CS3000

5/15 - thurs :)

Admin

- HW1 due 9pm
- HW2 due 5/22 9pm
- Fun-algo today! 1:30
- Exam #1 5/22 during class!

exam 1

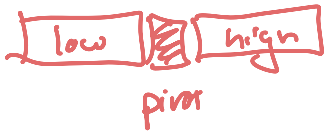
- in class, on paper
- 8.5x11 inch cheat sheet (one side)
- no other notes/devices
- DAS acc, reach out to them
- practice problems in rec on tues.

Agenda

1. Heaps
2. Heapsort
3. Example/run-time

0. Quicksort Reminder

- partition: pivot in place
  left - small
  right - big



1. Heaps!

→ can we do better?

strategies

1. D+C
2. Change the data structure ←
3. ...

Data structure

□ A <~, ~, ~, ..., ~>

🧍 different!

□ array

🧍 heap

Algorithmic concept (doesn't change)

- input: array A, length n
- split array into sorted, unsorted
- repeatedly find max in unsorted subarray
- put into its final, sorted position

ex 1, $\underline{10}$, 2, 9, 8

1, 8, 2, 9, $\underline{10}$
sorted

1, 8, 2, $\underline{9, 10}$
sorted

Concept, with traditional array

1, 2, 8 7, 10
sorted

- outer loop - n times
- inner loop - find max in unsorted array

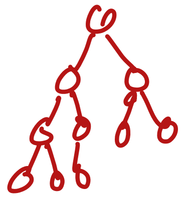
average run time: $\Theta(n^2)$ n

→ can we do better? ←

(ideally w/no extra space)

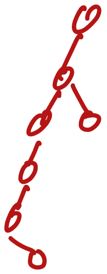
array & heap

Strategy: data structure changes from array to heap!

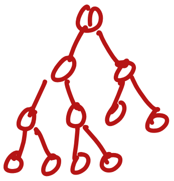


root

A heap is a complete binary tree
- each node has ≤ 2 children
- node w/o children is a leaf
- top node is root
- levels are filled in top to bottom left to right
- every node ≥ all descendents (max heap)

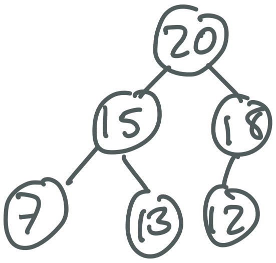


not a heap :(



heap :)

1 2 3 4 5 6
A: ⟨20, 15, 18, 7, 13, 12⟩



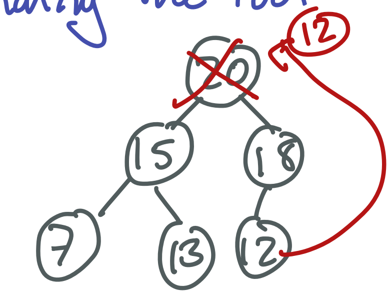
position of root: 1

position of max value: 1

assumption: array A starts as max-heap

for a node at position i:
- what pos is left child? $2i$
- what pos is right child? $2i+1$
- what pos is parent? $\lfloor i/2 \rfloor$

removing the root



- last node becomes the root
- then, fix so it stays a heap

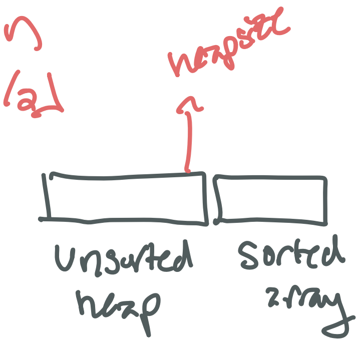
2. Heapsort

↳ same algo approach: repeatedly find max, put where it belongs

Can we beat $\Theta(n^2)$?

Assumptions:

- input array A repping a heap, length n
- left: 2i right: 2i+1 parent: $\lfloor i/2 \rfloor$
- attribute A.heapsize
- output: A in sorted order



Heapify(A, i)

- root is possibly incorrect, but the rest of the heap is a valid max heap
- i is position of root

Heapsort(A, n)

- repeatedly find max
- put into correct sorted order
- (can call heapify)

