Παρουσίαση για χρήση με το σύγγραμμα, Αλγόριθμοι Σχεδίαση και Εφαρμογές, των M. T. Goodrich and R. Tamassia, Wiley, 2015 (στα ελληνικά από εκδόσεις Μ. Γκιούρδας)

# Δομές Ένωσης-Εύρεσης

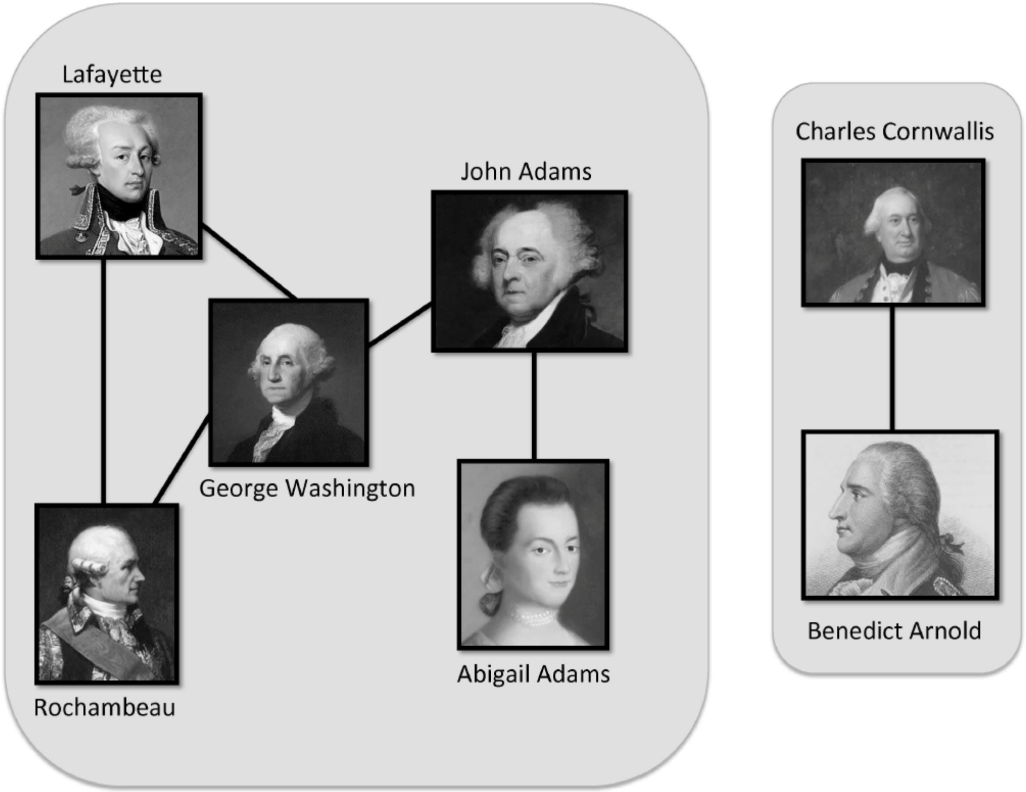
Merging galaxies, NGC 2207 and IC 2163. Combined image from NASA's Spitzer Space Telescope and Hubble Space Telescope. 2006. U.S. government image. NASA/JPL-Caltech/STSci/Vassar.

# Εφαρμογή: Συνδεδεμένες συνιστώσες σε κοινωνικό δίκτυο

- Η έρευνα της κοινωνικής δικτύωσης μελετά τον τρόπο που οι σχέσεις μεταξύ διαφόρων ανθρώπων μπορούν να επηρεάσουν τη συμπεριφορά.
- Δεδομένου ενός συνόλου, S, n ανθρώπων, μπορούμε να ορίσουμε εάν κοινωνικό δίκτυο για το S δημιουργώντας ένα σύνολο, E, ακμών ή δεσμών μεταξύ ζευγών ανθρώπων που έχουν συγκεκριμένο είδος σχέσης. Για παράδειγμα, σε ένα δίκτυο φίλων όπως το Facebook, οι δεσμοί θα ορίζονταν από ζεύγη φίλων.
- Μία **συνδεδεμένη συνιστώσα** σε ένα δίκτυο φίλων είναι ένα υποσύνολο, T, από ανθρώπων από το S που ικανοποιεί τις παρακάτω ιδιότητες:
  - Κάθε άτομο στο T σχετίζεται μέσω φιλίας, δηλαδή για οποιαδήποτε άτομα x και y στο T, το x ή το y ή υπάρχει μια αλυσίδα φιλίας, όπως μέσω ενός φίλου ενός φίλου ενός φίλου, που συνδέει τα x και y.
  - Κανείς στο T δεν είναι φίλος με οποιονδήποτε εκτός του T.

