Διακριτά Μαθηματικά

Ενότητα 3: Δένδρα

Αν. Καθηγητής Κ. Στεργίου
e-mail: kstergiou@uowm.gr

Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών

BY NC ND
Ευρωπαϊκή Ένωση
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης
ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.
- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.

Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «**Ανοικτά Ψηφιακά Μαθήματα στο Πανεπιστήμιο Δυτικής Μακεδονίας**» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Περιεχόμενα

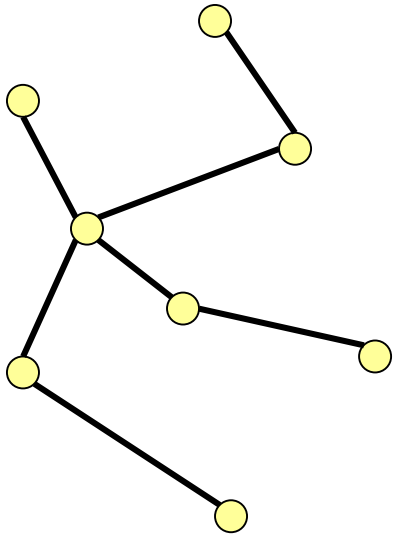
- Εισαγωγή στα Δέντρα και βασικός ορισμός δέντρου.
  - Εναλλακτικοί Ορισμοί.
- Δέντρα με ρίζα.
- Δυαδικά και m-αδικά δέντρα
  - Βασικές έννοιες και ιδιότητες.
- Διάσχιση δέντρων.
  - Post-order, pre-order, in-order, κατά πλάτος.
- Δυαδικά δέντρα αναζήτησης.
  - Βασικές λειτουργίες (αναζήτηση, εισαγωγή, διαγραφή).
- Ελάχιστα επικαλύπτοντα δέντρα.
  - Οι αλγόριθμοι των Prim και Kruskal.

Στόχοι

- Εισαγωγή στη μελέτη δέντρων ως σημαντικότατης ειδικής κατηγορίας γραφημάτων.
- Εξοικείωση με την ορολογία των δέντρων και με τις κατηγοριοποιήσεις τους.
- Μελέτη ορισμένων βασικών προβλημάτων σε δέντρα, όπως η διάσχιση ενός δέντρου, η αναζήτηση/εισαγωγή/διαγραφή κόμβων σε δυαδικά δέντρα αναζήτησης, και η εύρεση ελάχιστου επικαλύπτοντος δέντρου σε εμβαρημένο γράφημα (με τους αλγορίθμους των Prim και Kruskal).

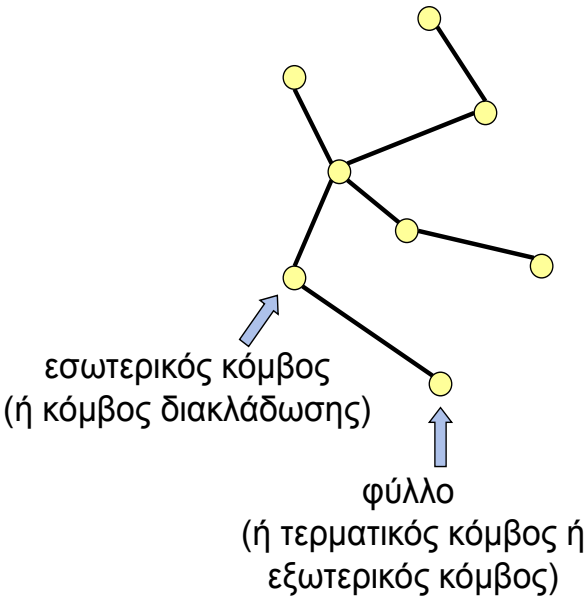
Δένδρα (1/14)

- **Δένδρο**: Συνεκτικό (μη κατευθυνόμενο) άκυκλο γράφημα.
- Τα δένδρα παίζουν κεντρικό ρόλο στην επιστήμη των υπολογιστών:
  - Ανάλυση αλγορίθμων (π.χ. δένδρα αναδρομής).
  - Δομές δεδομένων (π.χ. δένδρα αναζήτησης).
- Κατηγορίες (αύξουσα σειρά γενικότητας):
  - Δυαδικά και m-αδικά δένδρα
  - Διατεταγμένα δένδρα.
  - Δένδρα με ρίζα.
  - Ελεύθερα δένδρα.



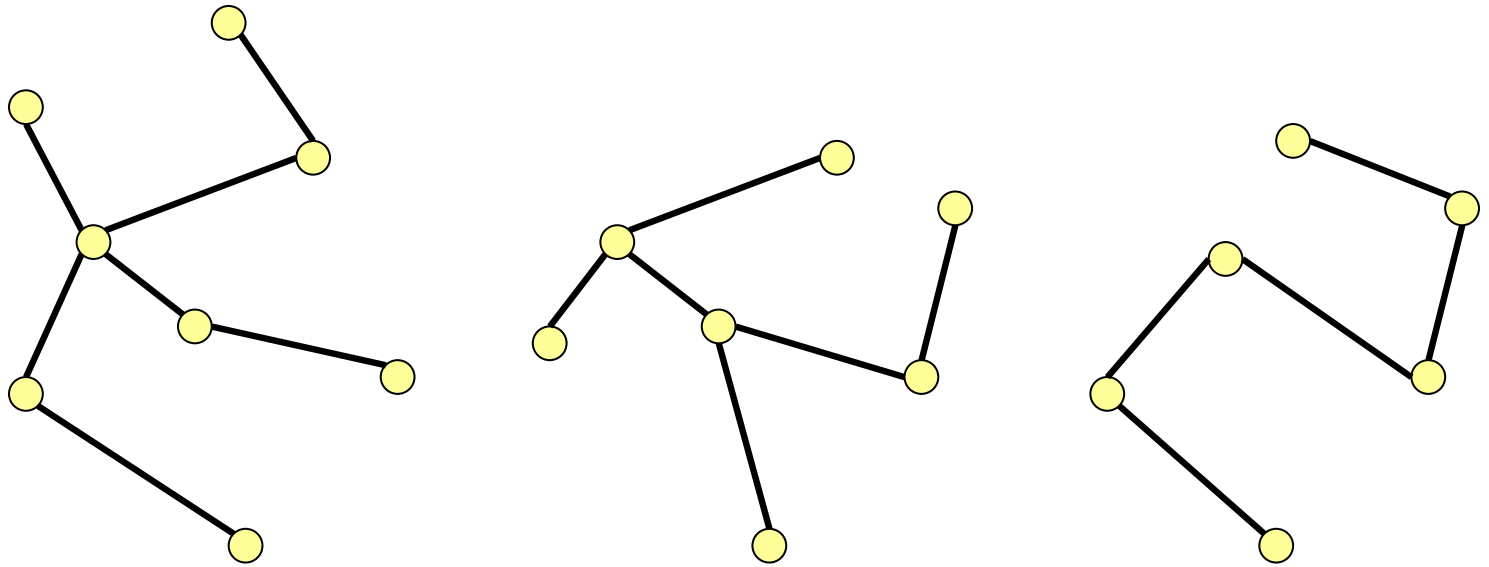
Δένδρα (2/14)

- **Δένδρο**: Συνεκτικό (μη κατευθυνόμενο) άκυκλο γράφημα.
- Τα δένδρα παίζουν κεντρικό ρόλο στην επιστήμη των υπολογιστών:
  - Ανάλυση αλγορίθμων (π.χ. δένδρα αναδρομής).
  - Δομές δεδομένων (π.χ. δένδρα αναζήτησης).
- Κατηγορίες (αύξουσα σειρά γενικότητας):
  - Δυαδικά και m-αδικά δένδρα
  - Διατεταγμένα δένδρα.
  - Δένδρα με ρίζα.
  - Ελεύθερα δένδρα.



Δένδρα (3/14)

- **Δένδρο**: Συνεκτικό (μη κατευθυνόμενο) άκυκλο γράφημα.
- **Δάσος**: Συλλογή από διαζευγμένα δένδρα (άκυκλο γράφημα).



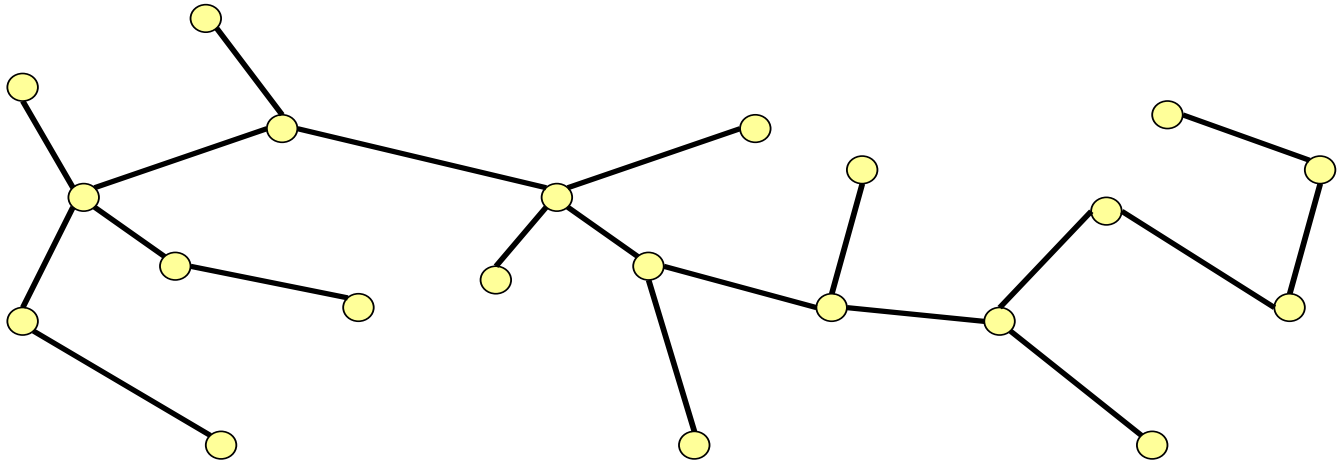
Δένδρα (4/14)

- **Δένδρο**: Συνεκτικό (μη κατευθυνόμενο) άκυκλο γράφημα.

Έστω $\mathbf{v}$ το πλήθος των κορυφών και $\mathbf{e}$ το πλήθος των ακμών

Ιδιότητες

1. Υπάρχει μοναδικό μονοπάτι μεταξύ κάθε δύο κορυφών
2. $\mathbf{v} = \mathbf{e} + 1$
3. Εαν $\mathbf{v} \geq 2$ τότε υπάρχουν τουλάχιστον 2 φύλλα



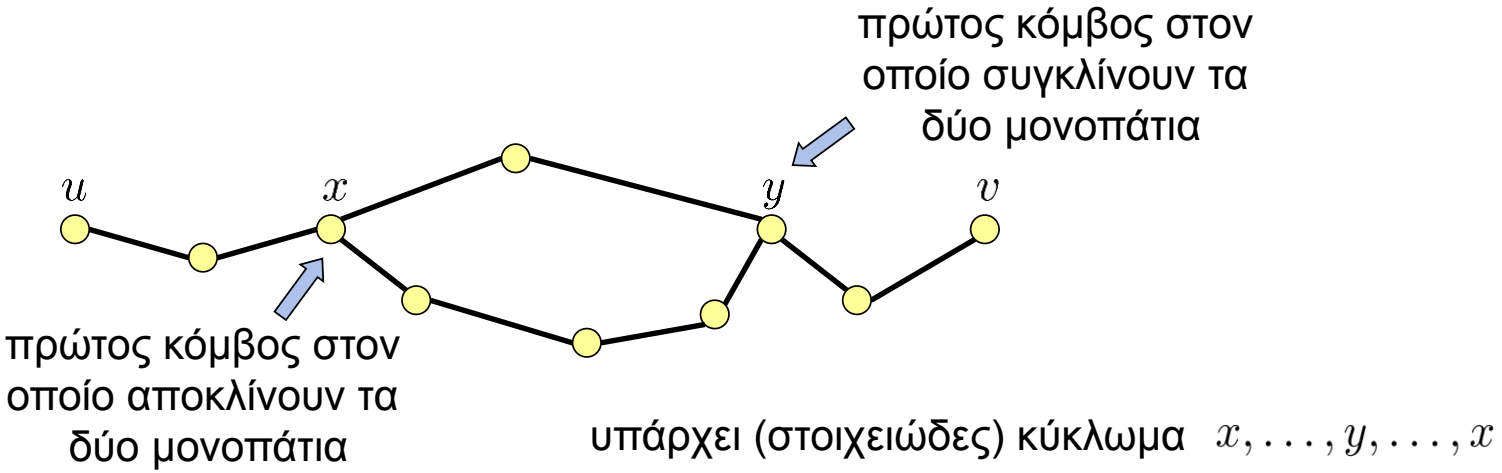
Δένδρα (5/14)

- **Δένδρο**: Συνεκτικό (μη κατευθυνόμενο) άκυκλο γράφημα.

Έστω $\mathbf{v}$ το πλήθος των κορυφών και $\mathbf{e}$ το πλήθος των ακμών

Ιδιότητες 1. Υπάρχει μοναδικό μονοπάτι μεταξύ κάθε δύο κορυφών

Απόδειξη Έστω ότι υπάρχουν 2 μονοπάτια μεταξύ των κορυφών u και v



Δένδρα (6/14)

- **Δένδρο**: Συνεκτικό (μη κατευθυνόμενο) άκυκλο γράφημα.

Έστω $\mathbf{v}$ το πλήθος των κορυφών και $\mathbf{e}$ το πλήθος των ακμών

Ιδιότητες 1. Υπάρχει μοναδικό μονοπάτι μεταξύ κάθε δύο κορυφών

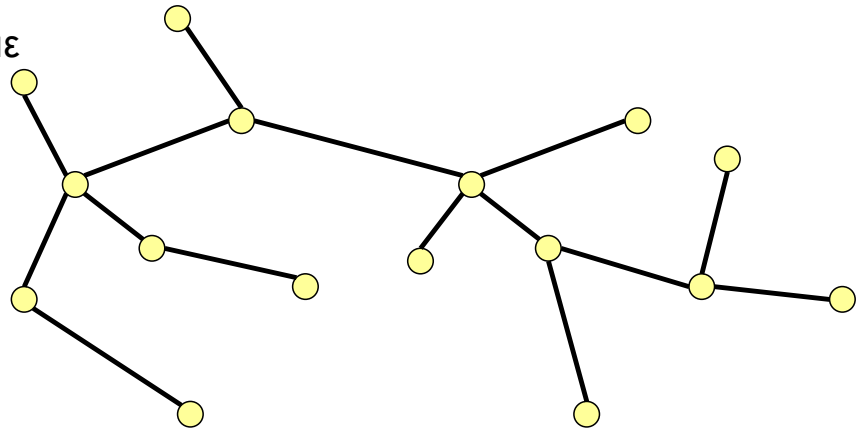
2. $\mathbf{v} = \mathbf{e} + 1$

Απόδειξη Επαγωγικά ως προς $\mathbf{v}$. Προφανώς ισχύει για $\mathbf{v} = 1$

Υποθέτουμε ότι ισχύει για κάθε δένδρο με λιγότερες από $\mathbf{v}$ κορυφές.

Θεωρούμε δένδρο με $\mathbf{v}$ κορυφές.

Αφαιρώντας μία ακμή λαμβάνουμε δάσος με 2 δένδρα.



Δένδρα (7/14)

- **Δένδρο**: Συνεκτικό (μη κατευθυνόμενο) άκυκλο γράφημα.

Έστω $\mathbf{v}$ το πλήθος των κορυφών και $\mathbf{e}$ το πλήθος των ακμών

Ιδιότητες 1. Υπάρχει μοναδικό μονοπάτι μεταξύ κάθε δύο κορυφών

2. $\mathbf{v} = \mathbf{e} + 1$

Απόδειξη Επαγωγικά ως προς $\mathbf{v}$. Προφανώς ισχύει για $\mathbf{v} = 1$

Υποθέτουμε ότι ισχύει για κάθε δένδρο με λιγότερες από $\mathbf{v}$ κορυφές.

Θεωρούμε δένδρο με $\mathbf{v}$ κορυφές. Αφαιρώντας μία ακμή λαμβάνουμε δάσος με 2 δένδρα. Έστω ότι το ένα έχει $\mathbf{v_1}$ κορυφές και $\mathbf{e_1}$ ακμές και το άλλο $\mathbf{v_2}$ κορυφές και $\mathbf{e_2}$ ακμές. Τότε $\mathbf{v_1} = \mathbf{e_1} + 1$ και $\mathbf{v_2} = \mathbf{e_2} + 1$

Προσθέτωντας τις δύο ισότητες λαμβάνουμε

$$\mathbf{v_1} + \mathbf{v_2} = \mathbf{e_1} + \mathbf{e_2} + 2 \Rightarrow$$
$$\mathbf{v} = \mathbf{e} + 1$$

Δένδρα (8/14)

- **Δένδρο**: Συνεκτικό (μη κατευθυνόμενο) άκυκλο γράφημα.

Έστω $\mathbf{v}$ το πλήθος των κορυφών και $\mathbf{e}$ το πλήθος των ακμών

Ιδιότητες

1. Υπάρχει μοναδικό μονοπάτι μεταξύ κάθε δύο κορυφών
2. $\mathbf{v} = \mathbf{e} + 1$
3. Εαν $\mathbf{v} \geq 2$ τότε υπάρχουν τουλάχιστον 2 φύλλα

Απόδειξη

Σε οποιοδήποτε γράφημα το άθροισμα των βαθμών όλων των κορυφών του είναι ίσο με $2\mathbf{e}$. Επομένως αν υπάρχει το πολύ μία κορυφή με βαθμό 1 τότε $2\mathbf{e} \geq 2(\mathbf{v} - 1) + 1$.

Αν το γράφημα είναι δένδρο τότε από την ιδιότητα 2 έχουμε

$$2\mathbf{e} \geq 2(\mathbf{v} - 1) + 1 = 2\mathbf{e} + 1$$

που προφανώς είναι άτοπο για $\mathbf{v} \geq 2$ αφού $\mathbf{e} \geq 1$

Δένδρα (9/14)

- **Δένδρο**: Συνεκτικό (μη κατευθυνόμενο) άκυκλο γράφημα.
- **Λήμμα:** Ένα γράφημα στο οποίο υπάρχει μοναδικό μονοπάτι μεταξύ δύο οποιονδήποτε κορυφών είναι δένδρο.
- **Απόδειξη**: Συνεπάγεται ότι το γράφημα είναι συνεκτικό και άκυκλο άρα είναι δένδρο.

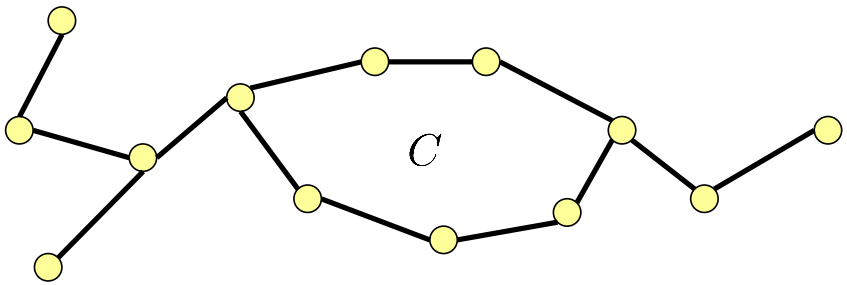
Δένδρα (10/14)

- **Ικανές συνθήκες για να είναι δένδρο ένα γράφημα.**

Λήμμα Ένα συνεκτικό γράφημα στο οποίο $\mathbf{v} = \mathbf{e} + 1$ είναι δένδρο.

Απόδειξη Υποθέτουμε ότι το γράφημα έχει ένα στοιχειώδες κύκλωμα C με $\mathbf{c}$ κορυφές και επομένως και $\mathbf{c}$ ακμές.

Αφού το γράφημα είναι συνεκτικό πρέπει να έχει τουλάχιστον $\mathbf{v} - \mathbf{c}$ ακμές έτσι ώστε οι κορυφές που δεν ανήκουν στο C να συνδέονται με τις κορυφές του C.



Επομένως υπάρχουν τουλάχιστον $\mathbf{c} + \mathbf{v} - \mathbf{c} = \mathbf{v}$ ακμές, που είναι άτοπο.

Δένδρα (11/14)

- **Ικανές συνθήκες για να είναι δένδρο ένα γράφημα.**

Λήμμα Ένα γράφημα με $\mathbf{v} = \mathbf{e} + 1$ που δεν περιέχει κύκλωμα είναι δένδρο.

Απόδειξη Από το προηγούμενο λήμμα αρκεί να δείξουμε ότι το γράφημα είναι συνεκτικό. Έστω $G_1, G_2, \ldots, G_k$ οι συνεκτικές συνιστώσες του γραφήματος όπου η συνιστώσα G_i έχει $\mathbf{v_i}$ κορυφές και $\mathbf{e_i}$ ακμές.

Αφού το γράφημα δεν περιέχει κύκλωμα κάθε G_i είναι συνεκτικό και άκυκλο άρα είναι δένδρο και επομένως ισχύει $\mathbf{v_i} = \mathbf{e_i} + 1$. Αθροίζοντας έχουμε

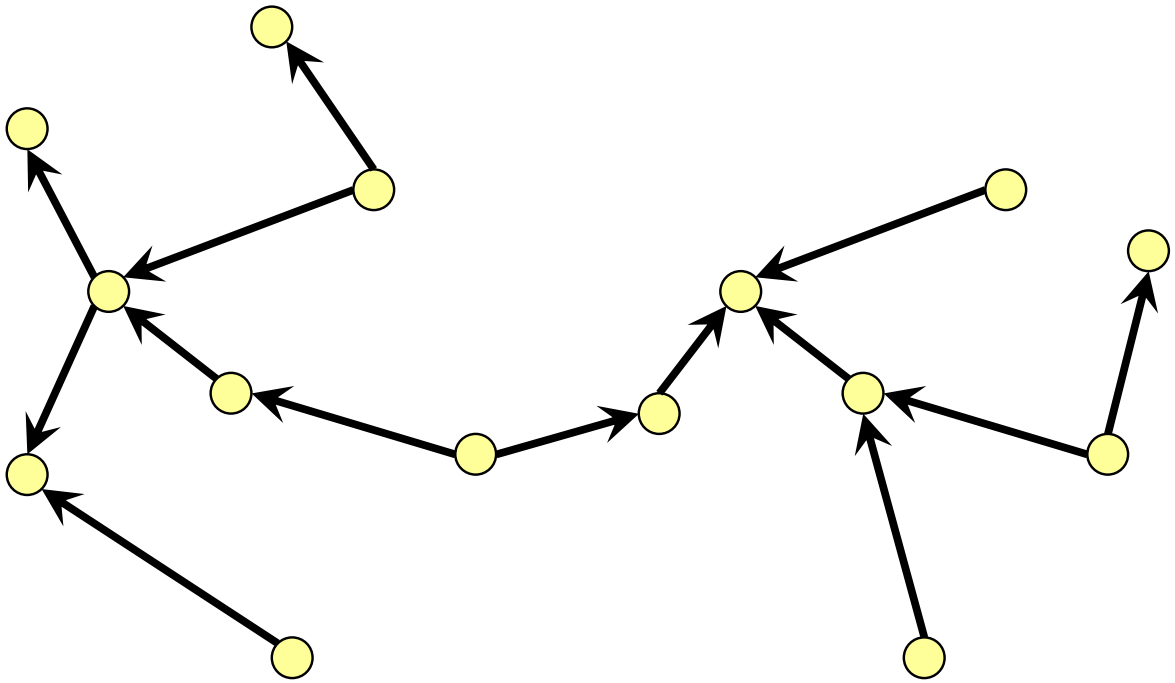
$$\sum_{i=1}^{k} \mathbf{v_i} = \sum_{i=1}^{k} (\mathbf{e_i} + 1) \Rightarrow \sum_{i=1}^{k} \mathbf{v_i} = \sum_{i=1}^{k} \mathbf{e_i} + k \Rightarrow \mathbf{v} = \mathbf{e} + k$$

και από την υπόθεση του λήμματος προκύπτει $k = 1$.

Δένδρα (12/14)

Δένδρα με ρίζες.

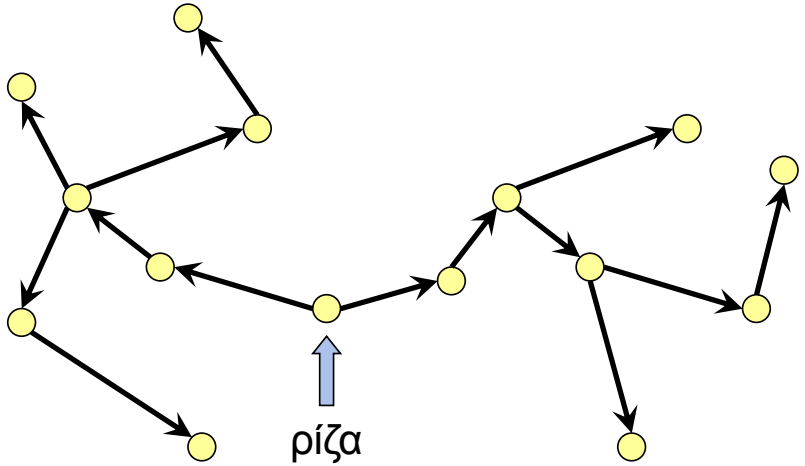
- **Κατευθυνόμενο δένδρο:** Κατευθυνόμενο γράφημα που μετατρέπεται σε δένδρο αν αγνοήσουμε τις κατευθύνσεις των ακμών.



Δένδρα (13/14)

Δένδρα με ρίζες.

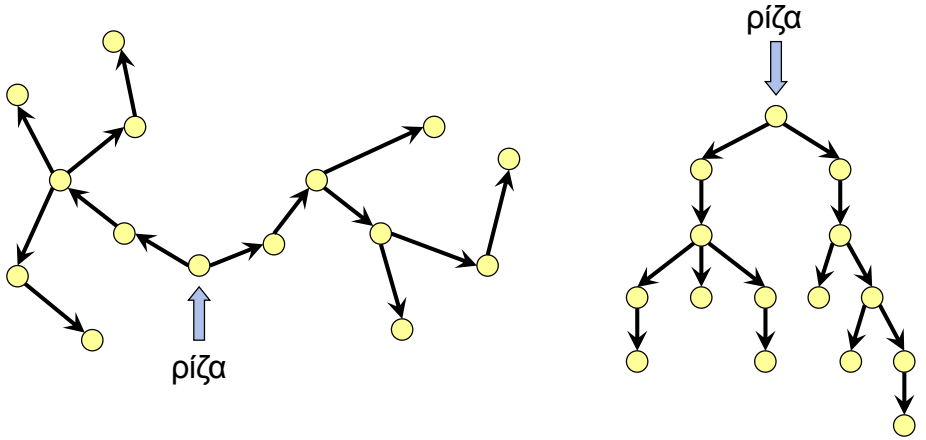
- **Κατευθυνόμενο δένδρο:** Κατευθυνόμενο γράφημα που μετατρέπεται σε δένδρο αν αγνοήσουμε τις κατευθύνσεις των ακμών.
- **Δένδρο με ρίζα:** Κατευθυνόμενο δένδρο στο οποίο υπάρχει ένας κόμβος με βαθμό εισόδου 0 και όλοι οι υπόλοιποι κόμβοι έχουν βαθμό εισόδου 1.



