Δομές Δεδομένων

2η Διάλεξη

Αλγόριθμοι Ένωσης-Εύρεσης (Union-Find)

Ε. Μαρκάκης

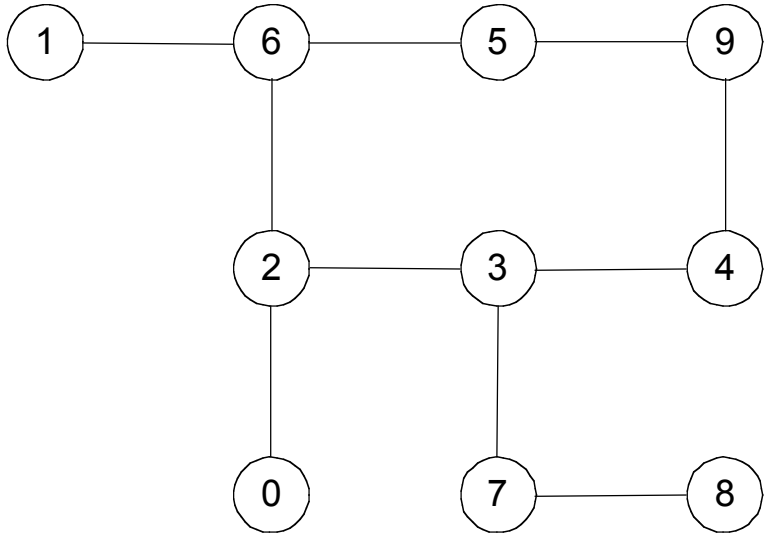
Βασίζεται στις διαφάνειες των R. Sedgewick – K. Wayne

Περίληψη

- Συνδετικότητα δικτύου
- Αφαιρέσεις
- Συνδεδεμένα συστατικά
- Αφηρημένη εύρεση-ένωση
- Γρήγορη εύρεση
- Γρήγορη ένωση
- Σταθμισμένη γρήγορη ένωση
- Σταθμισμένη γρήγορη ένωση με συμπίεση μονοπατιών

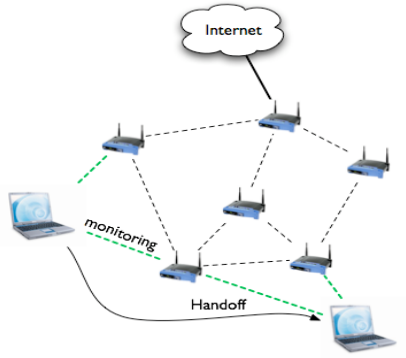
Προβλήματα Συνδετικότητας

- *Γράφος (graph)*:
  - Ένα σύνολο από αντικείμενα (κόμβοι ή κορυφές)
  - Πλευρές (σύνδεσμοι μεταξύ αντικειμένων)
  - Αν έχουμε N αντικείμενα, τα ονοματίζουμε από 0 ως N-1



Προβλήματα Συνδετικότητας

- Τα αντικείμενα μπορεί να απεικονίζουν:
  - Κόμβους σε ένα δίκτυο υπολογιστών (δρομολογητές, πάροχοι δικτύου, κτλ)
  - Ιστοσελίδες (web graph)
  - Τρανζίστορ ενός μικροτσιπ
  - Χρήστες του facebook (κοινωνικά δίκτυα)
  - Πρωτείνες (βιολογικά δίκτυα για την αλληλεπίδραση πρωτεϊνών)

