θεώρημα 2.31. *Ένας άκυκλος κατευθυνόμενος γράφος $DAWG_T$ μιας συμβολοσειράς T αποτελείται από $2|T|$ κόμβους και $3|T|-1$. Άρα αν κάθε κόμβος σημειώνει το $suffix\ link$ ενώ κάθε πλευρά τον χαρακτήρα και έναν δείκτη συνολικά χρειάζονται το πολύ $2|T|+6|T|-2 \approx 8|T|$ στοιχεία.*

Παρατηρήστε ότι ο χώρος που καταλαμβάνουν είναι αρκετά στοιχεία παραπάνω από τους πίνακες επιθεμάτων ($2n$) και λίγο πιο πάνω από το δένδρο επιθεμάτων ($6n$). Ο χώρος που καταλαμβάνει ένας άκυκλος κατευθυνόμενος γράφος $DAWG_T$ μπορεί να μειωθεί περαιτέρω συμπιέζοντας τις αλυσίδες κόμβων με μόνο μία εξερχόμενη πλευρά. Κάτι αντίστοιχο γίνεται στο trie επιθεμάτων λαμβάνοντας το δένδρο επιθεμάτων. Με αυτή την διαδικασία προκύπτουν οι άκυκλοι κατευθυνόμενοι γράφοι $cDAWG_T$.

Ορισμός 2.32. *Ένας συμπαγής άκυκλος κατευθυνόμενος γράφος $cDAWG_T$ μιας συμβολοσειράς T είναι το ελάχιστο ντετερμινιστικό αυτόματο που δέχεται όλα τα επιθέματα της T και το οποίο βρίσκεται σε συμπτυγμένη μορφή. Δηλαδή αλυσίδες καταστάσεων με βαθμό εξόδου ίσου με ένα αντικαθιστώνται με μία ακμή.*

Παράδειγμα 2.33. Όπως και στο Παράδειγμα 2.2, έστω ότι έχουμε τη συμβολοσειρά T =μνημη\$. ο συμπαγής άκυκλος κατευθυνόμενος γράφος λέξεων της T φαίνεται στο σχήμα 2.14β. :

Ένα $cDAWG$ μιας συμβολοσειράς μπορεί λογικά να σχηματιστεί από το δένδρο επιθεμάτων με τρόπο αντίστοιχο του σχηματισμού των $DAWG's$ από το trie επιθεμάτων. Έχοντας στην διάθεση το δένδρο επιθεμάτων που ήδη είναι σε συμπαγή μορφή, μπορούν να βρεθούν τα ισομορφικά υποδένδρα του και να ενωθούν σχηματίζοντας έτσι έναν συμπαγή άκυκλο κατευθυνόμενο γράφο



Σχήμα 2.18. cDAWG συμβολοσειράς "μνημη$"

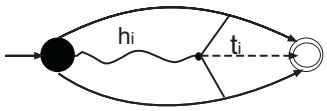
της συμβολοσειράς. Με βάση αυτή την λογική ένα $cDAWG$ θα καταλαμβάνει λιγότερο χώρο και από το $DAWG$ αλλά και από το αντίστοιχό δένδρο επιθεμάτων. Στην [Kur99], όπου προτάθηκαν οι αποδοτικές τεχνικές υλοποίησης των δένδρων επιθεμάτων και τις οποίες παρουσιάσαμε στην Ενότητα 2.1.3, δείχθηκε κατόπιν πειραματικής αξιολόγησης ότι τα $cDAWG's$ χρησιμοποιούν λίγο περισσότερο χώρο από τα δένδρα επιθεμάτων. Η λύση του κύριου προβλήματος του ταιριάσματος (*Πρόβλημα 1.2*). από τους άκυκλους κατευθυνόμενους γράφους είναι πανομοιότυπη και τελείται σε χρόνο $O(|P|)$ ή $O(|\Sigma||P|)$, ανάλογα με τον τρόπο που έχουν οι υλοποιηθεί οι εξερχόμενες πλευρές στους κόμβους (με πίνακες/*hashing* σχήματα ή με λίστες) με τις συναλλαγές στο χώρο που αναφέρθηκαν στην παράγραφο των δένδρων επιθεμάτων.

2.3.2 Αλγόριθμοι Κατασκευής

Η κατασκευή του $DAWG$ ή του $cDAWG$ μιας συμβολοσειράς T σε καμία περίπτωση δεν θα χρησίμευε να αρχίσει με δομή έναρξης το trie επιθεμάτων μιας και θα χρειαζόταν προσωρινά μεγάλος αποθηκευτικός χώρος, πολύ μεγαλύτερος από το χώρο που τελικά θα καταλαμβάνουν οι τελικές δομές, αλλά και μη γραμμικό χρόνο. Στις εργασίες [Cro85, BBH$^+$87, CR02] παρουσιάζονται απευθείας γραμμικοί αλγόριθμοι για την κατασκευή $DAWG's$ οι οποίοι είναι ταυτόσημοι ενώ στις [CV97, IHS$^+$01] προτείνονται δύο γραμμικοί αλγόριθμοι για την απευθείας κατασκευή ενός $cDAWG$. Ο πρώτος στηρίζεται στον αλγόριθμο του McCreight για την κατασκευή των δένδρων επιθεμάτων ενώ ο δεύτερος στο αλγόριθμο του Ukkonen.

Και στους δύο παραπάνω αλγόριθμους συνδυάζονται οι αλγόριθμοι κατασκευής των δένδρων επιθεμάτων με τις ιδιότητες που εμφανίζουν τα DAWG's. Ο γραμμικός αλγόριθμος της εργασίας [CV97] όπως και ο αλγόριθμος του McCreight εισάγει τα επιθέματα στη δομή δεικτοδότησης από το μεγαλύτερο προς το μικρότερο. Σε κάθε τέτοιο βήμα i βρίσκει την θέση στην υπάρχουσα δομή του μέγιστου προθέματος h_i του προς εισαγωγή επιθέματος $T_{i...n}$ και στη συνέχεια προσθέτει το υπόλοιπο $t_i = T_{i+|h_i|...n}$. Το υπόλοιπο δεν προστίθεται σαν μια πλευρά και ένα φύλλο όπως στο δένδρο επιθεμάτων αλλά ως μια μετάβαση πολλών χαρακτήρων που οδηγεί στην τελική κατάσταση του $cDAWG$ (βλ. σχήμα 2.19). Αν το μονοπάτι που υπαγορεύεται από το h_i τερματίζει στη μέση

μιας πλευράς τότε αυτή διασπάται και δημιουργείται μια νέα κατάσταση στην οποία προστίθεται και η μετάβαση του t_i, όπως συμβαίνει και με τα δένδρα επιθεμάτων.

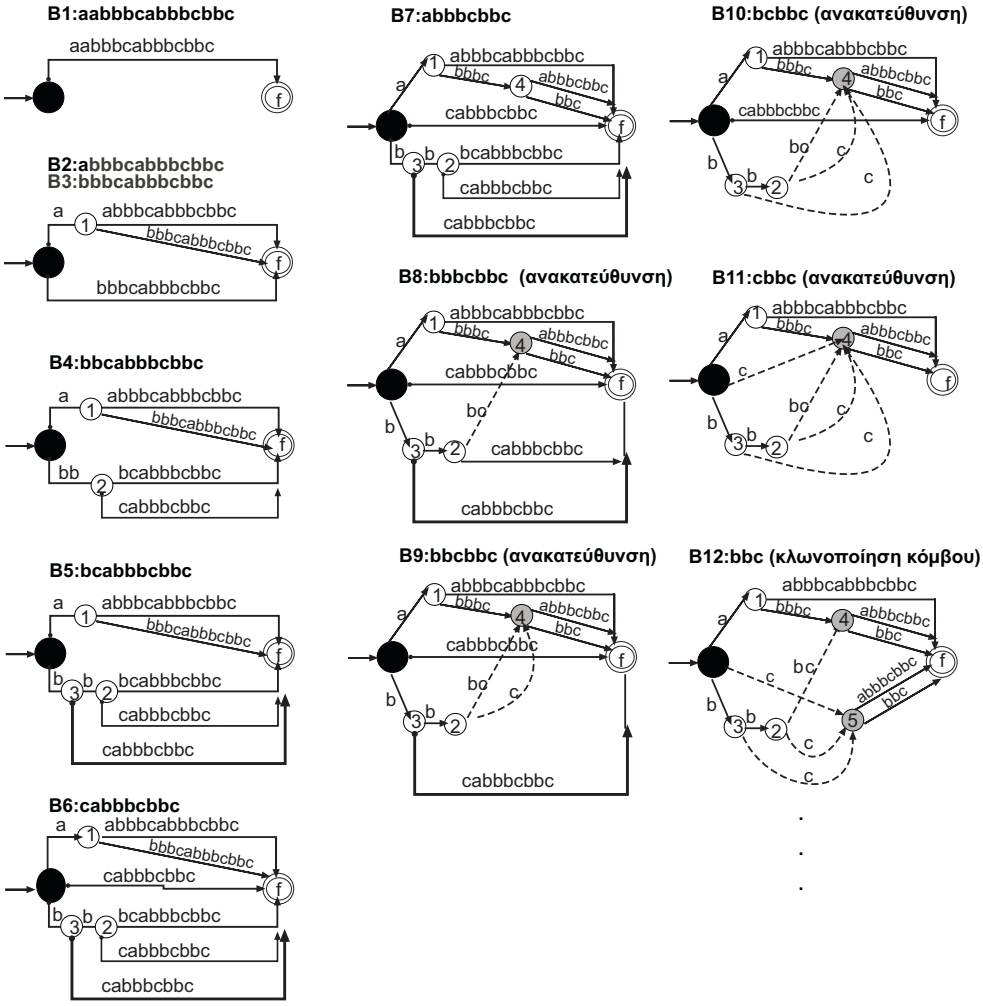


Σχήμα 2.19. Βήμα i αλγόριθμου κατασκευής cDAWG

Υπάρχουν δύο δραματικές διαφοροποιήσεις από τον αρχικό αλγόριθμο του McCreigt. Ενώ τα αδρά βήματα παραμένουν τα ίδια στο i-στο βήμα μπορούν να υπάρξουν επιπρόσθετα τα εξής δύο ενδεχόμενα:

→ Έστω ότι βρισκόμαστε στο βήμα i ενώ στο αμέσως προηγούμενο βήμα $i-1$ έχει δημιουργηθεί ένας νέος κόμβος. Η διαδικασία έχει κατέβει μέχρι το σημείο που τερματίζει το h_i και έστω ότι το σημείο τερματισμού είναι στο ενδιάμεσο της πλευράς. Σε αυτή την περίπτωση ότι η πλευρά ανακατευθύνεται προς τον κόμβο που δημιουργήθηκε στο προηγούμενο βήμα i και η ετικέτα μονοπατιού γίνεται το αριστερότερο μέρος της πλευράς που άνηκε στο μονοπάτι h_i (Βλ. το παράδειγμα του Σχήματος 2.20 όπου στο Βήμα 8 η πλευρά $2 \overset{bcabbbcbbc}{\rightarrow} f$ γίνεται $2 \overset{bc}{\rightarrow} 4$). Αφού στο βήμα i μόλις έχει δημιουργηθεί ένας νέος κόμβος u στο τέλος του μονοπατιού h_{i-1} τότε το $end\_pos(u)$ θα έχει ίσο με δύο με βάση τα επιθέματα που μόλις έχουν εισαχθεί. Το h_i το οποίο είναι μικρότερο ή ίσο με $|h_{i-1}-1|$ θα είναι ένα επίθεμα του h_{i-1}. Το μονοπάτι h_i σίγουρα έχει εισαχθεί από την πρώτη εμφάνιση του h_{i-1} συνεπώς αν στο τρέχον βήμα σταματήσει στο μέσο μιας πλευράς τότε αυτή η εμφάνιση αντιστοιχεί στην δεύτερη εμφάνιση του h_{i-1} που δημιούργησε τον κόμβο στο προηγούμενο βήμα. Συνεπώς στο τέλος του h_i αν σχηματίσουμε το κόμβο v' όπως ορίζει ο αλγόριθμος του McCreight αυτός ο κόμβος θα έχει το ίδιο σύνολο τερματικών θέσεων, δηλαδή $end_pos(u) = end_pos(v)$ οπότε και θα ενωθούν στο $cDAWG$. Η ένωση αυτή στην ουσία υλοποιείται με την ανακατεύθυνση την πλευράς. Αν στην συνέχεια και στα επόμενα βήματα συμβαίνει κάτι αντίστοιχό, δηλαδή τα επόμενα $h_i's$ τερματίζουν εντός πλευρών, τότε και αυτές ανακατευθύνονται προς τον κόμβο u αφού πάλι με την διάσπαση θα δημιουργούνται κόμβοι με τα ίδια σύνολα $end\_pos(.)$ (Βλ. παράδειγμα Σχήματος 2.20 Βήμα 9-11).

→ Οι ακμές του $cDAWG$ μπορούν να χαρακτηριστούν ως στέρεες ή μη (solid ή non-solid, αναλόγως με το αν ανήκουν στα μεγαλύτερα μονοπάτια από την ρίζα μέχρι και τον κόμβο που δείχνουν. Έστω εσωτερικός κόμβος u όπου υπάρχουν παραπάνω της μίας εισερχόμενες πλευρές τότε η πλευρά που εισέρχεται στο κόμβο και ανήκει στο μεγαλύτερο μονοπάτι, όσον αφορά το μήκος ετικέτας που αυτό αντιπροσωπεύει, καλείται στερεή ενώ οι υπόλοιπες που οδηγούν στον ίδιο κόμβο πρόκειται για συντομεύσεις που δημιουργούνται λόγω της συνένωσης ισομορφικών κόμβων. Έτσι συνοπτικά στέρεες καλούνται πλευρές $u \overset{l}{\rightarrow} v$ για τις οποίες ισχύει $|L(v)| = |L(u)| + |l|$ και οι οποίες είναι προγενέστερες των μη-στέρεων πλευρών που εισέρχονται σ-



Σχήμα 2.20. Βήματα κατασκευής του cDAWG για την συμβολοσειρά *aabbbcabbbcbbc*

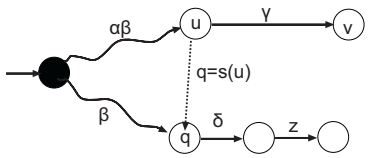
το κόμβο v. Ο διαχωρισμός αυτός των πλευρών μπορεί να γίνει με ένα *bit* πληροφορίας το οποίο για τις στέρεες θέτεται ίσο με 1 όταν ο κόμβος που δείχνουν μόλις δημιουργείται. Αν σε ένα βήμα i του παραπάνω αλγορίθμου καθώς κατευθυνόμαστε πάνω στο μονοπάτι h_i διασχίσουμε μη-στερεή πλευρά τότε ο κόμβος που δείχνει αυτή η πλευρά θα πρέπει να κλωνοποιηθεί (βλ. παράδειγμα Σχήματος 2.20 Βήμα 12 όπου ο κόμβος 4 κλωνοποιείται δημιουργώντας έτσι τον κόμβο 5). Αν διέλθει η διαδικασία πάνω από μία μη-στέρεα πλευρά προς ένα κόμβο u, αυτό σημαίνει ότι ενώ πριν το μονοπάτι της στέρεας πλευράς, με τα μονοπάτια των μη-στέρεων πλευρών εμφανίζονται στις ίδιες τερματικές θέσεις (για αυτό και δείχνουν τον ίδιο κόμβο από λογική συνένωση των κόμβων που έδειχναν αρχικά), τώρα αφού για το h_i διατρέχει τη μη -στέρεα πλευρά τότε το μονοπάτι που αντιστοιχεί σε αυτή έχει μια επιπρόσθετη τερματική θέση σε σχέση με την στερεά και τις υπόλοιπες μη-στερεές που δείχνουν τον ίδιο κόμβο. Έτσι αυτός ο κόμβος κλωνοποιείται και στον κλωνοποιημένο κόμβο u' αντιστοιχίζονται η μη-στερεή πλευρά που διατρέχθηκε και που αντιστοιχεί σε προηγούμενη εμφάνιση μαζί

με την εμφάνιση που συνδέεται με το h_i μαζί με τις υπόλοιπες μη-στέρεες πλευρές του u που αντιστοιχούν σε μονοπάτια με το νέο τερματικό σύνολο. Η μη-στερεή πλευρά που ανακατευθύνθηκε προς τον u' γίνεται πλέον στερεή αφού αντιστοιχεί στο προγενέστερο μεγαλύτερο μονοπάτι που οδηγεί στον u'.

