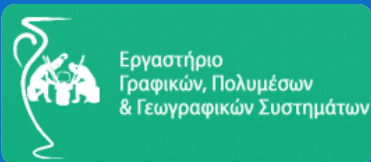


# Union – Find

Δομές Δεδομένων

Μάριος Κενδέα



19 Μαΐου 2015

[kendea@ceid.upatras.gr](mailto:kendea@ceid.upatras.gr)

# ΤΟ ΠΡΟΒΛΗΜΑ

- Πως χειριζόμαστε τις διαμερίσεις πάνω στα στοιχεία του σύμπαντος  $U$ ;
- Έστω ότι  $U = \{0, 1, \dots, N - 1\}$ ,  $x \in U$  και  $A, B, C$  ονόματα συνόλων
  - $\text{Find}(x)$ : Επιστρέφει το όνομα του συνόλου στο οποίο ανήκει το  $x$ .
  - $\text{Union}(A, B, C)$ : Ενώνει τα σύνολα  $A$  και  $B$  κάτω από το κοινό όνομα  $C$ .

# Υλοποίηση Find (1)

## ■ Απλή Λύση

- Για κάθε στοιχείο έχουμε το σύνολο που ανήκει
- Πράξη Union(A,B,C) -> απλή με κόστος  $O(N)$

0	A
1	A
2	B
3	F
.	.
.	.
.	.
N-1	B

# Υλοποίηση Find (2)

- Χρήση ανεστραμμένων λιστών
  - Μία για κάθε σύνολο
  - $L(A) = \{0, 1, \dots\}$
  - $L(B) = \{2, \dots, N - 1\}$
- Find είναι το ίδιο με πριν
- Union εκτελείται σε χρόνο  $O(|A \cup B|)$ 
  - Πιο αποδοτικό αλλαγή μόνο στο όνομα των στοιχείων του μικρότερου συνόλου στο όνομα του άλλου εσωτερικά
  - Και κρατάμε και απεικόνιση για τον εξωτερικό παρατηρητή
    - MAPOUT: πχ το B εσωτερικά είναι το C εξωτερικά
    - MAPIN: πχ το C εξωτερικά είναι το B εσωτερικά
      - Γιατί στο union(A,B,C)
        - Τα A έγιναν B
        - Και κρατάμε ότι το B είναι το C

0	A
1	A
2	B
3	F
.	.
.	.
.	.
N-1	B

# Θεώρημα Αποδοτικού Find

- Μία ακολουθία από
  - $n-1$  Unions και (Union κόστος επιμερισμένο  $O(\log n)$ )
  - $m$  Find (Find κόστος  $O(1)$ )
  - σε σύνολα αρχικού μεγέθους 1
  - έχει κόστος  $O(m + n \log n)$ 
    - Απόδειξη στο βιβλίο

# Αποδοτική Υλοποίηση Union

- Θυσιάζει χρόνο Find για σταθερό χρόνο στο Union
- 3 πίνακες
  - Father [1...N] : το  $i$ -οστό στοιχείο δείχνει τον πατέρα  $j$  στον ίδιο πίνακα
  - Name [1...N] : το  $i$ -οστό στοιχείο δείχνει την ομάδα του  $i$  αν είναι η ρίζα στην ομάδα αλλιώς κενό
  - Root [] : το  $i$ -οστό στοιχείο του πίνακα δείχνει την ρίζα της ομάδας με όνομα  $i$ .
- Find( $x$ ): αναζήτηση πατέρα μέχρι πριν να καταλήξουμε σε κενό. Τότε εκτυπώνουμε το αντίστοιχο όνομα
- Union(A,B,B) :  $i \leftarrow \text{Root}[A], j \leftarrow \text{Root}[B], \text{Father}[i] \leftarrow j$

# Weighted Union Rule

- Για αποφυγή μεγάλων βαθών δέντρων
- Η ρίζα μικρότερου γίνεται γιός του μεγαλύτερου
- Χρειάζεται πίνακας  $size[1..N]$ :  $i$ -οστή θέση πλήθος στοιχείων με ρίζα το στοιχείο  $i$ .
- Εφαρμογή: Οι αριθμοί στις παρενθέσεις δηλώνουν τον αριθμό των στοιχείων κάθε συνόλου. Τα κεφαλαία είναι ονόματα συνόλων
  1.  $U(1,2,A)$
  2.  $U(3,4,B)$
  3.  $U(A,B,C)$
  4.  $U(5,6,D)$
  5.  $U(7,8,E)$
  6.  $U(D,C,F)$
  7.  $U(E,F,G)$

