Ουρές Προτεραιότητας

Δημήτρης Φωτάκης

Τμήμα Μηχανικών Πληροφοριακών και Επικοινωνιακών Συστημάτων

Πανεπιστήμιο Αιγαίου

Ουρές Προτεραιότητας (priority queues)

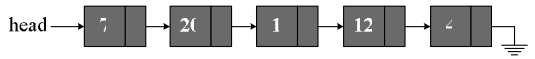
- Ουρά όπου **σειρά διαγραφής** καθορίζεται από **προτεραιότητα** (**μεγαλύτερη** – μικρότερη).
- Στοιχεία (προτεραιότητα, πληροφορία).
- Λειτουργίες:
  - insert(x): εισαγωγή x.
  - deleteMax(): διαγραφή και επιστροφή στοιχείου μέγιστης προτεραιότητας.
  - max(): επιστροφή στοιχείου μέγιστης προτεραιότητας (χωρίς διαγραφή.)
  - changePriority(k): αλλαγή προτεραιότητας θέσης k.
  - isEmpty(), size(): βοηθητικές λειτουργίες.

Εφαρμογές

- Άμεσες εφαρμογές:
  - Υλοποίηση ουρών αναμονής με προτεραιότητες.
    - Δρομολόγηση με προτεραιότητες.
    - Largest (Smallest) Processing Time First.
- Έμμεσες εφαρμογές:
  - Βασικό συστατικό **πολλών** ΔΔ και αλγορίθμων:
    - HeapSort (γενικά ταξινόμηση με επιλογή).
    - Αλγόριθμος Huffman.
    - Αλγόριθμοι Prim και Dijkstra.
    - ...

Αριθμοί σαν Στοιχεία

- Γραμμικές Λίστες, Ουρές, Στοίβες:
  - Αριθμοί δήλωναν **ταυτότητα** στοιχείων (keys).
  - **Καθόλου διάταξη** (<, >).
- Ουρές Προτεραιότητας, ... :
  - Αριθμοί δηλώνουν **ταυτότητα**.
  - Αριθμοί δηλώνουν επίσης **προτεραιότητα.**
  - Χρήση **διάταξης** μικρότερο – μεγαλύτερο!

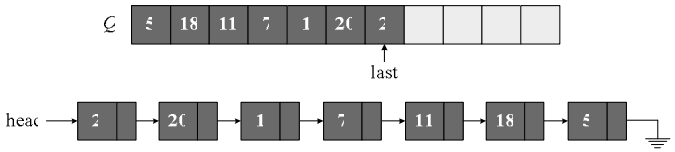


Σχέσεις Ολικής Διάταξης

- Διμελής σχέση $x \le y$:
  - Ανακλαστική : $x \le x$.
  - Αντισυμμετρική : $x \le y$ και $y \le x \Rightarrow x = y$.
  - Μεταβατική : $x \le y$ και $y \le z \Rightarrow x \le z$.
  - **Μερικής διάταξης** : ανακλ., αντισυμμετρ., μεταβατ.
  - **Ολικής διάταξης** : ανακλ., αντισυμμετρ., μεταβατ. και $\forall x, y$ είτε $x \le y$ είτε $y \le x$.
- **Προτεραιότητες** : στοιχεία **κάθε συνόλου** με σχέση **ολικής διάταξης** (αριθμοί, λέξεις, εισοδήματα, ...).

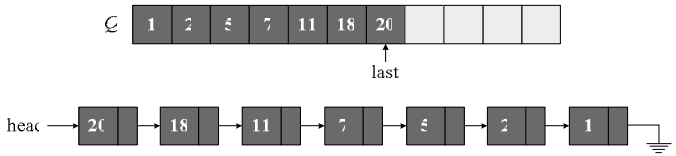
Υλοποίηση με ΑΤΔ Γραμμικής Λίστας

- **Όχι ταξινομημένη** διασυνδεδεμένη λίστα (πίνακας).
- Εισαγωγή στην **αρχή** (διασυνδ. λίστα) / **τέλος** (πίνακας).
- insert: O(1).
- deleteMax, Max: O(*n*).



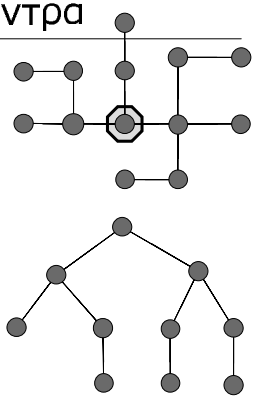
Ταξινομημένη Γραμμική Λίστα

- Εισαγωγή σε **φθίνουσα** (λίστα) / **αύξουσα** (πίνακας) σειρά.
- deleteMax, max: O(*1*).
- insert: Θ(*n*).



