ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

ΑΝΟΙΚΤΑ
ΑΚΑΔΗΜΑΪΚΑ
ΜΑΘΗΜΑΤΑ

Δομές Δεδομένων

Ενότητα 4: Ουρές

Απόστολος Παπαδόπουλος
Τμήμα Πληροφορικής

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

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 Ακαδημαϊκά Μαθήματα στο Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης» έχει χρηματοδοτήσει μόνο την 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.

Ουρές

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

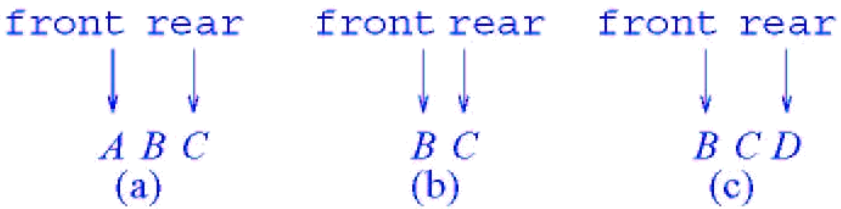
Περιεχόμενα ενότητας

1. Ουρές (queues).
2. Ουρές Προτεραιότητας (priority queues).

Ουρές

Ουρά (queue)

- Δομή τύπου FIFO:
  First In - First Out
  (πρώτη εισαγωγή – πρώτη εξαγωγή)
- Περιορισμένος τύπος γραμμικής λίστας: Εισαγωγή στο ένα άκρο (στο τέλος) και διαγραφή από το άλλο (την αρχή)



AbstractDataType Queue {

instances

ordered list of elements; one end is called the front; the other is the rear;

operations

Create (): create an empty queue;

IsEmpty (): return true if queue is empty, return false otherwise;

First (): return first (front) element of queue;

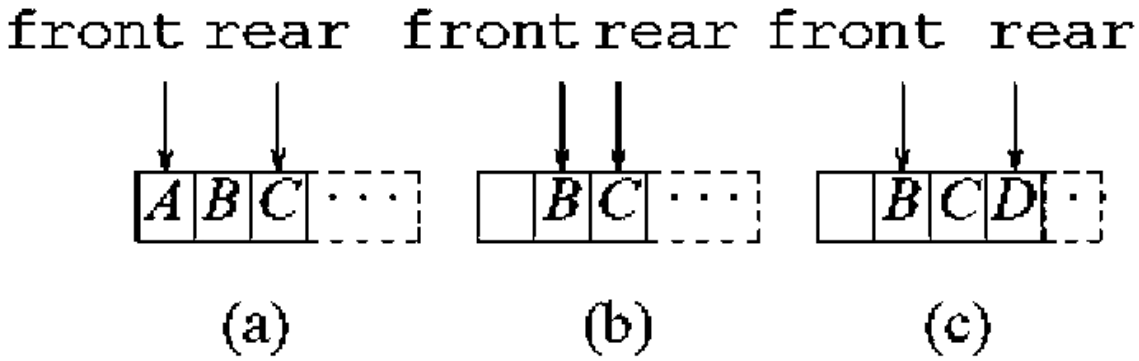
Last (): return last (rear) element of queue;

Add (x): add element x to the queue;

Delete (x): delete front element from queue and put it in x;

}

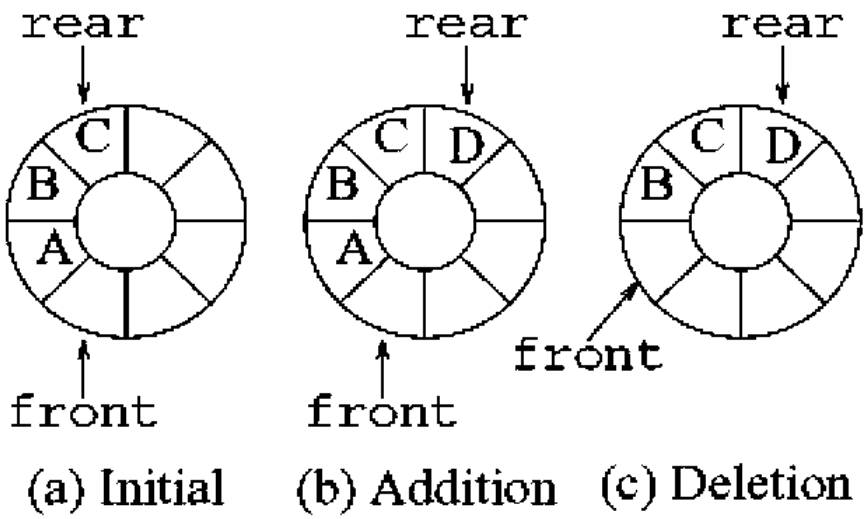
Υλοποίηση ουράς με πίνακα



location(i)=front+i-1

- Ερωτήσεις:
  - Γιατί να μη ξεκινά η ουρά πάντα από την αρχή του πίνακα;
  - Τι πρόβλημα δημιουργείται με την ‘ολίσθηση’ της ουράς προς τα δεξιά;

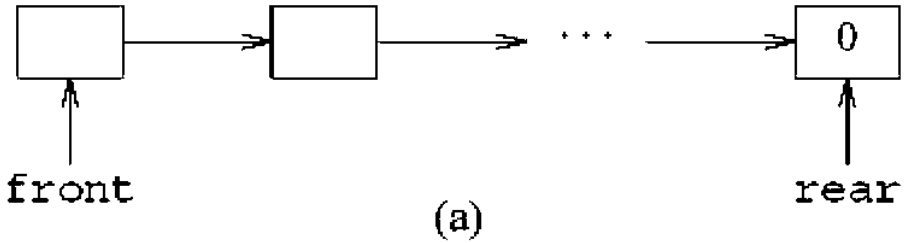
Υλοποίηση κυκλικής ουράς (με πίνακα)



location(i)=(front+i)%MaxSize

- Ερωτήσεις:
  - Πού δείχνουν οι δείκτες front και rear;
  - Πώς διακρίνουμε μεταξύ μιας άδειας και μιας γεμάτης ουράς;

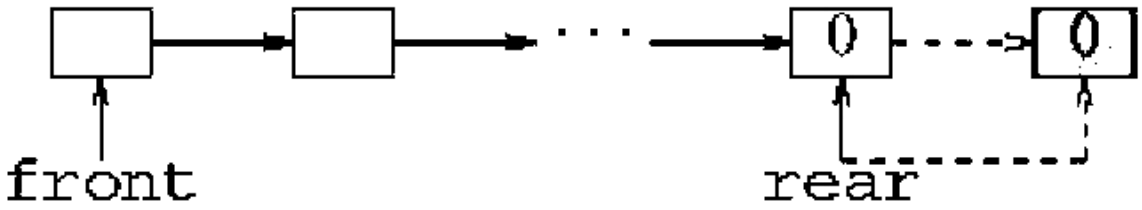
Συνδεδεμένη ουρά




```
class LinkedQueue {
// FIFO objects
  public:
    LinkedQueue()
                              {front = rear = 0;}; // constructor
    ~LinkedQueue(); // destructor
    int IsEmpty() {return ((front) ? 0 : 1);}
    int IsFull();
    int First(type& x); // return first element of queue
    int Last(type& x);  // return last element of queue
    int operator +(type x); // add x to queue
    int operator -(type& x); // delete x from queue
                              // First,+,- return 0 on failure, 1 on success
  private:
    Node<type> *front, *rear;
}
```

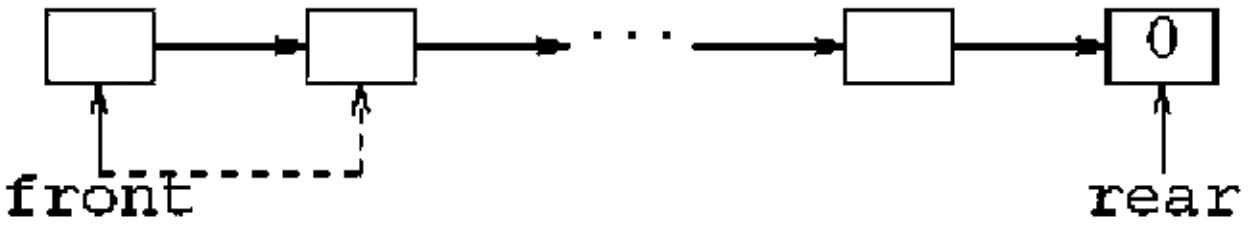
Προσθήκη (ως τελευταίο στοιχείο) σε συνδεδεμένη ουρά

```
int LinkedQueue<type>::operator+(type x)
//add x to queue
{
Node<type> *i;
i = new Node<type>;
if (i) {
    i->data = x; i->link = 0;
    if (front) rear->link = i;
    else front = i;
    rear = i; return 1;
    };
return 0; // add fails
}
```