# Example

- 2 Συνδεδεμένες συνιστώσες σε εάν δίκτυο φίλων κάποιων σημαντικών μορφών της Αμερικάνικης Επανάστασης.




Όλες οι εικόνες είναι δημοσίας χρήσης.

# Ένωση-Εύρεση Λειτουργίες

- Μία **δομή ένωσης-εύρεσης** είναι μία δομή δεδομένων που υποστηρίζει μια συλλογή διαφορετικών συνόλων. Οι μέθοδοι της είναι οι εξής:
- makeSet(e): Δημιουργία ενός μοναδιαίου συνόλου που περιέχει το στοιχείο e και επιστροφή της θέσης που έχει αποθηκευτεί το e στο σύνολο
- union(A,B): Επιστρέφει το A U B, το οποίο θα ονομαστεί "A" ή "B"
- find(e): Επιστρέφει το όνομα του συνόλου που περιέχει το e

# Αλγόριθμος συνδεδεμένων συνιστωσών

- Η έξοδος του αλγορίθμου είναι ένας προσδιορισμός, για κάθε x στο S, της συνδεδεμένης συνιστώσας που περιέχει το x.

**Algorithm** UFConnectedComponents($S$, $E$):

***Input:*** A set, $S$, of $n$ people and a set, $E$, of $m$ pairs of people from $S$ defining pairwise relationships

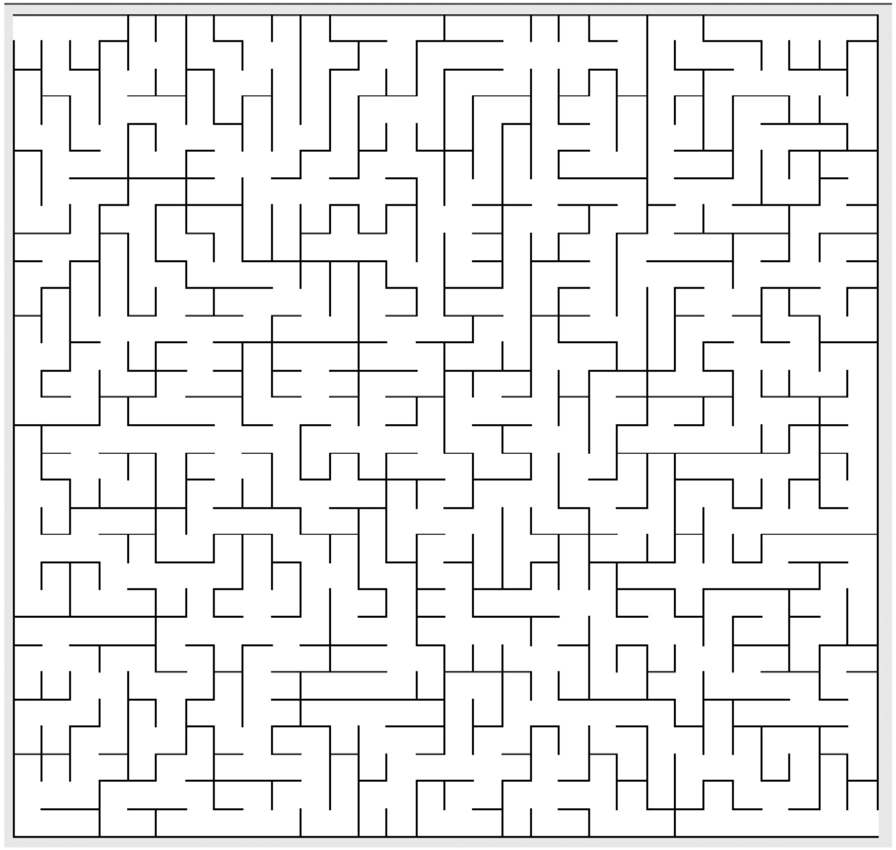
***Output:*** An identification, for each $x$ in $S$, of the connected component containing $x$