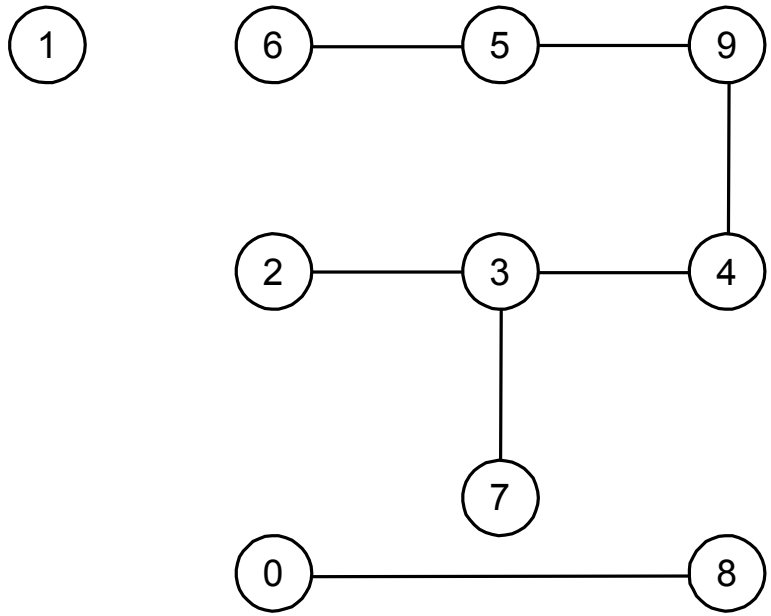


Συνδετικότητα δικτύου

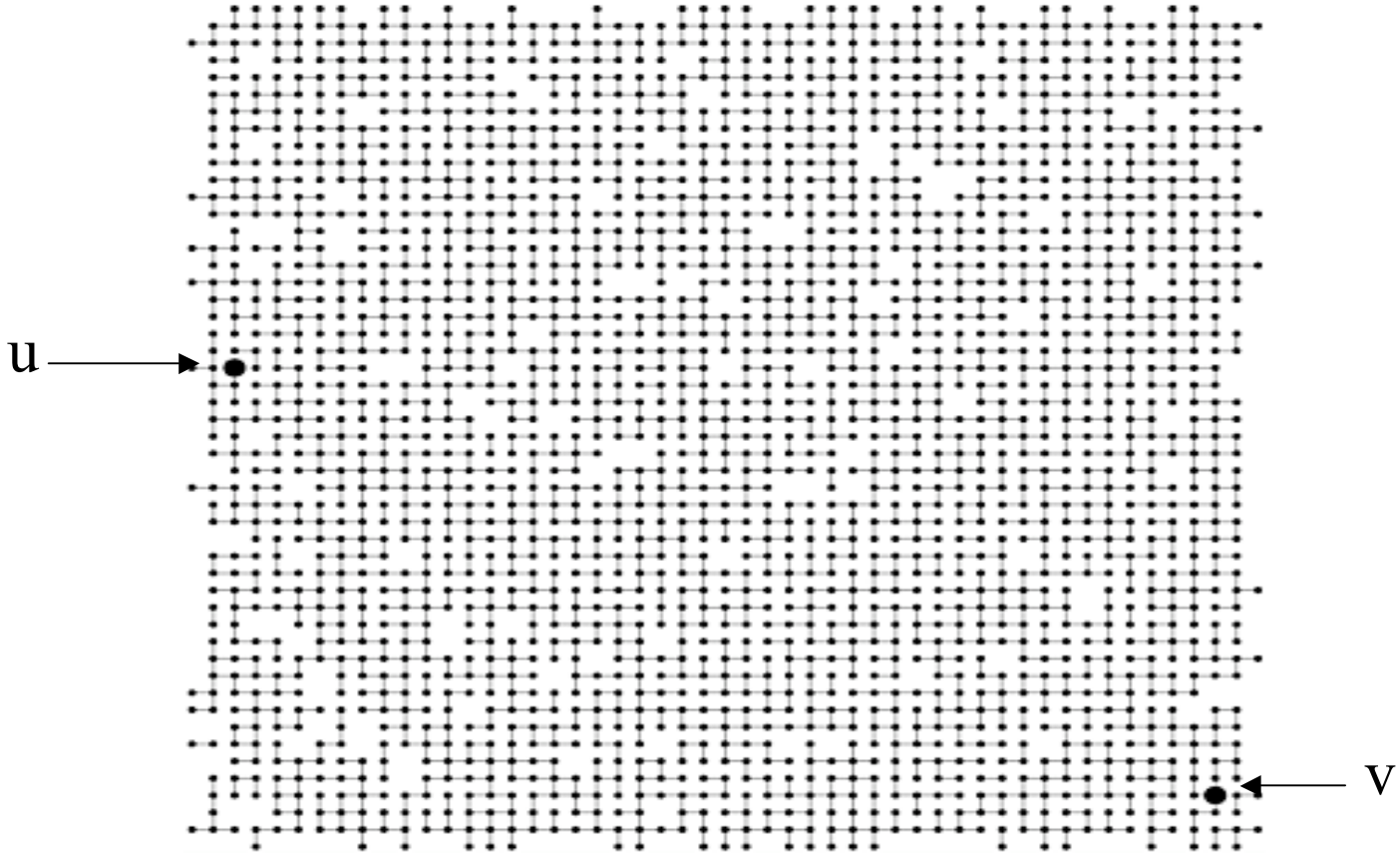
- Είσοδος: Δίνεται μία ακολουθία από ζεύγη ακεραίων
  - Κάθε ακέραιος είναι ένα αντικείμενο
  - Κάθε ζεύγος αντιπροσωπεύει μία σύνδεση
  - Η συνδετικότητα είναι μεταβατική σχέση:
    - Αν συνδέεται(p,q) και συνδέεται(q,r) τότε συνδέεται(p,r)
- Πρόβλημα: εντοπισμός των περιττών ζευγών
  - Ένα ζεύγος (p, q) είναι περιττό αν από τα ζεύγη που έχουν εξεταστεί πριν το (p, q) συνεπάγεται ότι συνδέεται(p,q).
  - Στην έξοδο τυπώνονται μόνο τα μη περιττά ζεύγη
- Προσοχή στο τι ακριβώς ζητάει το πρόβλημα!
  - Δεν ζητάει το μονοπάτι ανάμεσα στα αντικείμενα
  - Η εύρεση του μονοπατιού μπορεί να είναι πιο δύσκολη