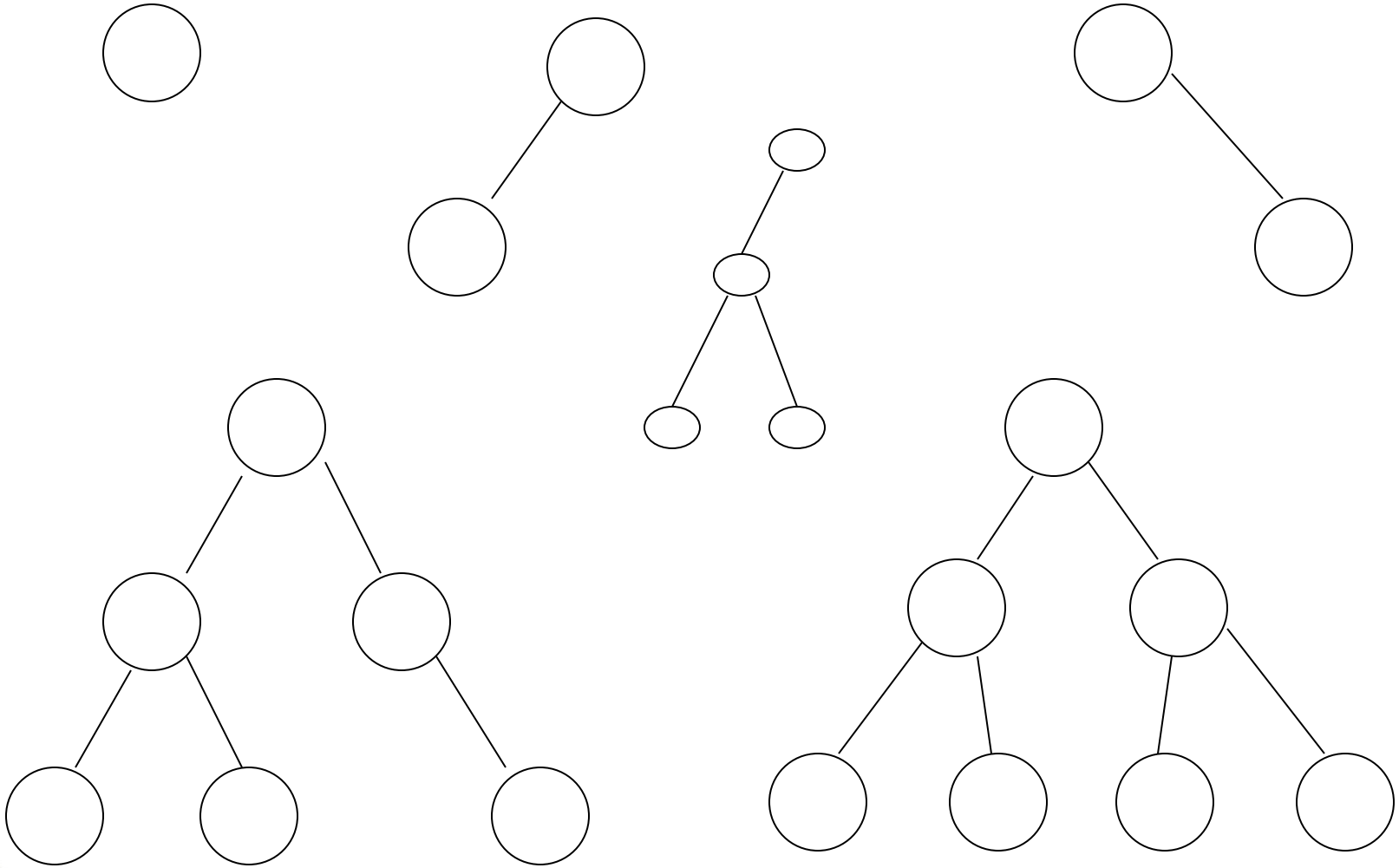


Διαγραφή (του πρώτου στοιχείου) από συνδεδεμένη ουρά

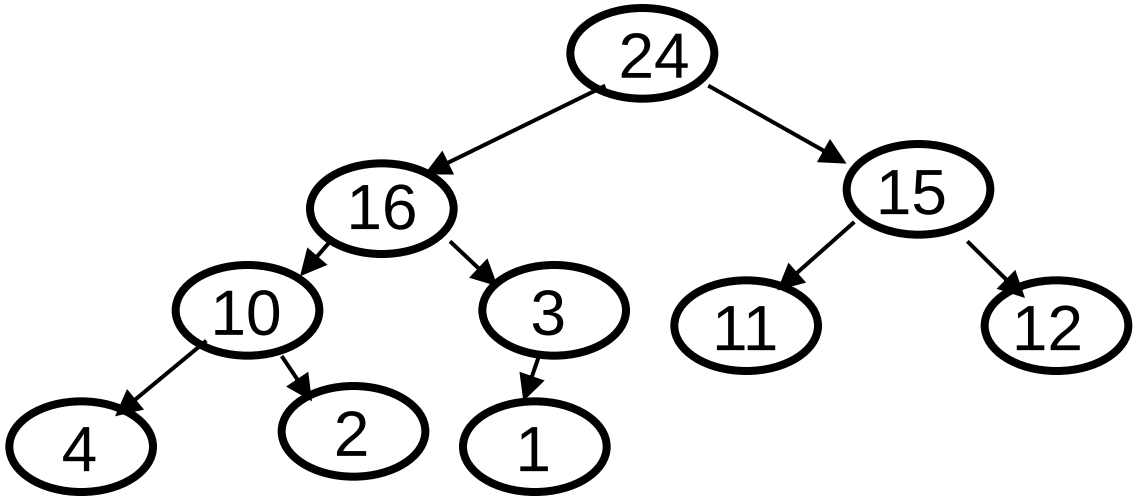
```
int LinkedQueue<type>::operator-(type& x)
//delete first element and return in x
{
if (IsEmpty()) return 0; //delete fails
x = front->data;
Node<type> *i = front;
front = front->link;
delete i;
return 1;
}
```



Ποιο από τα παρακάτω είναι σωρός;

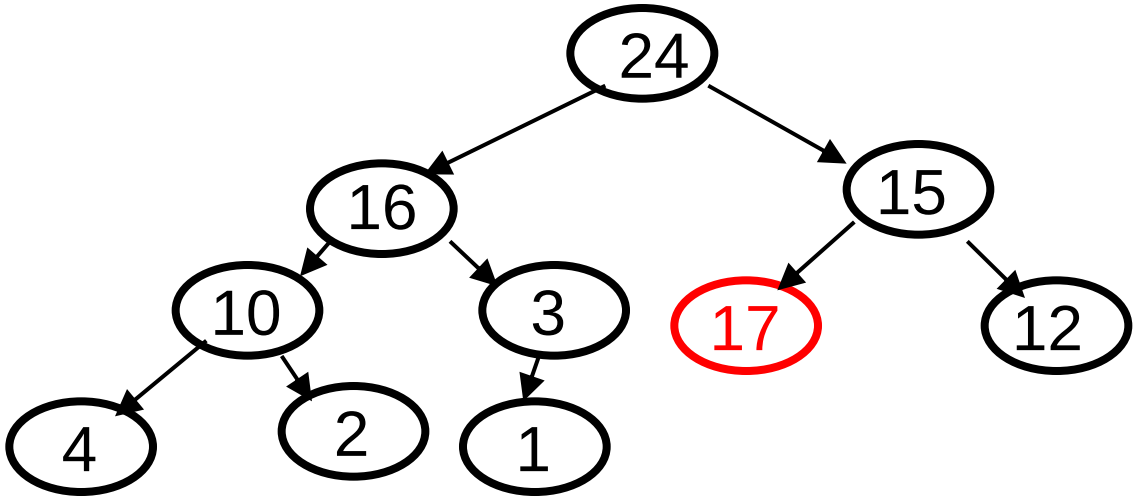


Παράδειγμα Ι



Αυτό είναι σωρός

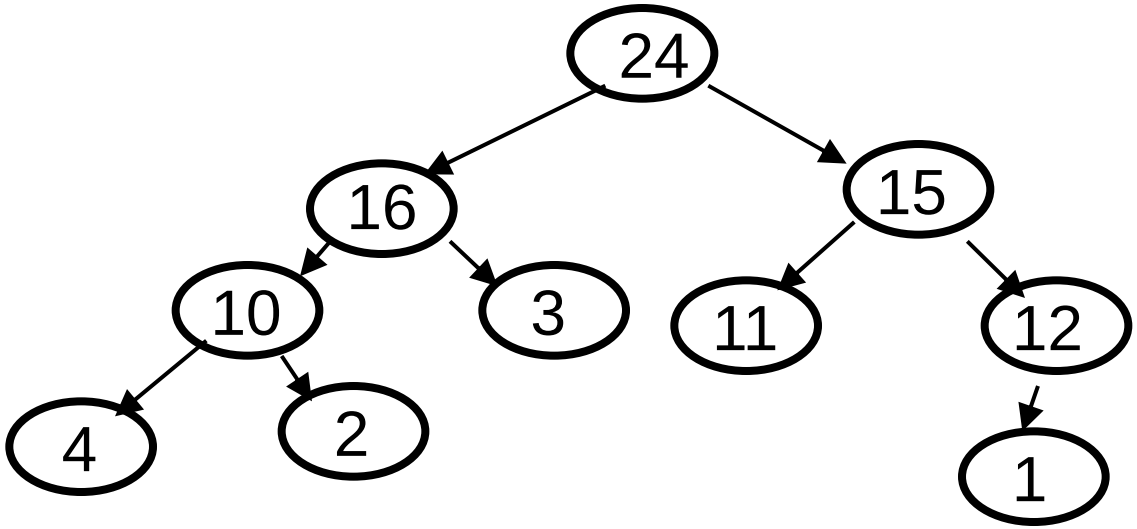
Παράδειγμα ΙΙ



Δεν είναι σωρός, γιατί δεν ικανοποιείται η ιδιότητα του σωρού

(17 > 15)

Παράδειγμα III



Δεν είναι σωρός, γιατί ο κόμβος με το 1 δεν είναι σε σωστή θέση.

