Notes lecture 8

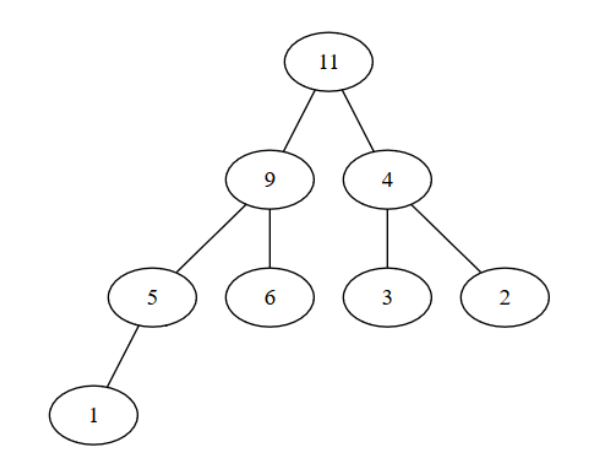
*Binary Heap*

* Add:
  + add to end.
  + bubble-up if needed.
* Remove:
  + the root is removed.
  + the last element is brought up as root.
  + the new root is bubbled-down if needed.
* Bubble-up:
  + new node will be swapped with its parent until it gets to its final place.
* Bubble-down:
  + new node will be swapped with its maximum child (in the case of MAX-HEAP), until it gets to its final place. (becomes a leaf, or is greater than both children)

*Heapsort*

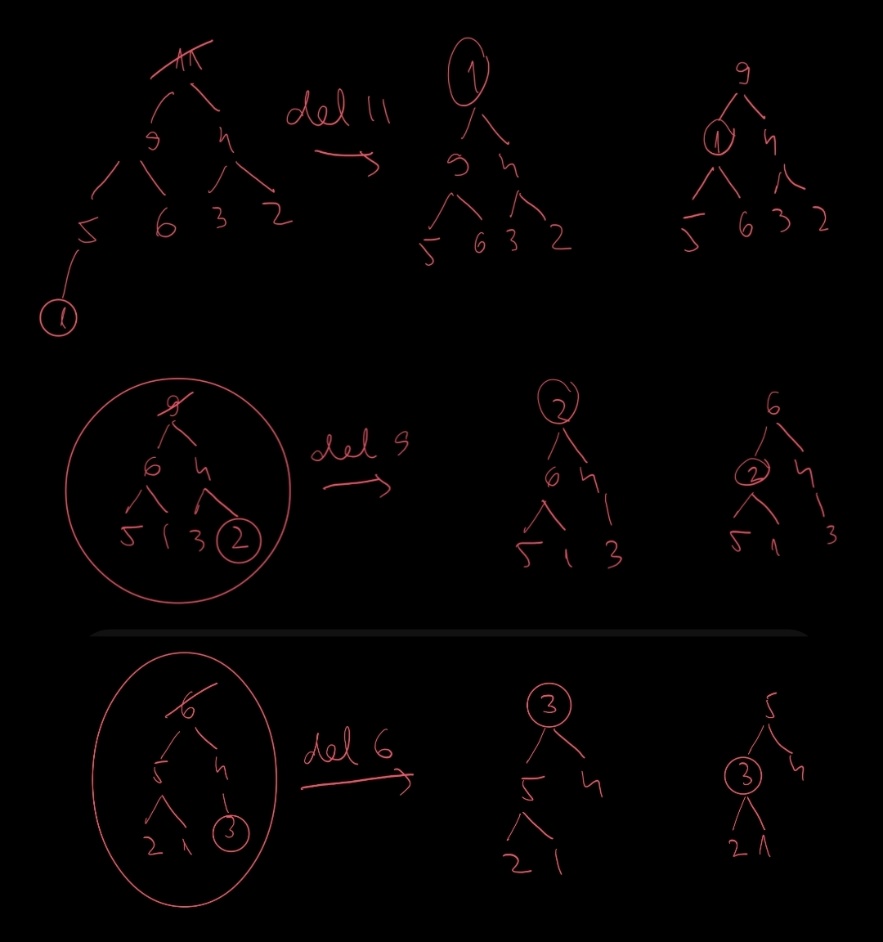
* we take the array and transform it into a MAX-HEAP:
  + the second half of the array will contain leaves (they can be left where they are);
  + Starting from the first non-leaf element (and going towards the beginning of the array), we will just call bubble-down for every element.
* remove elements from the heap (they will be in decreasing order) and place them at the end of the heap and so at the beginning of the new array that is constructed on the space of the initial array (so we can have extra space complexity)

*(example on the next page)*



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 5 | 3 | 4 | 2 | 1 | 6 | 9 | 11 |

The blue part above are the nodes that were removed and added to the end of the heap. After processing all the nodes, we will be left with a sorted array.



*Priority queue – representation on binary heap*

* Push = add to heap + bubble-up
* Pop = remove the root + bubble-down
* Top = return the root

*Binomial heap*

* Binomial tree:

A diagram of a tree

Description automatically generated

* + order 0 = a single node.
  + order k = a tree which has a root and k children, each being the root of a binomial tree of order k − 1, k − 2, ..., 2, 1, 0 (in this order).
* Representation of binomial tree
  + For each node we store
    - Info
    - Address of parent
    - Address of first child
    - Address of next sibling
  + For the tree we keep
    - Address of root
    - Order of tree
* Binomial Heap

A diagram of a tree

Description automatically generated

* + collection of heap trees
  + at most one binomial tree of a given order k
* Representation of a binomial heap
  + sorted linked list
  + each node contains a binomial tree
  + the list is sorted by the order of the trees
* Merging binomial heaps
  + Problem when they have 2 trees of same size
  + 2 trees of size n transform into 1 tree of size n+1
* Other operations:
  + Push =
    - create a new heap with only 1 element (the element to be pushed)
    - merge it with the existing heap
  + Top =
    - one of the roots of the trees in the heap
  + Pop =
    - Delete root of a binomial tree
    - Get a sequence of binomial heaps
    - Transform it into a binomial heap
    - Merge the new heap to the old one