Παράδειγμα

Είσοδος	Έξοδος
3 4	3 4
4 9	4 9
8 0	8 0
2 3	2 3
5 6	5 6
2 9	
5 9	5 9
7 3	7 3
7 5	
5 6	



Συνδετικότητα δικτύου



Συνδέεται το u με το v?

Αφαιρέσεις

- Ποιες είναι οι βασικές λειτουργίες που χρειαζόμαστε?

- Απλό μοντέλο της φύσης της συνδεσιμότητας
  - Αντικείμενα

  0 1 2 3 4 5 6 7 8 9 — κόμβοι του γράφου
  - Σύνολα από αντικείμενα που συνδέονται μεταξύ τους (ανανεώνεται κατά τη διάρκεια του αλγορίθμου)

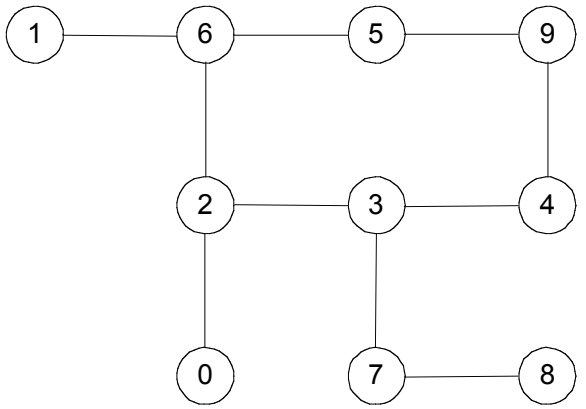
  0 1 { 2 3 9 } { 5 6 } 7 { 4 8 } — συνδεδεμένα υποσύνολα
  - Ερώτηση εύρεσης: τα 2 και 9 ανήκουν στο ίδιο σύνολο;

  0 1 { 2 3 9 } { 5 6 } 7 { 4 8 } — συνδέονται δύο σημεία;
  - Εντολή ένωσης: συγχώνευσε τα σύνολα με το 3 και το 8

  0 1 { 2 3 4 8 9 } { 5 6 } 7 — νέα σύνδεση μεταξύ σημείων

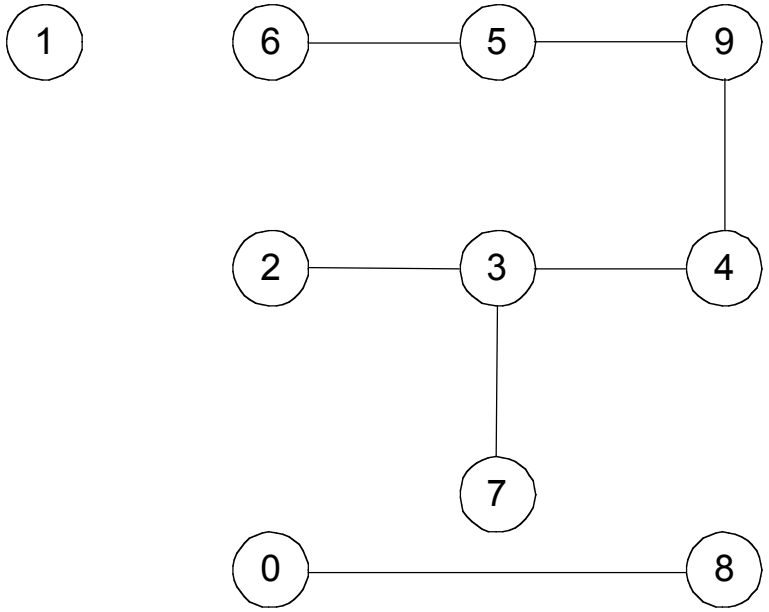
Αφαιρέσεις

- Οι αλγόριθμοι ένωσης-εύρεσης (union-find) διαχειρίζονται αντικείμενα στα οποία αντιστοιχίζουμε τους ακεραίους 0 έως N-1
  - Αποκρύπτονται λεπτομέρειες άσχετες με την ένωση-εύρεση
  - Οι ακέραιοι επιτρέπουν γρήγορη προσπέλαση στα αντικείμενα
    - Χρήση ως δείκτες σε πίνακες
  - Πίνακας συμβόλων για αντιστοίχιση
    - Μετάφραση ονομάτων σε αριθμούς



