Χάρτες

Διονύσης "dionyziz" Ζήνδρος

Camp Προετοιμασίας ΠΔΠ

Άνοιξη 2011

Χάρτες

- Σύνολο από ζεύγη τιμών: (0, 2), (8, 12), (-1, 17)
- Ίδιος τύπος για όλα τα πρώτα στοιχεία του ζεύγους
- Ίδιος τύπος για όλα τα δεύτερα στοιχεία του ζεύγους
- Πρώτο και δεύτερο στοιχείο του ζεύγους πιθανώς διαφορετικοί τύποι

π.χ. Χάρτες:

- από int σε int
- από string σε int
- από int σε string

Ζεύγη χαρτών

- Πρώτο στοιχείο κάθε ζεύγους: κλειδί
- Δεύτερο στοιχείο κάθε ζεύγους: τιμή

Βασικές ιδιότητες χαρτών:

- Κάθε κλειδί αντιστοιχεί σε μοναδική τιμή
- Μία τιμή μπορεί να αντιστοιχεί σε πολλά διαφορετικά κλειδιά
- Αναπαριστά μία συνάρτηση
- Η σειρά συνήθως δεν έχει σημασία
- Μόνο η αντιστοιχία κλειδιού - τιμής

Χάρτες και όχι χάρτες

- (1, 2), (5, 1), (5, 2), (9, 12), (29, 7)
- ("Γιώργος", 12), ("Θανάσης", 7), ("Λίτσα", 9)
- ("Γιώργος", 12), (12, "Γιώργος")
- (1, 2), (2, 2), (3, 2), (4, 2), (5, 2)

Πολύ χρήσιμη δομή. Παραδείγματα

- Αντιστοιχούμε μέσο όρο βαθμολογίας σε μαθητές
  - Κλειδί: Μαθητής
  - Τιμή: Μέσος όρος βαθμολογίας
  - Δύο μαθητές μπορούν να έχουν τον ίδιο Μ.Ο.
  - Ένας μαθητής δεν μπορεί να έχει 2 διαφορετικούς Μ.Ο.
- Αντιστοιχούμε ιστοσελίδα σε αριθμό επισκέψεων
  - Κλειδί: Ιστοσελίδα
  - Τιμή: Αριθμός επισκέψεων
  - Δύο ιστοσελίδες μπορούν να έχουν τον ίδιο αριθμό
  - Δύο αριθμοί δεν μπορούν να αντιστοιχούν στην ίδια ιστοσελίδα

Τα καλά των χαρτών

- Γρήγοροι
- Πολλές πρακτικές εφαρμογές σε διάφορα προβλήματα
- Έχουν μεταβλητό μέγεθος
- Είναι αποδοτικοί και σε χώρο
- Χρησιμοποιούνται σε άλλες δομές δεδομένων
- Μοιάζουν με τις ουρές προτεραιότητας που είναι άλλη χρήσιμη δομή

Υλοποίηση

- Χρησιμοποιούμε τους έτοιμους χάρτες της C++ (map)
- Εύκολη υλοποίηση με πίνακα από struct
- Καλύτερη υλοποίηση με δέντρα

Υλοποίηση με πίνακα

```
struct map_item {
    int key;
    int value;
};

map_item map[ 10 ];
```

Καλύτερη υλοποιήση

Υλοποίηση με δέντρο:

- Διατηρεί το χάρτη ταξινομημένο με βάση το κλειδί
- Έτσι μπορεί να χρησιμοποιεί δυαδική αναζήτηση
- Εισαγωγή σε O(log(n))
- Αναζήτηση κλειδιού σε O(log(n))
- Επεξεργασία τιμής κλειδιού σε O(log(n))

Απαιτούμε τα κλειδιά να μπορούν να διαταχθούν:

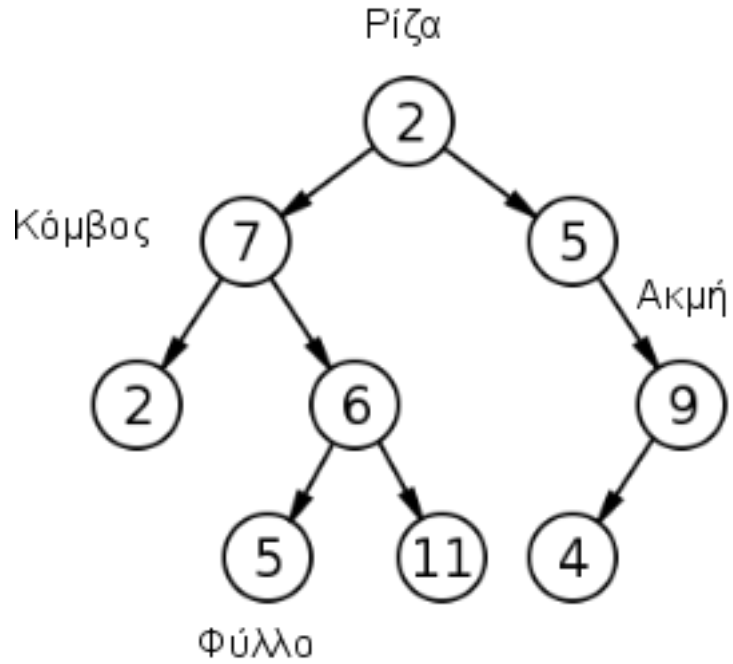
- Αριθμοί
- Χαρακτήρες
- Αλφαριθμητικά (λεξικογραφική διάταξη)

Δένδρα

- Δομή με κόμβους (ή κορυφές) και ακμές
- Κάθε κόμβος έχει έναν πατέρα
- Εκτός από τη ρίζα που δεν έχει
- Κάθε κόμβος έχει μία τιμή
- Όλες οι τιμές είναι ίδιου τύπου
- Όσοι κόμβοι δεν έχουν παιδιά λέγονται φύλλα

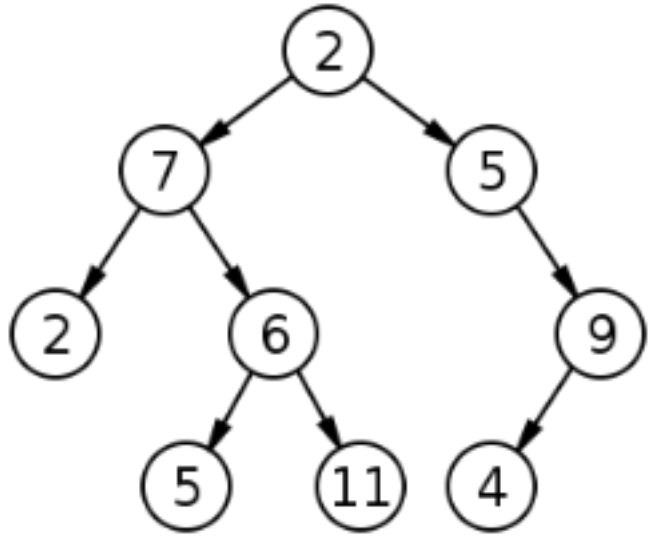
Δένδρα

- Η ρίζα έχει τιμή 2
- Η ρίζα έχει 2 παιδιά
- Ο κόμβος με τιμή 7 είναι το αριστερό παιδί της ρίζας
- Ο κόμβος με τιμή 7 έχει πατέρα την ρίζα
- Η κορυφή με τιμή 9 έχει μόνο ένα παιδί
- Η κορυφή με τιμή 4 είναι φύλλο
- Το δένδρο έχει 4 φύλλα