**for** each $x$ in $S$ **do**
    makeSet($x$)
**for** each $(x, y)$ in $E$ **do**
    **if** find($x$) $\neq$ find($y$) **then**
        union(find($x$), find($y$))
**for** each $x$ in $S$ **do**
    Output "Person $x$ belongs to connected component" find($x$)

- Ο χρόνος του αλγορίθμου είναι O(t(n,n+m)), όπου t(j,k) είναι ο χρόνος για k λειτουργίες εύρεσης-ένωσης που ξεκινούν από j μοναδιαία σύνολα.

# Ακόμη μία εφαρμογή: Κατασκευή Λαβύρινθου και Θεωρία Διήθησης

- Πρόβλημα: Κατασκευή ενός καλού λαβύρινθου.



# Ένας τυχαιοποιημένος αλγόριθμος για κατασκευή λαβύρινθων

**Algorithm** MazeGenerator$(G, E)$:

***Input:*** A grid, $G$, consisting of $n$ cells and a set, $E$, of $m$ "walls," each of which divides two cells, $x$ and $y$, such that the walls in $E$ initially separate and isolate all the cells in $G$

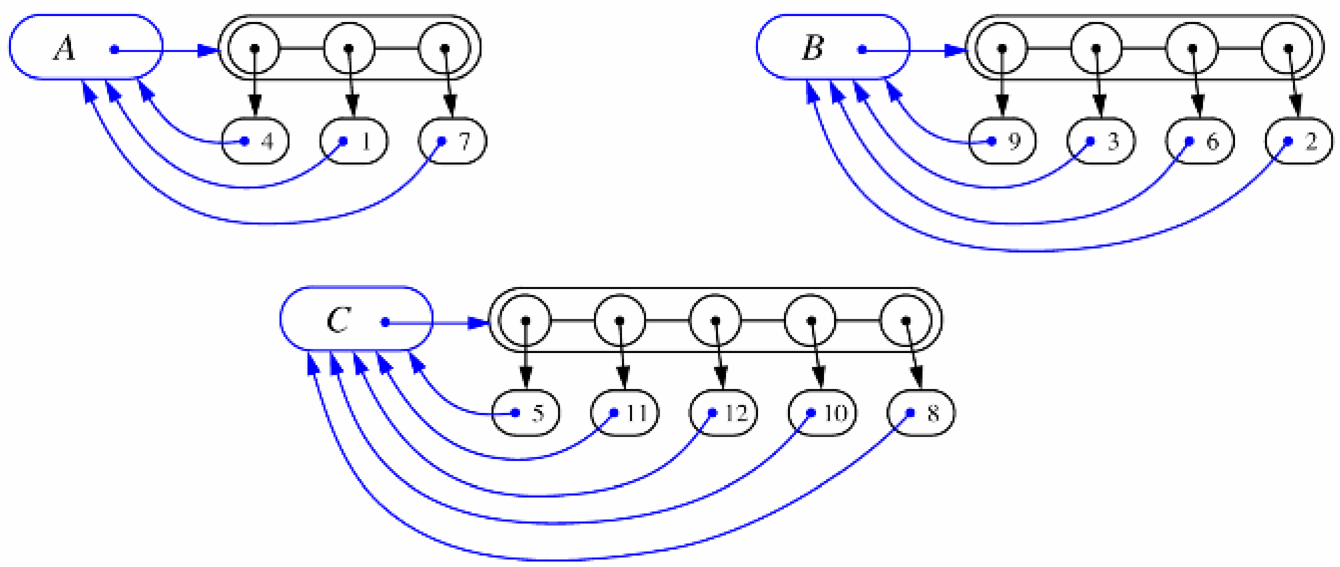
***Output:*** A subset, $R$ of $E$, such that removing the edges in $R$ from $E$ creates a maze defined on $G$ by the remaining walls

