Δομές δεδομένων

- Sorting & Searching, Springer Verlag, 1984
- Intro to Algorithms, Corner, Leirerson, Rivest, Stein, MIT PRESS, 200.
- Π Δ Μπογάνης, Δομές δεδομένων

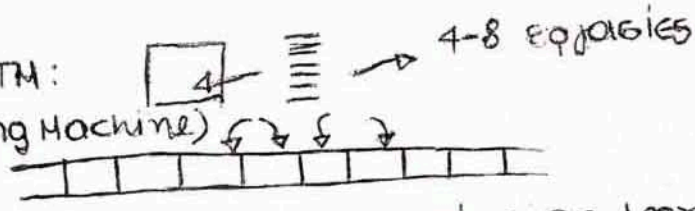
Για οποιαδήποτε απορία : panagis@ceid.upatras.gr

2 ασκήσεις → 1 μέχρι τα χριστούγεννα
→ 1 για την εξεταστική (μετράει)

1) Αλγοριθμοι ταξινόμησης
2) Αναζήτηση
3) Δέντρα

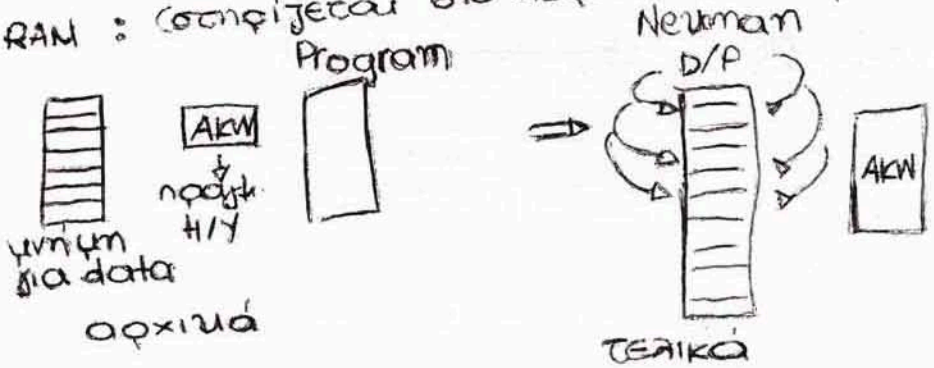
- Μοντέλα Υπολογισμού - 3/11/2003

1) TM: (Turing Machine)



Υπολογισμός : ορισμός κάποιας μηχανής και μετά υπολογίζω.

2) RAM : εισήχθεται στο πυρήνα έχει μειωθεί ο χρόνος + έχει αυξηθεί η ταχύτητα



□ 1cm² ← 200 transistors (1976)

□ 1cm² ← 1M transistors (1989) → Siemiens

□ 1cm² ← 1 τρις transistors (1995-1997) → Philips

3) Φωτονικοί Η/Υ (συνδεσμολογία στηριγμένη στο φως)

Βιολογικοί Η/Υ (-||- -||- -||- κύτταρα)

0	1
< 2V	3E-5V

} Leibniz 1675

$(TM)^3 \equiv RAM$

Είναι τρεις φορά πιο χρήσιμη στη RAM

3) PM (Pointer Machine)

4) PPM (Pure Pointer Machine)

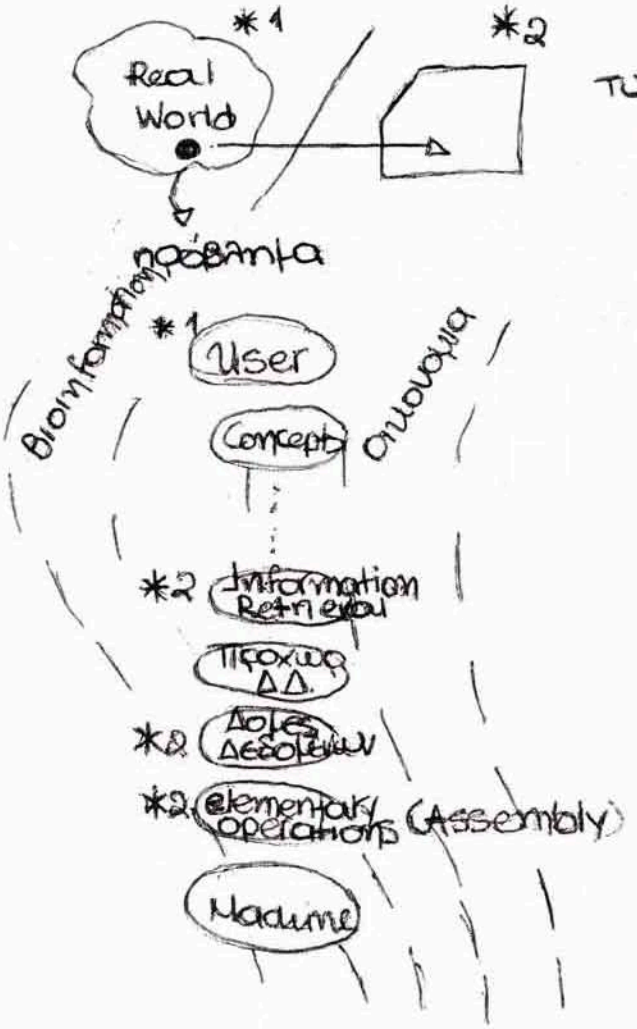
Searching is $\log\log n = \left(\log\left(\log 2^{35}\right)\right) = 5.5$

2^5 2^{10} 2^{10} 2^{10}

30.000.000.000 δισεκατ. ανθρωποι

$\hookrightarrow 2^{35}$

Distributed ⇒ Δεν είναι όλα τα κομμάτια στον ίδιο server.
Είναι διαμοιρασμένα σε διαφορους servers.



ΤΥΠΟΠΟΙΗΜΕΝΟΣ ΚΟΣΜΟΣ

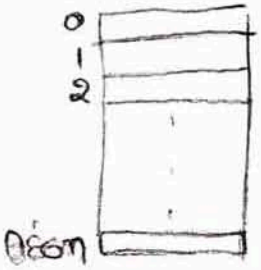
Bio	Info
DNA + Proteins so	Strings
Protein Clustering	DB-Clusters
Drug Design	Computation Geometry

ΔΙΕΠΙΣΤΗΜΟΝΙΚΟΤΗΤΑ

Μοντέλα Μηχανής

1) RAM (RANDOM ACCESS MACHINE) → πιο ισχυρό μοντέλο
↳ Αλληλοδοιμένη ενδοχή των Η/Υ
υποστηρίζει στοιχειώδεις πράξεις

- 1 ACC
- 3 indx registers

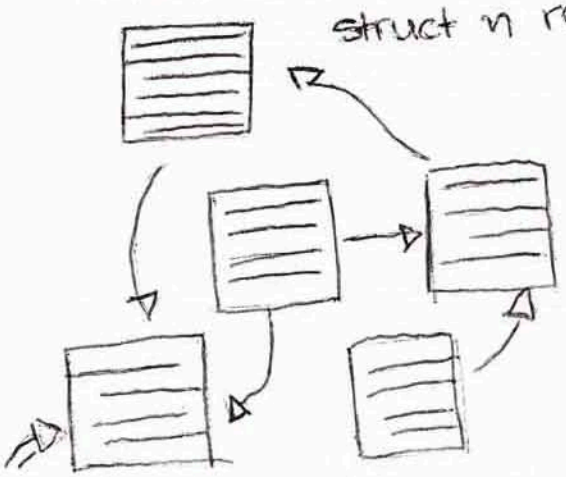


Ιδιότητα: ευθυγραμμιστός ο υπολογισμός διεύθυνσης

εντολές + δεδομένα: χωράνε σε μια θέση μνήμης
→ αλγορίθμους που υλοποιούνται με arrays

2) PM (POINTER MACHINE) → πιο κομψοί αλγόριθμοι
↳ είναι σαν να χώρουμε τις δ. μνήμης της RAM + και τις διασκορπίσουμε στο χώρο
Δεν έχει την δυνατότητα του υπολογισμού διεύθυνσης
struct ή records

→ ΜΟΝΟ structs (records)



```
MAX {
1. m=A[1]; i=j=1;
2. while (j<n) do {  → N
3.   if (A[j] > m) {  → N-1
4.     m=A[j];
5.     i=j;  }  → M
   }
6. j=j+1;  → N-1
}
}
```

* Η χειρότερη περίπτωση για M είναι N-1 *

A[1....N]

• Πολυπλοκότητα αλγορίθμου: αριθμός των βημάτων
↳ T(N) = 3N-2+N+1
- περιπτώσεις πολυπλοκότητας -
(α) ΠΧΠ → χειρότερη περίπτωση, μέγιστος αριθμός βημάτων.
↳ π.χ. ΠΧΠ = 3n-2+2n-2+1 = 5n-3
(β) ΠΜΟ → (στιγμιότυπα)
(μέσου όρου) P_k, $T_i(n)$
$\sum_{i=1}^{k} [P_i \cdot T_i(n)] \equiv ΠΜΟ(N) \geq 3n+1mn+12$
$\sum_{k} P_k = 1$

(γ) Επιμερισμένη πολυπλοκότητα (Amortized)
π.χ δυαδικός μετρητής

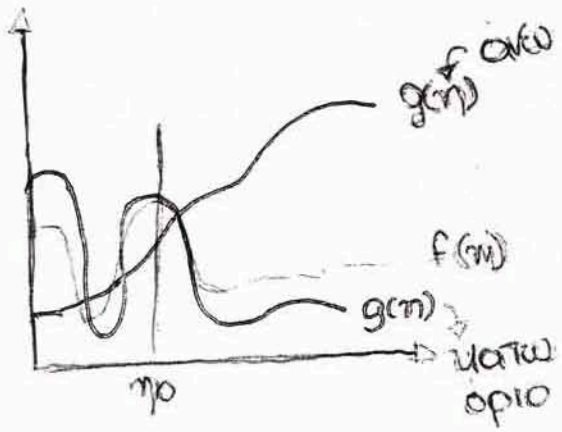
$Op_1, Op_2, \dots, Op_m$

$C(Op_1), C(Op_2), \dots, C(Op_m)$
(κόστος)

$$EK = \frac{\sum_{i=1}^{m} C(Op_i)}{m}$$

ΑΣΥΜΤΩΤΙΚΟΙ ΣΥΜΒΟΛΙΣΜΟΙ

Πάνω όριο:
(α) Ο μικρόν O: $f(n) = O(g(n))$: $\exists\, c>0,\ n_0 \Rightarrow \forall n>n_0 \Rightarrow$
$f(n) \le c \cdot g(n)$



$T(n)_{MAX} = 5n-3 = O(n)$
$T(n) = 15\log n = O(\log n)$
$T(n) = 5n^2+1100n+10 = O(n^2)$
$T(n) = 3n + 5\ln n = O(n)$
$3n\log n + 10 = O(n\log n)$
$T(n) = 2^{100} = O(1)$ ← σταθερή
↳ όσο μεγάλη και να είναι, είναι ανεξάρτητη του n άρα είναι σταθερά

(β) Κάτω όριο:
$f(n) = \Omega(g(n))$ όταν $\exists\, c>0$
υπάρχει $n_0 \mid \forall n>n_0 \Rightarrow f(n) \ge c\, g(n)$

γ) Ακριβής εκτίμηση
$f(n) = \Theta(g(n))$ * Της ίδιας τάξης μεγέθους *

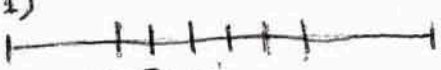
$f(n) = O(n)$
$f(n) = \Omega(n)$ } $f(n) = \Theta(g(n))$
(μεγαλύτερη)

π.χ $f(n) \le 5n$
$f(n) \ge 0{,}5n$ } $\Theta(n)$
(μικρότερη)
$f(n) = 0$

* Είναι γραμμική ως προς το μέγεθος εισόδου *

4/11/2003

Data Manipulation στον άξονα
(d=1)



οι δομές είναι για τον άξονα

αριθμοί
strings

(d=2) GIS, GUI } πολυδιάστατες δομές δεδομένων
(d=3) Graphics

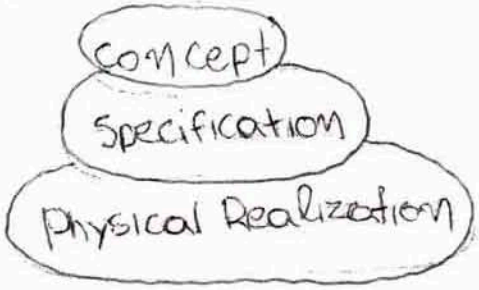
XML → specification για να ομαδοποιήσεις data

* οι Δ.Β. εμπεριέχουν πολυδιάστατες Δ.Δ. *

BIOML → Biodata
XML → δημόσια διοίκηση
ιδιωτική -||-

(d>3) DB (Data Base)

Για database πρέπει να κάνουμε 3 πράγματα:



searching - exact → το μάθημα Δ.Δ.
searching - approximate → για Web (by similarity, not exact)

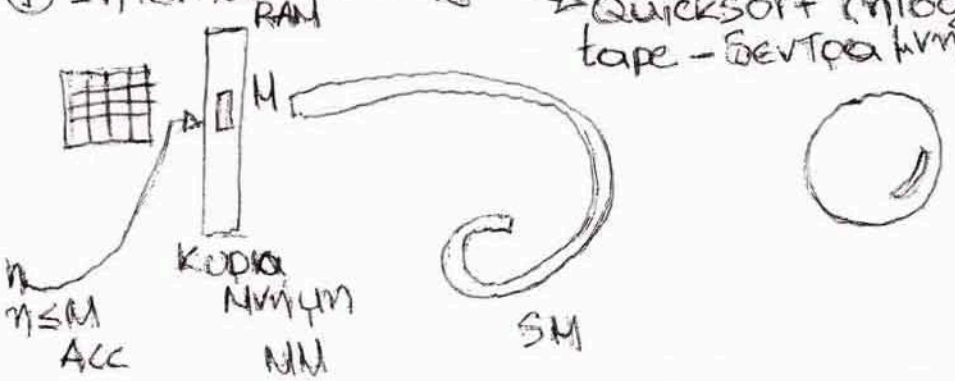
Hashing ⟺ Πρόβλημα του λεξικού
(Διαμοιρασμός)

Κεφ 1 Διαταξη στοιχείων (Sorting)

> Όλοι οι αριθμοί χωρούν σε μια θέση μνήμης και δεν μας ενδιαφέρει το μήκος τους

Handbook of Theoretical Compute Science, MIT PRESS, Elsevier, 199

① Internal Sorting
→ Bubblesort (n^2)
→ Heapsort ($n \log n$)
→ Quicksort ($n \log n$)
tape - δευτερα μνήμης



② External Sorting
(Δεν χωραν όλα στη μνήμη) → $n >> M$

Bubblesort

> Παραδειγμα:

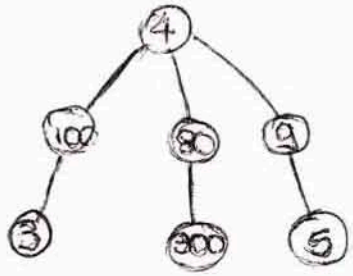
input 4, 3, 8, 6, 2, 5, 7
3, 4, 6, 2, 5, 7, 8
2, 3, 4, 5, 6, 7, 8
} n φορές

Πολυπλοκότητα → n^2 ή $\frac{n^2}{2}$ ή $\frac{n^2}{4}$

Heapsort

Heap = 


= μερική διάταξη



10/11/2003

Πρόβλημα Ταξινόμησης: Input → $S = \langle S_1, S_2, \dots, S_n \rangle$
Output → $S' = \langle S_1', S_2', \dots, S_n' \rangle$
$(S_1' \le S_2'' \le S_n')$

Αλγόριθμοι: (a) μόνο συγκρίσεις $\left\{ \begin{array}{l} \text{BubbleSort} \\ \text{Insertion sort} \\ \text{Heap Sort} \\ \text{Merge sort} \\ \text{Quick Sort} \end{array} \right.$ ⇒ $\Omega(\log n)$

(b) (+) $\left\{ \begin{array}{l} \text{radasort} \\ \text{bugasort} \\ \text{counting sort} \end{array} \right.$

(c) extended sort

→ Bubble Sort

S[1 n]

Πχ.

4	3	8	6	2	5	7
3	4	8	6	2	5	7
3	4	8	6	2	5	7
3	4	6	8	2	5	7
3	4	6	2	8	5	7
3	4	6	2	5	8	7
3	4	6	2	5	7	8
3	4	2	6	5	7	8
⋮						
2	3	4	5	6	7	8

```
Bubblesort(S,n){
 up ← n;
 while up > 1 {
   j ← 0;
   for i=1 to up-1 {
     if S[i] > S[i+1] {
       swap(S[i], S[i+1]);
       i++;
     }
   }
   if j==0 break;
   up--;
 }
```

Πολυπλοκότητα $= n^2$

while

$(n-1)+(n-2)+\ldots+(1)$ [αθροίζουμε το κόστος]

$1+2+\ldots+n = \frac{n(n+1)}{2}$

$O(n^2)$

▷ Insertion Sort ◁

(μια ταξινομημένη + μια μη ταξινομημένη)

S[1...n]

S'	S

S' | S

5 | 2 1 7

5

2 5

1 2 5

1 2 5 7

1 | 2 | 5 | 7

$n-1$

$O(n^2)$

Πχ

7 10 30	~~7~~ ~~10~~ ~~30~~ 9 22
7 9 10 30	
7 9 10 22 30	

Κόστος $\Rightarrow n^2$

⇒ Η χειρότερη περίπτωση είναι όταν η λίστα είναι τελείως ατοιχίστη ◁

```
Insertion(S,n){
 for j ← 2 to n {
   k ← S[j];
   i ← j-1;
   while i>0 && k < S[i] {
     S[i+1] ← S[i];
     i--;
   }
 S[i+1] ← k;
 }
```

→ Heap sort ←

S[1...n]

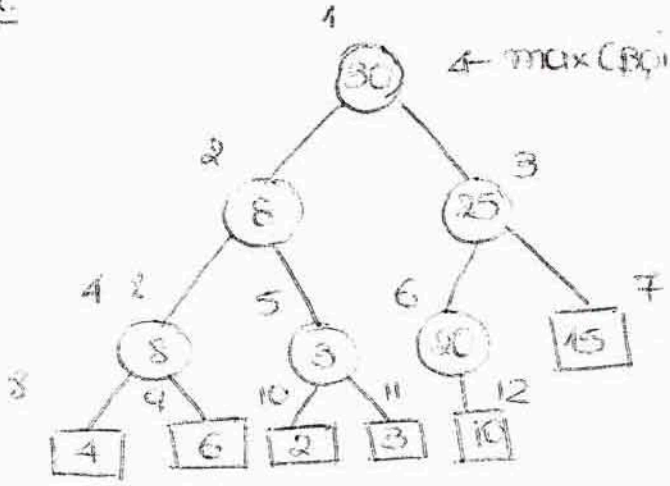
balanced binary heap:

i) $\forall u \rightarrow$ τιμή$(u) = S[i]$

ii) $\forall u \rightarrow$ Τιμή(πατέρα(u)) $\geq$ τιμή(u)

iii) $\forall u \rightarrow$ 0 ή 2 παιδιά εκτός αυτό τα φύλλα

π.χ.

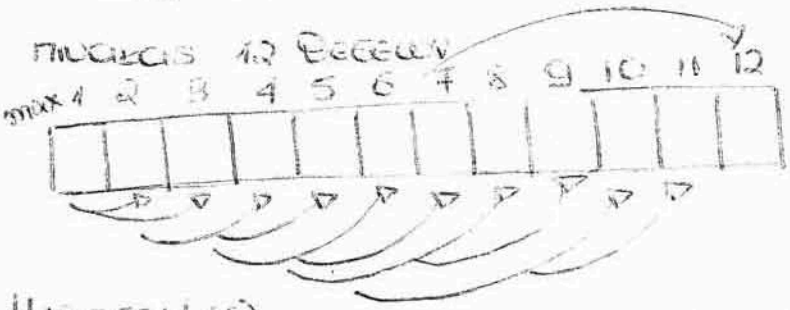


$$\text{αριστερός}(u) = \left[\frac{\text{αριστερός}(\rho)}{2}\right] = \left[\frac{\text{αριστερός}(i)}{2}\right]$$

○ → κορυφή

□ → φύλλο

πλευρές 12 βέλεν



$k \rightarrow 2k$, $k \rightarrow 2k+1$

Heapsort(S)

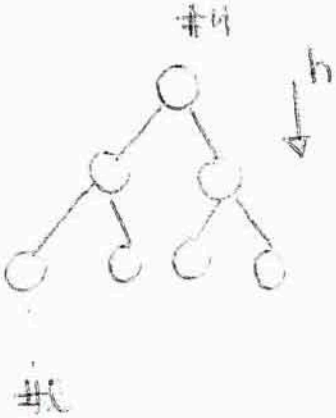
Build-heap( ), $(n-1)$ — $O(n)$ — $O(n) + O(n \log n)$

```
j←n;
for i←n to 2
  swap(S[i],S[j])
  j++;
  Heapify(i);
}
```

$O(n \log n)$

$O(n \log n)$

αποδ:  $\#n = \#l - 1$



$\boxed{\#n = \#l - 1}$

h	$\#n_h = \#l_{h-1}$
$h+1$	$\#n_{h+1} = \#n_h + \#l_h$
	$\#l_{h+1} = 2\#l_h$

$H(h) = \#n$ σε ύψος h

$H(h) \leq \frac{n}{2^{h+1}}$

$$C_{BuildHeap} = \sum_{i=1}^{n} a(h) = \sum_{h=1}^{\lceil \log n \rceil} H(h) \cdot O(h) \leq O\left[\sum_{h=1}^{\log n} \frac{n}{2^{h+1}} \cdot h\right] = O\left(n \sum_{h=1}^{\log n} \frac{h}{2^{h+1}}\right)$$

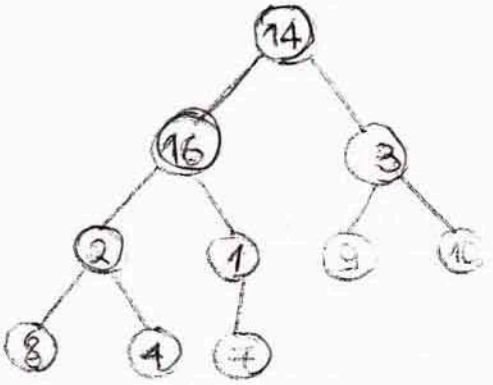
$= O(n)$ (εύκολη $\sum_{x=0}^{\infty} \frac{x}{x+1} = 2$)

$h = \lceil \log n \rceil$

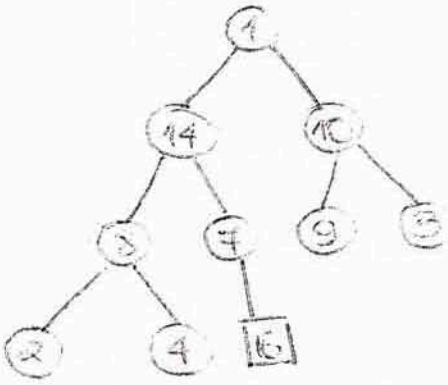
Π.χ.