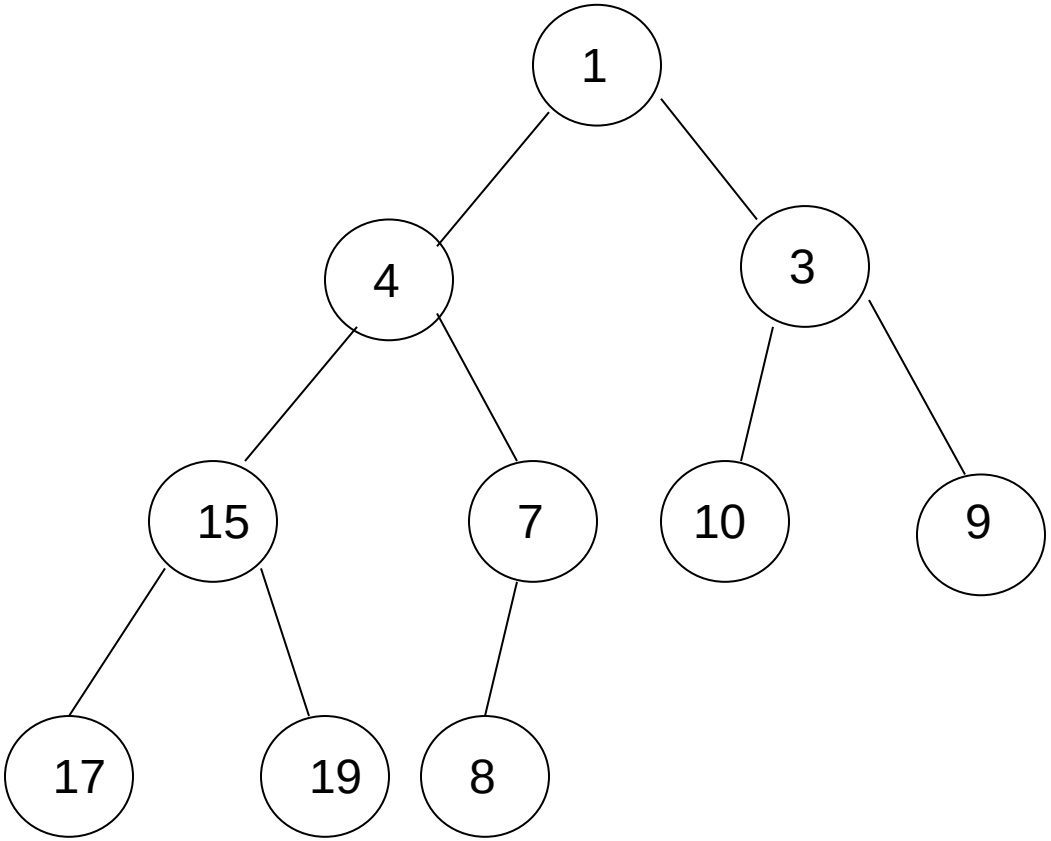
ΑΡΙΣΤΟΤΕΛΕΙΟ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΟΝΙΚΗΣ

Με πίνακα

Με δυναμική δέσμευση μνήμης

Αναπαράσταση

Αναπαράσταση με Πίνακα(1/2)



1	4	3	15	7	10	9	17	19	8

Αναπαράσταση με Πίνακα(2/2)

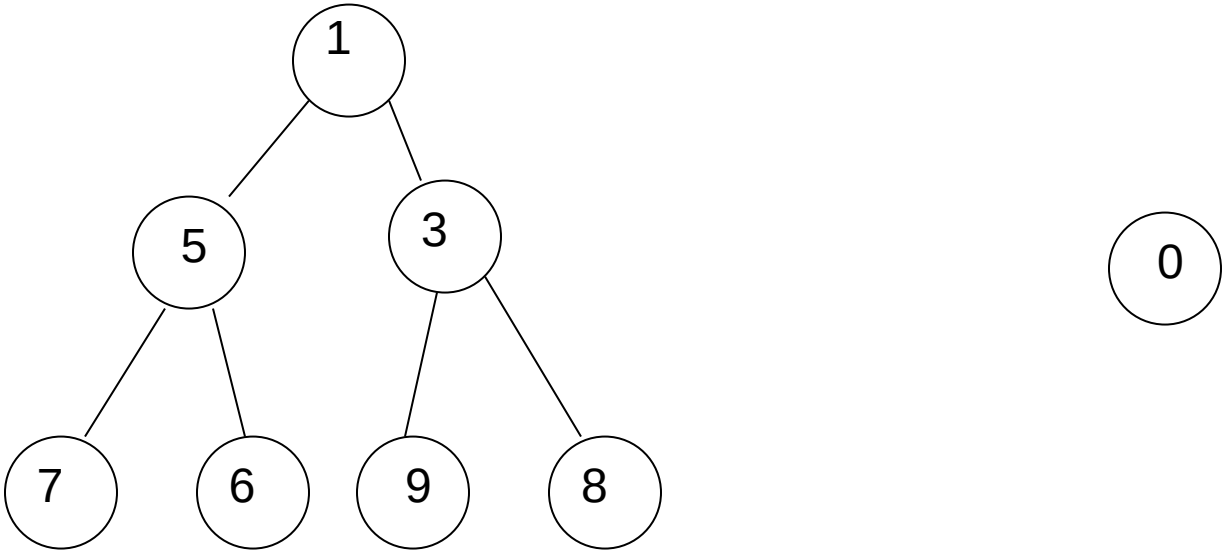
- Για τον κόμβο που είναι στη θέση A[i],
  - Αριστερό παιδί στη θέση A[2i]
  - Δεξιό παιδί στη θέση A[2i+1]
  - Γονέας στη θέση A[$\lfloor i/2 \rfloor$]

- Και τα τρία τα βρίσκουμε σε χρόνο O(1)

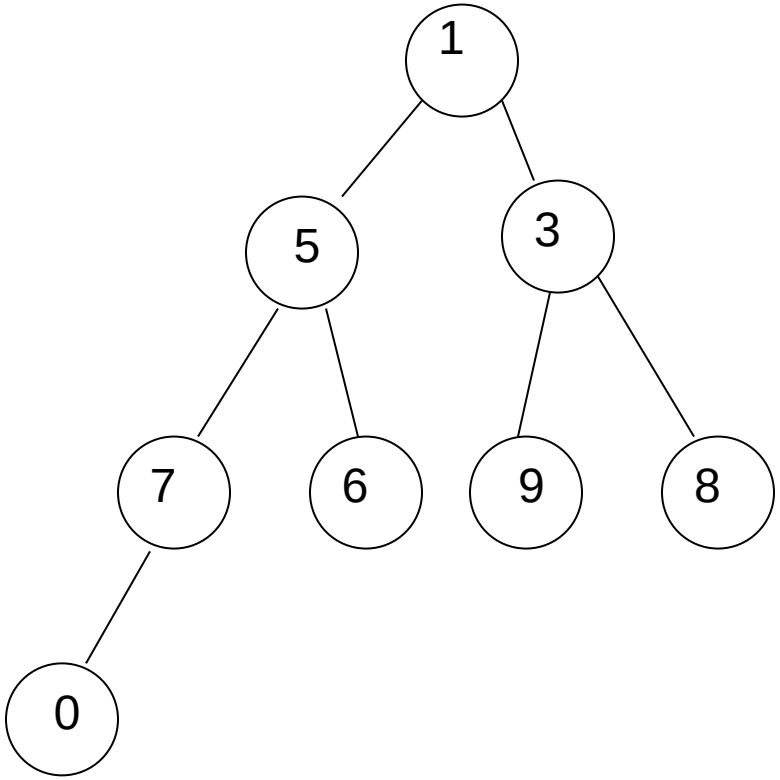
Εισαγωγή

- Έστω ότι έχουμε ένα σωρό και θέλουμε να εισάγουμε ένα νέο στοιχείο.
- Μέθοδος
  - Βάζουμε το νέο στοιχείο στο τέλος του σωρού.
  - Πρέπει να βεβαιωθούμε ότι ικανοποιείται η ιδιότητα του σωρού.

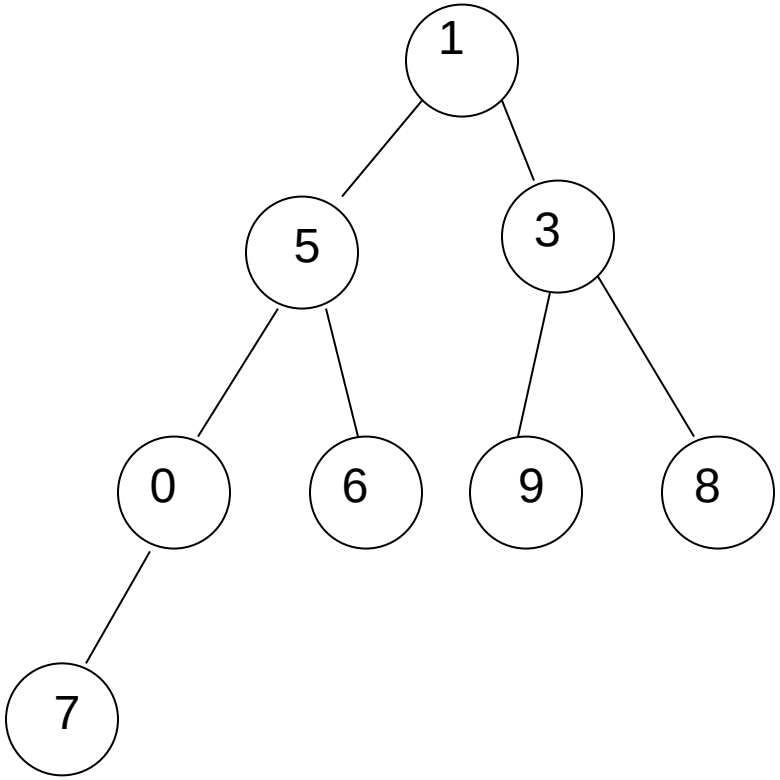
Εισάγουμε το 0



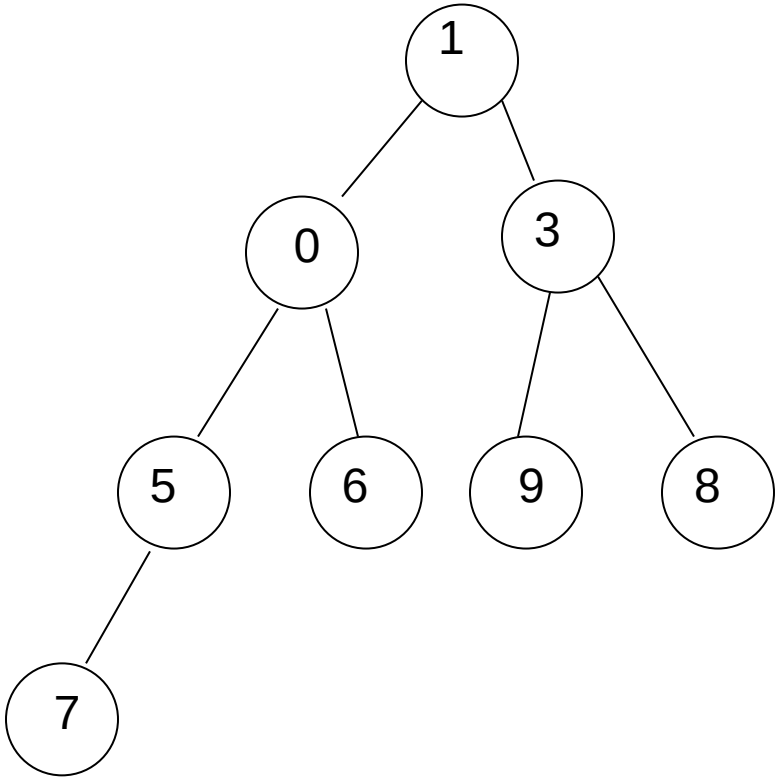
Τοποθετούμε το 0 στην τελευταία θέση του σωρού



