

ΑΤΔ, ΔΔ, Union Find, Λεξικά, Hash Tables

Αλγόριθμοι και Πολυπλοκότητα



Σχολή Ηλεκτρολόγων
Μηχανικών και Μηχανικών
Υπολογιστών
Εθνικό Μετσόβιο Πολυτεχνείο

Αριστείδης Παγουρτζής,
Δώρα Σούλιου,
Παναγιώτης Γροντάς

Αρχική Έκδοση Διαφανειών:
Δημήτρης Φωτάκης

Δομές Δεδομένων

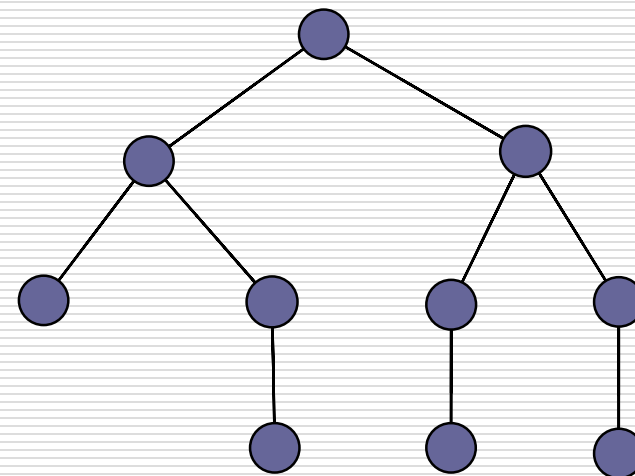
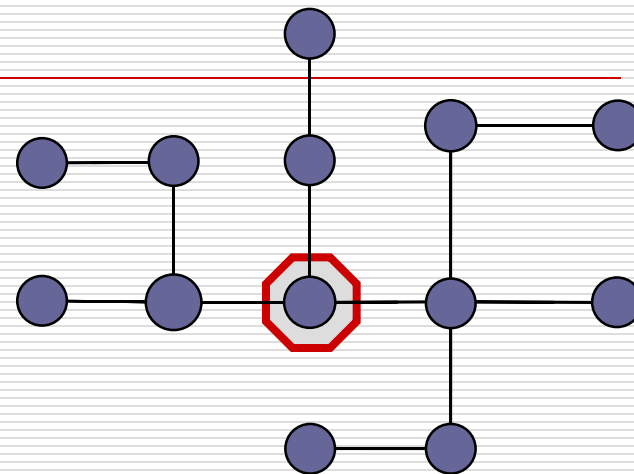
- (Αναπαράσταση,) οργάνωση και διαχείριση συνόλων αντικειμένων για αποδοτική **ενημέρωση** και **ανάκτηση** πληροφορίας.
 - Αποδοτική υλοποίηση Αλγόριθμων και Βάσεων Δεδομένων.
- (Αποδοτική) **αναπαράσταση – οργάνωση «σύνθετων»** αντικειμένων με χρήση:
 - Βασικών τύπων δεδομένων (ints, floats, chars, strings, arrays).
 - Μηχανισμών ομαδοποίησης που παρέχονται από γλώσσες προγραμματισμού (structs – records, objects).
- **Διαχείριση:** υλοποίηση στοιχειωδών λειτουργιών
 - Ταξινόμηση, αναζήτηση, min/max/median, first/last, ...
 - Εισαγωγή, διαγραφή, ενημέρωση.
- Λύσεις και τεχνικές για **αποδοτική διαχείριση** δεδομένων.
 - Ανάλυση για απαιτήσεις και καταλληλότητα.

Γενικευμένος Τύπος Δεδομένων

- **Abstract Data Type** (ADT): **σύνολο** (στιγμιότυπα) με **λειτουργίες** (μεθόδους) επί των στοιχείων του.
- **Δομή Δεδομένων: Υλοποίηση** ενός ADT
 - Αναπαράσταση – οργάνωση στιγμιτύπων και υλοποίηση λειτουργιών με κατάλληλους αλγόριθμους.
 - Διατύπωση: ορισμός αναπαράστασης και περιγραφή υλοποίησης λειτουργιών (ψευδο-κώδικας).
 - Ανάλυση: προσδιορισμός απαιτήσεων σε χώρο αποθήκευσης και χρόνο εκτέλεσης για κάθε (βασική) λειτουργία.
- **Αναλογία Java: Interface vs Κλάση**
 - Τι vs Πώς

Ιεραρχικές Δομές: Δέντρα

- Γράφημα **ακυκλικό** και **συνεκτικό**.
- Δέντρο με **n κορυφές** έχει **$m = n - 1$ ακμές**.
- Δέντρο με **ρίζα : Ιεραρχία**
- **Ύψος** : μέγιστη απόσταση φύλλου από ρίζα.
- **Δυαδικό δέντρο** : έχει **ρίζα** και κάθε κορυφή **≤ 2 παιδιά** :
 - **Αριστερό** και **δεξιό**.
 - Κάθε **υποδέντρο** είναι δυαδικό δέντρο.





