CS 314: Pseudocode

Note Title 1/20/2010

Announcements

- HW0 due!
- Look for HW1 tomorrow - will be due next Friday.
  - this will be written HW
  - may work with a partner

Today: Priority Queue (sec. 2.5 in book)
(introducing pseudo code)

What is a priority queue?

A data structure which stores items that have priorities.

Want to support:
- ExtractMin()
- Insert(p)

What is a simple way to try this?

- Store a list
  - insert – stick new item at end : $O(1)$
  - find min : $O(n)$ (if n items in list)

- Sort the list
  - insert : find spot in $\log n$ time $+ O(n)$ to move list down
    - how long to search in sorted list
  - find : $O(1)$

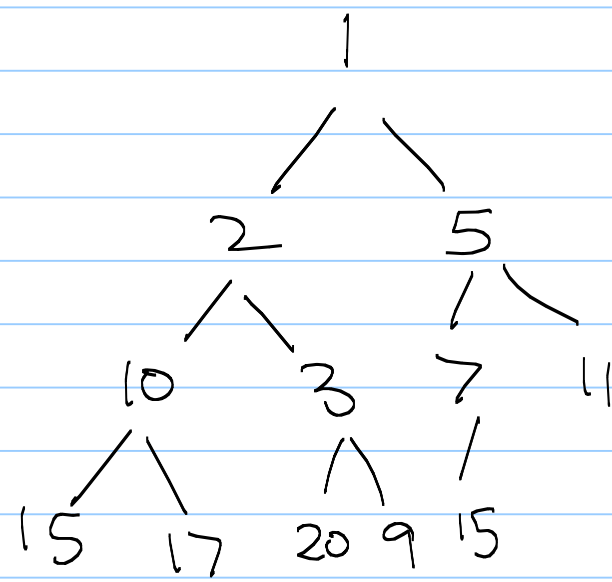
(Min) Heap:

Balanced complete binary tree where:

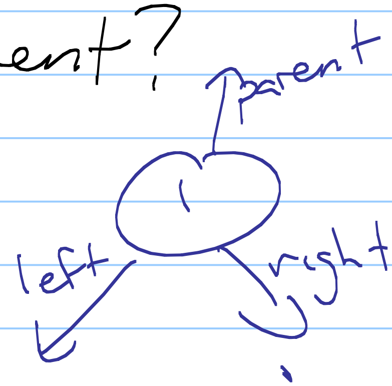
Heap property

for every node v, the ~~key~~ priority stored at v's children is ≤ the key stored at v.

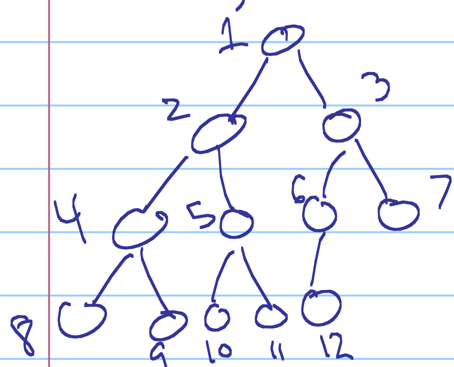
Ex:



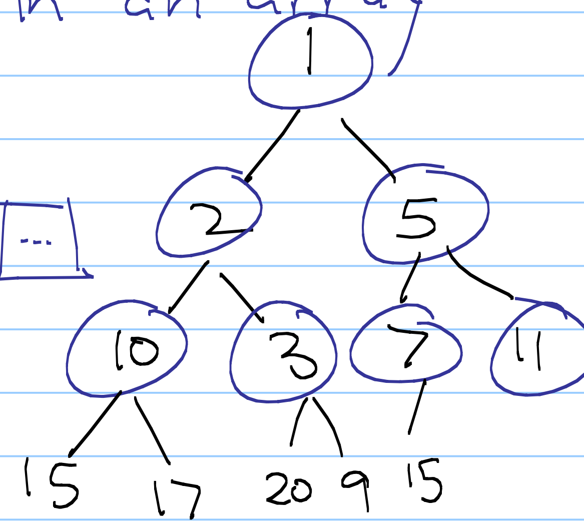
How to implement? Node/pointer structure



Here, we can do better: store in an array



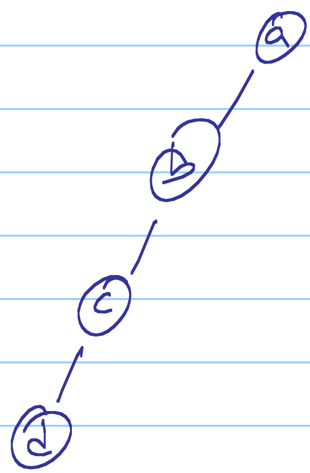
1	2	5	10	3	7	11	...

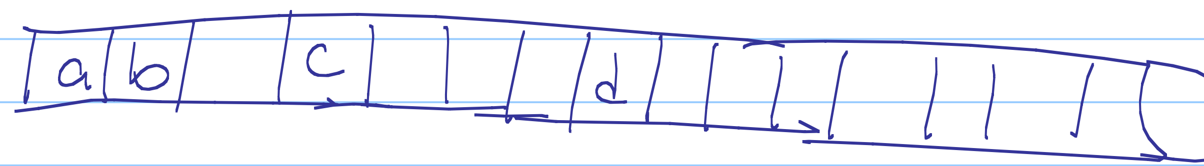


i's parent $= \lfloor i/2 \rfloor$

i's left child $= 2i$

i's right child $= 2i+1$





This method is not space efficient unless the tree is a complete binary tree.