14 16 3 2 1 9 10 8 4 7

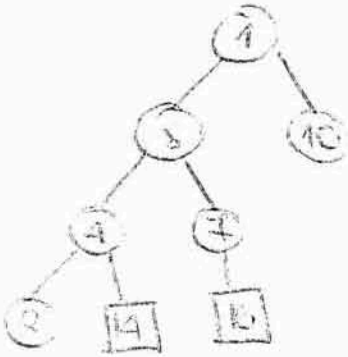
14 16



→ Το 1 με 7 δεν αναπαριστ.
Το 8 με 2 -||- -||-
Το 9 με 10 -||- -||-
Το 16 με 14 -||- -||-
(βάζω τα περισσότερα στα τέλη)

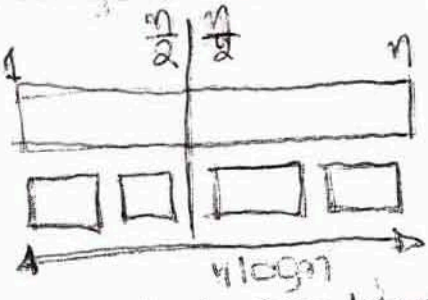


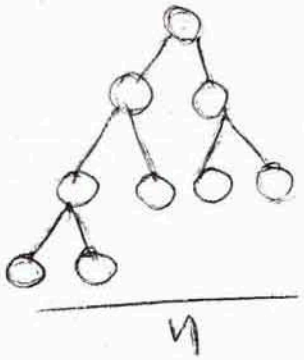
→ Το 14 δεν σκέφτεται τίποτε



→ Φτάνει στη ρίζα

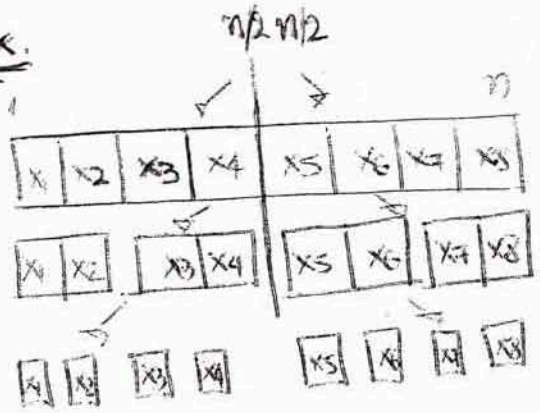
→ Mergesort ←





Το χωρίζω στη μέση

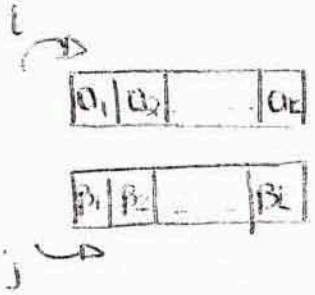
Πχ.



n log n εγγραφές

Τα διαλέγει ανά 2

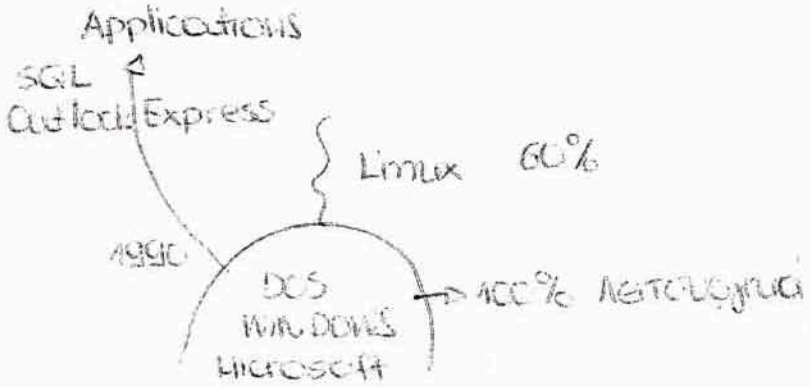
```
mergesort(S,n){
  S1 = S[1...n/2];
  S2 = S[n/2 ... n],
  if |S1| > 1   mergesort(S1, n/2)
  if |S2| > 1   mergesort(S2, n/2)
  merge(S1,S2);
}
```



α1β1 - - ___

→ Η χειρότερη περίπτωση είναι όταν γίνεται η εγγραφή σπάσμα μου δεν χρειάζεται το κόβω και μετά τα ένω ←

11/11/2000



→ Σύγκριση

W.C.

Bubblesort → $1 \cdot n$ (n^2)

Heapsort → $1 \cdot n$ $(n \log n)$ → Με τον όρο space εννοούμε το χώρο που καταλαμβάνει στη μνήμη

Mergesort → $2 \cdot n$

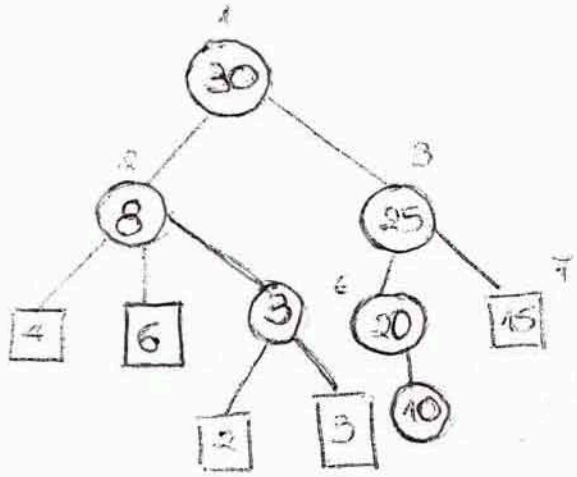
Quicksort → $1 \cdot n$ (n^2) → Ο χώρος έχει πιο μεγάλη σημασία από ότι ο χρόνος

External Sorting

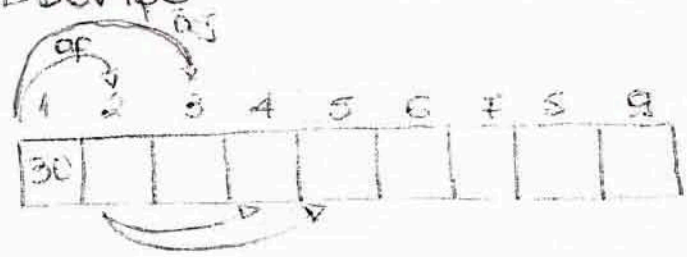
A-Sort (Adapting Sorting) → Προσαρμόζεται το sorting ανάλογα με το αν η λίστα είναι πρόσλοιπη ήδη (μέγιστο space $4 \cdot n$)
(n^2)

Αναπαράσταση

Linear Searching



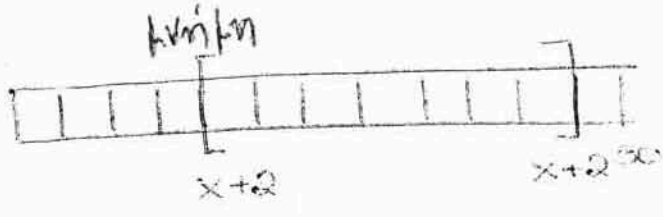
→ Δέντρο



Η PQP λοιπόν βασίζεται στο pointer machine αφού καθένας έχει ένα local finger και ακολουθεί το δικό του.

Θέμα: Initialization σε constant time

Compiler
A[1 . . 2^{30}]



fixarisma σε constant time

Initialization = $O(n)$

Χρησιμοποιούμε 2 stacks ώστε να βρίσκουμε το αν ορίσαι εγώ κάτι ή αν πέραν πάν ορισμένο

→ Quicksort 4-

Δοσμένη ακολουθία

5 36 5 10 8 41 66 75 43

χωρίζω σε δύο τμήματα υπαίρνοντας ένα στοιχείο και βάζοντας στο 1ο όλα τα μικρότερα αυτό αυτό και στο 2ο τα μεγαλύτερα

→ Στην χειρότερη περίπτωση, ο χρόνος που χρειάζεται είναι $\frac{n^2}{2}$

> A-Sort ←

F: πόσα γεγονότα είναι εκτός διάταξης

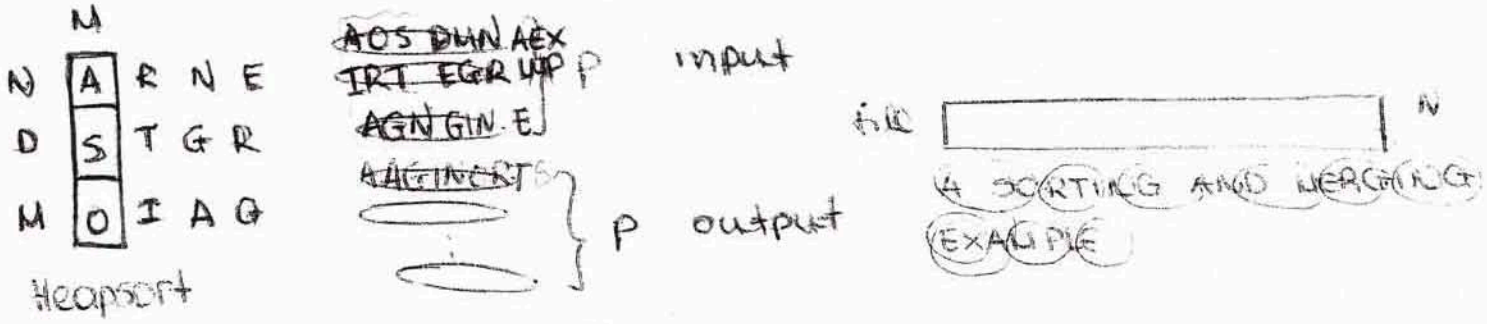
→ External Sorting ←

18/11/2003

file μήκους N

Μνήμη -"- M , M<<N

2p βοηθητικές ταινίες



→ Η μνήμη μπορεί να τα διαχειρίζεται.
→ (output το μήκος της μνήμης)

Αν είναι για heapsort χρησιμοποιούμε heap, τότε θέλουμε να παράγω runs > 3. (output μεγαλύτερο από το μήκος της μνήμης)

n = επίπεδο στοιχείο

m = χωρητ. Heap?

n ≥ m έξοδος

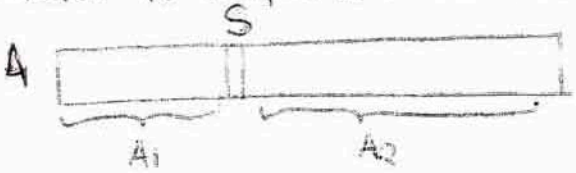
n < m n* → μεγαλύτερο ακέ. τα m μεταφερόμενα

Εφαρμογή	Μεθόδος Προγ.	Τύποι Στοιχείων
2000	Objects (=Data+Operations) → C++	
1990	Abstract data types	Machine, Assembly, D. Prog, User
1980	COBOL, Smalltalk	ASSEMBLY
1970 (PASCAL)	FORTRAN	INT, REAL, STRINGS
1960	ALGOL 60	
1950		

2 QuickSort(A1)

3 QuickSort(A2)

Μετα το βήμα 1, θα εχουμε:



πχ

S	i						
A: 7	3	5	9	8	1	2	17

Για λόγους ευκολίας, διαλέγουμε ως S το 1ο στοιχείο. Αυξάνω το i, ψάχνοντας για να βρω στοιχείο μεγαλύτερο του S. Ομοίως το j, αλλά για μικρότερο. Στη συνέχεια, τα εναλλάσσω.

7	3	5	2	8	1	9	17
				A_i	A_j		

Ξανανα μβαινουμε στα ίδια

7	3	5	2	1	8	9	17
				j	i		

Η 1 3 5 2 7 8 9 17 A2

Ξανάλλι τον quicksort για ∀ ένα από τα A1 και A2 (αναδρομική κλήση)

Ο αλγόριθμος τελειώνει όταν το μέγεθος γίνει ίσο με 2.

Χειρότερη περίπτωση

$$QS(n) = \underbrace{n+1}_{\text{συκρίσεις για διαχωρισμό του πίνακα}} + \max_{1 \le k \le n} \{QS(n-1) + QS(n-k)\}$$

$|A_1| = k-1$

$|A_2| = n-k$

$[n-1+2]$

συκρίνει με όλα τα στοιχεία το διαχωρισμό

τα 2 τελευταία παραμένουν θέσεις

→ Το μεγάλο κόστος είναι όταν δε κόβω με καλό διαχωρισμό

$QS(0) = QS(1) = 0$

Array

A[1... n]

Record

File

Stack →  → Heap (LIFO)

Queue →  → FIFO

Priority Queue

↳ ενεργεί ανουδήποτε (FIFO / ΚΑΛΟ ΕΙΝΑΙ ΝΕΑΟ)

Δίνει το μικρότερο

Κάνει το -||-

	insert anywhere	min	delete min
Heap	O(log n)	O(1)	O(log n)
TS	O(log d)	O(1)	O(1)

d = αριθμός από το min → χρόνος 25

SQL Server = DB εξυπηρέτησε (σε κρατάει transactions)

Persistant SQL server = DB + history

↳ Persistant Theory του Tarjan

24/11/2013

→ Quicksort

Divide & Conquer

Δοσμένο A ένας ΔΙΑΤΑΞΙΝΟΜΗΤΟΣ πίνακας

1. Χωρίζω A σε δύο S με

$A_1 = \{x \in A, x \leq S\}$ /* Δεξιοί */

$A_2 = \{x \in A, x > S\}$

Το μέγιστο κόστος προκύπτει για $k=1$ ή n ακολουθία είναι ταξινομημένη

$$QS(n) = n+1 + QS(0) + (n-1+1) + \underline{QS(n-2)}$$

$$QS(0) + n-1 + QS(n-3) = \dots$$

$$QS(n) = n+1+n+n-1+n-2+\dots+3 = \frac{(n+1)(n+2)}{2} - 3 = O(n^2)$$

ευρέως πίνακ

Ο Quicksort είναι πιο γρήγορος στη μέση περίπτωση

$O(n\log n)$ Heapsort
$O(n\log n)$ Mergesort

↳ Το $n\log n$ του Quicksort είναι πιο μικρό από το αντίστοιχο των Heapsort & Mergesort

Μέση Περίπτωση

1. Τα στοιχεία είναι σε 2 υποσύνολα
2. $n!$ μεταθέσεις είναι ισοπίθανες

$$\bar{QS}(n) = n+1 + \sum_{k=1}^{n} \frac{\left[Pr\{k=i\}\right]}{1/n} \left(\bar{QS}(k-1) + \bar{QS}(n-k)\right)\Big] = n+1 + \frac{1}{n}\sum_{k=1}^{n}\left(\bar{QS}(k-1) + \bar{QS}(n-k)\right) = n+1 + \frac{2}{n}\sum_{k=0}^{n-1}\bar{QS}(k) \Rightarrow$$

$$n\bar{QS}(n) = n(n+1) + 2\sum_{k=0}^{n-1}\bar{QS}(k) \quad (1)$$

$$\left.\begin{array}{l} \bar{QS}(0) + \bar{QS}(n-1) \\ + \bar{QS}(1) + \bar{QS}(n-2) \\ + \dots \\ + \bar{QS}(n-2) + \bar{QS}(1) \\ + \bar{QS}(n-1) + \bar{QS}(0) \end{array}\right\} 2\cdot\sum_{k=0}^{n-1}\bar{QS}(k)$$

$$(n+1)\bar{QS}(n+1) = (n+1)(n+2) + 2\sum_{k=0}^{n}\bar{QS}(k) \quad (2)$$

$$(2)-(1) = (n+1)\bar{QS}(n+1) - n\bar{QS}(n) = (n+1)(n+2) - n(n+1) + 2\bar{QS}(n) \Rightarrow$$

$$(n+1)\bar{QS}(n+1) = 2(n+1) + (n+2)\bar{QS}(n) \Rightarrow \bar{QS}(n+1) = 2 + \frac{(n+2)}{(n+1)}\bar{QS}(n)$$

$$\bar{QS}(n+1) = 2 + \frac{(n+2)}{(n+1)}\left(2 + \frac{n+1}{n}\bar{QS}(n-1)\right) = \dots = 2\left(1 + (n+2)\left(\frac{1}{n+1} + \frac{1}{n} + \dots + \frac{1}{3}\right)\right)$$

$$H_n = \sum_{i=1}^{n}\frac{1}{i} \le \ln n + 1$$

24/11/2000

$\sum_{i=2}^{n} \frac{1}{i} = \Theta(n)$

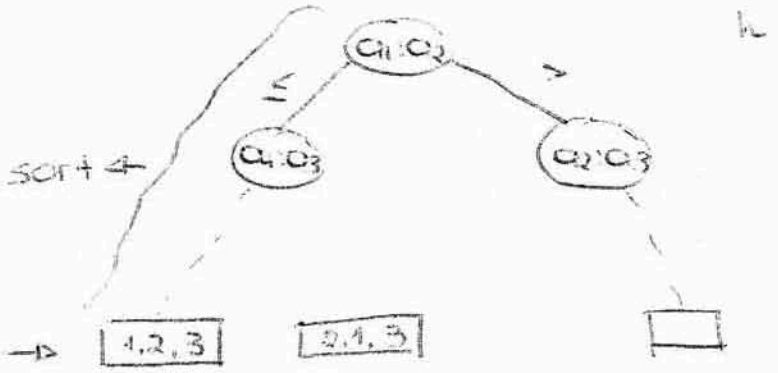
$\overline{QS}(n) = 2[1 + (n-1)(\frac{1}{n} + \frac{1}{n-1} + \dots + \frac{1}{2})] = 2 + 2(n-1)\sum_{i=2}^{n} \frac{1}{i}$

$\overline{QS}(n) = 2 + 2(n-1)\ln n \simeq 1.44 n \log n$

Γενικά κόστος $\Omega(n \log n)$

Δέντρο αποφάσεων: ∀ κόμβος του έχει 2 παιδιά.
(κάθε σύγκριση)

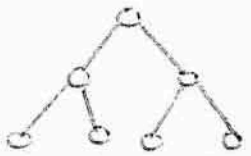
	1	2	3	4
π.χ.	3	10	2	1
	4	3	1	2



$2^h = n!$

$\Downarrow$

$h \geq \log_2 n!$



T

$h = 2$ (ύψος → αριθμός βημάτων που κάνει ένας αλγόριθμος)

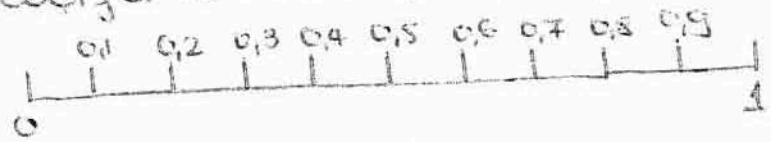
$\simeq \log n^n = n \log n$ (για αρκετά μεγάλα n) ← ΠΧΠ

Υβρίδιση Ταξινόμησης ←

$x \in [0,1)$

π.χ. [0,205, 0,101, 0,2, 0,47, 0,52, 0,78, 0,59]

χωρίζει το διάστημα [0,1) σε υποδιαστήματα



Σε ∀ υποδιάστημα αντιστοιχίζουμε ένα κάδο.
Παίρνουμε 1-1 τον στοιχείο της εισόδου και τα βάζουμε στους κάδους.

Ταξινομούμε ∀ κάδο (χρησιμοποιώντας insertion sort).

(ενώνουμε μετά κάθε κάδο).

24/11/2005

Um conjunto probabilístico dados com códigos bem [illegible]

[illegible]

0,101
0,2
0,205
0,47

⋮

O código variável de alguns vai cair

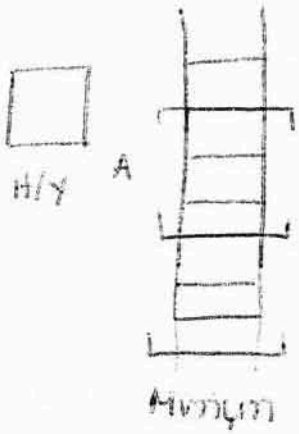
Discos e leitores M

O(M) código

O(M) código em fém redundância
(∀ código redexei estação aqueles existem)

25/11/2005

Trees
↳ Montagem [illegible]



TM → Turing Machine

RAM ⇒

$(TM)^3 \equiv RAM$

Página
Programática
arrays = A [1. . .100]
B [1. . .300]

O conceitos com arrays variável to dentro

AVL - Tree = Adelson Velsilly
Landis

(1) AVL - tree → 1961
(2) B - tree → 1962
(3) BB - tree → Tenjam
(4) IST - tree → MC/TS
(5) BB(α) → Reingold
Nievergelt
Bkam/MC

Median

nlogn

01/12/2003

External Sorting (SOS)

N : μέγεθος εισόδου

M : -||- μνήμης

2p ταινίες

1. SORTING AND MERGING EXAMPLE

→ αγνοούμε τα κενά

M = 3

2 · 3 = 6 ταινίες

1η φάση : (ASC) → κύριο πρόβλημα ταξινόμηση → ACS

1η ταινία : ACS DMN AEX LMP RTI → IRT NDM → DMN

2η -||- : IRT EGR

3η -||- : AGN GIN

μπορούν να χρησιμοποιούνται μόνο τις 2p ταινίες για να γράφω.

2η φάση : Διαχωρίζουμε τις ταινίες σε ταινίες εισόδου και σε ταινίες εξόδου. Στη συνέχεια, συγχωνεύουμε.

4 AAGINORST (Διαβάζουμε το 1ο στοιχείο από κ ταινία)

5

6

Μνήμη :

~~AIAG~~ OIN → INO

AAI (ταξινομημένο + το μεγαλύτερο στην ταινία 4)

Διαγράφω το A και τοποθετώ στη θέση του, ένα άλλο στοιχείο και αυξάνει στην ίδια ταινία με το A.

Ξαναπαίρνω μέχρι να τελειώσουν τα block.

3η φάση : Συνεχίζουμε για τις ταινίες 1, 2, 3.

1η φάση : $\lceil N/M \rceil$ runs

2η -||- : 1ο ζεύγος → $\frac{\lceil N/M \rceil}{p}$ runs

2ο -||- → $\frac{\lceil N/M \rceil}{p^2}$ -||-

⋮

k -||- → $\frac{\lceil N/M \rceil}{p^k} = 1 \Rightarrow k = \log_p \lceil N/M \rceil$

Replacement Selection EXTERNAL SORTING
M=5 (Knuth [illegible])

Heap	Tiger	Next	Output
EXT	E	E	E
EXT	E	R	EE
RXT	R	N	EER
N*XT	T	A	EERT
N*XA*	X	L	$\boxed{\text{EERTX}}$
N*L*A*	A*	S	A*
N*L*S*	L*	O	A*L*
N*O*S*	N*	R	A*L*N*
R*O*S*	O*	T	A*L*N*O*
R*T*S*	R*	I	A*L*N*O*R*
IT*S*	S*	N	A*L*N*O*R*S*
IT*N	T*	G	$\boxed{\text{A*L*N*O*R*S*T*}}$
IGN			$\boxed{\text{GIN}}$

Δεν χρειαζόμαστε *, σύμφωνα με αυτό το στοιχείο

1η τελικά

Κάθε το αντίστοιχο, βάζω * στο ήδη υπάρχον.

Δεν παίρνουμε το στοιχείο που δεν έχει *.

