Ουρές Προτεραιότητας – Σωροί

Γιάννης Θεοδωρίδης, Νίκος Πελέκης, Άγγελος Πικράκης
Τμήμα Πληροφορικής

Ορισμοί

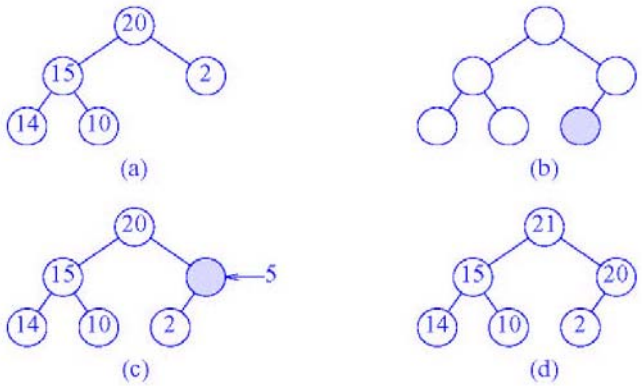
- Ουρά προτεραιότητας: συλλογή από μηδέν ή περισσότερα στοιχεία, όπου το καθένα έχει μια προτεραιότητα ή τιμή.
- Πράξεις πάνω σε μια ουρά προτεραιότητας μεγίστων (ελαχίστων):
  - Εύρεση του στοιχείου με τη μέγιστη (ελάχιστη) προτεραιότητα
  - Διαγραφή του στοιχείου με τη μέγιστη (ελάχιστη) προτεραιότητα
  - Εισαγωγή ενός στοιχείου
- Απλοϊκή αναπαράσταση: ταξινομημένη ή μη γραμμική λίστα
  - Χρόνος εισαγωγής: $\Theta(n)$ ή $\Theta(1)$, αντίστοιχα
  - Χρόνος διαγραφής: $\Theta(1)$ ή $\Theta(n)$, αντίστοιχα

Ορισμοί (συν.)

- Δένδρο μεγίστων (ελαχίστων): ένα δένδρο, όπου η τιμή κάθε κόμβου είναι μεγαλύτερη (μικρότερη) ή ίση των τιμών των παιδιών του
- Σωρός μεγίστων (ελαχίστων): ένα δένδρο μεγίστων (ελαχίστων), το οποίο είναι επίσης συμπληρωμένο δένδρο.
  - Αφού ο σωρός είναι συμπληρωμένο δένδρο, συνήθως χρησιμοποιούμε την αναπαράσταση πίνακα

(στη συνέχεια, θα θεωρήσουμε μόνο δυαδικά δένδρα)

Εισαγωγή σε σωρό μεγίστων (1)



(a) αρχική κατάσταση σωρού, (b) η θέση του νέου κόμβου

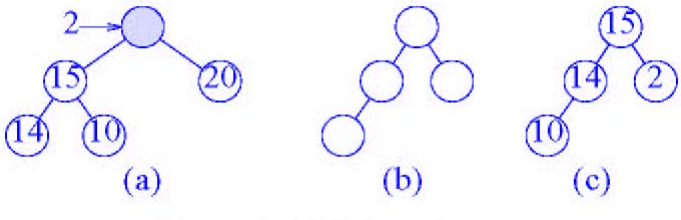
(c) ή (d) η τελική κατάσταση του σωρού μετά την εισαγωγή του 5 ή 21, αντίστοιχα

Κόστος = O(h) = O(logn)

Εισαγωγή σε σωρό μεγίστων (2)

```
MaxHeap<T>& MaxHeap<T>::Insert(const T& x)
{// Insert x into the max heap.
   if (CurrentSize == MaxSize)
      throw NoMem(); // no space

   // find place for x
   // i starts at new leaf and moves up tree
   int i = ++CurrentSize;
   while (i != 1 && x > heap[i/2]) {
      // cannot put x in heap[i]
      heap[i] = heap[i/2]; // move element down
      i /= 2;              // move to parent
      }

   heap[i] = x;
   return *this;
}
```