Αν ο παραπάνω αλγόριθμός, όπως και στην περίπτωση της naive κατασκευής του δένδρου επιθεμάτων, αναζητεί το τέλος του μονοπατιού h_i ξεκινώντας από την ρίζα και συγκρίνοντας χαρακτήρα προς χαρακτήρα, τότε θα χρειαστεί χρόνο $O(|T|^2)$ για μια συμβολοσειρά T. Ο αλγόριθμος γίνεται γραμμικός με την χρήση των δεικτών επιθεμάτων (suffix links) που δεν ταυτίζονται όπως είδαμε και στην προηγούμενη ενότητα με τους δείκτες επιθεμάτων των δένδρων επιθεμάτων. Οι δείκτες επιθεμάτων για τα $cDAWG's$ ορίζονται ως εξής:

Ορισμός 2.34. *Σε κάθε κόμβο u, με ετικέτα μονοπατιού $L(u)$, του $cDAWG's$ ορίζεται ο* δείκτης επιθέματος *(suffix link) $s(u)$, ο δείκτης ο οποίος υποδεικνύει τον κόμβο q ο οποίος έχει ετικέτα μονοπατιού το μέγιστο επίθεμα του $L(u)$ για το οποίο θα έχουν διαφορετικά σύνολα τερματικών θέσεων με τον u, $end\_pos(u) \neq end\_pos(q)$.*

Με βάση αυτόν τον ορισμό παρατηρούμε ότι αν η θέση (locus) μιας συμβολοσειράς $\alpha\beta$, με $|\alpha||\beta| \geq 1$ είναι ο κόμβος u και $q = s(u)$ είναι η θέση της συμβολοσειράς β, τότε ο u είναι η θέση των επιθεμάτων της $\alpha\beta$ που τα μήκη τους είναι μεγαλύτερα από $|\beta|$. Αν ήταν μικρότερα ή ίσα θα είχαν σαν θέση τον κόμβο q.



Σχήμα 2.21. Χρήση των δεικτών επιθεμάτων του cDAWG

Έστω ότι είμαστε στο βήμα i και έχουμε εισάγει τα $i-1$ πρώτα επιθέματα. Έχουμε να ορίσουμε έναν δείκτη επιθεμάτων πάντα όταν δημιουργείται ένας νέος κόμβος και αυτό συμβαίνει είτε όταν σπάει στο τέλος του μονοπατιού h_i μια πλευρά είτε όταν κλωνοποιείται ένας κόμβος. Κατά την πρώτη περίπτωση έστω ότι το μονοπάτι h_i έχει τερματίσει στο μέσο μιας πλευράς η οποία διασπάται και δημιουργείται ένας νέος κόμβος v. Θα πρέπει να ορισθεί ο δείκτης $s(v)$. Έστω ότι ο u είναι ο πρόγονος του v και ότι ενώνονται με την πλευρά με ετικέτα γ (βλ. Σχήμα 2.21). Ακολουθείται ο δείκτης $q = s(u)$ και στην συνέχεια κατέρχεται στο υποδένδρο του q στο μονοπάτι που υπαγορεύεται από το γ. Κάτι τέτοιο συμβαίνει και με τον αλγόριθμο του McCreight. Για η διαδικασία καθόδου κάτω από το q γίνεται γρήγορα βρίσκοντας σε κάθε κόμβο την κατάλληλη εξερχόμενη πλευρά και καταναλώνοντας τόσους χαρακτήρες όσο και το μήκος αυτής. Αυτό μπορεί να γίνει αφού το γ έχει ενταχθεί κάτω από τον q σε προηγούμενο βήμα. Αν η κάθοδος αυτή σταματήσει σε κόμβο x τότε αυτός

είναι η θέση του επιθέματος $i+1$ και θέτεται $s(v) = x$. Το υπόλοιπο του επιθέματος εισάγεται όπως περιγράφηκε προηγουμένως.

Η άλλη περίπτωση, όπως είδαμε που μπορεί να υπάρξει, είναι να έχουμε στο βήμα i κλωνοποίηση ενός κόμβου. Έστω ότι ο κόμβος u κλωνοποιείται δημιουργώντας τον κόμβο u'. Τότε $s(u') = s(u)$ μιας και η ετικέτα μονοπατιού του $s(u')$ είναι το μεγαλύτερο επίθεμα της ετικέτας μονοπατιού του u' με διαφορετικό τερματικό σύνολο και αντίστοιχα για τον ίδιο λόγο $s(u) = u'$.

Από τα παραπάνω βλέπουμε ότι στην αρχή ενός βήματος έχουν ορισθεί όλοι οι δείκτες επιθεμάτων πλην ενός στη μία περίπτωση, έτσι το επαγωγικό βήμα μπορεί να ισχύσει. Όσον αφορά την χρονική πολυπλοκότητα του αλγορίθμου αυτή είναι γραμμική και αυτό μπορεί να αποδειχθεί με τα ίδια επιχειρήματα που χρησιμοποιήθηκαν για την απόδειξη της χρονικής πολυπλοκότητας του αλγόριθμου του McCreigth (βλ Παράγραφο 2.1.2. Συνεπώς :

Θεώρημα 2.35. *Το συμπαγές άκυκλο κατευθυνόμενο γράφημα $cDAWG_T$ μιας συμβολοσειράς T σχηματισμένης από το αλφάβητο $\Sigma = \{1, 2, \cdots, |T|\}$ κατασκευάζεται σε $O(|T|)$ χώρο και χρόνο.*

2.3.3 Εφαρμογές - Σχολιασμός

Με την χρήση των $DAWG's$ και των $cDAWG's$ μπορούν να επιλυθούν προβλήματα που απαντώνται και με την χρήση των δένδρων επιθεμάτων. Τα τελευταία συνήθως προτιμώνται επειδή και καταλαμβάνουν τον ίδιο ή και λιγότερο χώρο αλλά και εκφράζουν με καλύτερο τόπο τις θέσεις του κειμένου και τα κοινά προθέματα των επιθεμάτων. Αυτό έχει ως αποτέλεσμα την πιο εύκολη υλοποίηση bottom-up αλγορίθμους που χρησιμοποιούν πιο άμεσα την ιδιότητα ότι η ετικέτα μονοπατιού ενός εσωτερικού κόμβου εμφανίζεται μέσα στο κείμενο στις θέσεις που υποδεικνύουν τα φύλλα του.

3

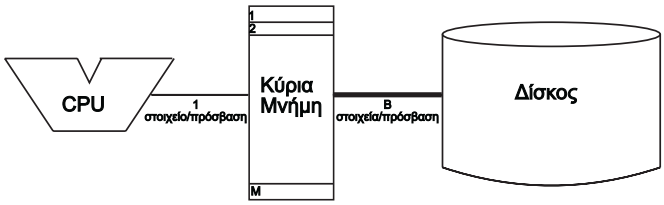
Δεικτοδότηση Συμβολοσειρών στη Δευτερεύουσα Μνήμη

Πολύ συχνά εφαρμογές έχουν να διαχειριστούν συμβολοσειρές οι οποίες είναι τόσο μεγάλες έτσι ώστε να μην χωρούν εξολοκλήρου στην κεντρική μνήμη κάποιου υπολογιστικού συστήματος. Ακόμη και στην περίπτωση που κάτι τέτοιο ήταν εφικτό, δεν θα ήταν αποδοτικό μιας και τα συνήθη υπολογιστικά συστήματα είναι πολυεπεξεργαστικά, δηλαδή εκτελούν παραπάνω από μια διεργασίες ταυτόχρονα κατανέμοντας την μνήμη τους σε όλες αυτές.

Αυτό έχει ως αποτέλεσμα τα δεδομένα και συγκεκριμένα στην περίπτωση μας οι συμβολοσειρές να φυλάσσονται σε δευτερεύοντα αποθηκευτικά μέσα όπως οι σκληροί δίσκοι, οι οπτικοί δίσκοι και οι μαγνητικές ταινίες. Οι δομές δεικτοδότησης πάνω σε αυτές τις συμβολοσειρές μπορεί είτε να κρατούνται μερικώς ή εξολοκλήρου στην κύρια μνήμη είτε όπως συμβαίνει συνήθως να αποθηκεύονται και αυτές σε δευτερεύοντα μέσα λόγω του μεγάλου μεγέθους τους. Αυτό το περιβάλλον λειτουργίας δημιουργεί ένα επιπρόσθετο πρόβλημα που οφείλεται στους μεγάλους χρόνους προσπέλασης των δεδομένων στα δευτερεύοντα μέσα που είναι έως και τρεις τάξεις μεγέθους μεγαλύτεροι από τους χρόνους προσπέλασης των δεδομένων στην κύρια μνήμη. Έτσι ο πραγματικός χρόνος κατασκευής μιας δομής δεικτοδότησης ή ο πραγματικός χρόνος απάντησης ενός ερωτήματος δεν εξαρτάται μόνο από της πλήθος των στοιχειωδών πράξεων που εκτελούνται στην μονάδα επεξεργασίας ενός συστήματος αλλά κυρίως καθορίζεται από το πλήθος των φορών που υπάρχει μια προσπέλαση (εγγραφή/ανάγνωση) στα αποθηκευτικά μέσα. Έτσι ο επιθυμητός στόχος είναι η κατάλληλη διάταξη τόσο των δεδομένων αλλά όσο και της δομής δεικτοδότησης στα δευτερεύοντα μέσα έτσι ώστε να ελαχιστοποιηθούν οι προσβάσεις σε αυτά κατά την εκτέλεση του αλγόριθμου κατασκευής της δομής αλλά και κατά την εκτέλεση ερωτημάτων.

3.1 I/O Μοντέλο

Πλέον, το πιο διαδεδομένο μέσο αποθήκευσης είναι οι σκληροί δίσκοι για αυτό και το μοντέλο που χρησιμοποιείται για την ανάλυση των αλγορίθμων στηρίζεται πάνω σε αυτούς. Η ακριβής μοντελοποίηση της συμπεριφοράς ενός συστήματος δίσκου είναι αρκετά σύνθετη μιας και εμπλέκονται πολλοί παράγοντες όπως η ταχύτητα περιστροφής, η ταχύτητα πρόσβασης καθώς και τα σημεία που τοποθετούνται τα δεδομένα ενός αρχείου [RW94].



Σχήμα 3.1. Μοντέλο I/O πρόσβασης στη δευτερεύουσα μνήμη

Στην βιβλιογραφία των αλγορίθμων της δευτερεύουσας μνήμης έχει επικρατήσει ένα απλό σχήμα δύο επιπέδων μνήμης, κύριας μνήμης και δίσκου (βλ. Σχήμα 3.1), το οποίο μοντελοποιεί την βασική ιδιότητα ενός δίσκου που είναι η επιμέριση του μεγάλου χρόνου πρόσβασης σε B στοιχεία. Δηλαδή σε κάθε πρόσβαση στο δίσκο δεν προσκομίζεται μόνο το στοιχείο που μας ενδιαφέρει αλλά και ολόκληρο το μπλοκ στο οποίο ανήκει. Θεωρούμε ότι η κύρια μνήμη έχει πεπερασμένο μέγεθος M, το πλήθος των στοιχείων του προβλήματος είναι N ενώ B είναι το πλήθος των στοιχείων που χωράει ένα μπλοκ του δίσκου. Συμπερασματικά στην μελέτη των αλγορίθμων μας ενδιαφέρει το αριθμός των προσβάσεων (I/Os) που γίνονται στο δίσκο για να επιλυθεί ένα πρόβλημα.

Ένα ερώτημα το οποίο μπορεί να προκύψει είναι: "ποια είναι ή η καλύτερη δυνατή πολυπλοκότητα σε I/Os ενός αλγόριθμου;". Για την απάντηση σε αυτό το ερώτημα ας θεωρήσουμε το απλό πρόβλημα που έχουμε N στοιχεία και θέλουμε να τα διατρέξουμε. Αν αυτά βρίσκονται τοποθετημένα σε γειτονικά μπλοκς του δίσκου τότε θα χρειαστούν το πολύ $\frac{N}{B}$ I/Os. Γενικότερα αν $C(N)$ είναι τα στοιχεία που αγγίζει μια πράξη στην κύρια μνήμη τότε το ιδανικό θα ήταν να εκτελεσθεί με $C(\frac{N}{B})$ I/Os στη δευτερεύουσα μνήμη.

3.2 Δένδρα Επιθεμάτων, Πίνακες Επιθεμάτων και Άκυκλοι Γράφοι Λέξεων στην Δευτερεύουσα Μνήμη

Παρόλο που οι δομές πλήρους δεικτοδότησης που εξετάσαμε στο Κεφάλαιο 2 είναι βέλτιστες όταν χρησιμοποιούνται στην κύρια μνήμη σε περίπτωση που εφαρμοστούν στην δευτερεύουσα μνήμη παρουσιάζουν φτωχή συμπεριφορά.

→ Τα **δένδρα επιθεμάτων** δεν παρουσιάζουν κανενός είδους ζύγιση, υψοζύγιση ή βαροζύγιση, μιας και οι εσωτερικοί κόμβοι δημιουργούνται ανάλογα με το πως επαναλαμβάνονται τμήματα της συμβολοσειράς. Η τοπολογία του δένδρου εξαρτάται από τα δεδομένα κάτι που δεν συμβαίνει σε άλλα ζυγισμένα δένδρα, όπως παραδείγματος χάρη στα AVL δένδρα. Αυτό έχει ως αποτέλεσμα να μην μπορούν να προβλεφθεί η τοπολογία του δένδρου και να τοποθετηθούν οι κόμβοι του δένδρου κατάλληλα στις σελίδες. Έτσι στην χειρότερη περίπτωση οι κόμβοι ενός μονοπατιού που μπορεί να ακολουθήσει ένα ερώτημα, π.χ. ερώτημα ταιριάσματος, μπορεί να βρίσκονται σε διαφορετικά μπλοκ του δίσκου άρα εκτελούνται $O(k)$ I/Os, όπου k το μήκος του μονοπατιού σε χαρακτήρες. Κάτι τέτοιο βρίσκεται μακριά από το επιθυμητό $C(\frac{k}{B})$ I/Os.
Επίσης οι κόμβοι των δένδρων επιθεμάτων έχουν μέγεθος $\Theta(|\Sigma|)$ που εξαρτάται από το μέγεθος του αλφάβητου. Αν το αλφάβητο είναι μεγαλύτερο

από B τότε δεν χωράει να αποθηκευτεί σε ένα μόνο μπλοκ του δίσκου. Αν ο κόμβος αποθηκευτεί απρόσεχτα στον δίσκο ίσος χρειαστούν $O(|\Sigma|)$ I/Os. Η καλύτερη δυνατή λύση θα ήταν ο κόμβος να αποθηκευτεί ως ένα B-δένδρο με αποτέλεσμα για την εύρεση την επιθυμητής εξερχόμενης πλευράς να χρειάζονται το που $O(\log_B |\Sigma|)$ I/Os.
Οι ακμές των δένδρων επιθεμάτων αποτελούνται από δύο ακέραιους που δείχνουν την τεμπέλα της ακμής μέσα στην συμβολοσειρά. Αυτό όπως είδαμε (βλ. Ενότητα 2.1) γίνεται για να έχουν σταθερό μέγεθος οι ακμές και άρα το δένδρο επιθεμάτων συνολικά να καταλαμβάνει γραμμικό χώρο. Η ταμπέλα μιας ακμής δεν έχει σταθερό μήκος με αποτέλεσμα να χρειαστούν επιπρόσθετα I/Os ώστε αυτή να προσκομιστεί από το δίσκο. Με βάση τα παραπάνω ένα ερώτημα ταιριάσματος θα χρειαζόταν $O(|P| \log_B |\Sigma|)$ I/Os ενώ η κατασκευή του δένδρου επιθεμάτων όπως την έχουμε δει θα χρειαστεί $O(|T| \log_B |\Sigma|)$ I/Os.

