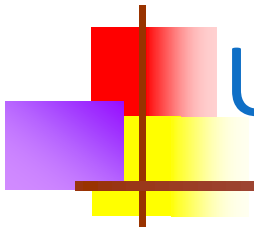
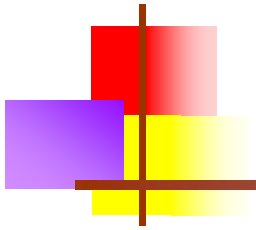


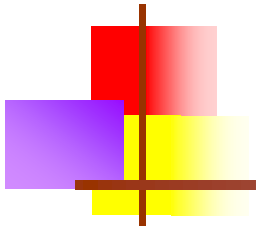
Union - Find



Union - Find

- Πως χειριζόμαστε τις διαμερίσεις πάνω στα στοιχεία του σύμπαντος U?

- Έστω ότι U={0,1,…., N-1}, x ∈ U και A,B,C ονόματα συνόλων
  - Find(x) : Επιστρέφει το όνομα του συνόλου στο οποίο ανήκει στο x
  - Union(A,B,C) : Ενώνει τα σύνολα A και B κάτω από το κοινό όνομα της C.



Για τις παραπάνω πράξεις υποθέτουμε ότι $x \in U$ και ότι A, B και C είναι τα ονόματα των συνόλων. Επιπλέον υποθέτουμε ότι ξεκινάμε από μια διαμέριση του U σε μονοσύνολα και ότι το όνομα του μονοσυνόλου $\{i\}$ είναι i.

Το Union - Find έχει αρκετές εφαρμογές ιδίως σε γραφοθεωρητικούς αλγορίθμους (υπολογισμός γεννητικών δέντρων ελαχίστου κόστους) και στον υπολογισμό του ελάχιστου κοινού προγόνου σε δέντρα. Θα παραθέσουμε δύο δομές δεδομένων για το πρόβλημα, η πρώτη που χρησιμοποιεί πίνακες και δίνει αποδοτική υλοποίηση στο Find και η δεύτερη με χρήση δέντρων που δίνει αποδοτική υλοποίηση στο Union.

Υλοποίηση Find

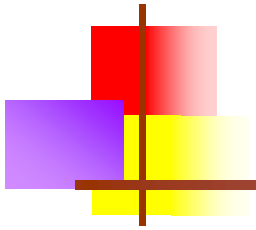
- **Απλή Λύση**
  - Για κάθε στοιχείο έχουμε το σύνολο που ανήκει
  - Πράξη Union(A,B,C) -> απλή με κόστος O(N)

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

Υλοποίηση Find

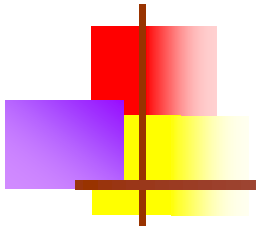
- Χρήση ανεστραμμένων λιστών
  - Μια για κάθε σύνολο
  - L(A) = {0,1,…}
  - L(B) = {2,…,N-1}
- Find είναι το ίδιο με πριν
- Union εκτελείται σε χρόνο O({AUB})
  - Πιο αποδοτικό αλλαγή μόνο στο όνομα των στοιχείων του μικρότερου συνόλου στο όνομα του άλλου εσωτερικά
  - Κρατάμε και απεικόνιση για τον 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



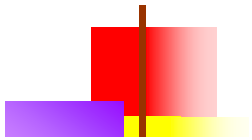
UNION(A,B,C)

1. If size(A)<=size(B) then z← MAPIN[A]; y<-MAPIN[B];
2. else z<- MAPIN[B]; y<-MAPIN[A];
3. For x in List[z] IS_IN[x]←y;
4. List[y]← List[z] ∪ List[y];
5. MAPOUT[y]←C;
6. MAPIN[C]←y;
7. size[C]←size[A]+size[B];