→ 2η τελικά

→ 3η -||-

MEDIAN

Δίνεται Μ: |M|=n

i-στο ήδη υπάρχον στοιχείο του

$i = \frac{n}{2}$

1η λύση: ταξινόμηση O(nlogn)

2η -||- : χπ: O(n²) μπ: O(n)

3η -||- : χπ: O(n)

3η λύση → Find

1. Διαχωρίζεται Μ βάση τυχαίου S, $M_1 = \{x \in M, x < S\}$

$M_2 = \{x \in M, x > S\}$

M_1	S	M_2
<		>

4/12/2003

```
{
if (|M1| = i-1) return S;
else if (|M1| = i-1)
    find (k=|M1|-1, Mα)        (CAR Hoare)
else find (i, M1)
}
```

$i = 1$

$|M| - 1$

$T(n) = n+1 + T(n-c) = n+1+n-c+1 + T(n-2c) = \dots = O(n^2)$

$T(n) = \max_i T(n,i)$

$T(n) \le cn$

κόστος για να βγάλεις το i-οστό στοιχείο στο σωστό κομμάτι → σταθερός για το δεξί κομμάτι

$$T(n,i) \le cn + \frac{1}{n}\left[\sum_{k=1}^{i-1} T(n-k, i-k) + \sum_{k=i+1}^{n} T(k-1, i)\right]$$

κόστος για διαχωρισμό

πιθανότητα το στοιχείο που διαλέγουμε να μείνει για από τις n θέσεις

$$T(n) = cn + \frac{1}{n}\left[\sum_{k=1}^{i-1} T(n-k) + \sum_{k=i+1}^{n} T(k-1)\right]$$

$$= cn + \frac{1}{n}\left[\sum_{k=1}^{i-1} T(n-k) + \sum_{k=i}^{n-1} T(k)\right]$$

$T(n) \le c \cdot n \cdot 4$

$T(0) = 0$

$$T(n) \le cn + \frac{1}{n}\left[\sum_{k=1}^{i-1} 4c(n-k) + \sum_{k=i}^{n-1} 4ck\right] \le cn + \frac{4c}{n}\left[\underbrace{n(n-1) - \frac{(n-i+1)(n-i)}{2}}_{\le 4n.} - \underbrace{\frac{i(i-1)}{2}}_{\le 4n.}\right] \le cn + \frac{4c}{n}\left[n(n-1) - \frac{n(n-1)}{4}\right] \le cn + \frac{4c}{n}\cdot\frac{3n^2}{4} \quad i = \frac{n+1}{2}$$

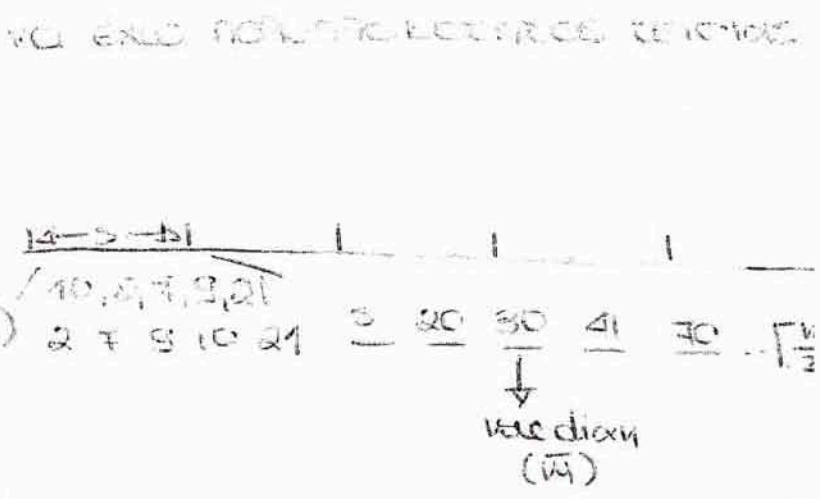
Τον quicksort τον υλοποιούμε αναδρομικά και για τις δύο υποδιαστάσεις ενώ ένα τελικό διάστημα τον καλούμε μια φορά αναδρομικά γι' αυτό και βγαίνει $O(n)$ πολυπλοκότητα.

1/12/200

$T(n) = bn + T(n-1)$
$T(n) = bn + T(n/2)$ } Πρέπει να ορίσουμε ένα κριτήριο διαχωρισμού
$T(n) = c = 10$
9 Πρόβλημα να βρω πρόγνωσεις τιμές
λόγω

Select (i, n)

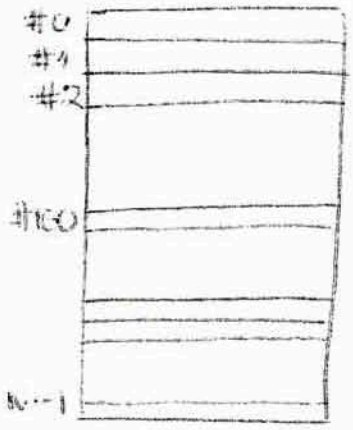
1. Κάνε καλό διαχωρισμό



2. if (i > |M1|) Select (i − |M1|, M2)
   else select (i, M1)

2/12/2003

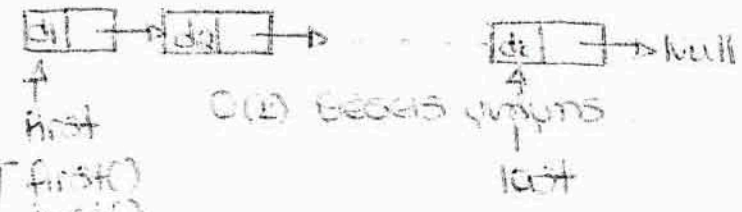
- Βασικοί και δομημένοι τύποι στοιχείων
- Πρόβλεψη του μεγέθους
- Στατικές δομές Δ.Δ.



- Βασικοί τύποι δεδομένων { integer, real, char, pointers }
- Πίνακες (Arrays) A[1...n]
  A[1,5]
- Εγγραφές (Records) → Pointer Machine
  struct { integer, pointer }

  Μπορεί να εκτελεί μόνο ορισμένες συνθήκες πρόσβασης αυτών των δεδομένων.

- Λίστα (List)



O(1) βήματα γιατί

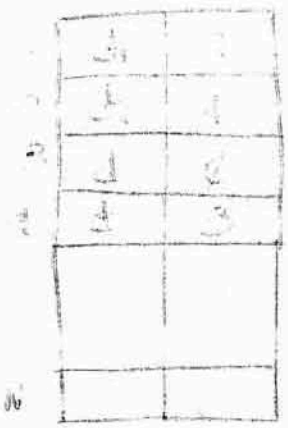
* σταθερός αριθμός πράξεις

[first()
  last()
  size() → σε σχέση με το μέγεθος επιστρέφει
- insert_after(pointer, data)
  find(data) O(n) πράξεις
  concate_with(list) → O(n) (σταθερός αριθμός πράξεων)
  → συνένωση λιστών delete(pointer)

2/12/2006

Armazenamento Alternativo → (1) Matrizes (2) Ligações

(1)



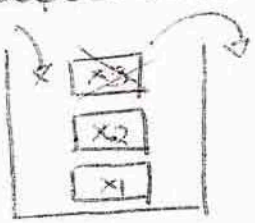
(2)

struct {
data
pointer
}

→ Árv ∃ procura de as ligações.

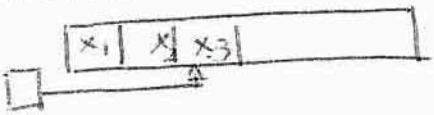
Listas Ligadas

• Pilhas (stack)

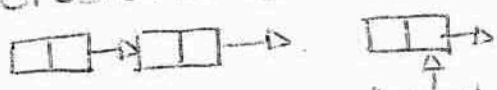


push(data)
pop()
top() → Certas coisas
is_empty()

• Matrizes

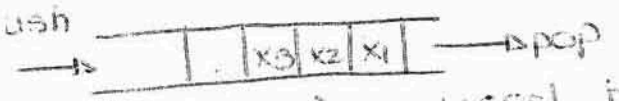


• records (Ficheiros)



insert_after(last, data)
delete(last)
last()
size()

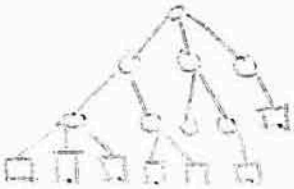
• Fila (Queue)



push(data)
pop()
is_empty()

insert_before(first, data)

• Árvores (Trees)



nós

• E < N − 1

Letras

$T = (N, E)$ $E = N - 1$

$T' = (N+1, E')$

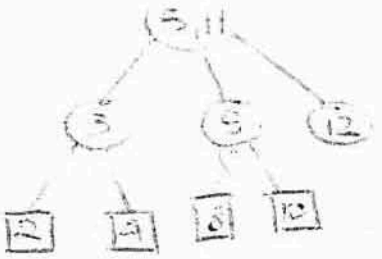
$T_1 = m$

$T_2 = N+1-m$

2/12/20

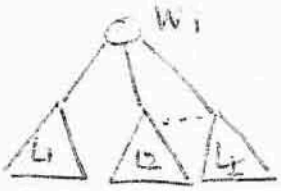
Δέντρο → ιεραρχία – πολυεπίπεδη ιεραρχία

π.χ.

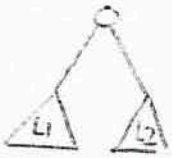


{2,4,8,10,12}

- preorder
  - → W1
  - → L1, L2, ...

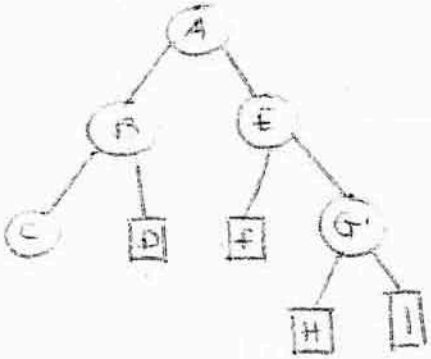


- postorder
  - → L1, L2, ...



- symmetric order
  - L1
  - W1
  - L2

π.χ.



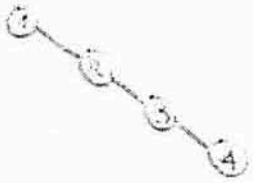
preorder: ABCDEFGHI

postorder: CDBFHIGEA

symmetric order: CBDAFEHGI

search → ψάχνω το δεύτερο για ένα τριών

1 2 3 4



→ το δέντρο εξελίσσεται σε λίστα λόγω κακού συγχρονισμού

- Abstract data types → έχει κάποιες ιδιότητες

concrete – " – – " –

2/12/2009

Access(x) : $x \in U$, $m(x)$ → ΔΔ static

Insert(x) : $x \in U$: $S \leftarrow S \cup \{x\}$ →  →  ΔΔ δυναμικές

Delete(x) : $x \in S$: $S \leftarrow S - \{x\}$

ΔΔ → εφήλεση

→ διαχρονική (persistent) → μπορούμε να δούμε τα προηγούμενα της (καταστάσεις) → κόστος σε χώρο.

persistent → partial (μπορούμε να ελέγξουμε την ουρά + να δημιουργούμε της προηγούμενες)

→ fully

ΔΔ → comparison based

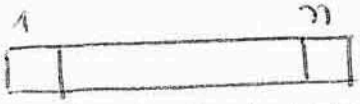
→ representation -||-

ΔΔ → explicit $O(n)$

→ implicit $n + O(1)$ χώρος που χρειαζόμαστε

$S \subseteq U$, $|S| = n$ $x \in S$

σταθερή συνάρτηση δείκτη

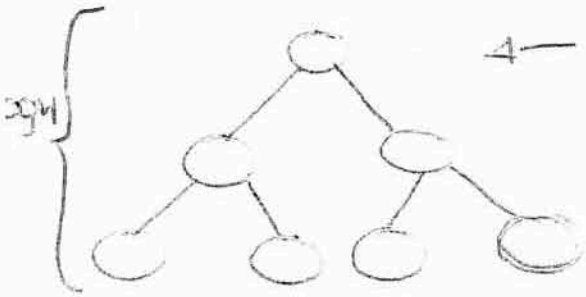


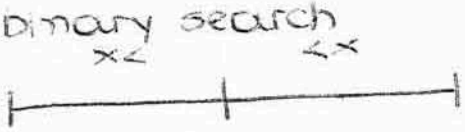
$x_{i-1} \le x_i \le x_{i+1}$

$O(n)$ → στην χειρότερη περίπτωση

Access(x)

• binary search





$T_n = \frac{1}{2} T_{n-1}$

$n \left(\frac{1}{2}\right)^h = 1 \rightarrow n = 2^h$

$\boxed{h = \log_2 n}$ χπ

$n \to n/2 \to n/4 \to \dots \to 1$ (τέλος του χ.π.)

$\frac{\lceil x - x_1 \rceil}{x_n - x_1}$

[illegible]

[illegible]

1. [illegible]

2. if c (|M1| < c) [illegible]

   τοποθετούμε εκεί πάρα c-στο

3. Βρίσκω ένα κατάλληλο στοιχείο S

5. Βάζω S χωρίζοντας σε $M_1 = \{x \in M, x < S\}$

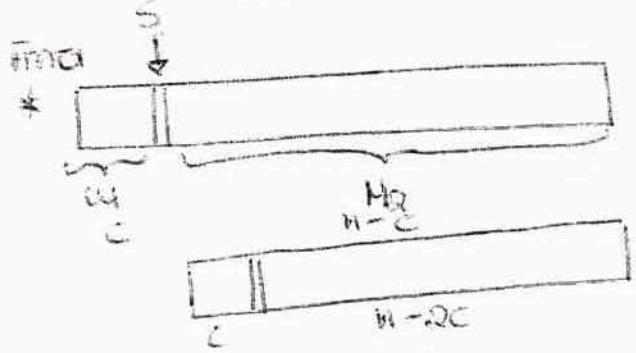
$M_2 = \{x \in M, x > S\}$

4. if (i ≤ |M1|)

   Select (i, M1);

   else

   Select (i − |M1|, M2);



$T(n) = n + T(n-c) = n + n - c + T(n-2c) = \cdots = O(n^2)$

$T(1) = 0$

1: [4 5 1] [ ] [ ] [ ] [ ]

100 90 47 12 21 20 50 71 42 → $\bar{m}$

Χωρίζω το array σε πεντάδες
Τοποθετώ το στοιχείο και
βρίσκω το median

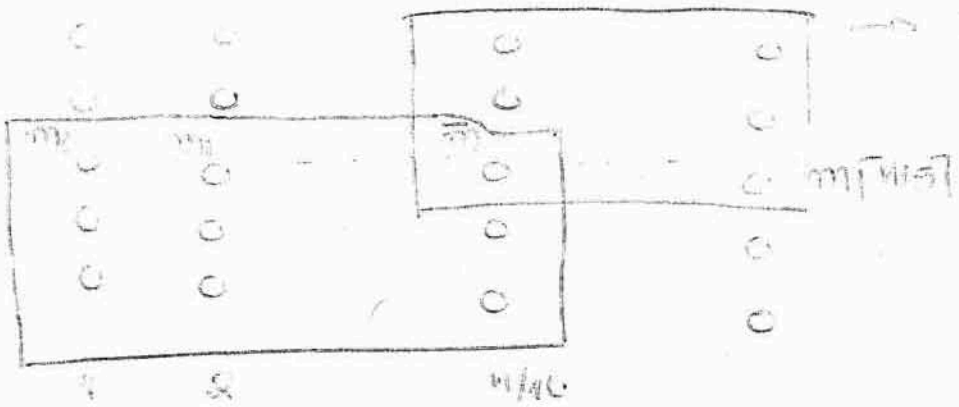
12 21 30 47 100

m0 m1 m2 m⌈n/5⌉

$\bar{m} = \text{Select}\left(\lceil \frac{n}{10} \rceil, (m_0, m_1, \ldots, m_{\lceil n/5 \rceil})\right)$

→ βρίσκω το μέσο των μέσων (πάλι καλός διαχωρισμός)

Βάλτε τις πεντάδες στις στήλες:



→ Στο σχέδιο υπάρχει ένα σύνολο μικρότερο από το μισό στοιχεία πάνω από το $\bar{m}$.

Τα στοιχεία του σχεδίου είναι γραμμένα από τον median.
Το $\bar{m}$ είναι ο median των median

$\frac{3n}{10}$, $n \bmod 10 = 0 \Leftrightarrow 10 \mid n$

$\frac{3n}{11}$ στοιχεία όταν $n \bmod 10 \neq 0 \Leftrightarrow 10 \nmid n$
(λιγότερα από το διαχωρισμό)

$\frac{3n}{11}$ στοιχεία το πολύ μεγαλύτερα από $\bar{m}$

Η χειρότερη περίπτωση είναι όταν καλούμε n select για το μεγαλύτερο από τα 2 κομμάτια. (δηλ. $\frac{8n}{11}$)

$$T(n) = \begin{cases} an & , n \le 100 \\ bn + T\left(\frac{8n}{11}\right) + T\left(\frac{21}{100}n\right) & , n > 100 \end{cases}$$

διαχωρισμός, καλούμε select για βρούμε το median των median

$T(n) \le cn$

- για $n \le 100$ προφανές
- για $n > 100$ $T(n) \le \underbrace{T\left(\frac{8n}{11}\right)}_{\le c \cdot \frac{8n}{11}} + \underbrace{T\left(\frac{21n}{100}\right)}_{c \cdot \frac{21n}{100}} + bn \le c\frac{8n}{11} + c\frac{21n}{100} + bn =$

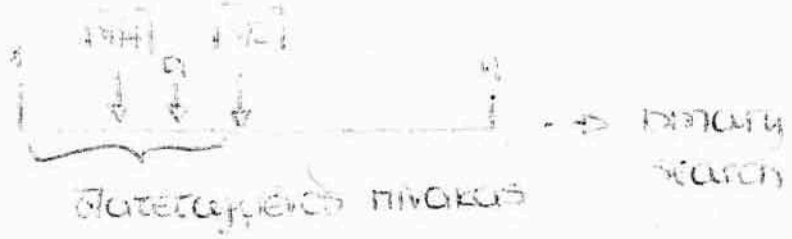
$cn\left(\frac{8}{11} + \frac{21}{100} + \frac{b}{c}\right) \le cn$

↓ πρέπει

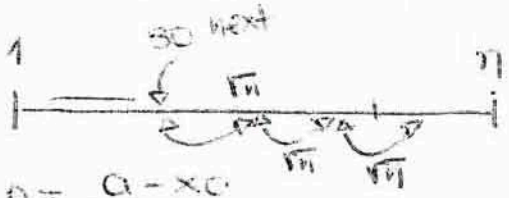
$\frac{8}{11} + \frac{21}{100} + \frac{b}{c} \le 1 \Rightarrow \boxed{c \ge \frac{1100b}{69}}$

Επιλέγουμε τελικά
$c = \max\left(\frac{1100b}{69}, a\right)$

Interpolation search



1. $a \le A[n/2]$
2. $a \le A[n/4]$ ox1



ψάχνω το a

$$p = \frac{a - x_0}{x_{n+1} - x_0}$$

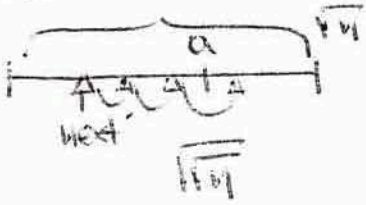
$next = \lceil p\, n \rceil$

$a \ge A[next]$

$a \ge A[next + \sqrt{n}]$

Συνεχίζω έως $a \le A[next + i \cdot \sqrt{n}]$

$A[next + (i-1)\sqrt{n}] \le a \le A[next + i\sqrt{n}]$



Μέση Πολυπλοκότητα

$\log\log n$ $\quad n = 2^{32}$ $\quad \log\log \simeq 5$

c = # επιχειρήσεων έως να βρεθεί υποσύνολο $\sqrt{n}$ $\quad P_i = Pr\{$χρειάζονται $\ge i$ επίπεδα$\}$

$$c = \sum_{i \ge 1} (i \cdot Pr\{\text{καλώ } i \text{ φορές}\}) = \sum_{i \ge 1} i(p_i - p_{i+1}) = p_1 - p_2 + 2p_2 - 2p_3$$

$$+ 3p_3 - 3p_4 + \dots = \sum_{i \ge 1} p_i$$

$p_1 = p_2 = 1$

$p_i \le Pr\{|\text{σέση } a - next| \ge (i-2)\sqrt{n}\}, i \ge 3$

$Pr\{|x - \mu| \ge \varepsilon\} \le \frac{\sigma^2}{\varepsilon^2}$ ανισότητα Chebyshev

Τυχαία μεταβλητή $X = \{$αριθμός $x_j \le a\}$

$Y = \{$αριθμός ίδιων 4 ή 5$\}$

$Pr(Y=2) = \binom{5}{2} (2/6)^2 (1 - 2/6)^{5-2}$ → ομοιόμορφη

$P(x=j) = \binom{n}{j} p^j (1-p)^{n-j}$ $\quad k = c \cdot n = next$

$p = \frac{x_{i+1} - x_0}{x_{n+1} - x_0}$ $\quad$ (αναμενόμενο πλήθος)

$\sigma^2 = p(1-p)n$

(τετραγωνικό σφάλμα)

$p \le \frac{p(1-p)n}{(1-2)^2 n} \le \frac{1}{4(1-2)^2}$ εφόσον $p(1-p) \le \frac{1}{4}$, $p \in (0,1)$

└ το μέγιστο κείμενο το $\frac{1}{4}$

$\sum_{i \ge 1} c_i = 2 + \sum_{i \ge 3} c_i = 2 + \sum \frac{1}{4(i-2)^2} = 2 + \frac{\pi^2}{24} \simeq 2.4$

$\Rightarrow \boxed{c \simeq 2.4}$

$\bar{T}(n) = c + \bar{T}(\sqrt{n}) \simeq 2 + c \cdot \log\log n = O(\log\log n)$

c: κόστος για βρες το κατάλληλο για το αρχείο του αναζητούμενου

$\bar{T}(\sqrt{n})$: εφαρμόζω αναζήτηση

χειρότερη περίπτωση $\rightarrow \sqrt{n}$

9/12/2003

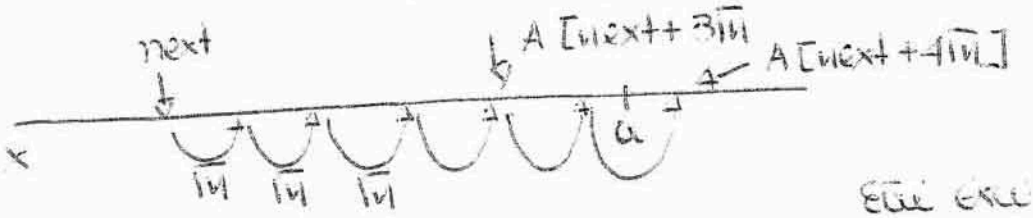
Interpolation Search

ΠΜΠ: $\bar{T}(n) = O(\log\log n)$ για ομοιόμορφα κατανεμημένα στοιχεία

3 5 7 9,5 11

5 15 20 25 26 26.2 26.5 27 27.3 28 32 37 39 41

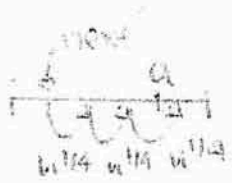
ΠΧΠ: $next = \lceil pn \rceil$ (αρχικό εκτίμηση)



εδώ έχω $i = 4$.