Διαγραφή από σωρό μεγίστων (1)



(a) μετά τη διαγραφή του 21, το τελευταίο στοιχείο (2) τοποθετείται προσωρινά στη ρίζα και ακολουθεί τακτοποίηση σωρού (αντιμετάθεση του 2 με το 20)

(b-c) μετά τη διαγραφή του 21, το τελευταίο στοιχείο (2) τοποθετείται προσωρινά στη ρίζα και ακολουθεί τακτοποίηση σωρού (αντιμετάθεση του 2 με το 15)

Κόστος = O(h) = O(logn)

Διαγραφή από σωρό μεγίστων (2)

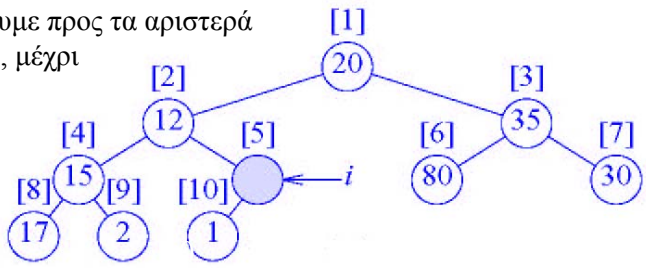
```
MaxHeap<T>& MaxHeap<T>::DeleteMax(T& x)
{// Set x to max element and delete
 // max element from heap.
   // check if heap is empty
   if (CurrentSize == 0)
      throw OutOfBounds(); // empty

   x = heap[1]; // max element

   // restucture heap
   T y = heap[CurrentSize--]; // last element

   // find place for y starting at root
   ...
```


Διαγραφή από σωρό μεγίστων (3)

```
...
   // find place for y starting at root
   int i = 1,  // current node of heap
       ci = 2; // child of i
   while (ci <= CurrentSize) {
      // heap[ci] should be largest child of i
      if (ci < CurrentSize &&
          heap[ci] < heap[ci+1]) ci++;
      // can we put y in heap[i]?
      if (y >= heap[ci]) break;   // yes
      // no
      heap[i] = heap[ci]; // move child up
      i = ci; ci *= 2;    // move down a level
      }
   heap[i] = y;
   return *this;
}
```

Αρχικοποίηση σωρού μεγίστων (1)

- Έστω ότι n>0 στοιχεία έχουν τοποθετηθεί σε ένα συμπληρωμένο δυαδικό δένδρο, το οποίο πρέπει να μετατρέψουμε σε σωρό
  - Ξεκινάμε από το τελευταίο στοιχείο που έχει παιδί(-ά) και τακτοποιούμε, αν χρειάζεται, το υποδένδρο από εκεί και κάτω να είναι σωρός
  - Συνεχίζουμε προς τα αριστερά και πάνω, μέχρι τη ρίζα



Αρχικοποίηση σωρού μεγίστων (2)

```
void MaxHeap<T>::Initialize(T a[], int size,
                            int ArraySize)
{// Initialize max heap to array a.
   delete [] heap;
   heap = a;
   CurrentSize = size;
   MaxSize = ArraySize;

   // make into a max heap
   for (int i = CurrentSize/2; i >= 1; i--) {
      T y = heap[i]; // root of subtree

      // find place to put y
      int c = 2*i; // parent of c is target
                   // location for y
      ...
```

Αρχικοποίηση σωρού μεγίστων (3)

```
...
      while (c <= CurrentSize) {
         // heap[c] should be larger sibling
         if (c < CurrentSize &&
             heap[c] < heap[c+1]) c++;

         // can we put y in heap[c/2]?
         if (y >= heap[c]) break;  // yes

         // no
         heap[c/2] = heap[c]; // move child up
         c *= 2; // move down a level
         }
      heap[c/2] = y;
      }
}
```

Κόστος = Θ(n)