CSE 101 3-8-22

- SETs : evals

  due: Sun. Mar 13 11:59 PM

  incentive : ?

- Pset : 2 more days, that's it!

- quiz 5 tonight

Continue... heaps...

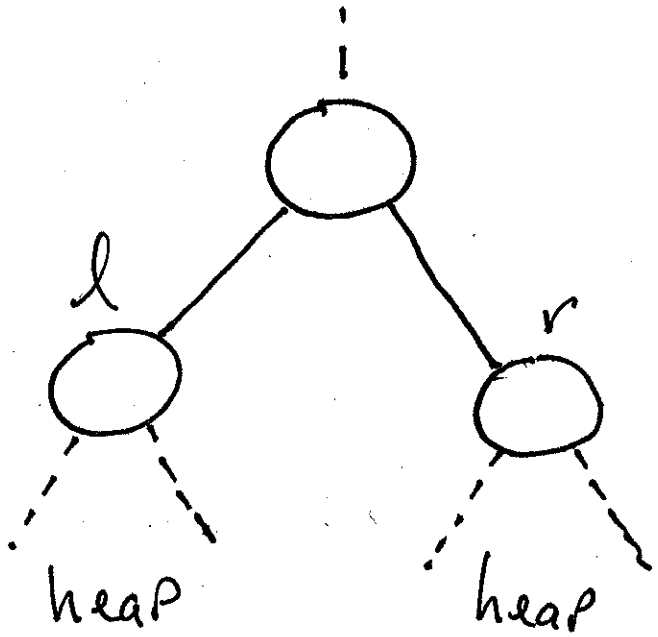
Priority Queue OPS.

- ExtractMax
- IncreaseKey
- Insert
- Maximum

6.2 Heapify

Heapify (A, i) fixes (max) heap property at index i

Precondition: subtrees at left(i) and right(i) are valid heaps



see pseudo-code

runtime:

$T(n) \stackrel{\text{def}}{=}$ [worst case # of recursive invocations of Heapify( ) on a subtree (rooted at i) consisting of n nodes]

Then

$T(n) = (\overbrace{\text{height of subtree at } i}^{h})$

$= \lfloor \lg n \rfloor$

$= \Theta(\log n)$

$= \Theta(h)$

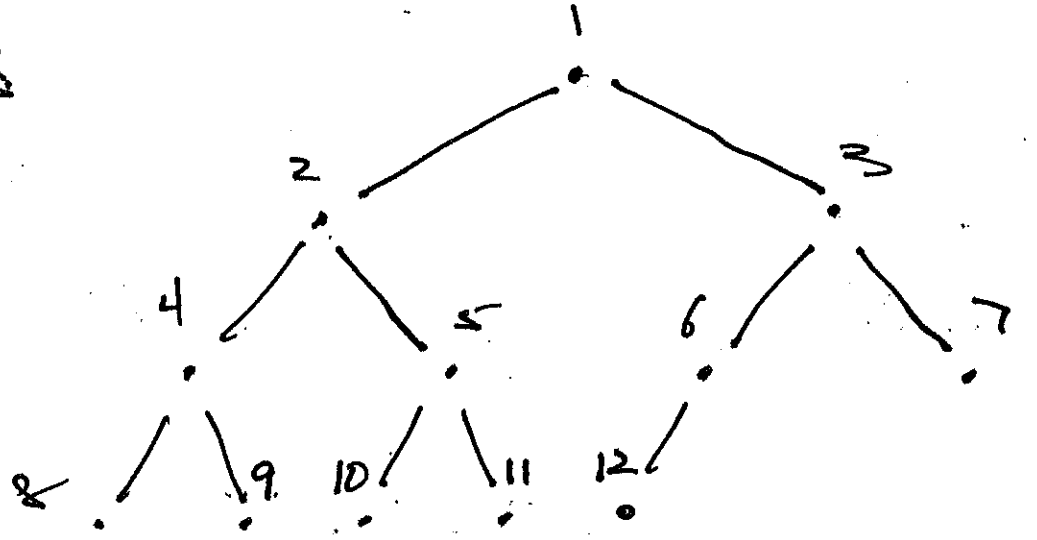
6.3 BuildHeap

Goal: create a max-heap out of an unordered array.

Let:

$$n = heapSize[A] = length[A]$$

Ex



$A_1 \ldots A_6$ $A_7 \ldots A_{12}$

internal nodes — leaves

note:

rightmost internal node is Parent of rightmost leaf

(index of rightmost leaf) $= n$

(index of rightmost internal node) $= \lfloor \frac{n}{2} \rfloor$

see pseudo-code

Runtime:

let $T(n) \stackrel{\text{def}}{=}$ [worst case # of calls to heapify() on an array of length n]

$= \Theta(n)$

↑ see text.