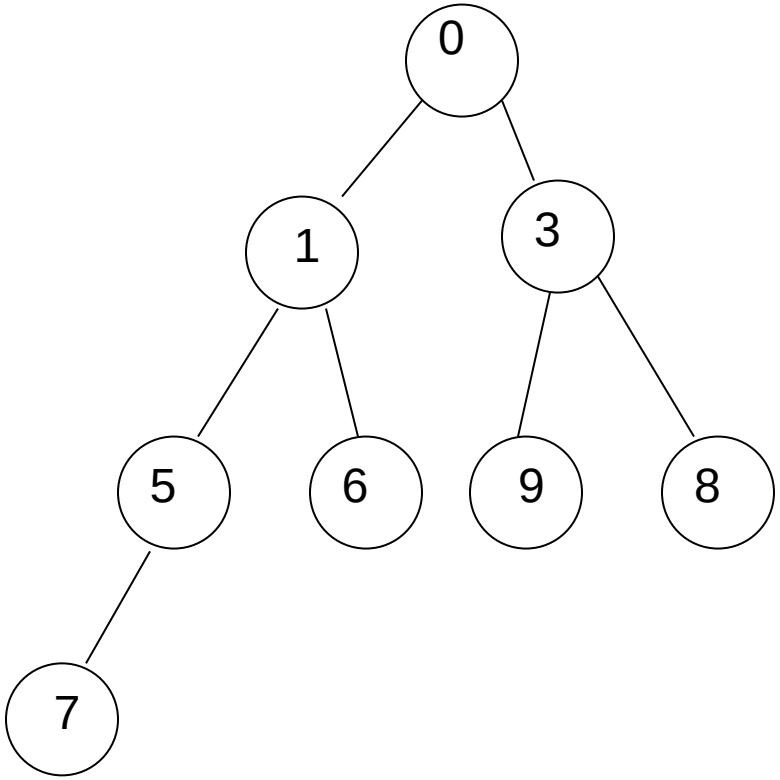
Ανταλλαγή 7 ←→ 0



Ανταλλαγή 5 ←→ 0



Ανταλλαγή 1 ←→ 0



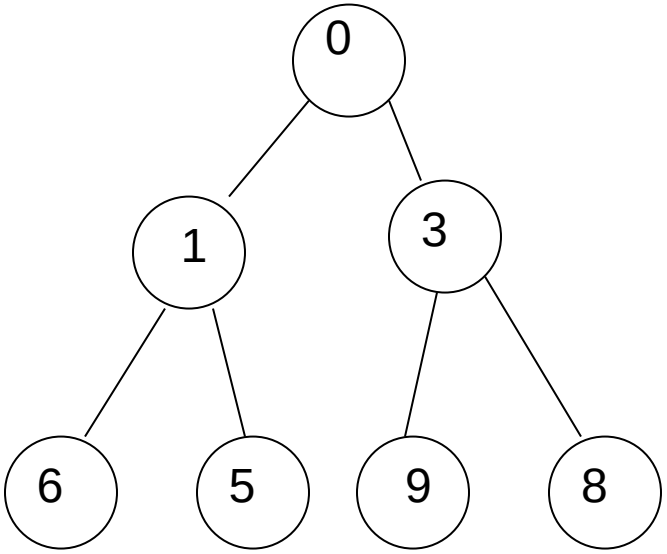
Κόστος Εισαγωγής

- Για την αναδιάρθρωση του σωρού δεν απαιτείται πάντα να φτάσουμε μέχρι τη ρίζα.
- Απαιτούμενο κόστος: O(*log* n) αφού το ύψος του σωρού που έχει n στοιχεία είναι O(*log* n)

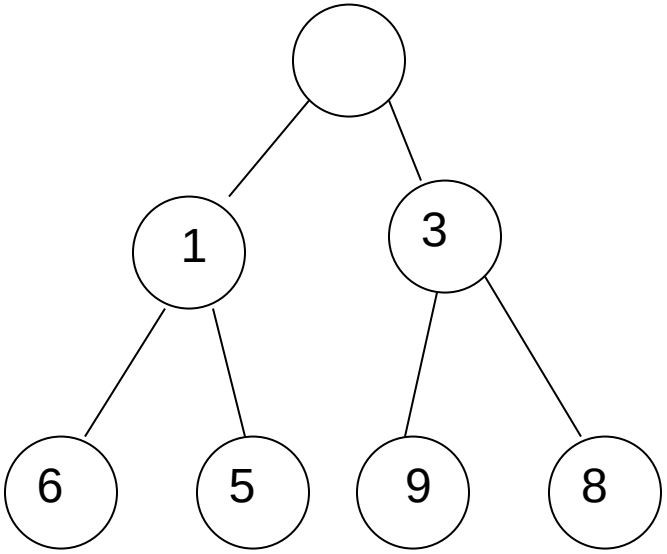
Διαγραφή της Κορυφής

- Ακολουθούμε την ακόλουθη μέθοδο:
  - Διαγράφουμε το στοιχείο στην κορυφή και το αντικαθιστούμε με το τελευταίο στοιχείο του σωρού.
  - Ακολουθεί αναδιάρθρωση του σωρού εάν απαιτείται.

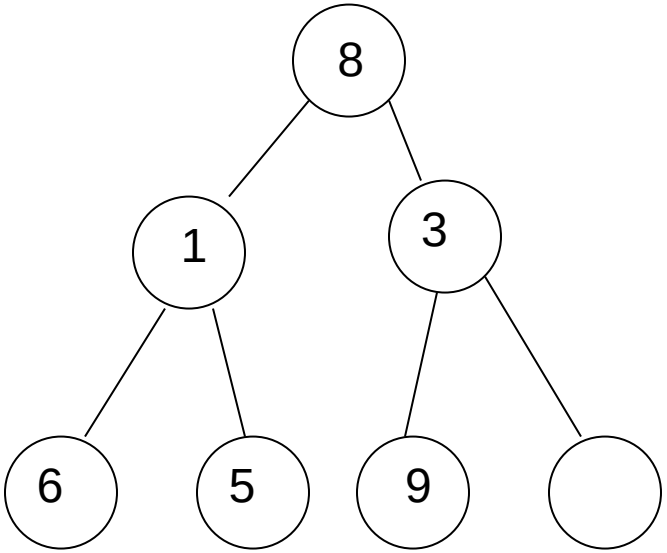
Διαγραφή του 0 (1/2)



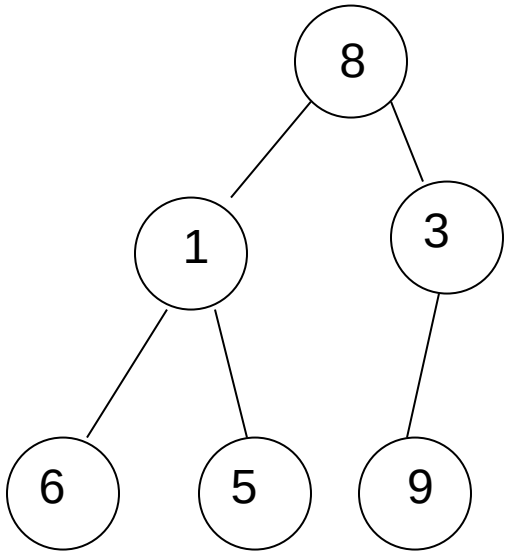
Διαγραφή του 0 (2/2)



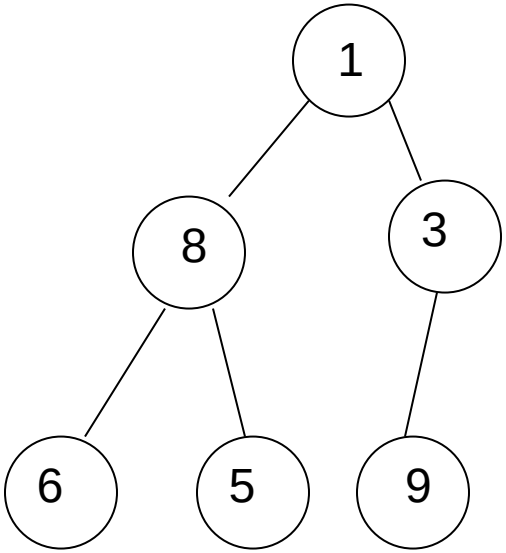
Αντικατάσταση με το τελευταίο στοιχείο (8)



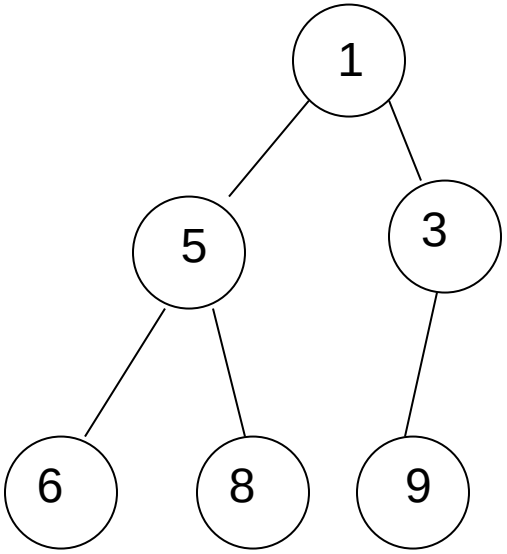
Διαγραφή του τελευταίου κόμβου που είναι άδειος



Ανταλλαγή 8←→1



Ανταλλαγή 8←→5



Είναι σωρός ;;;

Πόσο Κοστίζει η Διαγραφή

O(logn) ?