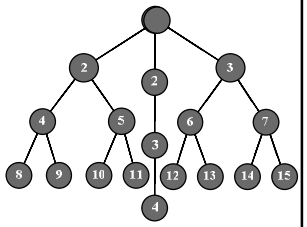
Ιεραρχικές Δομές: Δέντρα

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



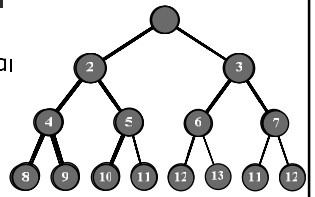
Δυαδικά Δέντρα

- **Ύψος *h*** :
  $h+1 \le$ #κορυφών $\le 2^{h+1} - 1$
  - $h+1$ επίπεδα, ≥ 1 κορ. / επίπ.
  - $\le 2^i$ κορυφές στο επίπεδο i .
    $1 + 2 + \ldots + 2^h = 2^{h+1} - 1$
- **#κορυφών n** :
  $\log_2(n+1) - 1 \le h \le n - 1$
- **Πλήρες** (full) : $n = 2^{h+1} - 1$



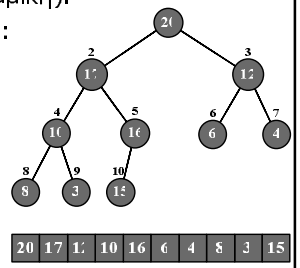
Σχεδόν Πλήρες (complete)

- Όλα τα επίπεδα πλήρη εκτός από τελευταίο που πληρώνεται από αριστερά προς τα δεξιά.
- **Ύψος *h*** :
  $2^h \le n \le 2^{h+1} - 1$
  - Πλήρες(h) : $2^{h+1} - 1$
  - Πλήρες($h-1$) $+ 1$: $(2^h - 1) + 1 = 2^h$.
- **#κορυφών n** :
  $\log_2(n+1) - 1 \le h \le \log_2 n$
- **Ύψος** : $h(n) = \lfloor \log_2 n \rfloor$
- **#φύλλων** $= \lceil n / 2 \rceil$



Αναπαράσταση

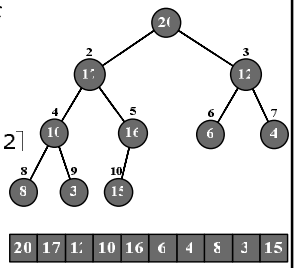
- **Δείκτες** σε παιδιά, πατέρα (δυναμική).
- **Σχεδόν πλήρη** δυαδικά δέντρα :
  - **Πίνακας** (στατική).
  - Αρίθμηση αριστερά → δεξιά και πάνω → κάτω.
  - **Ρίζα** : **Π[1]**
  - **Π[*i*]** : πατέρας **Π[*i* / 2]**
    αριστερό παιδί **Π[2*i*]**
    δεξιό παιδί **Π[2*i*+1]**



20	17	12	10	16	6	4	8	3	15

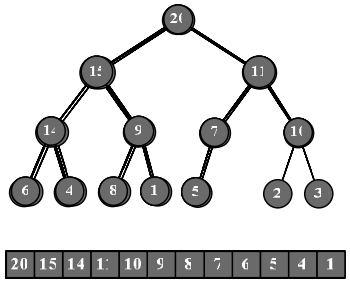
Σωρός (heap)

- Δέντρο **μέγιστου** (ελάχιστου): Τιμές στις κορυφές και τιμή *κάθε* κορυφής $\ge$ ($\le$) τιμές παιδιών της.
- **Σωρός** : σχεδόν πλήρες δυαδικό δέντρο μέγιστου (ελάχιστου).
  - Ύψος $\Theta(\log n)$, #φύλλων $= \lceil n / 2 \rceil$
- Πίνακας **A[ ]** ιδιότ. **σωρού** :
  $\forall i$ A[i] $\ge$ A[$2i$], A[$2i$+1].
- **Μέγιστο** : ρίζα
  Ελάχιστο : φύλλο



20	17	12	10	16	6	4	8	3	15

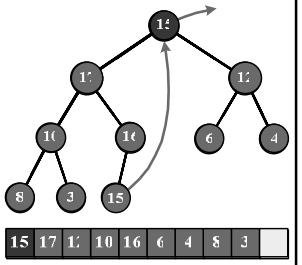
Σωροί και Μη-σωροί



Σωρός σαν Ουρά Προτεραιότητας

- `int A[n], hs;`
- max() : O(1)
  `int max() { return(A[1]); }`
- deleteMax() :

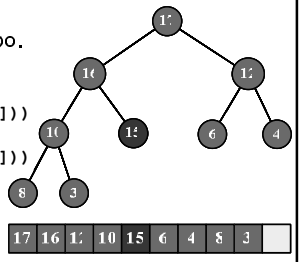
```
int deleteMax() {
  if (isEmpty()) return(EMPTY);
  max = A[1]; A[1] = A[hs--];
  fixHeap(1);
  return(max);  }
```