Δένδρα (14/14)

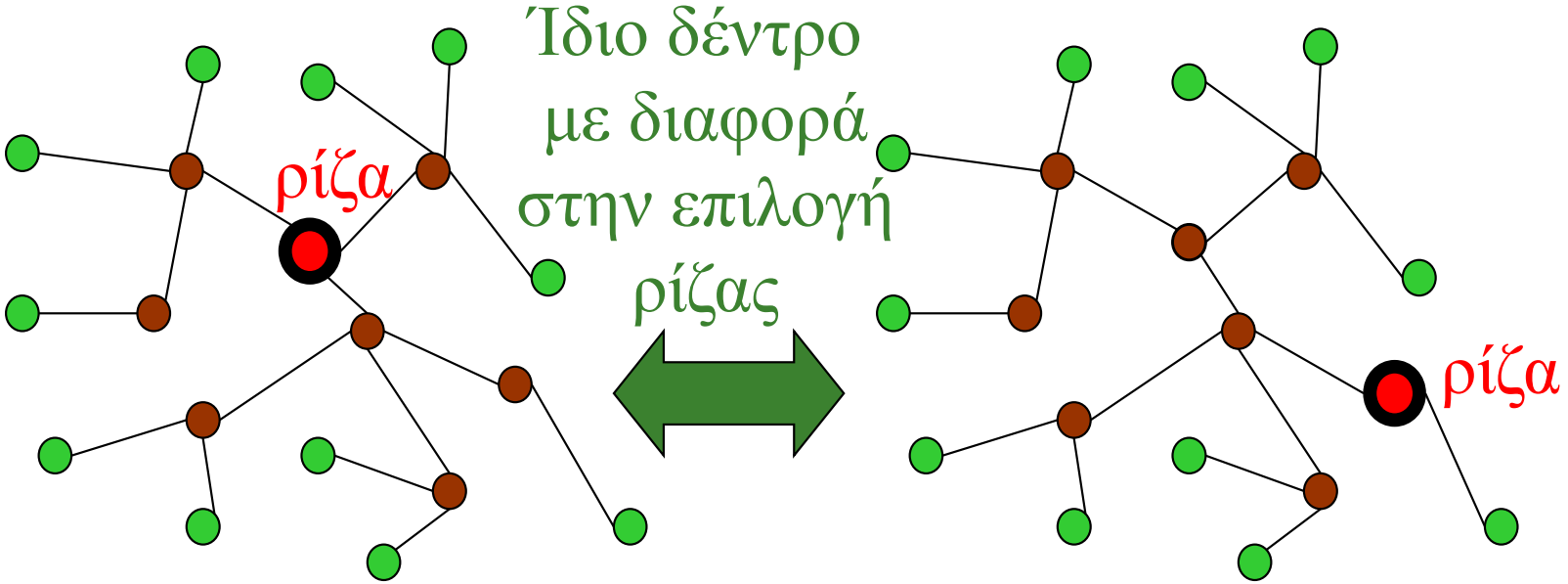
Δένδρα με ρίζες.

- **Κατευθυνόμενο δένδρο:** Κατευθυνόμενο γράφημα που μετατρέπεται σε δένδρο αν αγνοήσουμε τις κατευθύνσεις των ακμών.
- **Δένδρο με ρίζα:** Κατευθυνόμενο δένδρο στο οποίο υπάρχει ένας κόμβος με βαθμό εισόδου 0 και όλοι οι υπόλοιποι κόμβοι έχουν βαθμό εισόδου 1.



Δέντρα με Ρίζα

- Από ένα δέντρο χωρίς ρίζα με *n* κόμβους μπορούμε να πάρουμε *n* δέντρα με ρίζα.



Δέντρα με Ρίζες (1/8)

- **Κατευθυνόμενο δένδρο:** Κατευθυνόμενο γράφημα που μετατρέπεται σε δένδρο αν αγνοήσουμε τις κατευθύνσεις των ακμών.
- **Δένδρο με ρίζα:** Κατευθυνόμενο δένδρο στο οποίο υπάρχει ένας κόμβος με βαθμό εισόδου 0 και όλοι οι υπόλοιποι κόμβοι έχουν βαθμό εισόδου 1.

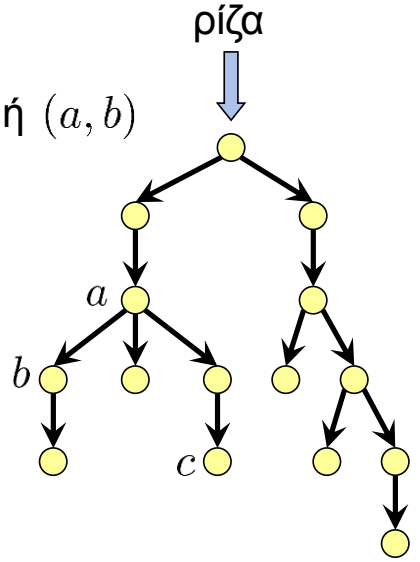
Ο κόμβος a είναι **γονιός** του κόμβου b εάν υπάρχει η ακμή (a, b)

Επίσης λέμε ότι ο b είναι **παιδί** του a.

Δύο κόμβοι είναι **αδέλφια** εάν έχουν τον ίδιο γονιό.

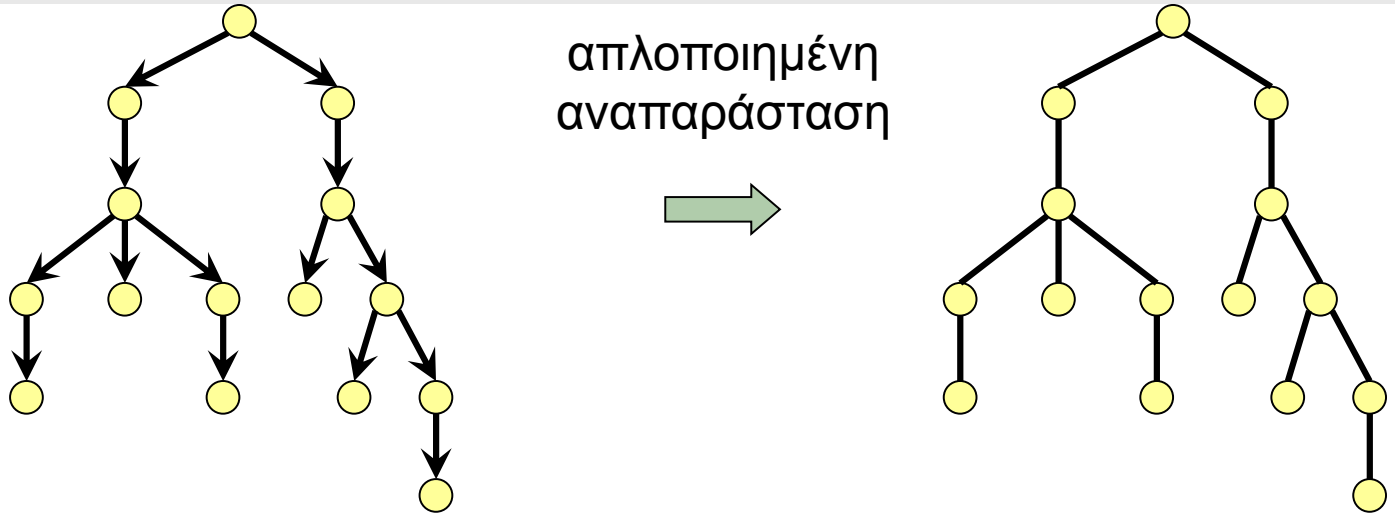
Ο a είναι **πρόγονος** του c εάν υπάρχει μονοπάτι από τον a στον c.

Επίσης λέμε ότι ο c είναι **απόγονος** του a.



Δέντρα με Ρίζες (2/8)

- **Κατευθυνόμενο δένδρο:** Κατευθυνόμενο γράφημα που μετατρέπεται σε δένδρο αν αγνοήσουμε τις κατευθύνσεις των ακμών.
- **Δένδρο με ρίζα:** Κατευθυνόμενο δένδρο στο οποίο υπάρχει ένας κόμβος με βαθμό εισόδου 0 και όλοι οι υπόλοιποι κόμβοι έχουν βαθμό εισόδου 1.



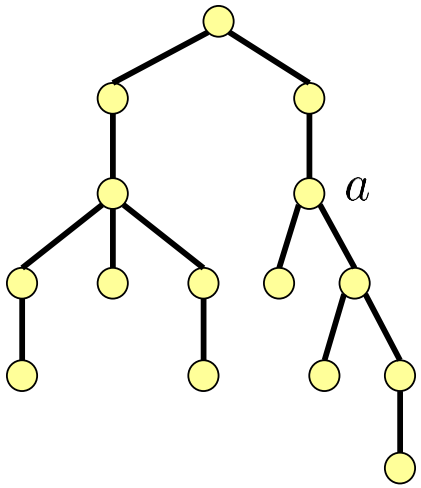
Μπορούμε να παραλείψουμε τις κατευθύνσεις των ακμών.

Δέντρα με Ρίζες (3/8)

- **Κατευθυνόμενο δένδρο:** Κατευθυνόμενο γράφημα που μετατρέπεται σε δένδρο αν αγνοήσουμε τις κατευθύνσεις των ακμών.
- **Δένδρο με ρίζα:** Κατευθυνόμενο δένδρο στο οποίο υπάρχει ένας κόμβος με βαθμό εισόδου 0 και όλοι οι υπόλοιποι κόμβοι έχουν βαθμό εισόδου 1.

Υποδένδρο του a (υποδένδρο με ρίζα το a)

Έχει κόμβους το a και τους απογόνους του στο αρχικό δένδρο και περιλαμβάνει όλες τις ακμές του αρχικού δένδρου που συνδέουν κόμβους του υποδένδρου.



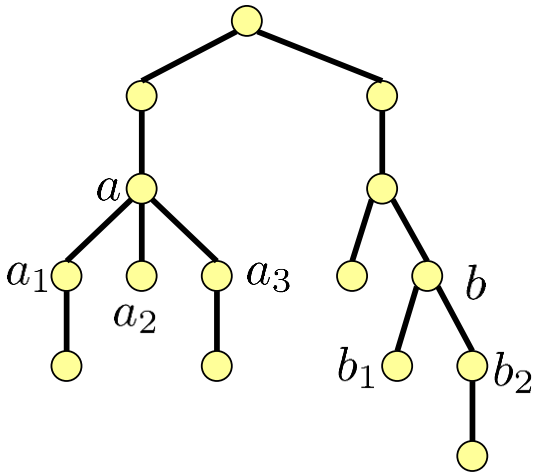
Δέντρα με Ρίζες (4/8)

- **Κατευθυνόμενο δένδρο:** Κατευθυνόμενο γράφημα που μετατρέπεται σε δένδρο αν αγνοήσουμε τις κατευθύνσεις των ακμών.
- **Δένδρο με ρίζα:** Κατευθυνόμενο δένδρο στο οποίο υπάρχει ένας κόμβος με βαθμό εισόδου 0 και όλοι οι υπόλοιποι κόμβοι έχουν βαθμό εισόδου 1.
- **Διατεταγμένο δένδρο:** Δένδρο με ρίζα στο οποίο υπάρχει διάταξη των παιδιών κάθε κόμβου.

Π.χ. a_1 = 1ο παιδί του a

a_2 = 2ο παιδί του a

a_3 = 3ο παιδί του a



b_1 = 1ο παιδί του b

b_2 = 2ο παιδί του b

Δέντρα με Ρίζες (5/8)

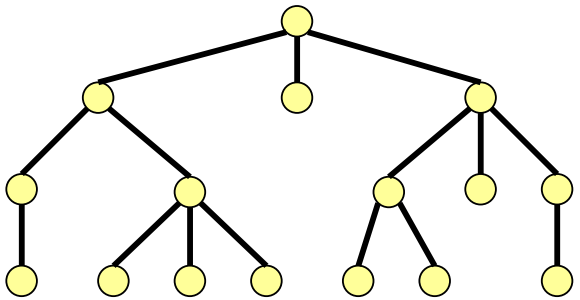
- **Κατευθυνόμενο δένδρο:** Κατευθυνόμενο γράφημα που μετατρέπεται σε δένδρο αν αγνοήσουμε τις κατευθύνσεις των ακμών.
- **Δένδρο με ρίζα:** Κατευθυνόμενο δένδρο στο οποίο υπάρχει ένας κόμβος με βαθμό εισόδου 0 και όλοι οι υπόλοιποι κόμβοι έχουν βαθμό εισόδου 1.
- **Διατεταγμένο δένδρο:** Δένδρο με ρίζα στο οποίο υπάρχει διάταξη των παιδιών κάθε κόμβου.

Δέντρα με Ρίζες (6/8)

- **Κατευθυνόμενο δένδρο:** Κατευθυνόμενο γράφημα που μετατρέπεται σε δένδρο αν αγνοήσουμε τις κατευθύνσεις των ακμών.
- **Δένδρο με ρίζα:** Κατευθυνόμενο δένδρο στο οποίο υπάρχει ένας κόμβος με βαθμό εισόδου 0 και όλοι οι υπόλοιποι κόμβοι έχουν βαθμό εισόδου 1.
- **Διατεταγμένο δένδρο:** Δένδρο με ρίζα στο οποίο υπάρχει διάταξη των παιδιών κάθε κόμβου.

m-αδικό δένδρο Διατεταγμένο δένδρο όπου κάθε κόμβος έχει το πολύ m παιδιά.

Π.χ. $m = 3$



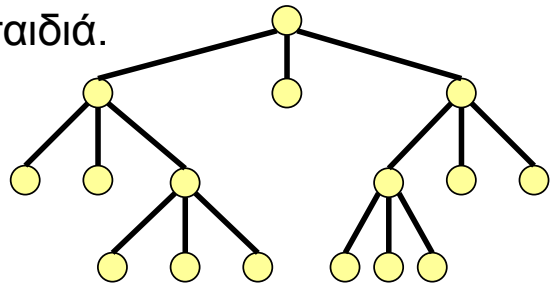
Δέντρα με Ρίζες (7/8)

- **Κατευθυνόμενο δένδρο:** Κατευθυνόμενο γράφημα που μετατρέπεται σε δένδρο αν αγνοήσουμε τις κατευθύνσεις των ακμών.
- **Δένδρο με ρίζα:** Κατευθυνόμενο δένδρο στο οποίο υπάρχει ένας κόμβος με βαθμό εισόδου 0 και όλοι οι υπόλοιποι κόμβοι έχουν βαθμό εισόδου 1.
- **Διατεταγμένο δένδρο:** Δένδρο με ρίζα στο οποίο υπάρχει διάταξη των παιδιών κάθε κόμβου.

m-αδικό δένδρο Διατεταγμένο δένδρο όπου κάθε κόμβος έχει το πολύ m παιδιά.

Κανονικό m-αδικό δένδρο m-αδικό δένδρο όπου κάθε εσωτερικός κόμβος έχει ακριβώς m παιδιά.

Π.χ. $m = 3$



Δέντρα με Ρίζες (8/8)

- **Κατευθυνόμενο δένδρο:** Κατευθυνόμενο γράφημα που μετατρέπεται σε δένδρο αν αγνοήσουμε τις κατευθύνσεις των ακμών.
- **Δένδρο με ρίζα:** Κατευθυνόμενο δένδρο στο οποίο υπάρχει ένας κόμβος με βαθμό εισόδου 0 και όλοι οι υπόλοιποι κόμβοι έχουν βαθμό εισόδου 1.
- **Διατεταγμένο δένδρο:** Δένδρο με ρίζα στο οποίο υπάρχει διάταξη των παιδιών κάθε κόμβου.

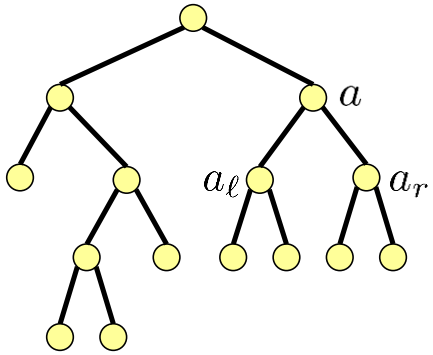
m-αδικό δένδρο Διατεταγμένο δένδρο όπου κάθε κόμβος έχει το πολύ m παιδιά.

Κανονικό m-αδικό δένδρο m-αδικό δένδρο όπου κάθε εσωτερικός κόμβος έχει ακριβώς m παιδιά.

Δυαδικό δένδρα Σημαντική κατηγορία m-αδικών δένδρων με $m = 2$.

Δυαδικό δένδρα (1/4)

Δυαδικό δένδρα: Σημαντική κατηγορία m-αδικών δένδρων με $m = 2$.

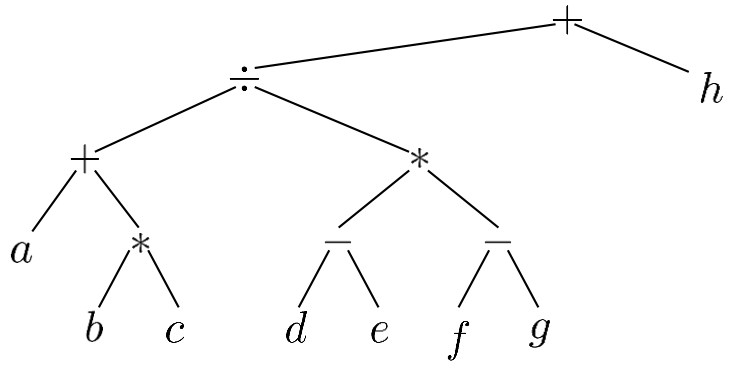


a_ℓ = αριστερό παιδί του a

a_r = δεξί παιδί του a

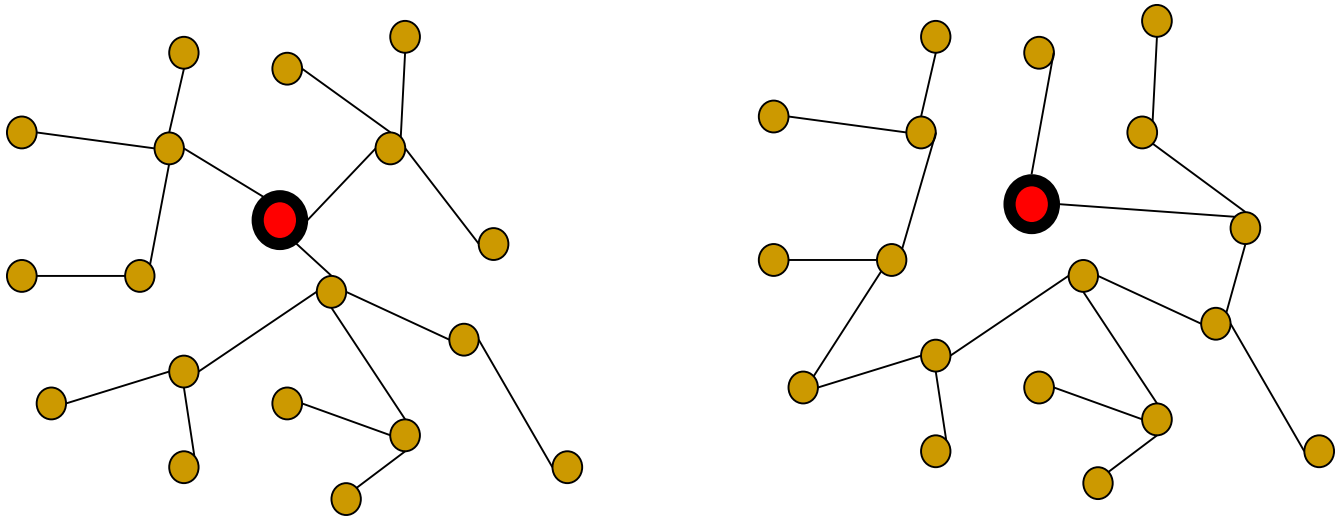
Παράδειγμα: Αναπαράσταση αριθμητικών εκφράσεων

$$(a + b * c)/((d - e) * (f - g)) + h$$



Δυαδικό δένδρα (2/4)

- **Θεώρημα:** Ένα δέντρο με ρίζα είναι δυαδικό αν και μόνο αν κάθε κόμβος εκτός από τη ρίζα έχει βαθμό ≤ ___, και η ρίζα έχει βαθμό ≤ ___.

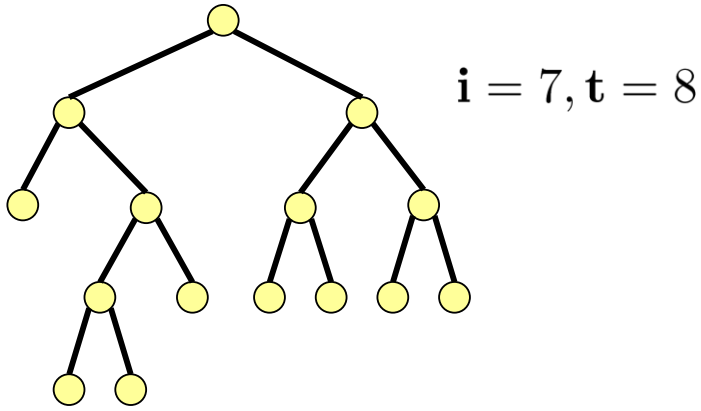


Δυαδικό δένδρα (3/4)

Θεωρούμε ένα κανονικό δυαδικό δένδρο με $\mathbf{i}$ εσωτερικούς κόμβους και $\mathbf{t}$ φύλλα

Αφού κάθε εσωτερικός κόμβος έχει 2 παιδιά και η ρίζα δεν έχει γονέα τότε ισχύει

$$2\mathbf{i} + 1 = \mathbf{t} + \mathbf{i} \Rightarrow \mathbf{i} = \mathbf{t} - 1$$

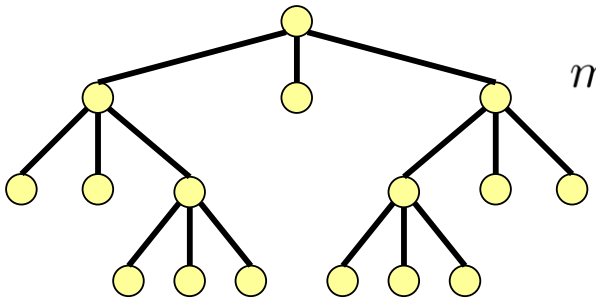


$\mathbf{i} = 7, \mathbf{t} = 8$

Δυαδικό δένδρα (4/4)

Γενικά σε ένα κανονικό m-αδικό δένδρο με $\mathbf{i}$ εσωτερικούς κόμβους και $\mathbf{t}$ φύλλα έχουμε ότι κάθε εσωτερικός κόμβος έχει m παιδιά και η ρίζα δεν έχει γονέα οπότε

$$m\mathbf{i} + 1 = \mathbf{t} + \mathbf{i} \;\Rightarrow\; (m-1)\mathbf{i} = \mathbf{t} - 1$$



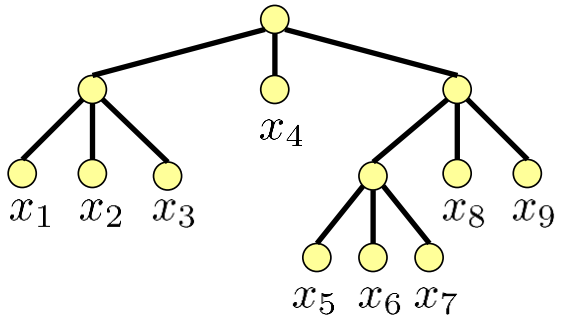
$$m = 3, \mathbf{i} = 5, \mathbf{t} = 11$$

Παράδειγμα (1/2)

Παράδειγμα Έστω ότι έχουμε μία μηχανή που υπολογίζει το άθροισμα 3 αριθμών με μία εντολή. Πόσες εντολές πρέπει να εκτελέσει για να υπολογίσει το άθροισμα 9 αριθμών $x_1, x_2, \ldots, x_9$;

Κάθε ακολουθία εκτελέσεων αυτής της εντολής αντιστοιχεί σε κάποιο κανονικό 3-αδικό δένδρο όπου κάθε εσωτερικός κόμβος αναπαριστά μία πράξη πρόσθεσης 3 αριθμών και τα φύλλα αντιστοιχούν στους αριθμούς της εισόδου.

Π.χ. $(x_1 + x_2 + x_3) + x_4 + ((x_5 + x_6 + x_7) + x_8 + x_9)$

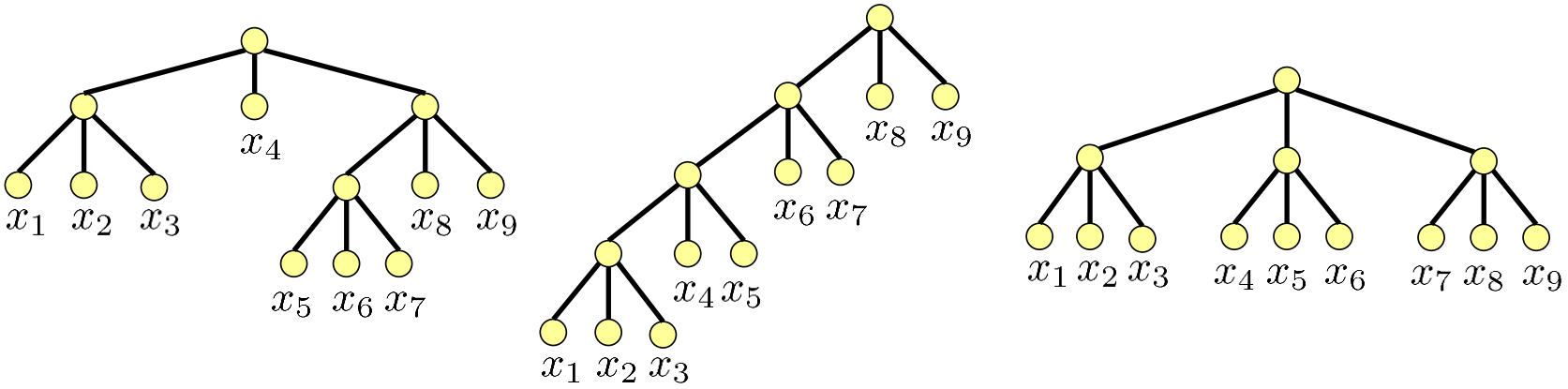


Έχουμε $\mathbf{i} = (\mathbf{t} - 1)/(m - 1) = 8/2 = 4$ άρα χρειαζόμαστε 4 πράξεις ανεξάρτητα από τη σειρά των προσθέσεων με την οποία υπολογίζεται το ολικό άθροισμα.