**while** $R$ has fewer than $n - 1$ edges **do**
  Choose an edge, $(x, y)$, in $E$ uniformly at random from among those previously unchosen
  **if** find$(x) \neq$ find$(y)$ **then**
    union(find$(x)$, find$(y)$)
    Add the edge $(x, y)$ to $R$
**return** $R$

- Σχετίζεται με την επιστήμη της **θεωρία διήθησης**, η οποία μελετά πώς τα ρευστά διαπερνούν πορώδη υλικά.
  - Για παράδειγμα, ένα πορώδες υλικό μπορεί να μοντελοποιηθεί ως ένα πλέγμα κελίων τριών διαστάσεων **n x n x n**. Τα εμπόδια που χωρίζουν γειτονικά ζεύγη κελιών θα μπορούσαν να αφαιρεθούν εικονικά, με κάποια πιθανότητα **p** και να παραμείνουν με πιθανότητα 1 **– p**. Η προσομοίωση ενός τέτοιου συστήματος είναι μία ακόμα εφαρμογή των δομών ένωσης-εύρεσης.

# Υλοποίηση που βασίζεται σε λίστα

- Κάθε σύνολο αποθηκεύεται σε μία ακολουθία που αναπαρίσταται με μία συνδεδεμένη λίστα.
- Κάθε κόμβος πρέπει να αποθηκεύει ένα αντικείμενο που περιέχει το στοιχείο και μία αναφορά στο όνομα του συνόλου.

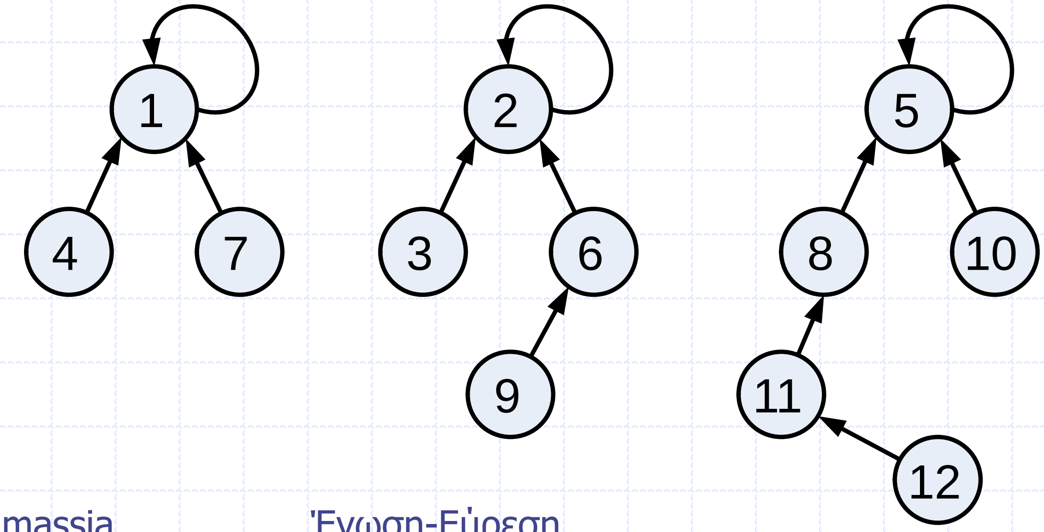


# Ανάλυση υλοποίησης που βασίζεται σε λίστα

- Όταν εκτελείτε union, πάντα θα μετακινούνται τα στοιχεία από το μικρότερο σύνολο στο μεγαλύτερο
  - Κάθε φορά που ένα στοιχείο μετακινείται πηγαίνει σε ένα σύνολο τουλάχιστον διπλάσιου μεγέθους από το παλιό
  - Έτσι, ένα στοιχείο μπορεί να μετακινηθεί το πολύ O(log n) φορές
- Ο συνολικός χρόνος για n union και m αναζητήσεις είναι O(n log n + m).