$A[next + (i-1)\sqrt{n}] \le a \le A[next + i \cdot \sqrt{n}]$

$$\boxed{i \le \frac{m}{\sqrt{n}} = \sqrt{n}}$$



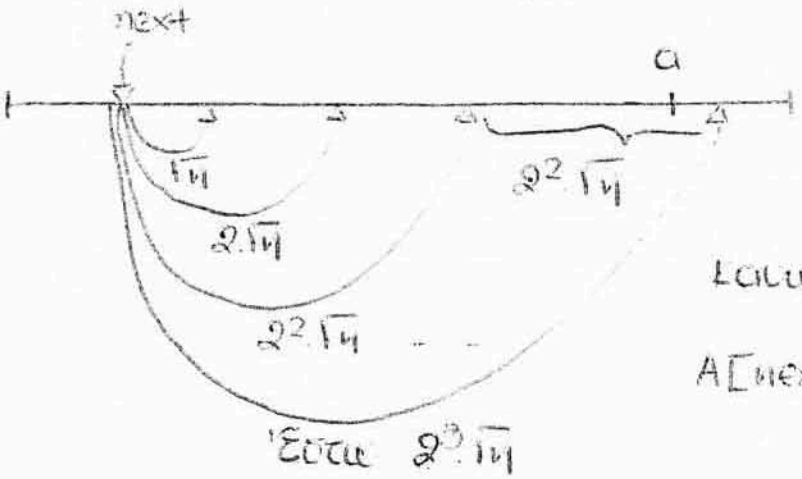
Άρα υπάρχει $\sqrt{n}$ σημεία στις ακμές $\sqrt{n}$.

Άρα πχπ: $n^{1/2} + n^{1/4} + n^{1/8} + \dots = \sqrt{n} = O(\sqrt{n})$

στο αρχικό κομμάτι — $2^{\text{ο}}$ σταθό — $3^{\text{ο}}$ σταθό

Σε γεν. περίπτωση, το $\sqrt{n}$ περιορίζεται με το $\lceil \sqrt{n} \rceil$

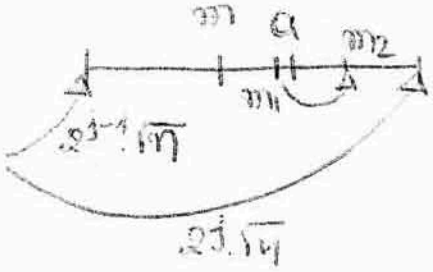
Σταματάμε στη διάσχιση του αλγορίθμου, όταν φτάνουμε σε μικρό σταθερό πλήθος.



Υπολογίζω στη αρχική ελπίδα

Κάθε $2^j.\sqrt{n}$ βήματα

$A[next + 2^{j-1}.\sqrt{n}] \le a \le A[next + 2^j \sqrt{n}]$



πηγαίνω στη θέση και βλέπω $A[m] \le a$.
Στη συνέχεια, βρίσκω τη θέση του δεύτερου σταθμόνος. $A[m_1] \le a$. $A[m_2] \ge a$.
Ψάχνω μέχρι $|m_2 - m_1| \le \sqrt{n}$

Το σύνολο η ελεύθερη αναζήτηση

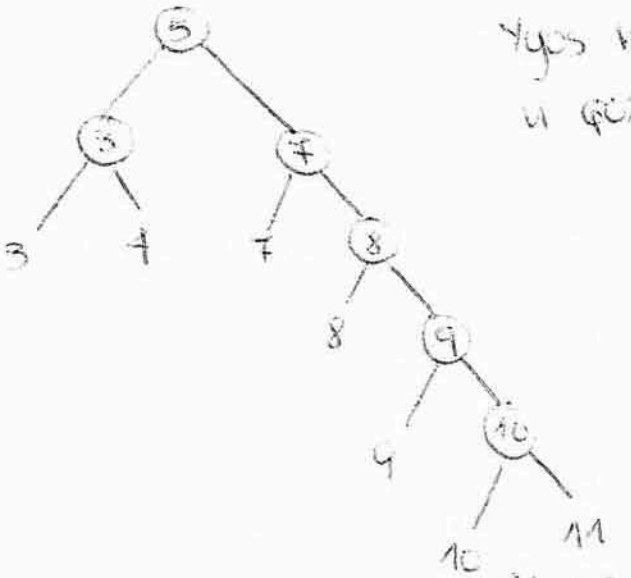
Έστω 2^j με j ελάχ. ώστε ο φάκελος στη θέση $2^j \sqrt{n} \le m$ $\Rightarrow j = O(\log_n)$

Σε σειρά : $2^{i-1}\sqrt{n}$

το πλήθος $\log(2^{i-1})$ ελάχιστα [illegible] $\Theta(\log n)$

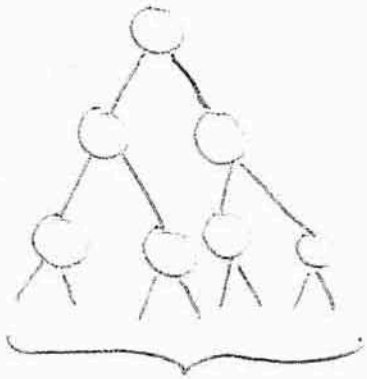
$$\log n + \log n^{1/2} + \log n^{1/4} + \dots = \log n\left(1 + \frac{1}{2} + \frac{1}{4} + \dots\right) = O(\log n)$$

AVL δέντρα



Ύψος n: $h \geq \log n$

n φύλλα



2^3 φύλλα

$n=3$

Δυαδικά δέντρα n φύλλων → Υψοζυγισμένα π.χ. AVL, Red-black,

→ Βαθοζυγισμένα B-δέντρο, (a,b)-δέντρα

Ύψος : $\Theta(\log n)$

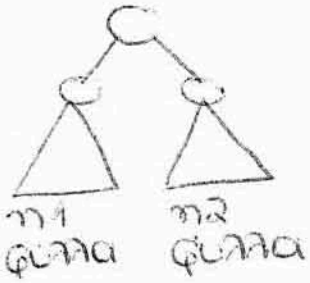
2	10	17	32	41	52	68

22 32 41 52 68

Βάθος → αριθμός φύλλων

Πλεονέκτημα

Αναζήτηση $O(\log n)$



n1 φύλλα n2 φύλλα

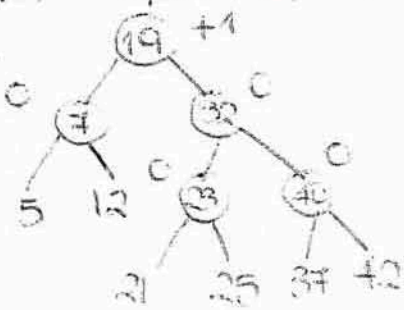
Εισαγωγή

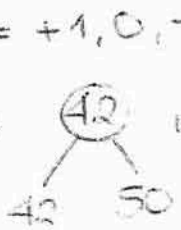
Διαγραφές? $O(n)$ → χειρότερη περίπτωση

↳ $O(\log^2 n)$ αλλά search $O(\log^2 n)$

Ύψος ζυγού v : $hb(v)$ = Ύψος δεξιού παιδιού(v) − Ύψος αριστερού παιδιού(v)

- $hb(v) = +1, 0, -1$ ειστρεπτές τιμές π.χ.



Αν είχα  , τότε δεν έχουμε έγκυρο δέντρο

αλλιώς $hb(\text{ρίζας}) = 2$.

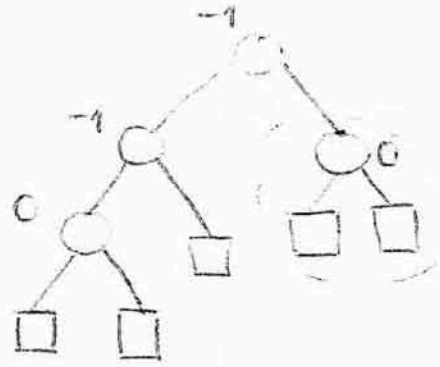
9/12/2011

Δυαδικά Τελεστικά AVL Δέντρα

↳ Τυχαίες Ανάλυση

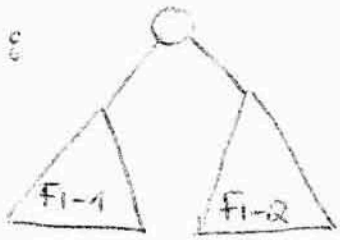
F_0 : □ (1 φύλλο)

F_1 :  (1 κόμβος + 2 φύλλα)

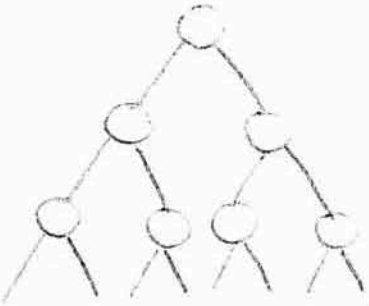
F_2 : 

π.χ. F_2 : → F_1 } F_3

Είναι AVL δέντρα

F_i : 

Έχουν περισσότερα κόμβοι με βάθος -1



→ καλύτερη περίπτωση AVL δέντρου (πάνω)

Πάνω δυαδικό δέντρο n φύλλων : $2^h = n \Rightarrow h = \log n$
(καλύτερη περίπτωση)

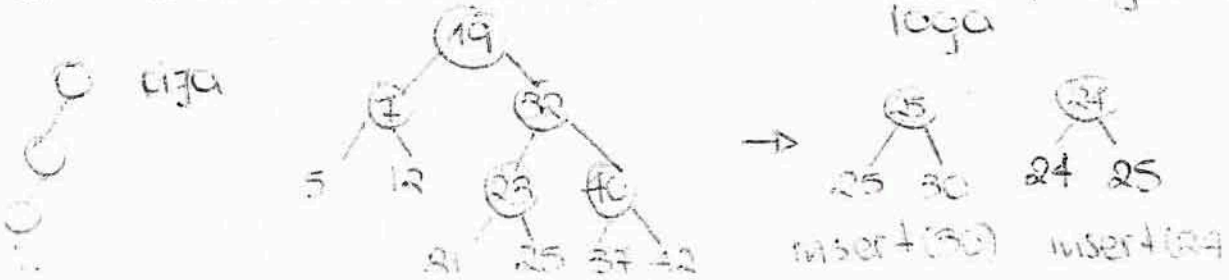
- F_h έχει ύψος h → φ_{h+1} φύλλα → αριθμοί Fibonacci

π.χ. F_3 → ύψος = 3 $\qquad \varphi_{h+1} = \varphi_h + \varphi_{h-1}$

$\varphi_h = \dfrac{\alpha^{h+1} - \beta^{h+1}}{\sqrt{5}}$ (1) όπου $\alpha = \dfrac{1+\sqrt{5}}{2}$ και $\beta = \dfrac{1-\sqrt{5}}{2} \leq 1$

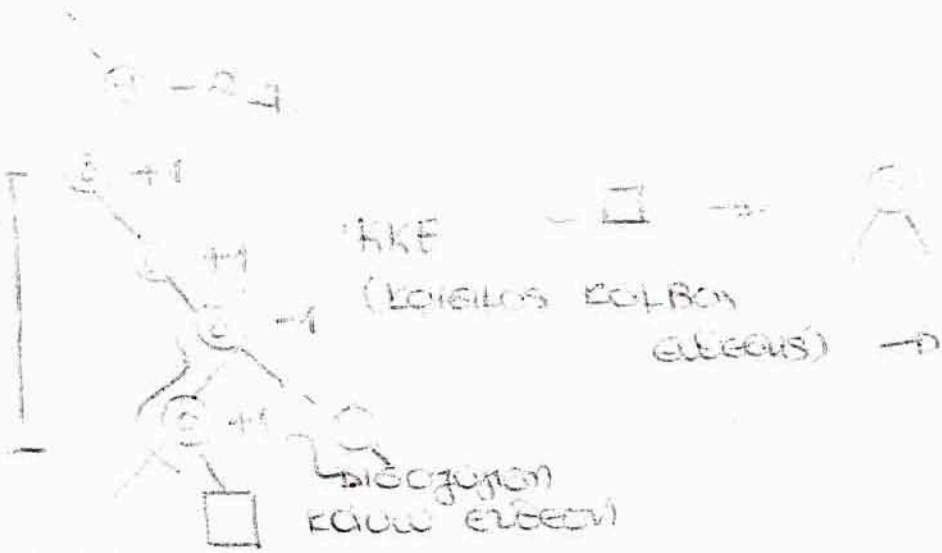
Έστω $\varphi_{h+1} = n \overset{(1)}{\Rightarrow} \dfrac{\alpha^{h+2} - \beta^{h+2}}{\sqrt{5}} = n \leq \dfrac{\alpha^{h+2} - 1}{\sqrt{5}} \leq n \Rightarrow$

$\alpha^{h+2} \leq \sqrt{5}\, n + 1 \leq \sqrt{5}(n+1) \Rightarrow h \leq \dfrac{\log \sqrt{5}}{\log \alpha} + \log(n+1) - 2 = O(\log n)$



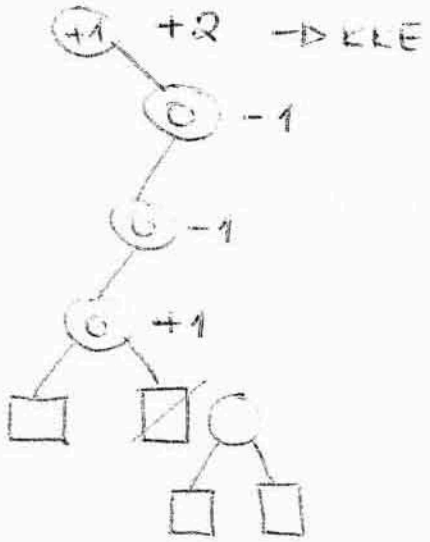
Πιο ισοζυγή δέντρο

9/12/200



→ ΕΑΝ Ο ΑΛΛΟΣ ΚΟΛΒΟΣ ΤΟΥ ΠΑΤΕΡΑ ΕΙΝΑΙ Η ΖΩΓΙΑΝΗ ΚΕΝΟΡΑΣ ΑΕΙΑΝΑΙ ΕΔΩ ΜΠΑΙΝΕΙ ΕΔΩ ΟΡΙΖΟΝ

π.χ.



15/12/200

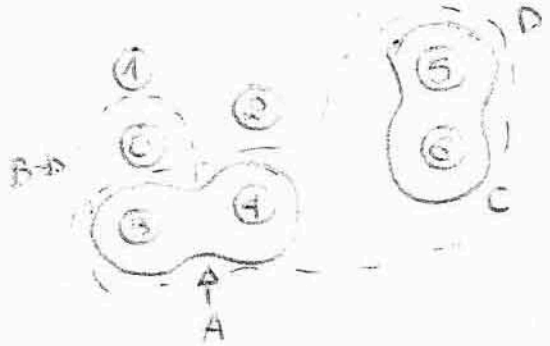
Union Find

Ζέγιαν $U := [0, \dots, N-1]$ ουσι N στοιχεία

Find(x): επιστρέφει το σύνολο του x

Union(A,B,C): ενώνει A και B και C

π.χ.



Union(3,4,A)

Union(A,C,B)

Find(C) = B

Union(5,6,C)

Find(C) = D (Δίνουν το πρόγονέο σύνολο + ζητούνε τα υπόλοιπα)

Union(B,C,D)

Χρησιμοποιούμε ένα array και τον ονομάζουμε 15-1N

B	N	2	B	B	C	C	E
0	1	2	3	4	5	6	7

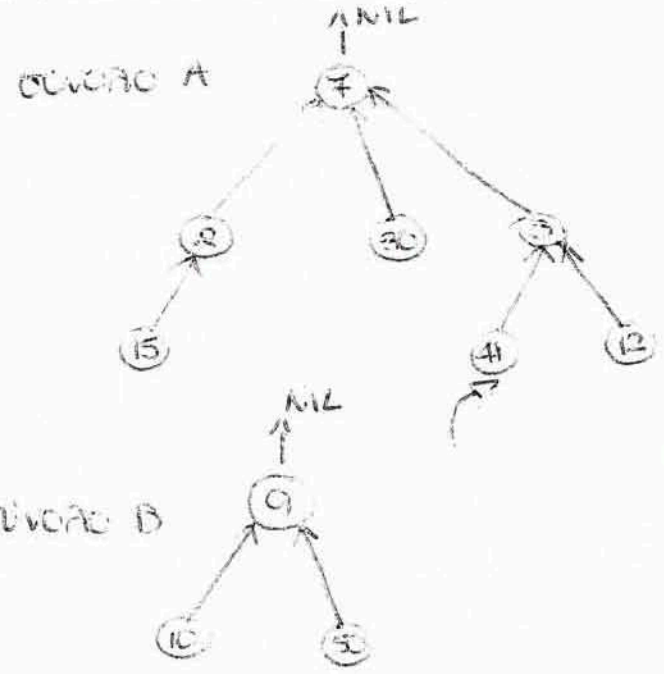
N-1 → Δίνει τα σύνολα και ελέγχει στο ίδιο array + στοιχεία

15/12/200

Abordagem n-1 Union → custos

Internamente custos 1 Union → $\frac{\text{custos } n-1}{n-1} = O(\log n)$

Em alguns livros aparecem este union

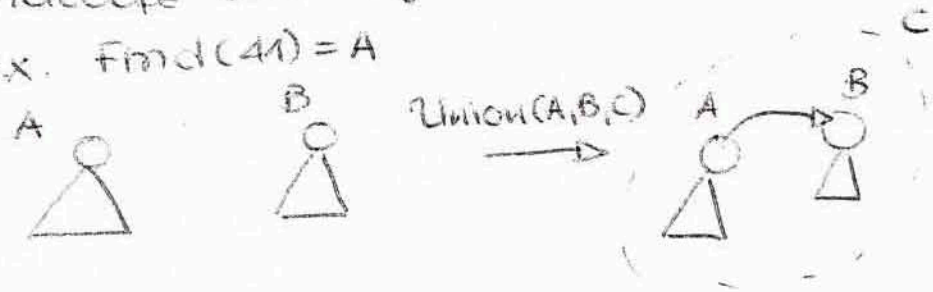


Automaticamente a árvore pequena. Coloca conforme a outra grande

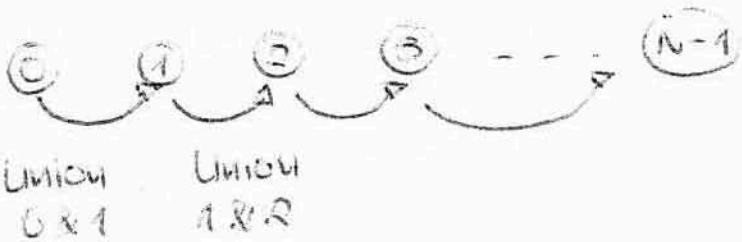
Automaticamente na outra as pequena com comprimento todos pelos (na verdade deixam cada vez menos)

Na hora de fazer find no 41, percorre até o 41 e encontra em qual árvore está no outro tms.

ex. find(41) = A



Coloca em qual árvore a A na pequena o B.

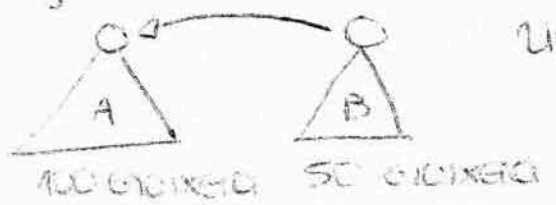


Pequeno aresta ano corpos (mais antigo pequeno)

find = O(N) → custos

Tem maneira de melhorar?

Weighted Union Rule



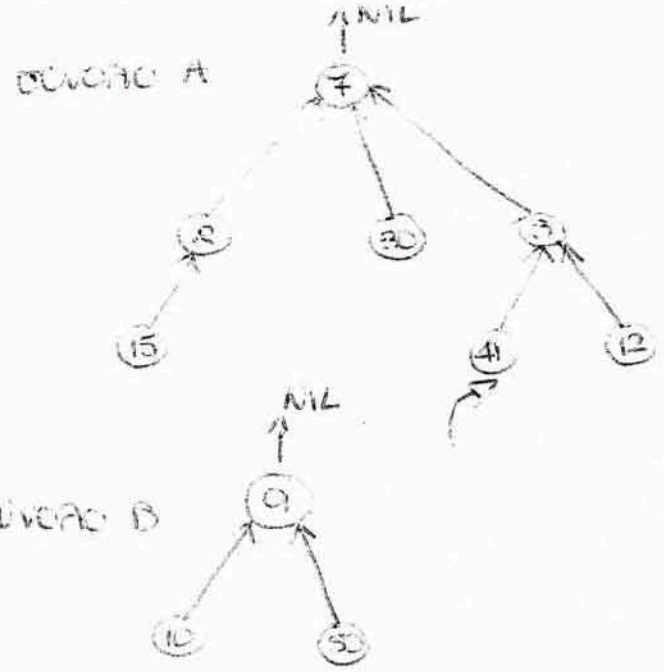
Union(A,B)

A árvore de menor altura pequena o árvore de maior altura.

Αλγόριθμος n-1 Union → [illegible]

[illegible] κόστος 1 Union → [illegible]

Μη ισοζυγή δένδρα στο Union

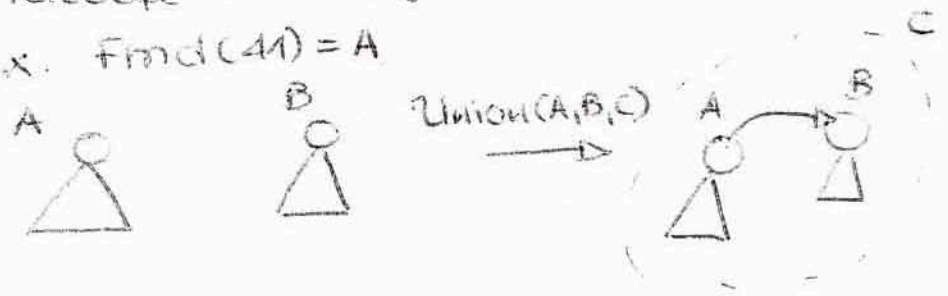


Ανάποδο δένδρο. Ένα φύλλο ρίζα

Ανάποδο δένδρο και όλοι τους δείκτες (τα ένα κοντέρα)

Για να κάνουμε find το 41, βλέπουμε ότι φτάνουμε στην ρίζα ότι κάνουμε το ίδιο

π.χ. find(41) = A



Κάνει A για

Union 0 & 1

Union 1 & 2

Φτιάχνει κόμβους μέχρι Ν-1

find = O(N) → [illegible]

Πώς μπορούμε να το αποφύγουμε?

