cis112
Tree- Heap

Haluk O. Bingol

Faculty of Computer and Information Sciences
Yeditepe University

v2024-05-12

Content

[[1], [2], [3], [4], [5], [6], [7]]

Motivation

Hierarchy

Definitions

[[1], [2], [3], [4], [5], [6], [7]]

Balanced Binary Trees

Definition

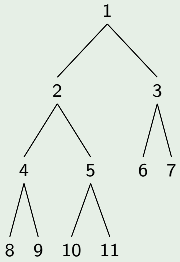
A binary tree $T = \{r, T_l, T_r\}$ is balanced iff

- T is empty or
- T_l, T_r have "almost the same height".

Remark.

- Consider "almost the same height" as difference of 1.
- How to keep a tree balanced is an important issue that we will deal with

Example



Max-Heap Property

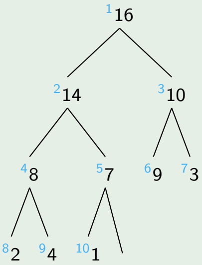
Definition

An binary tree T has Max-Heap Property iff

$$\texttt{i.parent.key} \geq \texttt{i.key}$$

for all nodes i.
A binary tree with max-heap property is called max-heap.

Example



index	1	2	3	4	5	6	7	8	9	10
Value	16	14	10	8	7	9	3	2	4	1

Priority Queue

Definition (Mathematical)

A priority queue is a data structure of a set S of elements x, each has a key `x.key`.
A max-priority queue has the following operations

1. `insert(S, x)` inserts the element `x`
2. `maximum(S)` returns the element of `S` with the largest `key`
3. `extractMax(S)` removes and returns the element of `S` with the largest `key`

Example

Q. What happens if there are more than one element with the largest `key`?

Implementation

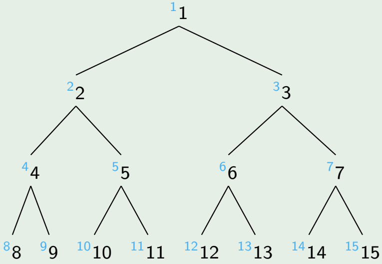
Array-Like Approach

A heap can be implemented as an array (array-like structures).
For node at index i

- Starting index 1, i.e., not using index 0
  - $parent = \lfloor i/2 \rfloor$
  - $left = 2i$
  - $right = 2i + 1$
- Using index 0
  - $parent = \lfloor (i-1)/2 \rfloor$
  - $left = 2i + 1$
  - $right = 2i + 2$

Example

Starting at index 1.



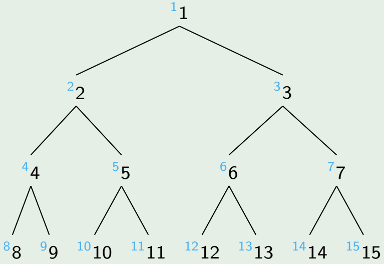
index	0	1	2	3	4	5	6	7	8	9	...
Value	×	1	2	3	4	5	6	7	8	9	...

"left-justified"

- Use array-like implementation.
- For a tree with N nodes, use indices $1, 2, \ldots, N$

Example

Starting at index 1.



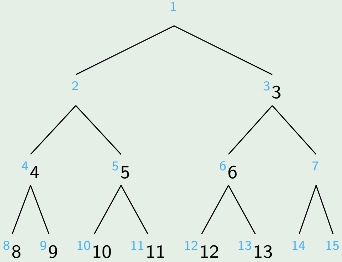
index	0	1	2	3	4	5	6	7	8	9	...
Value	×	1	2	3	4	5	6	7	8	9	...

