ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

Δομές Δεδομένων (Εργ.)
Ακ. Έτος 2018-19
Διδάσκων: Ευάγγελος Σπύρου

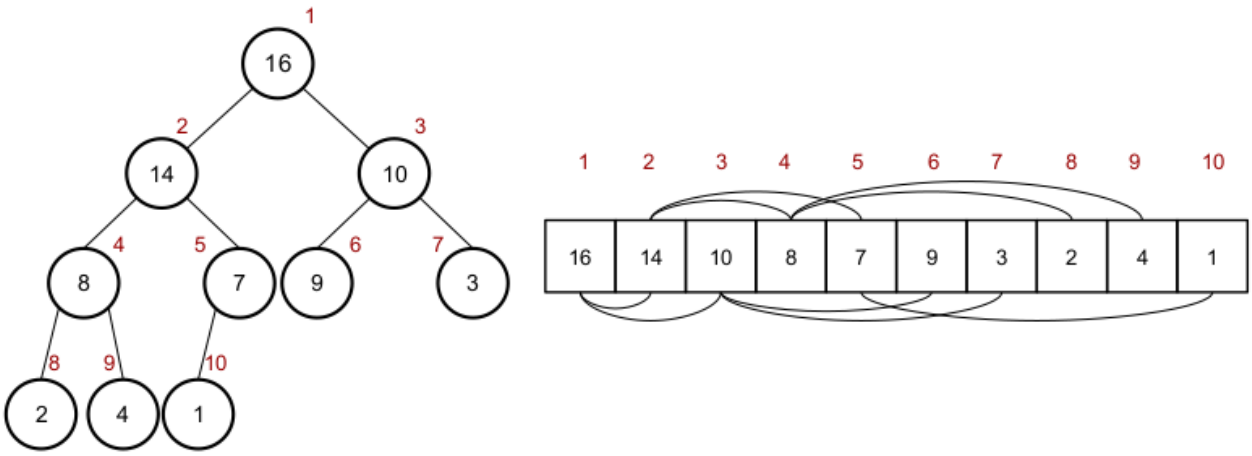
Εργαστήριο 6 – Ταξινόμηση Σωρού

1. Στόχος του εργαστηρίου

Στόχος του έκτου εργαστηρίου είναι η κατανόηση του αλγορίθμου της ταξινόμησης σωρού (heapsort) η υλοποίηση του σε γλώσσα C, καθώς και η εφαρμογή του σε ένα απλό πρόβλημα.

2. Ταξινόμηση σωρού

Η δομή δεδομένων Δυαδικός Σωρός είναι ένα αντικείμενο τύπου συστοιχίας, το οποίο μπορεί να θεωρηθεί ως ένα πλήρες δυαδικό δένδρο. Κάθε κόμβος του δένδρου αντιστοιχεί σε ένα στοιχείο της συστοιχίας και περιέχει την τιμή του. Το δένδρο είναι συμπληρωμένο σε όλα τα επίπεδα, εκτός ίσως από το χαμηλότερο.



Για μια δεδομένη θέση i ενός κόμβου, οι θέσεις του πατρικού, του αριστερού θυγατρικού και του δεξιού θυγατρικού υπολογίζονται ως:

- Parent(i)=$\lfloor i/2 \rfloor$
- Left(i)=$2 \cdot i$
- Right(i)=$2 \cdot i + 1$

Υπάρχουν δύο τύποι δυαδικού σωρού:

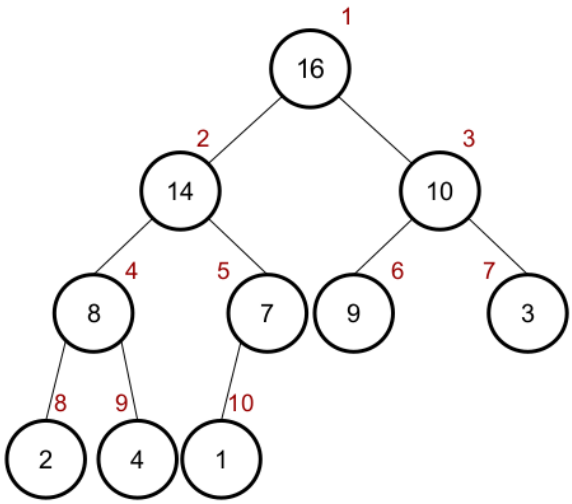
- Σωροί μεγίστου
- Σωροί ελαχίστου

Σε κάθε τύπο, οι τιμές στους κόμβους ικανοποιούν μια ιδιότητα:

- σωρός μεγίστου: για κάθε κόμβο i, εκτός της ρίζας, A[Parent(i)] ≥ A[i]. Άρα η τιμή οποιουδήποτε κόμβου είναι το πολύ ίση με την τιμή του πατρικού του στοιχείου. Η ρίζα περιέχει το μεγαλύτερο στοιχείο!

- σωρός ελαχίστου: για κάθε κόμβο i εκτός της ρίζας, $A[\text{Parent}(i)] \leq A[i]$. Άρα για κάθε κόμβο, η τιμή του πατρικού του στοιχείου είναι μικρότερη ή ίση από την τιμή του. Η ρίζα περιέχει το μικρότερο στοιχείο!

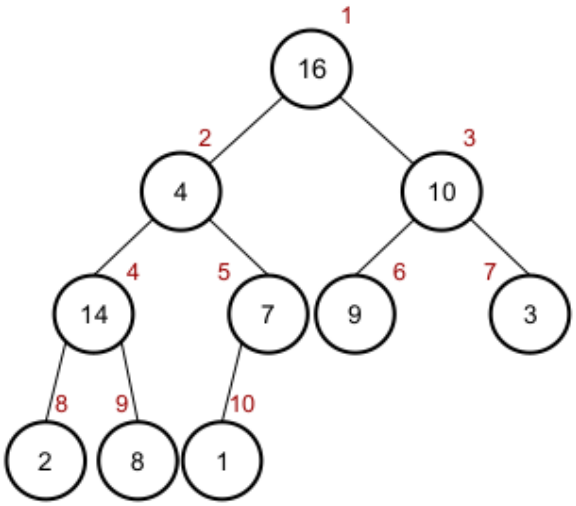
Σε αυτή την άσκηση θα χρησιμοποιήσαμε σωρούς μεγίστου για αύξουσα ταξινόμηση. Ένα παράδειγμα σωρού μεγίστου είναι το εξής:



- Στη συνέχεια, θα περιγράψουμε ορισμένες διαδικασίες και θα αναλύσουμε τη χρήση τους σε έναν αλγόριθμο ταξινόμησης, τον οποίο και θα υλοποιήσουμε:Αποκατάσταση σωρού μεγίστου (max-heapify)
- Κατασκευή σωρού μεγίστου (build-max-heap)
- Ταξινόμηση σωρού (heapsort)

Αποκατάσταση σωρού μεγίστου

Η διαδικασία της αποκατάστασης σωρού μεγίστου δέχεται σαν είσοδο μια συστοιχία A, το μέγεθός της n και τη θέση i του στοιχείου της συστοιχίας. Η λειτουργία της προϋποθέτει ότι τα δυαδικά δένδρα που έχουν ρίζες τους κόμβους Left(i), Right(i) **είναι ήδη** σωροί μεγίστου. Το $A[i]$ ενδέχεται να είναι μικρότερο από τους θυγατρικούς του κόμβους, δηλαδή να παραβιάζει την ιδιότητα σωρού μεγίστου. Η διαδικασία αυτή μεταφέρει την τιμή του $A(i)$ **προς τα κάτω** στο σωρό, έτσι ώστε το υποδένδρο με ρίζα το στοιχείο στη θέση i να καταστεί σωρός μεγίστου. Δείτε π.χ. στο παρακάτω σχήμα:



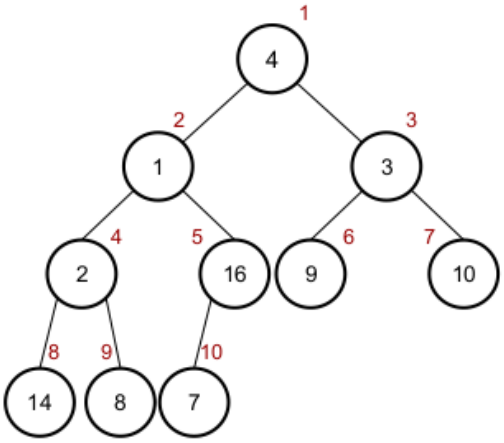
Αρχικά, το στοιχείο *A*[*2*] στον κόμβο *i = 2* παραβιάζει την ιδιότητα του σωρού μεγίστου (είναι μικρότερο των θυγατρικών του). Δεν φαίνεται να παραβιάζεται η ιδιότητα από άλλον κόμβο. Ο αλγόριθμος της αποκατάστασης σωρού μεγίστου θα αποκαταστήσει αυτή την παραβίαση. Ωστόσο, ενδέχεται να προκύψει εκ νέου, σε άλλο κόμβο. Ο αλγόριθμος λειτουργεί ως εξής: Αρχικά βρίσκει ποιος από τους κόμβους *i*, Left(*i*) και Right(*i*) έχει τη μεγαλύτερη τιμή. Αν την έχει ο *i*, τότε αφού θεωρούμε ως δεδομένο ότι τα υποδένδρα με ρίζες τους Left(*i*), Right(i) είναι ήδη σωροί μεγίστου, τότε και το υποδένδρο με ρίζα τον *i*, θα είναι σωρός μεγίστου. Αν όμως δεν την έχει ο *i*, παραβιάζεται η ιδιότητα του σωρού μεγίστου και εναλλάσσεται η τιμή του *i* με τη μεγαλύτερη από τις δύο τιμές των παιδιών του. Η διαδικασία αυτή συνεχίζεται αναδρομικά και όταν περατωθεί, το υποδένδρο με ρίζα τον *i* θα είναι πλέον σωρός μεγίστου.

Δοκιμάστε να τρέξετε το προηγούμενο παράδειγμα με το χέρι, εώς ότου να τερματισθεί η αναδρομή.

Κατασκευή Σωρού

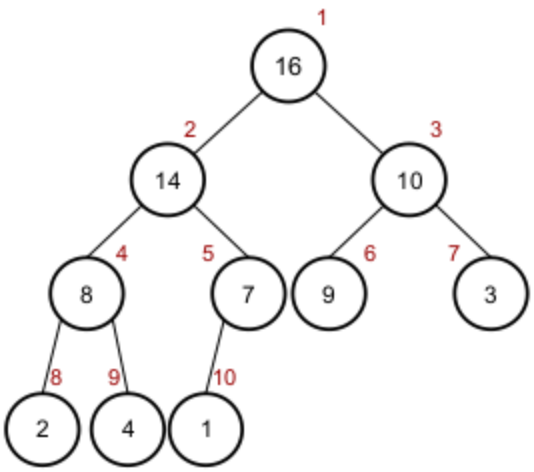
Η διαδικασία της αποκατάστασης που είδαμε μπορεί να χρησιμοποιηθεί για τη μετατροπή μιας οποιασδήποτε συστοιχίας Α σε σωρό μεγίστου. Για την υλοποίηση πρέπει να σκεφτούμε ως εξής: Τα στοιχεία [(⌊*n/2*⌋ + *1*) . . . *n*] είναι όλα καταληκτικοί κόμβοι του δένδρου, το καθένα είναι αρχικά ένας σωρός με ένα μέλος, άρα και σωρός μεγίστου! Αρκεί η διαδικασία της κατασκευής να διέλθει από τους υπόλοιπους κόμβους του δένδρου και να εκτελέσει αποκατάσταση σε κάθε έναν από αυτούς