Συνδεδεμένα συστατικά

- Σύνολο συνδεδεμένων σημείων: Συνδεδεμένο συστατικό (connected component)
- Κάθε εντολή ένωσης μειώνει τα σύνολα κατά ένα
  - Είσοδος Έξοδος
  - 3 4 3 4
  - 4 9 4 9
  - 8 0 8 0
  - 2 3 2 3
  - 5 6 5 6
  - 2 9
  - 5 9 5 9
  - 7 3 7 3
  - Έχουμε 10 – 7 = 3 συστατικά



Αφηρημένη ένωση-εύρεση

- Γενική μορφή της λύσης του προβλήματός μας
- Για κάθε νέο ζεύγος (p, q) που διαβάζουμε:
  - Κάνε μία ερώτηση εύρεσης για να δούμε αν ήδη συνδέεται το p με το q
  - Αν ναι, τότε το (p, q) είναι περιττό ζεύγος
  - Αν όχι, το (p, q) δεν είναι περιττό: εκτέλεσε μία εντολή ένωσης για να ενώσουμε τα σύνολα που περιέχουν το p και το q
- Στόχος: Σχεδιασμός αποδοτικής δομής δεδομένων
- Παρατηρήσεις
  - Οι ευρέσεις και οι ενώσεις μπορούν να αναμειγνύονται
    - Τα ερωτήματα εύρεσης απαντώνται μόλις τεθούν
  - Το πλήθος των εντολών M μπορεί να είναι τεράστιο
  - Το πλήθος των αντικειμένων N μπορεί να είναι τεράστιο

Γρήγορη εύρεση (Quick Find)

- Δομή δεδομένων
  - Χρήση πίνακα ακεραίων `id[]` μεγέθους N
  - Ερμηνεία: τα p και q συνδέονται αν έχουν το ίδιο `id`

```
i      0  1  2  3  4  5  6  7  8  9
id[i]  0  1  9  9  9  6  6  7  8  9
```

  - Υλοποίηση Εύρεσης: Έλεγξε αν τα p και q έχουν το ίδιο id
    - Π.χ. Αν `(p,q) = (3,6)`, `id[3]=9` και `id[6]=6` άρα τα 3 και 6 δεν συνδέονται
  - Υλοποίηση Ένωσης: Συγχώνευση των συστατικών των p και q
    - Αλλαγή των καταχωρήσεων με `id[p]` σε `id[q]`
    - Ένωση των συστατικών που περιέχουν τα 3 και 6

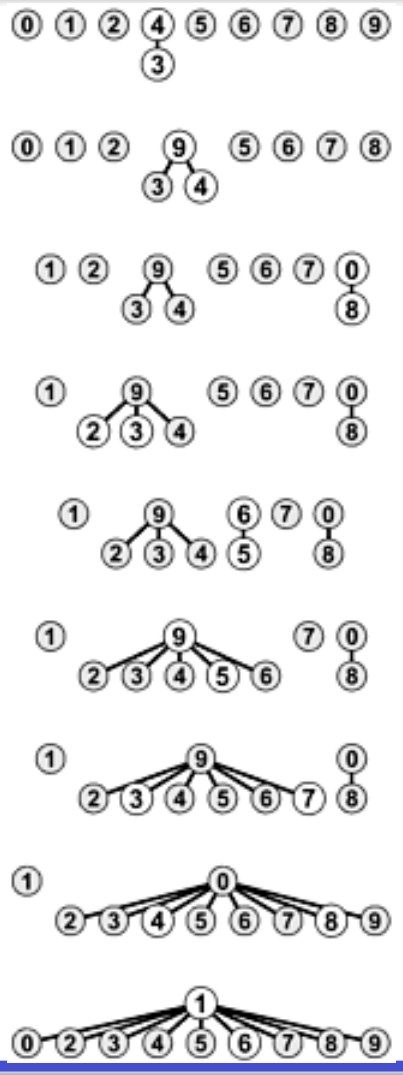
```
i      0  1  2  3  4  5  6  7  8  9
id[i]  0  1  6  6  6  6  6  7  8  6
```

Γρήγορη εύρεση

```
public class QuickF {
  public static void main(String[] args) {
    int N = Integer.parseInt(args[0]);
    int id[] = new int[N];
    for (int i = 0; i < N ; i++)
          id[i] = i;
    for( In.init(); !In.empty(); ) {
          int p = In.getInt(), q = In.getInt();
          int t = id[p];
          if (t == id[q]) continue; //περιττό ζεύγος
          for (int i = 0;i<N;i++)
                if (id[i] == t) id[i] = id[q];
          Out.println(" " +p+""+q);
    }
  }
}
```

