CS3000

5/30 - Tue.

Admin

- exams back today!
- HW3 - Long due tom. 9pm
- HW4 - Long goes out thurs
- Rec 3 today (graded)
- Fun optional recitation thurs.

Agenda

1. Heaps
2. Heapsort overview
3. Heapsort Implementation

1. Heaps

Heapsort ~ sort of like selection sort

Find smallest unsorted element
$\Theta(n^2)$

Can we do better? } Apply a strategy

Quicksort
mergesort } D+C

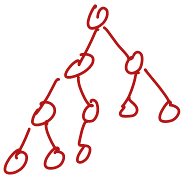
Another way to do better: Change the data structure

- makes algo faster ←
- or, need/want to do the algorithm

Heap

A heap is a complete binary tree

- Binary tree: every node has ≤ 2 children
- Complete: levels get filled in lef to right
- Height: $\Theta(\lg n)$



A min heap:

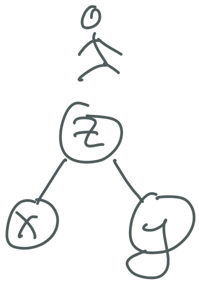
- every node is smaller than all its descendents

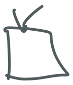
Heap Properties:

- min, completeness

Comparison

BT



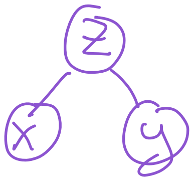


z.left = x

z.right = y

(tree imp.)

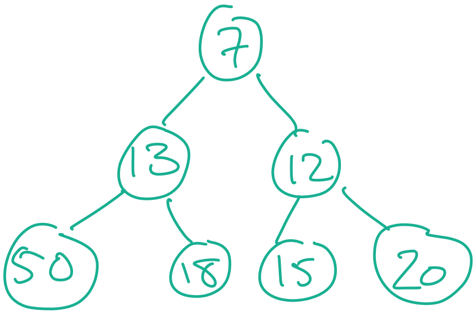
Heap



<z, x, y>

(array imp.)

ex



min heap

Array:

< 7,	13,	12,	50,	18,	15,	20 >
1	2	3	4	5	6	7

Q: For a node at i, where are its children?

left $2i$

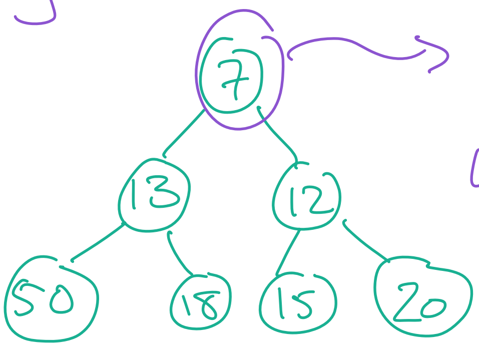
right $2i+1$

Array A to represent a Heap:

attribute A.heapsize (pos in A where heap ends)

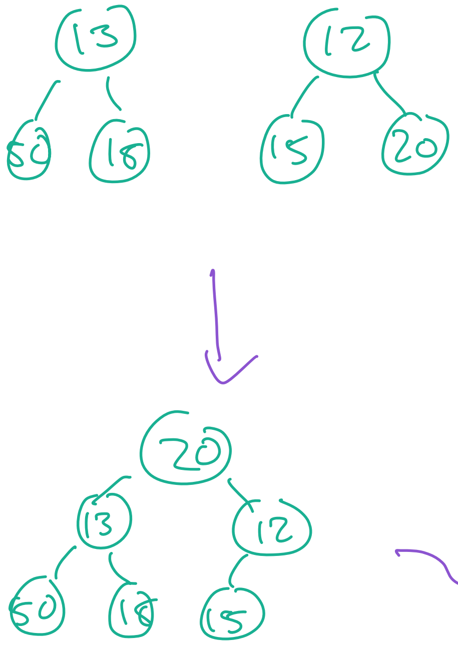
2. Heapsort Overview

- Given a min-heap, stored in array
- Take the min element (position 1), put in next space in output
- Re-heapify!
- Sort in descending order



remove root (smallest)

Sorted
[7]



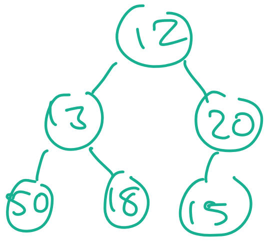
Heapify

- complete BT
- min-ness
- last element is new root

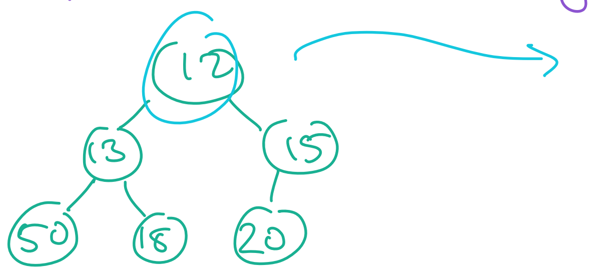
(maintains structure, need to fix min-ness)

mostly perfect min-heap (root might be messed up)

- swap 20 with 12 (swap root with smaller of 2 children)