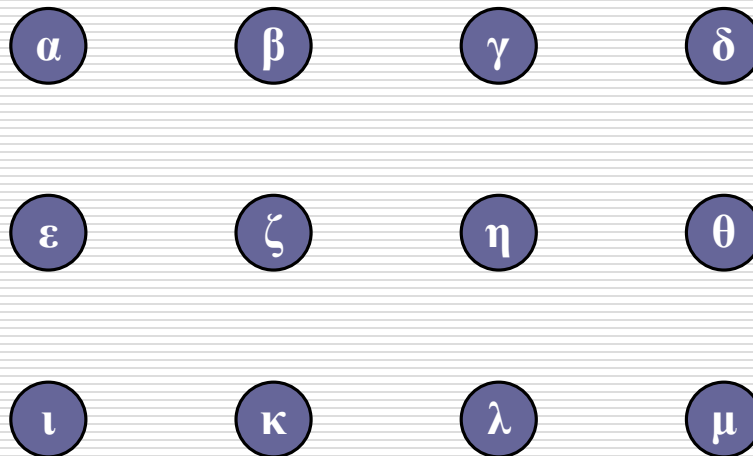
Union – Find

Disjoint Sets

Διαχείριση Διαμερίσεων

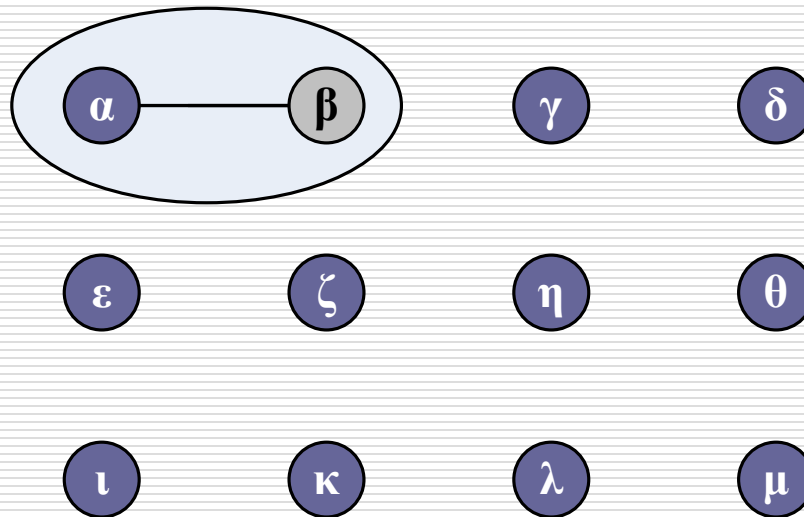
- ❑ Στοιχεία **σύμπαντος** που διαμερίζονται σε **κλάσεις ισοδυναμίας** που **μεταβάλλονται δυναμικά** με ένωση.
- ❑ Τα στοιχεία είναι **στατικά**.
- ❑ Οι διαμερίσεις **μεταβάλλονται** - μεγαλώνοντας πάντα!
- ❑ **Λειτουργίες:**
 - **Εύρεση:** Βρες κλάση όπου ανήκει κάποιο στοιχείο.
 - **Ένωση:** κλάσεων όπου ανήκουν δύο στοιχεία.
- ❑ **Εφαρμογή:**
 - Εύρεση συνεκτικών συνιστωσών ενός γραφήματος
 - Στοιχεία: κορυφές γραφήματος.
 - Κλάσεις Ισοδυναμίας: Συνεκτικές συνιστώσες.
 - Minimum Spanning Tree (Kruskal)

Διαχείριση Διαμερίσεων (Παράδειγμα)

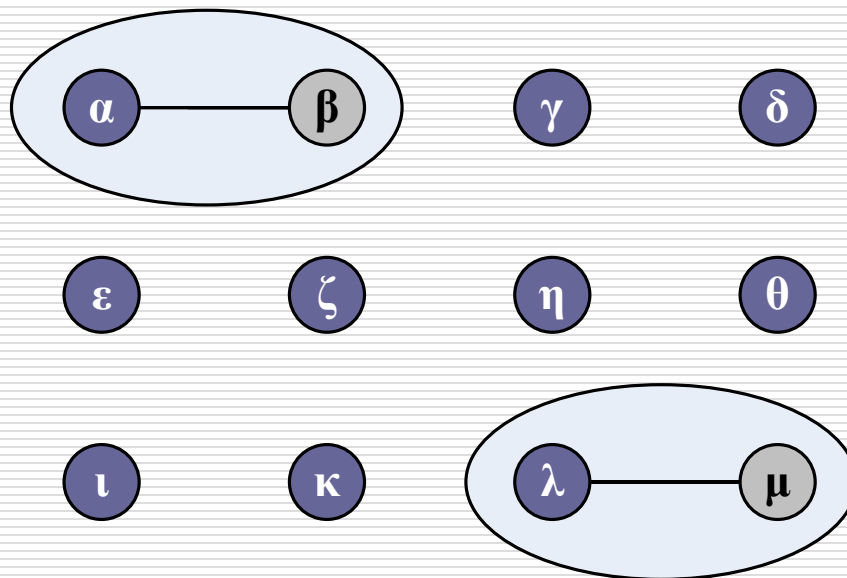


Διαχείριση Διαμερίσεων (Παράδειγμα)

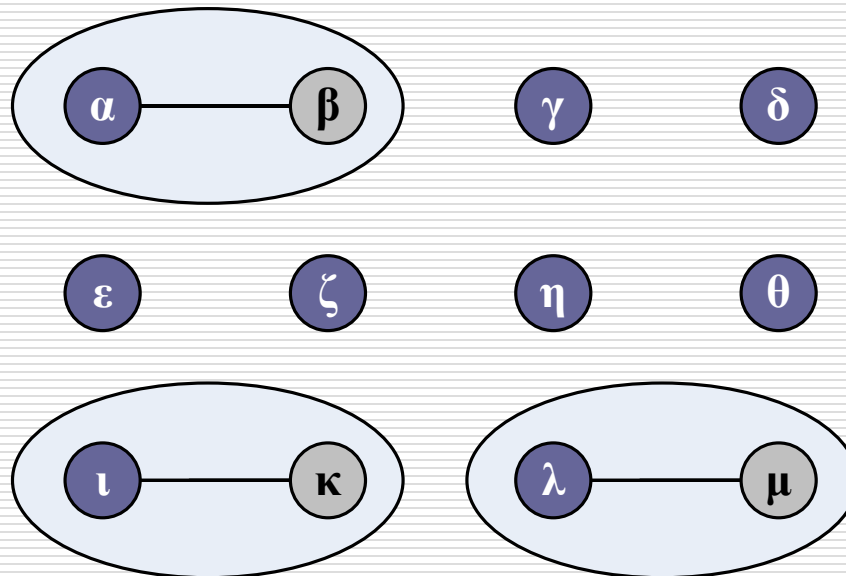
Ένωση -
δημιουργία
κλάσης και
επιλογή
αντιπροσώπου



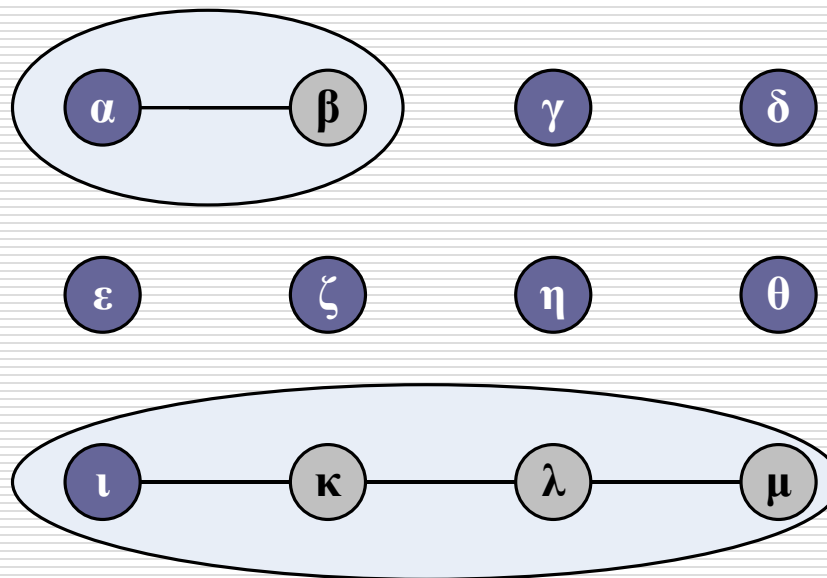
Διαχείριση Διαμερίσεων (Παράδειγμα)