Αποδοτικό Find

- Μια ακολουθία από
  - n-1 Unions και (Union κόστος επιμερισμένο O(logn))
  - m Find (Find κόστος O(1))
  - σε σύνολα αρχικού μεγέθους 1
    - έχει κόστος O(m + nlogn)
    - απόδειξη στο βιβλίο

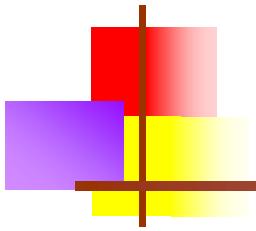


Θεώρημα 10.1. *Μια ακολουθία από* $n-1$ *Union και* m *Find σε σύνολα αρχικού μεγέθους 1 έχει κόστος* $O(m + n \log n)$.

Απόδειξη. Κάθε Find κοστίζει $O(1)$, οπότε τα m Find κοστίζουν $O(m)$. Ας δούμε τώρα τον χρόνο για τα Union. Ένα επιμέρους Union μπορεί να έχει κόστος $O(n)$ αλλά το επιμερισμένο κόστος μιας πράξης είναι αρκετά μικρότερο. Η πράξη Union(A, B, C) έχει κόστος $O(\min\{size[A], size[B]\}) \leq c \cdot \min\{size[A], size[B]\}$, για μια σταθερά c, από τον ορισμό του ασυμπτωτικού συμβολισμού. Συνεπώς μπορούμε να θεωρήσουμε ότι κάθε στοιχείο x του μικρότερου από τα δύο σύνολα πληρώνει c Ευρώ για να καλύψει το κόστος της αλλαγής ονόματος στο $IS - IN[x]$.

Το κλειδί της απόδειξης είναι ότι κανένα στοιχείο δεν πληρώνει παραπάνω από $c \cdot \log n$ συνολικά για όλες τις αλλαγές συνόλων που θα υποστεί κατά τη διάρκεια των $n-1$ Union.

Επισημαίνεται πάλι, ότι κάθε φορά που πληρώνει χρήματα ένα στοιχείο, τότε ανήκει στο μικρότερο από τα δύο σύνολα που μετέχουν στο Union, δηλαδή στο σύνολο z. Το τελικό σύνολο C θα περιέχει τουλάχιστον τα διπλάσια στοιχεία από το z γιατί στη χειρότερη περίπτωση $size[A] = size[B]$ στη γραμμή 1 και $size[C] = size[A] + size[B]$ στη γραμμή 12. Ισχύουν και τα παρακάτω δύο γεγονότα:



i) Τα αρχικά σύνολα έχουν μέγεθος 1

ii) Μετά από $n-1$ Union το μέγιστο μέγεθος συνόλου είναι $\leq n$.

Συνεπώς θα έχουμε την εξής ακολουθία:

$$1 \xrightarrow{c} 2 \xrightarrow{c} 2^2 \xrightarrow{c} \ldots \xrightarrow{c} 2^k$$

Αποδοτική υλοποίηση Union

Σε αυτή την προσέγγιση κάθε διαμέριση (ομάδα) αναπαρίσταται σαν ένα δένδρο όπου τα στοιχεία της ομάδος βρίσκονται στους κόμβους και στα φύλλα. Το όνομα κάθε τέτοιας ομάδας βρίσκεται στη ρίζα του δένδρου που την αναπαριστά. Η ένωση δύο ομάδων όπως θα δούμε στην συνέχεια μπορεί να γίνει σε σταθερό χρόνο, θέτοντας το δένδρο της μίας ομάδας κάτω από την ρίζα του δένδρου της άλλης, ενώ το κόστος της πράξης $\mathrm{Find}(x)$ εξαρτάται από το βάθος που βρίσκεται το στοιχείο x στο δένδρο που αναπαριστά την ομάδα του, αφού από τον κόμβο αυτό πρέπει να προσεγγισθεί η ρίζα του δένδρου και να προσπελασθεί το όνομα της ομάδος.