# Υλοποίηση που βασίζεται σε δέντρα

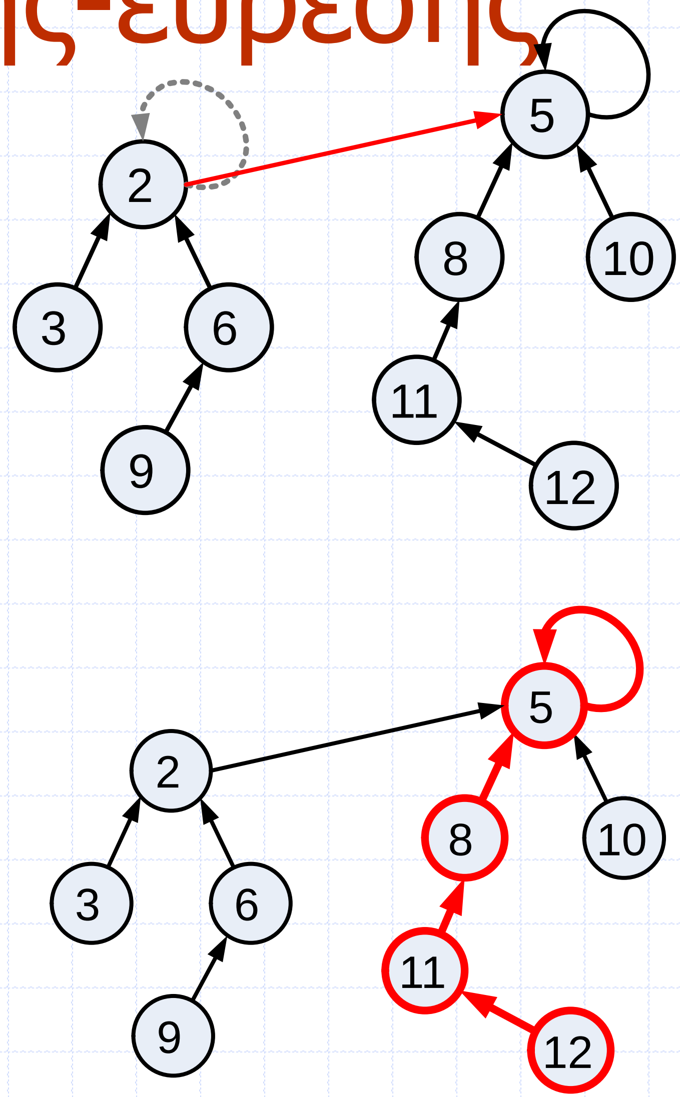
- Κάθε στοιχείο αποθηκεύεται σε ένα κόμβο, που περιέχει ένα δείκτη προς το όνομα του συνόλου
- Ένας κόμβος v που ο δείκτης συνόλου δείχνει προς το v είναι επίσης το όνομα του συνόλου
- Κάθε σύνολο είναι ένα δέντρο η ρίζα του οποίου είναι ένας κόμβος με αυτό-αναφορά στον δείκτη συνόλου
- Για παράδειγμα: Τα σύνολα “1”, “2”, και “5”:



# Λειτουργίες ένωσης-εύρεσης

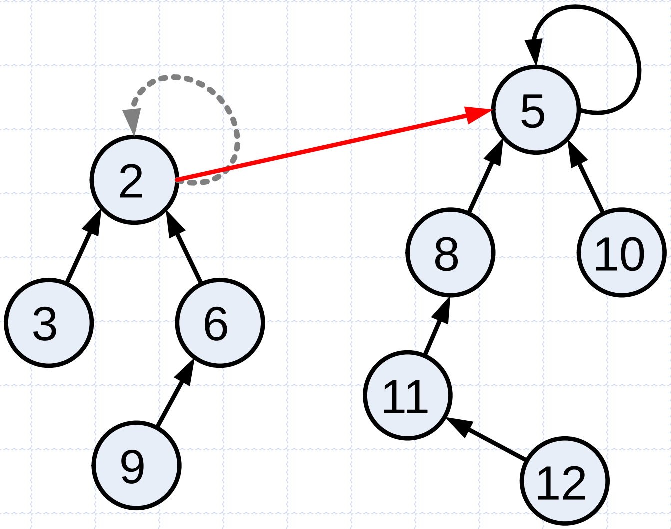
- Για να γίνει union, απλώς κάνουμε την ρίζα του ενός δέντρου να δείχνει την ρίζα του άλλου

- Για να γίνει find, από τον αρχικό κόμβο ακολουθούμε τους δείκτες μέχρι να βρούμε κόμβο που ο δείκτης είναι προς τον εαυτό του