Γρήγορη εύρεση

p	q	0	1	2	3	4	5	6	7	8	9
		0	1	2	3	4	5	6	7	8	9
3	4	0	1	2	4	4	5	6	7	8	9
4	9	0	1	2	9	9	5	6	7	8	9
8	0	0	1	2	9	9	5	6	7	0	9
2	3	0	1	9	9	9	5	6	7	0	9
5	6	0	1	9	9	9	6	6	7	0	9
2	9	0	1	9	9	9	6	6	7	0	9
5	9	0	1	9	9	9	9	9	7	0	9
7	3	0	1	9	9	9	9	9	9	0	9
4	8	0	1	0	0	0	0	0	0	0	0
5	6	0	1	0	0	0	0	0	0	0	0
0	2	0	1	0	0	0	0	0	0	0	0
6	1	1	1	1	1	1	1	1	1	1	1



Πρόβλημα: μπορεί να αλλάζουν πολλά στοιχεία του πίνακα `id[]` πολύ συχνά

Γρήγορη εύρεση

- Πόσο γρήγορη είναι η γρήγορη εύρεση;
  - Μπορεί να πάρει MN βήματα για M ενώσεις με N αντικείμενα
- Αριθμητικό παράδειγμα
  - Δίνονται 10^{10} ζεύγη που συνδέουν N =10^9 αντικείμενα
  - Με πρόχειρους υπολογισμούς για ένα σύγχρονο υπολογιστή η γρήγορη ένωση απαιτεί περίπου 300 χρόνια
- Τι γίνεται αν δεκαπλασιαστεί η ταχύτητα/χωρητικότητα;
  - Στη μνήμη χωράει δεκαπλάσιο πρόβλημα
  - Ο χρόνος που θα χρειαστεί όμως είναι κι αυτός δεκαπλάσιος!
  - 10N x 10M / 10 = 10 x N x M

Γρήγορη ένωση (Quick Union)

- Ιδέα: Απεικόνιση κάθε συνδεδεμένου συνόλου ως δέντρο
  - Κάνει πολύ γρήγορη την ένωση με αντίτιμο στην εύρεση
- Δομή δεδομένων
  - Πίνακας ακεραίων `id[]` μεγέθους N
  - Ερμηνεία: το `id[i]` είναι ο πατέρας του i
  - Η ρίζα στο δέντρο που βρίσκεται ο i είναι το `id[id[id[…id[i]…]]]`
    - Σταματάμε όταν `id[j]=j` (μόνο η ρίζα έχει πατέρα τον εαυτό της)

```
i      0  1  2  3  4  5  6  7  8  9
id[i]  0  1  9  4  4  6  6  7  8  9
```

  - Εύρεση: Έλεγξε αν τα p και q έχουν την ίδια ρίζα έστω π.χ. το (4, 9)
  - Ένωση: Συγχώνευση των συστατικών των p και q
    - «Κρεμάμε» το δέντρο του p στη ρίζα του q

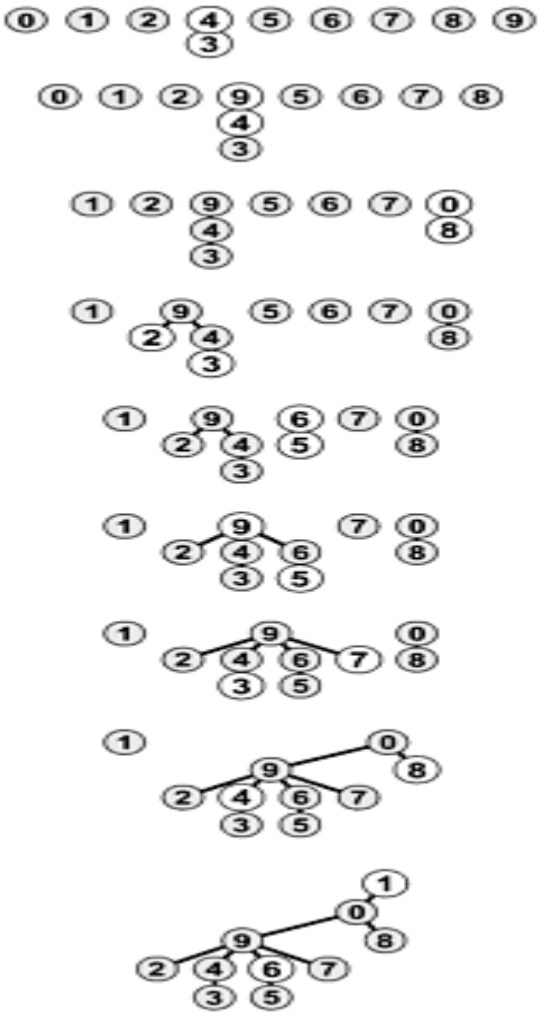
```
i      0  1  2  3  4  5  6  7  8  9
id[i]  0  1  9  4  9  6  6  7  8  9
```