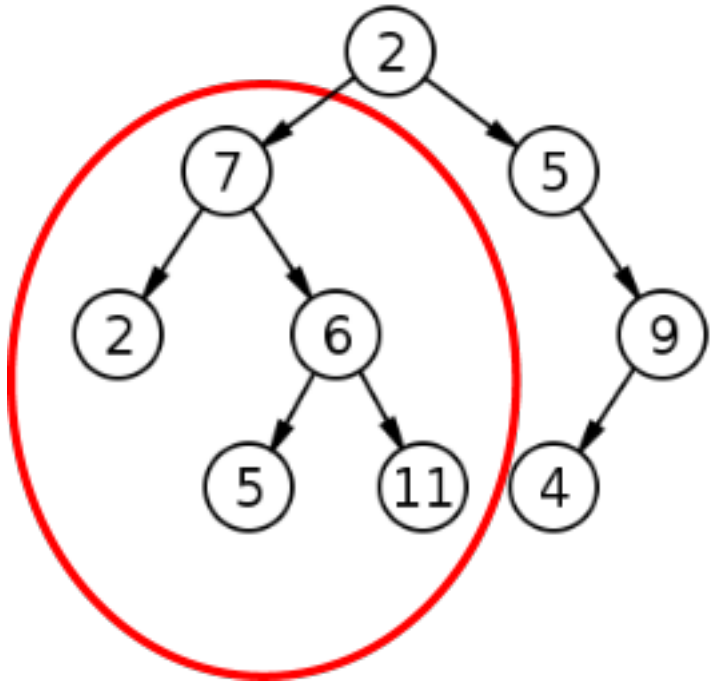
Δένδρα

- Οι κορυφές είναι μία παραπάνω από τις ακμές
  - Κάθε κορυφή έχει πατέρα, εκτός από τη ρίζα
- Συμβολισμός:
  - Κορυφές: V
  - Ακμές: E
  - Πλήθος κορυφών: |V|
  - Πλήθος ακμών: |E|



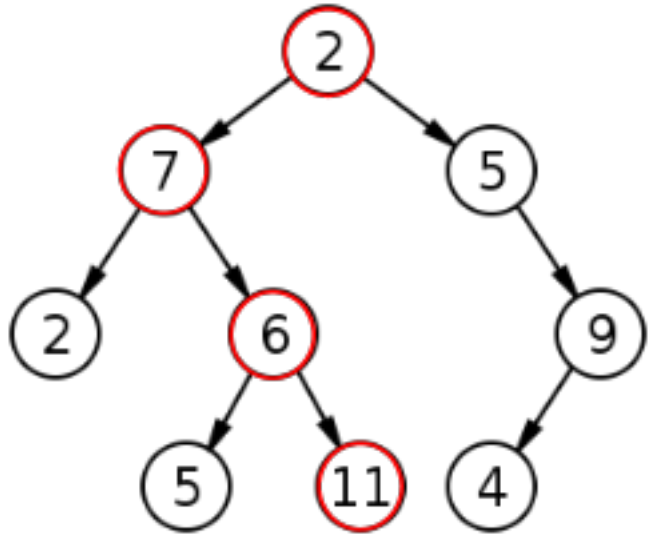
Υπόδεντρο

- Επιλέγουμε μία κορυφή
- Την κάνουμε ρίζα και κρατάμε τα παιδιά της, τα παιδιά των παιδιών, κ.ό.κ., αλλά όχι τους προγόνους



Ύψος (ή βάθος)

- Κοιτάμε την απόσταση κάθε κόμβου από τη ρίζα
  - Απόσταση: Πλήθος ακμών που πρέπει να περάσουμε
- Από όλες κρατάμε την μεγαλύτερη

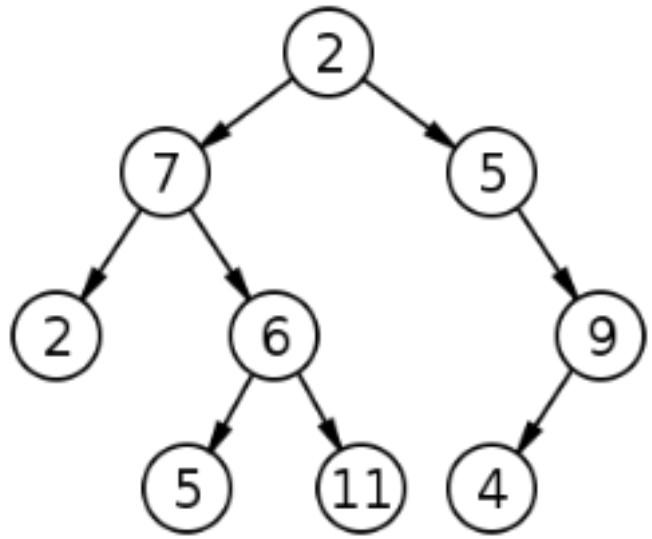


Δυαδικά δέντρα

- Δυαδικό δέντρο: Κάθε κόμβος έχει το πολύ 2 παιδιά

Στο παράδειγμα:

- Το 5 έχει 1 παιδί
- Το 4 έχει 0 παιδιά
- Το 6 έχει 2 παιδιά



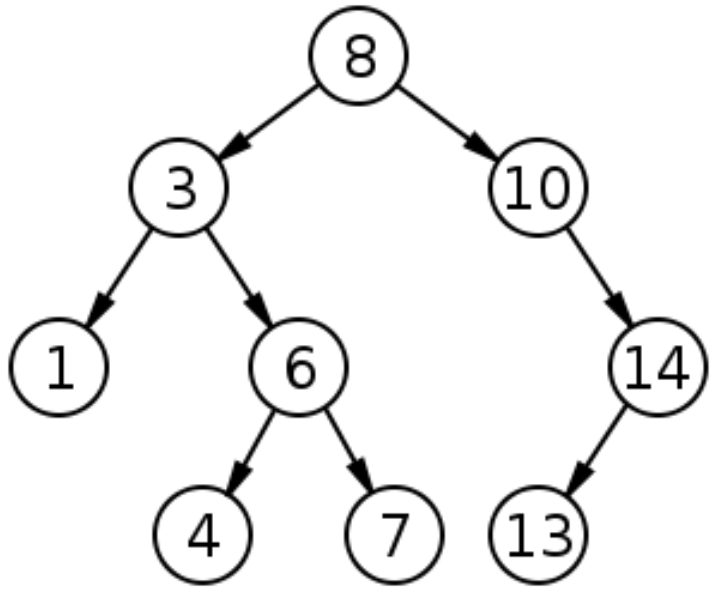
Υλοποίηση

```
struct tree_item {
    int value
    tree_item* left;
    tree_item* right;
};

tree_item* root;
```

Δυαδικά δέντρα αναζήτησης

- Οι κόμβοι στα αριστερά είναι μικρότεροι της ρίζας
- Οι κόμβοι στα δεξιά είναι μεγαλύτεροι της ρίζας
- Το υπόδεντρο του αριστερού παιδιού της ρίζας είναι επίσης δυαδικό δέντρο αναζήτησης
- Το ίδιο και του δεξιού

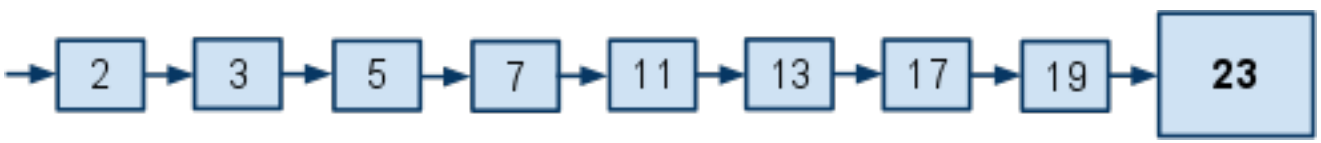


Δυαδικά δέντρα αναζήτησης

- Αναζήτηση στοιχείου σε O(log(N)) με δυαδική αναζήτηση
- Εισαγωγή στοιχείου κατά μέσο όρο σε O(log(N))
- Εξαιρετικά για την υλοποίηση χαρτών:
  - Δημιουργούμε ένα δέντρο που περιέχει structs κλειδιού - τιμής
  - Εύκολα επιτυγχάνουμε καλή πολυπλοκότητα

Ουρές προτεραιότητας

- Σύνολο από στοιχεία με διάταξη
- Μας ενδιαφέρει να παίρνουμε κάθε φορά το μεγαλύτερο
  - (ή το μικρότερο)



Λειτουργίες ουράς προτεραιότητας

- Προσθήκη στοιχείου
  - Προστίθεται στη "σωστή" θέση διάταξης μέσα την ουρά
- Ανάγνωση/διαγραφή στοιχείου
  - Το μεγαλύτερο κάθε φορά
- Αλλαγή τιμής κάποιου στοιχείου
  - Όχι απαραίτητα το μεγαλύτερο