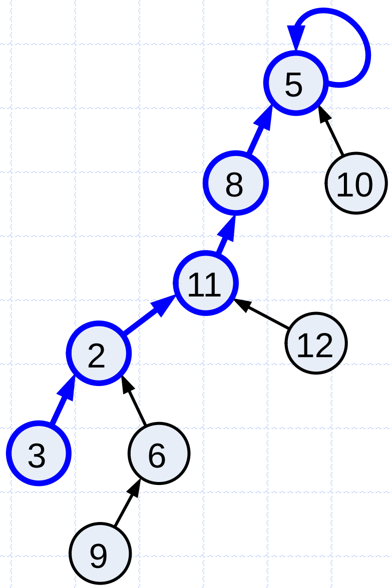
# Ένωση-Εύρεση Ευριστικό 1

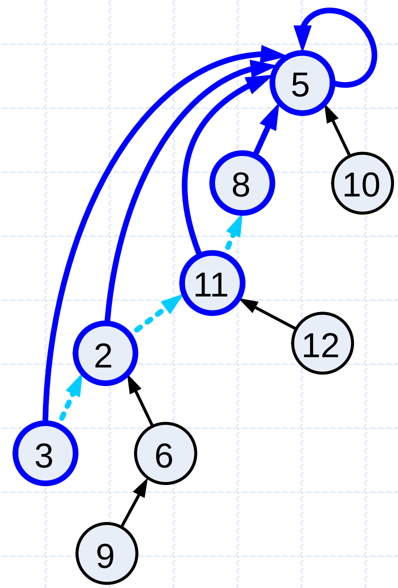
- **Ένωση κατά μέγεθος**:
  - Κατά την εκτέλεση union, κάνουμε την ρίζα του μικρότερου δέντρου να δείχνει στο μεγαλύτερο.
- Χρειάζεται χρόνος O(n log n) για την εκτέλεση n λειτουργιών ένωσης-εύρεσης:
  - Κάθε φορά που ακολουθούμε κάποιον δείκτη πηγαίνουμε σε ένα υπό-δέντρο τουλάχιστον διπλάσιου μεγέθους από το προηγούμενο υπό-δέντρο.
  - Έτσι ακολουθούμε το πολύ O(log n) δείκτες για κάθε αναζήτηση.



# Ένωση-Εύρεση Ευριστικό 2

- **Συμπίεση διαδρομής**:
  - Μετά την εκτέλεση κάποιας αναζήτησης, συμπίεση όλων των δεικτών στο μονοπάτι ώστε να δείχνουν στην ρίζα




- Θέλει «σχεδόν γραμμικό» χρόνο για n λειτουργίες ένωσης-εύρεσης.

# Συνάρτηση Ackermann

The version of the Ackermann function we use is based on an indexed function, $A_i$, which is defined as follows, for integers $x \geq 0$ and $i > 0$:

$$\begin{aligned} A_0(x) &= x + 1 \\ A_{i+1}(x) &= A_i^{(x)}(x), \end{aligned}$$

where $f^{(k)}$ denotes the $k$-fold composition of the function $f$ with itself. That is,

$$\begin{aligned} f^{(0)}(x) &= x \\ f^{(k)}(x) &= f(f^{(k-1)}(x)). \end{aligned}$$

So, in other words, $A_{i+1}(x)$ involves making $x$ applications of the $A_i$ function on itself, starting with $x$. This indexed function actually defines a progression of functions, with each function growing much faster than the previous one:

- $A_0(x) = x + 1$, which is the increment-by-one function
- $A_1(x) = 2x$, which is the multiply-by-two function
- $A_2(x) = x2^x \geq 2^x$, which is the power-of-two function
- $A_3(x) \geq 2^{2^{\cdot^{\cdot^{\cdot^{2}}}}}$ (with $x$ number of 2's), which is the tower-of-twos function
- $A_4(x)$ is greater than or equal to the tower-of-tower-of-twos function
- and so on.

# Συνάρτηση Ackermann

Ορίζουμε την **συνάρτηση Ackermann function** ως

$$A(x) = A_x(2),$$

Η οποία είναι συνάρτηση που αυξάνεται απίστευτα γρήγορα.