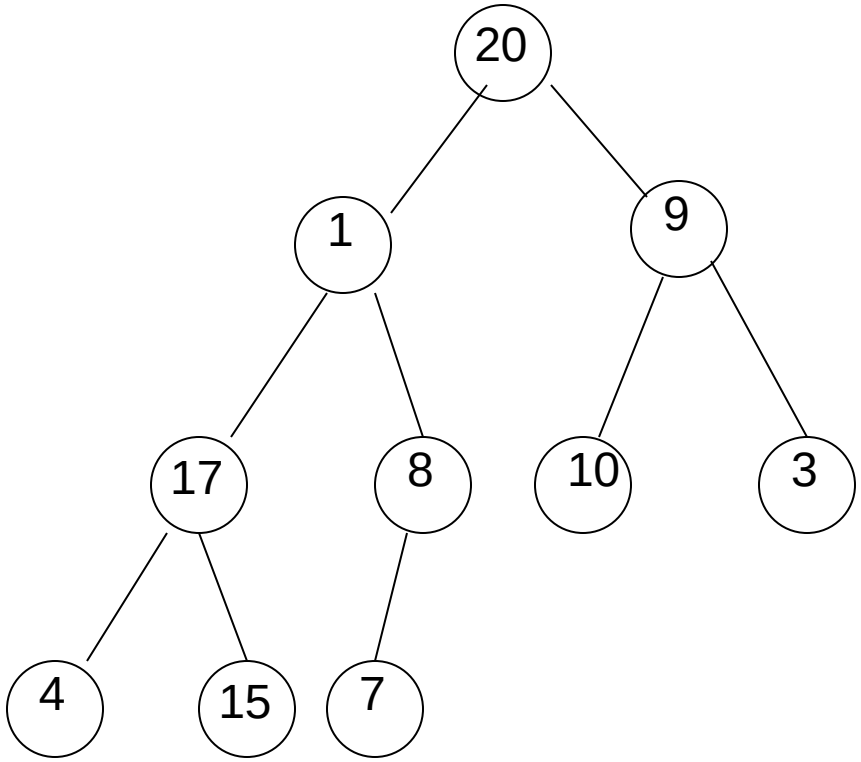
O(loglogn) ?

O(n logn) ?

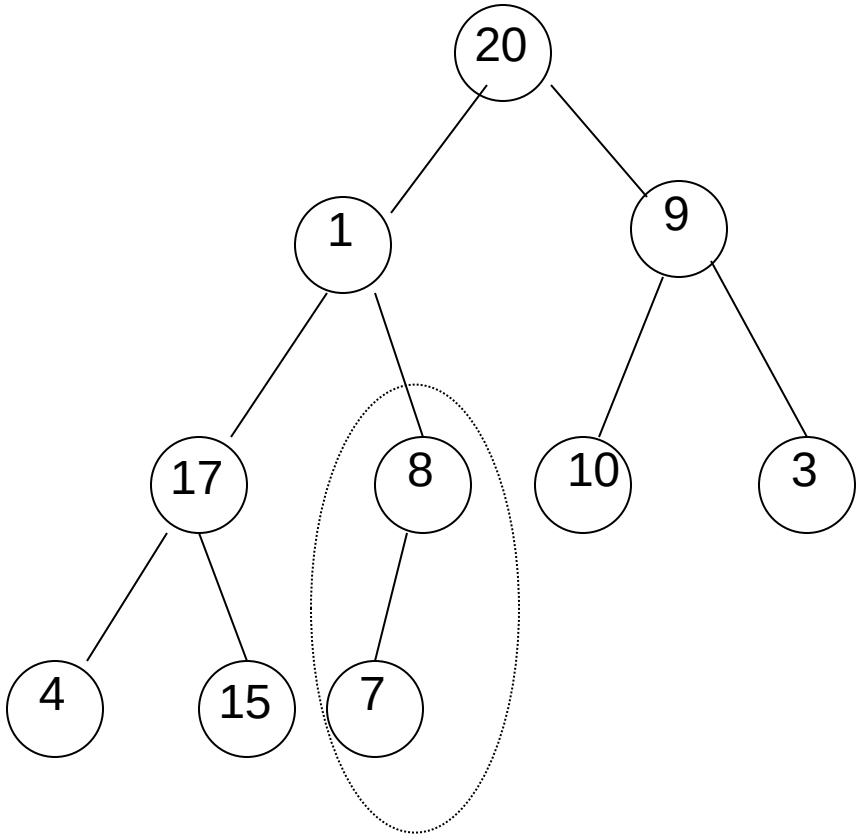
Κατασκευή Σωρού (1/11)

- Πολλές φορές έχουμε ένα σύνολο στοιχείων τα οποία ικανοποιούν τις δομικές απαιτήσεις του σωρού (έχουμε δηλαδή ένα σχεδόν πλήρες δυαδικό δένδρο).
- Πως θα κατασκευάσουμε σωρό ;

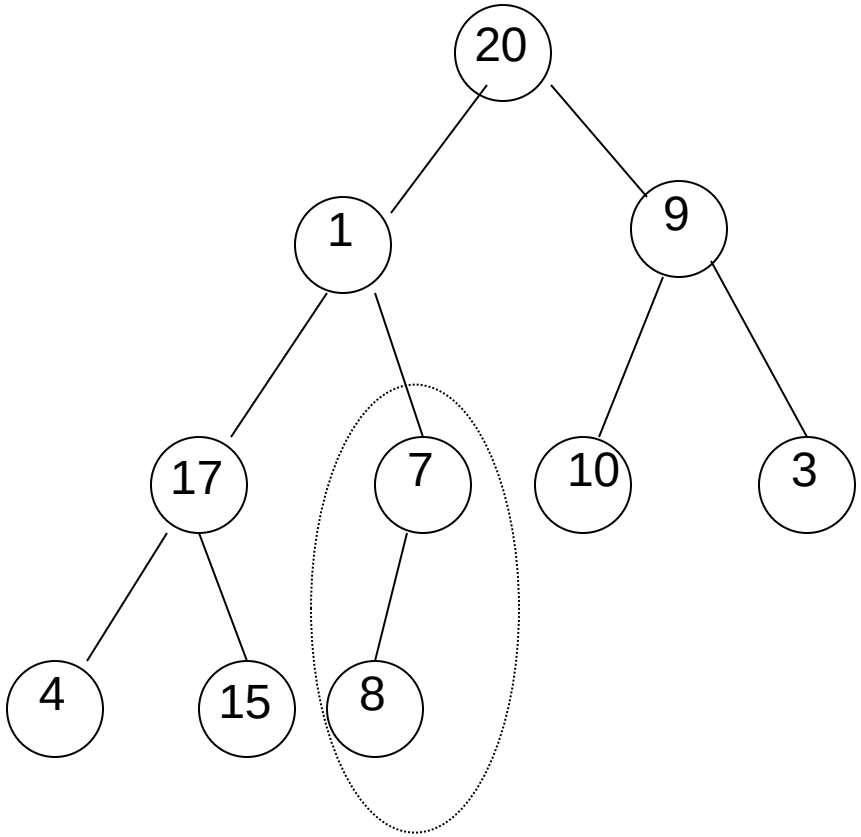
Κατασκευή Σωρού (2/11)



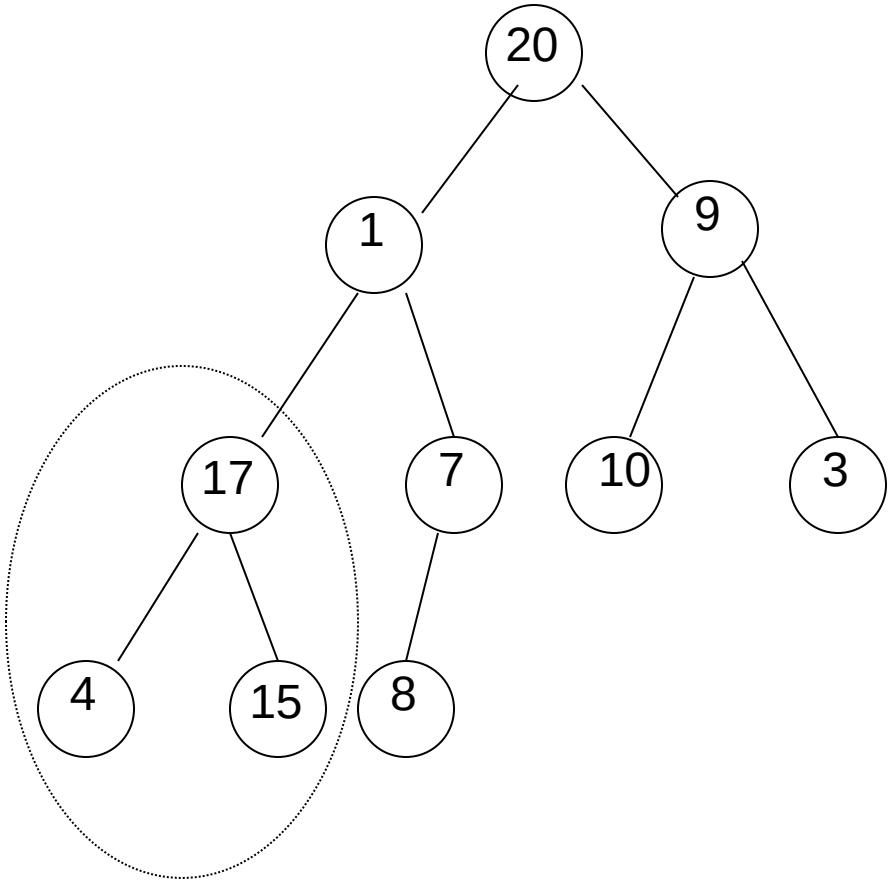
Κατασκευή Σωρού (3/11)



