Priority Queues

We are looking for a data structure that can do the following operations very fast:

1 - Insert an element into the set

2 - Find the smallest (largest) element in the set.

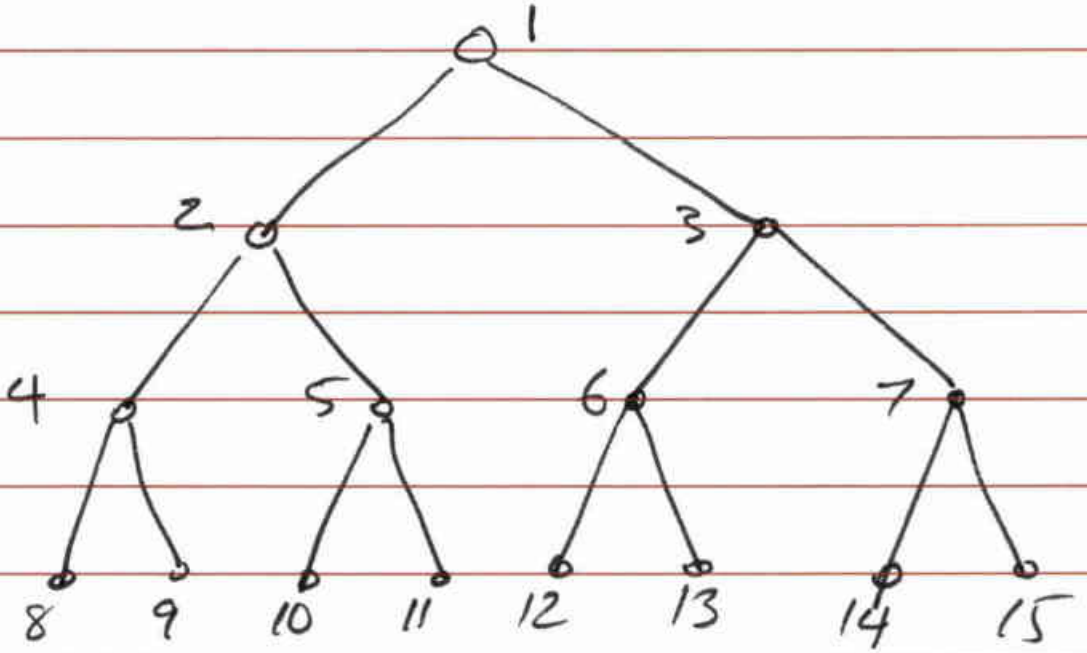
	insert	Find smallest el.
unsorted array	$O(1)$	$O(n)$
sorted array	$O(n)$	$O(1)$
linked list	$O(1)$	$O(n)$
sorted linked list	$O(n)$	$O(1)$

Def. A data structure which provides fast
- insert
- find_min

operations is called a priority queue.

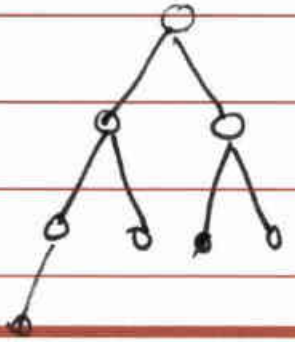
Background

Def. A binary tree of depth k which has exactly $2^k - 1$ nodes is called a *full* binary tree.



Def. A binary tree with n nodes and of depth k is **complete** iff

its nodes correspond to the nodes which are numbered 1 to n in the full binary tree of depth k.



Traversing a complete binary tree stored as an array

Parent(i) is at $\lfloor i/2 \rfloor$ if $i \neq 1$

if $i = 1$, Then i is the root

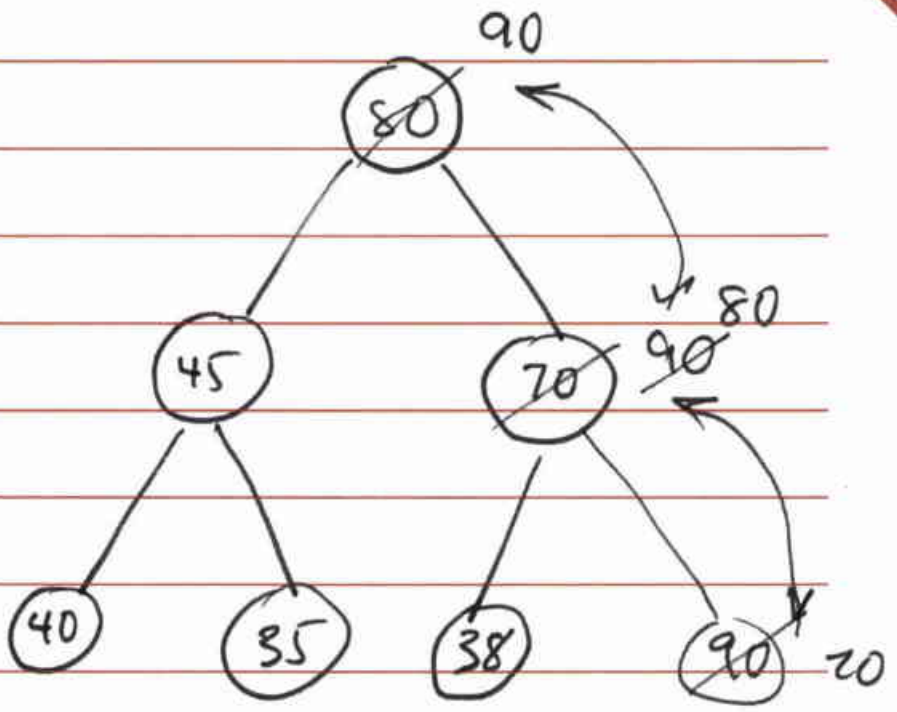
Lchild(i) is at $2i$ if $2i \leq n$
, otherwise it has no left child

Rchild(i) is at $2i+1$ if $2i+1 \leq n$
otherwise it has no right child

Def. A binary heap is a complete binary tree w/ the property that the value (of the key) at each node is at least as large as the values at its children. (Max heap)

Insertion

insert (90)



takes $O(\log n)$