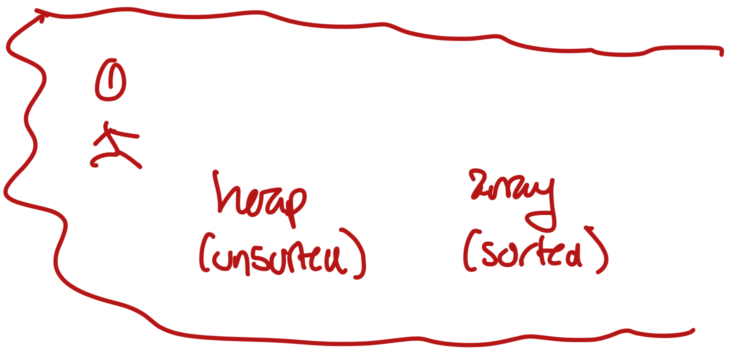
LEFT(i) returns 2i

RIGHT(i) returns 2i+1

3. Heapsort Example + Run-Time

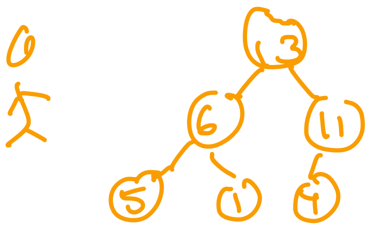
📺 A = ⟨–, –, –, –, ..., –⟩

time to beat: $\Theta(n^2)$



ex 📺: A = 13, 6, 11, 5, 1, 4

n=6



tracking:

n   overall array size

A.heapsize   end of unsorted heap   (↓)

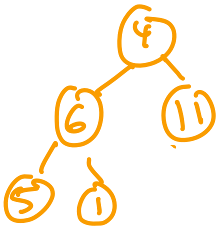
next sorted position   (↓)

(heap, unsorted)   array

A.heapsize = n

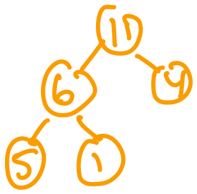
① root (13) goes to sorted array

A.heapsize = n-1



13

② Heapify

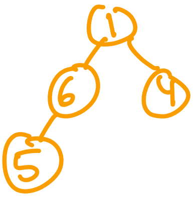


11, 6, 4, 5, 1, 13

13

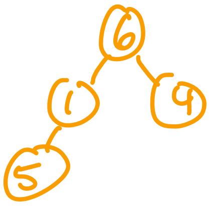
③

n-2

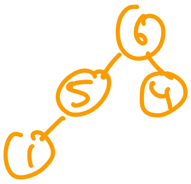


11, 13

④ Heapify

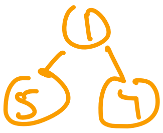


6, 5, 4, 1, 11, 13



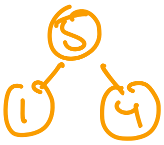
11, 13

⑤



6, 11, 13

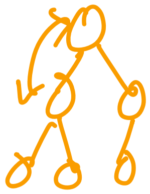
⑥ Heapify



5, 1, 4, 6, 11, 13
heap   sorted

Heapify:

- swap root with greater of left, right child
- bubble down until we reach the bottom or both children are smaller
- original root becomes root of a smaller tree when it swaps

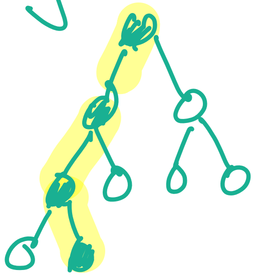


Algorithm:

- loop from n downto 2 (n steps)
- at each step, we call heapify (?)

Run-time: n · (heapify)

Heapify worst case:



(root is minimum)

↳ heapify run-time is linear in height of the tree

$\Theta(h)$ h=height

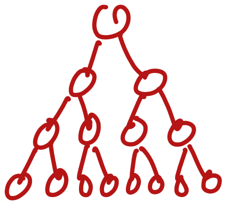
what is h?

- height = # edges from root to deepest leaf
- what is h in terms of n?

↳ size of A

# nodes

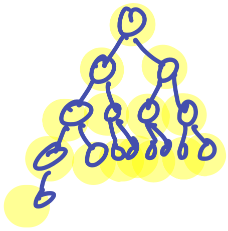
ex tree levels are all full



h	n
0	1
1	3
2	7
3	15

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

ex lowest level has only one node



h	n
0	1
1	2
2	4
3	8
4	16

$n=2^h$

$\lg n = h+1$

$\lg n = h$

For a complete binary tree, height = $\Theta(\lg n)$

Heapsort: $n *$ heapify

$= \Theta(n \lg n)$ ☺

(original selection sort was n^2)