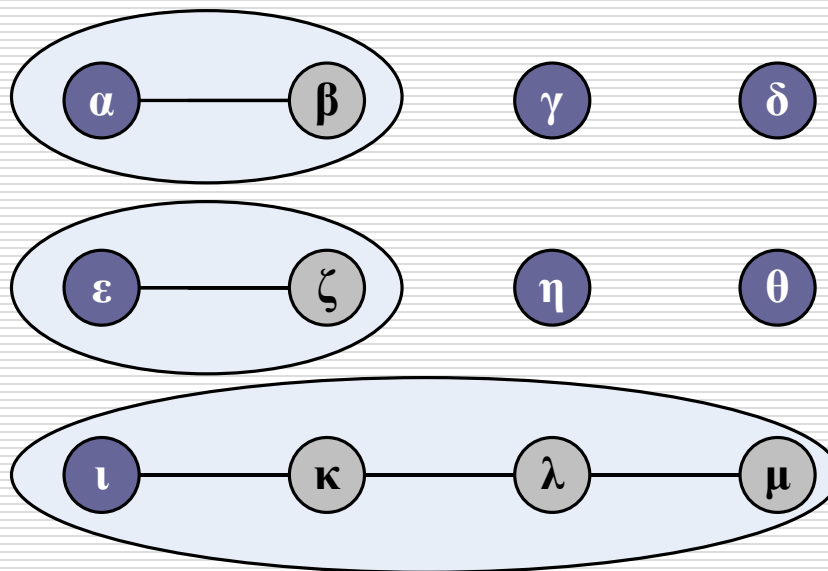
Διαχείριση Διαμερίσεων (Παράδειγμα)



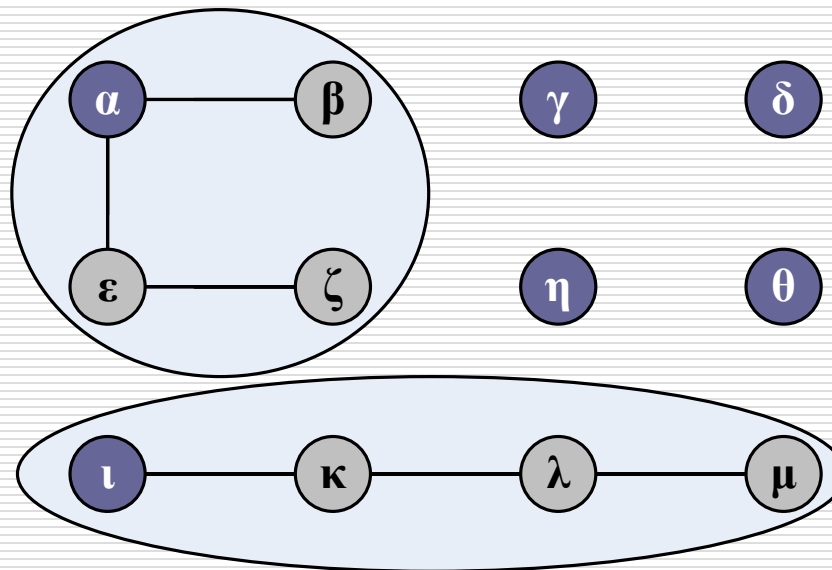
Διαχείριση Διαμερίσεων (Παράδειγμα)



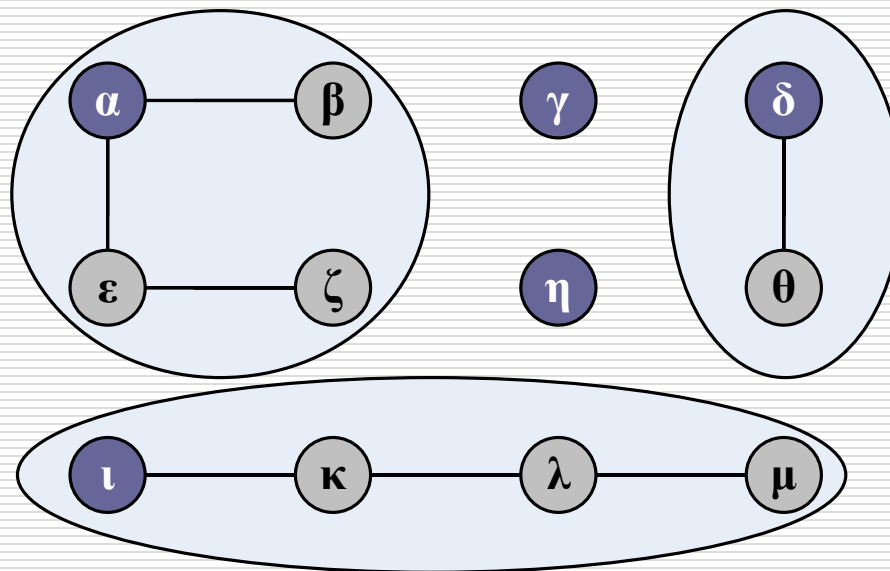
Διαχείριση Διαμερίσεων (Παράδειγμα)



Διαχείριση Διαμερίσεων (Παράδειγμα)



Διαχείριση Διαμερίσεων (Παράδειγμα)



Πρόβλημα Union – Find

- Στοιχεία $U = \{1, 2, \dots, n\}$ αρχικά σε n κλάσεις.
 - Κάθε κλάση προσδιορίζεται από στοιχείο – αντιπρόσωπο.
- $\text{find}(x)$: αντιπρόσωπος κλάσης όπου ανήκει x .
 - Διατηρούμε μοναδικό αντιπρόσωπο για κάθε κλάση.
 - Οποιοσδήποτε αλλά δεν αλλάζει για την ίδια κλάση
- $\text{union}(x, y)$: αντικατάσταση (αντιπροσώπων) κλάσεων x και y με (αντιπρόσωπο) κλάση(ς) που προκύπτει από ένωση.
 - Ελέγχουμε αν x και y ανήκουν σε διαφορετική κλάση.
 - Νέος αντιπρόσωπος από τους αντιπροσώπους κλάσεων x, y
 - Πάντα διαμέριση του U σε κλάσεις.
 - $\leq n - 1$ ενώσεις (μετά από $n - 1$, μία μόνο κλάση).
 - Σας θυμίζει κάτι?