- Για να καταλάβετε καλύτερα τα πράγματα, σημειώστε ότι $A(3) = 2048$ και το $A(4)$ είναι μεγαλύτερο από ή ίσο με μία επαναλαμβανόμενη ύψωση στη δύναμη 2048 δύο, δηλαδή ένας αριθμός πολύ μεγαλύτερος από τον αριθμό των υποατομικών σωματιδίων στο συμπάν.

Ομοίως το αντίστροφο,

$$\alpha(n) = \min\{x: A(x) \geq n\},$$

Είναι μια συνάρτηση που αυξάνεται απίστευτα αργά. Ακόμα κι αν το $\alpha(n)$ πράγματι αυξάνεται όσο το n τείνει στο άπειρο, για να είμαστε πρακτικοί, θεωρούμε ότι $\alpha(n) \leq 4$.

# Γρήγορη επιμερισμένη ανάλυση

- Για κάθε κόμβο v στο ενωμένο δέντρο που είναι ρίζα
  - ορίζουμε n(v) να είναι το μέγεθος του υπό-δένδρου με ρίζα στο v (συμπεριλαμβανομένου του v)
  - Ανιχνεύουμε ένα σύνολο με την ρίζα στο σχετικό δέντρο.
- Ενημερώνουμε το πεδίο του μεγέθους του v κάθε φορά που ένα σύνολο ενώνεται με το v. Έτσι, αν το v δεν είναι ρίζα, τότε n(v) είναι μεγαλύτερο όσο μεγάλο μπορεί να είναι το υπό-δένδρο με ρίζα το v, που συμβαίνει ακριβώς πριν ενώσουμε το v σε κάποιον άλλον κόμβο που το μέγεθος του είναι τουλάχιστον όσο του v.
- Για κάθε κόμβο v, ορίζουμε τον βαθμό του v, τον οποίο συμβολίζουμε ως r(v), ως εξής $r(v) = [\log n(v)] + 2$:
- Συνεπώς, $n(v) \geq 2^{r(v)-2}$.
- Επίσης, επειδή υπάρχουν το πολύ n κόμβοι στο δέντρο του v, $r(v) \leq [\log n]+2$, για κάθε κόμβο v.

# Επιμερισμένη ανάλυση (2)

- Για κάθε κόμβο v με γονέα τον w:
  - r (v ) < r (w )

**Proof:** We make $v$ point to $w$ only if the size of $w$ before the union is at least as large as the size of $v$. Let $n(w)$ denote the size of $w$ before the union and let $n'(w)$ denote the size of $w$ after the union. Thus, after the union we get

$$
\begin{aligned}
r(v) &= \lfloor \log n(v) \rfloor + 2 \\
&< \lfloor \log n(v) + 1 \rfloor + 2 \\
&= \lfloor \log 2n(v) \rfloor + 2 \\
&\leq \lfloor \log(n(v) + n(w)) \rfloor + 2 \\
&= \lfloor \log n'(w) \rfloor + 2 \\
&\leq r(w).
\end{aligned}
$$

■

- Έτσι οι βαθμοί αυξάνονται αυστηρά όσο ακολουθούμε τους δείκτες προς τους γονείς.

# Επιμερισμένη ανάλυση (3)

- Λήμμα: Υπάρχουν το πολύ $n/ 2^{s-2}$ κόμβοι βαθμού s.
- Απόδειξη:
  - Επειδή $r(v) < r(w)$, για κάθε κόμβο v με γονέα w, οι βαθμοί αυξάνονται αυστηρά όσο ακολουθούμε τους δείκτες γονέων ανοδικά σε οποιοδήποτε δέντρο.
  - Συνεπώς, αν $r(v) = r(w)$ για δύο κόμβους v και w, τότε οι κόμβοι που μετράμε στο n(v) πρέπει να είναι διαφορετικοί απ' τους κόμβους που μετράμε στο n(w).
  - Εάν ένας κόμβος είναι v είναι βαθμού s, τότε $n(v) \geq 2^{s-2}$.
  - Επομένως, επειδή υπάρχουν το πολύ n κόμβοι συνολικά, μπορεί να υπάρχουν το πολύ $n/2^{s-2}$ που είναι βαθμού s.