Αποκατάσταση Προς-τα-Κάτω

- fixHeap(*i*) :
  Ενόσω όχι σωρός,
  - A[i] ↔ max{A[2*i*], A[2*i*+1]}
  - συνεχίζω στο αντίστοιχο υποδέντρο.

```
fixHeap(int i) {
  lc = 2*i; rc = 2*i+1; mc = i;
  if ((lc <= hs) && (A[lc] > A[mc]))
     mc = lc;
  if ((rc <= hs) && (A[rc] > A[mc]))
     mc = rc;
  if (mc != i) {
     swap(A[i], A[mc]);
     fixHeap(mc); } }
```

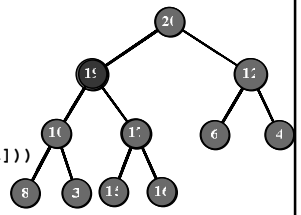


- Χρόνος για deleteMax() : O(ύψος) = O(log n)

Εισαγωγή

- insert(*k*) :
  - Εισαγωγή στο τέλος.
  - Ενόσω όχι σωρός, A[*i*] ↔ A[*i* / 2]

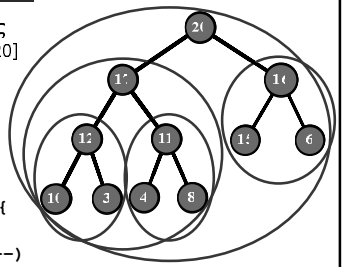
```
insert(int k) {
  A[++hs] = k;
  i = hs; p = i / 2;
  while ((i > 1) && (A[p] < A[i]))
  {    swap(A[p], A[i]);
       i = p; p = i / 2; }  }
```



- Χρόνος για insert() : O(ύψος) = O(log n)
- Αύξηση προτεραιότητας : εισαγωγή (αποκατ. προς-τα-άνω).
  Μείωση προτεραιότητας : διαγραφή (αποκατ. προς-τα-κάτω).

Δημιουργία Σωρού

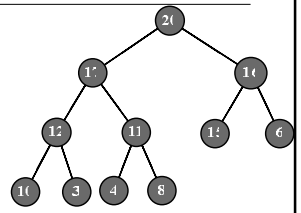
- A[n] → σωρός με n εισαγωγές
  [3, 4, 6, 10, 8, 15, 16, 17, 12, 11, 20]
- Χρόνος O(n log n) .
- **Ιεραρχικά** (bottom-up):
  Υποδέντρα-σωροί ενώνονται σε δέντρο-σωρό.



```
createHeap(int A[], int n) {
   hs = n;
   for (i = n / 2; i > 0; i--)
      fixHeap(A, i);
}
```


Χρόνος Δημιουργίας

```
for (i = n / 2; i > 0; i--)
   fixHeap(A, i);
```



- Χρέωση fixHeap(i) στη θέση i .
  Χρόνος fixHeap(i) = O(ύψος i).
  n / 4 στοιχεία χρόνος O(1)
  n / 8 στοιχεία χρόνος O(2)
  . . . . . .
  $n / 2^k$ στοιχεία χρόνος O(k), $k < \log_2 n$
- $\sum_{k=2}^{\log n} \frac{n}{2^k} O(k) = O\left(n \sum_{k=2}^{\log n} \frac{k}{2^k} \right) = O(n)$, γιατί $\sum_{k=0}^{\infty} \frac{k}{2^k} = 2$
- Χρόνος createHeap( ) = Θ(n).

Απόδοση Σωρού

- Χώρος : Θ(n)
  - In-place για στατικές εφαρμογές.
- Χρόνοι :
  - createHeap : Θ(n)
  - insert, deleteMax : O(log n)
  - max, size, isEmpty : Θ(1)
- Εξαιρετικά εύκολη υλοποίηση!
- Συμπέρασμα:
  - Γρήγορη και ευρύτατα χρησιμοποιούμενη ουρά προτεραιότητας.

Ασκήσεις

- Εύρεση ελάχιστου σε σωρό μέγιστου;
- Ημιάθροισμα μέγιστου και ελάχιστου ανήκει στον πίνακα; Χρόνος εκτέλεσης αν πίνακας (α) δεν είναι ταξινομημένος, και (β) έχει ιδιότητα σωρού.
- Αποκατάσταση ιδιότητας του σωρού αν μεταβληθεί τιμή στοιχείου A[i];
  - Αύξηση στοιχείου:

```
p = i / 2;
while ((i > 1) && (A[p] > A[i])) {
      swap(A[p], A[i]);
   i = p; p = i / 2; }  }
```

  - Μείωση στοιχείου: `fixHeap(A, i);`