→ Επίσης και οι **πίνακες επιθεμάτων** παρουσιάζουν φτωχή συμπεριφορά στην δευτερεύουσα μνήμη. Το γεγονός ότι και οι δύο πίνακες από τους οποίους αποτελείται καταλαμβάνουν συνεχόμενο χώρο με αποτέλεσμα αυτό να γίνεται περιοριστικό. Ένα ερώτημα ταιριάσματος που όπως είδαμε (βλ. Ενότητα 2.2) πάνω στον ένα πίνακα θα χρειαστεί $O(\frac{|P|}{B} + \log_2 |T|)$ και η κατασκευή του $O(|T|)$ I/Os.

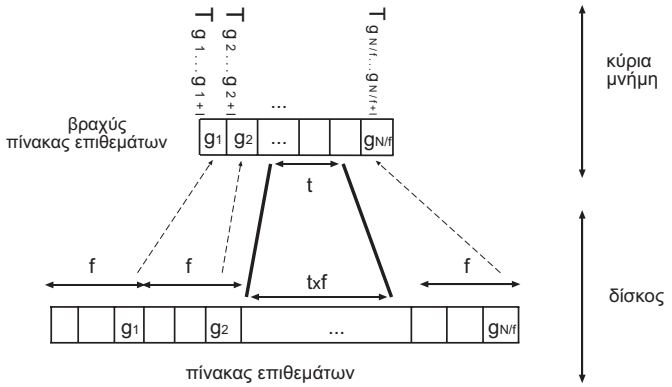
→ Οι **κατευθυνόμενοι άκυκλοι γράφοι λέξεων**, συμπαγείς ή μη, μοιράζονται της ίδιες αδυναμίες με τα δένδρα επιθεμάτων μιας και δομούνται με τον ίδιο τρόπο.

Για την δεικτοδότηση συμβολοσειρών στην δευτερεύουσα μνήμη έχουν σχεδιαστεί δομές οι οποίες είναι παραλλαγές των παραπάνω δομών και οι οποίες κινούνται σε δύο άξονες για να αντιμετωπίσουν τα μειονεκτήματα που είδαμε πως έχουν. Στη πρώτη κατεύθυνση χρησιμοποιούνται δύο επίπεδα για την δεικτοδότηση των συμβολοσειρών. Το ένα επίπεδο δεικτοδοτεί ένα δείγμα (sample) της πληροφορίας και κρατιέται στην κύρια μνήμη ενώ η δομή δεικτοδότησης για όλη την πληροφορία βρίσκεται εξολοκλήρου στην δευτερεύουσα μνήμη. Έτσι το πρώτο επίπεδο διευκολύνει στην πρόσβαση της πληροφορίας με λιγότερα I/Os. Σύμφωνα με την δεύτερη κατεύθυνση συνδυάζεται το B-δένδρο, το οποίο έχει καλή απόδοση στην δευτερεύουσα μνήμη με κάποια χαρακτηριστικά των κυρίων δομών δεικτοδότησης συμβολοσειρών.

3.3 Υπερ-Πίνακας Επιθεμάτων (Supra Suffix Array)

Στην [BYBZ96] προτείνεται ένα σχήμα δύο επιπέδων χρησιμοποιώντας τόσο την κύρια όσο και την δευτερεύουσα μνήμη. Στην δευτερεύουσα μνήμη αποθηκεύονται το κείμενο που δεικτοδοτείται καθώς και ο πίνακας επιθεμάτων αυτής σε συνεχόμενα μπλοκ. Τα στοιχεία του πίνακα επιθεμάτων χωρίζονται σε ομάδες μεγέθους f. Για κάθε τέτοια ομάδα διατηρείται ένα στοιχείο αυτής στην κύρια μνήμη, συνήθως είναι το τελευταίο στοιχείο της ομάδας, μαζί με τους l πρώτους χαρακτήρες του αντίστοιχου επιθέματος. Τα f και l είναι παράμετροι της δομής που όπως θα δούμε επηρεάζουν την συμπεριφορά της. Χρησιμοποιώντας αυτό το σχήμα δημιουργείται ένας μικρότερος πίνακας

επιθεμάτων ο οποίος θα έχει μέγεθος $\frac{N}{f} + \frac{N}{f}l$ και αποτελεί ένα υποσύνολο του κανονικού πίνακα επιθεμάτων. Ο πίνακας αυτός καλείται Βραχύς Πίνακας Επιθεμάτων (Short Suffix Array).



Σχήμα 3.2. Υπερ - Πίνακας Επιθεμάτων (Supra Suffix Array)

Με την χρήση του βραχύ πίνακα επιθεμάτων μπορούμε να περιορίσουμε το ερώτημα ταιριάσματος, δηλαδή την δυαδική αναζήτηση πάνω στο πίνακα επιθεμάτων, σε ένα υποσύνολο του χωρίς να εκτελεστεί κάποια πρόσβαση σε αυτόν που βρίσκεται στην δευτερεύουσα μνήμη. Με δύο δυαδικές αναζητήσεις στον βραχύ πίνακα επιθεμάτων βρίσκονται δύο ομάδες στοιχείων που περιέχουν το κατάλληλα επιθέματα έτσι ώστε $T_{SA_T[i]\cdots SA_T[i]+|l|} \leq_L P_{1\cdots l}$ και $P_{1\cdots l} \leq_L T_{SA_T[j]\cdots SA_T[j]+|l|}$ χωρίς κανένα I/O. Δηλαδή έχει περιοριστεί η αναζήτηση σε ένα διάστημα t στοιχείων του βραχύ πίνακα επιθεμάτων που αντιστοιχεί σε ένα διάστημα $t \times f$ στοιχείων του κανονικού πίνακα επιθεμάτων (βλ. Σχήμα 3.2).

Στην περίπτωση που $|P| \leq l$ τότε το επίθεμα, για το οποίο $T_{SA_T[i]\cdots SA_T[i]+|P|} \leq_L P$, και το επίθεμα, για το οποίο $P \leq_L T_{SA_T[j]\cdots SA_T[j]+|P|}$, θα βρίσκονται στην ομάδα του πρώτου και του τελευταίου από τα t στοιχεία αντίστοιχα. Έτσι γίνεται προσπέλαση σε αυτές τις ομάδες και με δύο δυαδικές αναζητήσεις καθορίζονται τα ακριβή αριστερά και δεξί όρια στον πίνακα επιθεμάτων και στη συνέχεια αναφέρονται όλα τα στοιχεία που βρίσκονται μεταξύ τους μιας και αυτά αποτελούν την απάντηση. Συνεπώς για αυτή τη διαδικασία εκτελούνται $2\log_2 f + \frac{\alpha}{B}\ I/Os$, όπου α το μέγεθος της απάντησης.

Στην περίπτωση που $|P| \geq l$ δεν είναι σίγουρο ότι τα ακριβή όρια της απάντησης θα βρίσκονται στην πρώτη και τελευταία ομάδα του διαστήματος που προσδιορίστηκε στο πρώτο βήμα. Έτσι εκτελούνται δύο επιπρόσθετες δυαδικές αναζητήσεις στο διάστημα των t στοιχείων του βραχύ πίνακα επιθεμάτων για να βρεθούν οι αριστερή και η δεξιά ομάδα που περιορίζει την απάντηση. Σε κάθε βήμα των δυαδικών αναζητήσεων χρειάζονται επιπρόσθετες προσβάσεις στο δίσκο για τις λεξικογραφικές συγκρίσεις πέρα των l πρώτων χαρακτήρων και γενικότερα στην χειρότερη περίπτωση θα χρειαστούν $\frac{|P|-l}{B}\ I/Os$ για κάθε βήμα. Συνεπώς συνολικά $2\log_2 t\frac{|P|-l}{B}\ I/Os$ εκτελούνται. Στην συνέχεια όπως και στην περίπτωση που $|P| \leq l$ εκτελούνται δυαδικές αναζητήσεις σε αυτές τις δύο ομάδες και προσδιορίζονται τα ακριβή όρια πάνω στον πίνακα επιθεμάτων.

Τελικά θα εκτελεστούν σε αυτή την περίπτωση $2\log_2 t\frac{|P|-l}{B} + 2\log_2 f + \frac{\alpha}{B}$ I/Os, όπου α το μέγεθος της απάντησης.

Βλέπουμε λοιπόν ότι στην χειρότερη περίπτωση θα εκτελεστούν $O(\log_2 t\frac{|P|-l}{B} + \log_2 f + \frac{\alpha}{B})$ I/Os. Με βάση αυτή την πολυπλοκότητα για τις προσβάσεις στο δίσκο επιθυμητή είναι μια μικρή τιμή για το f, μεγάλη τιμή για το l και τέλος έναν τρόπο για να περιοριστεί η απάντηση t του πρώτου βήματος αναζήτησης πάνω στον βραχύ πίνακα επιθεμάτων. Μικραίνοντας το f και αυξάνοντας το l μεγαλώνει και το μέγεθος του βραχύ πίνακα επιθεμάτων ο οποίος όπως είδαμε έχει μέγεθος $\frac{N}{f} + \frac{N}{f}l = \frac{N}{f}(l+1)$. Παρατηρήστε λοιπόν ότι υπάρχει συναλλαγή χώρου- πολυπλοκότητας I/Os αφού μεγάλο l και μικρό f προκαλεί μεγαλύτερες απαιτήσεις σε χώρο στην κύρια μνήμη. Έστω ότι το μέρος της κύριας μνήμης που μπορούμε να χρησιμοποιήσουμε για τον βραχύ πίνακα επιθεμάτων έχει μέγεθος M τότε $\frac{N}{f}(l+1) = M \Rightarrow f = \frac{N}{M}(l+1)$. Παρατηρούμε ότι τα f και l είναι ευθέως ανάλογα έτσι οι επιλογές αυτών για ελαχιστοποίηση της πολυπλοκότητας είναι αντικρουόμενες. Συνεπώς το ερώτημα είναι ποια τιμή θα επιλέξουμε για το l η οποία και κατά προέκταση θα καθορίσει το f για να ελαχιστοποιηθούν οι $O(\log_2 t\frac{|P|-l}{B} + \log_2 f + \frac{\alpha}{B}) = O(\log_2 t\frac{|P|-l}{B} + \log_2 \frac{N}{M}(l+1)$ προσβάσεις στο δίσκο συν τις προσβάσεις για την απάντηση. Επιλέγοντας ολοένα μεγαλύτερο l ναι μεν αυξάνεται ο δεύτερος όρος της παραπάνω πολυπλοκότητας όμως μειώνεται ο πρώτος αφού μειώνεται ο αριθμητής αλλά και η τιμή t. Χρησιμοποιώντας ολοένα μεγαλύτερα προθέματα των επιθεμάτων τότε είναι λιγότερο πιθανό αυτά τα προθέματα να είναι ίδια και άρα να επιστραφούν σε μια ενδεχόμενη απάντηση ως μια μεγάλη απάντηση με μέγεθος t. Βέβαια αυτό δεν μπορεί να λεχθεί αβίαστα αφού εξαρτάται από την εγγενή δομή της συμβολοσειράς και αν το αν έχει πολλά επαναλαμβανόμενα τμήματα αλλά και από το γεγονός μείωσης τους πρώτου όρου της πολυπλοκότητας δεν ξέρουμε αν αντισταθμίζεται από την αύξηση του δευτέρου. Ο προσδιορισμός του l είναι δύσκολο να επιλεγεί αναλυτικά και εκ των προτέρων παρά μόνο αν είναι γνωστά κάποια χαρακτηριστικά της συμβολοσειράς.

Στην [Nav96] δίνουν έναν τρόπο να επιλεγεί το l συλλέγοντας κάποια στατιστικά στοιχεία του κειμένου και θεωρώντας ότι το πρότυπα που θα αναζητηθούν έχουν και αυτά παρόμοια χαρακτηριστικά με την συμβολοσειρά. Ας θεωρηθεί ως p_l η πιθανότητα δύο συμβολοσειρές του ιδίου μήκους l από το κείμενο ή από το πρότυπο να είναι ίδιες. Ο πίνακας επιθεμάτων αντιστοιχεί σε ένα τυχαίο ομοιόμορφο δείγμα των l πρώτων χαρακτήρων των επιθεμάτων της συμβολοσειράς συνεπώς ο αναμενόμενος αριθμός από αυτές τις συμβολοσειρές που είναι ίδιος με ένα συγκεκριμένο ερώτημα θα είναι $N \times p_l$. Συνεπώς αυτά τα επιθέματα αναμένεται να συγκροτούν $t = \frac{Np_l}{f} = \frac{Mp_l}{l+1}$ ομάδες. Στην [Nav96] παραθέτονται αλγόριθμοι για τον υπολογισμό του p_l εξετάζοντας την συμβολοσειρά για κάθε δυνατό l έτσι ώστε να επιλεγεί η καλύτερη τιμή του l που ελαχιστοποιεί το t.

Στην [BYBZ96] όπου έγινε εκτενής πειραματική αξιολόγηση του υπερπίνακα διαστημάτων διαπιστώνεται ότι με την χρήση αυτού του σχήματος των δύο επιπέδων το κόστος αναζήτησης είναι κατά 10% μικρότερο από το να γινόταν η αναζήτηση απευθείας στον πίνακα επιθεμάτων στο δίσκο. Επίσης επιπλέον πλεονεκτήματα είναι ότι δεν χρησιμοποιεί σημαντικά επιπρόσθετη

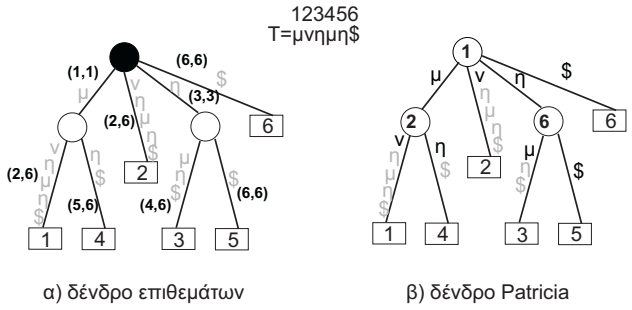
μνήμη για τον βραχύ πίνακα επιθεμάτων σε σχέση με το μέγεθος των δεδομένων που δεικτοδοτεί καθώς και ότι υλοποιείται εύκολα. Σε αντίθεση τα κύρια μειονεκτήματα της μεθόδου αυτής είναι η κατασκευή του πίνακα επιθεμάτων στην δευτερεύουσα μνήμη καθώς και η δυσκολία να χειριστεί της ενημερώσεις των δεδομένων που δεικτοδοτεί.

Η φύση του προβλήματος είναι στην ουσία η λεξικογραφική διάταξη όλων των επιθεμάτων μιας συμβολοσειράς. Στην εργασία [AFGV97] δείχνουν ότι το κόστος είναι τουλάχιστον ίδιο με το κόστος της ταξινόμησης ισάριθμων αντικειμένων στην δευτερεύουσα μνήμη. Έτσι δείχνουν ότι θα χρειαστούν τουλάχιστον $\Omega(\frac{|T|}{B}\log_{\frac{M}{B}}\frac{|T|}{B})$ I/Os ενώ προτείνουν και ένα αλγόριθμο που εκτελεί $O(\frac{|T|}{B}\log_{\frac{M}{B}}\frac{|T|}{B}\log_2|T|)$ προσβάσεις.

3.4 Συμπαγή Pat-δένδρα (Compact Pat-Trees)

Στην [CM96] προτείνεται ως δομή πλήρους δεικτοδότησης στη δευτερεύουσα μνήμη τα Compact Pat-Trees τα οποία στην ουσία είναι μια εναλλακτική συνοπτική αναπαράσταση των δένδρων επιθεμάτων. Ένα Compact Pat-Tree αποτελεί στην ουσία ένα Patricia Trie όπου τα κλειδιά είναι σε δυαδική κωδικοποίηση.