Πρόβλημα Union – Find

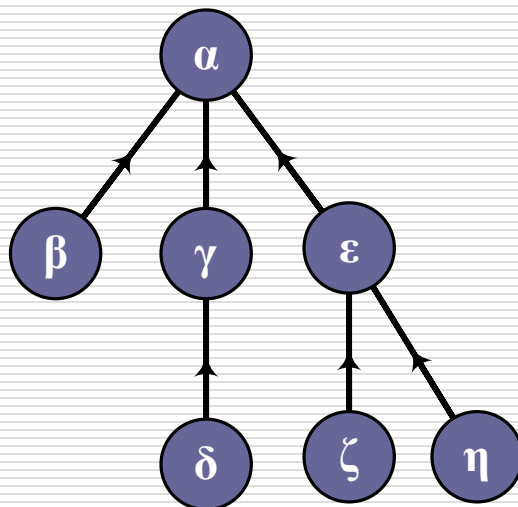
Στόχος:

Δομή δεδομένων που ελαχιστοποιεί συνολικό χρόνο για ακολουθία m ευρέσεων και $n - 1$ ενώσεων.



Αναπαράσταση Δέντρου & Δάσους

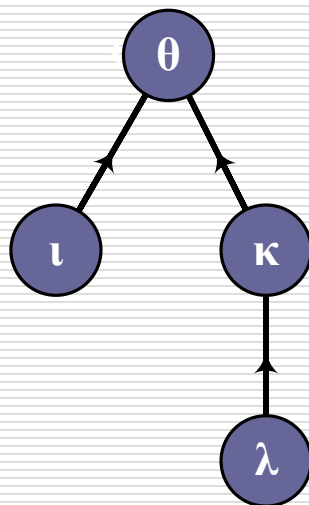
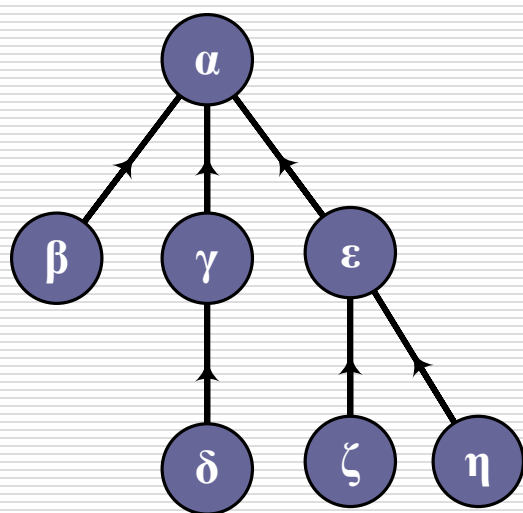
- Πίνακας **γονέων** $\text{parent}[1 \dots n]$ για δέντρο με **ρίζα** και **n κόμβους**:
 - $\text{parent}[i] = j$ αν j γονιός του i στο δέντρο.
 - $\text{parent}[\text{ρίζας}] = \text{ρίζα}$ (ή -1).
 - Κατευθυνόμενο δέντρο.



α	α	α	γ	α	ε	ε
α	β	γ	δ	ε	ζ	η

Αναπαράσταση Δέντρου & Δάσους

- Πίνακας **γονέων** $\text{parent}[1 \dots n]$ για δέντρο με **ρίζα** και **n κόμβους**:
 - $\text{parent}[i] = j$ αν j γονιός του i στο δέντρο.
 - $\text{parent}[\text{ρίζας}] = \text{ρίζα}$ (ή -1).
 - Γενίκευση για **δάσος** όπου κάθε δέντρο έχει ρίζα.



α	α	α	γ	α	ε	ε	θ	θ	θ	κ
α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ

Αναπαράσταση με Δέντρα Μεγέθους

□ Κλάση: **δέντρο** με **ρίζα** το στοιχείο-**αντιπρόσωπο**.

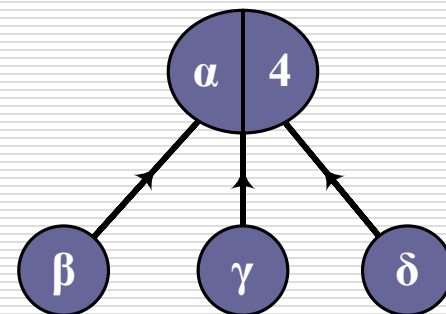
- Όνομα ρίζας.
- Μέγεθος κλάσης.

□ Στοιχείο: **κόμβος** δέντρου με πεδία

- Όνομα στοιχείου.
- Όνομα γονέα: όνομα προηγούμενου στοιχείου στο μονοπάτι προς τη ρίζα-αντιπρόσωπο.

□ Αναπαράσταση με πίνακα γονέων:

- **parent[x]** : γονέας στοιχείου x.
- Ρίζα – στοιχείο αντιπρόσωπος έχει **parent[x] = x** και επιπλέον πεδίο **size**.