Αποδοτική υλοποίηση Union

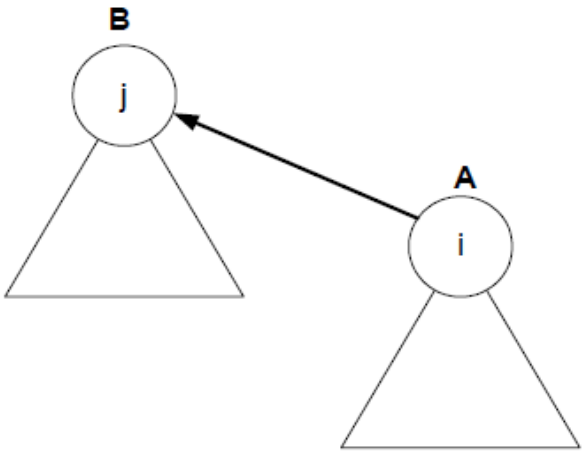
- Καταναλώνει χρόνο Find για σταθερό χρόνο στο Union
- 3 πίνακες
  - Father [1…N] : το i-οστό στοιχείο δείχνει τον πατέρα j στον ίδιο πίνακα
  - Name [1…N] : το i-οστό στοιχείο δείχνει το ονομα της ομάδας του i αν είναι η ρίζα στην ομάδα αλλιώς κενό
  - Root [] : το i-οστό στοιχείο του πίνακα δείχνει 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 <- Root[A], j <- Root[B], Father[i] <- j

FIND(x)

1. **while** $father[x] \neq nil$ **do**
2. $\quad x \leftarrow father[x]$;
3. **od**
4. print(Name[x]);

UNION(A, B, B)

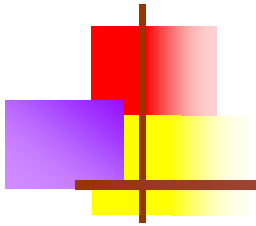
1. $i \leftarrow Root[A]$;
2. $j \leftarrow Root[B]$;
3. $Father[i] \leftarrow j$;



Σχήμα 10.1: Union(A,B,B)

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)



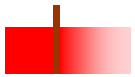
Θεώρημα 10.2. *Μια ακολουθία από* $n-1$ *Unions και* m *Finds στοιχίζει* $O(n+m\log n)$.

Απόδειξη. Έστω F το δάσος που προκύπτει μετά από $n-1$ Union. Ορίζουμε ως :

- rank(x)= το ύψος του x στο F
- weight(x)= το πλήθος των απογόνων του x στο F

Κάνοντας επαγωγή ως προς το $rank(x)$ βλέπουμε ότι :

για $rank(x) = 0 \Rightarrow 2^{rank(x)} = 1 \leq weight(x)$



για $rank(x) > 0 \Rightarrow$ Έστω y ο γιός του x με $rank(y) = rank(x-1)$. Όταν ο u έγινε γιος του x, ο x είχε τουλάχιστον τόσους απογόνους όσους και ο y. Μετά το $Union$ ο y δεν μπορεί να πάρει άλλους απογόνους (μόνο μια ρίζα παίρνει επιπλέον απογόνους μέσω μιας $Union$ πράξης) ενώ ο x μπορεί. Συνεπώς $weight(x) \geq weight(x)_{\text{before\_union}} + weight(y) = 2 \cdot weight(y) = 2 \cdot 2^{rank(x)} = 2^{rank(y)+1} = 2^{rank(x)}$

$$weight(x) \geq 2^{rank(x)} \tag{10.1}$$

Επίσης με $n-1$ Unions μπορούνε να παραχθούν μόνο σύνολα μεγέθους $\leq n$. Συνεπώς