Το Patricia Trie είναι ένα συμπαγές ψηφιακό δένδρο αναζήτησης (compacted digital search tree) που περιέχει ως κλειδιά όλα τα επιθέματα μιας συμβολοσειράς T όπως και τα δένδρα επιθεμάτων (βλ. σχήμα 3.3β). Κάθε εσωτερικός κόμβος κρατάει τη αριστερότερη θέση (offset) της συμβολοσειράς που διαφοροποιούνται τα επιθέματα που βρίσκονται στα φύλλα του υποδένδρου ενώ κάθε πλευρά φυλάει τον πρώτο μόνο χαρακτήρα της ταμπέλας της. Για παράδειγμα στο σχήμα 3.3 τα επιθέματα $T_{1...6}, T_{4...6}$ διαφοροποιούνται στην θέση 2 ή στην θέση 5 συνεπώς ο εσωτερικός κόμβος κρατάει την τιμή 2. Με βάση αυτή την πληροφορία κατά την διάρκεια μιας αναζήτησης αν βρισκόμαστε σε έναν εσωτερικό κόμβο μπορεί να βρεθεί ποια εξερχόμενη πλευρά θα ακολουθηθεί με βάση τον χαρακτήρα που έχει. Στην συνέχεια γίνεται μετάβαση στο κόμβο που δείχνει η πλευρά αφού έχουν παραλειφθεί κάποιοι χαρακτήρες από το πρότυπο, τόσοι όσοι και οι υπόλοιποι χαρακτήρες που αντιστοιχούν στην πλευρά αλλά δεν έχουν αποθηκευτεί ρητά πάνω σε αυτή. Ο αριθμός αυτός μπορεί να υπολογιστεί από την θέση που έχει αποθηκευμένη ο κόμβος που δείχνει η πλευρά μείον την θέση του προγόνου του συν ένα. Με αυτό το τρόπο γίνεται μια χοντροκομμένη έρευνα από την ρίζα προς τα φύλλα. Αν η κάθοδος αποτύχει σημαίνει ότι δεν υπάρχει το πρότυπο στο κείμενο ενώ αν επιτύχει πρέπει να γίνει περαιτέρω διερεύνηση αφού δεν έχουν ταιριάξει όλοι οι χαρακτήρες του προτύπου. Έστω λοιπόν ότι αυτή η χοντροκομμένη έρευνα έχει τερματίσει σε έναν εσωτερικό κόμβο και έχουν καταναλωθεί όλοι οι χαρακτήρες του προτύπου τότε πρέπει να επιλεγεί ένα επίθεμα από το αντίστοιχο υποδένδρο που σταμάτησε η διαδικασία και να ελεγχθούν αν οι $|P|$ πρώτοι χαρακτήρες του ταιριάζουν με το πρότυπο. Αν όντως ταιριάζουν το πρότυπο υπάρχει στην συμβολοσειρά αλλιώς όχι. Η χρονική πολυπλοκότητα του ταιριάσματος είναι διπλάσια από ότι είναι στα δένδρα επιθεμάτων αφού εκτελούνται επιπρόσθετα P συγκρίσεις συνεπώς είναι της τάξης $O(|P|)$.



Σχήμα 3.3. δένδρο Patricia

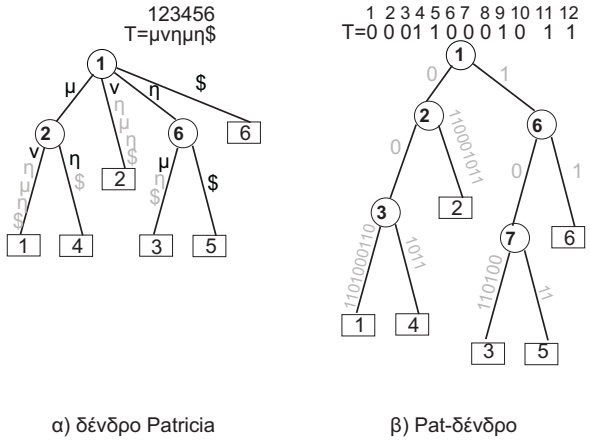
Τα Pat-trees πρόκειται για Patricia Tries όπου τα κλειδιά-επιθέματα έχουν δυαδική κωδικοποίηση. Κάθε σύμβολο του αλφάβητου κωδικοποιείται στον δυαδικό αριθμό που εκφράζει την θέση του μέσα στο αλφάβητο. Έτσι αν $\Sigma = \{\sigma_1\sigma_2\cdots\sigma_{|\Sigma|}\}$ τότε $\sigma_1 = 0\cdots00$ $\sigma_2 = 0\cdots01$ και ούτω καθεξής. Συνολικά χρειάζονται $\lceil \log_2 |\Sigma| \rceil$ *bits* για την κωδικοποίηση των χαρακτήρων. Έτσι για το παράδειγμα της συμβολοσειράς $T = \mu\nu\eta\mu\eta\$$ αν $\Sigma = \{\mu, \nu, \eta, \$\} = \{00, 01, 10, 11\}$ και άρα $T = 000110001011$. Χτίζοντας το Patricia Trie για την νέα κωδικοποιημένη συμβολοσειρά προκύπτει ένα δυαδικό δένδρο αφού πλέον η συμβολοσειρά αποτελείται από $\{0, 1\}$ (βλ. σχήμα 3.4β). Πλέον οι ακμές δεν είναι αναγκαίο να αποθηκεύουν κάτι αφού τα δύο σύμβολα μπορούν να αναπαρασταθούν ως το αριστερό και το δεξί παιδί ενός κόμβου. και συνεπώς αρκούν οι θέσεις πάνω στους κόμβους για τις διακλαδώσεις (branching) που προκαλούν τα επιθέματα. Το δένδρο που τελικά προκύπτει έχει $|T|$ φύλλα, $|T| - 1$ εσωτερικούς κόμβους και $2|T| - 2$ ακμές. Συνεπώς αν ένας εσωτερικός κόμβος αποτελείται από έναν ακέραιο και δύο δείκτες και τα φύλλα αποθηκεύουν τον δείκτη του επιθέματος τότε χρειάζονται $3(|T|-1)+|T| = 4|T|-3$ στοιχεία. Αυτή η χρησιμοποίηση χώρου είναι μεταξύ της χρησιμοποίησης χώρου από τους πίνακες επιθεμάτων και τις υλοποίησης των δένδρων επιθεμάτων που είδαμε στην Ενότητα 2.1.3. Η κατασκευή όμως μιας τέτοιας δομής είναι γραμμική κατασκευάζοντας αρχικά ένα δένδρο επιθεμάτων και στη συνέχεια μετασχηματίζεται στην παραπάνω δομή.

Για το ερώτημα ταιριάσματος το πρότυπο P μετασχηματίζεται χρησιμοποιώντας την ίδια δυαδική κωδικοποίηση του αλφαβήτου και στην συνέχεια με βάση αυτό το νέο πρότυπο ακολουθείται το μονοπάτι προς τα φύλλα. Ο χρόνος για την απάντηση του ερωτήματος θα είναι $O(|P| \log_2 |\Sigma|) = O(|P| + \alpha)$, όπου α το μέγεθος της απάντησης.

Στην [CM96] προτείνουν μια συμπαγή αναπαράσταση των Pat-trees αποθηκεύοντας με αποδοτικό τρόπο τα στοιχεία διακλάδωσης (την δομή του δένδρου δηλαδή), τα κλειδιά στους εσωτερικούς κόμβους και τα κλειδιά των επιθεμάτων στα φύλλα. Για την αποθήκευση της δομής των δένδρων χρησιμοποιούν μια υπονοούμενη (implicit) αναπαράσταση της μορφής: Ρίζα=

Επικεφαλίδα	Κωδικοποίηση Αριστερού Δένδρου	Κωδικοποίηση Δεξιού Δένδρου

όπου η επικεφαλίδα αποτελείται από ένα *bit* που υποδεικνύει ποιο από τα δύο υποδένδρα είναι το μεγαλύτερο και έναν ακέραιο i που εκφράζει το μέγεθος



Σχήμα 3.4. Pat-tree

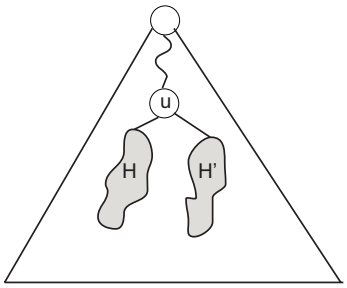
του μικρότερου. Οι κωδικοποιήσεις του αριστερού και του δεξιού υποδένδρου ορίζονται αναδρομικά με τον ίδιο τρόπο.

Για την ελαχιστοποίηση του αριθμού των προσβάσεων στο δίσκο κατά την διάρκεια ενός ερωτήματος ταιριάσματος προτείνουν έναν ειδικό τρόπο ομαδοποίησης των κόμβων του Compact Pat δένδρου. Κάθε τέτοια ομάδα κόμβων τοποθετείται σε ξεχωριστή σελίδα του δίσκου έτσι γίνεται προσπάθεια λοιπόν να συναντηθούν όσον το δυνατόν λιγότερες ομάδες κατά την διάρκεια ενός ερωτήματος. Η ομαδοποίηση γίνεται με $|T|$ άπληστες διεργασίες οι οποίες ξεκινούν από τα φύλλα και κατευθύνονται προς την ρίζα. Αρχικά κάθε διεργασία έχει σαν ομάδα το αντίστοιχο φύλλο και αυτή η όλοι οι κόμβοι μέσα στην ίδια ομάδα χαρακτηρίζονται με το ύψος $H = 1$. Καθώς οι διεργασίες ανηφορίζουν προς την ρίζα σε κάθε βήμα μόνο μία θα συνεχίσει με αποτέλεσμα σε κάθε κόμβο να εισέρχονται το πολύ δύο διεργασίες. Οι δύο αυτές διεργασίες συμπεριφέρονται με βάση τα ακόλουθα βήματα :

→ Αν και οι δύο διεργασίες έχουν το ίδιο ύψος H τότε:
  α′) Αν και οι δύο ομάδες των δύο διεργασιών μαζί με τον τρέχοντα κόμβο χωράνε σε μια σελίδα του δίσκου τότε ενώνονται σε μία ομάδα η οποία χαρακτηρίζεται με ίδιο ύψος $H' = H$
  β′) Αν και οι δύο ομάδες των δύο διεργασιών μαζί με τον τρέχοντα κόμβο *δεν* χωράνε σε μια σελίδα του δίσκου τότε κλείσε τις δύο αυτές ομάδες και ξεκίνα μια νέα ομάδα/διεργασία που να περιέχει τον τρέχοντα κόμβο με ύψος $H = H' + 1$

→ Αν και οι δύο διεργασίες *δεν* έχουν το ίδιο ύψος τότε: Κλείσε την ομάδα με το μικρότερο ύψος και:
  α′) Αν ο τρέχον κόμβος χωράει στην μεγαλύτερη ομάδα τοποθέτησε τον εκεί και χαρακτήρισε τον με το ίδιο ύψος $H' = H$
  β′) Αν ο τρέχον κόμβος *δεν* χωράει στην μεγαλύτερη ομάδα κλείσε και αυτή και ξεκίνα νέα διεργασία/ομάδα με τον νέο κόμβο και με ύψος $H' = H + 1$ όπου H το ύψος της μεγαλύτερης ομάδας

Η παραπάνω μέθοδος μπορεί να δημιουργήσει αρκετές μισογεμάτες σελίδες με αποτέλεσμα να υπάρχει σπατάλη χώρου. Για να αντιμετωπιστεί αυτό εξ-



Σχήμα 3.5. Ομαδοποίηση Κόμβων

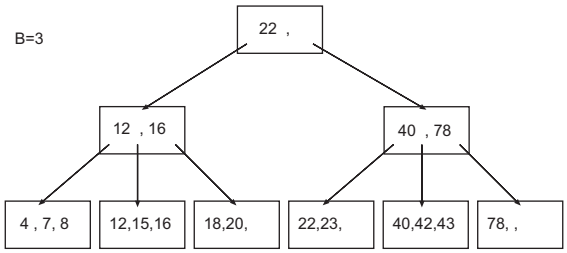
ετάζεται αν κάποιο από τα παιδιά ενός κόμβου μπορεί να συνενωθεί με την ομάδα του ενώ μια άλλη λύση θα ήτανε να παραλλαχθούν λίγο οι παραπάνω κανόνες. Από το παραπάνω σχήμα φαίνεται ότι γίνεται προσπάθεια να υπάρξουν όσο το δυνατόν μακρύτερα μονοπάτια από ένα φύλλο προς την ρίζα που να τοποθετούνται σε ένα μπλοκ. Στην μέση περίπτωση στην [Szp92] ο Szpankowski δείχνει ότι το ύψος του Pat-δένδρου αναμένεται να είναι λογαριθμικό. Συνεπώς με το παραπάνω σχήμα από το ύψος της δομής των μπλοκ θα γίνουν $O(\frac{H}{\sqrt{B}} + \log_B |T|)\ I/Os$, όπου H το ύψος του Pat-δένδρου, το οποίο αναμένεται λογαριθμικό. Πειραματικά η παραπάνω μέθοδος δείχθηκε ότι έχει σχετικά καλά αποτελέσματα ενώ η κύρια αδυναμία της είναι ο μεγάλος χρόνος κατασκευής της αφού απαιτεί $O(|T|)\ I/Os$.

3.5 B-δένδρο Προθεμάτων (Prefix B-Tree)

Στην εργασία [BU77] παρουσιάζεται μια παραλλαγή του κλασικού B-δένδρου για την δεικτοδότηση συμβολοσειρών στην δευτερεύουσα μνήμη. Το B-δένδρο είναι ένα ζυγισμένο δένδρο εύρεσης το οποίο δουλεύει ικανοποιητικά στη δευτερεύουσα μνήμη και έχει καθιερωθεί για δεικτοδότηση αντικειμένων σε αυτή.

Στα B-δένδρα με εξαίρεση την ρίζα, η οποία έχει βαθμό τουλάχιστον δύο, κάθε κόμβος έχει βαθμό από $\left\lceil \frac{B}{2} \right\rceil$ μέχρι B. Κάθε τέτοιος κόμβος, έστω με x ($\left\lceil \frac{B}{2} \right\rceil \leq x \leq B$) παιδιά, αποθηκεύει $x-1$ κλειδιά σε αύξουσα σειρά $k_1 \leq k_2 \leq \cdots \leq k_{x-1}$ και x δείκτες προς τα παιδιά του. Κάθε τέτοιος δείκτης αντιστοιχίζεται σε ένα από τα x διαστήματα που ορίζουν τα κλειδιά. Κάθε αντικείμενο d της δομής θα βρίσκεται στο υποδένδρο που αντιστοιχεί στο διάστημα $k_i \leq d < k_{i+1}$ (βλ. Σχήμα 3.6). Θεωρείται ότι ένας τέτοιος κόμβος χωράει να αποθηκευτεί σε μία σελίδα του δίσκο οπότε θα χρειαστεί μόνο ένα I/O για να διαβαστεί.

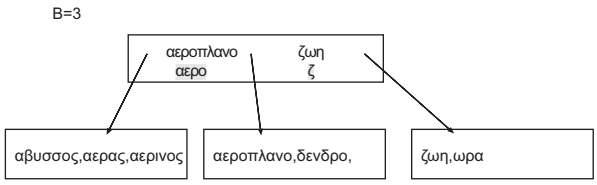
Για την εισαγωγή ή την αναζήτηση ενός νέου αντικειμένου d ξεκινώντας από την ρίζα, σε κάθε κόμβο ακολουθείται το υποδένδρο που αντιστοιχεί στο αντικείμενο. Μόλις η διαδικασία φτάσει στα φύλλα αυτό τοποθετείται στη σελίδα του δίσκου που αντιστοιχεί στο φύλλο. Αν αυτή η σελίδα είναι γεμάτη τότε σπάει στα δύο και τα αντικείμενα μοιράζονται. Χρειάζεται όμως έτσι να προστεθεί ένα διάστημα, δηλαδή ένα επιπλέον κλειδί και ένας δείκτης για τη νέα σελίδα, στο πατέρα των δύο σελίδων. Αν ο πατέρας είναι γεμάτος τότε σπάει



Σχήμα 3.6. B-δένδρο

και αυτός και στην χειρότερη περίπτωση οι διασπάσεις αυτές φτάνουν μέχρι και την ρίζα. Στην περίπτωση της διαγραφής ενός αντικειμένου αντιθέτως η σελίδα του φύλλου μπορεί μετά την διαγραφή να παραβιάζει το κάτω όριο του βαθμού του πατέρα. Σε αυτή την περίπτωση εξετάζονται ο αριστερός και ο δεξιός κόμβος του πατέρα. Όποιος από τους δύο έχει τουλάχιστον $\left\lceil \frac{B}{2} \right\rceil + 1$ παιδιά μοιράζεται τελικά ομοιόμορφα τα παιδιά του με το κόμβο που είχε υποχειλίσει. Η ενημέρωση των διαστημάτων είναι αναγκαία στους προγόνους και στην χειρότερη περίπτωση μπορεί να φτάσει μέχρι και την ρίζα. Το ύψος του B-δένδρου είναι $O(\log_B N)$, όπου N ο αριθμός των αντικειμένων συνεπώς κάθε μία από τις πράξεις θα χρειαστούν $O(\log_B N)$ I/Os. Για αναλυτικότερη παρουσίαση των B-δένδρων ανατρέξτε στο [CLR90].