Παράδειγμα (2/2)

Παράδειγμα Έστω ότι έχουμε μία μηχανή που υπολογίζει το άθροισμα 3 αριθμών με μία εντολή. Πόσες εντολές πρέπει να εκτελέσει για να υπολογίσει το άθροισμα 9 αριθμών $x_1, x_2, \ldots, x_9$;

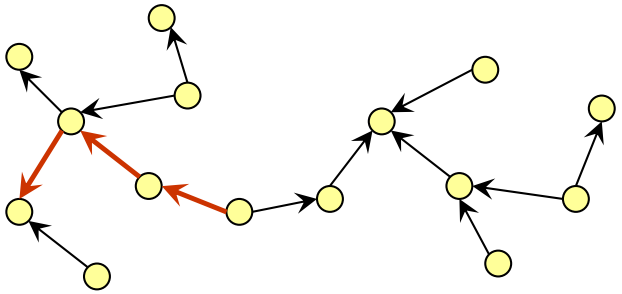
Κάθε ακολουθία εκτελέσεων αυτής της εντολής αντιστοιχεί σε κάποιο κανονικό 3-αδικό δένδρο όπου κάθε εσωτερικός κόμβος αναπαριστά μία πράξη πρόσθεσης 3 αριθμών και τα φύλλα αντιστοιχούν στους αριθμούς της εισόδου.



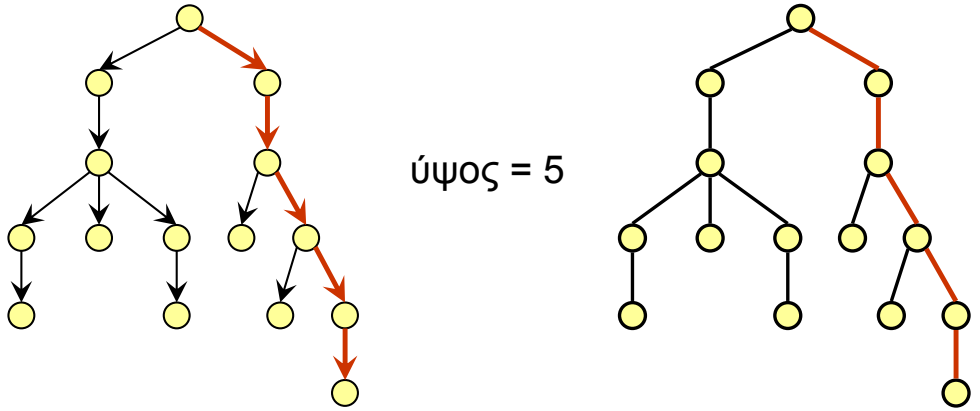
Ποίος τρόπος είναι προτιμότερος όταν μπορούμε να κάνουμε παράλληλους υπολογισμούς;

Ύψος δένδρου (1/18)

΄Υψος δένδρου Μέγιστο από τα μήκη μονοπατιών του δένδρου

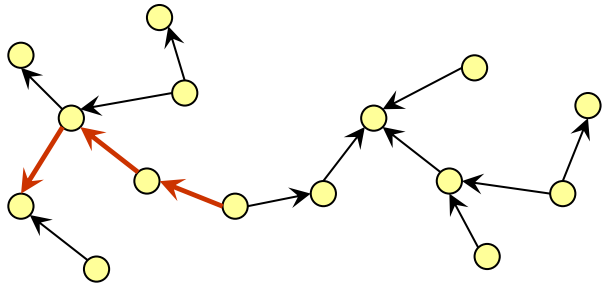


Στα δένδρα με ρίζα το ύψος αντιστοιχεί στο μέγιστο αριθμό ακμών από τη ρίζα προς κάποιο φύλλο

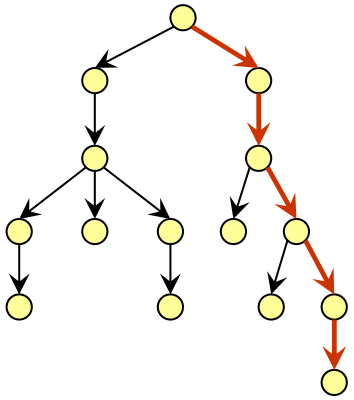


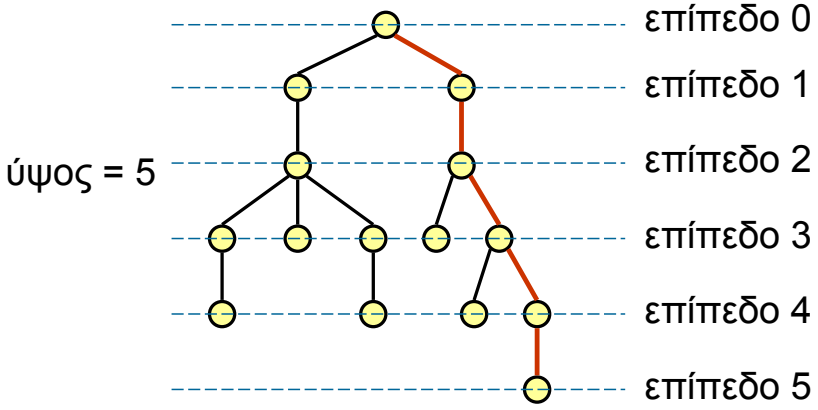
Ύψος δένδρου (2/18)

΄Υψος δένδρου Μέγιστο από τα μήκη μονοπατιών του δένδρου



Στα δένδρα με ρίζα το ύψος αντιστοιχεί στο μέγιστο αριθμό ακμών από τη ρίζα προς κάποιο φύλλο





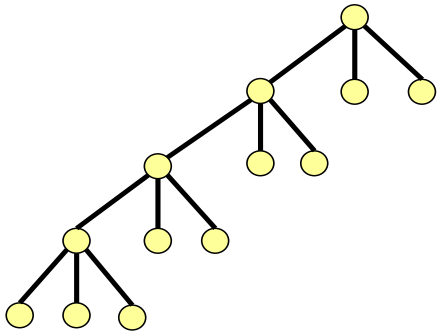
Ύψος δένδρου (3/18)

΄Υψος δένδρου Μέγιστο από τα μήκη μονοπατιών του δένδρου

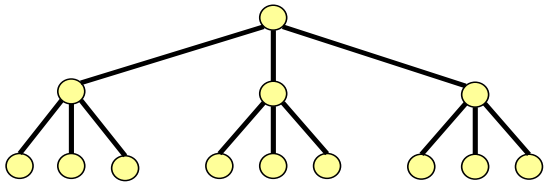
Στα δένδρα με ρίζα το ύψος αντιστοιχεί στο μέγιστο αριθμό ακμών από τη ρίζα προς κάποιο φύλλο

Ένα κανονικό m-αδικό δένδρο με ύψος h έχει αριθμό φύλλων $\mathbf{t}$ όπου

$$(m-1)h+1 \leq \mathbf{t} \leq m^h$$



(m-1) φύλλα στα επίπεδα 1 έως h-1 και m φύλλα στο επίπεδο h



όλα τα φύλλα στο τελευταίο επίπεδο

Ύψος δένδρου (4/18)

Ύψος δένδρου Μέγιστο από τα μήκη μονοπατιών του δένδρου

Στα δένδρα με ρίζα το ύψος αντιστοιχεί στο μέγιστο αριθμό ακμών από τη ρίζα προς κάποιο φύλλο

Ένα κανονικό m-αδικό δένδρο με ύψος h έχει αριθμό φύλλων $\mathbf{t}$ όπου

$$(m-1)h+1 \leq \mathbf{t} \leq m^h \Rightarrow$$

$$\log_m \mathbf{t} \leq h \leq \frac{\mathbf{t}-1}{m-1}$$

Επιπλέον για το πλήθος των εσωτερικών κόμβων $\mathbf{i}$ ισχύει $(m-1)\mathbf{i} = \mathbf{t}-1$

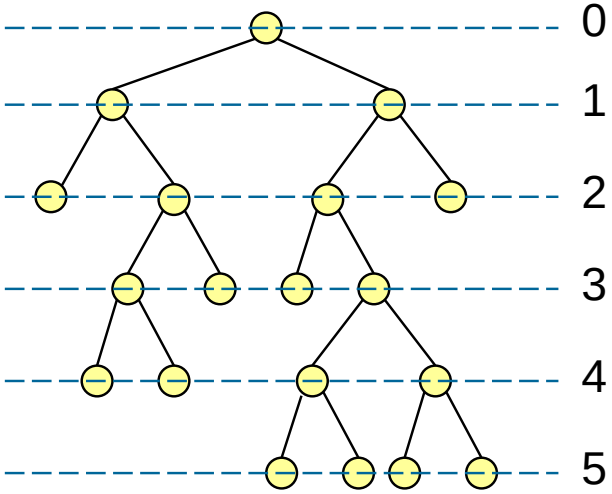
Άρα το δένδρο έχει συνολικά $\mathbf{v} = \mathbf{t} + \frac{\mathbf{t}-1}{m-1} = \frac{m\mathbf{t}-1}{m-1}$ κόμβους και επομένως

$\mathbf{t} = \frac{(m-1)\mathbf{v}+1}{m}$. Έτσι $h \geq \log_m \frac{(m-1)\mathbf{v}+1}{m} = \log_m [(m-1)\mathbf{v}+1] - 1$

Ύψος δένδρου (5/18)

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1

ύψος δένδρου = μέγιστο επίπεδο



I : μήκος εσωτερικής διαδρομής = άθροισμα επιπέδου κάθε εσωτερικού κόμβου (=20)

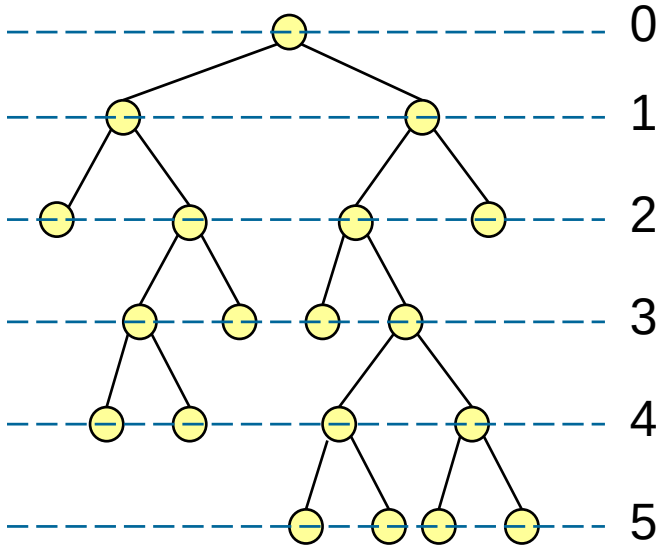
E : μήκος εξωτερικής διαδρομής = άθροισμα επιπέδου κάθε τερματικού κόμβου (=38)

Σε ένα κανονικό δυαδικό δένδρο ισχύει $E = I + 2\mathbf{i}$

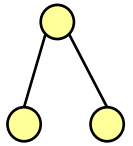
Ύψος δένδρου (6/18)

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1

ύψος δένδρου = μέγιστο επίπεδο



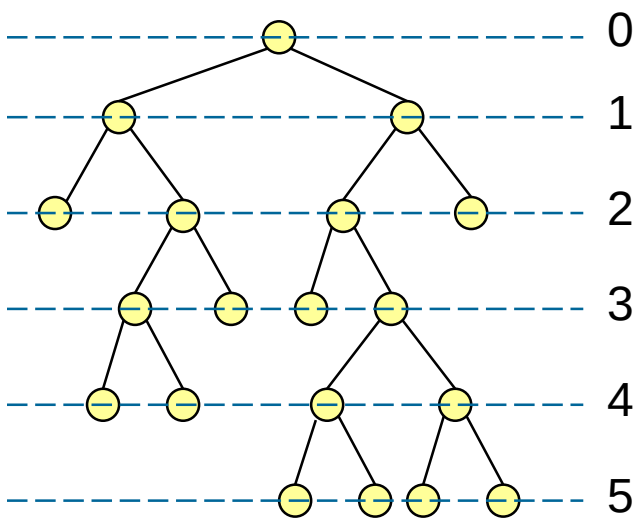
ξεκινάμε με ένα φύλλο

σε κάθε βήμα   

Ύψος δένδρου (7/18)

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1

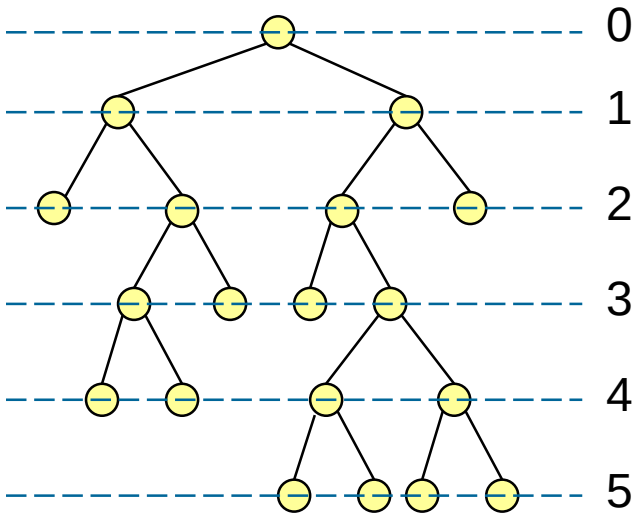
ύψος δένδρου = μέγιστο επίπεδο

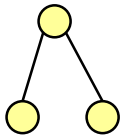


Ύψος δένδρου (8/18)

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1

ύψος δένδρου = μέγιστο επίπεδο

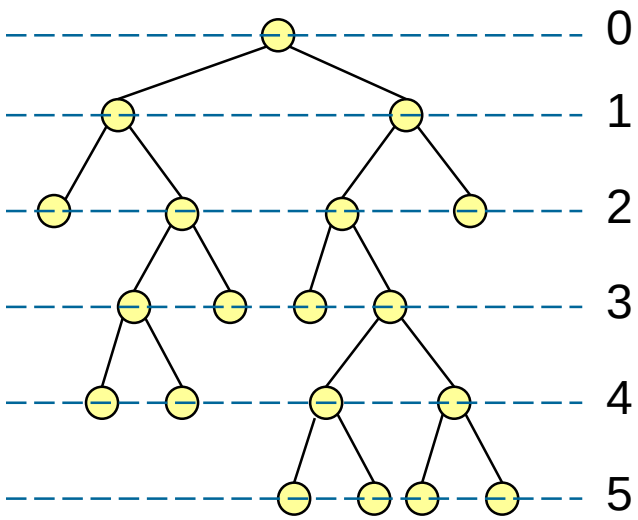


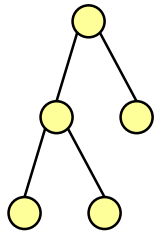


Ύψος δένδρου (9/18)

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1

ύψος δένδρου = μέγιστο επίπεδο

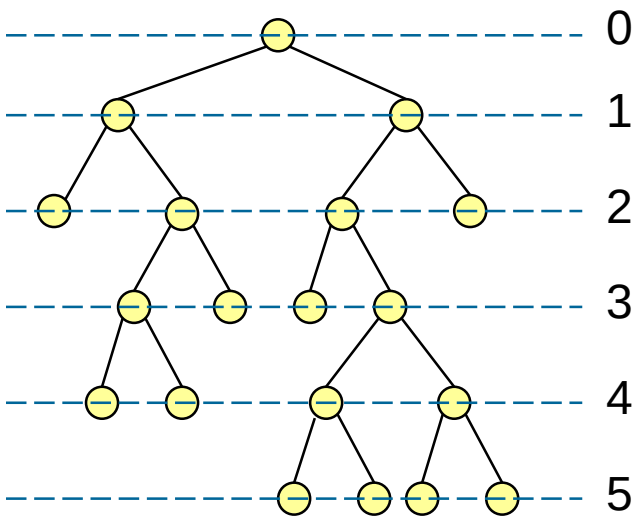


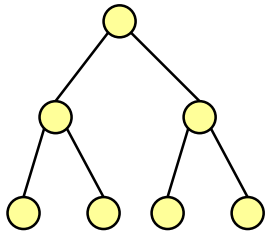


Ύψος δένδρου (10/18)

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1

ύψος δένδρου = μέγιστο επίπεδο

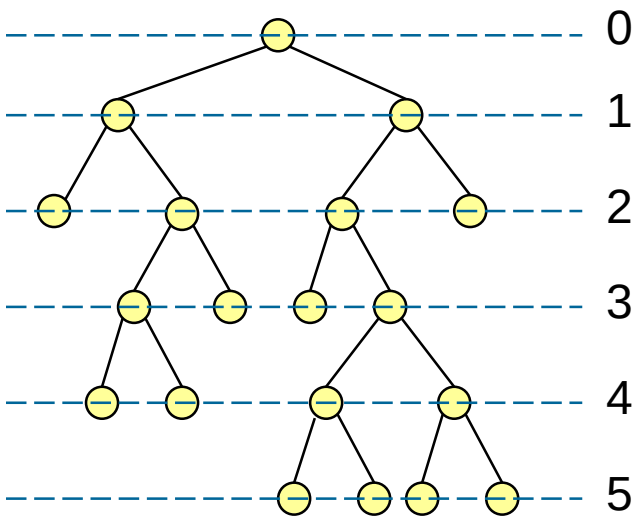


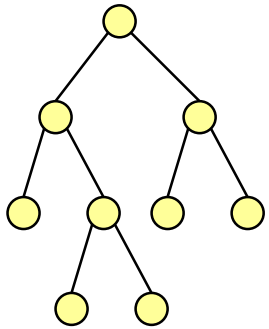


Ύψος δένδρου (11/18)

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1

ύψος δένδρου = μέγιστο επίπεδο

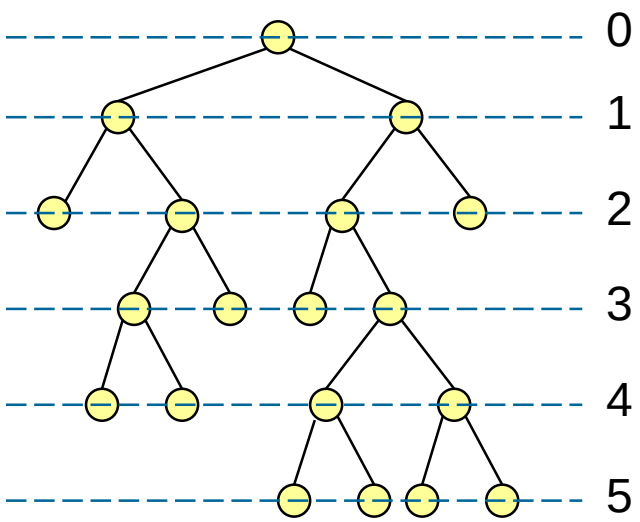


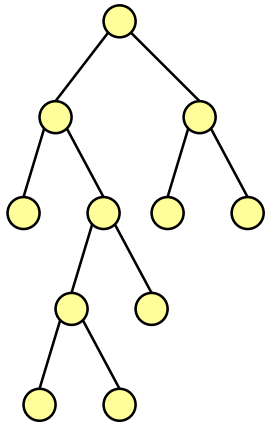


Ύψος δένδρου (12/18)

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1

ύψος δένδρου = μέγιστο επίπεδο

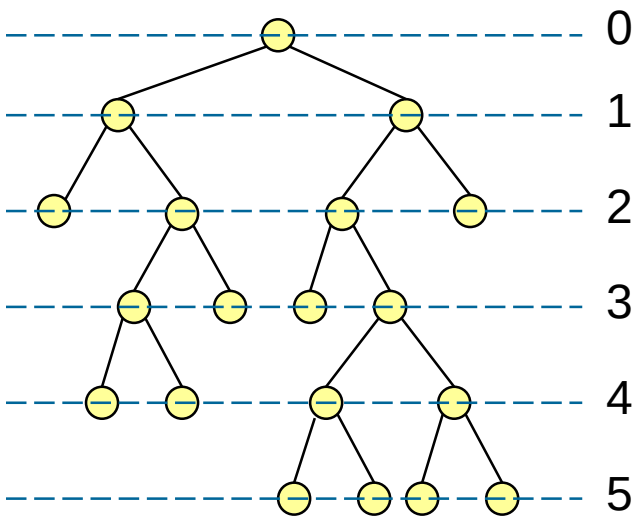


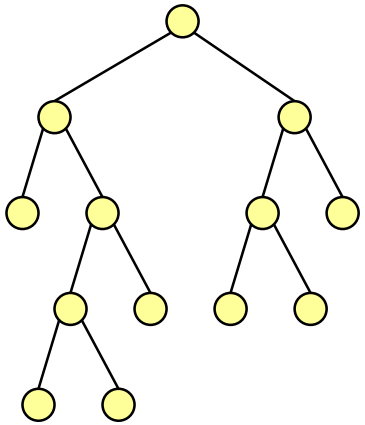


Ύψος δένδρου (13/18)

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1

ύψος δένδρου = μέγιστο επίπεδο

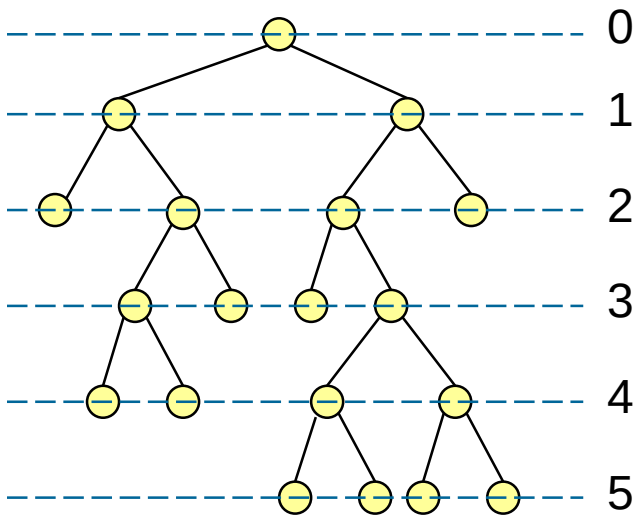


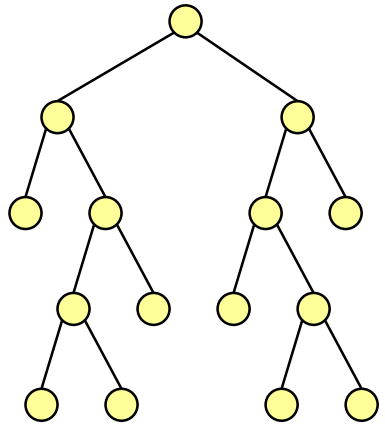


Ύψος δένδρου (14/18)

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1

ύψος δένδρου = μέγιστο επίπεδο

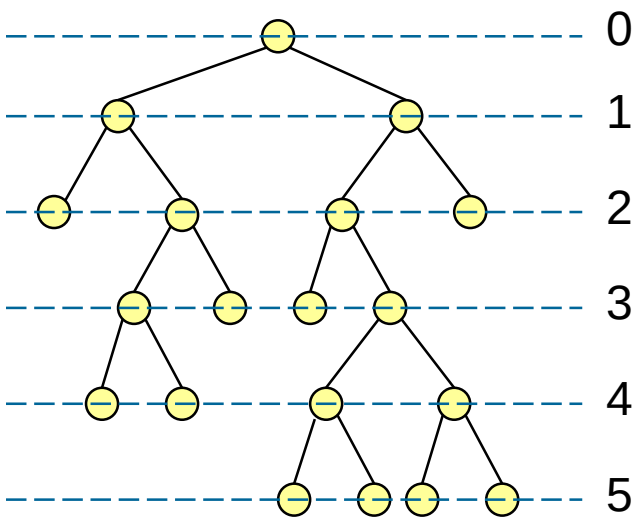


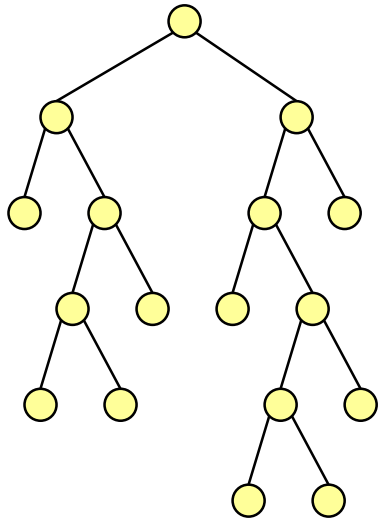


Ύψος δένδρου (15/18)

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1

ύψος δένδρου = μέγιστο επίπεδο

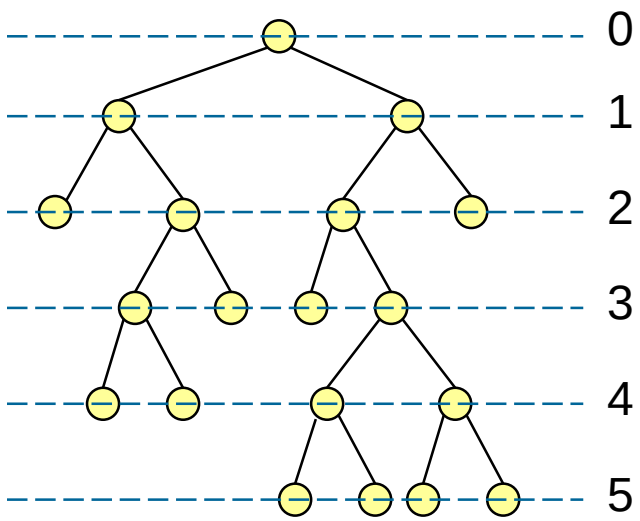


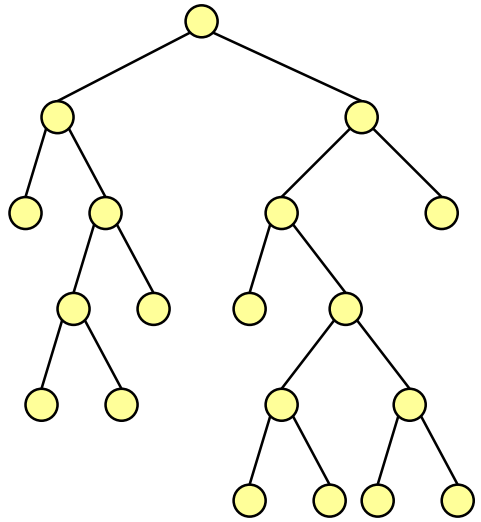


Ύψος δένδρου (16/18)