Weighted Union Rule

Union(A,B)

Αν [illegible] σε [illegible]

Το ύψος αυξάνεται με το 1 [illegible]

Find [illegible]

Union(A,B)

C [illegible]

D |A| |B|

Αν [illegible] Ανάπ

Union (A,B,C)

1) List [Z] = { [illegible]

π.χ. List [B] = { [illegible]

List [C] = { [illegible]

2) size (Z)

3) Ενημέρωσε [illegible]

MAPOUT & [illegible]

MAPIN & [illegible]

IS-IN	C	1	2
	B	1	2

Για να κάνω Union [illegible] ρίζα του [illegible]

MAPOUT[B] = D

MAPIN [D] = B

Find(x) = MAPOUT [illegible]

π.χ. Find([illegible]

Union(A,B,C) {

if (size(A) = size [illegible]

= ← MAPIN [A [illegible]

} else { = ← MAP [illegible]

forall x ∈ List ([illegible]

IS-IN (x) = y;

List [y] = List [x [illegible]

size (y) = size([illegible]

15/12/2

U-F με weighted union rule + path compression
[illegible]

$O(m \cdot \underline{a(m,n)}) \rightarrow$ συνάρτηση Ackermann

συνάρτηση 2 μεταβλητών

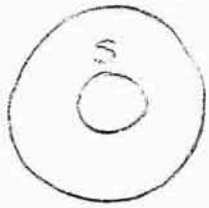
$A(n,1) = 2^{2^{\cdot^{\cdot^{2}}}}\}$ n φορές

$a(m,n) \approx A^{-1}(m,n)$ όπου $\log a(m,n) < 2^{2^{\cdot^{\cdot^{2}}}}$ $a(m,n) \leq 4$

12/1/2004

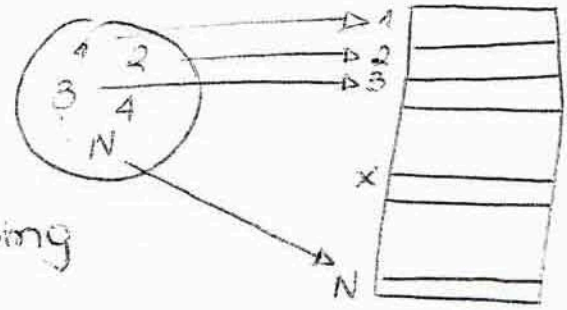
Κατακερματισμός (Hashing)

U (σύμπαν)



insert(x) : $x \in U$: $S = S \cup \{x\}$
delete(x) : $S = S - \{x\}$
access(x) ή search(x)

Έστω $|U| = N$

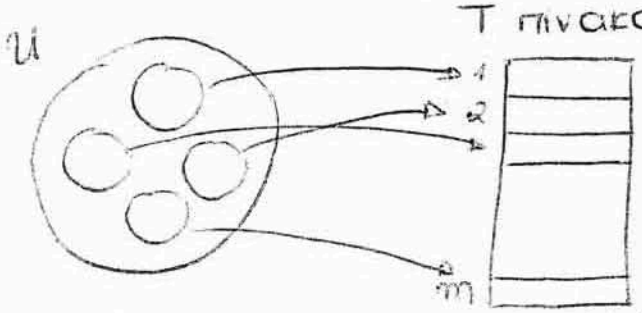


direct addressing

(+) insert $O(1)$
delete
search

(-) μπορεί να μη δεσμεύει η μνήμη
$|S| << |U|$ σπατάλη μνήμης

Έστω ότι θελήσουμε μια περιορισμένη μνήμη
T πίνακας κατακερματισμού



$h : U \rightarrow \{0, \ldots, m-1\}$
συνάρτηση κατακερματισμού

Εάν το U μπορεί σταθερά από δύο στοιχεία, τότε έχουμε συγκρούσεις (collision).

15/12/2003

Ορίζω weight(x) = # απογόνων x και rank(x) = ύψος δένδρου ρίζας x

1) weight(x) ≥ $2^{rank(x)}$

αποδείξη με επαγωγή

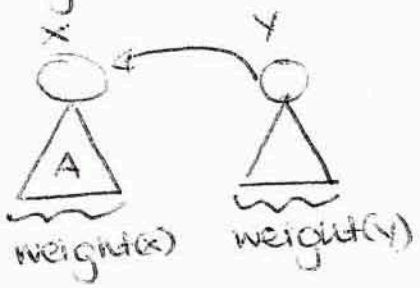
a) rank(x)=0 τότε $2^{rank(x)} = 1 =$ weight(x)

b) weight(x) = # απογόνων x πριν ενωθεί + # απογόνων y

$\geq 2 * weight(y) \geq$ (επαγ. υπόθεση)

$\geq 2 * 2^{rank(y)} \geq 2 * 2^{rank(x)-1} =$

$= 2^{rank(x)}$



2) rank(x) ≤ log n , n-1 unions

Έχω ήδη δείξει ότι $2^{rank(x)} \leq$ weight(x) ≤ n ⇒

rank(x) ≤ log n

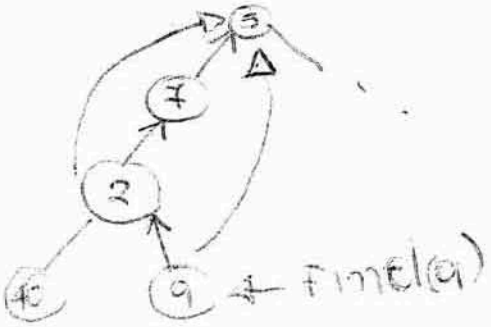
ρίζα με n-1 κόμβους

rank → αριθμός βάθους στον χειρότερο περίπτωση για εφαρμογή find.

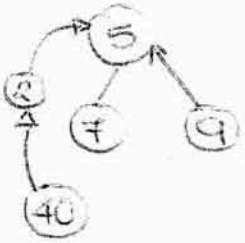
n-1 unions και m find

Συνολο : O(n+m log n) 15-11 και O(m+m log n)

Path compression



Ενώνουμε δύο δένδρα με μικρό ρίζα ⇒ το find



U-F με weighted union rule + path compression

m-1 union και m find [illegible]

$O(m, \underline{a(m,n)})$ → συνάρτηση Ackermann

συνάρτηση 2 παραμέτρων

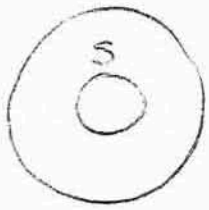
$A(i,j) = 2^{2^{\cdot^{\cdot^{2}}}}\}$ j φορές

$a(m,n) \simeq A^{-1}(m,n)$ όπου $\log a(m,n) < 2^{2^{\cdot^{\cdot^{2}}}}$ $a(m,n) \le 4$

12/1/2004

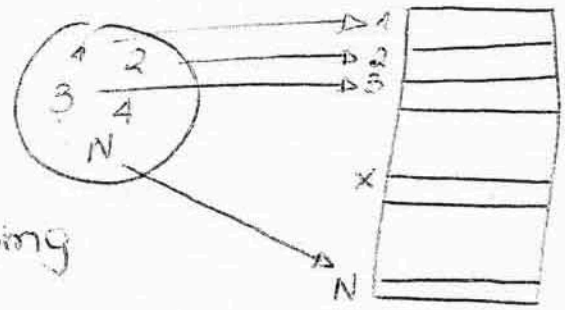
Κατακερματισμός (Hashing)

$\mathcal{U}$ (σύμπαν)



insert(x) : $x \in \mathcal{U}$: $S = S \cup \{x\}$
delete(x) : $S = S - \{x\}$
access(x) ή search(x)

Έστω $|\mathcal{U}| = N$



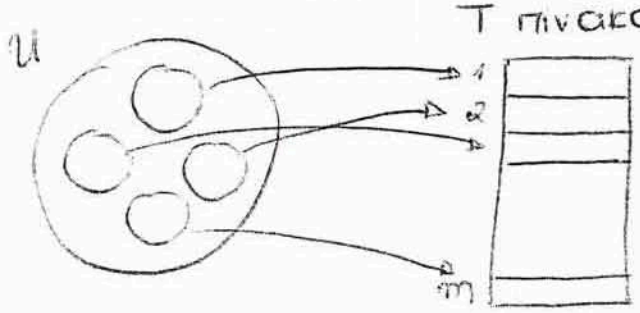
direct addressing

(+) insert $O(1)$
delete
search

(−) μπορεί να μη φτάνει η μνήμη
$|S| << |\mathcal{U}|$ σπατάλη μνήμης

Έστω ότι θέλουμε μια περιορισμένη μνήμη

T πίνακας κατακερματισμού



$h : \mathcal{U} \to \{0, \dots, m-1\}$

συνάρτηση κατακερματισμού

Εάν το u μπαίνει σε δύο θέσεις, τότε έχουμε σύγκρουση (collision).

12/11/2004

Κατακερματισμός. Έστω $U = \{0, 1, \dots, N-1\}$ και σύνολος T ($|T| = m$) και $h: U \to \{0, 1, \dots, m-1\}$ τότε $x \in U$ → $T[h(x)]$

Πχ. $m = 5$, $U = \{0, 1, 2, \dots, 99\}$

$S = \{3, 15, 20, 22, 24\}$

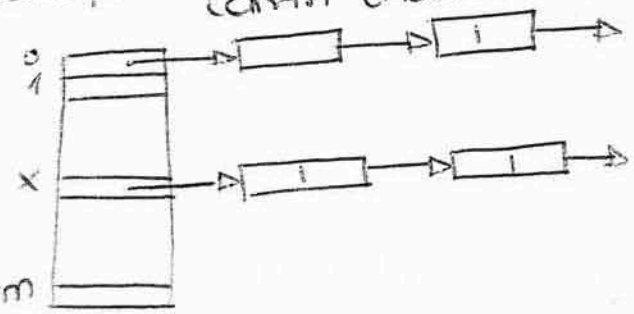
$h = x \bmod 5$

x	$h(x)$
3	3
15	0
20	0
22	2
24	4

T

0	15 * collision
1	
2	22
3	3
4	24

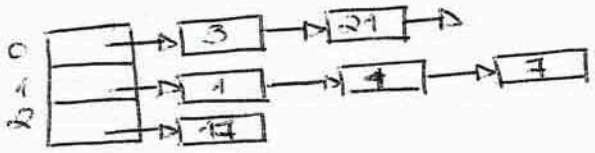
Κατακερματισμός (chained hashing) με αλυσίδες
(Chain hashing)



Πχ. $m = 3$ $U = \{0, 1, \dots, 99\}$

$S = \{1, 3, 4, 7, 10, 17, 21\}$

$h = x \bmod 3$



$O(m + |S|)$

Χρησιμοποιώντας λίστες, έχουμε θυμάσαι σε πράξεις

$O(1)$ (σταθερό)
insert(x) {
T[h(x)] + push(x)
}

search(x) {
T[h(x)] → search(x)
}
$O(|S|)$ τοπικός λίστας

delete(x) {
search(x);
T[h(x)] → delete(x);
}
$O(|S|)$ τοπικός λίστας

Κεφάλαιο σε πίνακα, χάνουμε σε πράξεις

Πολυπλοκότητα Νέας Τεχνικής

Παραδοχές

$h: U \to \{0, \dots, m-1\}$ ομοιόμορφα

K πράξεις { Μια από τις βασικές πράξεις που περιγράψαμε με την ίδια πιθανότητα }

Αν για K πράξεις, απαιτούμενο χρόνο T, τότε ο μέσος χρόνος είναι T/K.

• Κάθε κελί, σε κάθε πρόσβαση και διαφέρεται με την κάθε πιθανότητα

Οι λίστες θα ονομάζονται συνεχόμενα.

Θεώρημα → Οι k πράξεις $O(k(1+b/2))$ [Η αναζήτηση εκτός]

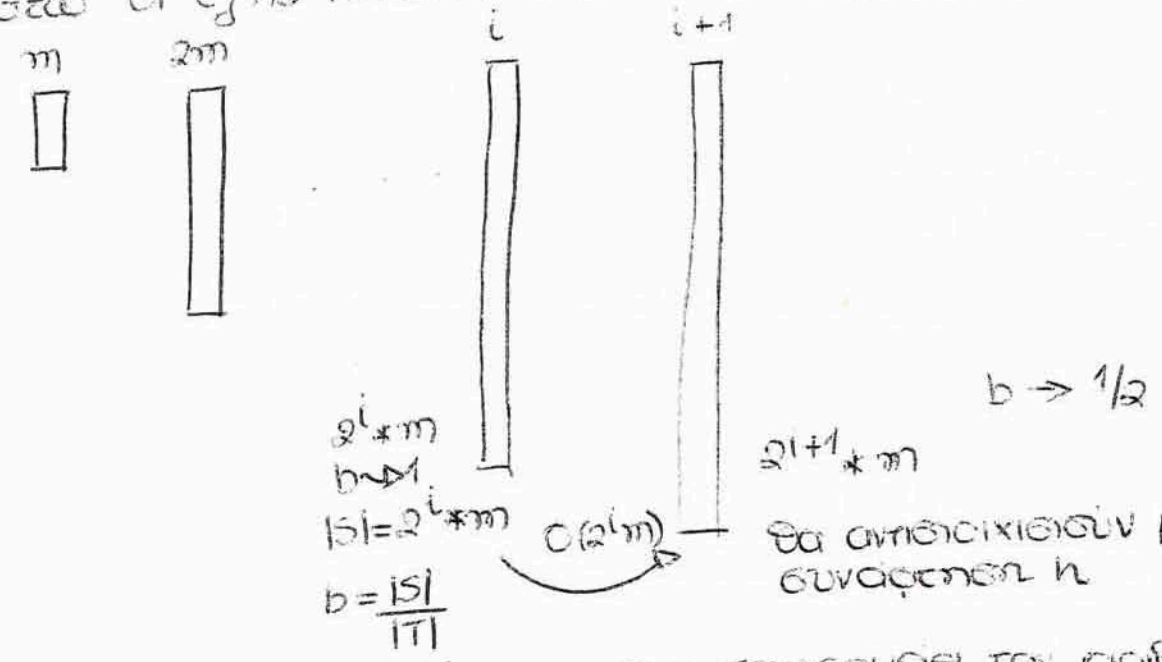
$b = \frac{|S|}{m}$ load factor

• $b \to 1$ τότε $|S| \simeq m$, $O(1) \to O(k) \xrightarrow[\text{πράξη}]{\text{η μια}} O(1)$

↳ Δεν είναι καλό για real time εφαρμογές.

• $b >> 1$ τότε $|S| >> m$ (οι λίστες μεγαλώνουν σε μήκος → κακή συμπεριφορά)

Έστω οι εξής πίνακες κατακερματισμού:



Το κόστος της μετάβασης δεν μεταβάλλει τον αριθμό των πράξεων που χρειάζονται για την μετάβαση

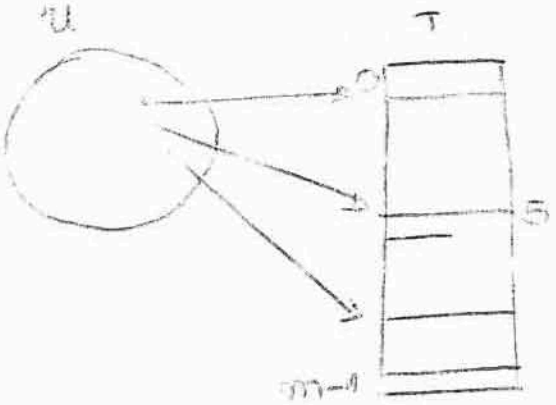
Υπάρχουν 2 τρόποι για να εκτελέσεις x πράξεις:

(1) όλες μαζί

(2) κάθε φορά μια χρωστούμενη πράξη και μια από τις εξωτερικές (insert, delete, access)

Κατακερματισμός με ανοικτή διεύθυνση (open addressing hashing)

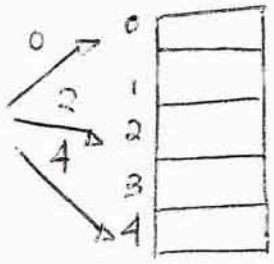
12/11/2004



$O(m)$ $\rightarrow \{0, 1, \ldots, m-1\}$ m!

$h(x,i)$

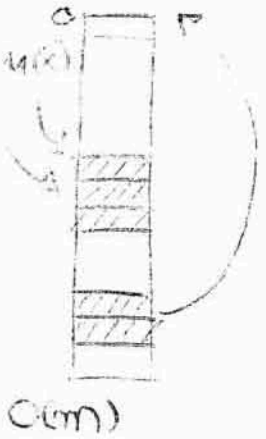
$i = 0, 1, \ldots, m-1$



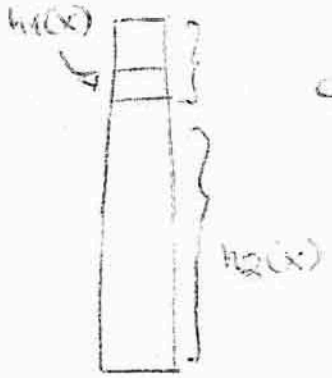

```
insert(x) {        O(m)
  i ← 0;
  while (T[h(x,i)] != NULL || i < m)   i++;
      if (i == m) fail
      else T[h(x,i)] = x;
}  → loopei ua jeliesi o wiuakos (reloveenta)

search(x) {
  i ← 0;
  while (T[h(x,i)] = NULL || i < m) {
        if (T[h(x,i)] == x) return h(x,i);
        else i++;
  }

delete(x) {
   if (pos = search() != NULL)  T[pos] = NULL;
}
```

Linear probing

$h(x,i) = (h_1(x) + i) \bmod m$



quadratic probing $h(x,i) = (h_1(x) + c_1 + c_2 i^2) \bmod m$

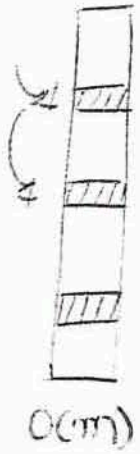


double hashing $h(x,i) = (h_1(x) + h_2(x) \cdot i) \bmod m$

O(m2)

m !

(πιο κοντά στο m!)



Θεώρημα → Ο μέσος χρόνος αποτυχημένης έρευνας & μέσος χρόνος εισαγωγής:

$$1/(1-b) \quad b = \frac{|S|}{m} \quad b \le 1 \quad |S| \le m$$

Θεώρημα → Ο μέσος χρόνος επιτυχημένης έρευνας

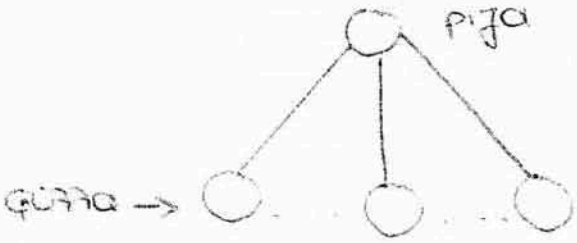
$$\frac{1}{b} \ln \frac{1}{1-b} + \frac{1}{b} \quad , b < 1$$

$|S|$ → μικρό → ακολουθούμε την αρχική $h(x)$

$x < S$, $\nexists$ x, y $h[x] = h[y]$

13/01/2004

Δέντρα ⇒ Δοτές μέσα στη κορυφή



minimum spanning tree → όταν βρούμε όλες τις ακμές (edges) και μετά υπολογίζουμε την μικρότερη.

$$\left.\begin{array}{l}\text{search}\\ \text{insert}\\ \text{delete}\end{array}\right\} \le \log n$$

13/1/2004

AVL-tree (1964)

searching

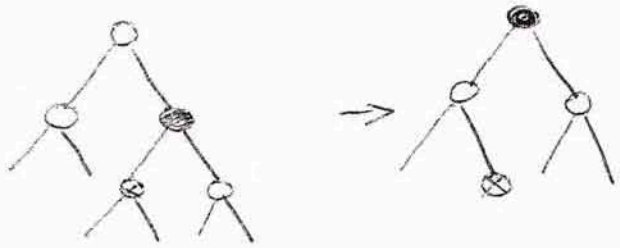
Hashing IST → loglogn O(1)

BB-tree → Rebalancing operations σε constant time.

BB[+]

→ BB-tree

γεμάτα" πράξεις logn



αλλαγή δομή → "ακριβά πράξεις" O(1)

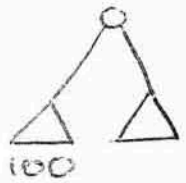
Ανάθεση, στο AVL-tree ισχύει O(logn)

→ AVL-tree

- Βαρυζυγισμένα → BBA, → IST
- Υψοζυγισμένα

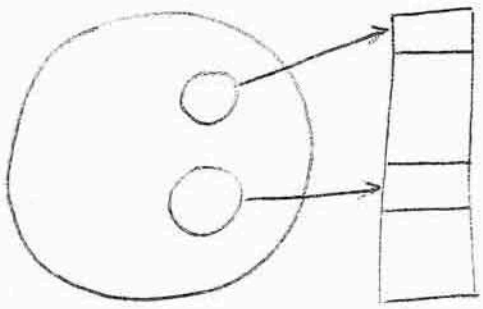
Βάρος → το πλήθος των στοιχείων σε ένα υποδένδρο

π.χ.

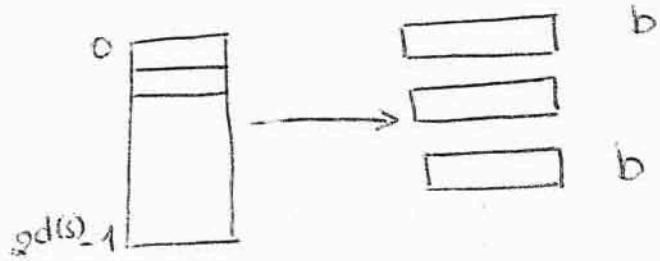


βάρος=100

19/01/2004



Τα στοιχεία αντιστοιχούνται σε ένα σταθερό αριθμό τα κελιά.



Μας ενδιαφέρει ο αριθμός των προσπελάσεων του φορτωμένου block.

$h: U \to \{0,1\}^k$, $\exists\, x,y \quad h(x)=h(y)$

$h_d: U \to \{0,1\}^d$, $d \le k$, $x \in U \quad h_d(x) = \text{prefix}_d(h(x))$

→ 1101 $h(1101)=11$

→ 1110 $h(1110)=11$

$x \quad h(x)_{1..m} \; \underbrace{b_1 b_2 \ldots b_k}_{d}$

19/04/2004

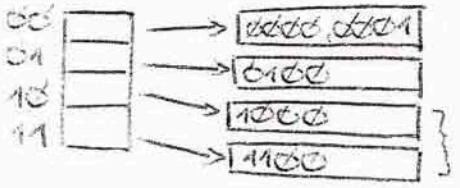
x	h(x)	$h_1(x)$	$h_2(x)$	$h_3(x)$
0	0000	0	00	
1	0001	0	00	
4	0100	0	01	
8	1000	1	10	
12	1100	1	11	

Τοπικό βάθος d(S)
όπου S τα στοιχεία που έχουμε αποθηκεύσει στην δομή.

