Binary Heaps

Tuesday, November 3, 2020 5:00 PM

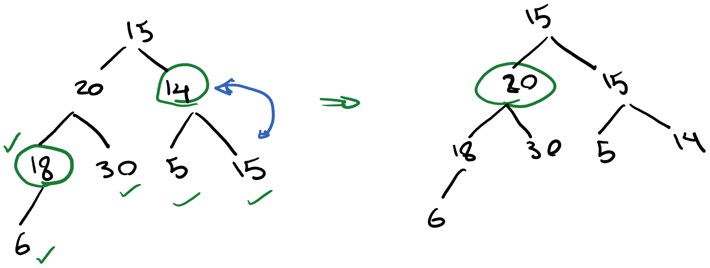
Reminder: HW 7 & Lab5 are due this Sunday.

Example: Sort this array using heapsort.

a = [15, 20, 14, 18, 30, 5, 15, 6]

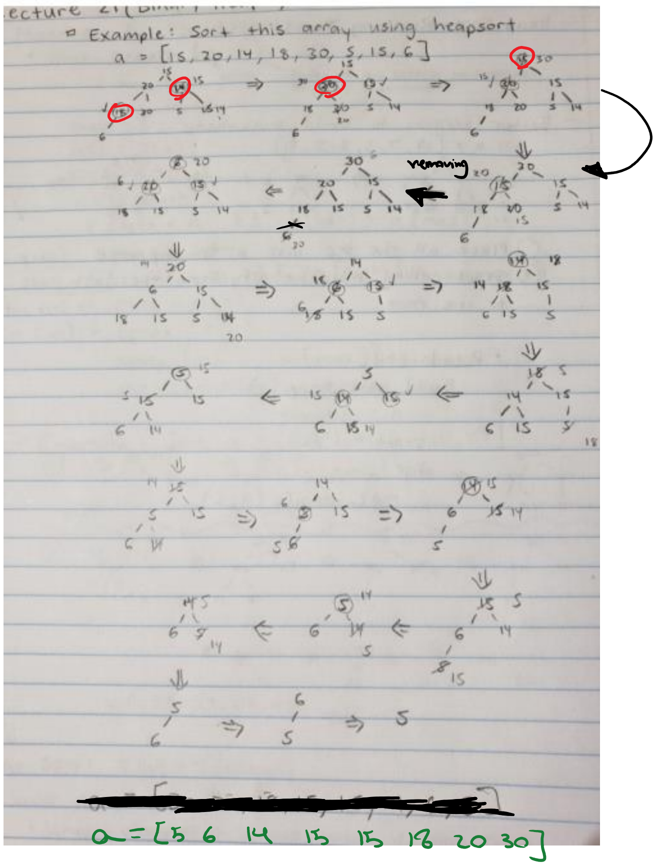
→ Build max-heap

→ Sort: removing root one by one!



```
heap Sort (a)
Build_max-heap (a)
for ??
    |
    max-heapify(?)
end
```

Selina



Lecture 21 (Binary Heaps)

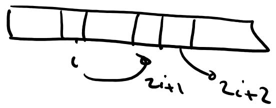
Example: Sort this array using heapsort

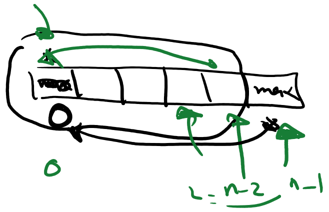
a = [15, 20, 14, 18, 30, 5, 15, 6]

removing

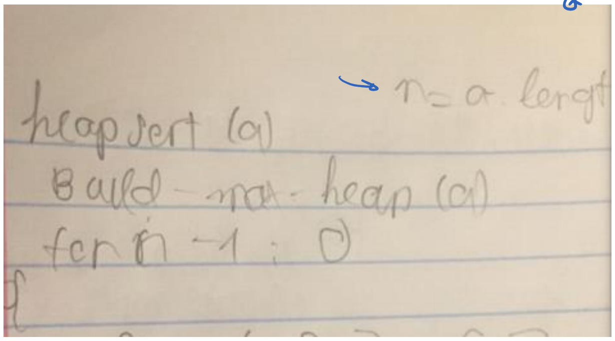
[redacted]

a = [5 6 14 15 15 18 20 30]



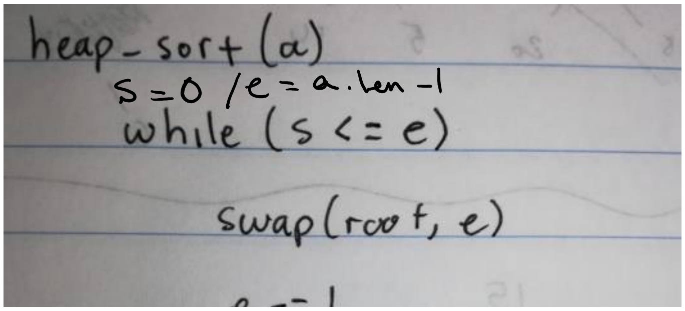


Khang

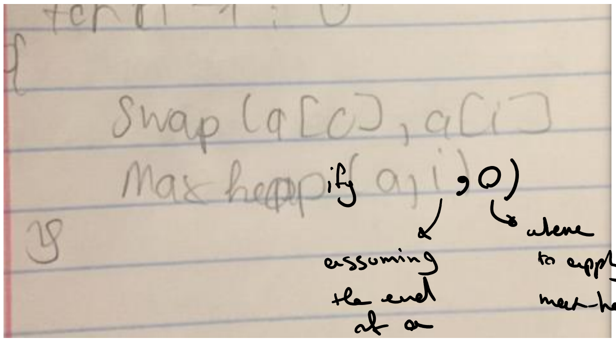


```
heapSort (a)          → n = a. length
Build-max-heap (a)
for n-1 : 0
{
```