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1

ύψος δένδρου = μέγιστο επίπεδο

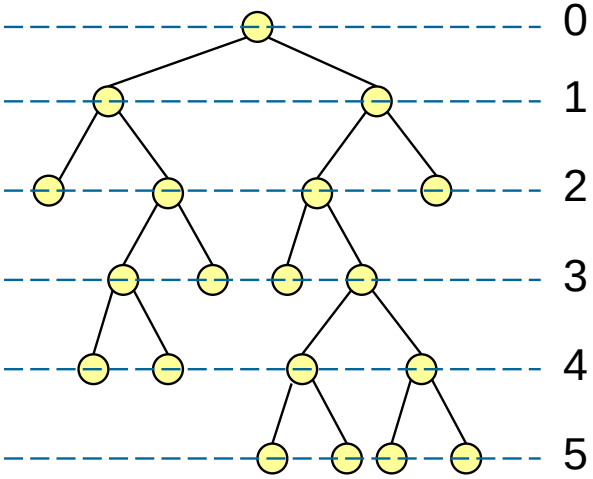




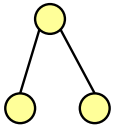
Ύψος δένδρου (17/18)

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1

ύψος δένδρου = μέγιστο επίπεδο



ξεκινάμε με ένα φύλλο

σε κάθε βήμα   

Αντικατάσταση εσωτερικού κόμβου στο επίπεδο k:

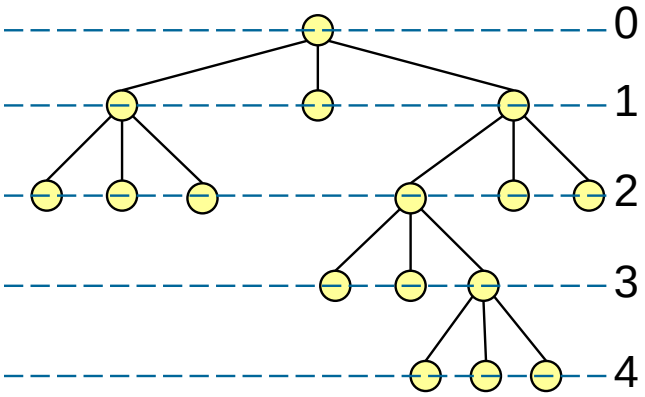
- μήκος εσωτερικής διαδρομής αυξάνει κατά k
- μήκος εξωτερικής διαδρομής αυξάνει κατά k+2

Επομένως $E = I + 2\mathbf{i}$

Ύψος δένδρου (18/18)

επίπεδο ρίζας = 0
επίπεδο κόμβου = επίπεδο γονέα + 1

ύψος δένδρου = μέγιστο επίπεδο



I : μήκος εσωτερικής διαδρομής = άθροισμα επιπέδου κάθε εσωτερικού κόμβου (=7)

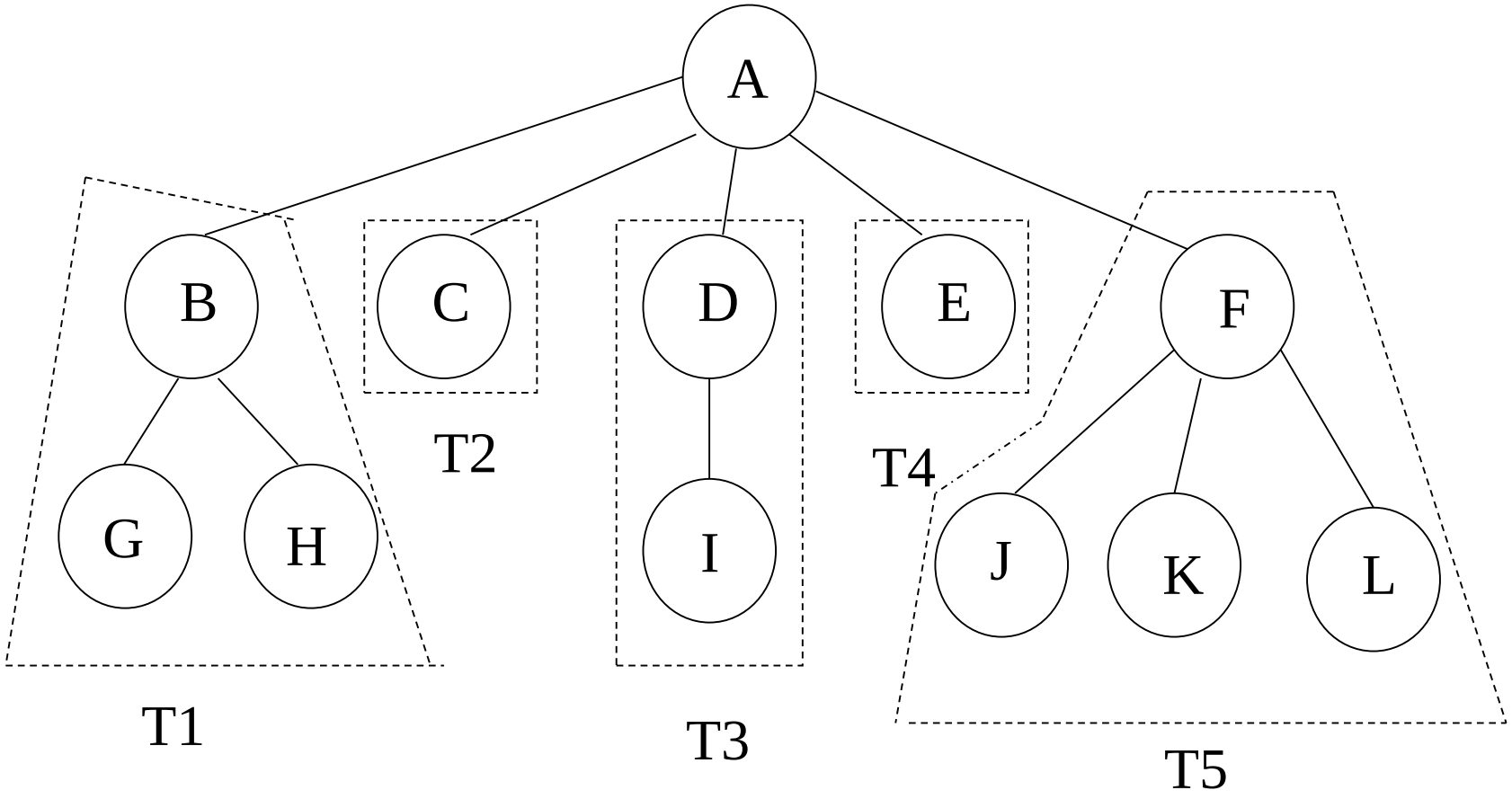
E : μήκος εξωτερικής διαδρομής = άθροισμα επιπέδου κάθε τερματικού κόμβου (=29)

Σε ένα κανονικό m-αδικό δένδρο ισχύει $E = (m-1)I + m\mathbf{i}$

Διάσχιση Δέντρου

- **Η διάσχιση (traversal)** ενός δέντρου είναι η διαδικασία επίσκεψης όλων των κόμβων ενός δέντρου (αρχίζοντας από τη ρίζα).
  - Βασική λειτουργία όταν π.χ. θέλουμε να βρούμε έναν κόμβο με κάποια συγκεκριμένα χαρακτηριστικά και να τυπώσουμε τα στοιχεία του.
- Υπάρχουν διάφοροι τρόποι διάσχισης ενός δέντρου (δηλ. διάφοροι τρόποι να προσδιοριστεί η σειρά εκτύπωσης των στοιχείων των κόμβων).
  - **depth first:**
    - **pre-order,**
    - **post-order,**
    - **in-order.**
  - **breadth first.**

Διάσχιση Δέντρου - Παράδειγμα

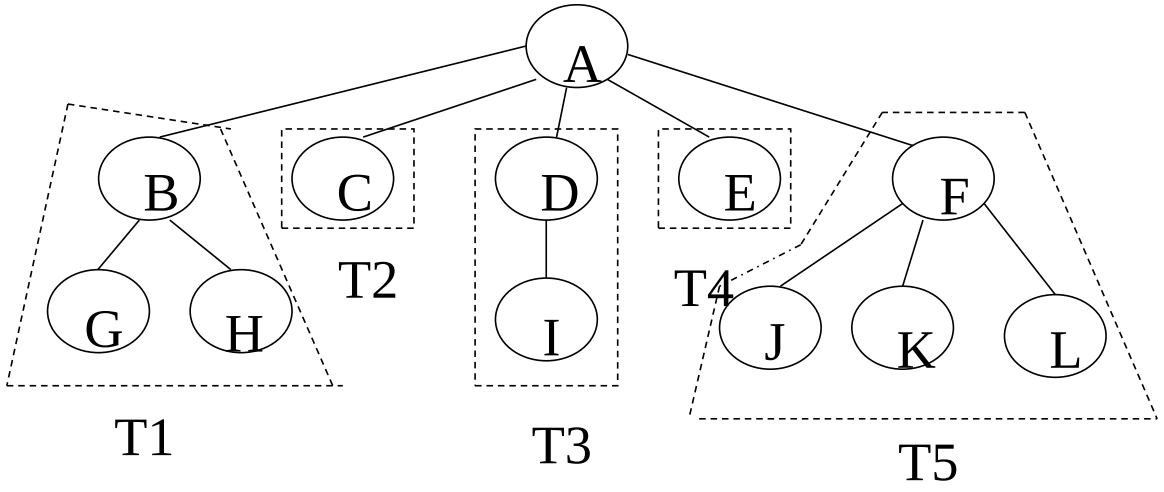


Pre-order (προδιάταξη)

- **Αλγόριθμος:**
  1. Επισκέψου τον τρέχοντα κόμβο.
  2. Κάνε ένα pre-order traversal των υποδέντρων κάτω από αυτόν τον κόμβο από αριστερά προς δεξιά.
- Ο αλγόριθμος είναι αναδρομικός μια και τα υποδέντρα του τρέχοντος κόμβου έχουν κι αυτά υποδέντρα κ.ο.κ.
  - περισσότερα για αναδρομικούς αλγόριθμους αργότερα.
- Η διάσχιση με προδιάταξη είναι ισοδύναμη με τη γνωστή κατά βάθος διάσχιση του δέντρου.

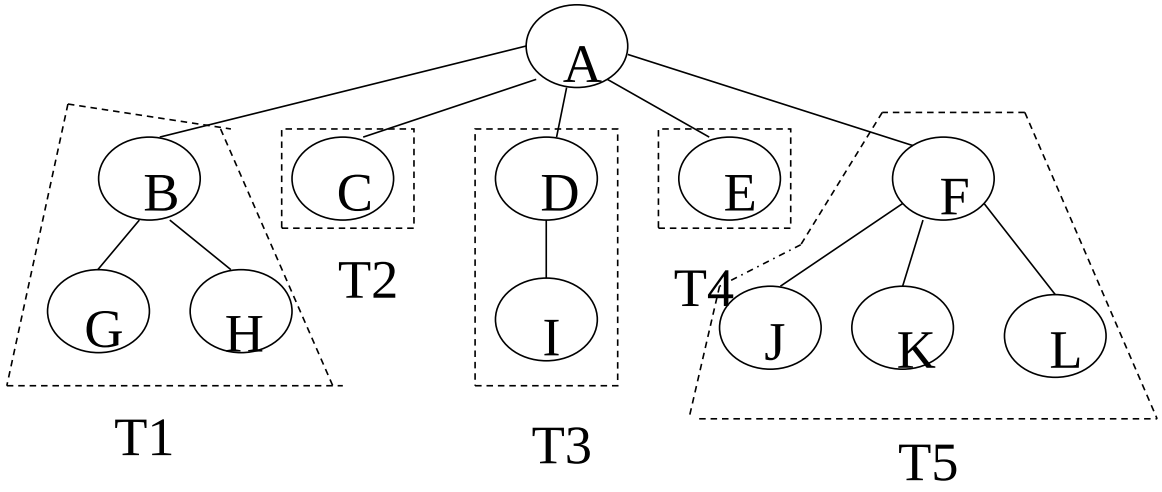
Pre-order (παράδειγμα) (1/5)

- Αρχίζουμε στη ρίζα Α.
  - Κόμβοι που επισκεφθήκαμε: Α.
- Ο κόμβος Α έχει 5 υποδέντρα. Αρχίζουμε με το πιο αριστερό Τ1.
- Η ρίζα του υποδέντρου Τ1 είναι ο κόμβος Β, οπότε επισκεπτόμαστε τον Β.
  - Κόμβοι που επισκεφθήκαμε: Α Β.



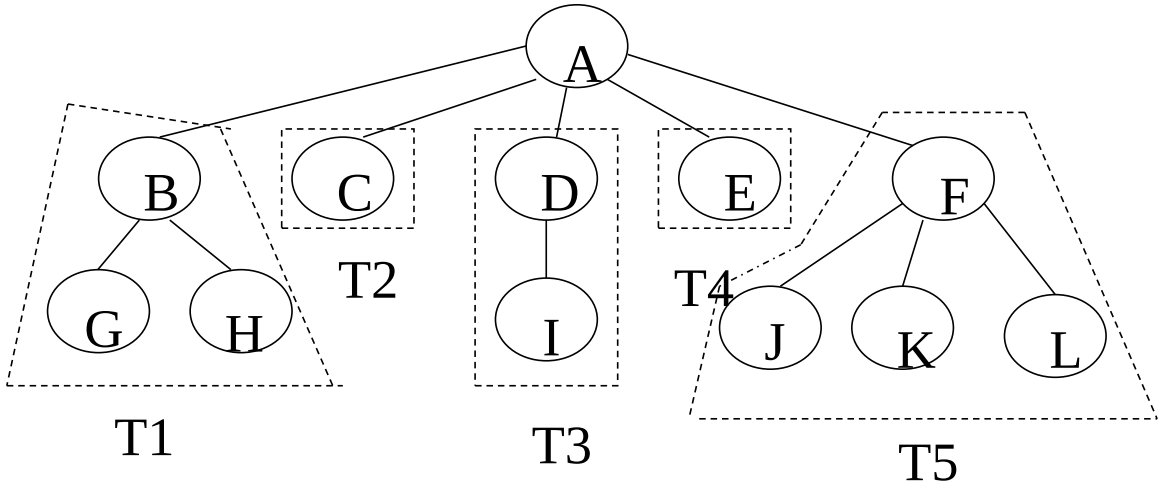
Pre-order (παράδειγμα) (2/5)

- Ο Β έχει δύο υποδέντρα που περιέχουν τους κόμβους G and H. Πρώτα επισκεπτόμαστε τον G γιατί είναι πιο αριστερά.
  - Κόμβοι που επισκεφθήκαμε: A B G
- Ο G δεν έχει υποδέντρα, οπότε πηγαίνουμε στο δεξί υποδέντρο κάτω από το Β που έχει τον κόμβο H.
  - Κόμβοι που επισκεφθήκαμε: A B G H



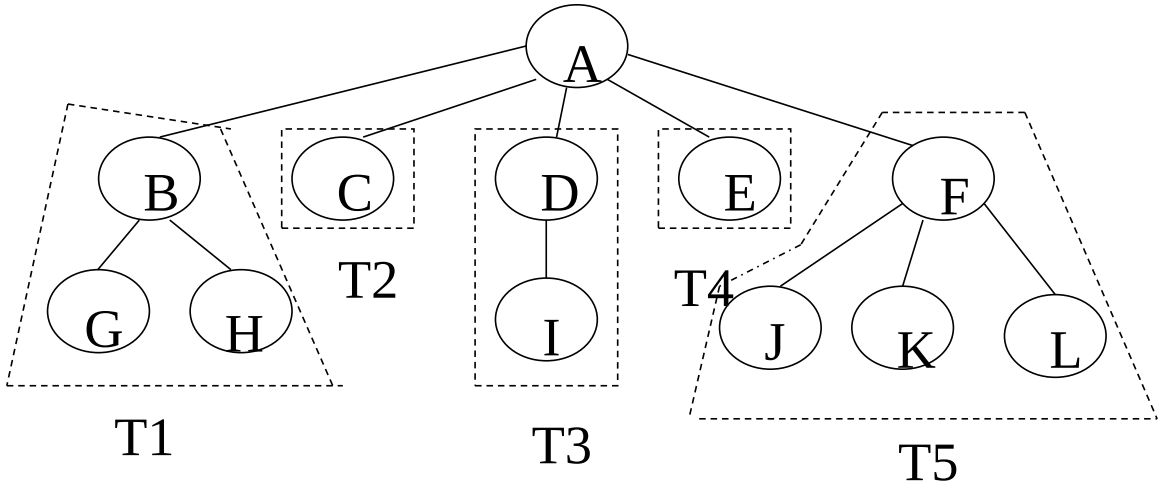
Pre-order (παράδειγμα) (3/5)

- Ο Η δεν έχει υποδέντρα. Τελειώσαμε με το υποδέντρο Τ1 κάτω από το Α και τώρα προχωράμε στο υποδέντρο Τ2.
- Το Τ2 έχει μόνο έναν κόμβο, τον C. Επισκεπτόμαστε τον C.
  - Κόμβοι που επισκεφθήκαμε: A B G H C
- Ο C δεν έχει υποδέντρα, οπότε τελειώσαμε με το υποδέντρο Τ2 και μετακινούμαστε στο υποδέντρο Τ3.



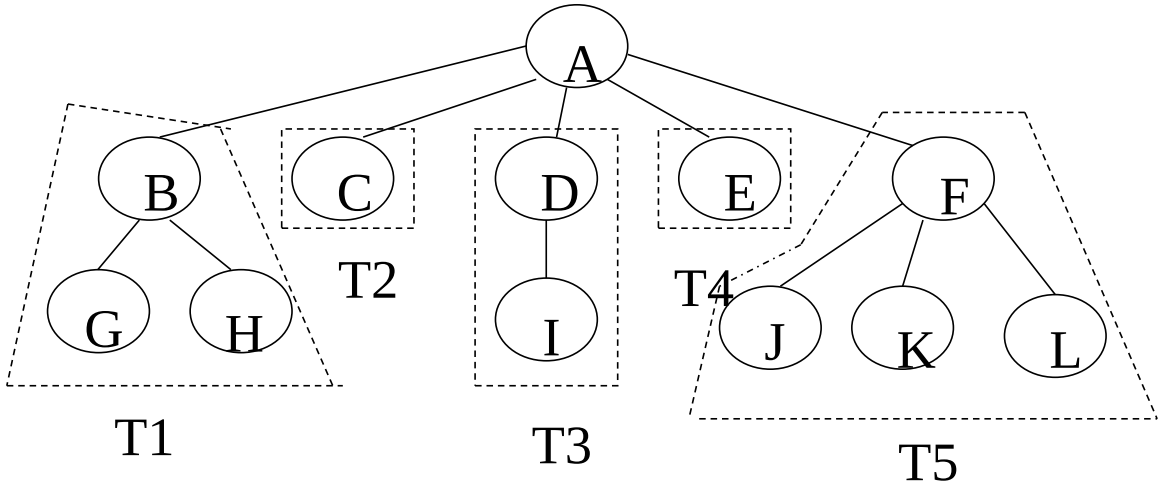
Pre-order (παράδειγμα) (4/5)

- Το pre-order traversal του T3 δίνει D και I.
  - Κόμβοι που επισκεφθήκαμε: A B G H C D I
- Το pre-order traversal του υποδέντρου T4 δίνει E.
  - Κόμβοι που επισκεφθήκαμε: A B G H C D I E



Pre-order (παράδειγμα) (5/5)

- Τέλος, κάνουμε ένα pre-order traversal του υποδέντρου T5 που δίνει F, J , K , L
  - Κόμβοι που επισκεφθήκαμε: A B G H C D I E F J K L
- Τελειώσαμε με όλα τα υποδέντρα κάτω από τη ρίζα A και ο αλγόριθμος τερματίζει.



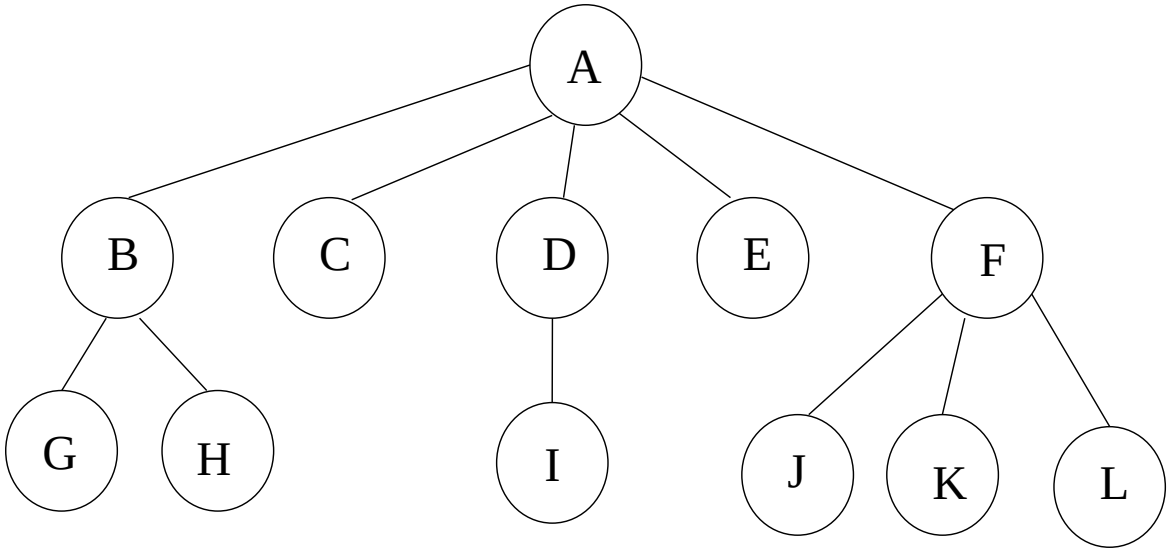
Post-order (μεταδιάταξη)

- **Αλγόριθμος:**
  1. Κάνε ένα post-order traversal των υποδέντρων κάτω από τον τρέχοντα κόμβο από αριστερά προς τα δεξιά.
  2. Επισκέψου τον τρέχοντα κόμβο.

- Παρόμοια με την pre-order traversal. Μόνο που η σειρά επίσκεψης του τρέχοντος κόμβου και των υποδέντρων του είναι αντίστροφη.

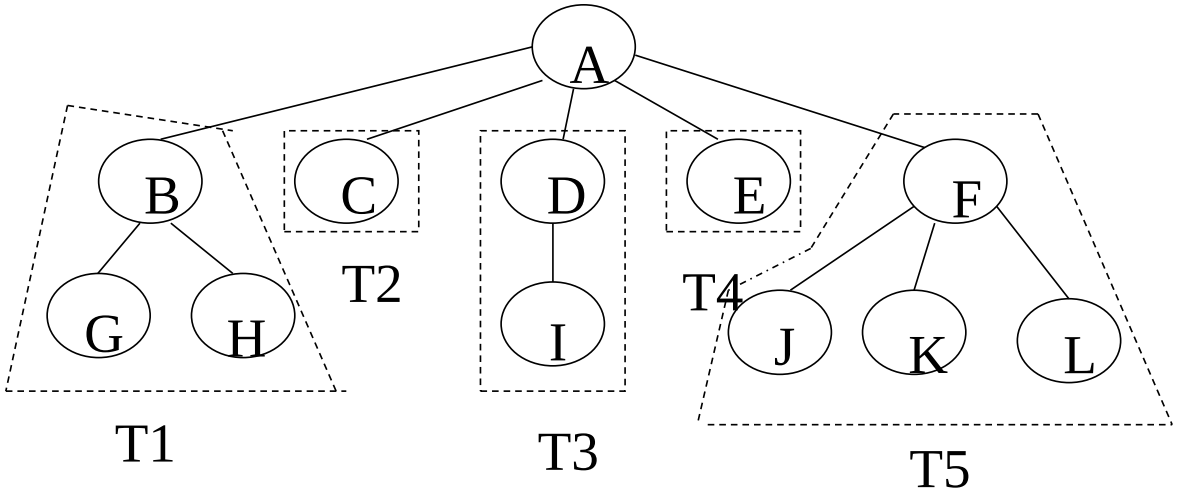
Post-order (παράδειγμα) (1/6)

- Θα κάνουμε post-order διάσχιση του παρακάτω δέντρου.