# Επιμερισμένη ανάλυση (4)

For the sake of our amortized analysis, let us define a ***labeling function***, $L(v)$, for each node $v$, which changes over the course of the execution of the operations in $\sigma$. In particular, at each step $t$ in the sequence $\sigma$, define $L(v)$ as follows:

$$L(v) \ = \ \text{the largest } i \text{ for which } r(p(v)) \geq A_i(r(v)).$$

Note that if $v$ has a parent, then $L(v)$ is well-defined and is at least 0, since

$$r(p(v)) \ \geq \ r(v) + 1 \ = \ A_0(r(v)),$$

because ranks are strictly increasing as we go up the tree $U$. Also, for $n \geq 5$, the maximum value for $L(v)$ is $\alpha(n) - 1$, since, if $L(v) = i$, then

$$\begin{aligned} n &> \lfloor \log n \rfloor + 2 \\ &\geq r(p(v)) \\ &\geq A_i(r(v)) \\ &\geq A_i(2). \end{aligned}$$

Or, put another way,

$$L(v) \ < \ \alpha(n),$$

for all $v$ and $t$.

# Επιμερισμένη ανάλυση (5)

- Έστω ότι v είναι κόμβος κατά μήκος μίας διαδρομής P. Χρησιμοποιούμε δύο κανόνες για την χρέωση ενός κυβερνοδολαρίου όταν ακολουθούμε το δείκτη γονέα για το v:
  - Αν το v έχει έναν πρόγονο w στο P έτσι ώστε L(v) = L(w), σ' αυτό το χρονικό σημείο, τότε χρεώνουμε ένα κυβερνοδολάριο στο ίδιο το v.
  - Αν το v δεν έχει τέτοιον πρόγονο, τότε χρεώνουμε ένα κυβερνοδολάριο σ' αυτήν την πράξη find.
- Καθώς υπάρχουν $\alpha(n)$ ομάδες βαθμού, αυτός ο κανόνας εγγυάται ότι κάθε λειτουργία find θα χρεωθεί το πολύ $\alpha(n)$ κυβερνοδολάρια.

# Επιμερισμένη ανάλυση (6)

- Αφού χρεώσουμε έναν κόμβο v τότε το v θα αποκτήσει έναν νέο γονέα, που είναι ένας κόμβος ψηλότερα στο δέντρο του v.
- Ο βαθμός του νέου γονέα του v θα είναι μεγαλύτερο από τον παλιό γονέα του v τον w.
- Κάθε κόμβος v μπορεί να χρεωθεί το πολύ r(v) κυβερνοδολάρια πριν το L(v) αυξηθεί τουλάχιστον κατά 1.
- Επειδή το L(v) μπορεί να αυξηθεί το πολύ $\alpha$(n)-1 φορές, αυτό σημαίνει ότι μπορεί να χρεωθούν το πολύ r(n)$\alpha$(n) κυβερνοδολάρια.

# Επιμερισμένη ανάλυση (7)

- Αν συνδυάσουμε αυτό το γεγονός με το όριο του αριθμού των κόμβων κάθε βαθμού, χρεώνονται το πολύ

$$s\,\alpha(n)\frac{n}{2^{s-2}} = n\,\alpha(n)\frac{s}{2^{s-2}}$$

  κυβερνοδολάρια σε όλες τις κορυφές του βαθμού s.
- Αν αθροίσουμε όλους τους πιθανούς βαθμούς, ο συνολικός αριθμός κυβερνοδολαρίων που χρεώνονται σε όλους τους κόμβους είναι το πολύ

$$\begin{aligned}\sum_{s=0}^{\lfloor\log n\rfloor+2} n\,\alpha(n)\frac{s}{2^{s-2}} &\le \sum_{s=0}^{\infty} n\,\alpha(n)\frac{s}{2^{s-2}}\\ &= n\,\alpha(n)\sum_{s=0}^{\infty}\frac{s}{2^{s-2}}\\ &\le 8n\,\alpha(n),\end{aligned}$$

- Οι συνολικές χρεώσεις που γίνονται σε όλους τους κόμβου είναι O((n+m)$\alpha$(n)).