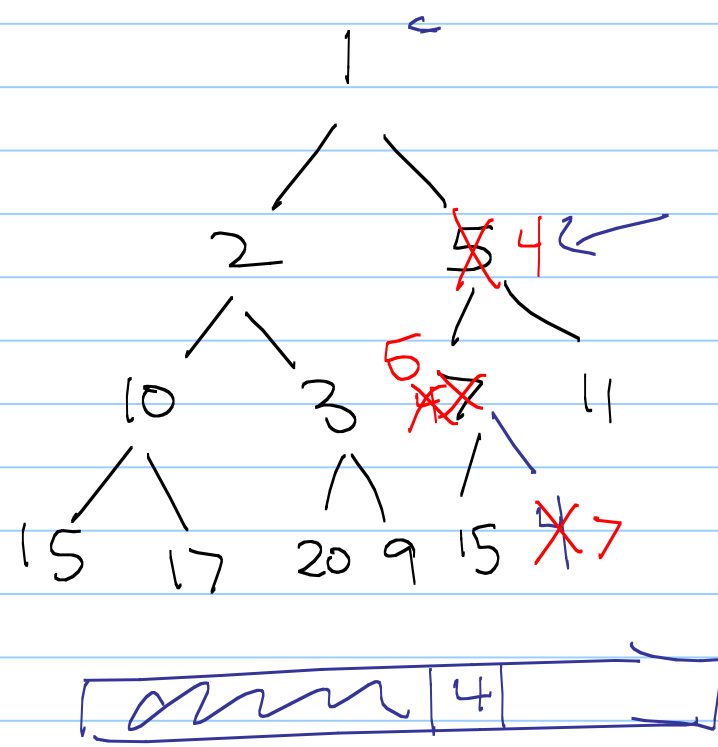
Adding a new element:

- Put new value in $H[n+1]$ insert(4)

Then what?

Want to "heapify":

bubble up the element in tree until heap property is satisfied

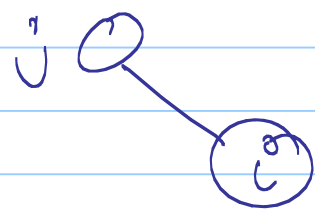


Pseudocode: i is where heap property may be violated.

function name & inputs

```
Heapify-up(H, i):
    If i > 1
        Let j ← ⌊i/2⌋
        If H[j] > H[i]
            Swap array entries at H[j] & H[i]
            Heapify-up(H, j)
        endif
    end if
```

set j equal to $\lfloor i/2 \rfloor$



Prop: Heapify-up(H,i) fixes the heap property is $O(\log i)$ time (assuming H is heap with $H[i]$ too small).

pf: induction on i.

Base case: $i=1$: H is already a heap (since $H[1]$ is the root)

IH: Heapify-up(H,k) works as stated for $k<i$.

IS: Spend $O(1)$ time find & checking the parent & swapping if necessary. Call Heapify$(H, \lfloor i/2 \rfloor)$. H is a heap which satisfy H.P. everywhere except possibly $\lfloor i/2 \rfloor$, so by IH work in $O(\lg \lfloor i/2 \rfloor)$ time.

Total time $= O(1 + \log(i/2)) = O(\log i)$ ☒

What about deleting?

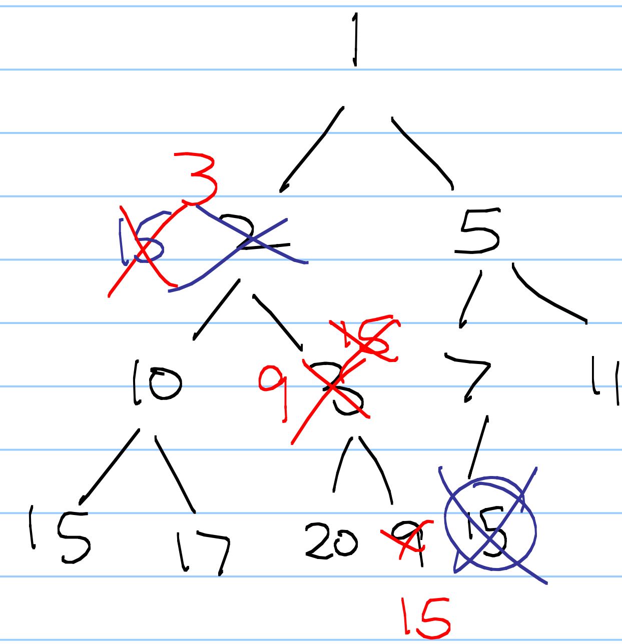
Delete(H, i)

position in array

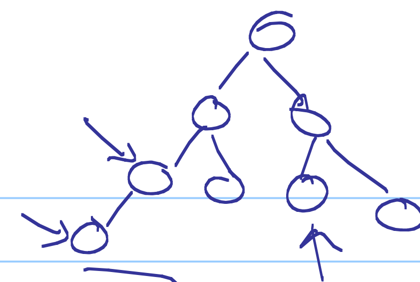
Delete(H, 2)

Heapify!

bubble down in tree



```
Heapify-down(H, i):
  n ← length(H)
  if 2i > n              ← at a leaf
      terminate with H unchanged
  else if 2i < n                      have 2
      left ← 2i                         children
      right ← 2i+1
      j ← min(H[left], H[right])
  else if 2i = n
      j ← 2i
  end if
  if H[j] < H[i]
      swap H[i] and H[j]
      Heapify-down(H, j)
  endif
```



Prop: The procedure Heapify-down(H, i) fixes the heap property in $O(\log n)$ time (assuming H is a heap with $H[i]$ too big).

Proof

(similar - see p. 64 in text)

__Thm:__ Priority queues can be implemented using heaps, where all operations will take $O(\log n)$ time.

(use heapify-up & heapify-down)