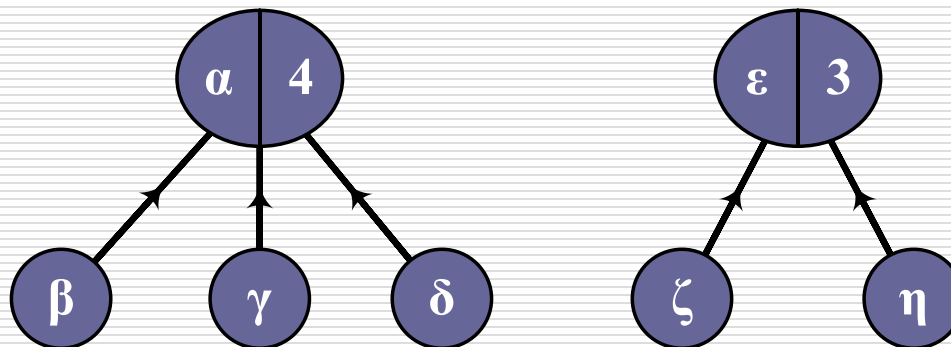
4			
α	α	α	α
α	β	γ	δ

Αναπαράσταση με Δέντρα

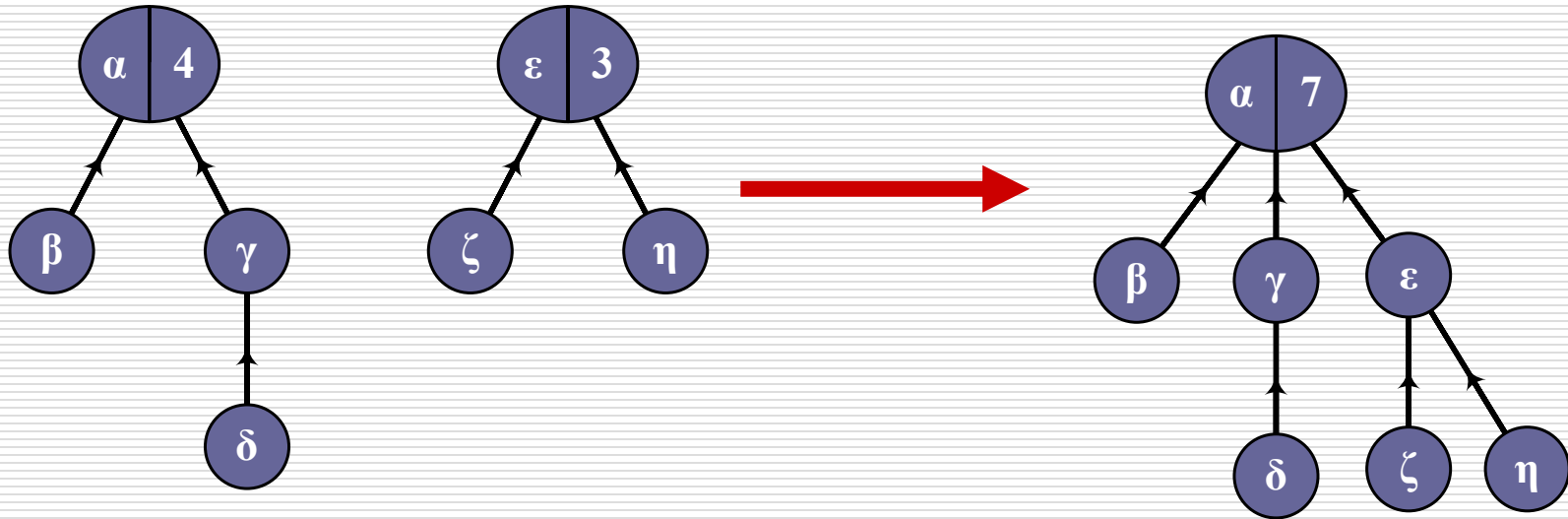
- `find(x)`: ακολουθούμε δείκτες σε γονέα μέχρι τη ρίζα.

```
def find(x):  
    while x != parent[x]:  
        x = parent[x]  
    return x
```

- `union(x, y)`: x και y αντιπρόσωποι διαφορετικών συνόλων
 - Συνένωση δέντρων: ρίζα 1^{ου} συν. γίνεται γονέας ρίζας 2^{ου} συν.
 - Ενημέρωση μεγέθους



Ένωση



```
def union_tree(rx, ry):  
    if rx == ry:  
        return  
    parent[ry] = rx  
    size[rx] += size[ry]
```

Υποθέτουμε ότι ενώνουμε
ρίζες.

Σε διαφορετική περίπτωση
χρήση find

Απόδοση

- Χρόνος χ.π. για m finds και n unions: $O(mn + n)$
 - Union : $O(1)$ χρόνος.
 - Find : $O(\text{ύψος δέντρου})$
 - Χειρότερη περίπτωση: $\text{ύψος} = n - 1$
 - $\text{union}(n-1, n), \text{union}(n-2, n-1), \text{union}(n-3, n-2), \text{union}(n-4, n-3), \dots, \text{union}(2, 3), \text{union}(1, 2).$
 - Δέντρο εκφυλίζεται σε λίστα
- Απλή δομή, εύκολη υλοποίηση, αλλά ακριβό find!
- Πώς δεν θα ψηλώνουν τα δέντρα;
- Δύο λύσεις:
 - Union by Size / Rank
 - Path compression

Βεβαρυμένη Ένωση (Union By Size)

- «Δεύτερο» σύνολο αυτό με τα λιγότερα στοιχεία.

```
def union(x, y):  
    rx, ry = find(x), find(y)  
    if rx == ry:  
        return False  
    if size[rx] < size[ry]:  
        rx, ry = ry, rx  
    parent[ry] = rx  
    size[rx] += size[ry]  
    return True
```

Βεβαρυμένη Ένωση

(Union By Size)

- «Δεύτερο» σύνολο αυτό με τα λιγότερα στοιχεία.
 - Προκύπτει Λογαριθμικό ύψος δέντρου : $O(\log n)$.
- **Θδο:** Δέντρο ύψους h έχει $\geq 2^h$ στοιχεία.
- Απόδειξη με επαγωγή:
 - Ισχύει για $h = 0$ (δέντρο ενός στοιχείου).

Βεβαρυμένη Ένωση

(Union By Size)

- «Δεύτερο» σύνολο αυτό με τα λιγότερα στοιχεία.
 - Προκύπτει Λογαριθμικό ύψος δέντρου : $O(\log n)$.
- Δέντρο ύψους h έχει $\geq 2^h$ στοιχεία.
- Απόδειξη με επαγωγή:
 - Ισχύει για $h = 0$ (δέντρο ενός στοιχείου).
 - Ένωση δέντρων x και y με ύψη h_x, h_y , και στοιχεία $s_x \geq s_y$

Βεβαρυμένη Ένωση

(Union By Size)

- «Δεύτερο» σύνολο αυτό με τα λιγότερα στοιχεία.
 - Προκύπτει Λογαριθμικό ύψος δέντρου : $O(\log n)$.
- Δέντρο ύψους h έχει $\geq 2^h$ στοιχεία.
- Απόδειξη με επαγωγή:
 - Ισχύει για $h = 0$ (δέντρο ενός στοιχείου).
 - Ένωση δέντρων x και y με ύψη h_x, h_y , και στοιχεία $s_x \geq s_y$
 - Επαγωγικά, υποθέτουμε $s \geq 2^h$ (άρα και $s_x \geq 2^{h_x}$ και $s_y \geq 2^{h_y}$)