Γρήγορη ένωση

```
public class QuickU {
  public static void main(String[] args) {
    int N = Integer.parseInt(args[0]);
    int id[] = new int[N];
    for (int i = 0; i < N ; i++)
          id[i] = i;
    for( In.init(); !In.empty(); ) {
          int i, j, p = In.getInt(), q = In.getInt();
          for (i = p; i != id[i]; i = id[i]);
          for (j = q; j != id[j]; j = id[j]);
          if (i == j) continue;
          id[i] = j;
          Out.println(" " + p + " " + q);
    }
  }
}
```

Γρήγορη ένωση

p	q	0	1	2	3	4	5	6	7	8	9
3	4	0	1	2	4	4	5	6	7	8	9
4	9	0	1	2	4	9	5	6	7	8	9
8	0	0	1	2	4	9	5	6	7	0	9
2	3	0	1	9	4	9	5	6	7	0	9
5	6	0	1	9	4	9	6	6	7	0	9
2	9	0	1	9	4	9	6	6	7	0	9
5	9	0	1	9	4	9	6	9	7	0	9
7	3	0	1	9	4	9	6	9	9	0	9
4	8	0	1	9	4	9	6	9	9	0	0
5	6	0	1	9	4	9	6	9	9	0	0
0	2	0	1	9	4	9	6	9	9	0	0
6	1	1	1	9	4	9	6	9	9	0	0
5	8	1	1	9	4	9	6	9	9	0	0



Πρόβλημα: μπορεί να διατρέξουμε πολλούς συνδέσμους

Γρήγορη Ένωση

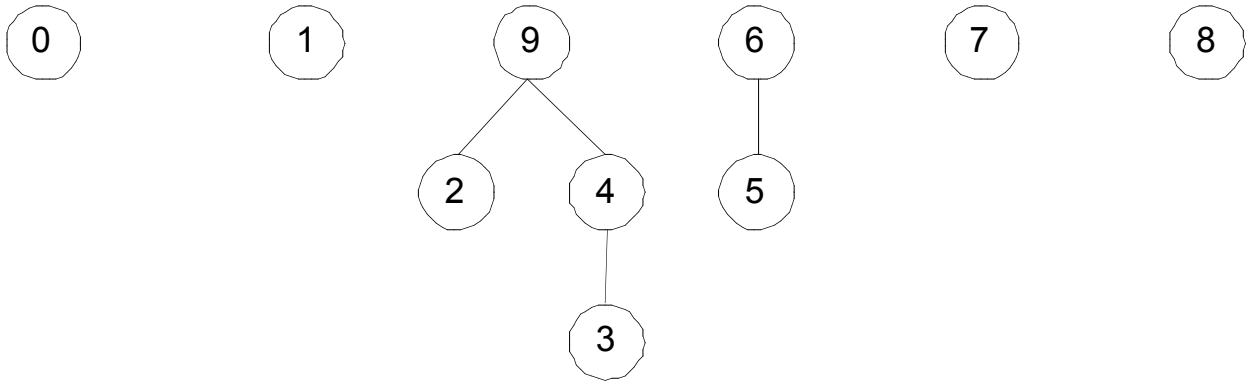
- Το πόσο ρηχά θα είναι τα δέντρα εξαρτάται από την είσοδο
  - Είσοδος Έξοδος
  - 8 7 8 7
  - 7 6 7 6
  - 6 5 6 5
  - 5 4 5 4
  - 4 3 4 3
  - 3 2 3 2
  - 2 1 2 1
  - 1 0 1 0
- Εδώ σε κάθε βήμα πρέπει να διατρέξουμε όλο το δέντρο για να βρούμε τη ρίζα (O(N) βήματα για τις τελευταίες συνδέσεις)