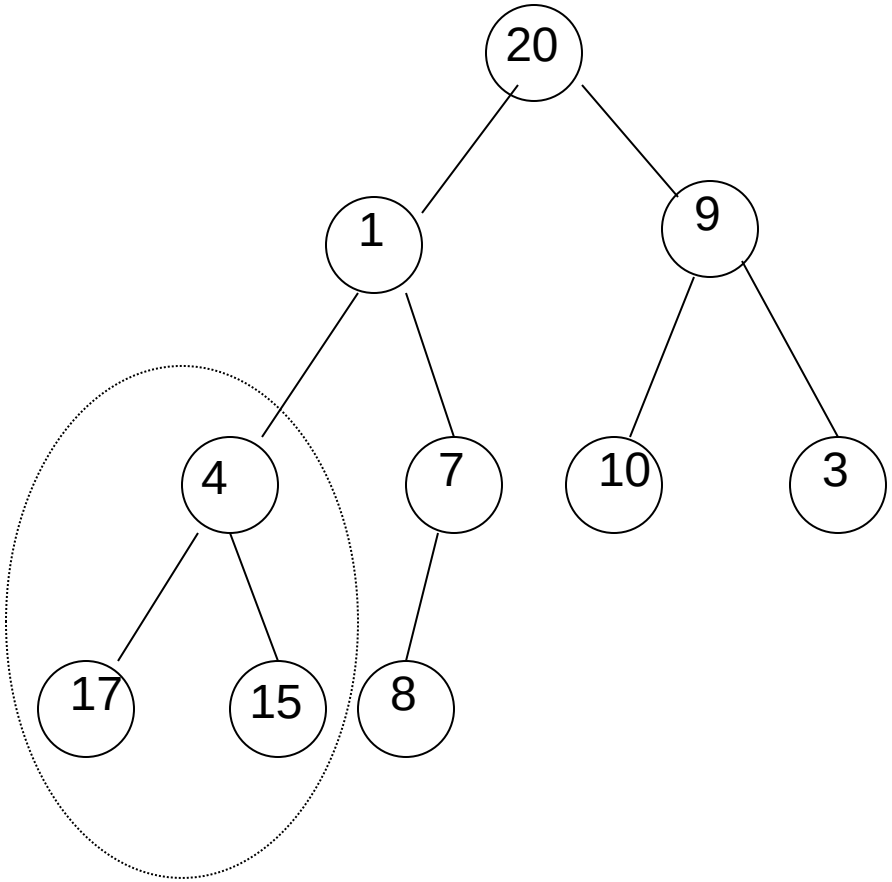
Κατασκευή Σωρού (4/11)



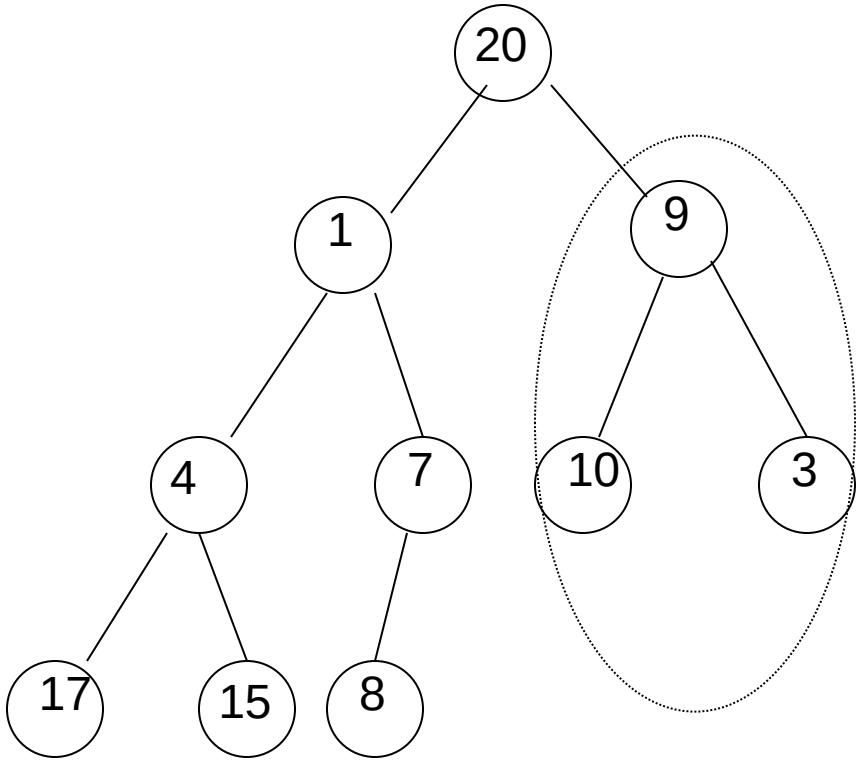
Κατασκευή Σωρού (5/11)



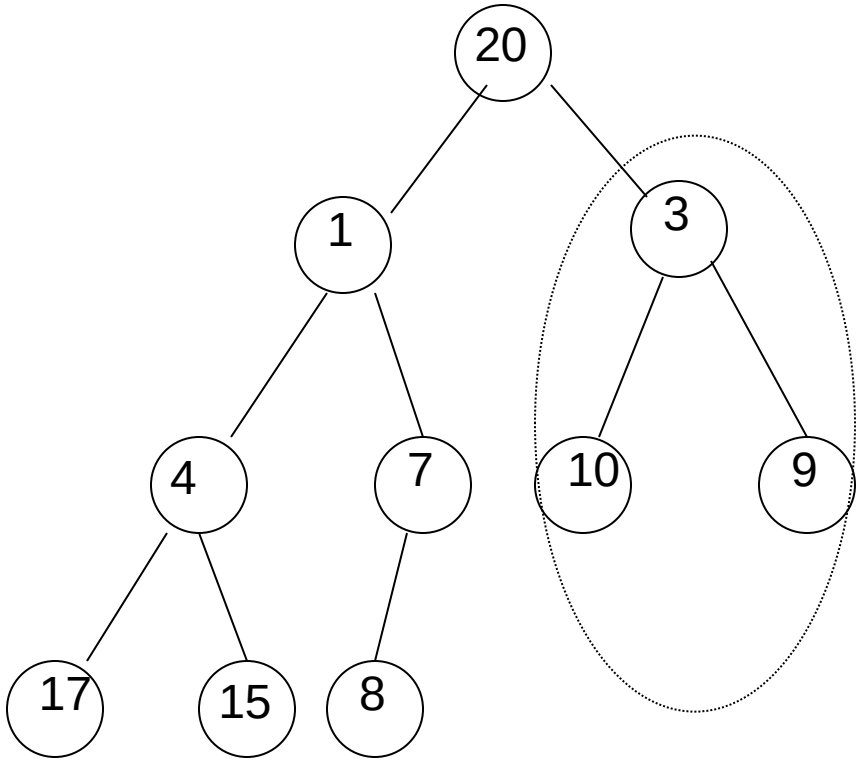
Κατασκευή Σωρού (6/11)



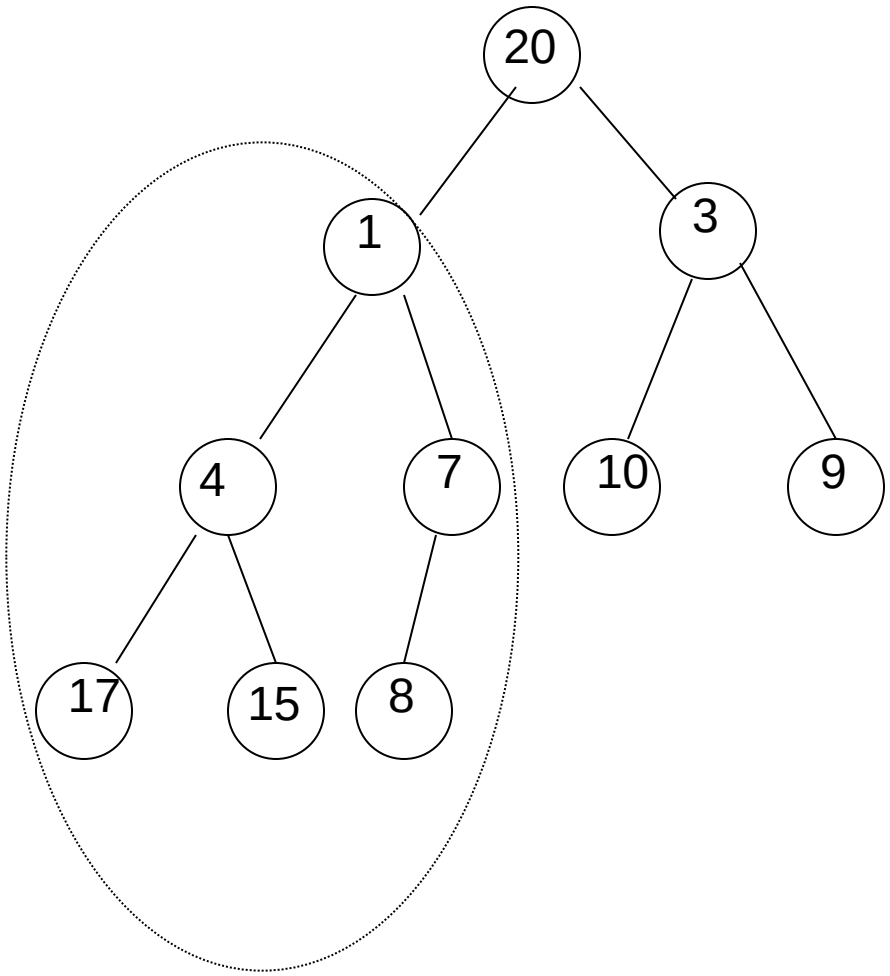
Κατασκευή Σωρού (7/11)