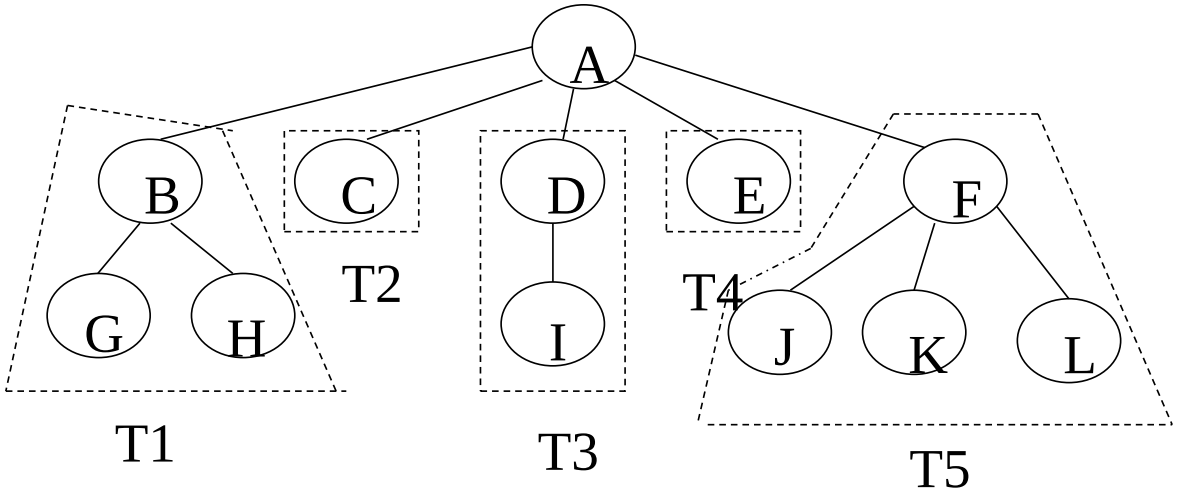
Post-order (παράδειγμα) (2/6)

- Πρώτα κάνουμε post-order διάσχιση του πιο αριστερού υποδέντρου της ρίζας Α.
  - Κόμβοι που επισκεφθήκαμε: -
- Ο κόμβος Β έχει 2 υποδέντρα. Κάνουμε post-order διάσχιση πιο αριστερού.
- Ο κόμβος G δεν έχει υποδέντρα, οπότε τον επισκεπτόμαστε.
  - Κόμβοι που επισκεφθήκαμε: G



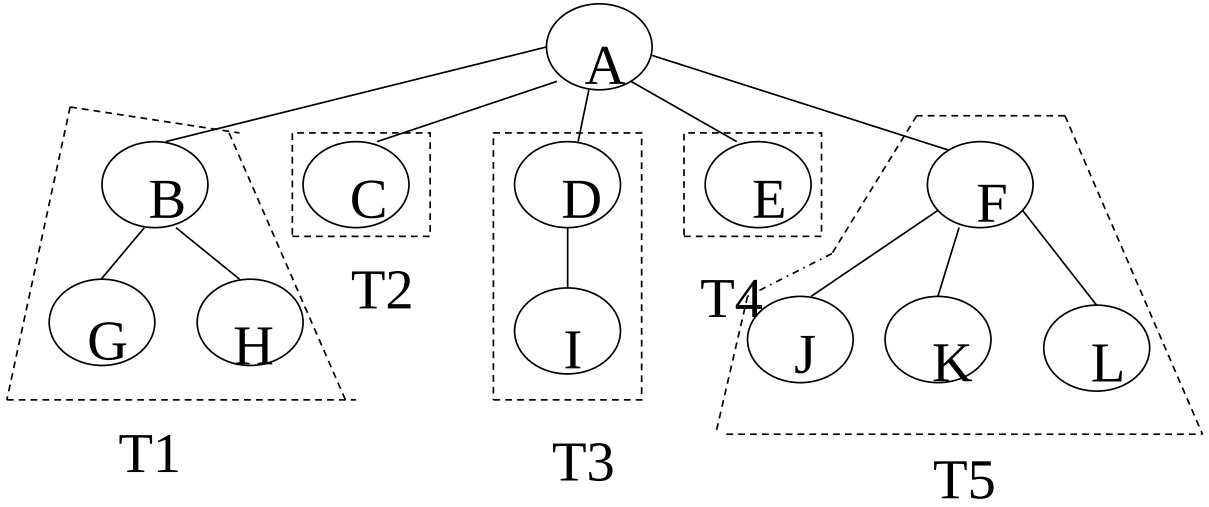
Post-order (παράδειγμα) (3/6)

- Κάνουμε τώρα post-order διάσχιση του δεξιού υποδέντρου του B.
- Ο H δεν έχει υποδέντρα, οπότε τον επισκεπτόμαστε.
  - Κόμβοι που επισκεφθήκαμε: G H



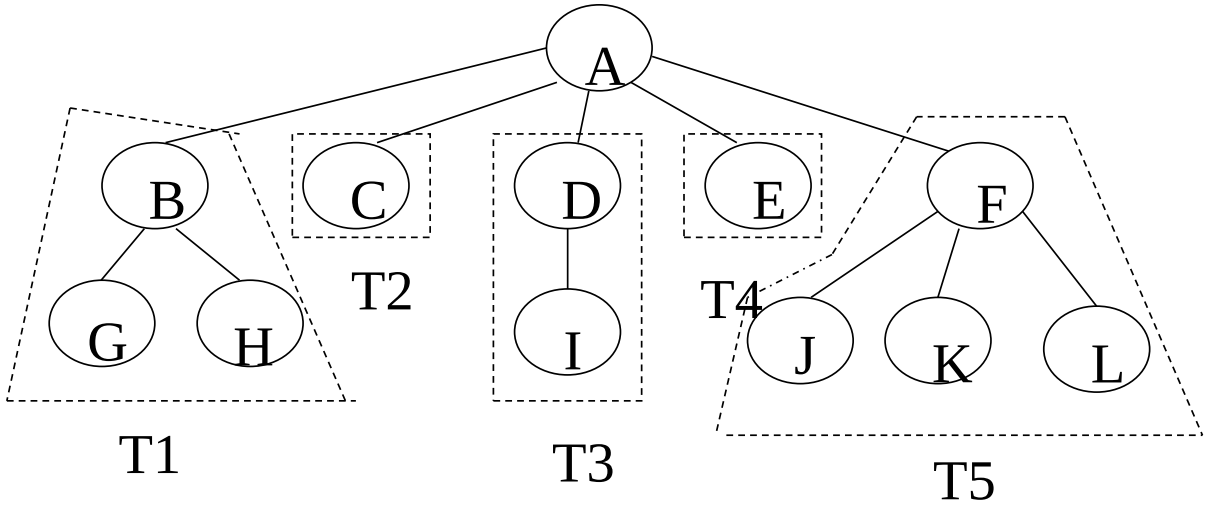
Post-order (παράδειγμα) (4/6)

- Επισκεπτόμαστε τώρα τον B.
  - Κόμβοι που επισκεφθήκαμε: G H B
- Κάνουμε τώρα post-order διάσχιση του επόμενου υποδέντρου του A. Το C δεν έχει υποδέντρα. Επισκεπτόμαστε τον C
  - Κόμβοι που επισκεφθήκαμε: G H B C



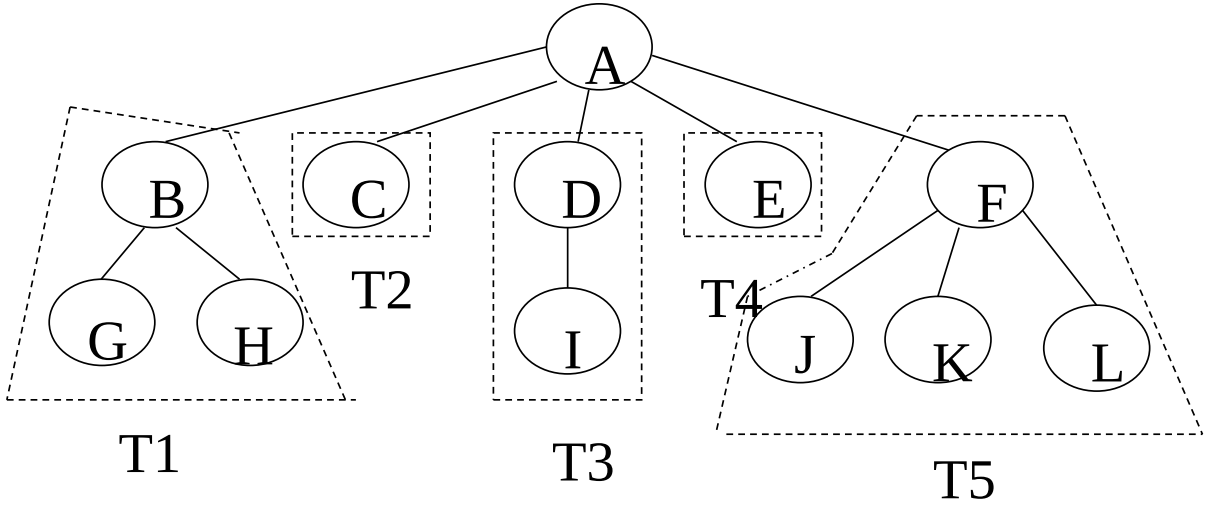
Post-order (παράδειγμα) (5/6)

- Κάνουμε τώρα post-order διάσχιση του επόμενου υποδέντρου του A. Το post-order traversal του T3 δίνει I και D.
  - Κόμβοι που επισκεφθήκαμε: G H B C I D
- Το post-order traversal του υποδέντρου T4 δίνει E
  - Κόμβοι που επισκεφθήκαμε: G H B C I D E



Post-order (παράδειγμα) (6/6)

- Τέλος, κάνουμε ένα post-order traversal του υποδέντρου T5 που δίνει J , K , L , F
  - Κόμβοι που επισκεφθήκαμε: G H B C I D E J K L F A
- Τελειώσαμε και με τη ρίζα A και ο αλγόριθμος τερματίζει.



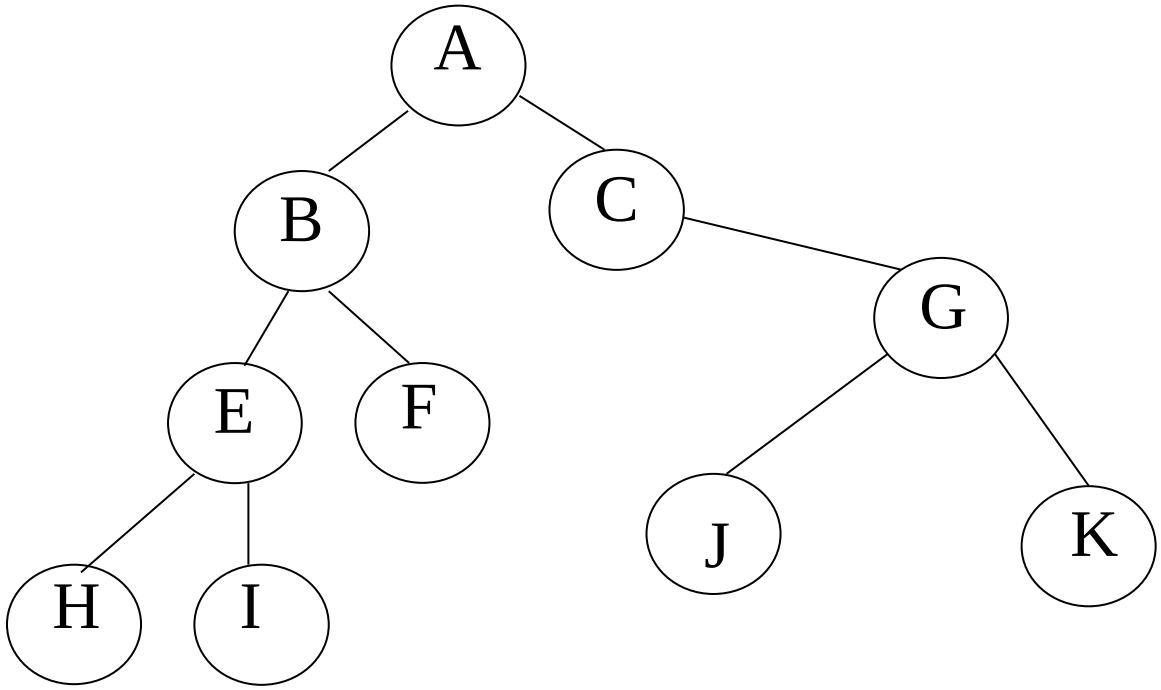
In-order (ενδοδιάταξη)

- **Αλγόριθμος:**
  1. Κάνε ένα in-order traversal του αριστερού υποδέντρου κάτω από τον τρέχοντα κόμβο.
  2. Επισκέψου τον τρέχοντα κόμβο.
  3. Κάνε ένα in-order traversal του δεξιού υποδέντρου κάτω από τον τρέχοντα κόμβο.

Ο αλγόριθμος in-order εφαρμόζεται μόνο σε δυαδικά δέντρα!

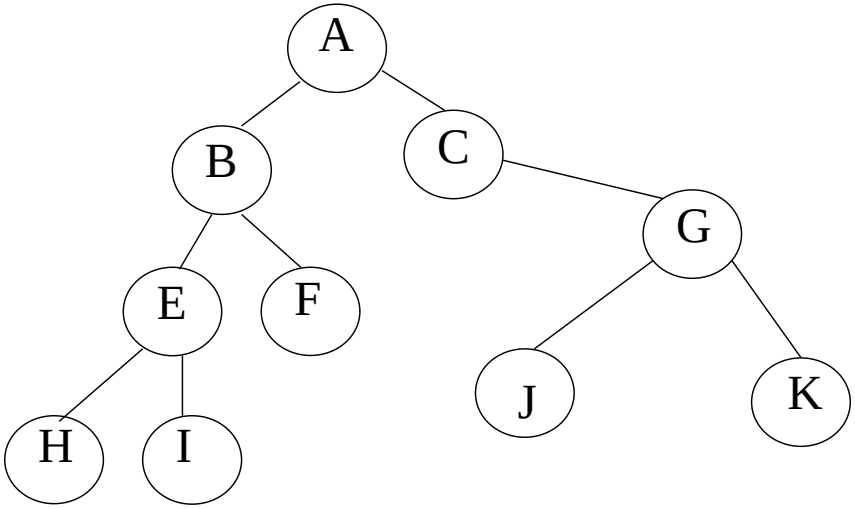
In-order (παράδειγμα) (1/7)

- Θα κάνουμε in-order διάσχιση στο παρακάτω δέντρο:



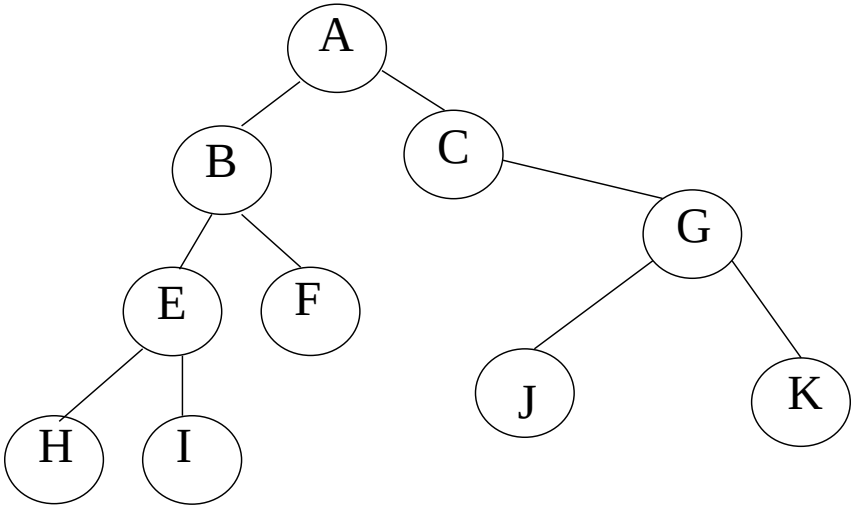
In-order (παράδειγμα) (2/7)

- Πρώτα κάνουμε in-order διάσχιση του αριστερού υποδέντρου της ρίζας Α.
  - Κόμβοι που επισκεφθήκαμε: -
- Κάνουμε in-order διάσχιση του αριστερού υποδέντρου του Β.
- Κάνουμε in-order διάσχιση του αριστερού υποδέντρου του Ε.



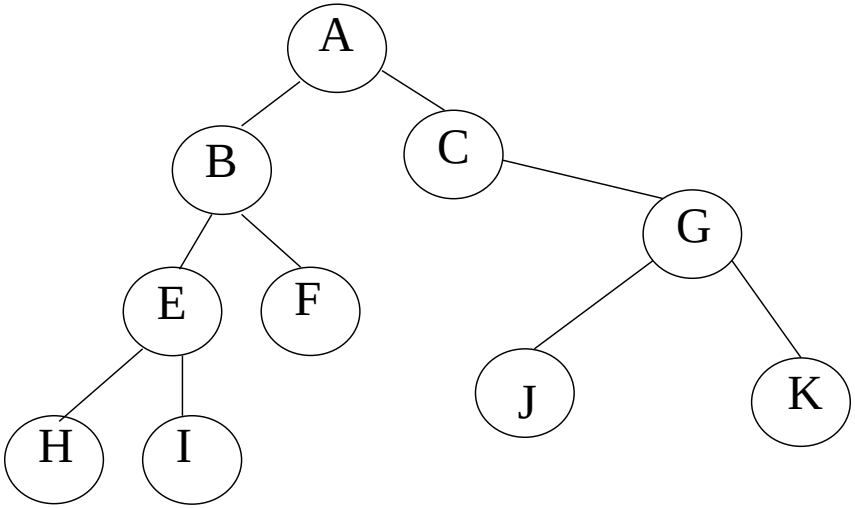
In-order (παράδειγμα) (3/7)

- Ο Η δεν έχει υποδέντρα, οπότε τον επισκεπτόμαστε.
  - Κόμβοι που επισκεφθήκαμε: Η
- Επισκεπτόμαστε τώρα τον κόμβο Ε.
  - Κόμβοι που επισκεφθήκαμε: Η Ε
- Κάνουμε in-order διάσχιση του δεξιού υποδέντρου του Ε.



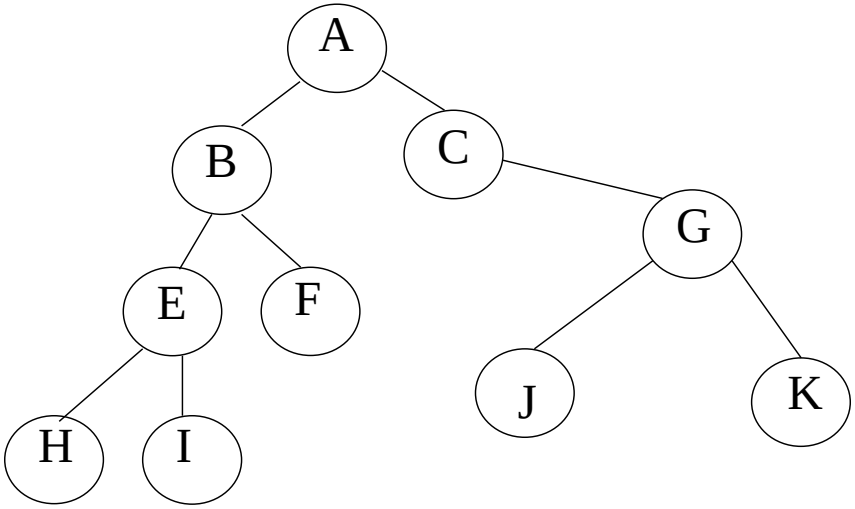
In-order (παράδειγμα) (4/7)

- Ο Ι δεν έχει υποδέντρα, οπότε τον επισκεπτόμαστε.
  - Κόμβοι που επισκεφθήκαμε: Η Ε Ι
- Επισκεπτόμαστε τώρα τον κόμβο Β.
  - Κόμβοι που επισκεφθήκαμε: Η Ε Ι Β
- Κάνουμε in-order διάσχιση του δεξιού υποδέντρου του Β.



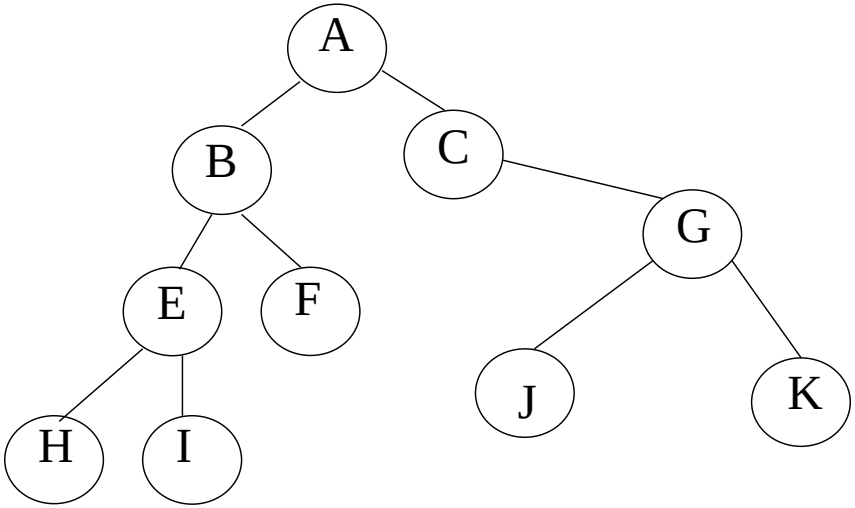
In-order (παράδειγμα) (5/7)

- Ο F δεν έχει υποδέντρα, οπότε τον επισκεπτόμαστε.
  - Κόμβοι που επισκεφθήκαμε: H E I B F
- Επισκεπτόμαστε τώρα τον κόμβο A.
  - Κόμβοι που επισκεφθήκαμε: H E I B F A



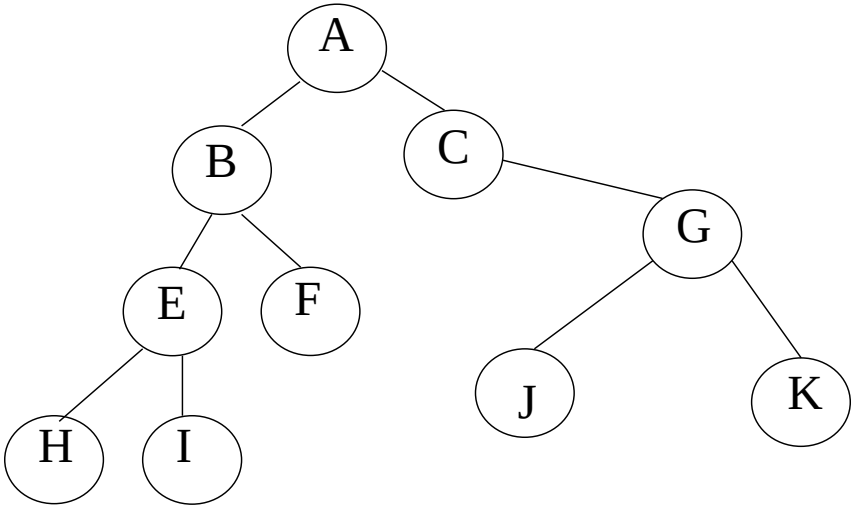
In-order (παράδειγμα) (6/7)

- Τώρα κάνουμε in-order διάσχιση του δεξιού υποδέντρου της ρίζας A.
- Ο κόμβος C δεν έχει δεξιό υποδέντρο, οπότε τον επισκεπτόμαστε.
  - Κόμβοι που επισκεφθήκαμε: H E I B F A C



In-order (παράδειγμα) (7/7)

- Η in-order διάσχιση του αριστερού υποδέντρου του C δίνει J G K
  - Κόμβοι που επισκεφθήκαμε: H E I B F A C J G K
- Τελειώσαμε έχοντας επισκεφτεί όλους τους κόμβους.

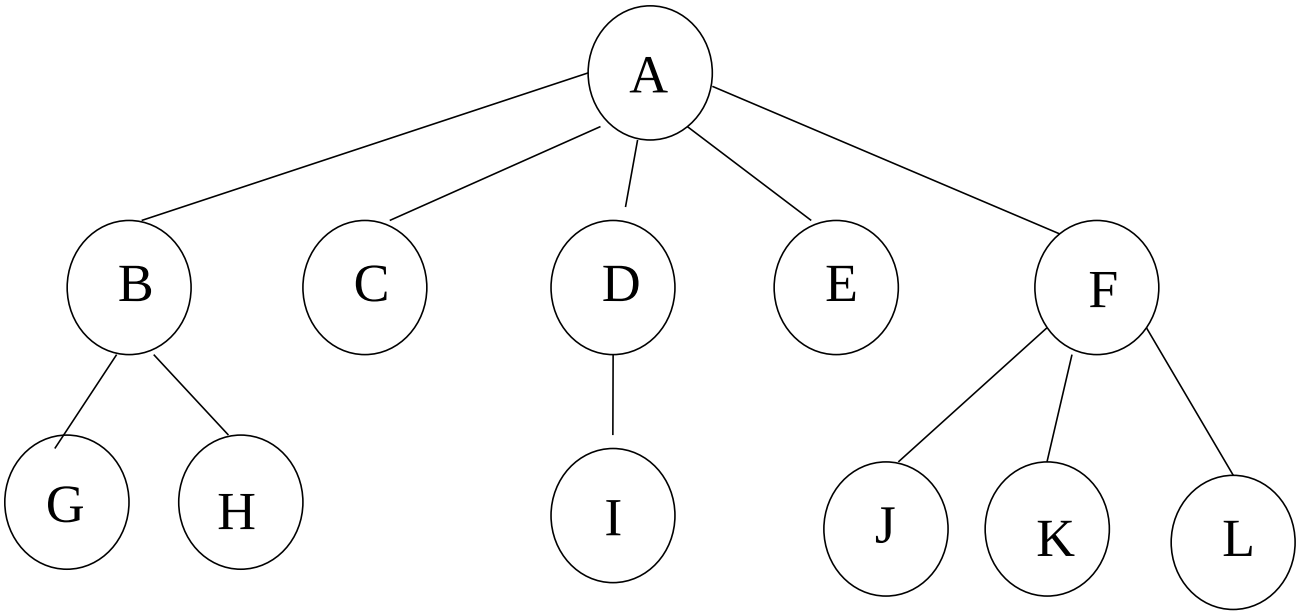


Αναζήτηση Πρώτα κατά Πλάτος (Breadth-First Search)

- Η **Breadth-First Search (BFS)** διασχίζει ένα **συνδεδεμένο κομμάτι** ενός γράφου.
- Η BFS σε ένα **δέντρο** G λειτουργεί ως εξής:
  - Στον αρχικό κόμβο (ρίζα) *s,* δίνεται η απόσταση 0.
  - Στην πρώτη επανάληψη, ο αλγόριθμος επισκέπτεται όλους τους κόμβους που είναι παιδιά του αρχικού. Σε αυτούς δίνεται απόσταση 1.
  - Στη συνέχεια η BFS επισκέπτεται όλους τους κόμβους που απέχουν απόσταση 2 από τον αρχικό, κ.ο.κ.

Breadth First (κατά πλάτος)

- Η breadth-first διάσχιση του παρακάτω δέντρου δίνει:
  A B C D E F G H I J K L



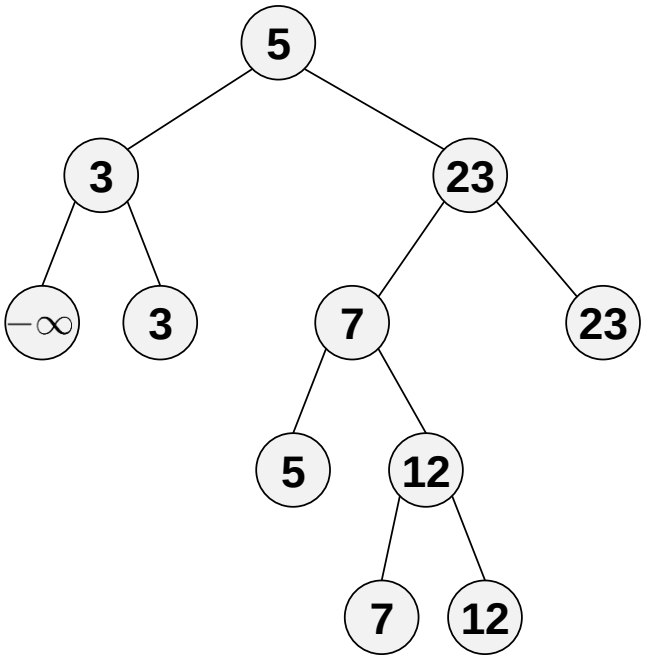
Δυαδικό δένδρο αναζήτησης (Binary Search Tree – BST)

- Χρησιμοποιούνται για την αναπαράσταση ενός συνόλου αντικειμένων τα οποία είναι **ολικά διατεταγμένα** με βάση κάποιο **κλειδί**.
- Τα κλειδιά ανήκουν σε ένα ολικά διατεταγμένο σύνολο (π.χ. ακέραιοι αριθμοί).
- Θέλουμε μια δομή που να επιτρέπει τη γρήγορη αναζήτηση ενός αντικειμένου όταν δίνεται το κλειδί του αντικειμένου.
- Για το σκοπό αυτό μπορούμε να χρησιμοποιήσουμε ένα δυαδικό δένδρο όπου τα κλειδιά αποθηκεύονται στους κόμβους του δένδρου.