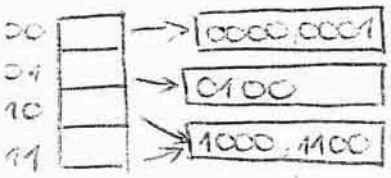
<u>Π.χ.</u> b=2

S = {0, 1, 4, 8, 12}

h(S) = {0000, 0001, 0100, 1000, 1100}



Θα εξοικονομήσουν τον δίσκο, κωδικοποιούμε οι αποθ. κείμενα στα στοιχεία στο ίδιο block



Βλέπουμε ότι δεν έχουν το ίδιο πρόθεμα

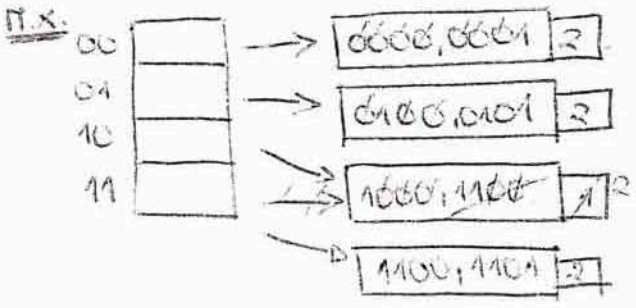
O(1) → I/O

```
Search(x) {
  iBIN = hd(S)(x);
  ψάχνουμε B T[iBIN]
}
```

```
insert(x) {
  iBIN = hd(S)(x);
  B ← T[iBIN]
  if (|B| < b) εύκολη x → B
  else if (iB != d(S))
        B → B
          → B'
  else d++, ξαναοργανώνουμε T
```

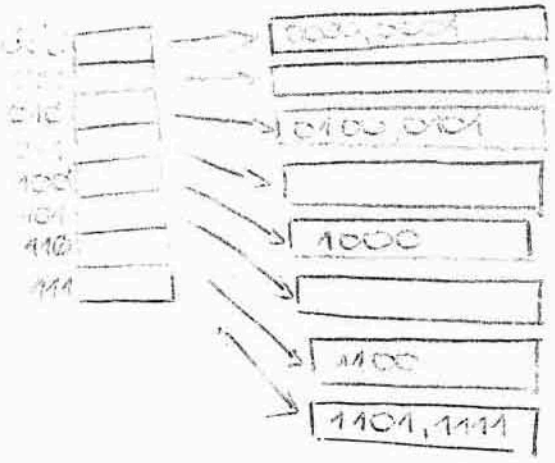
O(2^{d+1})

<u>Π.χ.</u>



insert(0101)
insert(1101)

Δεν ∃ τοπικό βάθος τώρα



Αντιστοιχεί σε ένα
ειδικό πίνακα
κατακερματισμού

μέγεθος → b

```
delete(x) {
  i_BIN = hd(s)(x);
  B ← T[i_BIN]
  if (x υπάρχει στο B) διαγραφή x;
    B' για το ομάδο υπάρχει ίδιο πρόθεμα
    if (|B'| + |B| < b)  B = B ∪ B'
  Αν (r_B = r_B' == d(s)) μειώνεται ο T, d = d-1
      και όχι άπαν σειριακά[ίδιοι]
}
```

$b_1 b_2 b_3 \ldots : b_{r-1}$ → ομάδες r τύπου βάθος

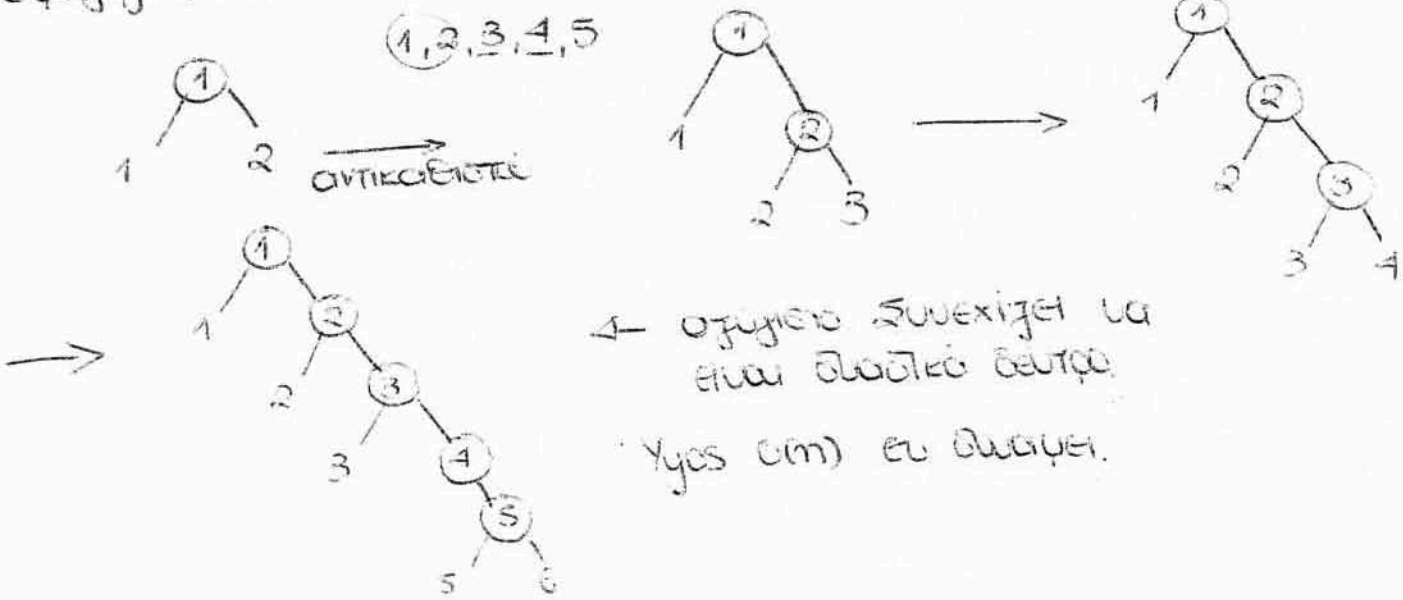
Θ.1 → Μέσο πλήθος σειριών είναι $n/b\ln 2$ και το μέσο μέγεθος κατακερματισμού $|T| = \left(e/b\ln 2\right) n^{1+1/b}$

→ Red - black δέντρα ←

Υποχρεωτικά : AVL, Red-black

Βαρύτητα : BB[α]

1, 2, 3, 4, 5



← ο γονέας δουλεύει ως
ένα δυαδικό δέντρο.

Υψος O(n) σε δυάνει.

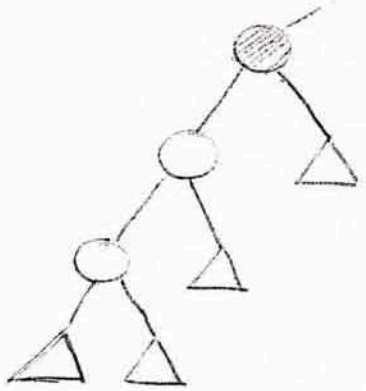
Ίδιοι κόμβοι αναζήτησης σε ένα ισορροπημένο δένδρο είναι $O(\log n)$.

Ορισμοί:

Π1: Κάθε κόμβος είναι κόκκινος ή μαύρος

Π2: Κάθε μονοπάτι από ρίζα στα φύλλα έχει τον ίδιο αριθμό μαύρους κόμβους (παρόμοιο με ισορροπημένο, όχι παρόμοιο από $\log n$)

Π3: Κάθε κόκκινος κόμβος έχει πατέρα μαύρο



Π4: Όλα τα φύλλα είναι μαύρα.

Χρήσιμοι ορισμοί ←

Κάθε κόμβος rank() → ένα τιμή που αναθέτουμε σε ∀ κόμβο

Ο1: αν x είναι φύλλο τότε rank(x)=0 και rank(p(x))=1 (p είναι ο γονέας)

Ο2: $rank(x) \le rank(p(x)) \le rank(x)+1$ (Διαφέρουν το πολύ κατά 1)

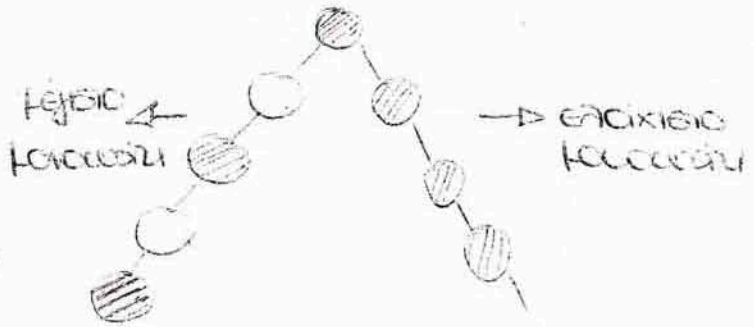
Ο3: $rank(x) < rank(p(p(x)))$

(Κάποιος κόμβος μπορεί να έχει το ίδιο rank με τον παππού αλλά όχι με τον πατέρα του παππού)

$χρέωμα(x) = rank(p(x)) - rank(x)$

* Το rank της ρίζας θα είναι ο αριθμός των μαύρων κόμβων του μονοπατιού της *

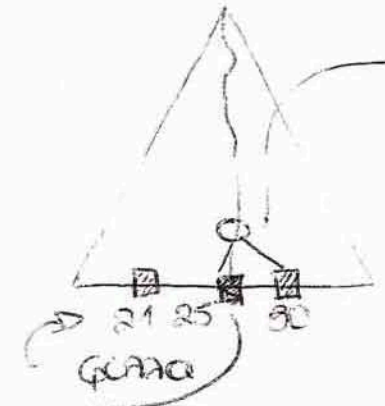
$\frac{1}{2}\log n \le Ύψος(T) \le \log n$



* Μπορούμε να έχουμε 2 μαύρους κόμβους σε σειρά αλλά όχι 2 κόκκινους *

19/04/2000

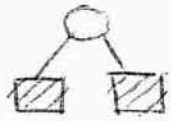
Πόσο θα διάρκεσε μια πράξη εύρεσης σε ένα red-black tree με n κόμβους το περισσότερο? Απ. Θ(logn) (ίδιο με το διάλειμμα + το μέγιστο logn)



→ Το μικρότερο υποδένδρο με ένα κόκκινο κόμβο (πρόβλημα)

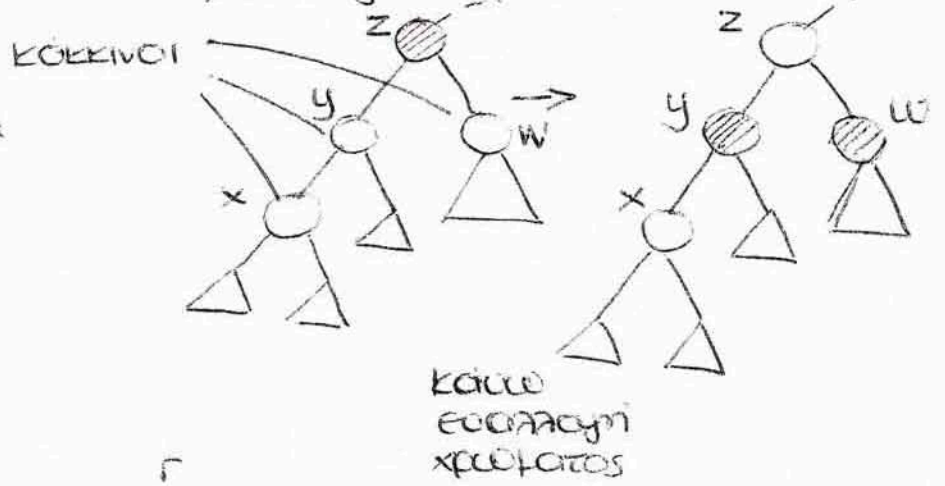
→ Μηδενικός κόμβος

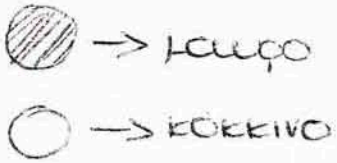
→ Αντικαθιστούμε το παιδί φύλλο με ένα κόκκινο κόμβο ο οποίος έχει 2 παιδιά φύλλα

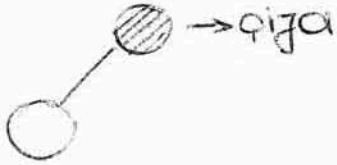


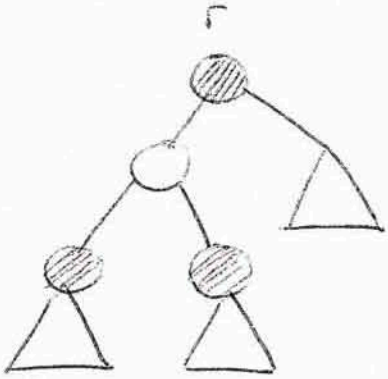
→ Αλλαγές ←

1) Αλλαγές χρώματος

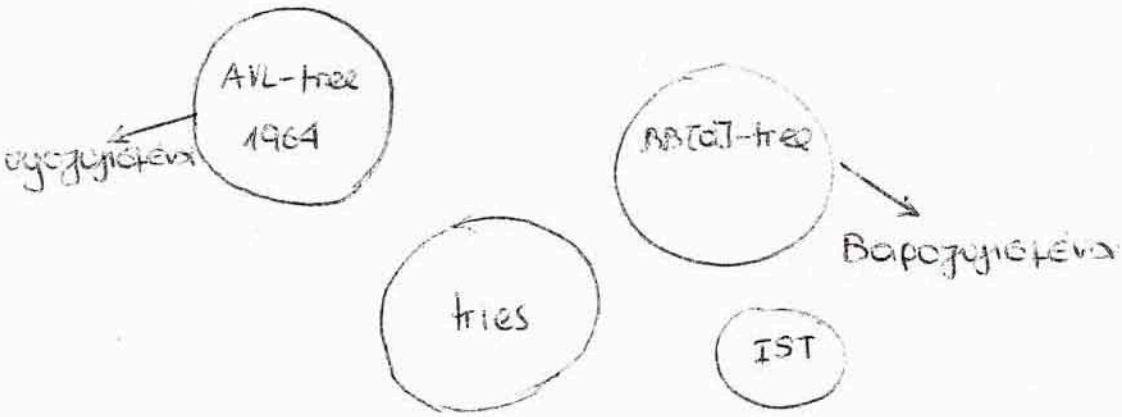






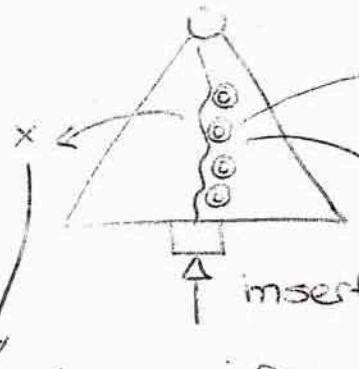


20/01/2001



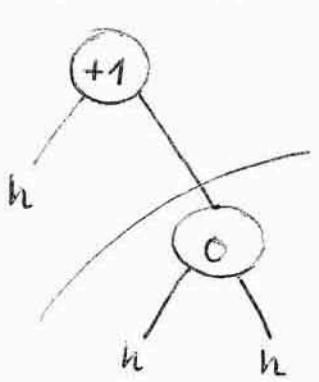
20/04/200

Λογαριθμικό ύψος → το πολύ 1.44 * log n

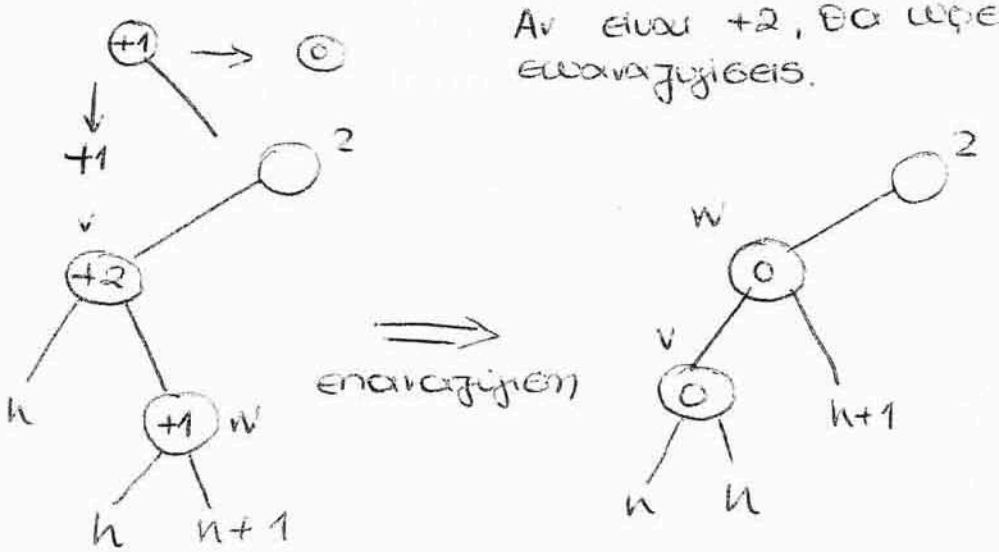


→ κρίσιμο πρόγονο
θα πρέπει να βρω ποιος
είναι ο κρίσιμος κόμβος
όλα μπαίνει, αν ενώσω: αν x=1 τότε
ανεβαίνει στο x το ύψος.
αν x=-1 τότε x=-2 – πρόβλημα

Κρίσιμος κόμβος ευθείας
↳ δ'αυτόν ελέγχω, αν το δέντρο χρειάζεται rebalance.

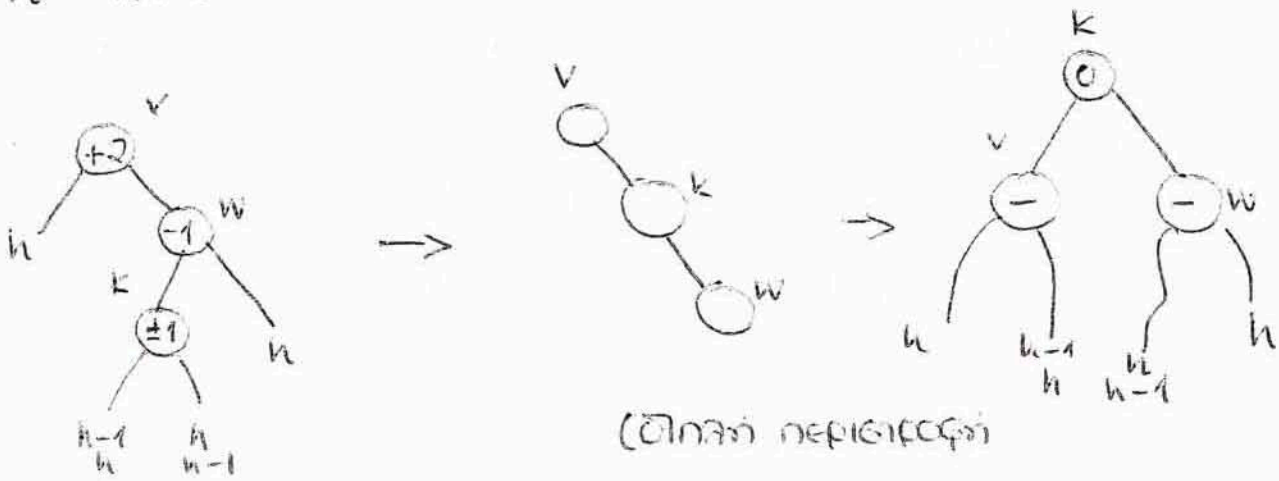


→ μέχρι εδώ θα έχει όλα μπει (ΚΚΕ)

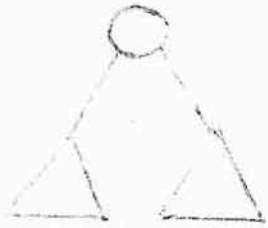


Αν είναι +2, θα πρέπει να κάνουμε εναλλαγή.

(ακαμπή περίπτωση)

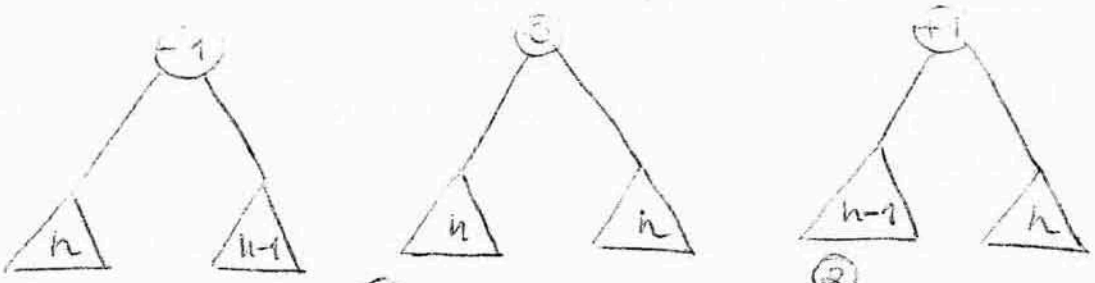


(σύνθετη περίπτωση)



$\pm 1, 0$

→ ισοζυγισμένο δέντρο

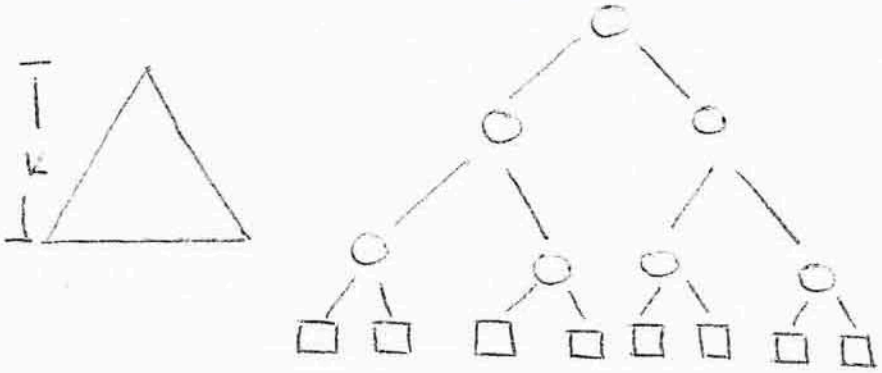


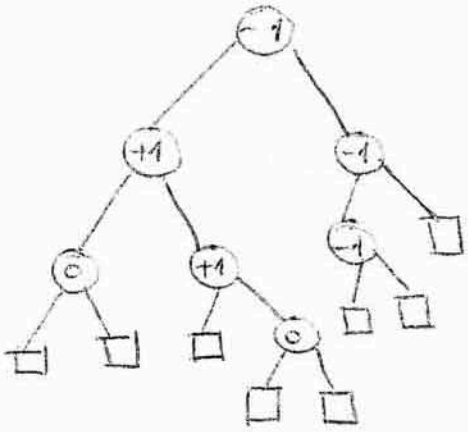
Το δέντρο έχει ① λογαριθμικό ύψος, ② και insert/delete σε log n στο worst case

n στοιχεία →

μεγάλο ύψος στο πάνω ισοζυγισμένο δέντρο.