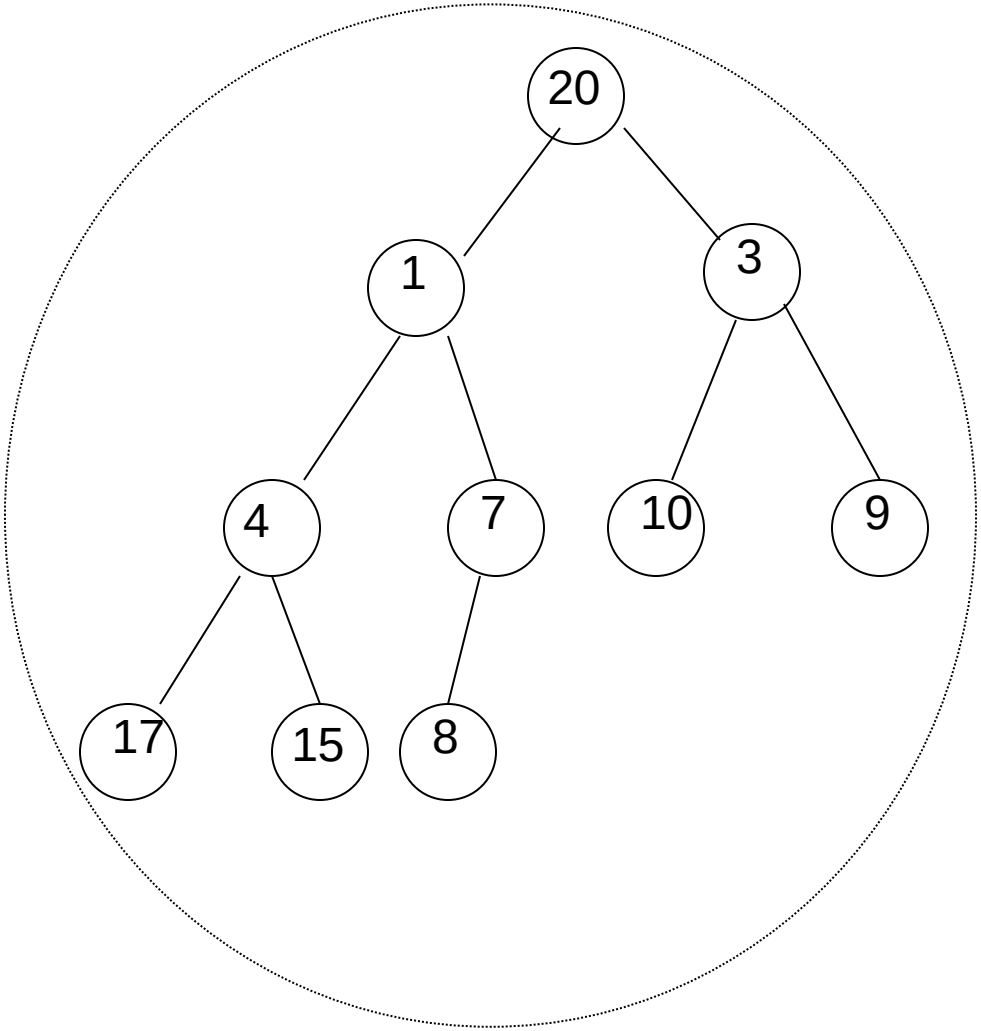
Κατασκευή Σωρού (8/11)



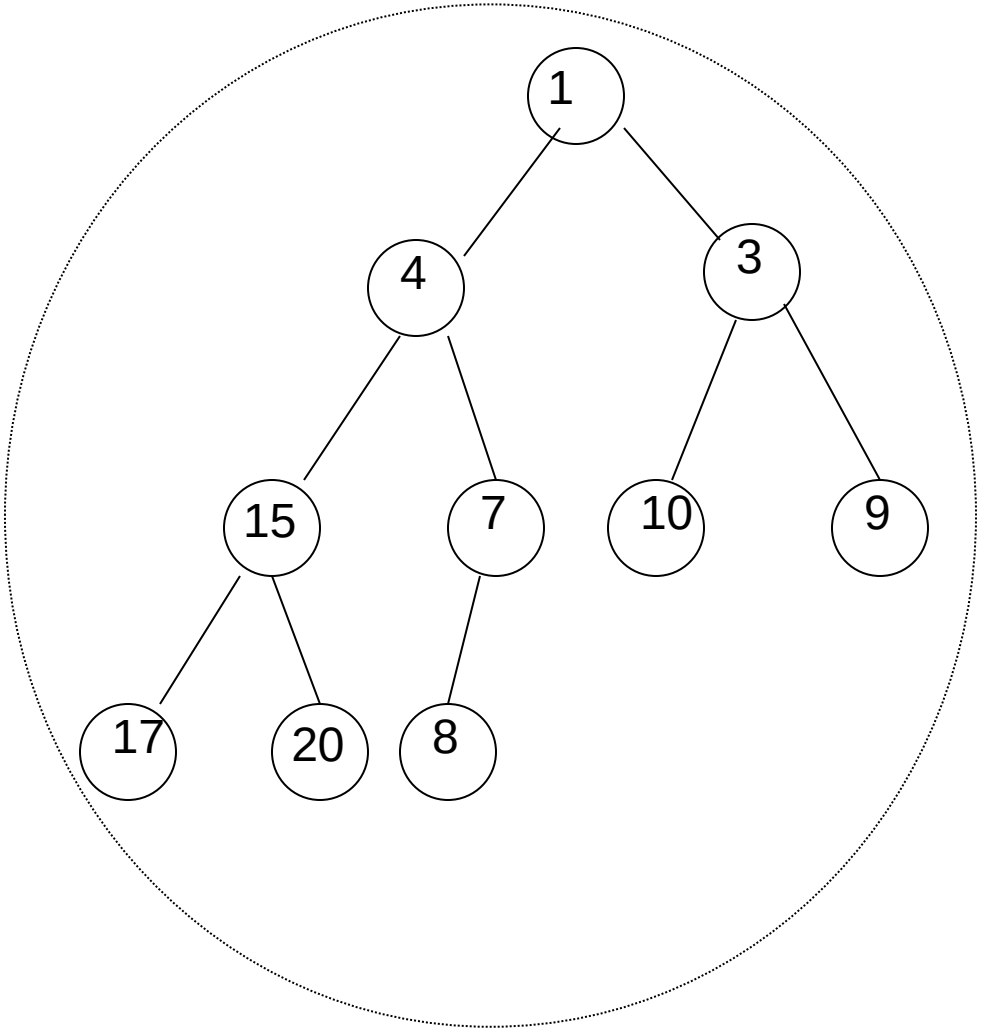
Κατασκευή Σωρού (9/11)



Κατασκευή Σωρού (10/11)



Κατασκευή Σωρού (11/11)



Κόστος Κατασκευής (1/2)

- Απλή εκτίμηση:
  - Αφού στη χειρότερη περίπτωση θα φτάσουμε μέχρι τη ρίζα και τα μισά στοιχεία πρέπει να μετακινηθούν, το συνολικό κόστος είναι
  - n/2 * C * logn = O(n logn)

Κόστος Κατασκευής (2/2)

- Καλύτερη εκτίμηση
  - Κατά την κατασκευή έχουμε το πολύ
    - n/22 στοιχεία που μετακινούνται 1 θέση κάτω
    - n/23 στοιχεία που μετακινούνται 2 θέσεις κάτω
    - n/24 στοιχεία που μετακινούνται 3 θέσεις κάτω
    - … …
  - t(n) = O(1*n/22 + 2*n/23 + 3*n/24 + …)
  - =O((n/2)(1/2 + 2/22 + 3/23 + 4/24 + …)
  - =O(n)

Εφαρμογές Σωρού

- Ταξινόμηση με σωρό (HeapSort).
- Εκτέλεση εργασιών με προτεραιότητες.
- Κωδικοποίηση Huffman.

Heapsort

```
Heapsort(A)
{
        BuildHeap(A);
        for (i = length(A) downto 2)
        {
                Swap(A[1], A[i]);
                heap_size(A) -= 1;
                Heapify(A, 1);
        }
}
```

Ανάλυση Heapsort

- **BuildHeap()** απαιτεί O(n) κόστος.
- Κάθε μία από τις n-1 κλήσεις της **Heapify()** απαιτεί O(log n) χρόνο.
- Συνολικά έχουμε για την **HeapSort()**
  = O(n) + (n - 1) O(log n)
  = O(n) + O(n log n)
  = O(n log n)

Κωδικοποίηση Huffman (1/9)

- Βασικά Σημεία:
  - Ο κύριος στόχος ενός κωδικοποιητή είναι η αντιστοίχιση μικρών κωδικών σε συχνά εμφανιζόμενα σύμβολα και μεγάλων κωδικών σε σπάνια εμφανιζόμενα σύμβολα.
  - Ο χρόνος κωδικοποίησης και αποκωδικοποίησης είναι σημαντικός. Μερικές φορές προτιμούμε να έχουμε μικρότερο λόγο συμπίεσης προκειμένου να κερδίσουμε σε χρόνο (π.χ. WinZIP).