Δοκιμάστε να τρέξετε τη διαδικασία αυτή στον ακόλουθο σωρό, με το χέρι, εώς ότου να μετατραπεί σε σωρό μεγίστου:

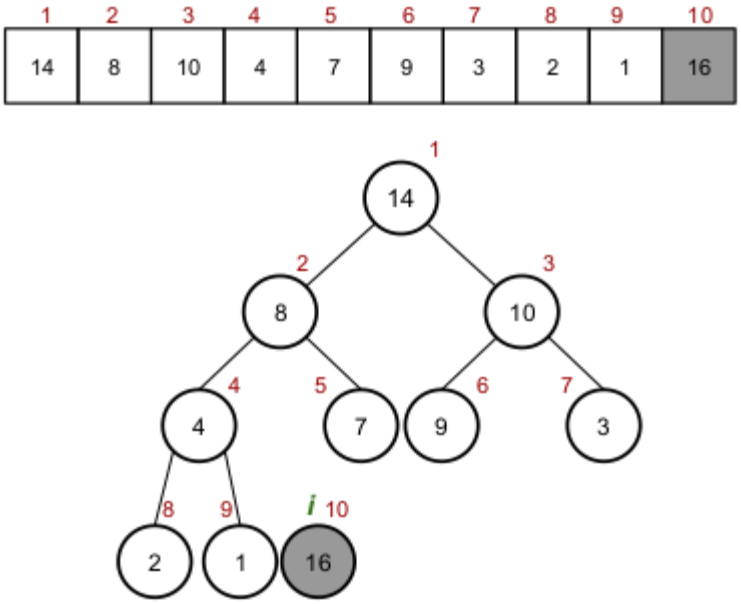


Ταξινόμηση Σωρού

Ο αλγόριθμος της ταξινόμησης σωρού αρχικά κατασκευάζει έναν σωρό μεγίστου και στη συνέχεια εναλλάσσει το πρώτο στοιχείο (ρίζα του σωρού) το οποίο και έχει τη μεγαλύτερη τιμή (ιδιότητα σωρού μεγίστου) με το τελευταίο στοιχείο του πίνακα. Έτσι, επειδή η ταξινόμηση είναι άυξουσα, το τελευταίο στοιχείο του πίνακα έχει τη σωστή τιμή. Στη συνέχεια, καθώς ο σωρός δεν είναι πλέον σωρός μεγίστου (έπειτα από την εναλλαγή), καλεί τη διαδικασία της αποκατάστασης. Αυτό επαναλαμβάνεται εώς ότου η συστοιχία ταξινομηθεί. Π.χ. έστω ο ακόλουθος σωρός μεγίστου:



Σε κάθε επανάληψη, η ρίζα “αφαιρείται” – εναλλάσσεται με το πρώτο στοιχείο της συστοιχίας και στη συνέχεια η διαδικασία της αποκατάστασης καλείται για την νέα υποσυστοιχία. Π.χ. το πρώτο βήμα είναι το ακόλουθο:



Δοκιμάστε να τρέξετε τη διαδικασία αυτή, με το χέρι, εώς ότου να ταξινομηθεί η συστοιχία.

4. Υλοποίηση Ταξινόμησης Σωρού

Να υλοποιηθεί η ταξινόμηση σωρού, υλοποιώντας τις διαδικασίες που περιγράφηκαν προηγουμένως. Πιο συγκεκριμένα, να υλοποιήσετε τις ακόλουθες συναρτήσεις:

- `void max_heapify(int A[], int i, int n);` (αποκατάσταση)
- `void build_max_heap(int A[], int n);` (κατασκευή)
- `void heapsort(int A[], int n);` (ταξινόμηση)

Να υλοποιήσετε επίσης όσες βοηθητικές συναρτήσεις χρειαστείτε και στη `main` να τρέξετε το προηγούμενο παράδειγμα, τυπώνοντας όλα τα αναγκαία ενδιάμεσα αποτελέσματα.