Priority Queues

A priority queue has to perform these two operations fast!

1. Insert an element into the set
2. Find the smallest element in the set

	Insert	Find Min
Array Implementation		

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.

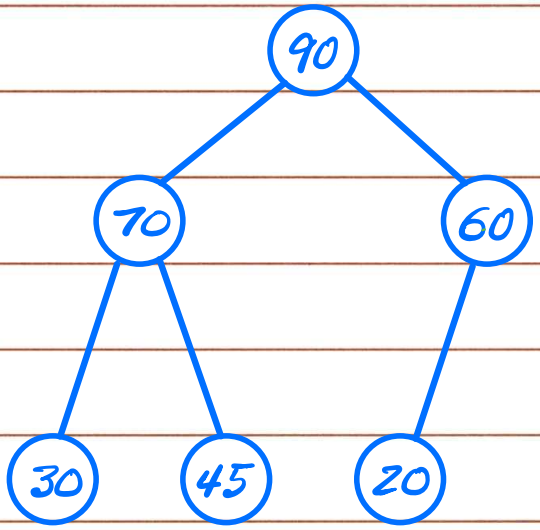
Parent(i)

Lchild(i)

Rchild(i)

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

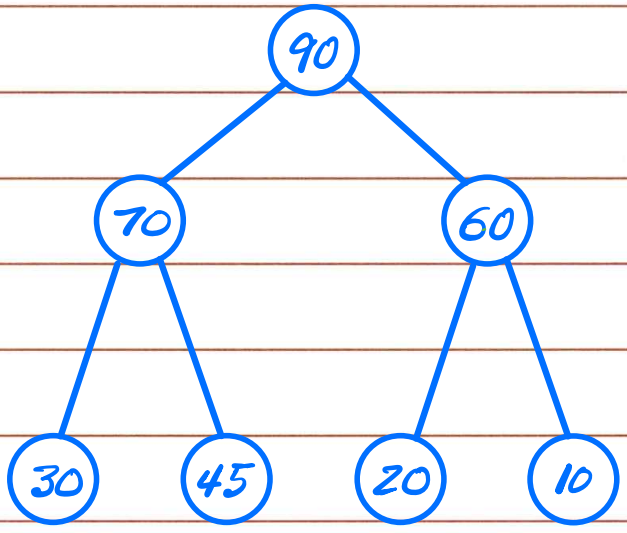
Find_Max



Insert

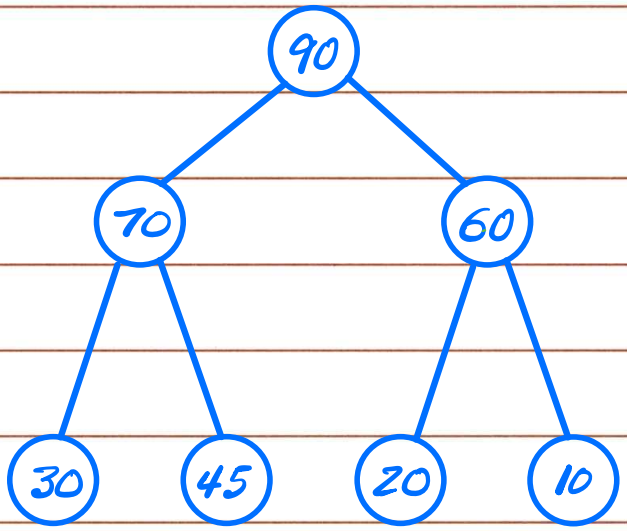
ex. insert

Extract_Max



Delete

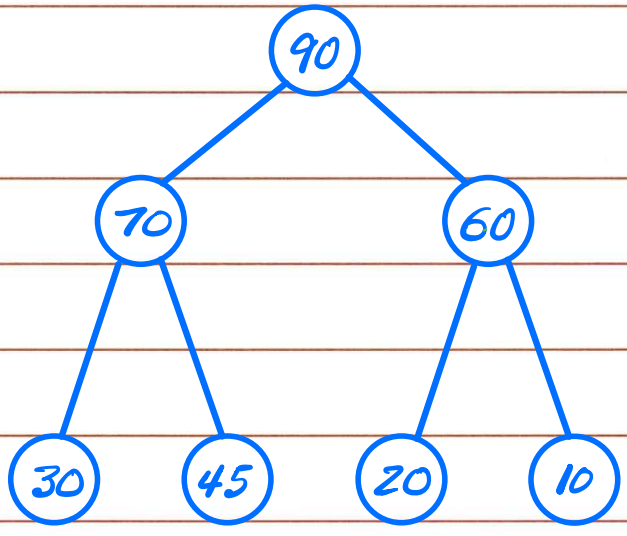
ex. Delete 70



Decrease-key

ex. Decrease-key

70 to 35



Construction

Can be done in $O(n \lg n)$ time using n insert operations.

Q: What is the best run time to merge two binary heaps of size n?

Finding the top k elements in an array.

Input: An unsorted array of length n.