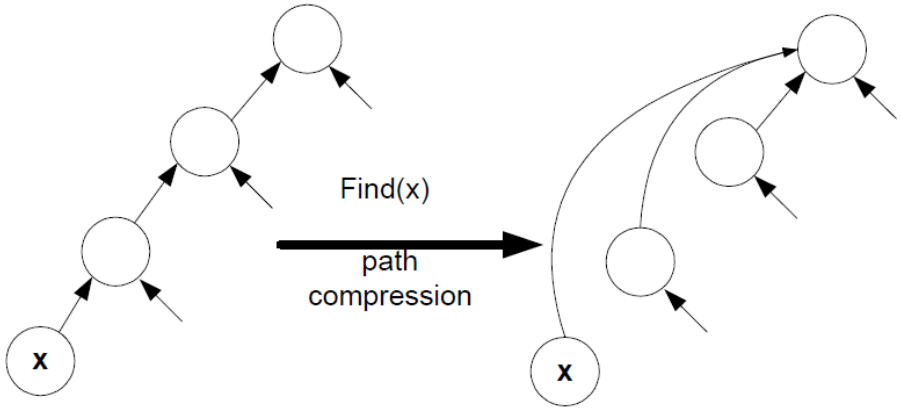
$$weight(x) \leq n \tag{10.2}$$

Από τις δύο παραπάνω σχέσεις προκύπτει ότι

$$2^{rank(x)} \leq weight(x) \leq n$$

$$2^{rank(x)} \leq n \tag{10.3}$$

Έτσι λοιπόν ένα Find θα στοιχίσει το πολύ χρόνο $\log n$. □



Σχήμα 10.3: path compression

Θεώρημα 10.3. *Μια ακολουθία από* $n-1$ *Unions και* m *Finds, με* $m \geq n$, *στοιχίζει* $O(m \cdot a(m,n))$. *Η συνάρτηση* $a(m,n)$ *είναι μια παραλλαγή της συνάρτησης του Ackermann για την οποία ισχύει*

$$a(m,n) = min\{z \geq 1 : A(z, 4\left\lceil \frac{m}{n} \right\rceil) > \log n\}.$$

Values of $A(m, n)$

m \ n	0	1	2	3	4	n
0	1	2	3	4	5	$n+1$
1	2	3	4	5	6	$n+2 = 2+(n+3)-3$
2	3	5	7	9	11	$2n+3 = 2\cdot(n+3)-3$
3	5	13	29	61	125	$2^{(n+3)}-3$
4	13 $= 2^{2^2}-3$ $= 2 \uparrow\uparrow 3-3$	65533 $= 2^{2^{2^2}}-3$ $= 2 \uparrow\uparrow 4-3$	$2^{65536}-3$ $= 2^{2^{2^{2^2}}}-3$ $= 2 \uparrow\uparrow 5-3$	$2^{2^{65536}}-3$ $= 2^{2^{2^{2^{2^2}}}}-3$ $= 2 \uparrow\uparrow 6-3$	$2^{2^{2^{65536}}}-3$ $= 2^{2^{2^{2^{2^{2^2}}}}}-3$ $= 2 \uparrow\uparrow 7-3$	$\underbrace{2^{2^{\cdot^{\cdot^{\cdot^2}}}}}_{n+3}-3$ $= 2 \uparrow\uparrow (n+3)-3$
5	65533 $= 2 \uparrow\uparrow (2 \uparrow\uparrow 2)-3$ $= 2 \uparrow\uparrow\uparrow 3-3$	$2 \uparrow\uparrow\uparrow 4-3$	$2 \uparrow\uparrow\uparrow 5-3$	$2 \uparrow\uparrow\uparrow 6-3$	$2 \uparrow\uparrow\uparrow 7-3$	$2 \uparrow\uparrow\uparrow (n+3)-3$
6	$2 \uparrow\uparrow\uparrow\uparrow 3-3$	$2 \uparrow\uparrow\uparrow\uparrow 4-3$	$2 \uparrow\uparrow\uparrow\uparrow 5-3$	$2 \uparrow\uparrow\uparrow\uparrow 6-3$	$2 \uparrow\uparrow\uparrow\uparrow 7-3$	$2 \uparrow\uparrow\uparrow\uparrow (n+3)-3$
m	$(2 \to (3) \to (m-2))-3$	$(2 \to (4) \to (m-2))-3$	$(2 \to (5) \to (m-2))-3$	$(2 \to (6) \to (m-2))-3$	$(2 \to (7) \to (m-2))-3$	$(2 \to (n+3) \to (m-2))-3$