6.4 Heapsort

See pseudo-code

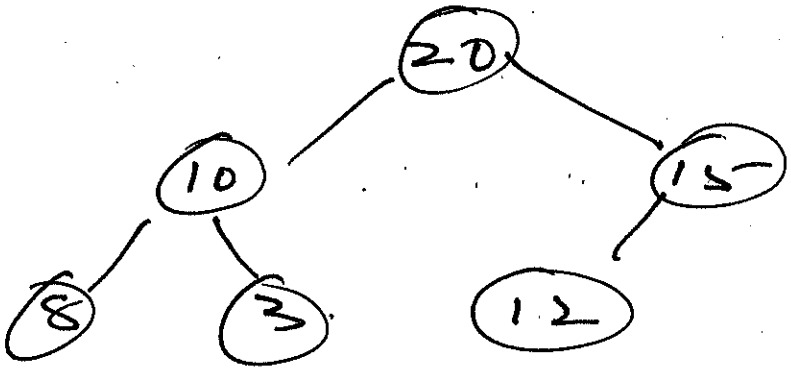
Ex. Perform Heapsort on

	1	2	3	4	5	6
A	10	8	12	20	3	15

Build Heap

heapSize

	1	2	3	4	5	6
A	20	10	15	8	3	12



Runtime:

$$T(n) = \Theta(n) + (n-1) \cdot \Theta(\log n)$$

$$= \Theta(n \log n).$$

Priority Queues

a Priority Queue (PQ) is an ADT that maintains a finite set S of elements (called records) each with an attribute called key

record

$$x = (\underbrace{\cdot, \cdot, \cdot, \cdot}_{\text{satellite data}}, key) \in S$$

$$S = \{\ldots\ldots, x, \ldots\ldots\}$$

key[x] defines Priority of x in S.

states : $\mathcal{S} = \{\ldots, S, \ldots\}$

A (max) P.Q. supports following operations,

- Insert(S, x)
  inserts x into S.
- Max(S):
  returns element with largest key
- ExtractMax(S)
  returns & deletes element with largest key.
- IncreaseKey(S, x, k):
  change key[x] to k if k > key[x], else do nothing.

Heap implemented P.Q.

See Pseudo-code.

Keys only, no satellite data

Runtimes

HeapMaximum(A): $\Theta(1)$

HeapDeleteMax(A): $\Theta(\log n)$

HeapExtractMax(A): $\Theta(\log n)$

HeapIncreaseKey(A, i, k): $\Theta(\log n)$

HeapInsert(A, k): $\Theta(\log n)$

Exercise

Write

MinI()

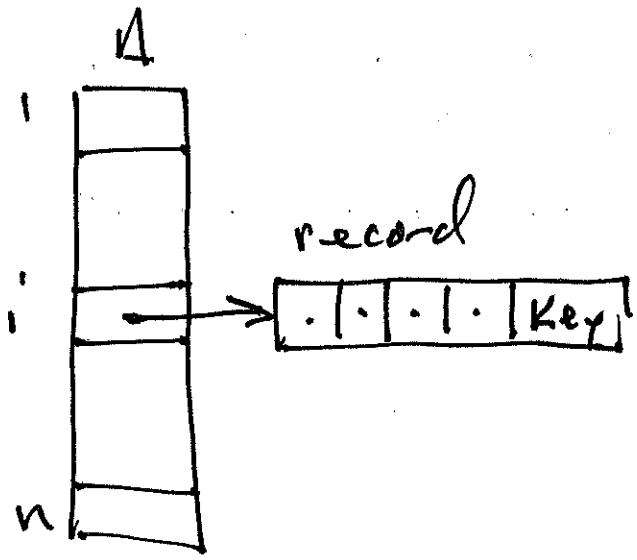
DeleteMinI()

ExtractMinI()

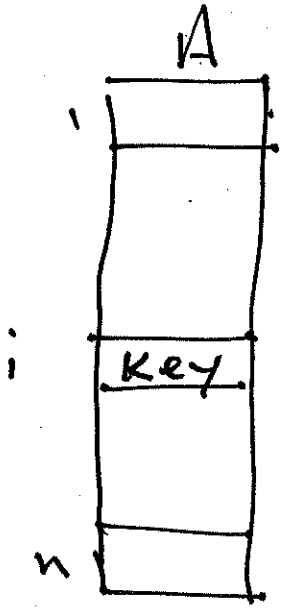
DecreaseKeyI()

for a min P.Q. implemented as a min heap.

General Picture



Our Picture



Exercise

write P.Q. algorithm for general Picture.

Exercise

implement a min P.Q. ADT in C or C++ (or Python).

Read chap 24:

SSSP in a weighted graph.

- BellmanFord
- Dijkstra