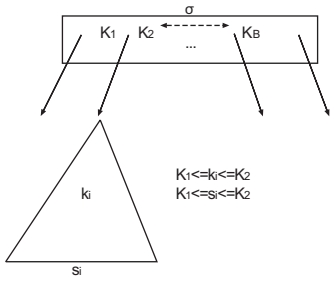
Στην [BU77] για να δεικτοδοτήσουν συμβολοσειρές στη δευτερεύουσα μνήμη προτείνουν να χρησιμοποιηθεί ένα B-δένδρο χρησιμοποιώντας ως κλειδιά στους εσωτερικούς κόμβους συμβολοσειρές και πιο συγκεκριμένα κοινά προθέματα των αντικειμένων που αποθηκεύει. Δύο σημαντικές διαφοροποιήσεις σε σχέση με το κλασικό B-δένδρο είναι ότι τα αντικείμενα πλέον δεν έχουν σταθερό μήκος, συνεπώς δεν μπορούμε να θεωρήσουμε ότι ένας κόμβος χωράει σε μία μόνο σελίδα του δίσκου, και επίσης ότι τα κλειδιά που χρησιμοποιούνται στους εσωτερικούς κόμβους σχηματίζουν διαστήματα πάνω στην λεξικογραφική διάταξη των αντικειμένων που έχουν εισαχθεί. Για παράδειγμα στο Σχήμα 3.7 ως κλειδιά στον πατέρα των τριών σελίδων/φύλλων που επιτρέπουν την σωστή διακλάδωση είναι οι λέξεις 'αεροπλανο' και 'ζωη'. Μικρότερες λεξικογραφικά λέξεις από το 'αεροπλάνο' θα βρίσκονται στο αριστερό υποδένδρο, μεγαλύτερες ή ίσες από αυτό και μικρότερες από τη 'ζωη' θα βρίσκονται στο μεσαίο υποδένδρο ενώ μεγαλύτερες ή ίσες λεξικογραφικά με τη λέξη 'ζωή' θα βρίσκονται στο δεξιό υποδένδρο.



Σχήμα 3.7. B-δένδρο προθεμάτων (απλοϊκή μορφή)

Παρατηρούμε ότι για την διακλάδωση δεν είναι αναγκαία η αποθήκευση στους εσωτερικούς κόμβους ολόκληρων των συμβολοσειρών αλλά το μέγισ-

το πρόθεμα αυτών που επιτρέπει τη σωστή διακλάδωση. Στο προηγούμενο παράδειγμα ως κλειδιά μπορούν να χρησιμοποιηθούν τα 'αερο' και 'ζ'. Αυτή η βελτίωση ναι μεν μειώνει την χρησιμοποίηση του χώρου όμως δεν αναιρεί το σοβαρό μειονέκτημα ότι οι κόμβοι δεν έχουν σταθερό μήκος για να είναι εγγυημένο ότι θα χωρέσει σε μία ή το πολύ σε ένα σταθερό αριθμό σελίδων του δίσκου. Τέλος μια επιπλέον βελτίωση η οποία μπορεί να εφαρμοστεί είναι όταν υπερχειλίσει ένας κόμβος αυτός να μην σπάει ακριβώς στη μέση αλλά σε ένα διάστημα γύρω από το κεντρικό κλειδί έτσι ώστε να επιλεγεί από το διάστημα αυτό το κλειδί που τοποθετεί το μικρότερο (σε αριθμό χαρακτήρων) κλειδί στον πατέρα των κόμβων της διάσπασης. Για παράδειγμα στο Σχήμα 3.7 αν υπερχειλίσει το αριστερότερο φύλλο είναι προτιμότερο μετά τη διάσπαση να πάνε στην ίδια σελίδα ο '*αερας*' με το '*αερινος*' αφού θα χρησιμοποιηθεί ως κλειδί το 'αβ' ενώ σε αντίθετη περίπτωση ως κλειδί θα χρησιμοποιηθεί το 'αερι'. Αυτή η βελτίωση ναι μεν βοηθά στο να μικρύνει το μέγεθος των κόμβων έτσι ώστε να προσπελαστούν με σταθερό αριθμό προσβάσεων στο δίσκο μπορεί να οδηγήσει σε φτωχή εκμετάλλευση του χώρου του δίσκου αφού μπορεί να υπάρξουν πολλές μισογεμάτες σελίδες αυτού.



Σχήμα 3.8. B-δένδρο προθεμάτων

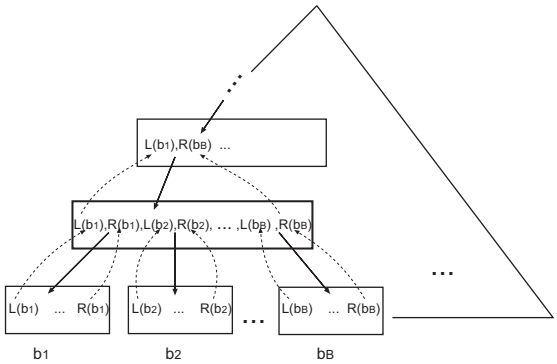
Έστω ότι σε έναν εσωτερικό κόμβο δύο γειτονικά κλειδιά μοιράζονται ένα κοινό πρόθεμα. Τότε σε ολόκληρο το υποδένδρο του διαστήματος που ορίζουν τα δύο αυτά κλειδιά οι συμβολοσειρές si αλλά και τα κλειδιά k_i μέσα σε αυτό μοιράζονται σίγουρα αυτό το ίδιο κοινό πρόθεμα. Έτσι η σαφής αποθήκευση αυτού του κοινού προθέματος σε όλους τους κόμβους είναι πλεονασμός. Το κοινό αυτό πρόθεμα λοιπόν αποθηκεύεται μια φορά στην ρίζα του υποδένδρου ενώ μέσα στο υποδένδρο αποθηκεύονται τα υπόλοιπα των διαχωριστών. Χρησιμοποιώντας και αυτή τη μείωση χρήσης αποθηκευτικού χώρου όμως πάλι δεν υπάρχει εγγύηση για το μέγεθος των κόμβων και το πλήθος των σελίδων του δίσκου που ο καθένας καταλαμβάνει.

Προσπαθώντας να επιλύσουμε το πρόβλημα του ταιριάσματος για μια συμβολοσειρά T χρησιμοποιώντας το B-δένδρο προθεμάτων (prefix B-tree) θα πρέπει να εισάγουμε όλα τα επιθέματα της συμβολοσειράς στη δομή. Για την εύρεση ενός προτύπου γίνεται αναζήτηση ξεκινώντας από την ρίζα προς τα φύλλα. Για να διατρέξουμε το δένδρο από τη ρίζα προς τα φύλλα σίγουρα θα χρειαστούν $O(\log_B |T|)$ I/Os, όσο και το ύψος της δομής. Επίσης σε κάθε εσωτερικό κόμβο για την εύρεση του κατάλληλης εξερχόμενης πλευράς θα χρειαστούν με χρήση δυαδικής αναζήτησης $O(\log_2 B)$ I/Os ενώ στην χειρότερη περίπτωση

που τύχει οι διαχωριστές στο μονοπάτι αυτό να έχουν μήκος μεγαλύτερο από το μήκος του προτύπου θα χρειαστεί να φορτωθούν από το δίσκο επιπλέον $\frac{|P|}{B}$ σελίδες. Έτσι θα χρειαστούν το πολύ $O(\frac{|P|}{B}\log_B|T|\log_2 B) = O(\frac{|P|}{B}\log_B|T|)$ I/Os για την εύρεση των δύο ακραίων επιθεμάτων της T με πρόθεμα το P και άρα τις εμφανίσεις του προτύπου που θα αντιστοιχούν στα κλειδιά ενδιάμεσα των δύο αυτών κλειδιών.

3.6 B-δένδρο Συμβολοσειρών (String B-Tree)

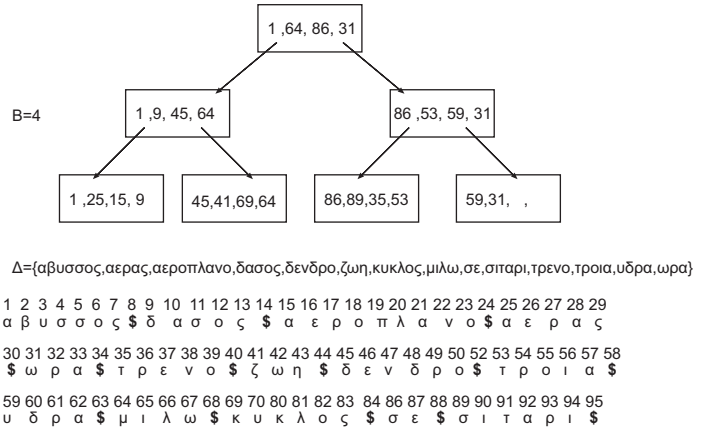
Το B-δένδρο συμβολοσειρών (String B-Tree) παρουσιάστηκε στην εργασία [FG99] και αποτελεί μια παραλλαγή του B-δένδρου προθεμάτων (Prefix B-tree) που είδαμε στην προηγούμενη ενότητα η οποία δεν εμφανίζει τις αδυναμίες που παρατηρήσαμε. Η κύρια δομή που χρησιμοποιείται για την δεικτοδότηση των συμβολοσειρών στη δευτερεύουσα μνήμη είναι και σε αυτή την περίπτωση το B-δένδρο, όμως οι διαχωριστές στους εσωτερικούς κόμβους δεν φυλάσσονται ρητώς ως συμβολοσειρές αλλά οργανώνονται σε δένδρα Patricia, τα οποία είδαμε στην ενότητα 3.4. Με τον τρόπο αυτό επιτυγχάνεται οι διαχωριστές στους εσωτερικούς κόμβους να μην έχουν αυθαίρετο μήκος αλλά να καταλαμβάνουν σταθερό χώρο. Επίσης είναι δυνατόν να επιλεγεί το επιθυμητό υποδένδρο κάθε εσωτερικού κόμβου με μικρότερο αριθμό I/Os. Αυτά τα στοιχεία κάνουν την δομή ισχυρότερη στις πράξεις αναζήτησης αλλά και στο συνολικό χώρο που καταλαμβάνει.



Σχήμα 3.9. B-δένδρο συμβολοσειρών (επιλογή διαχωριστών)

Θεωρούμε ότι οι συμβολοσειρές βρίσκονται τοποθετημένες στο δίσκο και διαχωρίζονται από έναν ειδικό χαρακτήρα (έστω αυτός να είναι ο χαρακτήρας \$), ο οποίος δεν ανήκει στον αλφάβητο των συμβολοσειρών. Η πληροφορία που αποθηκεύει το B-δένδρο συμβολοσειρών στα φύλλα του είναι οι θέσεις έναρξης των συμβολοσειρών στον δίσκο. Θα τοποθετηθούν με τέτοιο τρόπο στα φύλλα έτσι ώστε να υπάρχει αύξουσα λεξικογραφική διάταξη των αντίστοιχων συμβολοσειρών (βλ. Σχήμα 3.10). Αυτοί οι δείκτες των συμβολοσειρών τοποθετούνται στις σελίδες του δίσκου. Σε κάθε εσωτερικό κόμβο χρησιμοποιούνται ως διαχωριστές που ορίζουν το διάστημα του κάθε υποδένδρου το πιο αριστερό στοιχείο και το πιο δεξιό στοιχείο του κάθε παιδιού αντίστοιχα. Έτσι λοιπόν αν

ένας εσωτερικός κόμβος έχει k παιδιά c_i $(1 \leq i \leq k)$ τότε το σύνολο των διαχωριστών του θα είναι το $K = \{L(c_1), R(c_1), L(c_2), R(c_2), \cdots L(c_k), R(c_k)\}$ όπου $L(c_i), R(c_i)$ ο αριστερότερος και ο δεξιότερος διαχωριστής του παιδιού c_i (βλ. Σχήμα 3.9).



Σχήμα 3.10. B-δένδρο συμβολοσειρών (απλή μορφή)

Η παραπάνω μορφή του B-δένδρου Συμβολοσειρών δεν διαφοροποιείται ουσιαστικά από το B-δένδρο Προθεμάτων αφού όπως θα δούμε το ταίριασμα προτύπου επιλύεται στον ίδιο χρόνο. Η διαφοροποίηση που έχουν, εγγυάται στο γεγονός ότι όλοι οι κόμβοι έχουν αποκτήσει σταθερό μέγεθος αφού οι διαχωριστές πλέον δεν είναι συμβολοσειρές αυθαίρετου μήκους αλλά οι θέσεις έναρξης αυτών των συμβολοσειρών στο δίσκο. Το B-δένδρο Συμβολοσειρών σε αυτή του τη μορφή, όπως και το B-δένδρο Προθεμάτων, δύναται να απαντήσει ερωτήματα προθεμάτων, δηλαδή βρίσκει τις συμβολοσειρές που έχουν ως πρόθεμα το ερώτημα-συμβολοσειρά P αφού σε κάθε εσωτερικό κόμβο διακλαδίζεται ανάλογα με τους διαχωριστές που έχουν το ερώτημα ως πρόθεμα. Για να επιλυθεί το πρόβλημα του ταιριάσματος προτύπου ως σε μια συμβολοσειρά T θα πρέπει να εισαχθούν στην δομή ως κλειδιά όλα τα επιθέματα της T. Ο συνολικός αριθμός προσβάσεων στο δίσκο θα είναι $O(\frac{|P|}{B} \log_B |T| \log_2 B) = O(\frac{|P|}{B} \log_B |T|)$ I/Os αφού χρειάζονται $O(\log_2 B)$ προσβάσεις σε κάθε εσωτερικό κόμβο για να βρεθεί το κατάλληλο υποδένδρο χρησιμοποιώντας δυική αναζήτηση. Σε κάθε βήμα της δυαδικής αναζήτησης θα χρειαστεί να φορτωθεί από το δίσκο η συμβολοσειρά του διαχωριστή οπότε το πολύ να χρειαστούν $O(\frac{|P|}{B})$ προσβάσεις. Αυτή η διαδικασία γίνεται σε κάθε εσωτερικό κόμβο κινούμενοι από την ρίζα προς τα φύλλα συνεπώς επαναλαμβάνεται $O(\log_B |T|)$ φορές.