$n = 2^k$





Fibonacci Δέντρο

↳ $f_0 = \square$ $\quad f_1 =$  $\quad \dots \quad f_n =$ 

26/11/2004

3) Top-down rebalancing

4) Αν ένας κόμβος βρίσκεται σε ύψος h, σε μία ισορροπία η μπορούμε να εκμεταλλευτούμε $O(n/2^h)$ φορές

5) Ζύγιση με 1-bit

TRIES

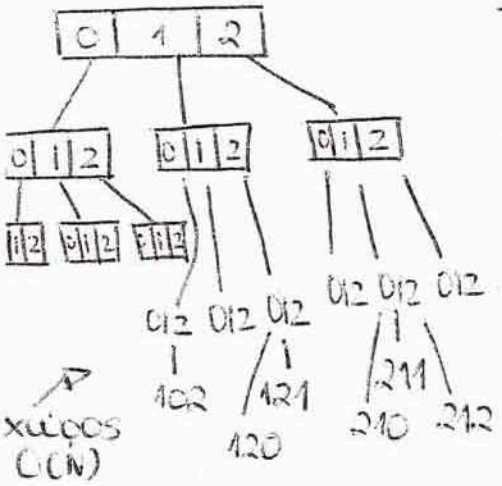
Υπάρχουν δομές δεδομένων

Έστω $S = \{121, 102, 211, 120, 210, 212\}$

$S \subseteq U$ (σύμπαν)

$\Sigma^3 = U \Rightarrow \Sigma = \{0,1,2\}$

έχουν μήκος 3



→ Κάθε επίπεδο του δέντρου, έχει ένα ορίσμα με συγκεκριμένο bit.

→ Η ρίζα έχει τόσες επιλογές όσα είναι τα γράμματα

$|S| = n$

$|U| = N$ → Μέγιστο μέγεθος του στοιχείου και εκεί

$N = k^{\ell}$ → όλοι οι δυνατοί συνδυασμοί που μπορούν να έχω

k χαρακτήρες του αλφαβήτου

$\Rightarrow \ell = \log_k N$

Αναζήτηση $O(\ell) = O(\log_k N)$

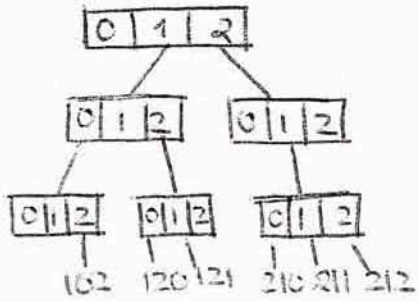
Αντίθετα στο RAM μπορούμε πολλαπλασιασμό γιατί χρησιμοποιεί πίνακες. Νωρίς με λίστα κάναμε και για το μπορούμε pointer machine, χρησιμοποιώντας λίστες.

PM $O(k \cdot \log_k N)$

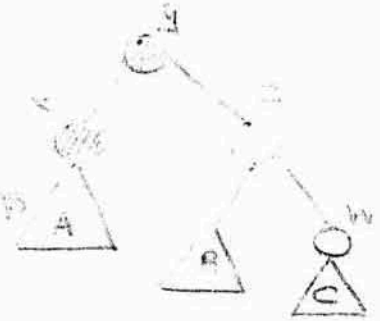
Τι μπορώ να κάνω για να μειώσω το χώρο?

Κόβω τα επίπεδα της δομής στα οποία δεν αλλάζει κάτι.

$O(\ell \cdot k \cdot n)$

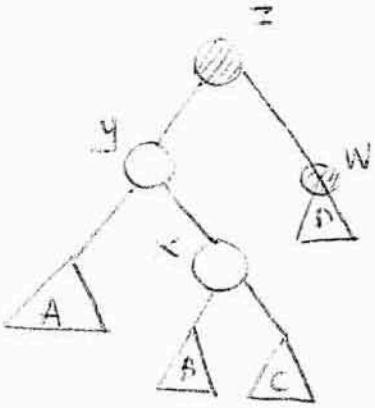


Είναι η συμμετρική του τριγωνικού

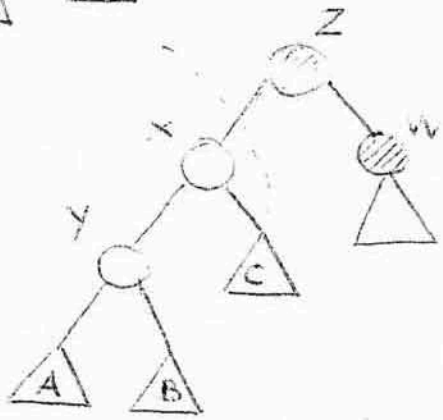


Αριστερή → περιστροφή

6)

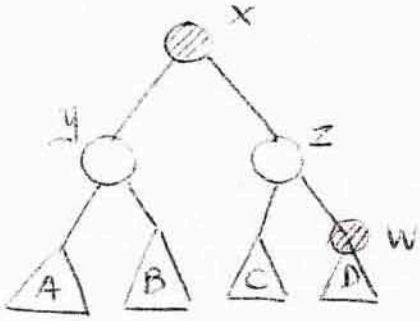


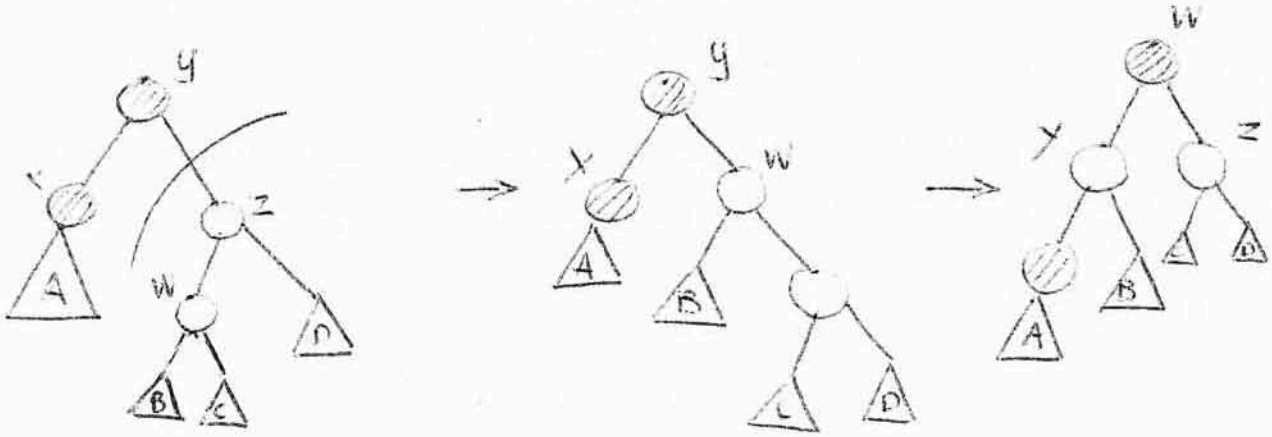
Διπλή → περιστροφή (xy προς αριστερά)



Η συμμετρική διατηρεί την διάταξη

→



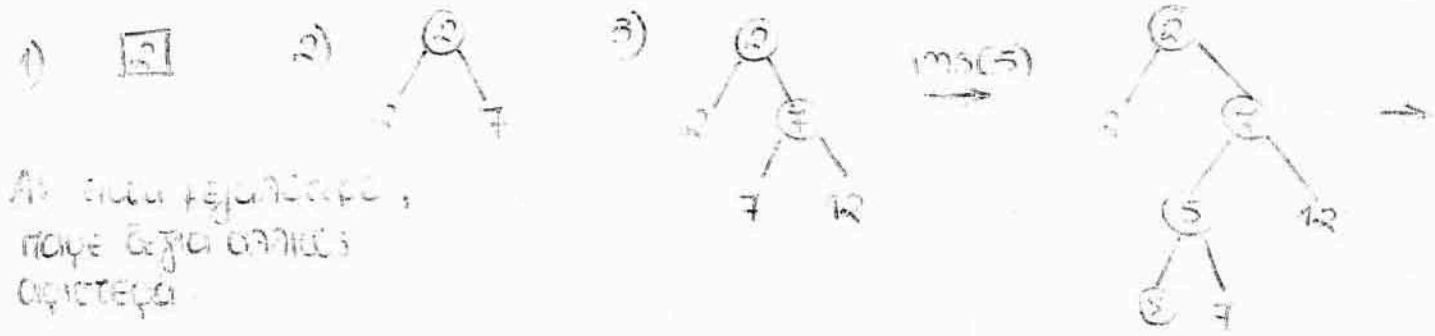


ΙΔΙΟΤΗΤΕΣ (SOS)

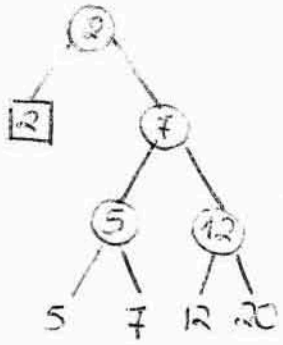
1) Μετά από 1 εισαγωγή/διαγραφή χρειάζονται $O(\log n)$ αλλαγές πράξεις σταθερές $O(1)$ βάθους.

2) Δυνατό ελάχιστο ύψος σταθερής από τη εισαγωγή/διαγραφής είναι $O(n)$

26/04/2014



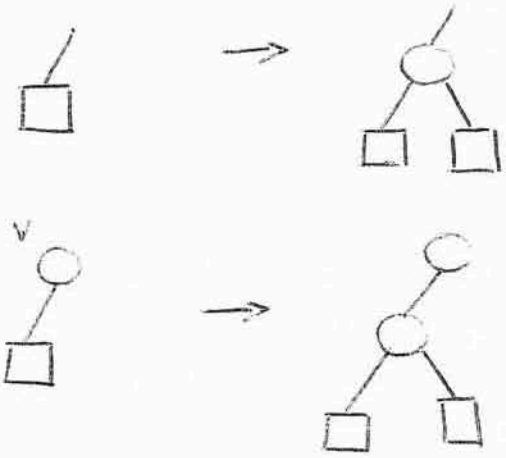
Αν ήταν κόκκινος, πάμε δεξιά αλλιώς αριστερά



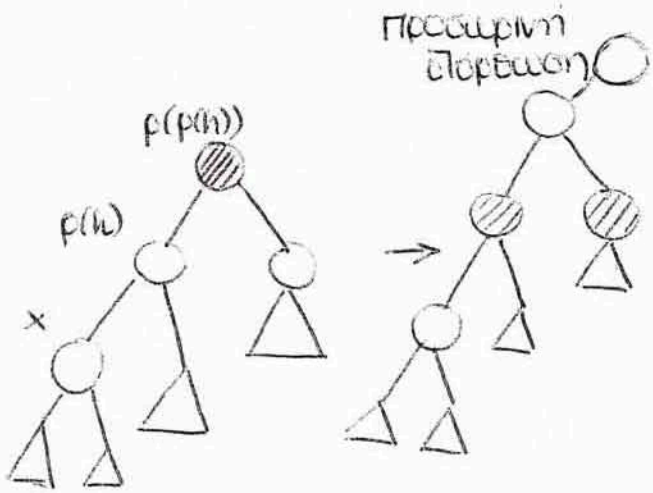
Αλλαγή χρώματος → AVL $0 \to \pm 1$
→ color flip

- S. Sahni
  Data structures in C++
- R. Sedgewick
  Algorithms in C++

Red-black δέντρα



1)

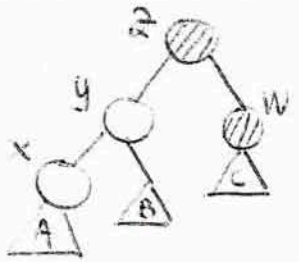


Εναλλαγή χρώματος (η τέρατική)

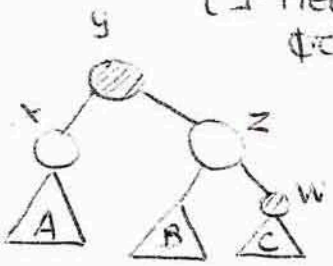
Όταν δεν είναι τέρατική, συνεχίζουμε προς τα πάνω (Ζ περίπτωση να φτάσουμε στη ρίζα)

Δομικές αλλαγές (περιστροφές)

2)



απλή περιστροφή της ακμής yz (δεξιά)



Τέρατική γιατί η ρίζα παραμένει μαύρη.

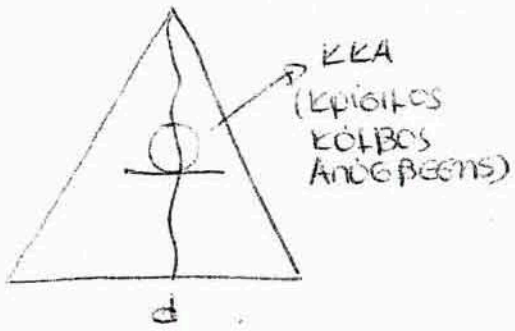
26/01/02

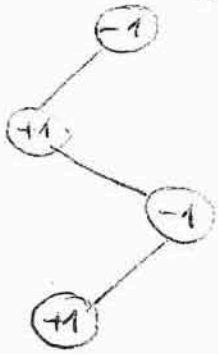
	[illegible]	[illegible]
i	log n	O(1)
d	log n	log n

→ BB-tree

Για την περίπτωση delete :

Κρίσιμο κόμβο ανάβαθμο;



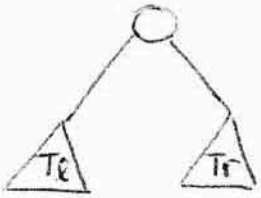


	[illegible]	[illegible]	
i	log n	O(1)	O(1)
d	log n	O(1)	O(1)

→ BB-tree amortized

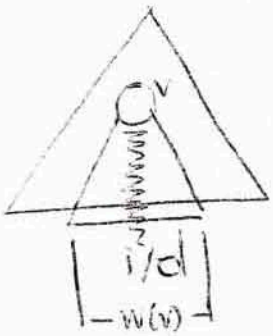
3n 4n

Σετ a → delete, IST



$$\frac{1}{3} < p = \frac{|T_l|}{|T_r|} < \frac{2}{3}$$

(π-α) (α)



(worst case του v)

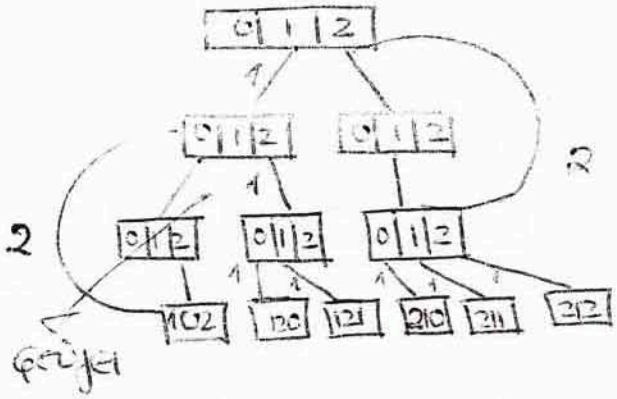
→ Στο πιο βάρος έλεγε, τόσο υψο αρχίζει θα του χρειαστεί για rebalance. ← (ιδιόμορφο του βάρους)

26/01/2004

0 7 12 18 20

Μπορούμε να το κάνουμε σε $O(m\,k)$

→ Αφαιρούμε από στη δομή τις ακμές που μας έχουν. Αρκούμε πάλι ένα για κάθε θέση αλλά γράφουμε κρεμασμένες.

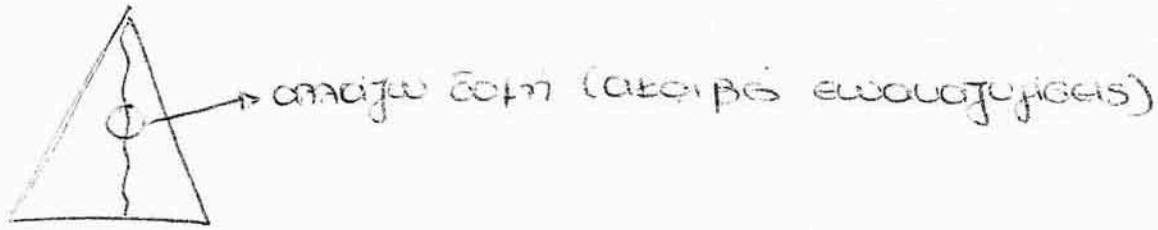


Κάθε χρόνος $O(\log_k m)$
search

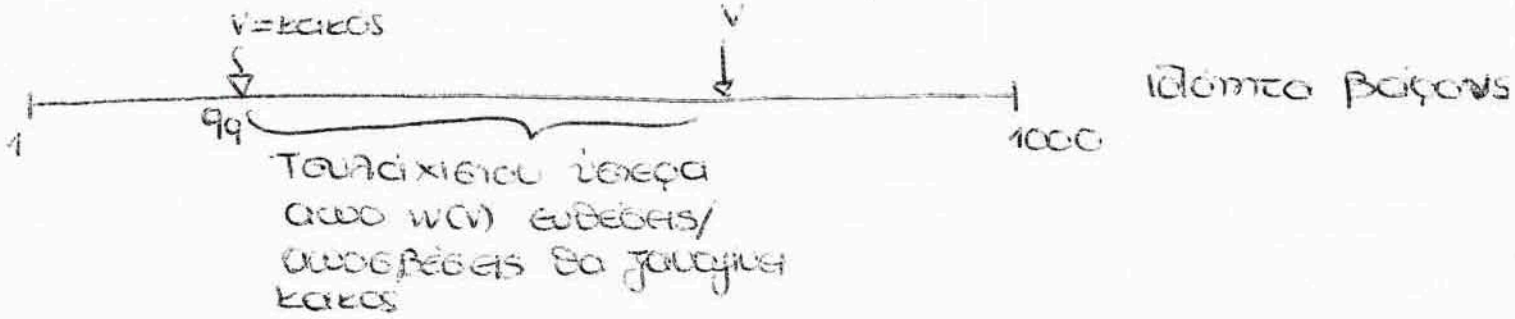
27/01/2004

BB[α] → Δυναμικοποίηση των Πολυδιάστατων Δομών

→ Βαρυσταθμισμένο Δέντρο

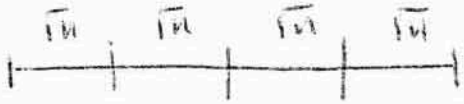


Θέλω αυτές οι ενημερώσεις να την χτυπάει ευχερώς. Άρα, εκμεταλλευόμαστε την ιδιότητα του βάρους.



IST - tree

Το BB - tree δεν έχει καλό reaction στις δυναμικές δομές



subtree ⇒ $O(m)$

21/2/2004

Interpolation Search Tree

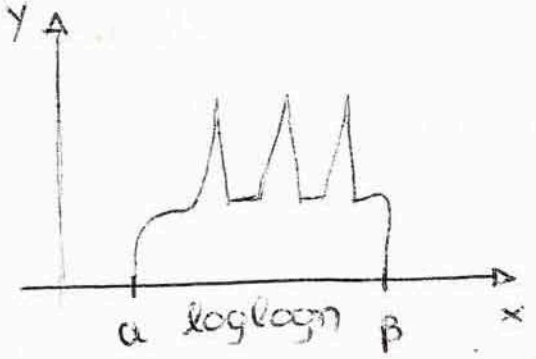
- Mehlhorn – Tsakalidis (1993) → IAUM
  Dynamic Interpolation Search

$O(\log\log n)$

ΔΙΟΤΗΤΕΣ

1. Χώρος $O(n)$ και κτίζεται σε $O(n)$ χρόνο.
2. Αναζήτηση $O(\log\log n)$ χρόνο στη μέση περίπτωση για smooth
   [Τα στοιχεία που εισάγουμε, οι τιμές τους υπακούουν κάποιες συγκεκριμένες κατανομές.

κατανομές.

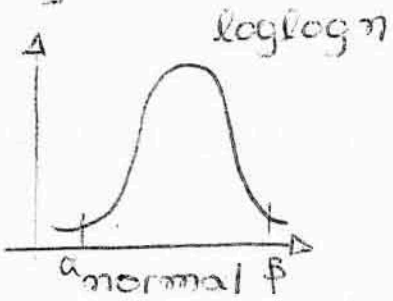


y → πιθανότητα εμφάνισης κάποιας τιμής

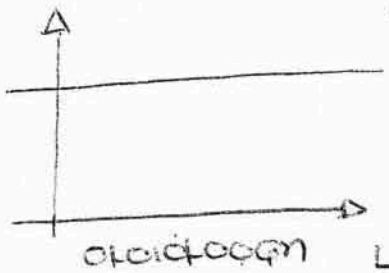
x → εύρος τιμών

Η πιθανότητα εμφάνισης είναι σχεδόν σταθερή στη μεσαία διαστήματα.

∫πιθανότητα είναι φραγμένο



(smooth κατανομή)



← ομοιόμορφη υπό έχει την ίδια πιθανότητα να επιλεγεί από ένα διάστημα τιμών.

↳ η αναζήτηση της παίρνει loglog n

Τα παραπάνω ισχύουν για τις εισαγωγές.

Διαγραφές

Διαλέγω τε την ίδια πιθανότητα όλες τις τιμές στη σειρά να μπορώ να διαγράψω.

τυχαίες ομοιόμορφες.

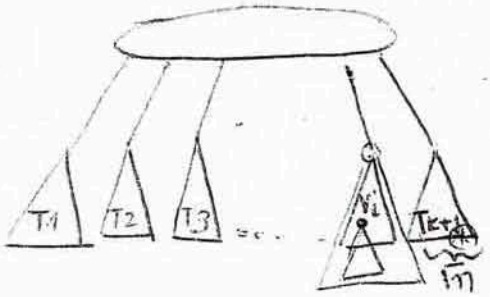
2/2/2004

Αν έχουμε $\overline{m}$ φύλλα, τότε το REP θα έχει $\overline{\overline{m}}$ κόμβους είναι ωραία να κάνει μια κάποια κατάσταση ώστε αυτό το ωραιότατό του μου στις φίλους θα ευνοήσουμε.

Ιδανικό IST έχουμε όταν $m = \lceil m^a \rceil$ για $\frac{1}{2} < a \le 1$, τα Sj (→ κόμβοι ID) έχουν όλα ακέραιο $\overline{m}$ κόμβους και τα $T_1, T_2, \dots, T_{k+1}$ είναι ideal.

Για να έχουμε $O(n)$ χώρο, πρέπει $\frac{1}{2} \le a < 1$

Εισαγωγές – Διαγραφές



Οι εισαγωγές μπορούν να κατακλύσουν υπερβολικά ένα δέντρο. Δεν έχουμε τρόπους εξισορρόπησης όπως στο red-black tree.

n Έστω w είναι αρχή και w + w/4 λέει. Εφαρμόζω partial rebuilding. Παίρνει το δέντρο που έχει κατακλύσει υπερβολικά και φτιάχνει ένα ideal IST δέντρο με w/4 + w φύλλα.

c(vi) size(vi) Όταν θέλουμε να διαγράψουμε ένα στοιχείο, σηματοδοτούμε το φύλλο που την περιέχει σημαδάκι ότι διαγράφουμε το φύλλο, αλλιώς το αγνοούμε. Μας μένουν χώρο χωρίς να τους χρειαζόμαστε.

Στην χειρότερη περίπτωση, θέλω $O(\log n)$

$H(n) \le 1 + H(n/2)$ → 060 ως προς το κάτω, μειώνεται (συμπληρωματικά) το ύψος.

Ορίζουμε σε κάθε κόμβο το size(vi) που είναι ο # των φύλλων χωρίς σημάδι ενώ πρέπει να ορίσουμε το isize(vi): isize(vi) = size(vi) (initial size). Ενδιαφέρομαι έτσι ώστε το δέντρο στον κόμβο vi να είναι γραμμικό. (Rebuild(vi)) → ① κατασκευάζει ένα ideal IST για τα μη σημαδεμένα φύλλα Tvi, ② θέσε c(w) = 0, ∀ w ∈ Tvi και σωστό isize(vi).