Βεβαρυμένη Ένωση

(Union By Size)

- «Δεύτερο» σύνολο αυτό με τα λιγότερα στοιχεία.
 - Προκύπτει Λογαριθμικό ύψος δέντρου : $O(\log n)$.
- Δέντρο ύψους h έχει $\geq 2^h$ στοιχεία.
- Απόδειξη με επαγωγή:
 - Ισχύει για $h = 0$ (δέντρο ενός στοιχείου).
 - Ένωση δέντρων x και y με ύψη h_x, h_y , και στοιχεία $s_x \geq s_y$
 - Επαγωγικά, υποθέτουμε $s \geq 2^h$ (άρα και $s_x \geq 2^{h_x}$ και $s_y \geq 2^{h_y}$)
 - Ύψος ένωσης = h_x : #στοιχείων ένωσης $\geq 2^{\text{ύψος}}$ ($s_x + s_y \geq 2^{h_x}$)

Βεβαρυμένη Ένωση

(Union By Size)

- «Δεύτερο» σύνολο αυτό με τα λιγότερα στοιχεία.
 - Προκύπτει Λογαριθμικό ύψος δέντρου : $O(\log n)$.
- Δέντρο ύψους h έχει $n \geq 2^h$ στοιχεία.
- Απόδειξη με επαγωγή:
 - Ισχύει για $h = 0$ (δέντρο ενός στοιχείου).
 - Ένωση δέντρων x και y με ύψη h_x, h_y , και στοιχεία $s_x \geq s_y$
 - Επαγωγικά, υποθέτουμε $s \geq 2^h$ (άρα και $s_x \geq 2^{h_x}$ και $s_y \geq 2^{h_y}$)
 - Ύψος ένωσης = h_x : #στοιχείων ένωσης $\geq 2^{\text{ύψος}}$ ($s_x + s_y \geq 2^{h_x}$)
 - Ύψος ένωσης = $h_y + 1$: #στοιχείων ένωσης $\geq 2^{\text{ύψος}}$
 - $s_x + s_y \geq 2s_y \geq 2^{h_y+1}$
 - Άρα: $n \geq 2^h \Rightarrow h \leq \log n$
- Χρόνος για m finds και n unions: $O(m \cdot \log n + n)$
 - Απλή υλοποίηση και αποδεκτή απόδοση.

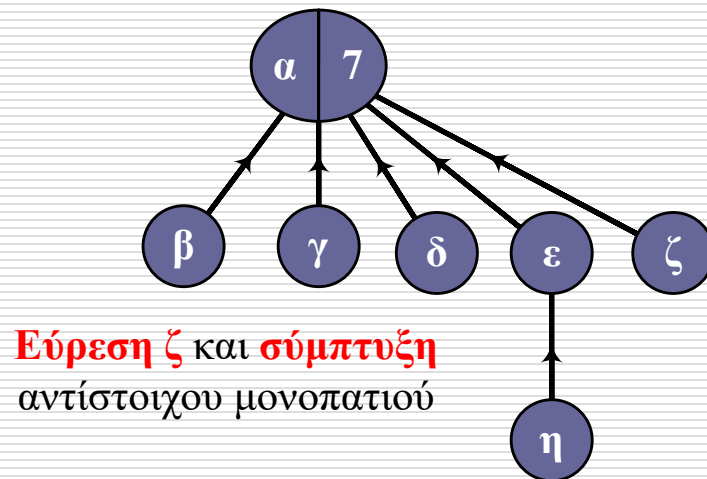
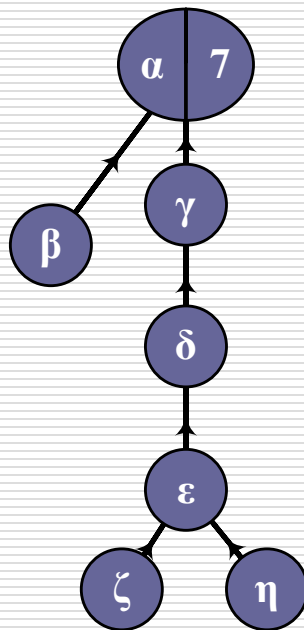
Σύμπτυξη Μονοπατιών

```
def find_with_path_compression(x):  
    if x != parent[x]:  
        parent[x] = find_with_path_compression(parent[x])  
    return parent[x]
```

Ιδέα:

Κάθε φορά που κάνουμε **find**, τακτοποιούμε όλους τους κόμβους που συναντάμε.

**Βελτίωση
μελλοντικών find**



Εύρεση ζ και σύμπτυξη
αντίστοιχου μονοπατιού

Απόδοση

- Δέντρα, βεβαρυμένη ένωση, και σύμπτυξη μονοπατιών.
 - Τα unions επηρεάζουν τις ρίζες
 - Τα finds επηρεάζουν το εσωτερικό τμήμα του δέντρου
- Απλή δομή, εύκολη υλοποίηση, και ουσιαστικά γραμμικός χρόνος!
 - Χρόνος για $m \geq n$ finds και n unions: $O(m \cdot \alpha(n, m))$ (Tarjan 1984)
 - $\alpha(n, m)$: αντίστροφη συνάρτηση Ackermann.
 - Μεγαλώνει εξαιρετικά αργά!
 - Στην πράξη, μπορεί να θεωρηθεί σταθερά.
 - Αρκετά σύνθετη ανάλυση στο CLRS (3rd edition -21.4)
- Θα δείξουμε κάτι λιγότερο tight.
- Χρόνος για $m \geq n$ finds και n unions: $O(m \cdot \log^* n)$ (Hopcroft – Ullman 1974)

Υπενθύμιση

□ Συνάρτηση Ackermann

$$A(0, n) = n + 1$$

$$A(m + 1, 0) = A(m, 1)$$

$$A(m + 1, n + 1) = A(m, A(m + 1, n))$$

$$■ A(1, 2) \rightarrow 4$$

$$■ A(4, 1) \rightarrow 65533$$

$$■ A(4, 3) \rightarrow 2^{2^{65536}} - 3$$

□ Αντίστροφη Ackermann

$$\alpha(m, n) = \min\{i \geq 1 : A(i, \lfloor m/n \rfloor) \geq \log_2 n\}.$$

$\alpha(n, n)$ μικρότερο από 5 για
κάθε n που μπορεί να
εμφανιστεί στην πράξη

□ Συνάρτηση $\log^* n$

■ Πόσες φορές πρέπει να
εφαρμοστεί επαναληπτικά ο
λογάριθμος μέχρι να δω ένα.