Γρήγορη ένωση

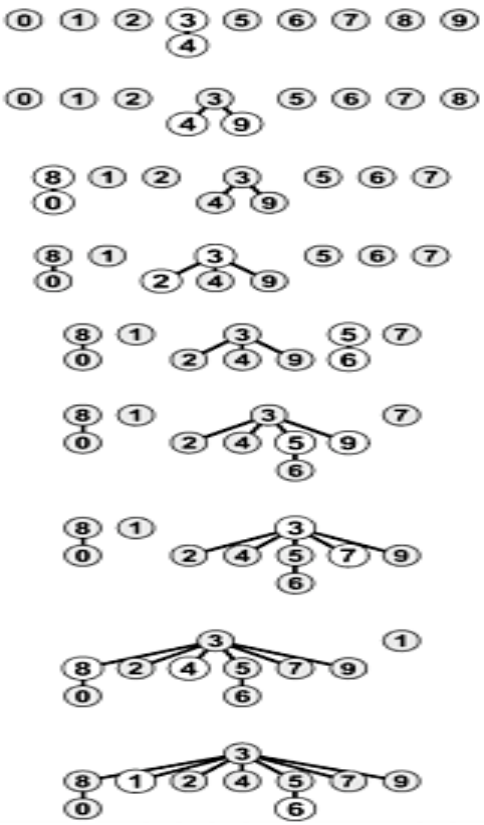
- Πόσο γρήγορη είναι η γρήγορη ένωση;
  - Δεν είναι πολύ πιο γρήγορη από τη γρήγορη εύρεση
  - Παρόμοια τάξη μεγέθους (εξαρτάται και από τη φύση της είσοδου)
- Μειονέκτημα γρήγορης εύρεσης
  - Η ένωση είναι αργή (διατρέχουμε όλο τον πίνακα)
  - Τα δέντρα είναι ρηχά αλλά το κόστος της ένωσης είναι μεγάλο
- Μειονέκτημα γρήγορης ένωσης
  - Τα δέντρα μπορεί να έχουν βάθος. Χειρότερη περίπτωση: όταν το δέντρο που σχηματίζεται είναι μία ευθεία γραμμή με βάθος N-1
  - Η εύρεση είναι αργή (μπορεί να διατρέξουμε τον πίνακα)
  - Για να γίνει η ένωση πρέπει πρώτα να κάνουμε την εύρεση
- Στόχος: ρηχά δέντρα αλλά με μικρό κόστος!

Σταθμισμένη γρήγορη ένωση

- Παραλλαγή της γρήγορης ένωσης
  - Τροποποιούμε την ένωση για να κρατάμε τα δέντρα ρηχά
  - Παρακολουθούμε το μέγεθος κάθε δέντρου
    - Χρήση ενός πρόσθετου πίνακα για να κρατάμε το μέγεθος
  - Το μικρότερο δέντρο τοποθετείται κάτω από το μεγαλύτερο
    - Το μέγεθος αναφέρεται στο σύνολο των κόμβων
  - Παράδειγμα: αν διαβάσουμε το ζεύγος (3, 5)
    - Η σταθμισμένη ένωση τοποθετεί το 6 κάτω από το 9



Σταθμισμένη γρήγορη ένωση



Τα δέντρα παραμένουν ρηχά

Σταθμισμένη γρήγορη ένωση

- Υλοποίηση σε Java
  - Σχεδόν ίδια με τη γρήγορη ένωση
  - Χρήση πρόσθετου πίνακα `sz[]` για το πλήθος κόμβων κάθε δέντρου
    - Έχει νόημα μόνο για τις ρίζες των δένδρων
  - Εύρεση: ίδια με αυτή στη γρήγορη ένωση
  - Ένωση: παραλλαγή της γρήγορης ένωσης
    - Συγχώνευση του μικρότερου δέντρου κάτω από το μεγαλύτερο
    - Αρχικοποίηση όλων των δένδρων σε μέγεθος 1
    - Ενημέρωση του πίνακα `sz[]`

```
if (sz[i] < sz[j]) {
    id[i] = j; sz[j] += sz[i];
} else {
    id[j] = i; sz[i] += sz[j];
}
```