Khai



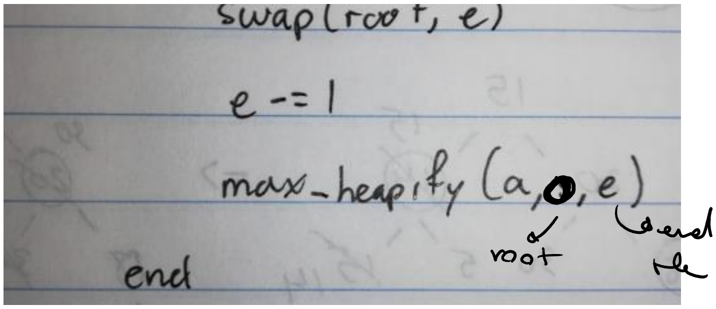
```
heap_sort (a)
    s = 0 / e = a.len - 1
    while (s <= e)

        swap(root, e)
```



```
for(i-1 ; 0
{
    Swap (a[0], a[i])
    Max heapify(a, i, 0)
}
```

assuming the end of a

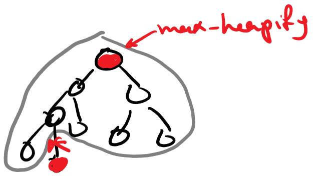
where to apply max-heapify



```
swap(root, e)
e -= 1
max_heapify (a, 0, e)
end
```

root

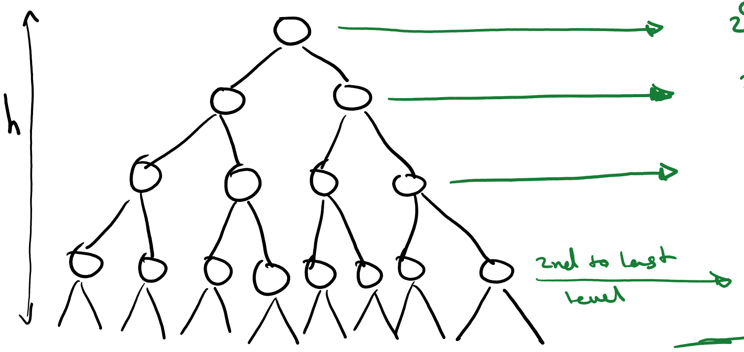
end of the a



Time complexity to "build" max-heap with the better version:

① Place all the elements in a binary tree (You skip this when coding)

② Start calling max-heapify from the last internal node to the root.



nr of nodes	# work for each node
2^0	$h-0$
2^1	$h-1$
2^2	$h-2$
$\vdots$	
2^{h-1}	$h-(h-1)=1$

$$T(n) = h + 2(h-1) + 2^2(h-2) + \sim + 2^{h-1}(1)$$

total amount of work

$$= \sum_{i=0}^{h-1} 2^i(h-i) = O\left(\int_0^{h-1} 2^x (h-x)\, dx\right)$$

2

another sol. $\Longrightarrow$ $S = h + 2(h-1) + 2^2(h-2) + \sim + 2^{h-1}$

* :) $\boxed{2S - S = S}$

$$2S = 2h + 2^2(h-1) + 2^3(h-2) + 2^4(h-3) + \sim + 2^h$$

$$-S = -h - 2(h-1) - 2^2(h-2) - 2^3(h-3) - - 2^{h-1}$$

$$S = -h + 2 + 2^2 + 2^3 + \sim + 2^h$$

$$= -h + \sum^{h} 2^i + 1 - 1$$

Calculations :)

* $2h - 2(h-1) = 2(\cancel{h} - \cancel{h} + 1) = 2$

□ $2^2(h-1) - 2^2(h-2) = 2^2(\cancel{h} - 1 - \cancel{h} + 2) = 2^2$

$\vdots$

$$= -h + \sum_{i=1} 2 \quad +1 \; -1$$

$$= -h + \sum_{i=0}^{h} 2^i \quad -1$$

$$= -h + \frac{2^{h+1} - 1}{2-1} - 1$$

$$= -h + 2^{h+1} - 2$$

$$= -h + 2(2^h) - 2$$

h=logn

$$= -\log n + 2(2^{\log n}) - 2$$

$$= -\log n + 2(n) - 2$$

$$\boxed{= O(n)}$$

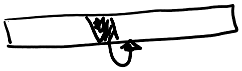
To build max-heap

Binary Search Tree (BST)

h(BST) = 6

* Siblings = share the same parents

* Successor = smallest greater element



node.p

node.left

node.right

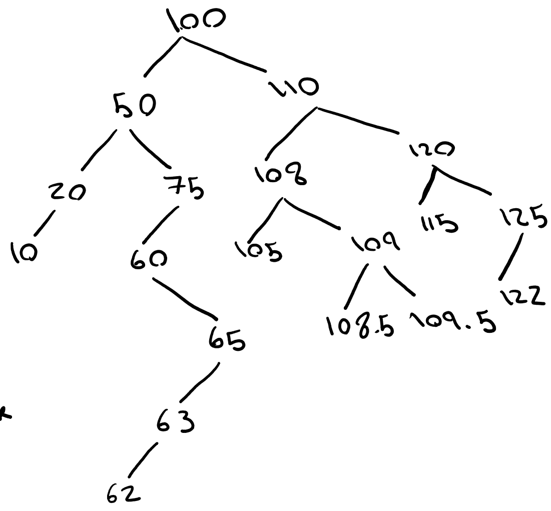
node.val

recursive

_find_min

_find_max

itr


100
50
110
20
75
108
120
10
60
105
109
115
125
65
108.5
109.5
122
63
62