$$□ \log^* 2^{2^{16}} = 5$$

$$1. \log 2^{2^{16}} = 2^{16}$$

$$2. \log 2^{16} = 16$$

$$3. \log 16 = 4$$

$$4. \log 4 = 2$$

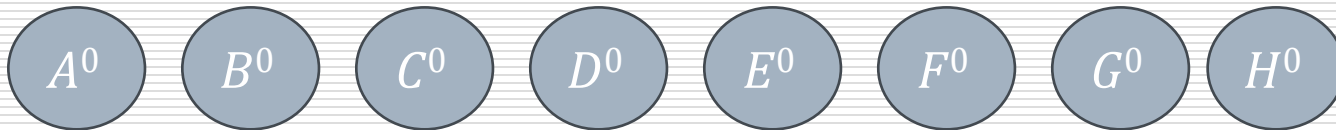
$$5. \log 2 = 1$$

Union By Rank

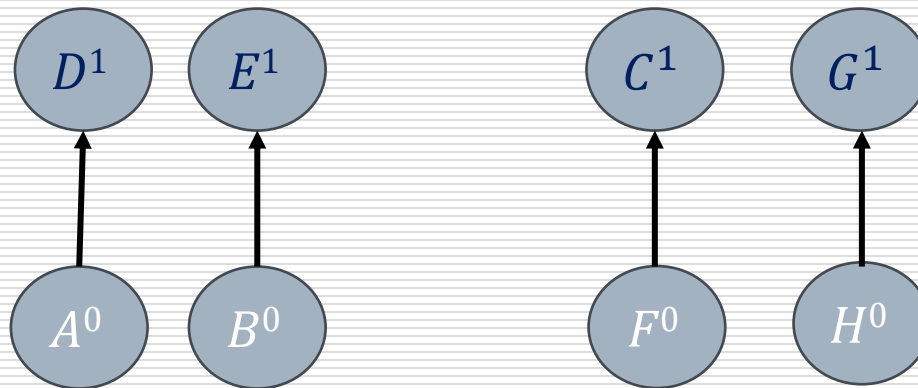
- Κάθε **κόμβος** έχει rank (αρχικά όλοι 0).
- Σε union:
 - Η ρίζα με **μικρότερο** rank μπαίνει κάτω από την άλλη
 - Σε ισοπαλία, η νέα ρίζα παίρνει **rank + 1**.
- Η σύμπτυξη μονοπατιού **δεν** αλλάζει ranks.

```
def union(x, y):  
    rx, ry = find(x), find(y)  
    if rx == ry:  
        return False  
    if rank[rx] < rank[ry]:  
        rx, ry = ry, rx  
    parent[ry] = rx  
    if rank[rx] == rank[ry]:  
        rank[rx] += 1  
    return True
```