Δυαδικό δένδρο αναζήτησης (1/11)

Έστω ότι έχουμε κλειδιά $\kappa_1, \kappa_2, \ldots, \kappa_n$ όπου $\kappa_1 \leq \kappa_2 \leq \ldots \leq \kappa_n$.
Κάθε κλειδί αποθηκεύεται σε ένα εσωτερικό κόμβο και ένα φύλλο.
Επιπλέον στο αριστερότερο φύλλο μπορεί να αποθηκεύουμε ένα κλειδί $\kappa_0 < \kappa_1$

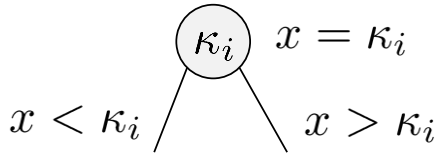
Π.χ. για τα κλειδιά 3,5,7,12 και 23 μπορούμε να έχουμε το δένδρο:



Αν ένας εσωτερικός κόμβος έχει κλειδι κ_i τότε
το αριστερό υποδένδρο του έχει κλειδιά $< \kappa_i$
και το δεξί υποδένδρο του έχει κλειδιά $\geq \kappa_i$

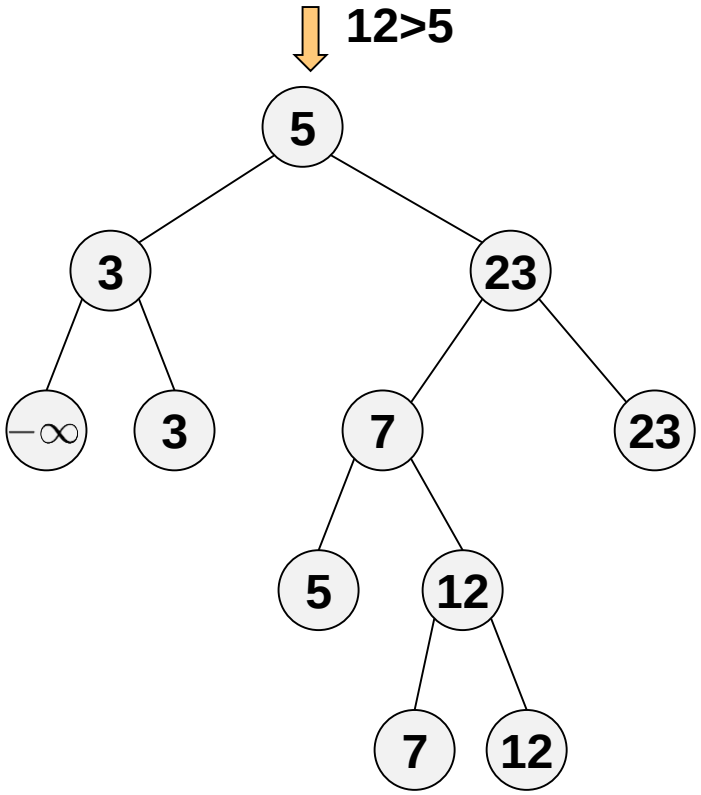
BST ιδιότητα

Η αναζήτηση ενός κλειδιού x ξεκινάει από τη ρίζα και κατευθύνεται προς τα κάτω.



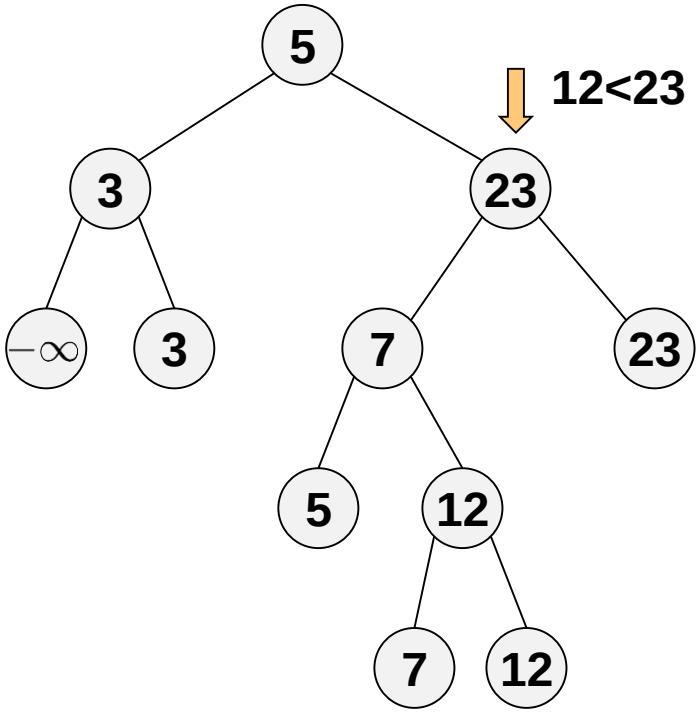
Δυαδικό δένδρο αναζήτησης (2/11)

Αναζήτηση x=12



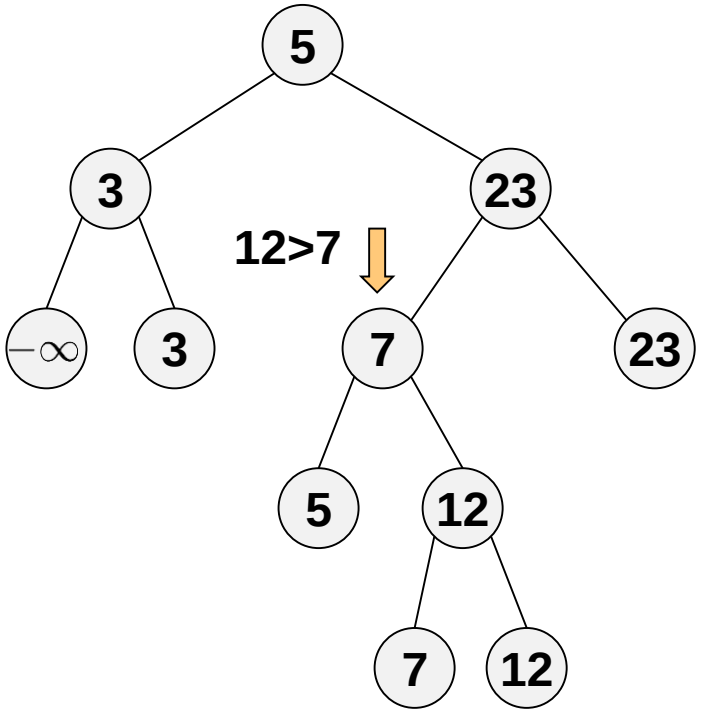
Δυαδικό δένδρο αναζήτησης (3/11)

Αναζήτηση x=12



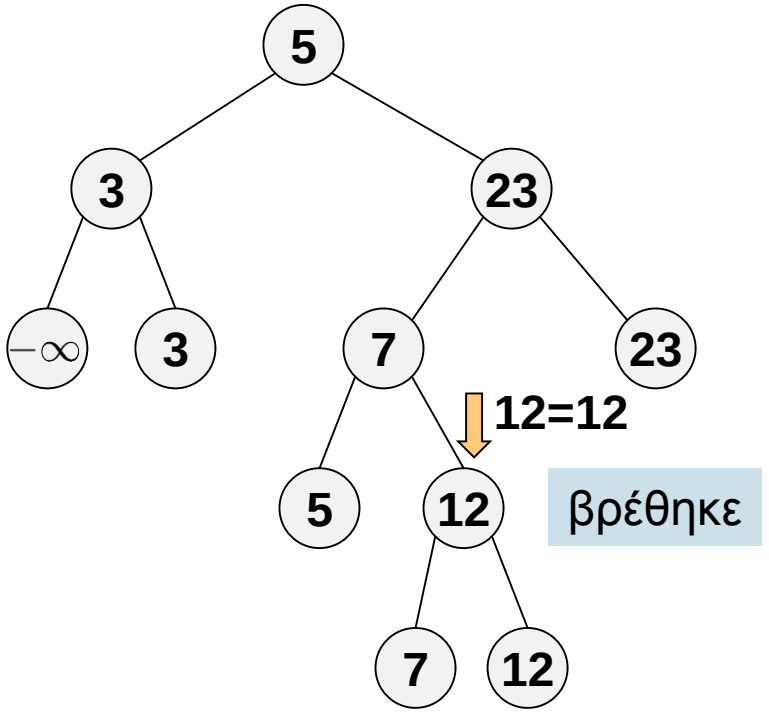
Δυαδικό δένδρο αναζήτησης (4/11)

Αναζήτηση x=12



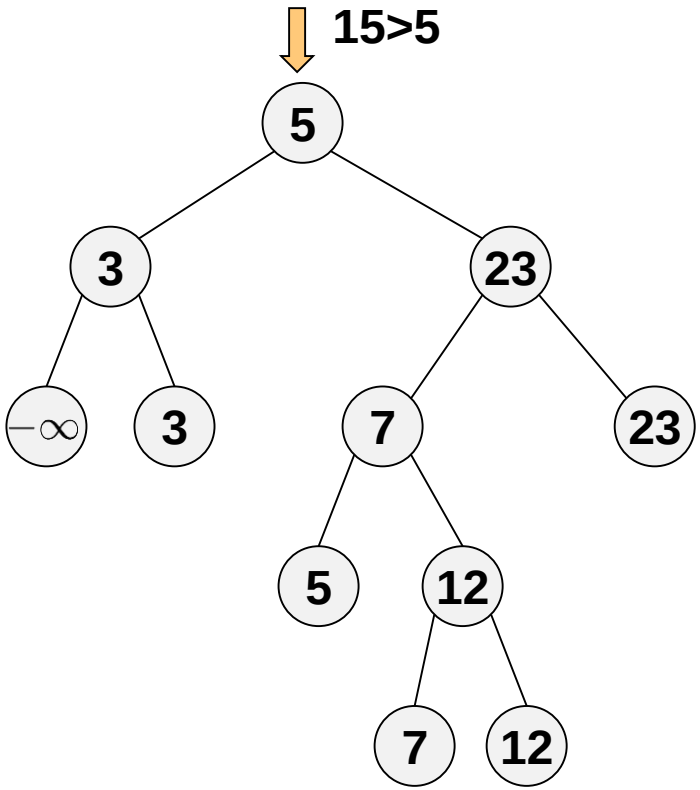
Δυαδικό δένδρο αναζήτησης (5/11)

Αναζήτηση x=12



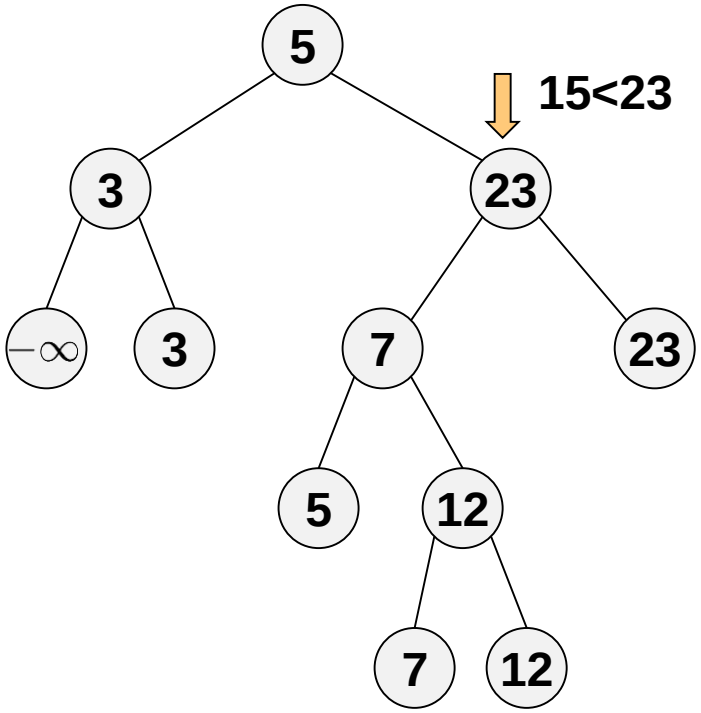
Δυαδικό δένδρο αναζήτησης (6/11)

Αναζήτηση x=15



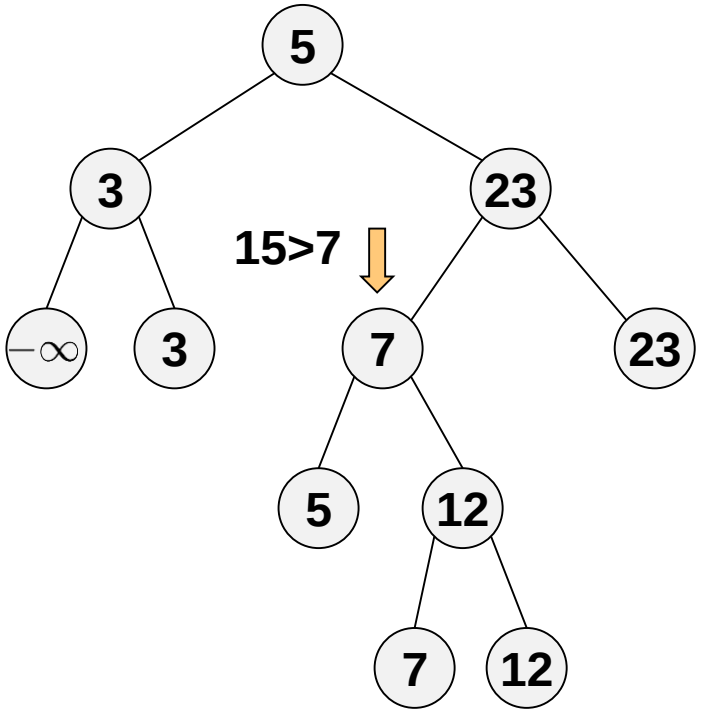
Δυαδικό δένδρο αναζήτησης (7/11)

Αναζήτηση x=15



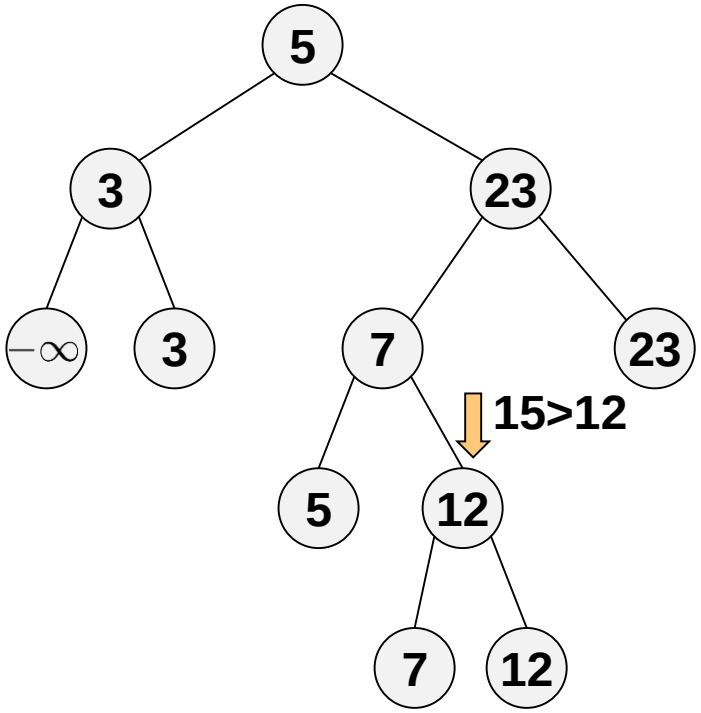
Δυαδικό δένδρο αναζήτησης (8/11)

Αναζήτηση x=15



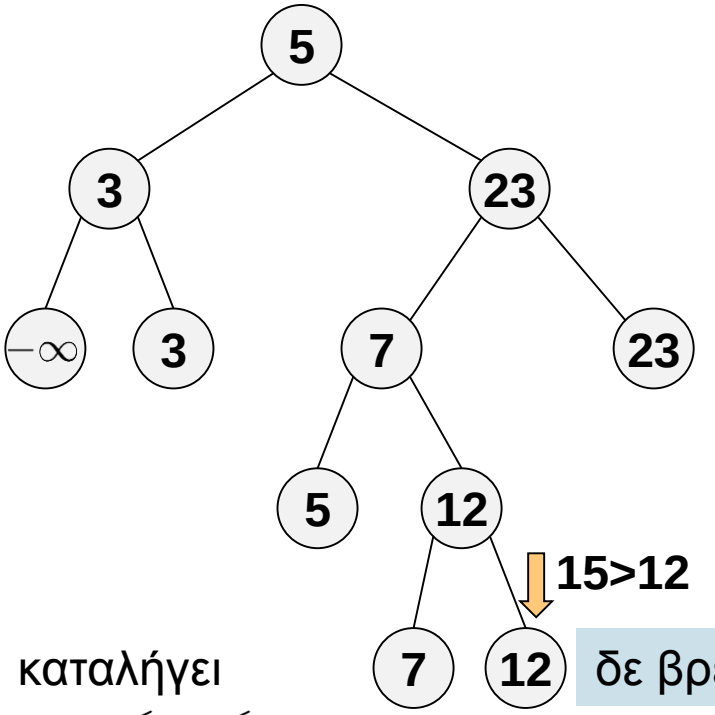
Δυαδικό δένδρο αναζήτησης (9/11)

Αναζήτηση x=15



Δυαδικό δένδρο αναζήτησης (10/11)

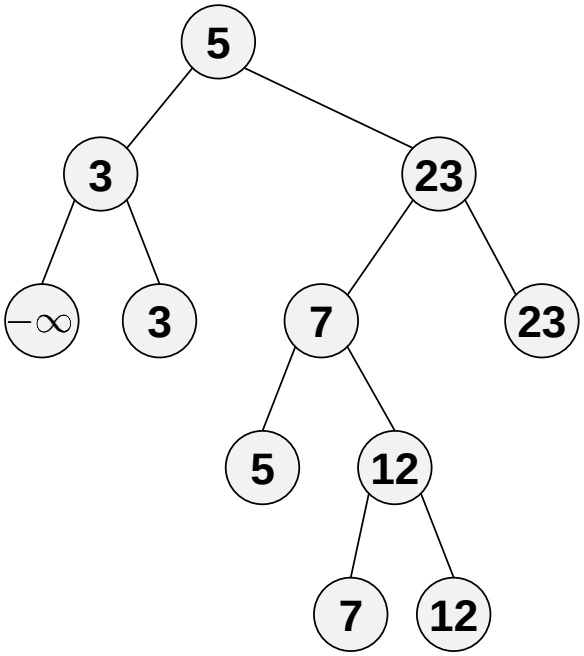
Αναζήτηση x=15

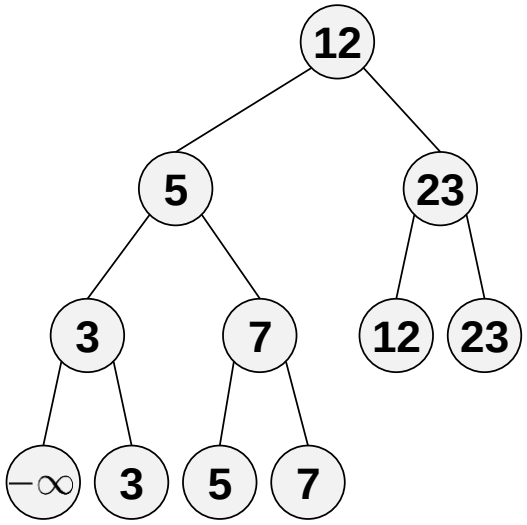


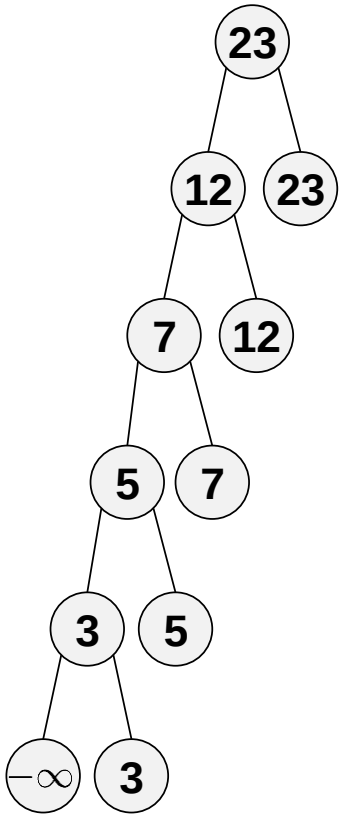
Όταν η αναζήτηση του x καταλήγει σε φύλλο με κλειδι κ_i τότε $\kappa_i < x < \kappa_{i+1}$

Δυαδικό δένδρο αναζήτησης (11/11)

Υπάρχουν διάφορετικά δυαδικά δένδρα αναζήτησης για δεδομένο σύνολο κλειδιών







Ποιό είναι καλύτερο;

Βασική Δομή Δεδομένων Δέντρου Αναζήτησης (1/2)

- Η βασική δομή δεδομένων σε ένα BST είναι ένας κόμβος που ονομάζουμε BinaryNode.
- Αντίθετα με τα γενικά δέντρα όπου το πλήθος των παιδιών κάθε κόμβου δεν είναι γνωστό στατικά, ένας BST κόμβος μπορεί να έχει το πολύ δύο παιδιά.
- Οπότε, είναι λογικό να χρησιμοποιήσουμε δύο δείκτες lchild (left-child) και rchild (right-child).
  - Υποθέτουμε ότι οι δείκτες των φύλλων είναι NULL.

Βασική Δομή Δεδομένων Δέντρου Αναζήτησης (2/2)

```
class BinaryNode {
private:
  int key; // the integer key
  BinaryNode *lchild; // pointer to left
  child
  BinaryNode *rchild; // pointer to right
  child
public:
// public methods go here
}
```

Λειτουργίες σε Δέντρα Αναζήτησης

- Οι βασικές λειτουργίες σε ένα BST είναι:
  - **findMin:** βρες το ελάχιστο κλειδί μέσα στο BST.
  - **findMax**: βρες το μέγιστο κλειδί μέσα στο BST.
  - **find(x):** βρες το κλειδί ‘x’ μέσα στο BST.
  - **insert(x):** εισήγαγε το κλειδί ‘x’ μέσα στο BST.
  - **remove(x):** διέγραψε το κλειδί ‘x’ από το BST.
  - **isEmpty:** δες αν το BST είναι άδειο.
  - **makeEmpty:** άδειασε το BST.
  - **printTree:** τύπωσε όλα τα κλειδιά στο BST.

findMin (1/2)

- Σε ένα δέντρο όλες οι διασχίσεις αρχίζουν από τη ρίζα.
- Σύμφωνα με την BST ιδιότητα, το κλειδί του lchild της ρίζας θα είναι < του κλειδιού της ρίζας.
- Το κλειδί του lchild του lchild της ρίζας θα είναι < του κλειδιού του lchild της ρίζας κ.ο.κ.
- Αν μια διάσχιση ενός BST αρχίσει στη ρίζα και «πηγαίνει συνέχεια αριστερά» καθώς κατεβαίνει προς τα κάτω, ο τελευταίος κόμβος στη διάσχιση θα έχει το μικρότερο κλειδί στο δέντρο.

findMin (2/2)

- Η παράμετρος εισόδου στη findMin μέθοδο είναι η ρίζα του BST.
  - findMin(root);
- Η findMin μέθοδος επιστρέφει τον κόμβο που περιέχει το μικρότερο κλειδί στο BST.
- Έτσι, ο τύπος που επιστρέφει η findMin είναι BinaryNode*.

findMin Algorithm

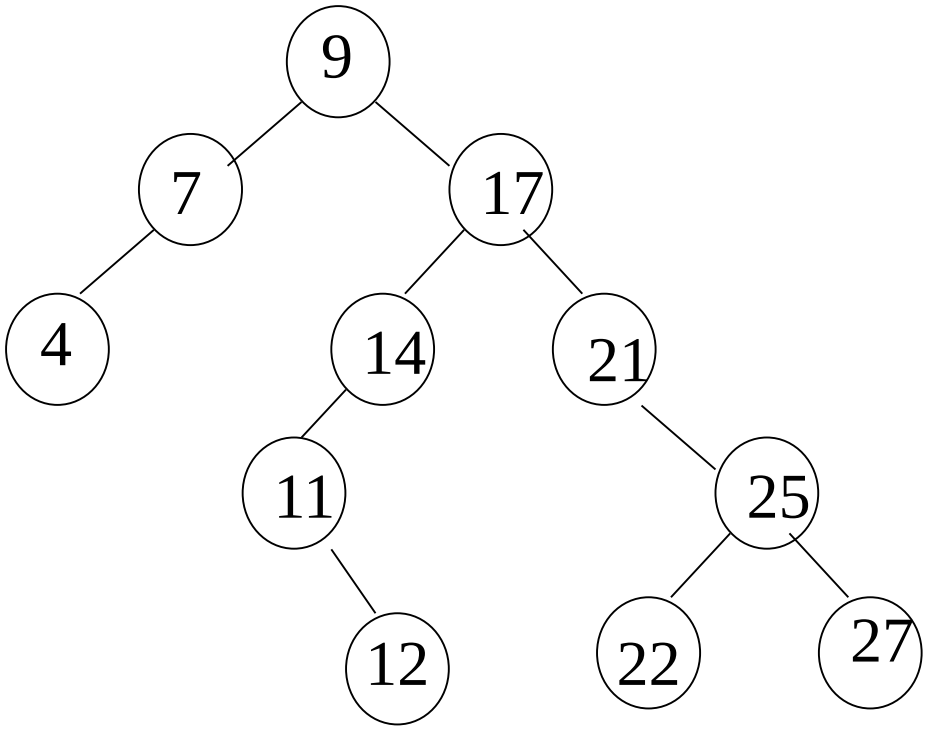
- Βήματα αλγορίθμου:
  - Ξεκίνα από τη ρίζα.
  - Αν η ρίζα είναι NULL (άδειο BST) επέστρεψε NULL.
  - Σε κάθε κόμβο πήγαινε στο lchild.
  - Ολοκλήρωσε τη διάσχιση όταν συναντηθεί ο πρώτος NULL δείκτης.
  - Επέστρεψε τον τελευταίο κόμβο που επισκέφτηκες.