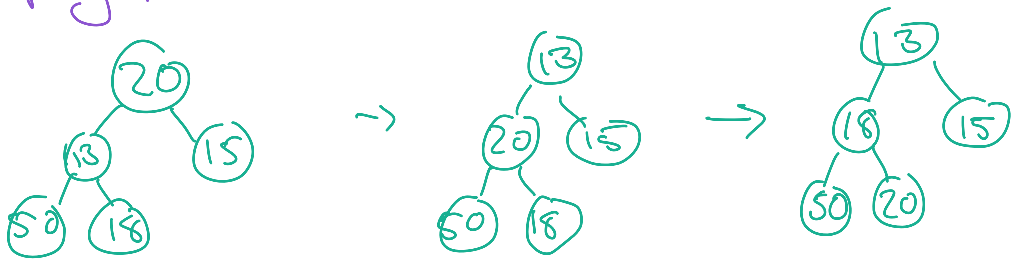
- swap 20 with 15 (only child)



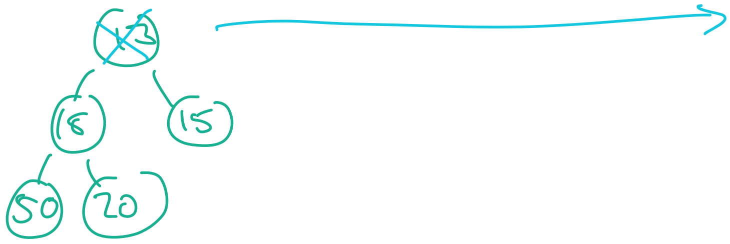
sorted output

[12, 7]

- Heapify!



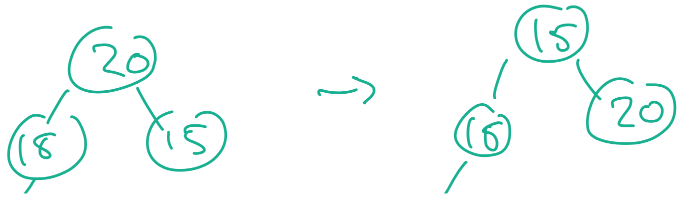
- remove smallest element



sorted output

[13, 12, 7]

- Heapify

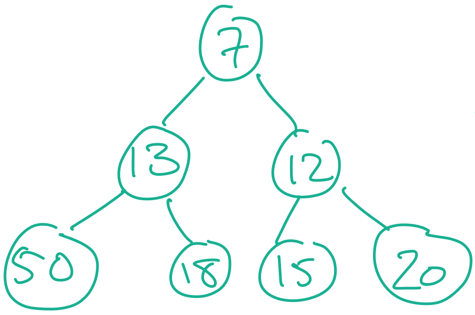


Run-time of ~~one~~ Heapify (worst case)

$\Theta(\lg n)$ → Finding the min element!

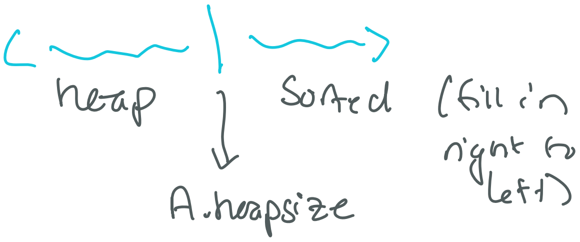
10:38

3. Heapsort Implementation



→ <7, 13, 12, 50, 18, 15, 20>

- In-place algorithm
  ↳ at any point, some subarray is sorted, and some is still a heap



- Call to Heapify function

↳ today: black box
takes in A, i
↳ current root

- assume the root is messed up, but the rest is ok
- bubbles down the root

```
Heapsort (A)                          A → starts as min heap
A.heapsize = A.length                 → starts with whole thing
                                        unsorted
for i = A.length downto 2
        swap A[1], A[i]               → swap root with last ele
        A.heapsize = A.heapsize - 1
        Heapify (A, 1)                → reheapify
```

> ⟨7, 13, 12, 50, 18, 15, 20⟩

7	13	12	50	18	15	20
1	2	3	4	5	6	7

A.heapsize = 7
i = 7
    swap A[1], A[7]
    A.heapsize = 6
    Heapify (A, 1)

↓
swap
↓

⟨20, 13, 12, 50, 18, 15, ¦ 7⟩

20	13	12	50	18	15	7
1	2	3	4	5	6	

(heap) (sorted)

↓
Heapify

<12, 13, 15, 50, 18, 20 ¦ 7>

↓
Swap

<20, 13, 15, 50, 18, ¦ 12, 7>

(heap) ¦ (sorted)

↓
Heapify

<13, 18, 15, 50, 20 ¦ 12, 7>

(Swap)

<20, 18, 15, 50, ¦ 13, 12, 7>

(heap) ¦ (sorted

i = 6
Swap [1], [6]
A.heapsize = 5
Heapify (A, 1)

i = 5
Swap [1], [5]
A.heapsize = 4
Heapify (A, 1)

Runtime of Heapsort:

- loop: n times
- Heapify: lgn each call

↳ total: $\Theta(n \lg n)$

Did we beat selection sort? $\boxed{\text{Yes!}}$

Did we take up extra space? $\boxed{\text{No!}}$