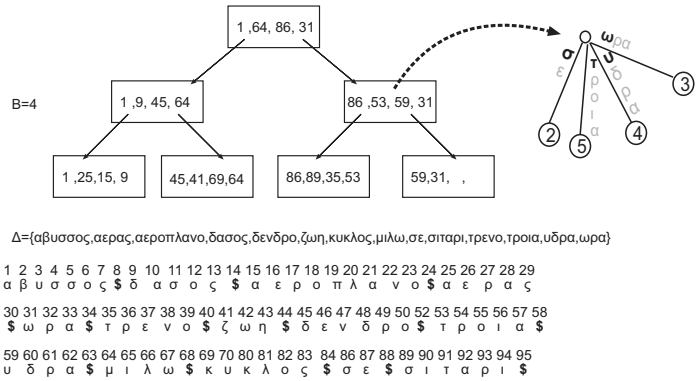
Στο B-δένδρο Συμβολοσειρών στην τελική του μορφή οι συγγραφείς προτείνουν τα στοιχεία σε κάθε σελίδα είτε αυτή ανήκει στα φύλλα είτε ανήκει σε εσωτερικό κόμβο να οργανωθεί σε ένα δένδρο Patricia. Τα δένδρα Patricia, όπως είδαμε και στην Ενότητα 3.4, είναι συμπαγή ψηφιακά δένδρα αναζήτησης (compacted digital search tree) που περιέχουν ως κλειδιά όλα τα επιθέματα μιας συμβολοσειρά T όπως ακριβώς και τα δένδρα επιθεμάτων. Κάθε εσωτερικός κόμβος κρατάει τη αριστερότερη θέση (offset) της συμβολοσειράς

που διαφοροποιούνται τα επιθέματα που βρίσκονται στα φύλλα του υποδένδρου ενώ κάθε πλευρά φυλάει τον πρώτο μόνο χαρακτήρα της ταμπέλας της. Στην ουσία τα δένδρα Patricia είναι δένδρα επιθεμάτων που σε κάθε πλευρά αποθηκεύεται μόνο ο πρώτος χαρακτήρας για να είναι δυνατή η διακλάδωση αλλά και το μήκος της πλευράς ώστε να να είναι δυνατός ο έλεγχος των σωστών χαρακτήρων της συμβολοσειράς στον επόμενο εσωτερικό κόμβο.

Με αυτή την πληροφορία κατά την διάρκεια μιας αναζήτησης αν βρισκόμαστε σε έναν εσωτερικό κόμβο μπορεί να βρεθεί ποια εξερχόμενη πλευρά θα ακολουθηθεί με βάση τον χαρακτήρα που έχει. Στην συνέχεια γίνεται μετάβαση στο κόμβο που δείχνει η πλευρά αφού έχουν παραλειφθεί κάποιοι χαρακτήρες από το πρότυπο, τόσοι όσο και οι υπόλοιποι χαρακτήρες που αντιστοιχούν στην πλευρά αλλά δεν έχουν αποθηκευτεί ρητά πάνω σε αυτή. Ο αριθμός αυτός μπορεί να υπολογιστεί από την θέση που έχει αποθηκευμένη ο κόμβος που δείχνει η πλευρά μείον την θέση του προγόνου του συν ένα. Με αυτό το τρόπο γίνεται μια χοντροκομμένη έρευνα από την ρίζα προς τα φύλλα. Αν η κάθοδος αποτύχει σημαίνει ότι δεν υπάρχει το πρότυπο στο κείμενο ενώ αν επιτύχει πρέπει να γίνει περαιτέρω διερεύνηση αφού δεν έχουν ταιριάξει όλοι οι χαρακτήρες του προτύπου. Έστω λοιπόν ότι αυτή η χοντροκομμένη έρευνα έχει τερματίσει σε έναν εσωτερικό κόμβο και έχουν καταναλωθεί όλοι οι χαρακτήρες του προτύπου τότε πρέπει να επιλεγεί ένα επίθεμα από το αντίστοιχο υποδένδρο που σταμάτησε η διαδικασία και να ελεγχθούν αν οι $|P|$ πρώτοι χαρακτήρες του ταιριάζουν με το πρότυπο. Αν όντως ταιριάζουν το πρότυπο υπάρχει στην συμβολοσειρά αλλιώς όχι. Η χρονική πολυπλοκότητα του ταιριάσματος είναι διπλάσια από ότι είναι στα δένδρα επιθεμάτων αφού εκτελούνται επιπρόσθετα P συγκρίσεις συνεπώς είναι της τάξης $O(|P|)$.

Αν κάθε κόμβος του B-δένδρου Συμβολοσειρών οργανωθεί σε ένα δένδρο Patricia (βλ. Σχήμα 3.11) τότε ελαττώνονται οι προσβάσεις στο δίσκο αφού πλέον δεν χρειάζεται να εκτελεστεί δυαδική αναζήτηση για την επιλογή του κατάλληλου υποδένδρου κατά την οποία σε κάθε βήμα της έχουμε κάποιες προσβάσεις στο δίσκο. Ένας εσωτερικός κόμβος με το πολύ β παιδιά χρειάζεται χώρο το πολύ 5β στοιχεία, 4β στοιχεία για την αναπαράσταση του Patricia δένδρου και β δείκτες στα παιδιά του στο B-δένδρο. Συνεπώς το β μπορεί να επιλεγεί κατάλληλα έτσι ώστε κάθε κόμβος του B-δένδρου να χωράει εξολοκλήρου σε μια σελίδα του δίσκου. Έτσι η πράξη της αναζήτησης σε κάθε εσωτερικό κόμβο θα εκτελεί ένα I/O για να φορτώσει ολόκληρο το κόμβο και στη συνέχεια με βάση το ερώτημα P θα εκτελεί το χοντροκομμένο ερώτημα πάνω στο Patricia δένδρο του κόμβου.

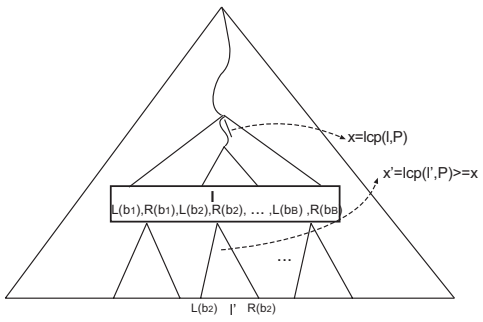
Η αναζήτηση αυτή τερματίζει είτε σε έναν εσωτερικό κόμβο μην έχοντας τη δυνατότητα παραπέρα διακλάδωσης ή έχοντας χρησιμοποιήσει όλους τους χαρακτήρες του προτύπου είτε έχει καταλήξει σε ένα φύλλο. Ας συμβολίσουμε ως l ένα οποιοδήποτε φύλλο που βρίσκεται στο υποδένδρο του κόμβου της πρώτης περίπτωσης ή το φύλλο που έχει τερματίσει η αναζήτηση στη δεύτερη περίπτωση. Στην συνέχεια η συμβολοσειρά του διαχωριστή που αντιστοιχεί στο l φορτώνεται εκτελώντας $O(\frac{|P|}{B})$ I/Os για συγκριθεί χαρακτήρα προς χαρακτήρα με το ερώτημα P. Με αυτή την ενέργεια προσδιορίζεται το μέγιστο κοινό πρόθεμα του προτύπου με το l καθώς και ο βαθύτερος πρόγονος στο Patricia δένδρο που αναπαριστά αυτό το κοινό πρόθεμα. Αυτός ο κόμβος, τον



Σχήμα 3.11. Β-δένδρο συμβολοσειρών

οποίο ας τον καλέσουμε κρίσιμο κόμβο, έχει την σημαντική ιδιότητα ότι όλα τα κλειδιά-συμβολοσειρές που βρίσκονται στα υποδένδρα που καλύπτει θα μοιράζονται το ίδιο μέγιστο κοινό πρόθεμα με το ερώτημα. Αυτό ισχύει επειδή όλα τα ενδιάμεσα κλειδιά-συμβολοσειρές περιορίζονται λεξικογραφικά από τους αντιπροσώπους/διαχωριστές σε κάποιο εσωτερικό κόμβο του Β-δένδρου Συμβολοσειρών. Αυτό η ιδιότητα είναι ταυτόσημη με την Ιδιότητα 2.22 των πινάκων επιθεμάτων. Έτσι λοιπόν αν ψάχνουμε και το λεξικογραφικά μικρότερο άκρο της απάντησης ακολουθείται το αριστερότερο φύλλο του κρίσιμου κόμβου και άρα το αντίστοιχο υποδένδρο στο Β-δένδρο Συμβολοσειρών αλλιώς για το μεγαλύτερο λεξικογραφικά άκρο της απάντησης το δεξιότερο υποδένδρο κάτω από τον κρίσιμο κόμβο. Τα παραπάνω συνοψίζονται στην ακόλουθη ιδιότητα.

Ιδιότητα 3.1. Σε κάθε εσωτερικό κόμβο του Β-δένδρου Συμβολοσειρών κάθε εσωτερικός κόμβος του αντίστοιχου Patricia δένδρου αναπαριστά το ελάχιστο κοινό πρόθεμα που μοιράζονται όλα τα κλειδιά/συμβολοσειρές που καλύπτει στο υποδένδρο του.



Σχήμα 3.12. Πράξη Ερώτησης στο Β-δένδρο συμβολοσειρών

Με την χρήση των Patricia δένδρων στους εσωτερικούς κόμβους βλέπουμε λοιπόν ότι εξαλείφεται πλήρως η δυαδική αναζήτηση και επιλέγεται το κατάλληλο υποδένδρο μόνο με $O(\frac{|P|}{B})$ I/Os. Η συνολική ασυμπτωτική πολυπλοκότητα των προσβάσεων στο δίσκο όμως κινούμενοι από την ρίζα στα φύλλα παραμένει $O(\frac{|P|}{B}\log_B|T|)$ I/Os.

Με μια προσεχτική παρατήρηση της Ιδιότητας 3.1 μπορεί να γίνει αντιληπτό ότι είναι δυνατό να εξοικονομηθούν προσβάσεις στο δίσκο αφού μεταβαίνοντας από έναν κόμβο στο παιδί του το μέγιστο κοινό πρόθεμα που μπορεί να έχει το ερώτημα με έναν από τους διαχωριστές είναι τουλάχιστον ίσο με το μέγιστο κοινό πρόθεμα που παρατηρήθηκε στον πατέρα. Έτσι σε έναν εσωτερικό κόμβο του B-δένδρου συμβολοσειρών έστω ότι έχει υπολογισθεί το μέγιστο κοινό πρόθεμα του ερωτήματος με τον επιλεγμένο από την αναζήτηση στο Patricia δένδρο φύλλο/διαχωριστή l, $x = lcp(l, P)$. Τότε για το επιλεγμένο παιδί του η ίδια διαδικασία θα υπολογίσει πάλι ένα μέγιστο κοινό πρόθεμα το οποίο είναι τουλάχιστον ίδιο με το προηγούμενο $x' = lcp(l', P) \geq x$ (βλ. Σχήμα 3.12). Για τον υπολογισμό του x' δεν είναι αναγκαίο να φορτωθούν από το δίσκο οι πρώτοι του $|x|$ χαρακτήρες αλλά οι υπόλοιποι $g = |x'| - |x|$ εκτελώντας $\frac{|x'|-|x|}{B} + 1$ I/Os. Έτσι καθ' όλη τη μετάβαση από την ρίζα του B-δένδρου Συμβολοσειρών προς τα φύλλα για την διακλάδωση ανάλογα με το έλεγχο του κοινού προθέματος του ερωτήματος με τους διαχωριστές θα φορτωθούν το πολύ $\sum g_s \leq P$ χαρακτήρες αφού καθώς κατεβαίνει το ερώτημα τα επίπεδα φορτώνει και ένα υπόλοιπο του προθέματος μέχρι το κοινό πρόθεμα να γίνει το πολύ ίσο με το ερώτημα. Τέλος για την επέκταση του κοινού αυτού προθέματος εκτελείται τουλάχιστον ένα I/O για τον έλεγχο αν υπάρχει προέκταση. Άρα συνολικά θα χρειαστούν $O(\frac{P}{B} + \log_B |T|)$ I/Os για την ολοκλήρωση της μετάβασης από την ρίζα στα φύλλα. Συνεπώς παρατηρούμε ότι το B-δένδρο δύναται να απαντήσει με μικρότερο αριθμό προσβάσεων στη δευτερεύουσα μνήμη, τα προβλήματα εύρεσης κοινού προθέματος και προτύπου σε σχέση με τις άλλες δομές δεικτοδότησης που είδαμε στο κεφάλαιο αυτό.

Όταν το B-δένδρο συμβολοσειρών δεικτοδοτεί ένα σύνολο Δ από συμβολοσειρές, όπως είδαμε προηγουμένως, έχει την δυνατότητα να απαντά ερωτήματα εύρεσης μεγίστου κοινού προγόνου. Για την εισαγωγή ή την διαγραφή ενός κλειδιού/συμβολοσειράς T' εκτελείται αρχικά ένα ερώτημα εύρεσης της συμβολοσειράς αυτής έτσι ώστε να βρεθεί η κατάλληλη θέση του στα φύλλα του B-δένδρου Συμβολοσειράς. Αν πρόκειται για εισαγωγή η νέα συμβολοσειρά τοποθετείται στο δίσκο και η διεύθυνση έναρξης της τοποθετείται στο κατάλληλο φύλλο του B-δένδρου Συμβολοσειρών. Αν το στοιχείο αυτό χωράει στην αντίστοιχη σελίδα του δίσκου απλώς τοποθετείται αλλιώς η σελίδα αυτή σπάει και τα στοιχεία μοιράζονται εξίσου. Επίσης μπορεί να χρειαστεί ενημέρωση του πατέρα προσθέτοντας ένα επιπλέον διαχωριστή. Γενικότερα ισχύουν οι πράξεις ζύγισης που ισχύουν στα B-δένδρα τόσο στην εισαγωγή όσο και στην διαγραφή ενός κλειδιού/συμβολοσειράς. Στην χειρότερη περίπτωση οι πράξεις επαναζύγισης να διαδοθούν μέχρι και την ρίζα, συνεπώς ο συνολικός αριθμός προσβάσεων στο δίσκο για μια πράξη ενημέρωσης θα είναι $O(\frac{|T'|}{B} + \log_B |\Delta|)$ I/Os. Η παραπάνω συζήτηση συνοψίζεται στο ακόλουθο θεώρημα.

Θεώρημα 3.2. *Το B-δένδρο Συμβολοσειρών όταν δεικτοδοτεί ένα σύνολο συμβολοσειρών Δ δύναται να απαντήσει ποιες από αυτές έχουν ως πρόθεμα ένα ερώτημα P με $O(\frac{P}{B} + \log_B |\Delta| + \frac{\alpha}{B})$ I/Os, όπου α το μέγεθος της απάντησης. Για την εισαγωγή ή την διαγραφή μιας συμβολοσειράς T' από το σύνολο Δ χρειάζονται $O(\frac{|T'|}{B} + \log_B |\Delta|)$ I/Os.*

Προσπαθώντας να επιλύσουμε το πρόβλημα του ταιριάσματος προτύπου με το B-δένδρο Συμβολοσειρών θα πρέπει να εισάγουμε σε αυτό όλα τα επιθέματα των συμβολοσειρών (συνεπώς $\Delta = \{Suffixes(T_i), \forall T_i\}$) έτσι ώστε χρησιμοποιώντας το ερώτημα προθέματος του Θεωρήματος 3.2 να επιστρέφονται οι θέσεις εμφάνισης ενός προτύπου P μέσα στις συμβολοσειρές που δεικτοδοτούνται. Το πλήθος των προσβάσεων στο δίσκο θα είναι αντίστοιχα $O(\frac{P}{B} + \log_B |\Delta| + \frac{\alpha}{B})$ I/Os, όπου α το μέγεθος της απάντησης. Χρησιμοποιώντας τις πράξεις ενημέρωσης (εισαγωγή/διαγραφή συμβολοσειράς) για μια συμβολοσειρά T', όπως τις έχουμε δει μέχρι στιγμής, θα χρειαστεί να εισάγουμε/διαγράψουμε μαζί με της T' και όλα τις τα επιθέματα. Συνεπώς σύμφωνα με το Θεώρημα 3.2 το συνολικό κόστος ενημέρωσης για μια συμβολοσειρά T' θα είναι $O(\sum_{i=1}^{|T'|}(\frac{|T'_{i\ldots|T'|}|}{B} + \log_B(|\Delta| + i - 1))) \leq O(\frac{|T'|^2}{B} + |T'| \log_B(|\Delta| + |T'|))$ I/Os. Με λίγη επιπλέον επιμέλεια οι πράξεις ενημέρωσης μπορούν να γίνουν πιο οικονομικά χρησιμοποιώντας επιπρόσθετους δείκτες στο B-δένδρο Επιθεμάτων και αποφεύγοντας την εκτέλεση προσβάσεων στο δίσκο που έχουν πραγματοποιηθεί στην εισαγωγή του αμέσως προηγούμενου επιθέματος. Οι πράξεις ενημέρωσης όπως θα δούμε και στην συνέχεια μπορεί να γίνουν με $O(|T'| \log_B(|\Delta| + |T'|))$ I/Os.