- Ο αλγόριθμος μπορεί να υλοποιηθεί αναδρομικά ή μη-αναδρομικά.

findMin Algorithm (χωρίς αναδρομή)

```
// Non-recursive algorithm
BinaryNode* findMin(BinaryNode *t)
begin
  // έλεγξε αν το δέντρο είναι άδειο
  if (t!=NULL)
  begin
    // προχώρα κάτω προς το lchild σε κάθε
  κόμβο
    while (t->lchild!=NULL)
          t = t->lchild;
  end
  return t;
end
```

findMin - Παράδειγμα



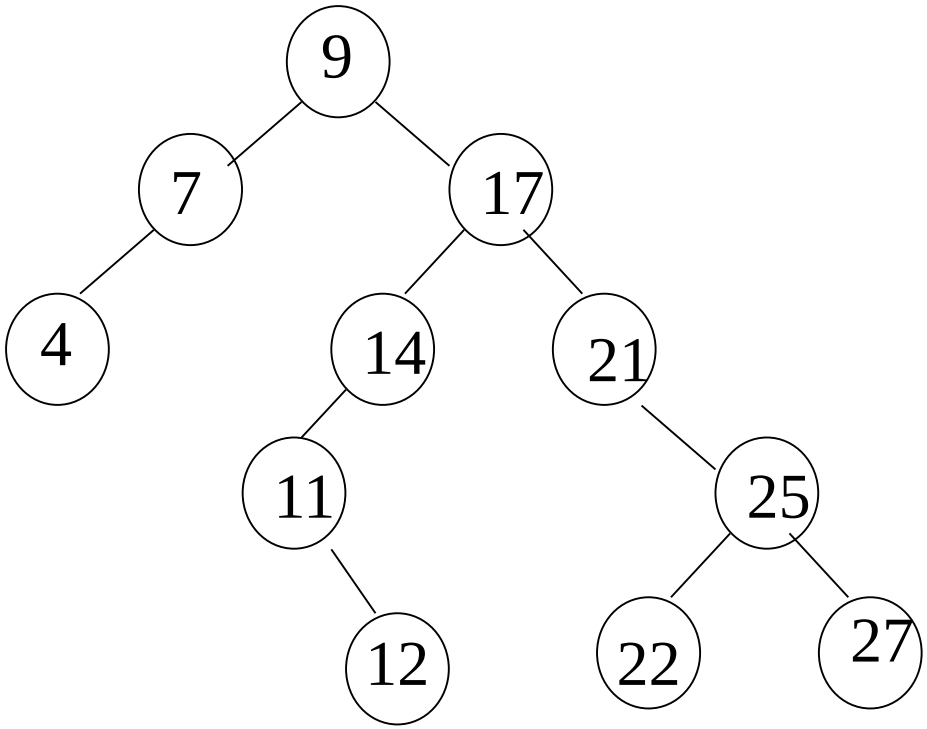
findMax

- **Βήματα αλγορίθμου:**
  - Ξεκίνα από τη ρίζα.
  - Αν η ρίζα είναι NULL (άδειο BST) επέστρεψε NULL.
  - Σε κάθε κόμβο πήγαινε στο rchild.
  - Ολοκλήρωσε τη διάσχιση όταν συναντηθεί ο πρώτος NULL δείκτης.
  - Επέστρεψε τον τελευταίο κόμβο που επισκέφτηκες.
- Η παράμετρος εισόδου στη findMax και ο τύπος που επιστρέφει είναι ίδια με τη findMin.

findMax Algorithm (χωρίς αναδρομή)

```
// Non-recursive algorithm
BinaryNode* findMax(BinaryNode *t)
begin
  // έλεγξε αν το δέντρο είναι άδειο
  if (t!=NULL)
  begin
    // προχώρα κάτω προς το rchild σε κάθε
  κόμβο
    while (t->rchild!=NULL)
       t = t->rchild;
  end
  return t;
end
```

findMax - Παράδειγμα



find (1/2)

- Ο αλγόριθμος find είναι συνδυασμός των τεχνικών της findMin και της findMax.
- Η διαφορά μεταξύ του find και των findMin/findMax:
  - Οι findMin/findMax εγγυώνται ότι θα επιστρέψουν έναν κόμβο όταν το BST δεν είναι άδειο. Από την άλλη, ο αλγόριθμος find επιστρέφει έναν κόμβο μόνο όταν το ζητούμενο κλειδί ‘x’ βρεθεί μέσα στο BST.

find (2/2)

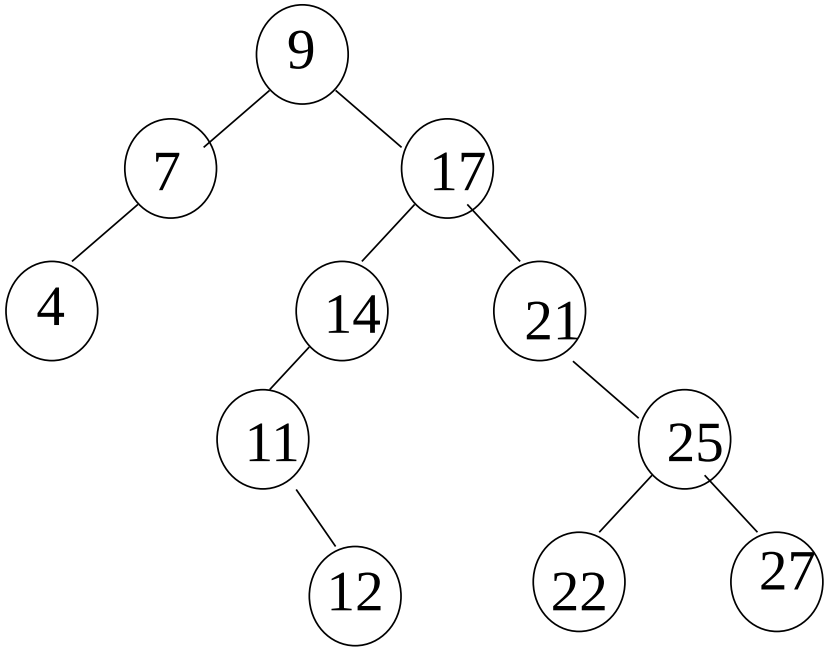
- Οι παράμετροι εισόδου είναι η ρίζα του BST και το κλειδί προς αναζήτηση: int x
  - find(x, root);
- Η μέθοδος find επιστρέφει τον κόμβο στο BST που περιέχει το κλειδί ‘x’ αν υπάρχει, και NULL αν δεν υπάρχει.
- Ο τύπος που επιστρέφει είναι πάλι BinaryNode*.

find Algorithm

```
// Non-recursive algorithm
BinaryNode* findMax(BinaryNode *t, key x)
begin
  // έλεγξε αν το δέντρο είναι άδειο
  if (t!=NULL)
  begin
    while (t!=NULL && x!=t->key)
       begin
      if (x < t -> key)
         t = t->lchild;
       else if (x > t -> key)
         t = t->rchild;
     end
  end
  return t;
end
```

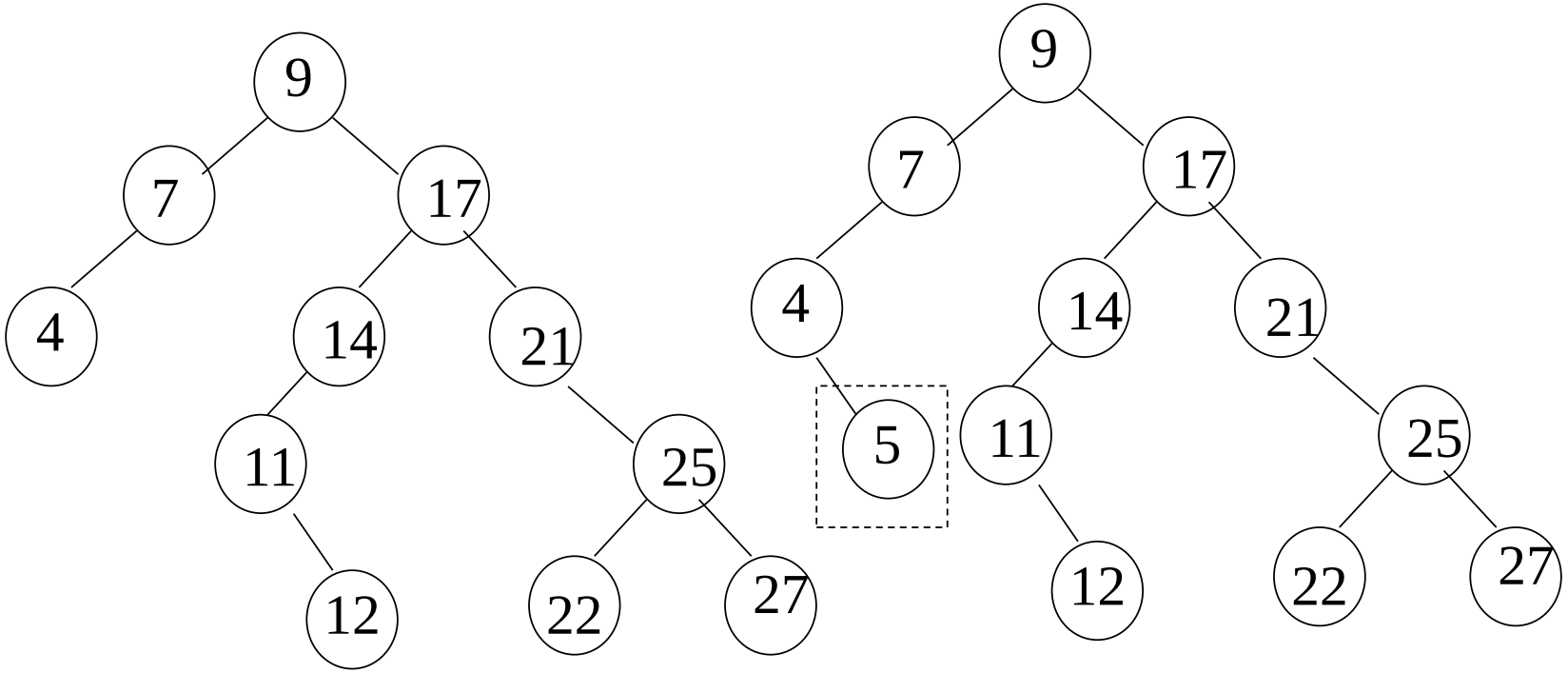
find - Παράδειγμα

- Εύρεση του κλειδιού ‘12’
- Εύρεση του κλειδιού ‘23’



Insert (1/4)

- Εισαγωγή του κλειδιού ‘5’



Insert – Παράδειγμα (εισαγωγή του 5)

- Ξεκινάμε από τη ρίζα που περιέχει το κλειδί 9.
- 5 < 9, πηγαίνουμε στο lchild του 9 που είναι το 7.
- 5 < 7, πηγαίνουμε στο lchild του 7 που είναι το 4.
- 5 > 4, πηγαίνουμε στο rchild του 4 που είναι το NULL.
- Δεν υπάρχει άλλο σημείο στο BST που μπορεί να περιέχει το 5 λόγω της BST ιδιότητας.
- Οπότε, εισάγουμε το 5 ως το rchild του 4.

Insert (2/4)

- Υποθέτοντας ότι **διπλά κλειδιά δεν υπάρχουν σε ένα BST.**
- Αν θέλουμε να εισάγουμε το κλειδί ‘x’, πρώτα κάνουμε ένα find(x).
- Αν το ‘x’ βρεθεί, δεν κάνουμε την εισαγωγή.
- Αν το ‘x’ δε βρεθεί, προχωράμε με την εισαγωγή.
- Αν το ‘x’ δε βρεθεί, τότε η find τελειώνει στον κόμβο κάτω από τον οποίο πρέπει να μπει το ‘x’.

Insert (3/4)

- insert (5, root).
- Πρώτα προσπαθούμε να βρούμε αν το 5 είναι ήδη στο BST.
- find algorithm.
  - 5 < 9 πηγαίνουμε στο lchild του 9 που είναι το 7.
  - 5 < 7 πηγαίνουμε στο lchild του 7 που είναι το 4.
  - 5 > 4 πηγαίνουμε στο rchild του 4 που είναι null και η find τερματίζει επιστρέφοντας NULL.
- Ο τελευταίος κόμβος που επισκεφτήκαμε ήταν ο 4.
- Το καινούργιο κλειδί 5 πρέπει να εισαχθεί κάτω από τον 4.
- Δημιουργούμε έναν καινούργιο κόμβο με κλειδί 5 και μια 5 > 4 προσθέτουμε τον καινούργιο κόμβο ως rchild του 4.

Insert (4/4)

- **Βήματα Αλγορίθμου:**
  - Τρέξε ένα find για το κλειδί που θέλουμε να εισαχθεί.
  - Αν το κλειδί βρεθεί, επέστρεψε NULL.
  - Αν το κλειδί δε βρεθεί, εισήγαγε έναν καινούργιο κόμβο που περιέχει το κλειδί ως αριστερό ή δεξιό παιδί του τελευταίου κόμβου που επισκέφτηκε η find.

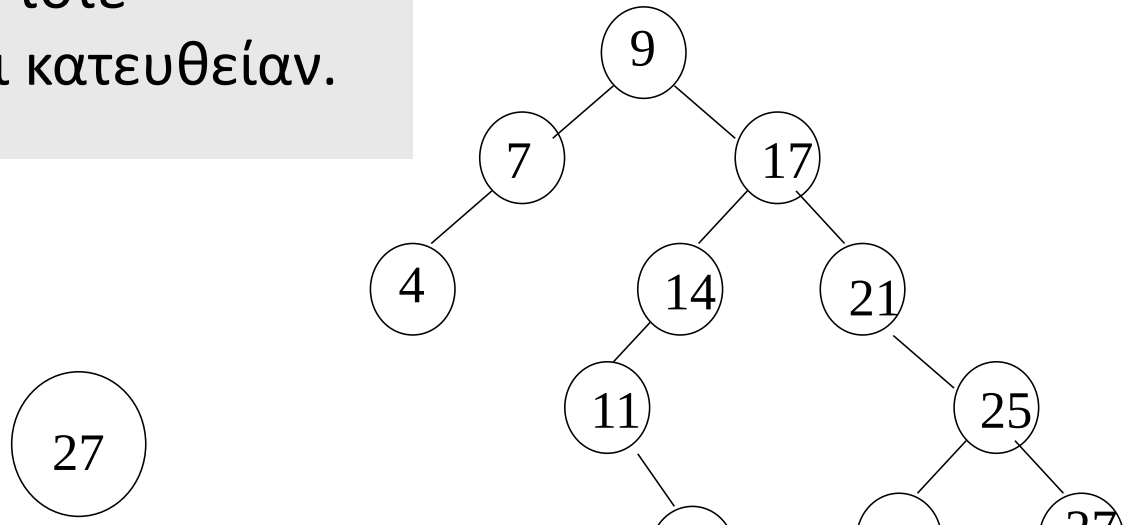
Παράδειγμα: Εισαγωγή των κλειδιών 6, 9, 14, 17, 5, 7, 16, 20, 18, 19, 4, 11 σε ένα κενό αρχικά δέντρο αναζήτησης.

remove

- Όπως στην περίπτωση του insert, πρέπει πρώτα να κάνουμε ένα find.
- Αν δε βρεθεί ο κόμβος, προφανώς δε μπορούμε να τον διαγράψουμε.
- Αν βρεθεί υπάρχουν 3 διαφορετικές περιπτώσεις ανάλογα με το είδος του κόμβου:
  - κόμβοι χωρίς παιδιά (φύλλα),
  - κόμβοι με ένα παιδί,
  - κόμβοι με δύο παιδιά.

remove – κανόνες (1/4)

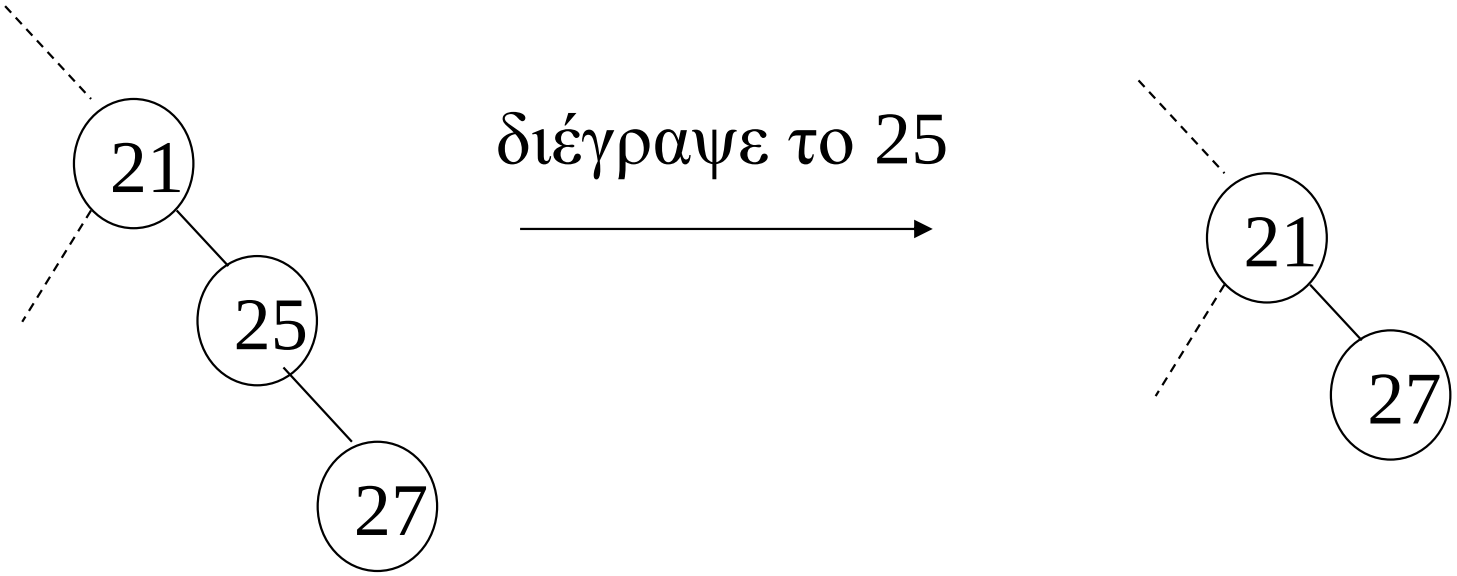
- **Κανόνες:**
  - 1) Αν ο κόμβος που θέλουμε να διαγράψουμε είναι φύλλο, τότε διαγράφεται κατευθείαν.



Μπορεί να διαγραφεί κατευθείαν

remove – κανόνες (2/4)

- 2) Αν ο κόμβος που θέλουμε να διαγράψουμε έχει ένα παιδί, το παιδί αυτό γίνεται το καινούργιο παιδί του πατέρα του διαγραμμένου κόμβου.

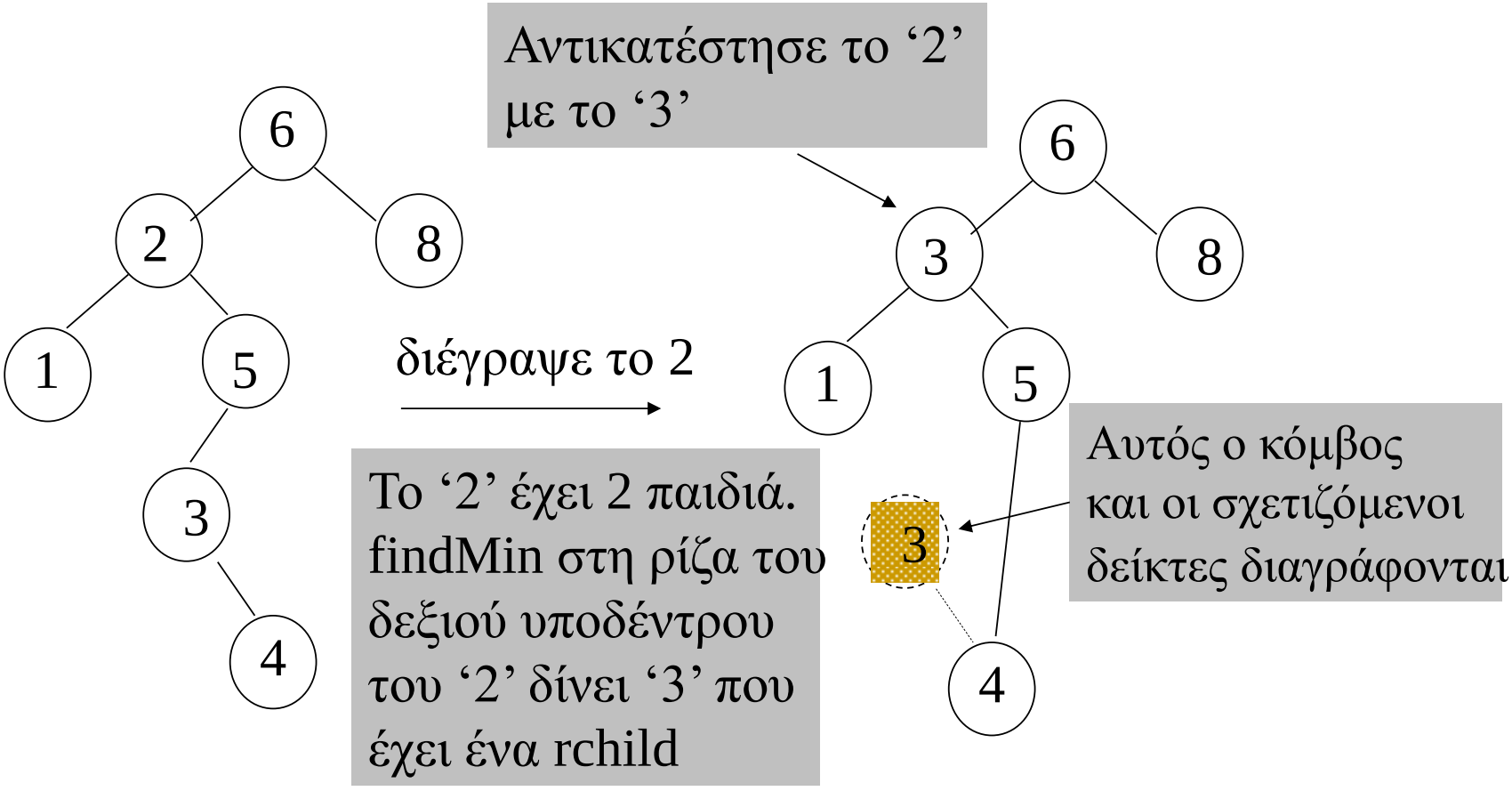


remove – κανόνες (3/4)

3) Αν ο κόμβος που θέλουμε να διαγράψουμε έχει δύο παιδιά.

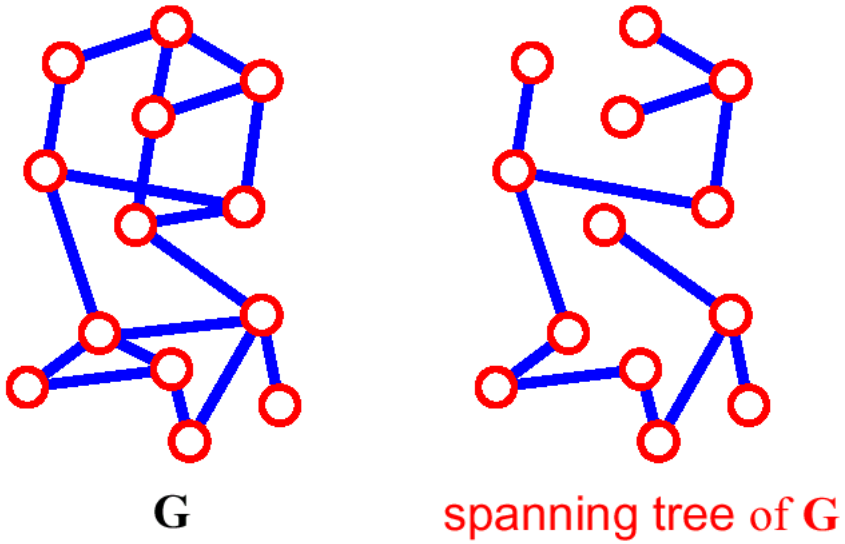
- Αντικατέστησε τον κόμβο προς διαγραφή με τον κόμβο που περιέχει το ελάχιστο κλειδί στο δεξιό του υποδέντρο (χρησιμοποιώντας τη findMin στη ρίζα του δεξιού υποδέντρου του κόμβου προς διαγραφή).
  - Ο κόμβος με το ελάχιστο κλειδί δε μπορεί να έχει lchild. Αν είχε τότε το lchild θα περιείχε το ελάχιστο κλεδί.
  - Οπότε ο κόμβος με το ελάχιστο κλειδί έχει μόνο rchild ή καθόλου παιδιά.

remove – κανόνες (4/4)



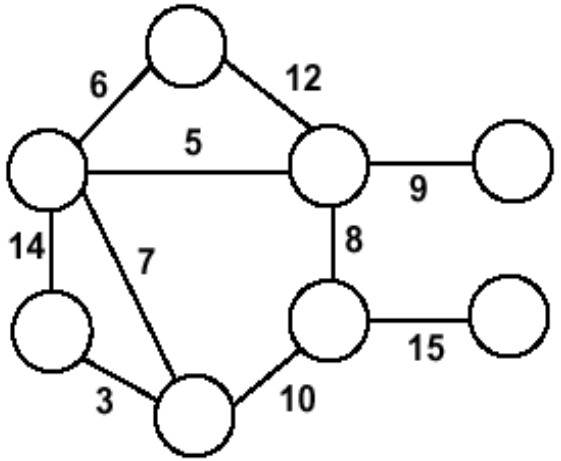
Επικαλύπτοντα Δέντρα (Spanning Trees)

- Ένα **επικαλύπτον δέντρο ενός γραφήματος** *G* είναι ένα υπογράφημα του *G* που
  - είναι δέντρο,
  - περιέχει όλους τους κόμβους του *G*.