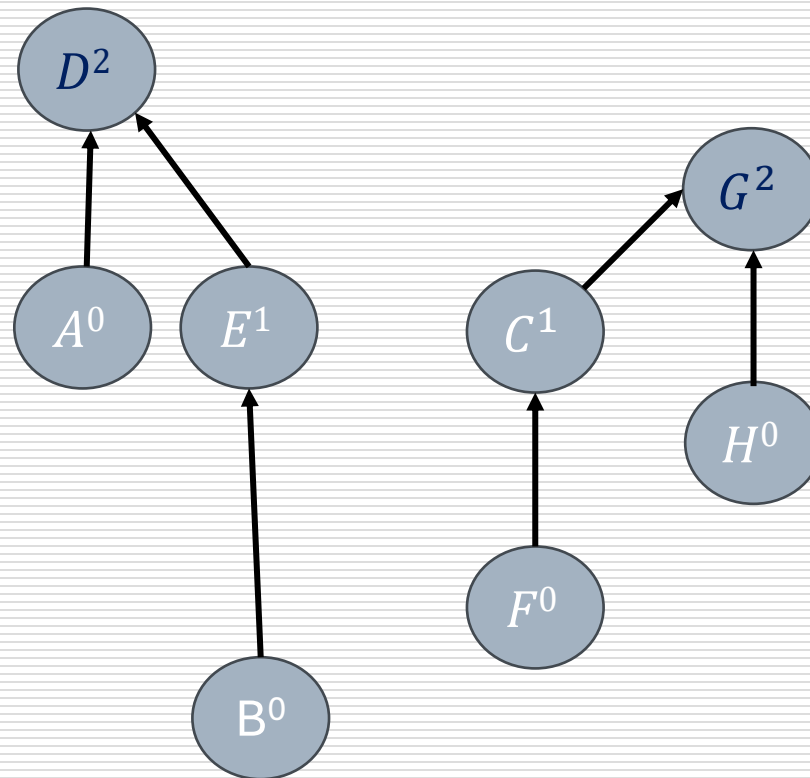
Union By Rank



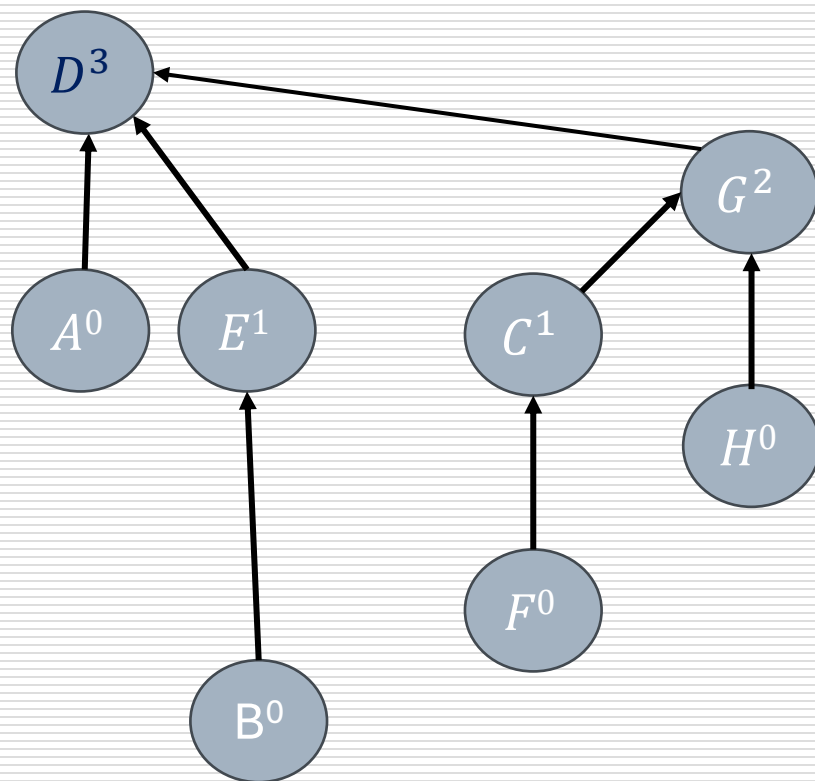
Union By Rank



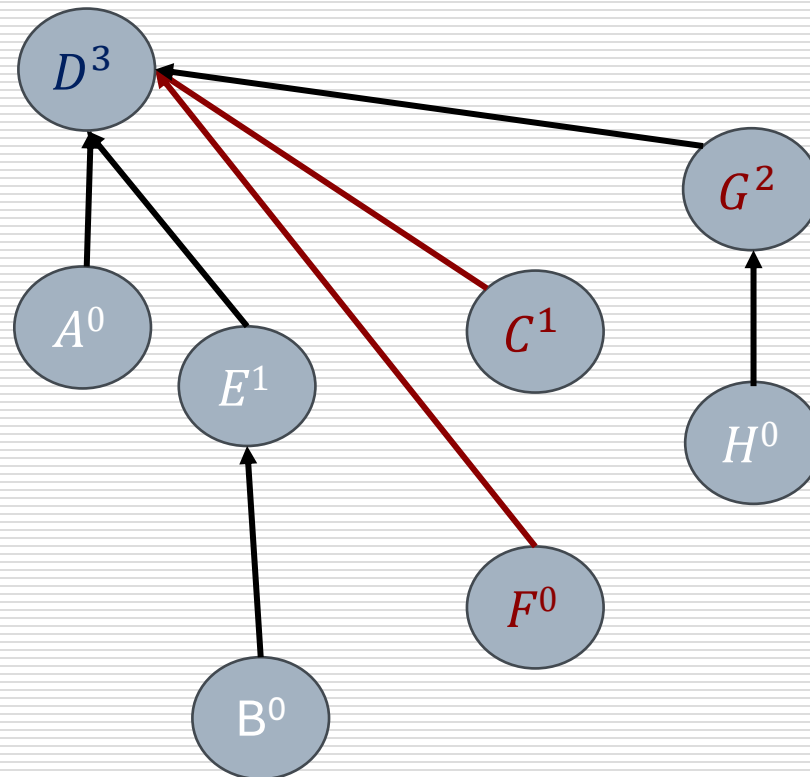
Union By Rank



Union By Rank



Find(F) με path compression



□ Union By Rank - Χρήσιμες ιδιότητες:

- Τα ranks αυξάνουν **γνησίως** σε κάθε μονοπάτι προς τη ρίζα.
 - $\forall x \text{ non-root: } \mathit{rank}(x) < \mathit{rank}(\mathit{parent}(x))$
- Κάθε ρίζα με rank r έχει $\geq 2^r$ στοιχεία στο δέντρο της.
 - Κάθε κόμβος με rank r έχει $\geq 2^r$ στοιχεία στο υποδέντρο του.
 - (χωρίς συμπτυξη)
- Το πολύ $n/2^r$ κόμβοι αποκτούν rank r .
 - Άρα $\forall \text{rank } r: r \leq \log n$
- Η σύμπτυξη δεν επηρεάζει το rank.
 - Η διαφορά του rank από το ύψος.
 - Rank: **άνω όριο ύψους**

Hopcroft – Ullman

- Μέτρηση βελτίωσης λόγω path compression
 - $rank(parent[x]) - rank(x)$
 - Όσο μεγαλύτερο, τόσο μεγαλύτερη είναι η βελτίωση
- Ομαδοποιούμε τα ranks ως εξής:
 - $\{0,1\}, \{2\}, \{3, 2^2\}, \{5,6, \dots, 2^4\}, \{17, \dots, 2^{16}\}, \{2^{16} + 1, \dots, 2^{2^{16}}\}$
 - Γενική μορφή: $\{k + 1, \dots, 2^k\}$
 - το k της επόμενης ομάδας είναι το 2^k της προηγούμενης
 - Υπάρχουν $\log^* n$ τέτοιες ομάδες
- Αποτελεσματικό path compression:
 - Όταν $rank(x), rank(parent(x))$ σε διαφορετικά μπλοκ

Hopcroft – Ullman

- **Τρεις περιπτώσεις finds**
- Π1. Σύνδεση στη ρίζα
 - m λειτουργίες $\text{Find}(x)$ σε $O(m)$
- Π2. Ranks $x, \text{parent}(x)$ σε διαφορετικά μπλοκ
 - m λειτουργίες $\text{Find}(x)$ σε $O(m \cdot \log^* n)$
- Π3. Ranks $x, \text{parent}(x)$ σε ίδιο μπλοκ
 - $\text{rank}(x), \text{rank}(\text{parent}(x))$ στην ομάδα $\{k + 1, \dots, 2^k\}$

Hopcroft - Ullman

- $rank(x), rank(parent(x))$ στην ομάδα $\{k + 1, \dots, 2^k\}$
- Κάθε path compression σε κόμβο x του μπλοκ αλλάζει τον πατέρα του (νέα ρίζα) σε κόμβο με μεγαλύτερο rank.
 - Ασχολούμαστε με την περίπτωση που παραμένει το rank του πατέρα του στο ίδιο μπλοκ (αλλιώς μεταφερόμαστε στην Π2).
 - Αυτό μπορεί να συμβεί το πολύ $2^k - k < 2^k$ φορές
 - Για $\frac{n}{2^{k+1}} + \frac{n}{2^{k+2}} + \dots + \frac{n}{2^{2^k}} \leq n/2^k$ κόμβους.
 - Για το πολύ $O(\log^* n)$ ομάδες
 - Συνολικά $O(n \cdot \log^* n)$
- Συνολικά: **Για $m \geq n$ finds: $O(m \cdot \log^* n)$**