Για την επίτευξη της παραπάνω πολυπλοκότητας χρησιμοποιείται επιπρόσθετη βοηθητική πληροφορία η οποία δεν αυξάνει ασυμπτωτικά το μέγεθος του B-δένδρου Συμβολοσειρών. Η επιπρόσθετη πληροφορία είναι σε κάθε κόμβο του B-δένδρου Συμβολοσειρών χρησιμοποιείται ένας επιπλέον δείκτης ο οποίος δείχνει τον πατέρα του κόμβου και για κάθε επίθεμα ένας δείκτης που υποδεικνύει το φύλλο του αμέσως επόμενου επιθέματος της συμβολοσειράς που προέρχεται, έτσι $succ(leaf_{T_i}) = leaf_{T_{i+1}}$ (βλ. Σχήμα 3.13). Το τελευταίο επίθεμα μιας συμβολοσειράς θα δείχνει τον ίδιο τον εαυτό. Με χρήση αυτών των δεικτών η εύρεση της θέσης όπου τοποθετείται το επίθεμα $i + 1$ δεν βρίσκεται ξεκινώντας διαδικασία αναζήτησης από την ρίζα αλλά ξεκινώντας από το φύλλο του αμέσως μεγαλύτερου επιθέματος i ανεβαίνοντας προς τον βαθύτερο εσωτερικό κόμβο π που στο υποδένδρο του περιέχει τη θέση όπου πρέπει να τοποθετηθεί το προς εισαγωγή επίθεμα και τέλος βρίσκοντας αυτή την θέση. Στην συνέχεια θα δούμε το επαγωγικό βήμα αυτής της διαδικασίας καθώς και πόσες προσβάσεις γίνονται στο δίσκο συνολικά.

Σχήμα 3.13. Επιπρόσθετοι δείκτες στο B-δένδρο Συμβολοσειρών

Έστω λοιπόν ότι στο B-δένδρο έχουν εισαχθεί ένα σύνολο συμβολοσειρών μαζί με τα επιθέματα τους, δηλαδή $\Delta = \{Suffixes(T_i), \forall T_i\}$, και ότι θέλουμε

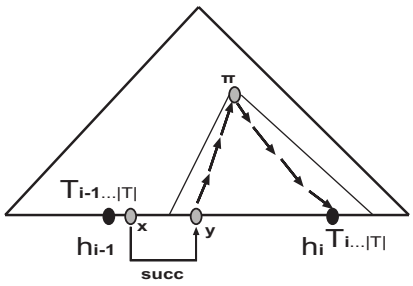
να εισάγουμε στη δομή μια νέα συμβολοσειρά T′ μαζί με όλα τα επιθέματα της. Υποθέτουμε ότι οι δείκτες *succ* για τις υπόλοιπες συμβολοσειρές έχουνε ήδη αρχικοποιηθεί στις προηγούμενες εισαγωγές. Όπως θα δούμε κατά την εισαγωγή της T′ αρχικοποιούνται όλοι οι δείκτες *succ* για κάθε ένα της επίθεμα οπότε η παραπάνω υπόθεση είναι αληθής. Η διαδικασία εισάγει διαδοχικά τα επιθέματα της T′ από το μεγαλύτερο προς το μικρότερο και στο τέλος κάθε τέτοιου βήματος ισχύουν τα εξής:

1. Έστω ότι βρισκόμαστε στο βήμα i όπου έχουν εισαχθεί στο B-δένδρο συμβολοσειρών τα i πρώτα επιθέματα. Το επίθεμα αυτό $T'_{i...|T'|}$ θα μοιράζεται ένα κοινό πρόθεμα με τα επιθέματα που αντιστοιχούν στα δύο γειτονικά φύλλα στο B-δένδρο Συμβολοσειρών όπου μόλις έχει τοποθετηθεί. Οι συμβολοσειρές/επιθέματα αυτών των γειτονικών φύλλων μπορεί να είναι είτε επιθέματα της T′ είτε κάποιας συμβολοσειράς που έχει είδη εισαχθεί. Αυτό το μέγιστο κοινό πρόθεμα μεταξύ του νέου επιθέματος και των είδη εισαγομένων επιθεμάτων θα εμφανίζεται είτε και στα δύο γειτονικά φύλλα είτε στο ένα από αυτά. Ας συμβολίσουμε με h_i το μέγιστο αυτό κοινό πρόθεμα τους, το οποίο βέβαια μπορεί να είναι και ίσο με το μηδέν.
2. Όλοι οι *succ* δείκτες για τα $i-1$ πρώτα επιθέματα της T′ καθώς και τα επιθέματα των άλλων συμβολοσειρών έχουν ήδη αρχικοποιηθεί ενώ ο δείκτης για το i-στο επίθεμα της T′ θα αρχικοποιηθεί στο επόμενο βήμα όταν θα έχει προσδιορισθεί η θέση/φύλλο του $i+1$-στου επιθέματος. Αν είναι το τελευταίο επίθεμα τότε δεν υπάρχει επόμενο βήμα και ο δείκτης δείχνει το ίδιο το φύλλο στο οποίο ανήκει.

Κατά την εισαγωγή του πρώτου επιθέματος ($i = 0$) της T′ η διαδικασία ξεκινάει από τη ρίζα του B-δένδρου Συμβολοσειρών για την εύρεση της θέσης τοποθέτησης του $T'_{1...|T'|}$ στα φύλλα. Η καθοδική αυτή πορεία γίνεται όπως η διαδικασία αναζήτησης που περιγράφηκε στο Θεώρημα 3.2. Κατά το τελικό στάδιο της αναζήτησης αυτής, κάτω στα φύλλα όπου τοποθετείται το επίθεμα υπολογίζεται και το μέγιστο κοινό πρόθεμα h_0 που έχει με κάποιο από τα γειτονικά φύλλα . Για τα επόμενα επιθέματα ($i > 0$) η διαδικασία εύρεσης της θέσης αρχίζει από το φύλλο του προηγούμενο βήματος, δηλαδή του επιθέματος $T'_{i-1...|T'|}$ που εμφανίζει μέγιστο κοινό πρόθεμα h_{i-1}. Από το φύλλο, έστω φύλλο x αυτή η διαδικασία κινείται στο γειτονικό φύλλο με το οποίο μοιράζεται το μέγιστο κοινό πρόθεμα και στη συνέχεια ακολουθείται ο δείκτης *succ* του φύλλου αυτού προς το φύλλο y.

Το φύλλο αυτό y θα περιέχει στις συμβολοσειρές αντιπροσώπους του μια συμβολοσειρά X που θα έχει μέγιστο κοινό πρόθεμα με το προς εισαγωγή επίθεμα με τουλάχιστον μήκος $h_{i-1} - 1$ ή καμία συμβολοσειρά δεν έχει κοινό πρόθεμα. Έτσι λοιπόν θα ισχύει $lcp(X, T'_{i...|T'|}) \geq max\{0, h_{i-1} - 1\}$. Αυτή η σχέση είναι λογική αφού αν η συμβολοσειρά του φύλλου x έχει ως μέγιστο κοινό πρόθεμα με το επίθεμα $T'_{i-1...|T'|}$ μήκους h_{i-1} τότε το φύλλο y/επίθεμα X με το επίθεμα $T'_{i...|T'|}$ προκύπτουν αφαιρώντας τους αρχικούς χαρακτήρες αντίστοιχα. Από αυτή την παρατήρηση προκύπτει η σχέση $h_i \geq max\{0, h_{i-1} - 1\}$. Άρα έμμεσα χωρίς να φορτωθούν από το δίσκο και να ελεγχθούν γνωρίζουμε το μήκος του μέγιστου κοινού πρόθεματος του προς εισαγωγή επιθέματος με κάποιο από τα επιθέματα που βρίσκονται ήδη στη δομή.

Από αυτό λοιπόν το φύλλο y, ακολουθώντας τους δείκτες που δείχνουν στους πατέρες μια διαδικασία κινείται προς την ρίζα. Η διαδικασία συνεχίζει να ανέρχεται όσο επιτυγχάνει να επεκτείνει το μέγιστο κοινό πρόθεμα του επιθέματος $T'_{i...|T'|}$ με κάποιο από τα κλειδιά/αντιπροσώπους που συναντάει στους κόμβους. Τελικά καταλήγει σε ένα εσωτερικό κόμβο p στον οποίο για τελευταία φορά κατάφερε να επεκτείνει το κοινό πρόθεμα αυτό. Εδώ αξίζει να σημειώσουμε ότι φορτώνονται σε κάθε εσωτερικό κόμβο της κίνησης σελίδες του δίσκου ανάλογες της επέκτασης, με άλλα λόγια δεν χρειάζεται οι συγκρίσεις να γίνονται από την αρχή των συμβολοσειρών. Εν συνεχεία από τον κόμβο π η διαδικασία κατέρχεται εκτελώντας την κλασική πράξη αναζήτησης μέχρι την κανονική θέση/φύλλο του επιθέματος $T'_{i...|T'|}$. Από την διαδικασία αυτή όπως αναφέραμε και προηγουμένων υπολογίζεται ταυτόχρονα η μήκος h_i του μέγιστου κοινού προθέματος (βλ. Σχήμα 3.14).



Σχήμα 3.14. Διαδικασία εισαγωγής Επιθέματος

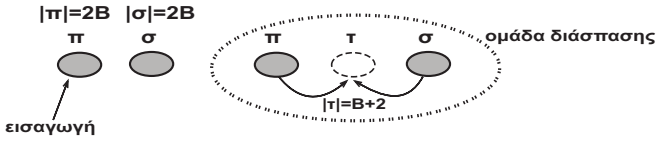
Από την παραπάνω συζήτηση του επαγωγικού βήματος της πράξης της εισαγωγής παρατηρούμε ότι για βρεθεί η θέση που πρέπει να τοποθετηθεί ένα επίθεμα χρειάζονται $O(\frac{h_i - max\{0,h_{i-1}\}}{B} + \log_B(|\Delta| + |T'|))\ I/Os$ όπου ο πρώτος όρος εκφράζει τις αναγκαίες σε κάθε εσωτερικό κόμβο προσβάσεις στο δίσκο για τις συγκρίσεις των προθεμάτων με τα κλειδιά ενώ ο δεύτερος εκφράζει τις προσβάσεις για τους εσωτερικούς κόμβους για την ανοδική-καθοδική πορεία της διαδικασίας. Συνεπώς για τα $|T'|$ επιθέματα θα χρειαστούν $O(\frac{|T'|}{B} + |T'| \log_B(|\Delta| + |T'|)) = O(|T'| \log_B(|\Delta| + |T'|))\ I/Os$

Μετά από την εισαγωγή ενός επιθέματος $T'_{i...|T'|}$ είναι δυνατόν να προκληθούν επαναζυγιστικές πράξεις όπως είδαμε και στην απλή εισαγωγή των B-δένδρων Συμβολοσειρών. Η διαδικασία διάσπασης των κόμβων διαφοροποιείται ελαφρώς εξαιτίας της ύπαρξης των επιπλέων βοηθητικών στοιχείων $succ$ και δεικτών γονέα. Έστω ότι το φύλλο είναι γεμάτο και με την εισαγωγή ενός φύλλο/επιθέματος σπάει, τότε σε αυτή την περίπτωση αναγκαίο είναι να ενημερωθούν $O(B)$ δείκτες γονέα και $O(B)$ δείκτες $succ$. Παρατηρήστε ότι για να ενημερώσουμε σε αυτό το χρόνο τους δείκτες της δεύτερης κατηγορίας αυτού θα πρέπει να είναι αμφίδρομοι. Επίσης για κάθε εσωτερικό κόμβο στην χειρότερη περίπτωση χρειάζονται εκτός από την διάσπαση του κόμβου άλλες $O(B)$ ενημερώσεις δεικτών γονέα. Συνεπώς αφού στην χειρότερη περίπτωση οι επαναζυγίσεις φτάνουν μέχρι την ρίζα για την πράξη εισαγωγής ενός μόνο επιθέματος θα χρειαστούν $O(B \log_B(|\Delta| + |T'|)\ I/Os$. Συνεπώς για την εισαγ-

ωγή όλων των επιθεμάτων μιας συμβολοσειράς στο B-δένδρο Συμβολοσειρών θα χρειαστούν $O(B|T'|\log_B(|\Delta|+|T'|))$ I/Os στην χειρότερη περίπτωση.

Όπως είδαμε παραπάνω η διαδικασία ενημέρωσης, στην χειρότερη περίπτωση, μπορεί να κινηθεί από ένα φύλλο προς την ρίζα εκτελώντας $O(B)$ I/Os σε κάθε έναν από τους $O(\log_B(|\Delta|+|T'|))$ εσωτερικούς κόμβους. Μπορεί να δειχθεί ότι ενημερώνοντας με χαλαρό (lazy) τρόπο τους εσωτερικούς κόμβους μπορεί να απαλειφθεί ο παράγοντας B από την συνολική πολυπλοκότητα. Στην ουσία αντί να εκτελεστούν $O(B)$ I/Os σε κάθε εσωτερικό κόμβο, έτσι ώστε αυτός να είναι συνεπής με την ενημέρωση που έγινε στο δένδρο, εκτελούνται $O(1)$ I/Os και οι επιπλέον ενέργειες που πρέπει να γίνουν αφήνονται για μεταγενέστερες προσβάσεις στον κόμβο αυτό. Μπορεί να δειχθεί ότι μέχρι να χρειαστεί να ξανα-ενημερώσουμε τον συγκεκριμένο εσωτερικό κόμβο εξαιτίας μιας νέας ενημέρωσης αυτός εν το μεταξύ θα έχει γίνει συνεπής.

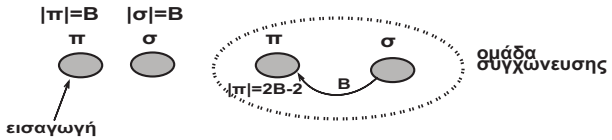
Έστω ότι κατά την διάρκεια μιας ενημέρωσης ένας κόμβος υπερχειλίζει. Έχει δηλαδή τον μεγαλύτερο επιτρεπτό βαθμό και χρειάζεται να προσθέσουμε σε αυτόν άλλο δύο κλειδιά που αντιστοιχούν σε ένα επιπλέον υποδένδρο. Έστω ότι υπερχειλίζει ο κόμβος π συνεπώς $|\pi| = 2B$ (βλ. σχήμα 3.15). Τότε τα δύο νέα κλειδιά εισάγονται μετακινώντας τα δύο δεξιότερα στον προς τα δεξιά αδελφικό του κόμβο ή τα δύο αριστερότερα στον προς τα αριστερά αδελφικό του κόμβο. Αν και οι δύο του αδερφικοί του κόμβοι έχουν το μέγιστο επιτρεπτό βαθμό τότε ο π θα πρέπει να σπάσει δημιουργώντας έναν νέο κόμβο τ. Έστω ότι αυτός ο κόμβος δημιουργείται δεξιά από τον π μεταξύ αυτού και του πρώην δεξιού του αδελφικού του κόμβου σ (βλ. σχήμα 3.15). Στον νεοσύστατο κόμβο θα πρέπει να τοποθετηθούν $B+2$ από τον π και τον σ έτσι ώστε να αποκτήσει και αυτός τον ελάχιστο επιτρεπτό βαθμό. Αντί να μεταφερθούν όλα αυτά τα κλειδιά μεταφέρονται μόνο τέσσερα (4) ενώ τα υπόλοιπα σημειώνονται ως υποχρεώσεις που πρέπει να εκπληρωθούν στο μέλλον για να γίνουν και οι τρεις συνεπείς με τη δομή του B-δένδρου. Επιπρόσθετα και οι τρεις κόμβοι σημειώνονται με ειδικό τρόπο (μέσω δεικτών) σχηματίζοντας μια ομάδα διάσπασης (split cluster). Κάθε φορά που γίνεται μια μεταγενέστερη πράξη ενημέρωσης (εισαγωγή ή διαγραφή) που αγγίζει αυτήν την ομάδα μετακινεί τέσσερα (4) κλειδιά από αυτά που έπρεπε να είχαν μετακινεί. Με αυτό τον τρόπο εφαρμόζουμε λοιπόν μια χαλαρή διάσπαση κόμβου.