Κωδικοποίηση Huffman (2/9)

Έστω τα σύμβολα A,B,C,D με τους εξής κωδικούς:

Code('A') = 0

Code('B') = 000

Code('C') = 11

Code('D') = 1

DDDAAA

DCB

CDAAA

DDDB

Ο κωδικός 111000 σε ποια σειρά χαρακτήρων αντιστοιχεί;

Κωδικοποίηση Huffman (3/9)

- Βασική προϋπόθεση:
  - Μετά τη φάση της κωδικοποίησης κανένας κωδικός δεν πρέπει να αποτελεί prefix άλλου κωδικού.

Κωδικοποίηση Huffman (4/9)

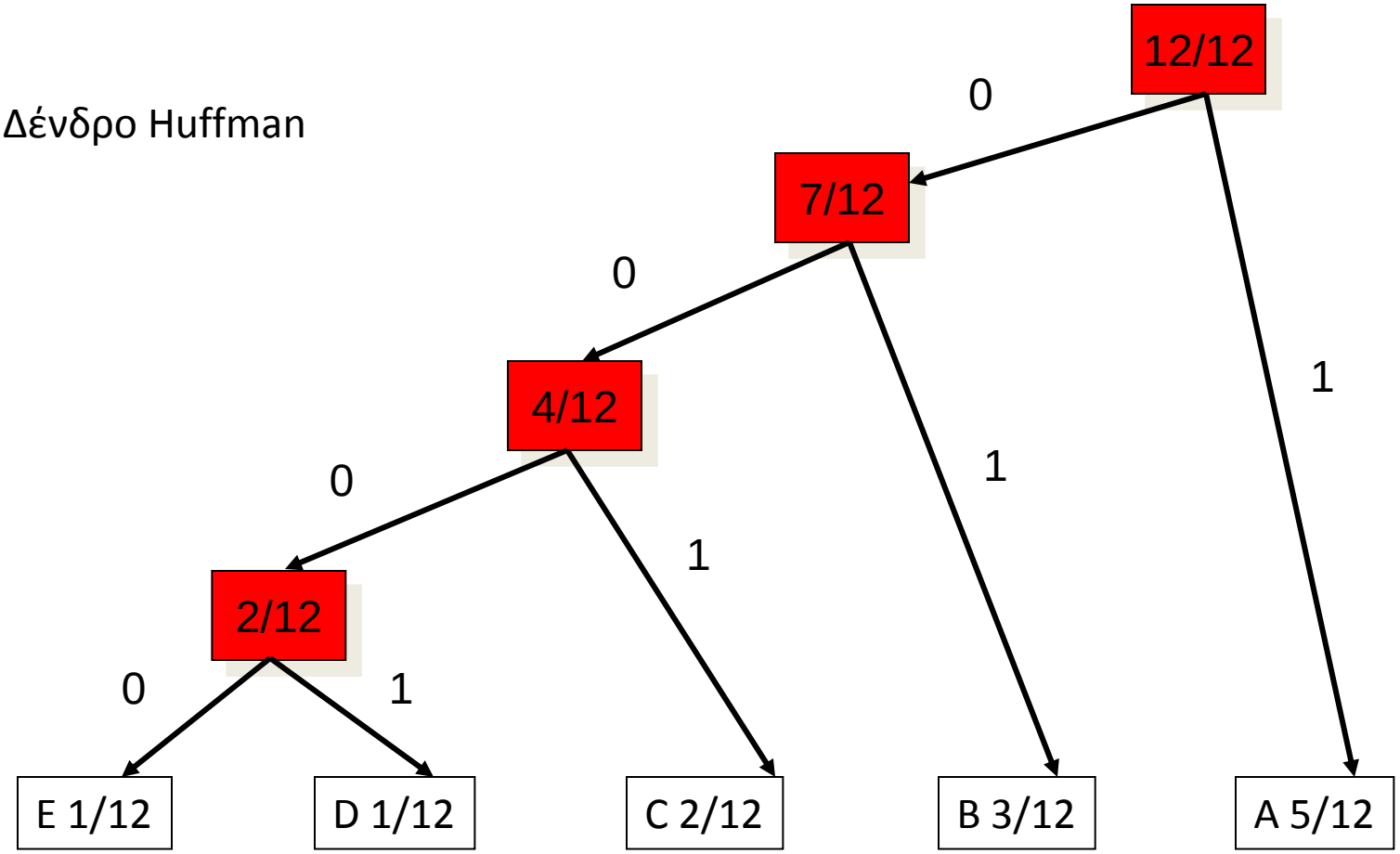
Έστω το ακόλουθο κείμενο:
A B C A B A A A B C D E

A:	5/12
B:	3/12
C:	2/12
D:	1/12
E:	1/12

Συχνότητες εμφάνισης

Κωδικοποίηση Huffman (5/9)

Δένδρο Huffman



Κωδικοποίηση Huffman (6/9)

- Μετά την κωδικοποίηση προκύπτουν οι εξής κωδικοί:

  E: 0000

  D: 0001

  C: 001

  B: 01

  A: 1

- Τι παρατηρούμε;

Κωδικοποίηση Huffman (7/9)

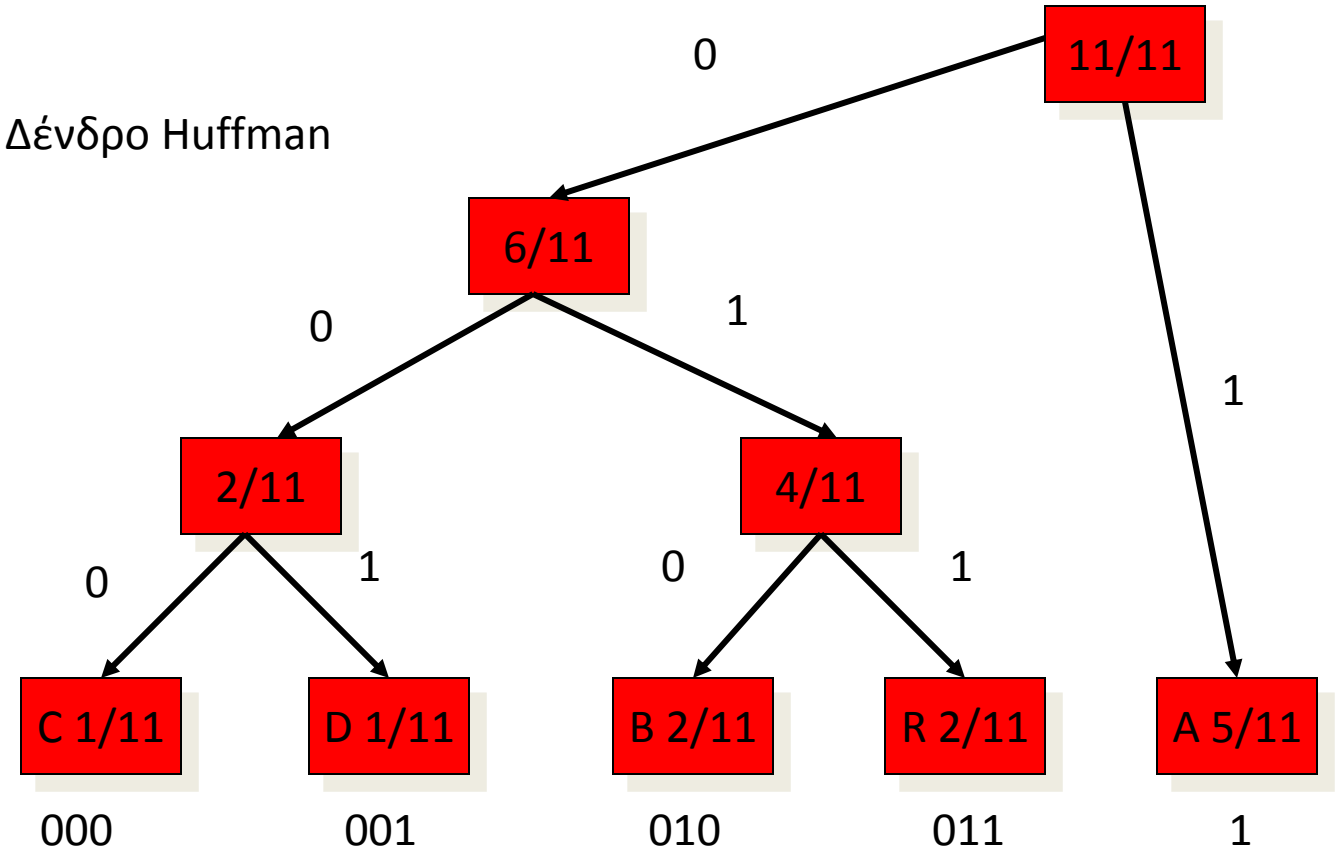
- Τι συμπίεση επιτυγχάνουμε για το παράδειγμα;
- Απαιτούνται 12*8 = 96 bits για το αρχικό κείμενο (χωρίς τους κενούς χαρακτήρες)
- Απαιτούνται 25 bits για το συμπιεσμένο κείμενο

Κωδικοποίηση Huffman (8/9)

- Έστω το ακόλουθο κείμενο
  - ABRACADABRA

A	5/11
B	2/11
C	1/11
D	1/11
R	2/11

Κωδικοποίηση Huffman (9/9)



Σημείωμα Αναφοράς

Σημείωμα Αδειοδότησης

Τέλος ενότητας

Επεξεργασία: <Μαυρίδης Απόστολος>
Θεσσαλονίκη, <Εαρινό εξάμηνο 2013-2014>

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

Σημειώματα

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.