Output: Top k values in the array ($k < n$)

Constraints:

- You cannot use any additional memory
- Your algorithm should run in time $O(n \lg k)$

Def. A binomial tree B_k is an ordered tree defined recursively

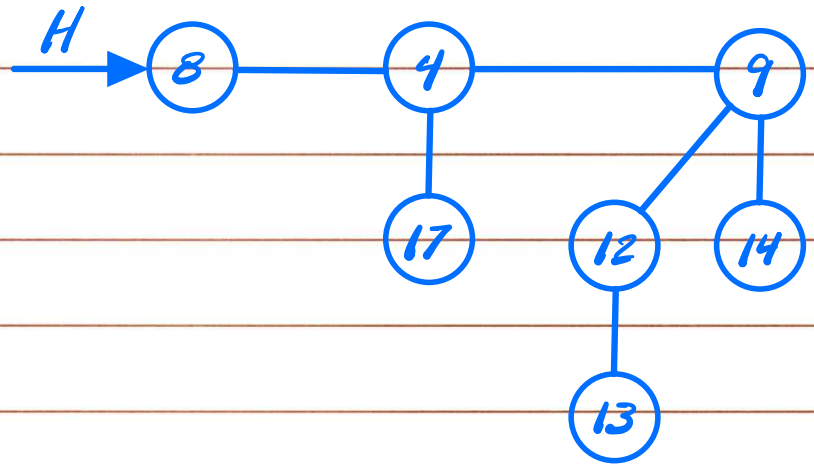
- Binomial tree B_0 consists of one node

- Binomial tree B_k consists of 2 binomial trees B_{k-1} that are linked together such that root of one is the leftmost child of the root of the other.

Def. A binomial heap H is a set of binomial trees that satisfies the following properties:

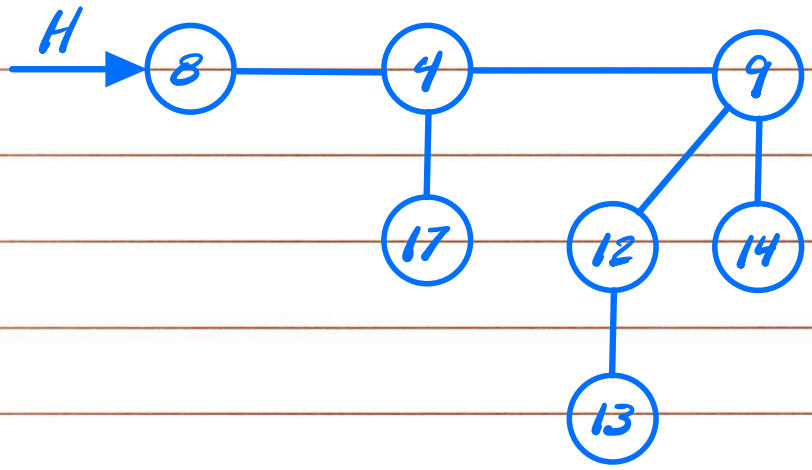
1. Each binomial tree in H obeys the min-heap property.

2. For any non-negative integer k, there is at most one binomial tree in H whose root has degree k.



Insert

ex. insert 5



Amortized Cost Analysis

Ex. 1 Starting from an empty stack
for $i = 1$ to n
    Push or Pop
endfor

Push and Pop take each
Worst case runtime complexity =

Ex. 2 Starting from an empty stack
for $i = 1$ to n
    Push or Pop or Multipop
endfor

Push and Pop take each (worst case)
Multipop takes (worst case)

Worst case runtime complexity =

Aggregate Analysis

- Specify the set of operations involved.

- Show that a sequence of n operations (for all n) takes worst case time $T(n)$ total.

- In the worst case, the amortized cost (average cost) per operation will be

Back to Ex. 2

Observation 1 : Multipop takes $O(n)$ time if there are n elements pushed on the stack.

Observation 2 :

Ex. 2 Starting from an empty stack

for $i = 1$ to n

Push or Pop or Multipop

endfor

Push takes

Pop "

Multipop "

Worst case runtime complexity =

Accounting Method

- Assign different charges to different operations.

- If the charge for an operation exceeds its actual cost, the excess is stored as credit.

- The credit can later help pay for operations whose actual cost is higher than their amortized cost.

- Total credit at any time =
  Total amortized cost − Total actual cost

- Total credit can never be negative.

Ex. 2 Starting from an empty stack

for $i = 1$ to n

Push or Pop or Multipop

endfor

try #1: assign a charge of ___ to each operation

Op.	Charge	Actual Cost	Total Credit

try #1: assign charges as follows:

Push	2
Pop	0
Multipop	0

Op.	Charge	Actual Cost	Total Credit

Ex. 2 Starting from an empty stack

for $i = 1$ to n

Push or Pop or Multipop

endfor

Push takes

Pop "

Multipop "

Worst case runtime complexity =

Fibonacci Heaps

- Fibonacci heaps are loosely based on binomial heaps.

- A Fibonacci heap is a collection of min-heap trees similar to binomial heaps. However, trees in a Fibonacci heap are not constrained to be binomial trees.

- Unlike binomial heaps, trees in Fibonacci heaps are not ordered.

See Fibonacci heap animation at:

www.cs.usfca.edu/~galles/JavascriptVisual/FibonacciHeap.html

	Binary Heap	Binomial Heap	Fibonacci Heap
Find_Min			
Insert			
Extract_Min			
Delete			
Decrease-key			
Merge			
Construct			

3. The values 1, 2, 3, . . . , 63 are all inserted (in any order) into an initially empty min-heap. What is the smallest number that could be a leaf node?

5. Suppose you have two min-heaps, A and B, with a total of *n* elements between them. You want to discover if A and B have a key in common. Give a solution to this problem that takes time O(n log n) and explain why it is correct. Give a brief explanation for why your algorithm has the required running time. For this problem, do not use the fact that heaps are implemented as arrays; treat them as abstract data types.