Observations

- insert a new node x at the first available location, e.g., index 14
- remove a node, the first available location becomes one less index (move to the left), e.g., index 13 becomes empty

Example

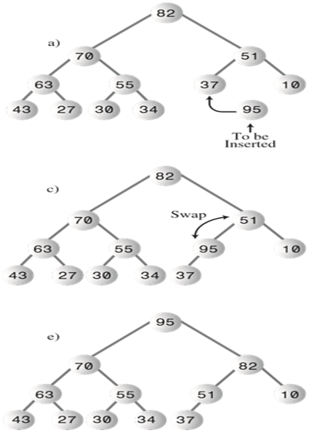
Starting at index 1.



index	0	1	2	3	4	5	6	7	8	9	...
Value	×	1	2	3	4	5	6	7	8	9	...

Insert

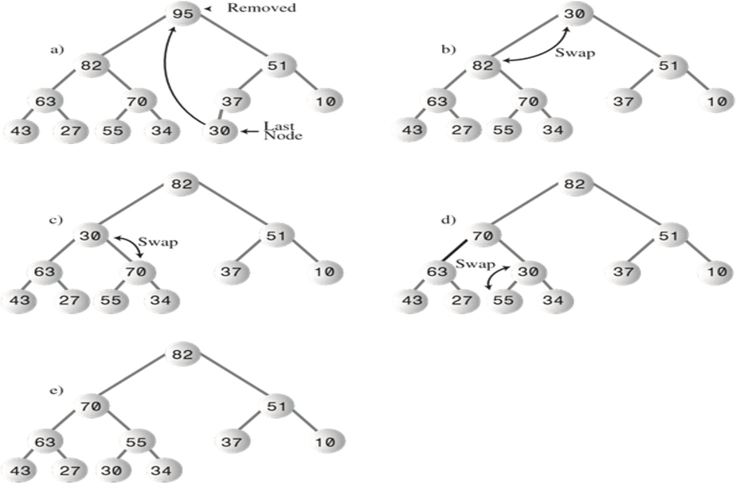
insert x to the first available location and move it up



[[8]]

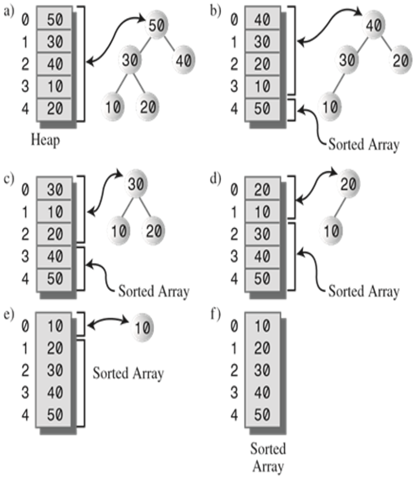
Delete

Delete the root, move the last node to the root and move it down



[[8]]

Heapsort


a)
0 1 2 3 4
50 30 40 10 20
Heap
b)
40 30 20 10 50
Sorted Array
c)
30 10 20 40 50
Sorted Array
d)
20 10 30 40 50
Sorted Array
e)
10 20 30 40 50
Sorted Array
f)
10 20 30 40 50
Sorted Array

[[8]]

References I

[1] D. E. Knuth, *The Art of Computer Programming: Fundamental Algorithms, volume 1*, 2nd ed. Addison-Wesley Professional, 1973.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms (CLRS)*, 4th ed. MIT Press, 2022.

[3] B. Preiss, *Data Structures and Algorithms with Object-Oriented Design Patterns in Java*. Wiley, 1999.

[4] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *The Design and Analysis of Computer Algorithms*. Addison Wesley, 1974.

[5] E. Horowitz and S. Sahni, *Fundamentals of Data Structures*. Pitman, 1982.

[6] C. S. Horstmann, *Big Java: Early Objects*, 7th ed. John Wiley & Sons, 2019.

[7] R. P. Grimaldi, *Discrete and Combinatorial Mathematics*, 5th ed. Pearson Education India, 2006.

[8] R. Lafore, *Data structures and algorithms in Java*. Sams publishing, 2003.