Σταθμισμένη γρήγορη ένωση

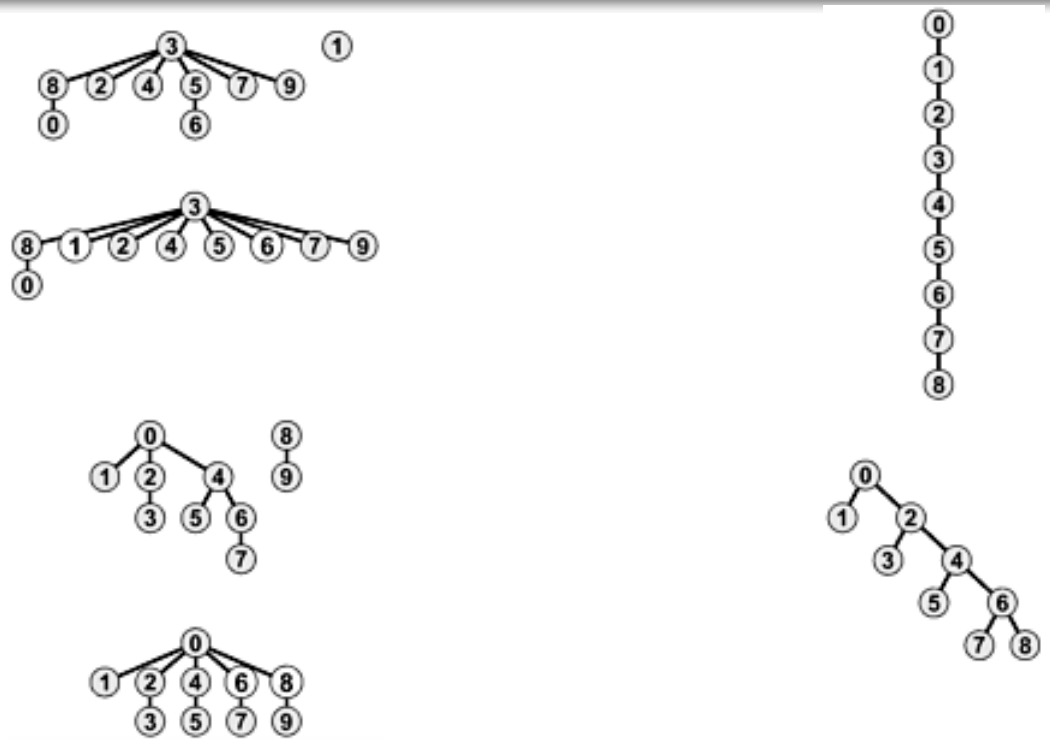
- Απόδοση της σταθμισμένης γρήγορης ένωσης
  - Εύρεση: χρόνος ανάλογος με το βάθος των p και q
  - Ένωση: σταθερός χρόνος αφού βρεθούν οι ρίζες
  - Αποδεικνύεται ότι το βάθος είναι μέχρι logN ($=\log_2 N$)
  - Άρα εύρεση-ένωση είναι ανάλογες του logN
  - Πολύ καλύτερη από τις προηγούμενες μεθόδους!
- Σταματάμε εδώ ή μπορούμε να κάνουμε κάτι ακόμα καλύτερο;
  - Μπορούμε: συμπίεση μονοπατιών!
  - Για να βρούμε τη ρίζα διατρέχουμε ένα δέντρο
    - Από τον κόμβο εκκίνησης ως τη ρίζα
  - Σε κάθε βήμα συμπιέζουμε το δέντρο
    - Τοποθετούμε τον τρέχοντα κόμβο κάτω από τη ρίζα
  - Το δέντρο συμπιέζεται σε κάθε εύρεση

Ένωση με συμπίεση μονοπατιών

- Βασική υλοποίηση συμπίεσης μονοπατιών
  - Προσθήκη δεύτερου βρόχου
  - Κάνει τον πατέρα κάθε κόμβου ίσο με τη ρίζα
  - Απαιτεί δεύτερο πέρασμα της διαδρομής
- Απλούστερη υλοποίηση συμπίεσης μονοπατιών
  - Κάθε δεύτερος κόμβος δείχνει στον πατέρα του πατέρα του
  - Το δέντρο συμπιέζεται κατά το ήμισυ
  - Στην πράξη η διαφορά είναι μικρή

```
for (i = p; i != id[i]; i = id[i])
    id[i] = id[id[i]];
for (j = q; j != id[j]; j = id[j])
    id[j] = id[id[j]];
```

Ένωση με συμπίεση μονοπατιών



- Παραδείγματα
  - Παράδειγμα 1: σχετικά ρηχά δέντρα, προσθήκη του 1-6
  - Παράδειγμα 2: λίγο βαθιά δέντρα, προσθήκη του 6-8
  - Παράδειγμα 3: πολύ βαθιά δέντρα, μειώνεται το βάθος στο μισό περίπου

Ένωση με συμπίεση μονοπατιών

- Σύγκριση των αλγορίθμων
  - F: γρήγορη εύρεση, U: γρήγορη ένωση
  - W: σταθμισμένη γρήγορη ένωση
  - P: με πλήρη συμπίεση, H: με ημιδιπλασιασμό

N	M	F	U	W	P	H
1000	3819	63	53	17	18	15
2500	12263	185	159	22	19	24
5000	21591	698	697	34	33	35
10000	41140	2891	3987	85	101	74
25000	162748			237	267	267
50000	279279			447	533	473
100000	676113			1382	1238	1174

Άλλες Εφαρμογές των μεθόδων Ένωσης-Εύρεσης

- Το παιχνίδι Hex [Piet Hein 1942, John Nash 1948, Parker Brothers 1962]

- Επεξεργασία Εικόνων