Ελάχιστα Επικαλύπτοντα Δέντρα (1/2)

- Μη κατευθυνόμενο, συνδεδεμένο γράφημα $G = (V,E)$.
- Συνάρτηση Βάρους W: $E \to R$ (δίνει κάποιο κόστος ή άλλη τιμή στις ακμές).
- **Επικαλύπτον δέντρο**: δέντρο που συνδέει όλους τους κόμβους (στο παράδειγμα;).
- **Ελάχιστο επικαλύπτον δέντρο**: συνδέει όλους τους κόμβους και ελαχιστοποιεί:



$$w(T) = \sum_{(u,v)\in T} w(u,v)$$

Ελάχιστα Επικαλύπτοντα Δέντρα (2/2)

- **Παράδειγμα:** Έστω κάποιες πόλεις που συνδέονται με χωματόδρομους. Η ασφαλτόστρωση κάθε δρόμου έχει κάποιο συγκεκριμένο χρηματικό κόστος. Θέλουμε να ασφαλτοστρώσουμε με το ελάχιστο κόστος ένα υποσύνολο των δρόμων έτσι ώστε να συνδέονται όλες οι πόλεις.
  - **δίκτυο χωματόδρομων – γράφημα**
  - **πόλεις – κορυφές γραφήματος**
  - **δρόμοι – ακμές γραφήματος**
  - **κόστος ασφαλτόστρωσης δρόμου – βάρος ακμής**
- Το ζητούμενο είναι να προσδιοριστεί ένα ελάχιστο επικαλύπτον δέντρο.

Ο Αλγόριθμος του Prim (1/2)

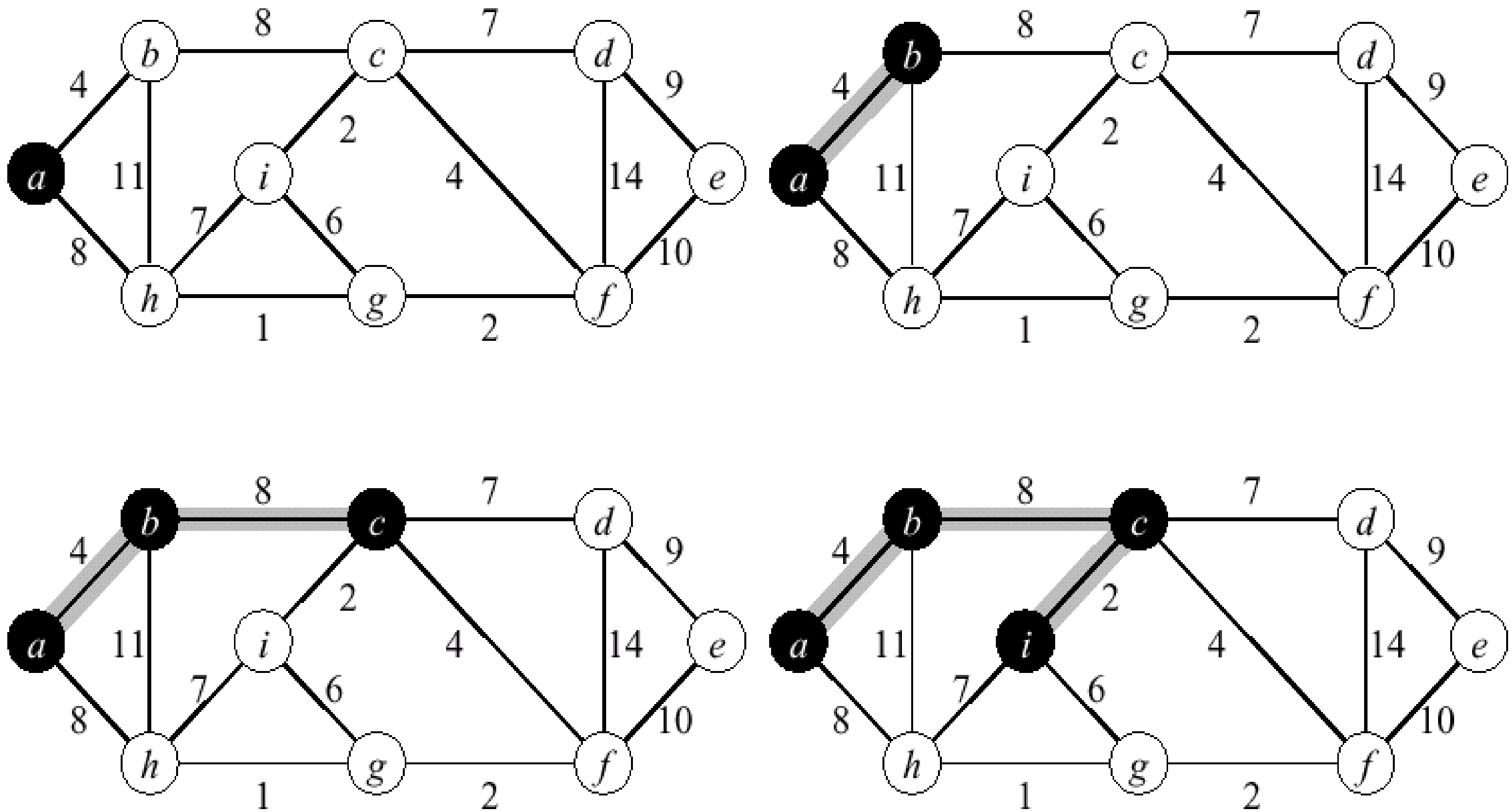
- Βασίζεται στους κόμβους.
- Χτίζει ένα δέντρο **T**, κόμβο με κόμβο.
  - Οι κόμβοι του **T** που έχουν ήδη υπολογιστεί ανήκουν σε ένα σύνολο **S.**
- Σε κάθε κόμβο v που βρίσκεται εκτός συνόλου δώσε την τιμή *key*[*v*] = το ελάχιστο βάρος μιας ακμής που συνδέει το v με έναν κόμβο μέσα στο σύνολο **S.**
  - *key*[*v*] = ∞, αν δεν υπάρχει τέτοιος κόμβος
- Πρόσθεσε στο **S** τον κόμβο με την ελάχιστη τιμή του *key.*
- Ο αλγόριθμος τελειώνει όταν όλοι οι κόμβοι ανήκουν στο **S.**

Ο Αλγόριθμος του Prim (2/2)

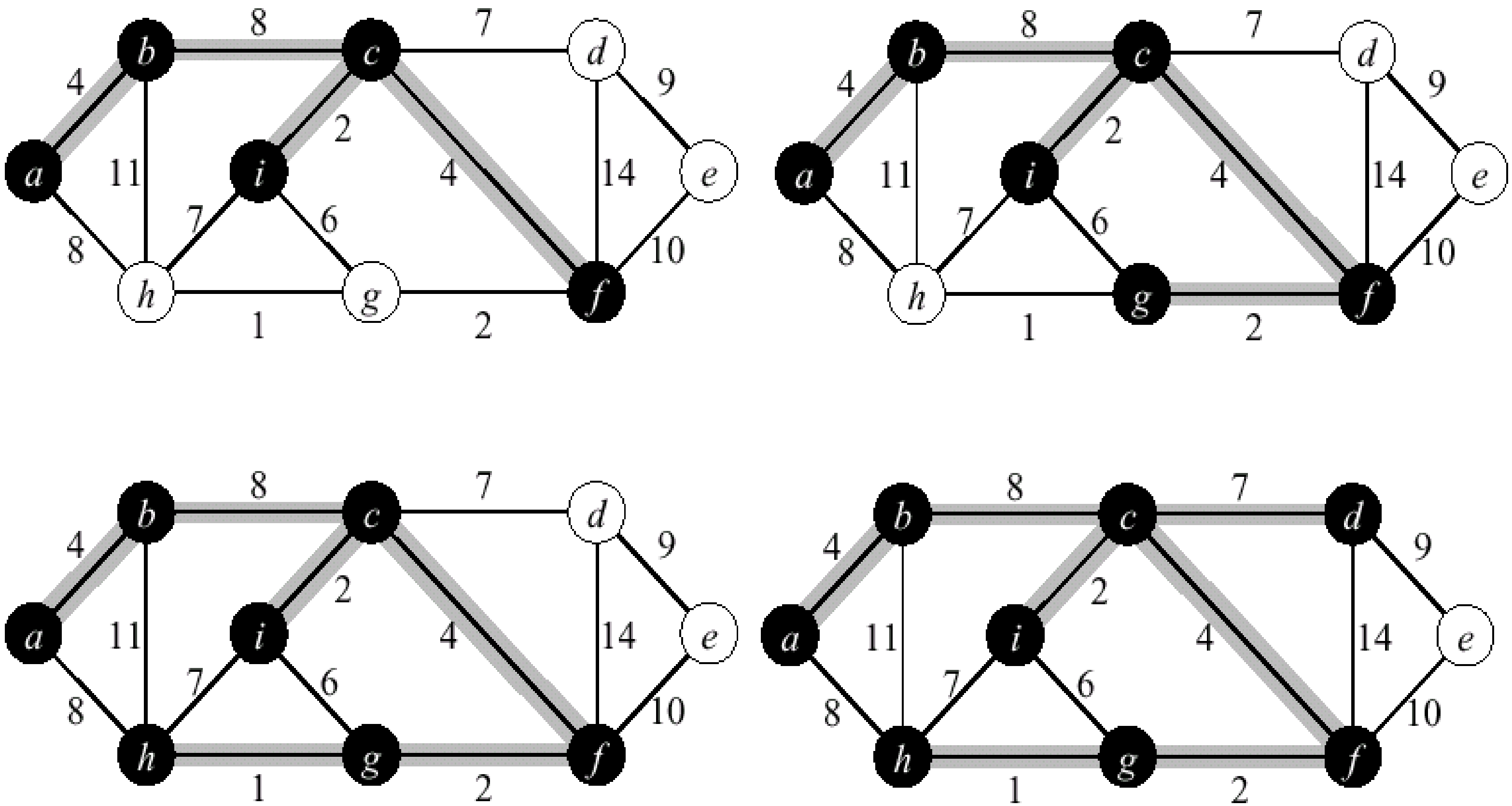
```
MST-Prim(G,w,r)
01 Q := V[G]
02 for each u ∈ Q do
03     key[u] := ∞
04 key[r] := 0
05 π[r] := NIL
06 while Q ≠ ∅ do
07    u := ExtractMin(Q)
08       for each v ∈ Adj[u] do
09          if v ∈ Q and w(u,v) < key[v] then
10             π[v] := u
11             key[v] := w(u,v)
```

updating keys

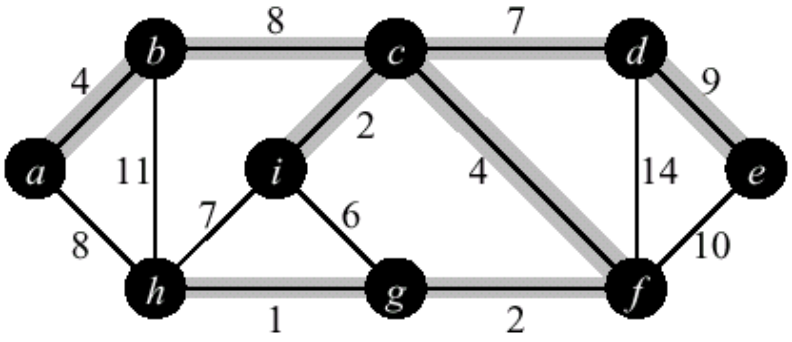
Ο Αλγόριθμος του Prim – Παράδειγμα (1/3)



Ο Αλγόριθμος του Prim
– Παράδειγμα (2/3)



Ο Αλγόριθμος του Prim – Παράδειγμα (3/3)



Ο αλγόριθμος του Kruskal (1/5)

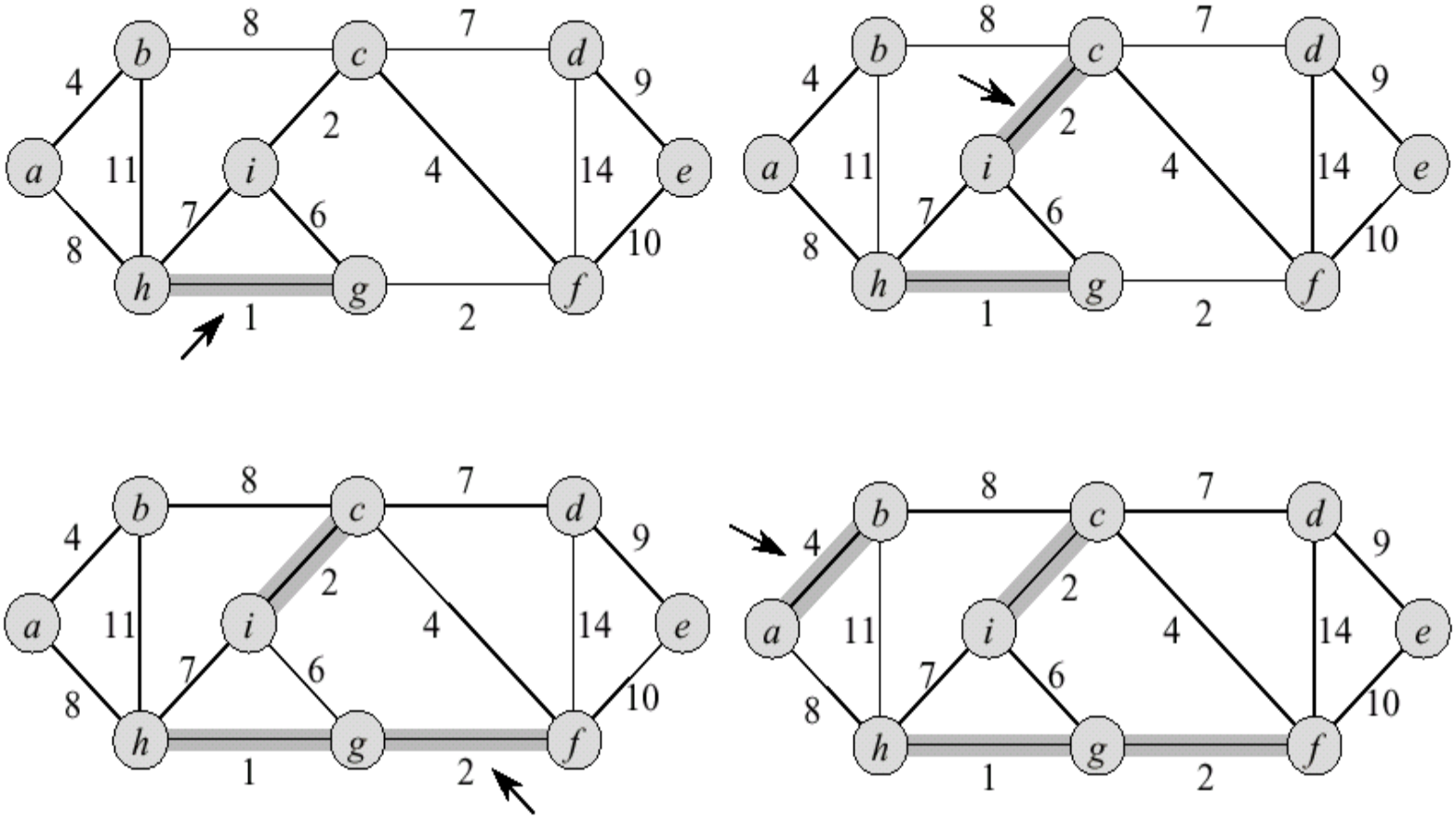
- Βασίζεται στις ακμές.
- Προσθέτει τις ακμές μία μία, με αυξανόμενο βάρος.
  - Πρέπει να είναι ταξινομημένες.
- Ο αλγόριθμος διατηρεί ένα **δάσος από δέντρα – Α.**
  - Αρχικά υπάρχουν μόνο κορυφές στο δάσος.
  - Μια ακμή γίνεται δεκτή αν συνδέει κόμβους από διαφορετικά δέντρα.
  - Ο αλγόριθμος χρησιμοποιεί δύο συναρτήσεις
    - `Find-Tree(u)` : βρίσκει το δέντρο στο οποίο ανήκει η κορυφή u.
    - `Union-Trees(u,v)` : ενώνει τα δέντρα στα οποία ανήκουν οι κορυφές u και v.

Ο αλγόριθμος του Kruskal (2/5)

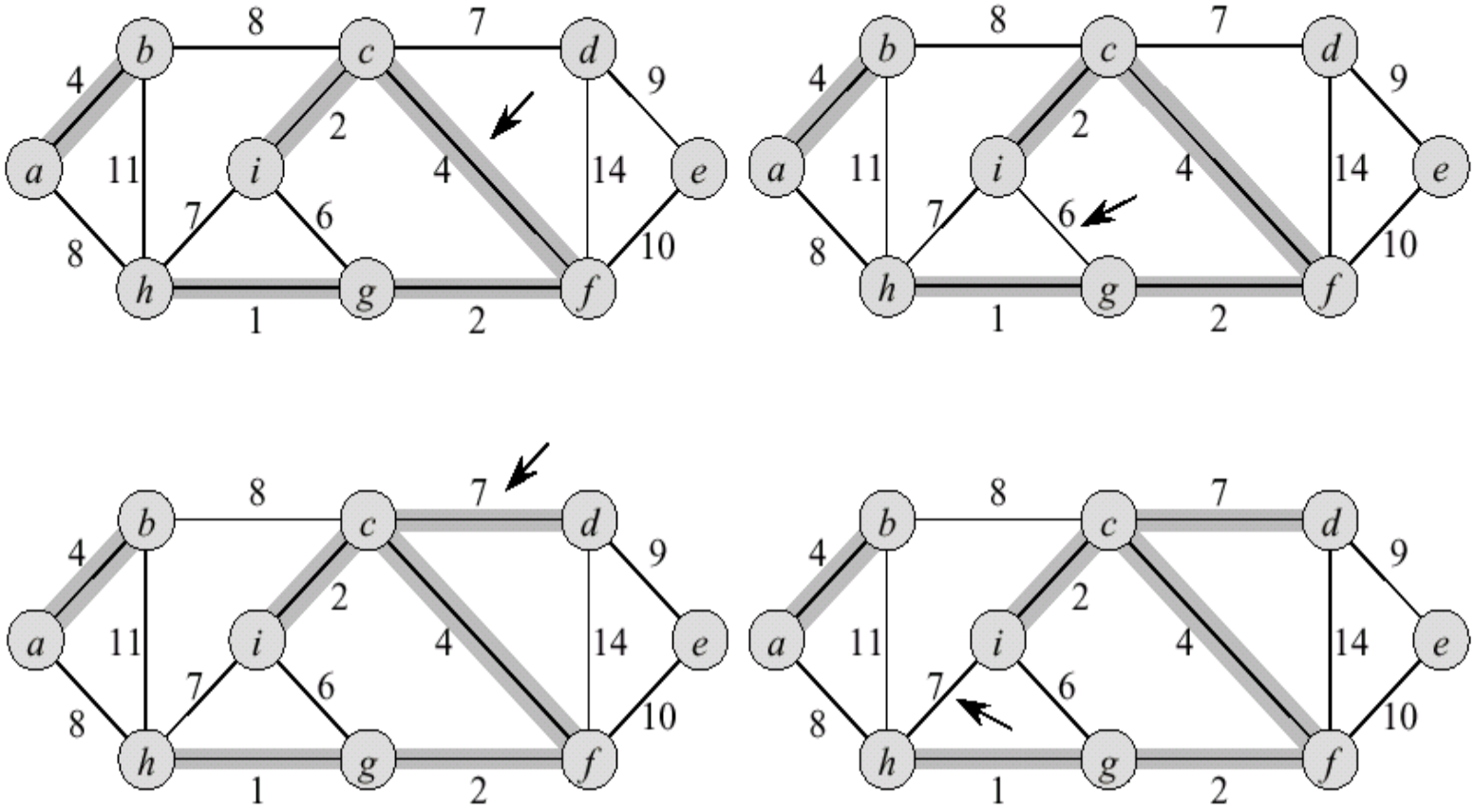
- Ο αλγόριθμος προσθέτει τη φτηνότερη ακμή που συνδέει δύο δέντρα του δάσους έτσι ώστε να μην προκύπτει κύκλος.

```
MST-Kruskal(G,w)
01 A ← ∅
02 for each vertex v ∈ V[G] do
03    Make-Tree(v)
04 sort the edges of E by non-decreasing
   weight w
05 for each edge (u,v) ∈ E, in order by
   non-decreasing weight do
06    if Find-Tree(u) ≠ Find-Tree(v) then
07       A ← A ∪ {(u,v)}
08       Union-Trees(u,v)
09 return A
```

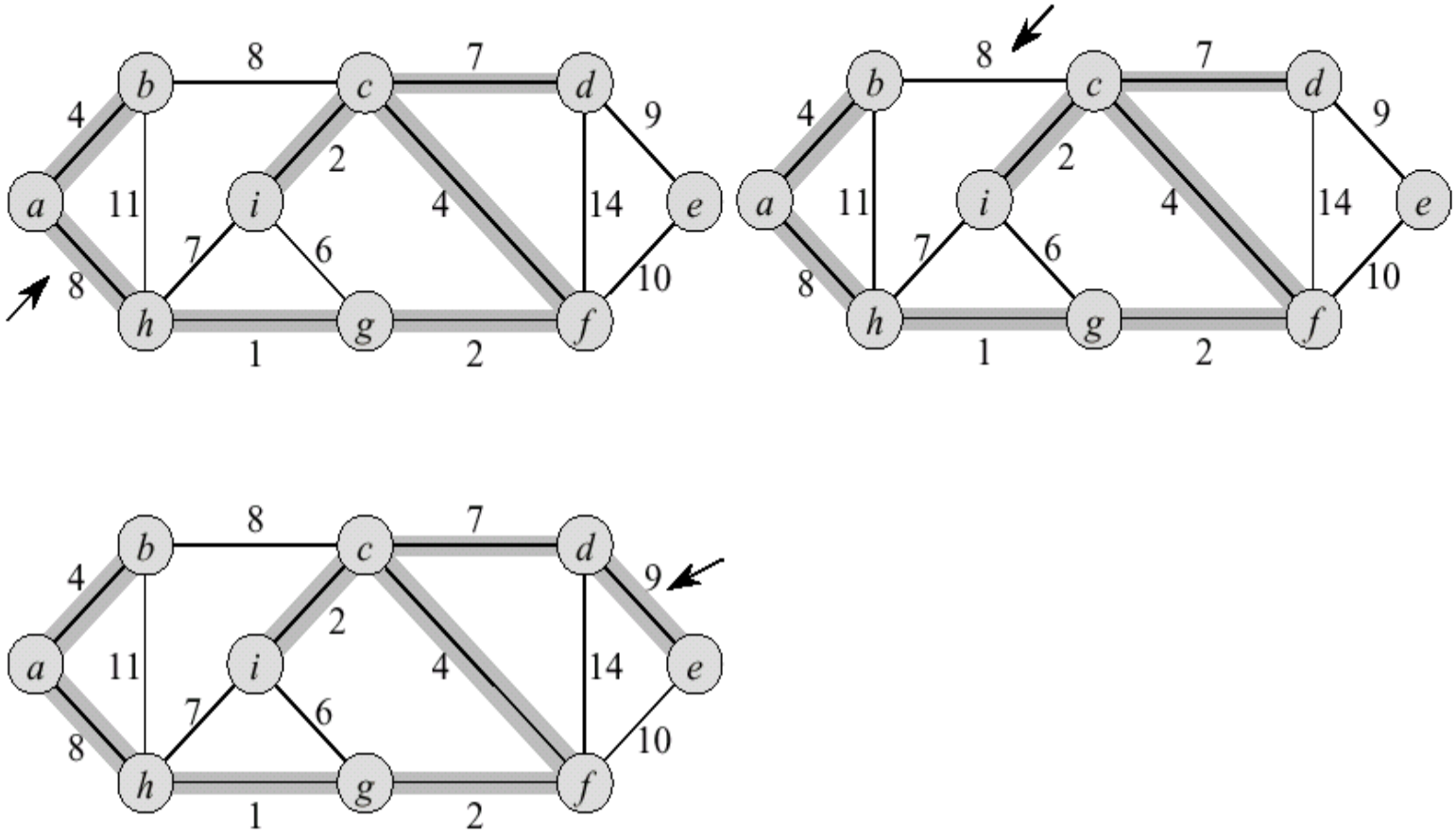
Ο αλγόριθμος του Kruskal (3/5)



Ο αλγόριθμος του Kruskal (4/5)



Ο αλγόριθμος του Kruskal (5/5)



Τέλος Ενότητας

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Σημείωμα Αναφοράς

- Copyright Πανεπιστήμιο Δυτικής Μακεδονίας, Τμήμα Μηχανικών Πληροφορικής και Τηλεπικοινωνιών, Στεργίου Κωνσταντίνος. «Διακριτά Μαθηματικά». Έκδοση: 1.0. Κοζάνη 2015. Διαθέσιμο από τη δικτυακή διεύθυνση:

  https: //eclass.uowm.gr/courses/ICTE257/

Σημείωμα Αδειοδότησης

Το παρόν υλικό διατίθεται με τους όρους της άδειας χρήσης Creative Commons Αναφορά, Όχι Παράγωγα Έργα Μη Εμπορική Χρήση 4.0 [1] ή μεταγενέστερη, Διεθνής Έκδοση. Εξαιρούνται τα αυτοτελή έργα τρίτων π.χ. φωτογραφίες, διαγράμματα κ.λ.π., τα οποία εμπεριέχονται σε αυτό και τα οποία αναφέρονται μαζί με τους όρους χρήσης τους στο «Σημείωμα Χρήσης Έργων Τρίτων».

[1] h t t p ://creativecommons.org/licenses/by-nc-nd/4.0/

Ως Μη Εμπορική ορίζεται η χρήση:

- που δεν περιλαμβάνει άμεσο ή έμμεσο οικονομικό όφελος από την χρήση του έργου για το διανομέα του έργου και αδειοδόχο
- που δεν περιλαμβάνει οικονομική συναλλαγή ως προϋπόθεση για τη χρήση ή πρόσβαση στο έργο
- που δεν προσπορίζει στο διανομέα του έργου και αδειοδόχο έμμεσο οικονομικό

Διατήρηση Σημειωμάτων

Οποιαδήποτε αναπαραγωγή ή διασκευή του υλικού θα πρέπει να συμπεριλαμβάνει:

- το Σημείωμα Αναφοράς
- το Σημείωμα Αδειοδότησης
- τη δήλωση Διατήρησης Σημειωμάτων
- το Σημείωμα Χρήσης Έργων Τρίτων (εφόσον υπάρχει)

μαζί με τους συνοδευόμενους υπερσυνδέσμους.