2/2/200[illegible]

3. Εισαγωγές/Διαγραφές

$O(\log n)$ [illegible] (amortized)

[illegible] $O(\log^2 n)$

4. Αναζήτηση $O((\log n)^2)$ στη χειρότερη περίπτωση

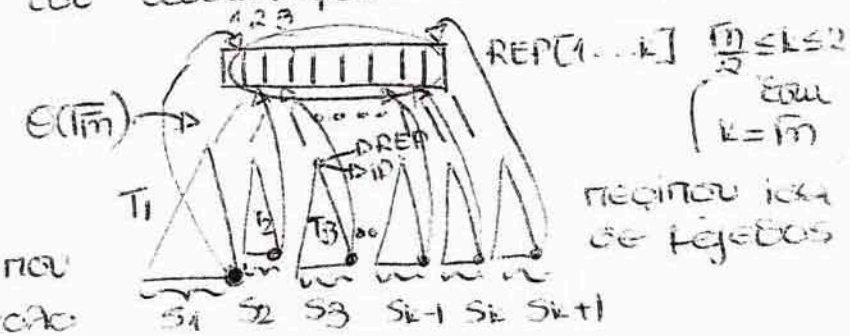
Έχει βελτιωθεί σε $O(\log n)$ σε μεταγενέστερη εργασία

5. Διατήρηση ακμών του ενεργού δυαδικού γράφου σε $O(m)$ χώρο, ενώ οι Predecessor, Successor και Min σε $O(1)$ χρόνο.

Στο IST δέντρο, ο βαθμός κάθε κόμβου εξαρτάται από τον αριθμό των φύλλων του υποδέντρου του.

- Βαθμός ρίζας είναι $\Theta(\sqrt{m})$

$\frac{\sqrt{m}}{2} \le$ βαθμός $\le 2\sqrt{m}$



Κάθε ένα από τα υποδέντρα που δημιουργούνται κόβουν το εύρος τιμών που έχουμε σε κομμάτια.

- Ideal IST: Κάθε μας αποθηκεύει $\sqrt{m}$ φύλλα και έχει βάθος $\Theta(\sqrt{m})$ (Μας δίνει ιδανικό να υπολογίσουμε το ύψος του δέντρου)

$H(m) = 1 + H(\sqrt{m})$

Στην ρίζα αποθηκεύουμε ένα array που το ονομάζουμε REP.

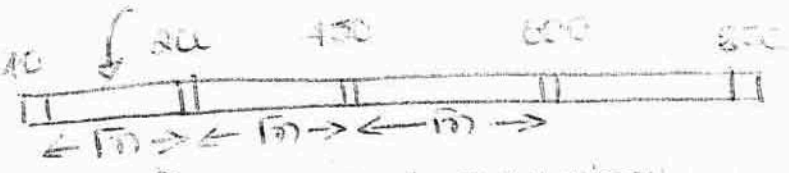
Το REP το χρησιμοποιούμε για να σύνδεσουμε την ανάμεσα στο φύλλο υποδέντρο. Ο REP αποθηκεύει τις από αριστερά προς τα φύλλα του δέντρου n τάξεις από το εύρος $S: x_1 < x_2 < \dots < x_m \le [a,b]$. REP είναι: $x_{i1}, x_{i2}, \dots, x_{ik}$ και $REP[j] = x_{ij}$

Την ίδια μου βάζουμε στο τέλος φύλλο του ∀ υποδέντρου στον τοποθετούμε στον πίνακα REP. Τα $S_1, S_2, \dots, S_{k+1}$ αποθηκεύονται σε IST δέντρα $T_1, T_2, \dots, T_{k+1}$ ενώ ideal H δεν αποθηκεύονται αναδρομικά. Επιπλέον, έχουμε $ID[1 \dots m]$ ενώ ορίζεται ανάδρομικά. ID = Inverse Distribution. Θέλουμε να υλοποιήσουμε την ανάγκη κατανομή των τιμών του εύρους ώστε μια εξοπλίζουμε την κατανομή. $ID[j] \Leftrightarrow REP[j] \le a + i \cdot \frac{b-a}{m} = REP[j+1]$

(ID [illegible] στην ρίζα)

2/2/2004

Τα περιεχόμενα του REP μοιάζουν με τους δείκτες και χρησιμοποιούμε ένα είδος interpolation search.



αναζήτηση σε αρίθμητικό πίνακα

Το array ID με βοήθεια του οποίου βρίσκουν γρήγορα.

Συμπεριφέρεται.

Το ID έχει ακριβώς όσες θέσεις όσο το REP

1) $i = \left\lceil \frac{y-a}{b-a} n \right\rceil$

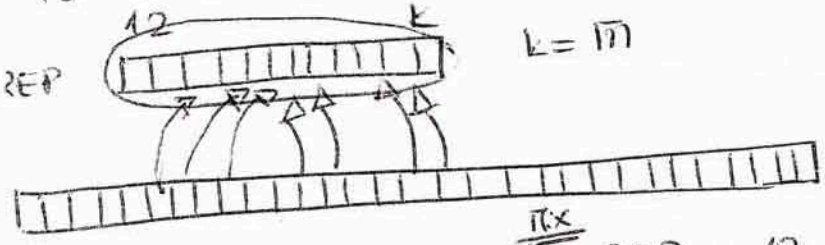
2) $j = ID[i]$

3) while $(y \geq REP[j])$ AND $(j<k)$ do

$j = j+1$

4) γράφε στο j-οστό κελί.

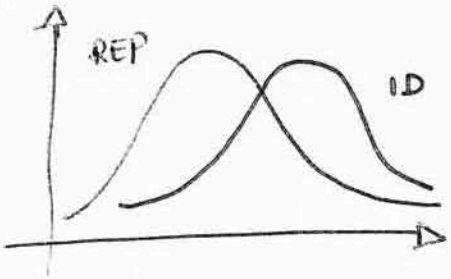
Το ID έχει δείκτες μέσα συμπεριφέρεται όπως του REP



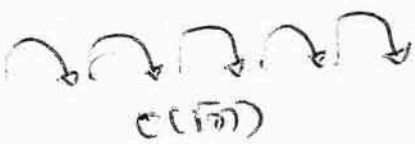
$k = \sqrt{n}$

Πίν.

REP	12	15	20	22	40	71	202	400	700
ID									



για $n=18$ Έχουμε κάποιο άλμα τιμών του ID δείχνουν στις θέσεις τιμές του REP και ανάλογα πολλές τιμές κάποιες δείχνουν σε θέσεις τιμές του REP.



Πίν.

$O(\log\log n)$ εφαρμόζουμε interpolation search στο REP

Με εκθετικό-δυαδικό γράφημα

$REP[j], REP[j+2^0], REP[j+2^1], \ldots, REP[j+2^{i-1}], REP[j+2^i]$

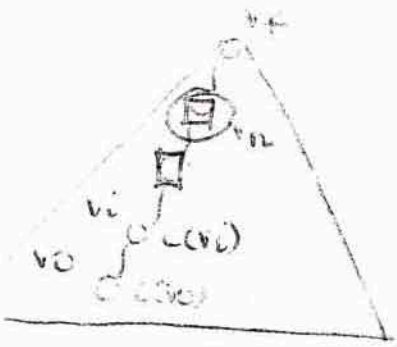
τ.ω. $REP[j+2^{i-1}] \leq y \leq REP[j+2^i]$

Δυαδικό γράφημα στις $REP[j+2^{i-1}+1], REP[j+2^{i-1}+2], \ldots, REP[j+2^i]$

σε χρόνο $O(\log n)$

2/2/2004

Διαγραφή (x)



Ξεκινάει με έναν κόμβο ανεξαρτησίας με μεγαλύτερο μέγεθος από το μέγεθος του στοιχείου x.

1. search(x, T) → 2. c(vo) = c και isize(vo)=
3. for i=1 to k
   c(vi) = c(vi) + 1

Κάθε κόμβος έχει έναν μετρητή

4. Πιο γρήγορος ως προς υπερχειλίσεις είναι ο vk
5. Αν ∃ vk Rebuild(vk)

c(v) ≥ isize(v)/4 → έχει τις περισσότερες ελεύθερες θέσεις το υποδέντρο του (υπερχειλίσεις). Στην συνέχεια, διαγράφουμε το μετρητή (πιο γρήγορος) από αυτούς.

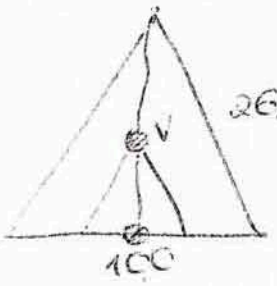
Συνήθεια με το βήμα 5, δημιουργούμε νέο υπερχειλίσεις και ένους υφιστάμενους του vk.

Διαγραφή (x)

1 . — —
2 Σπάσε φύλλο ως περιέχει x.
3 . . — —
4. — — —
5. . —

Τα υπόλοιπα βήματα είναι ίδια με την εισαγωγή(x)

Δεν μας ενδιαφέρει καθόλου αν έχουμε ακριβή διαγραφές και εισαγωγές



26 ≥ 100/4

Ο μετρητής συνεχώς μετράει του αριθμό των υφιστάμενων σε ένα ενεργειακό υποδέντρο του IST.

D = 12
E = 14
26 πράξεις → έχει κατάργη αυτό το υποδέντρο θέλει rebuild.

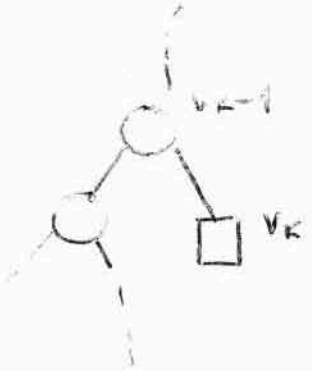
Ευρετήριο (y)

REP / k = √n

Το REP μας βοηθάει στο γράφο

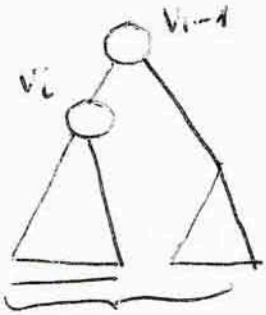
31/2/2014

w_i = # φύλλων T_{v_i} , $0 \le i \le k$



Το παιδί του v_{k-1} θέλω είναι εξωτερικό φύλλο.

$w_{k-1} \geq 2$. $w_{k-1} = 2$ όταν το παιδί του v_{k-1} είναι φύλλο.



$\frac{w_i}{w_{i-1}} \le (1-a) \Rightarrow \boxed{w_i \le (1-a) w_{i-1}}$

από αριστερό ΔΔΤ[a]

γιγαντώθηκαν του v_{i-1}

$2 \le w_{k-1} \le (1-a) w_{k-2} \le (1-a)^2 w_{k-3} \le \dots \le (1-a)^{k-1} w_0$

w_0 = # φύλλων του δέντρου $= n+1$

Άρα $2 \le (1-a)^{k-1}(n+1)$

Θέλουμε να βρούμε άνω όριο για το k και γι' αυτό λογαριθμούμε:

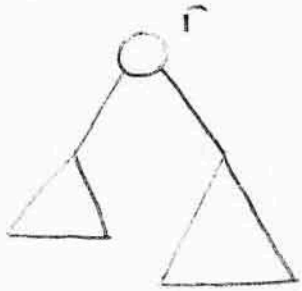
$1 \le (k-1)\log(1-a) + \log(n+1) \Rightarrow -(k-1)\log(1-a) \le \log(n+1) - 1$

$\Rightarrow (k-1)\log(1-a)^{-1} \le \log(n+1) - 1 \Rightarrow k\log(1-a)^{-1} \le \log(n+1) - 1$

$+\log(1-a)^{-1} \Rightarrow k \le \frac{\log(n+1)-1}{\log(1/1-a)} + 1 . \Rightarrow k = O(\log n)$

Έχουμε φράξει το ύψος ενός δέντρου με τον λογαριθμό του αριθμού των φύλλων που περιέχει.

Κάνοντας εισαγωγές/διαγραφές, χάλαει η γιγαντώθηση με αποτέλεσμα να εισερχόμαστε τα όρια με τα οποία φράσσουμε την γιγαντώθηση.



$1-a \geq p(r) \geq a$

χάλαει αν διαγράψουμε 1 φύλλο από το αριστερό υποδέντρο ή αν εισάγουμε 1 φύλλο στο δεξί υποδέντρο.

2/2/2004

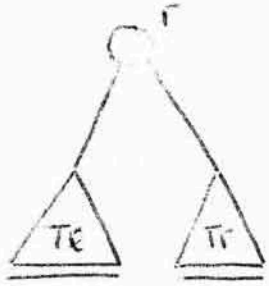
$\Omega(\log n)$ ύψος

$\hookrightarrow \Omega(\log_2 n)$

3/2/2004

BB[α] δέντρο

α)



$p(r) = \frac{|T_l|}{|T|}$

$|T| = 100$ φύλλα $\quad p(r) = 0.42$

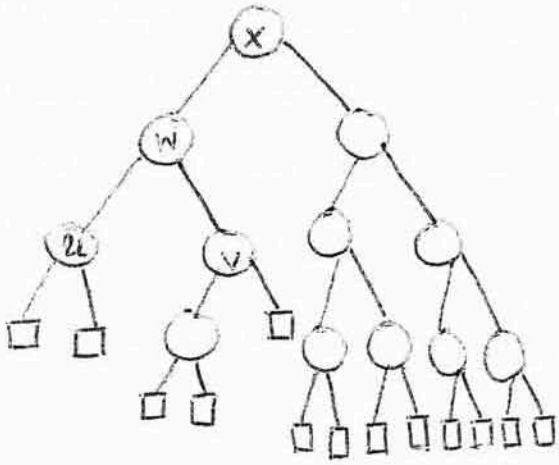
$|T_l| = 42$ - 11 -

β) Το T είναι περιορισμένης ισοζυγίσης αν $\alpha \le p(r) \le 1-\alpha$ (r ρίζα του T) και ισχύει ∀ υποδέντρο του T

Έστω $\alpha = 0.3 \quad 0,3 \le p(r) \le 0,7 \quad \alpha \in [1/4, 1-\sqrt{2}/2]$

γ) Το ελάχιστο ύψος των T περιορισμένης ισοζυγίσης είναι BB[α] δέντρο.

$\alpha = \frac{1}{3}$



$p(x) = 5/14$

$p(w) = 2/5$

$p(u) = 1/2$

$p(v) = 2/3$

$\frac{1}{3} \le p(v) \le \frac{2}{3} \quad \forall v \in T$

↓

ισοζυγίσιμη

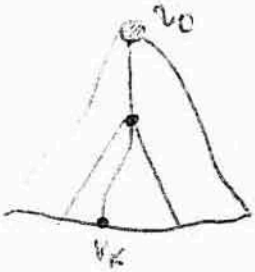
ΟΣ

Το T είναι BB[α] με n εσωτ. κόμβους

Απόδειξη με επαγωγή διαισθητικά ότι (n+1 φύλλα)

Έστω μονοπάτι $v_0, v_1, \dots, v_k$

v_0 ↑ ρίζα, v_k ↑ φύλλο



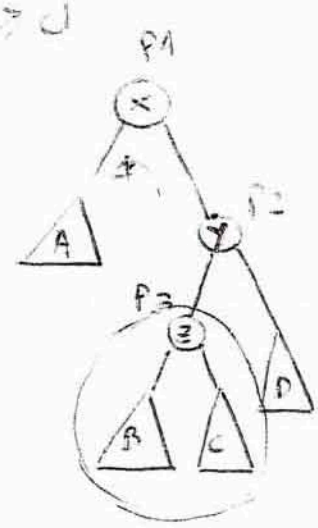
Το k είναι το βάθος του v_k ($k = \text{βάθος}(v_k)$).

Δεν είναι κατ' ανάγκη το μεγαλύτερο μονοπάτι αλλά θα μπορούμε να δείξουμε ότι είναι το ύψος του T.

$a \le \beta_1, \beta_2 \le 1-a$

Μπορεί να χρέωνα τα βάρη σε κάθε μετακίνηση.

2) $\beta_2 > d$

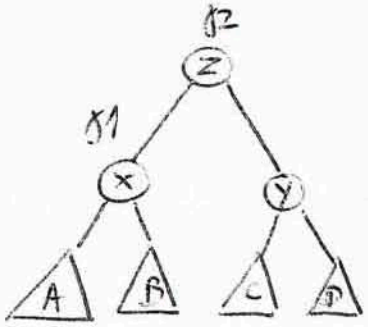


Θέλουμε κατά κάποιο τρόπο να περιορίσουμε τον αριθμό των φύλλων στο δεξί του x έτσι ώστε να χωρίσουμε παραγοντικά τα παιδιά του κόμβου x. Για το λόγο αυτό εφαρμόζουμε διπλή περιστροφή προς τα αριστερά.

$\beta_1 < a$

Διπλή περιστροφή → 2 απλές περιστροφές.

→



(Το θέμα το έχει λάθος)

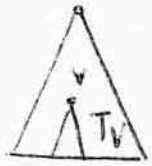
Αυτή τη διαδικασία τη εφαρμόζουμε στο πρώτο εσωτερικό κόμβο που υπάρχει πρόβλημα εκκαθάρισης.

Εφαρμόζουμε μέχρι $O(\log n)$ περιστροφές.

50S

ΙΔΙΟΤΗΤΑ ΒΑΡΟΥΣ

Ένας εσωτερικός κόμβος v θα εκκαθαριστεί μετά από $\Omega(|T_v|)$ πράξεις στο T_v.



Μετά από 1 περιστροφή, ο v θα έχει $|T_v|$ φύλλα.

$\Omega(k)$

π.χ. έχει k φύλλα. Θα εκκαθαριστεί μετά από $k/2$ διαγραφές ή εισαγωγές τουλάχιστον που συμβαίνουν σ'αυτό το δέντρο.

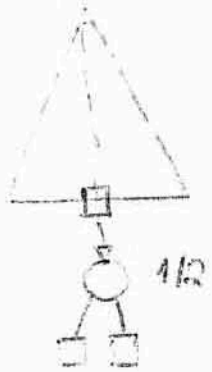
Στο IST, ισχύει εύκολη η ιδιότητα βάρους.

IST

isize(w) = |Tw|

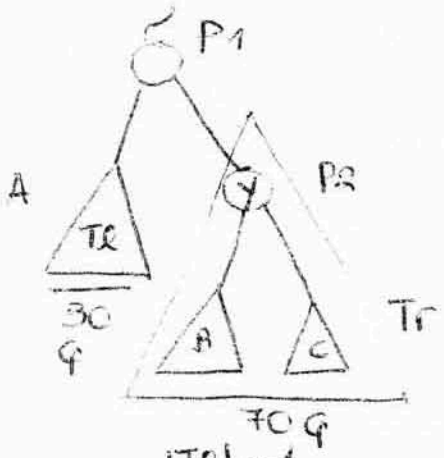
Θα εκκαθαριστεί μετά από $|T_w|/4$

31/2/2000



Αντιστοιχεί το φύλλο με ένα κόμβο με 2 παιδιά

$2^h = n \Rightarrow h = \log n$



Έστω ότι βρισκόμαστε σε ένα ισοζυγές κόμβο

→ $p1 \geq a$ ΠΡΙΝ

→ $p1 < a$ ΜΕΤΑ

• ΠΡΙΝ (παράφραση σε αριστερό δέντρο) $\frac{|Tl|+1}{|T|+1} \geq a$ το 1 δείχνει το φύλλο που έχω διαγράψει

• ΠΡΙΝ (εισαγωγή σε δεξί δέντρο) $\frac{|Tl|}{|T|-1} \geq a$

$\frac{30}{00}$ ok (στην αρχή, πριν την εφαρμογή ουσιαστικά αυτόματα)

$\frac{30}{101} < a$ αλλά $\frac{30}{101-1} \geq a$ = ok (εφαρμόζουμε εισαγωγή φύλλου)

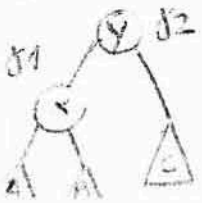
Την χρονική στιγμή i, είναι $\frac{|Tl_i|}{|T_i|} \geq a$. → T-1

Την -||- -||- i+1, είναι $\frac{|Tl_{i+1}|}{|T_{i+1}|} < a$ → T

$d > 0$

1) $p2 \leq d$ απλή περίπτωση

d → τυχαία σταθερά που επιλέγουμε εφείς με $d \in [a, 1-a]$



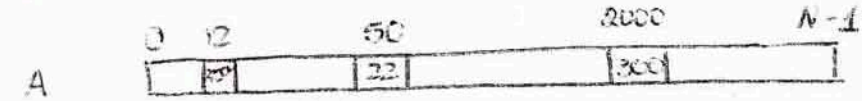
όπου $p1 = P1 + (1-P1)P2$

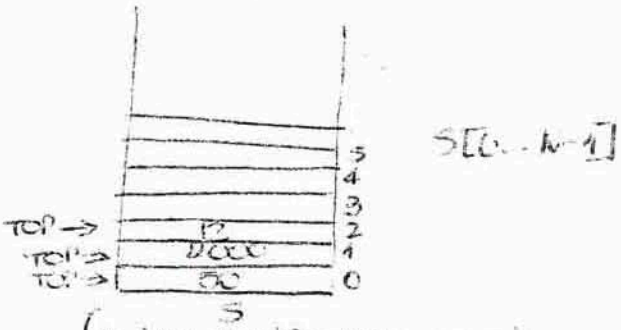
$p2 = P1/[P1+(1-P1)P2]$

3/2/2014

Η τεχνικά βάσης είναι πολύτιμη επιλογή σε εφαρμογές υπολογιστικής γεωμετρίας

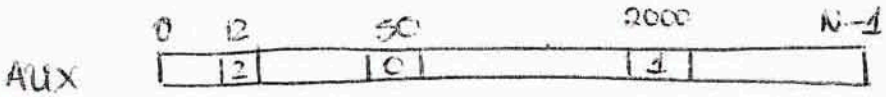
505 Array initialization σε O(1).





↳ Η σειρά που αυξάνεται με έναν array

Χρειαζόμαστε 1 επιπλέον πίνακα AUX



Αρχικά, TOP = −1

Βάζω λοιπόν στη θέση 50 το 22.

Εισαγωγή (v, i) → θέση στην οποία εισάγαμε → στη νέα εισάγαμε

```
  TOP = TOP+1
  AUX[i] = TOP
  S[TOP] = i
  A[i] = v;
}
```

Βάζω στη θέση 2000 εδώ αριθμό (έστω 300)

Η σειρά θα δείχνει στη σειρά με την οποία έχουν μπει οι στοιχεία ενώ ο AUX την θέση στο σωρό η οποία αντιστοιχεί σε αυτό έχει μπει εξόφλη

Ενώ δεν μέχρι αν έχω δοθεί ήδη αυτή η τιμή "αρχικοποίηση", ελέγχω τον πίνακα ως εξής:

```
Read(i) {
 if (S[AUX[i]] = i) AND (0 ≤ AUX[i] ≤ TOP)
    ok
 else
   fail
}
```