Σχήμα 3.15. Διαδικασία χαλαρής διάσπασης ενός κόμβου

Στην περίπτωση που κατά την διάρκεια μιας ενημέρωσης ένας κόμβος υποχειλίσει, δηλαδή έχει βαθμό B και θέλουμε να αφαιρέσουμε από αυτόν δύο κλειδιά τότε αυτός είτε δανείζεται δύο κλειδιά από κάποιον από τους αδελφικούς του κόμβους είτε όταν αυτό δεν είναι επιτρεπτό αναγκαστικά συγχωνεύεται με έναν από αυτούς. Έστω ότι εξετάζουμε την δεύτερη περίπτωση και ο κόμβος π αναγκαστικά συγχωνεύεται με τον προς τα δεξιά αδελφικό του κόμβο σ (βλ.

σχήμα 3.16). Αντί να μεταφερθούν και τα B κλειδιά του σ μεταφέρονται και σε αυτή την περίπτωση μόνο τέσσερα (4) ενώ τα υπόλοιπα μεταφέρονται πάλι σε κάθε μεταγενέστερη πράξη που αγγίζει έναν από τους δύο κόμβους. Σε αυτή την περίπτωση η ομάδα που σχηματίζουν καλείται ομάδα συγχώνευσης (merge cluster).



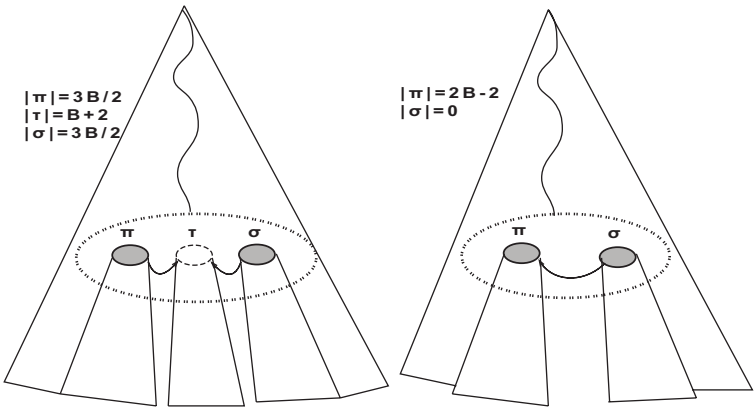
Σχήμα 3.16. Διαδικασία χαλαρής συγχώνευσης δύο κόμβων

Με τον χαλαρό τρόπο ενημέρωσης κάθε εσωτερικού κόμβου είτε όταν αυτός πρέπει να διασπαστεί είτε όταν πρέπει να συγχωνευθεί βλέπουμε ότι γίνεται σταθερός αριθμός από I/Os και όχι $O(B)$ όπως θα έπρεπε. Επίσης κάθε μεταγενέστερη ενημέρωση αφιερώνει σε αυτές τις ειδικές ομάδες πάλι μόνο σταθερό αριθμό από I/Os. Έτσι συνολικά κάθε πράξη ενημέρωσης που κινείται από την ριζά προς τα φύλλα και από τα φύλλα προς την ρίζα εκτελεί σε κάθε έναν από τους $O(\log_B(|\Delta|+|T'|))$ εσωτερικούς κόμβους σταθερό αριθμό από I/Os οπότε συνολικά θα απαιτούν $O(|T'|\log_B(|\Delta|+|T'|))$ I/Os στην χειρότερη περίπτωση. Δηλαδή θα έχει απαλειφθεί ο παράγοντας B που υπήρχε αρχικά. Το ερώτημα που τίθεται είναι αν αφήνοντας κάποια 'χρέη' σε έναν εσωτερικό κόμβο αυτά ξεπληρώνονται μέχρι μια νέα πράξη χρειαστεί να τον διασπάσει ή να τον συγχωνεύσει ξανά. Αν η απάντηση είναι αρνητική τότε ενδεχομένως να παραβιάζεται η δομή του B-δένδρου και άρα οι ιδιότητες τους με συνέπεια η παραπάνω πολυπλοκότητα να μην ισχύει. Κάτι τέτοιο όμως δεν συμβαίνει όπως μας λέει και το ακόλουθο λήμμα.

Λήμμα 3.3. *Κατά την διάρκεια που ένας κόμβος ανήκει σε μια ομάδα, είτε αυτή είναι ομάδα διάσπασης είτε ομάδα συγχώνευσης, καμιά πράξη ενημέρωσης δεν πρόκειται να επιχειρήσει να τον διασπάσει ή να τον συγχωνεύσει.*

Απόδειξη. Αμέσως μετά από μια διάσπαση ενός κόμβου, δηλαδή από τον σχηματισμό μιας ομάδας διάσπασης, οι τρεις κόμβοι της ομάδας θα έπρεπε να έχουν βαθμούς $|\pi| = \frac{3}{2}B, |\tau| = B+2$ και $|\sigma| = \frac{3}{2}B$ αντίστοιχα (βλ. σχήμα 3.17). Στην ουσία θεωρούμε ότι έχουν αυτούς τους βαθμούς αλλά τα κλειδιά που είναι να μεταφερθούν στην πραγματικότητα δεν έχουν μεταφερθεί ακόμη. Ο π και ο σ για να φθάσουν τον μέγιστο επιτρεπτό βαθμό θα χρειαστούν τουλάχιστον $\frac{B}{2}$ εισαγωγές στο υποδένδρο τους ενώ για να φτάσουν στον μικρότερο επιτρεπτό βαθμό θα χρειαστούν τουλάχιστον $\frac{B}{2}$ διαγραφές. Αφού σε κάθε άγγιγμα της ομάδας μεταφέρονται τέσσερα κλειδιά χρειάζονται να την αγγίζουν μόνο $\frac{B+2}{4}$ πράξεις ενημέρωσης μέχρι ο $|\tau|$ να αποκτήσει το ελάχιστο επιτρεπτό βαθμό. Μόλις αυτό γίνει τότε οι ομάδα παύει να υφίσταται και οι κόμβοι πλέον συμπεριφέρονται ως απλοί κόμβοι. Συνεπώς βλέπουμε τη δομή του δένδρου να αποκαθίσταται πριν χρειαστεί να επαναζυγισθεί ένας κόμβος σε μια ομάδα διάσπασης. Ομοίως μόλις σχηματισθεί μία ομάδα συγχώνευσης (βλ. σχήμα 3.17) θα

πρέπει στον π να μεταφερθούν $B-2$ κλειδιά. Ο π ή ο σ χρειάζονται B τουλάχιστον εισαγωγές στο υποδένδρο του για να φτάσουν το μέγιστο επιτρεπτό βαθμό ενώ για να φτάσουν τον μικρότερο επιτρεπτό βαθμό απαιτούνται τουλάχιστον $\frac{B}{2}$. Συνεπώς η συγχώνευση θα έχει ολοκληρωθεί σε $\frac{B}{4}$ προσβάσεις πολύ πριν χρειαστεί κάποιος από τους κόμβους να επαναζυγισθεί.



Σχήμα 3.17. Σχήμα Λήμματος 3.3

Με βάση τα παραπάνω προκύπτει το ακόλουθο θεώρημα.

Θεώρημα 3.4. *Το B-δένδρο Συμβολοσειρών όταν δεικτοδοτεί τα επιθέματα ενός συνόλου συμβολοσειρών $T_1+T_2\cdots T_k$, με $\Delta=\{Suffixes(T_i),\forall T_i\}$, δύναται να απαντήσει το ερώτημα ταιριάσματος προτύπου P με $O(\frac{P}{B}+\log_B|\Delta|+\frac{\alpha}{B})$ I/Os, όπου α το μέγεθος της απάντησης. Για την εισαγωγή ή την διαγραφή μιας συμβολοσειράς T' και των επιθεμάτων της από το σύνολο Δ χρειάζονται $O(|T'|\log_B(|\Delta|+|T'|))$ I/Os*

3.6.1 Εφαρμογές του B-δένδρου Συμβολοσειρών

Πέρα από τα προβλήματα εύρεσης προτύπου και ταιριάσματος προθέματος το B-δένδρο συμβολοσειρών δύναται να χρησιμοποιηθεί σε έναν μεγάλο αριθμό από εφαρμογές. Μια από τις πιο βασικές εφαρμογές του είναι η υλοποίηση δυναμικών πινάκων επιθεμάτων στην κύρια μνήμη. Αφού τα φύλλα-επιθέματα είναι διατεταγμένα λεξικογραφικά στην ουσία από το B-δένδρο συμβολοσειρών μπορεί άμεσα να εξαχθεί ο πίνακας επιθεμάτων μιας συμβολοσειράς. Στην περίπτωση που θέλουμε να εκτελούμε πράξεις ενημέρωσης στις συμβολοσειρές που δεικτοδοτούμε είτε υπό την μορφή εισαγωγής σε μια θέση είτε με την διαγραφή μιας υποσυμβολοσειράς. Υπό αυτές τις συνθήκες από την δομή θα πρέπει να προστεθούν επιθέματα ή να αφαιρεθούν αντίστοιχα συνεπώς το B-δένδρο Συμβολοσειρών μπορεί να υποστηρίξει αυτές τις πράξεις χωρίς να χρειαστεί να κατασκευαστεί η δομή από την αρχή. Το θέμα αυτό θα επανεξεταστεί στο κεφάλαιο **;;** όπου πραγματεύεται την δεικτοδότηση συμβολοσειρών που υπόκεινται σε πράξεις ενημέρωσης.

Αναφορές

[AFGV97] Lars Arge, Paolo Ferragina, Roberto Grossi, and Jeffrey Scott Vitter. On sorting strings in external memory. In *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pages 540–548, El Paso, 1997. ACM Press.

[AIL+88] A. Apostolico, C. Iliopoulos, G. Landau, B. Schieber, and U. Vishkin. Parallel construction of a suffix tree. *Algorithmica*, 3:347–365, 1988.

[BBH+87] A. Blumer, J. Blumer, D. Haussler, R. McConnell, and A. Ehrenfeucht. Complete inverted files for efficient text retrieval and analysis. *Journal of the ACM*, 34(3):578–595, 1987.

[BKM+00] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web: Experiments and models. In *9th International World Wide Web Conference*, pages 309–320, Amsterdam, 2000.

[BM77] R. Boyer and J.S. Moore. A fast string searching algorithm. *Communications of the ACM*, 20:762–772, 1977.

[BU77] R. Bayer and K. Unterauer. Prefix b-trees. *ACM Transactions on Database Systems*, 2(1):11–26, 1977.

[BYBZ96] R. Baeza-Yates, E. Barbosa, and N. Ziviani. Hierarchies of indices for text searching. *Inf. Syst.*, 21(6):497–514, 1996.

[CLR90] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms*. MIT Press, 1990.

[CM96] D. Clark and I. Munro. Efficient suffix trees on secondary storage. In *Proceedings of the seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 383–391. Society for Industrial and Applied Mathematics, 1996.

[CR02] M. Crochemore and W. Rytter. *Jewels of Stringology*. World Scientific, London,UK, 2002.

[Cro85] M. Crochemore. Transducers and repetitions. *Theoretical Computer Science*, 45:63–86, 1985.

[CS85] M. Chen and J. Seiferas. Efficient and elegant subword tree construction. *Combinatorial Algorithm on Words, NATO Advanced Science Institutes, Series F*, 12:97–107, 1985.

[CV97] M. Crochemore and R. Verin. Direct construction of compact directed acyclic word graphs. In *Proceedings of the 8th Annual Symposium on Combinatorial Pattern Matching*, pages 116–129. Springer-Verlag, Berlin, 1997.

[DJHK03] D.K.Kim, J.Sim, H.Park, and K.Park. Linear time construction of suffix arrays. In *Proceedings of the 14th Ann. Symp. on Combinatorial Pattern Matching*, 2003.

[Fal85] C. Faloutsos. Access methods for text. *ACM Computing Surveys*, 17(1):49–74, 1985.

[Far97] M. Farach. Optimal suffix tree construction with large alphabets. In *38th Annual Symposium on the Foundations of Computer Science, FOCS 97*, New York, 1997.

[Fer02] P. Ferragina. String algorithms and data structures (brics summer school). Technical report, Dipartimento di Informatica, University Of Pisa, Italy, 2002.

[FG99] P. Ferragina and R. Grossi. The string b-tree: a new data structure for string search in external memory and its applications. *J. ACM*, 46(2):236–280, 1999.

[GI93] R. Grossi and G. Italiano. Suffix trees and their applications in string algorithms. In *1st South American Workshop on String Processing (WSP)*, pages 57–76, 1993.

[GK97] R. Giegerich and S. Kurtz. From ukkonen to mccreight and weiner: A unifying view of linear-time suffix tree construction. *Algorithmica*, 19(3):331–353, 1997.

[Gus97] D. Gusfield. *Algorithms on Strings, Trees and Sequences.* Cambridge University Press, Cambridge, UK, 1997.

[Har94] R. Hariharan. Optimal parallel suffix tree construction. In *26th Annual ACM Symposium on the Theory of Computing*, Canada, 1994.

[HT84] D. Harel and R. Tarjan. Fast algorithms for finding nearest common ancestors. *SIAM Journal on Computing*, 13:338–355, 1984.

[IHS+01] S. Inenaga, H. Hoshino, A. Shinohara, M. Takeda, S. Arikawa, G. Mauri, and G. Pavesi. On-line construction of compact directed acyclic word graphs. In *Proceedings of the 12th Annual Symposium on Combinatorial Pattern Matching*, pages 169–180. Springer-Verlag, 2001.

[KLA+01] T. Kasai, G. Lee2, H. Arimura, S. Arikawa, and K. Park. Linear-time longest-common-preffix computation in suffix arrays and its applications. In *Proceedings of the 12th Ann. Symp. on Combinatorial Pattern Matching*, 2001.

[KMP77] D. E. Knuth, J. Morris, and V. Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):127–146, 1977.

[Knu98] D. E. Knuth. *Sorting and Searching*, volume 3 of *The Art of Computer Programming.* Addison-Wesley, Reading MA, second edition, 1998.

[KS03] J. Karkkainen and P. Sanders. Simple linear work suffix array construction. In *ICALP*, 2003.

[Kur99] S. Kurtz. Reducing the space requirement of suffix trees. *Software-Practice and Experience*, 29(13):1149–1171, 1999.

[McC76] E. M. McCreight. A space-economical suffix tree construction algorithm. *Journal of hte Association for Computing Machinery*, 23(2):262–272, 1976.

[MM93] U. Manber and G. Myers. Suffix arrays: A new method for on-line string searches. *SIAM Journal of Computing*, 25(5):935–948, 1993.

[Nav96] G. Navarro. An optimal index for PAT arrays. In N. Ziviani, R. Baeza-Yates, and K. Guimarães, editors, *Proceedings of the 3rd South American Workshop on String Processing*, pages 214–227, Recife, Brazil, 1996. Carleton University Press.

[Nav01] G. Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1):31–88, 2001.

[PS03] P.Ko and S.Aluru. Space efficient linear time construction of suffix arrays. In *Combinatorial Pattern Matching*, 2003.

[RW94] C. Ruemmler and J. Wilkes. An introduction to disk drive modeling. *Computer, bIEEE Computer Society Press*, 27(3):17–28, 1994.

[SV94] S. Sahinalp and U. Vishkin. Symmetry breaking for suffix tree construction. In *26th Annual ACM Symposium on the Theory of Computing*, Canada, 1994.

[Szp92] Wojciech Szpankowski. (un)expected behavior of typical suffix trees. In *SODA '92: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 422–431. Society for Industrial and Applied Mathematics, 1992.

[Ukk95] E. Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14:249–260, 1995.

[Wei73] P. Weiner. Linear pattern matching algorithms. In *14th IEEE Anual Symposium on Switching and Automata Theory*, pages 1–11, IEEE, 1973.

[WMB99] I. Witten, A. Moffat, and T. Bell. *Managing Gigabytes. Compressing and Indexing Documents and Images.* Morgan Kaufmann Publishers, Inc, San Francisco, USA, 1999.

[ZMK98] J. Zobel, A. Moffat, and K.Ramamohanarao. Inverted files versus signature files for text indexing. *ACM Transactions on Database Systems*, 23:453–490, 1998.