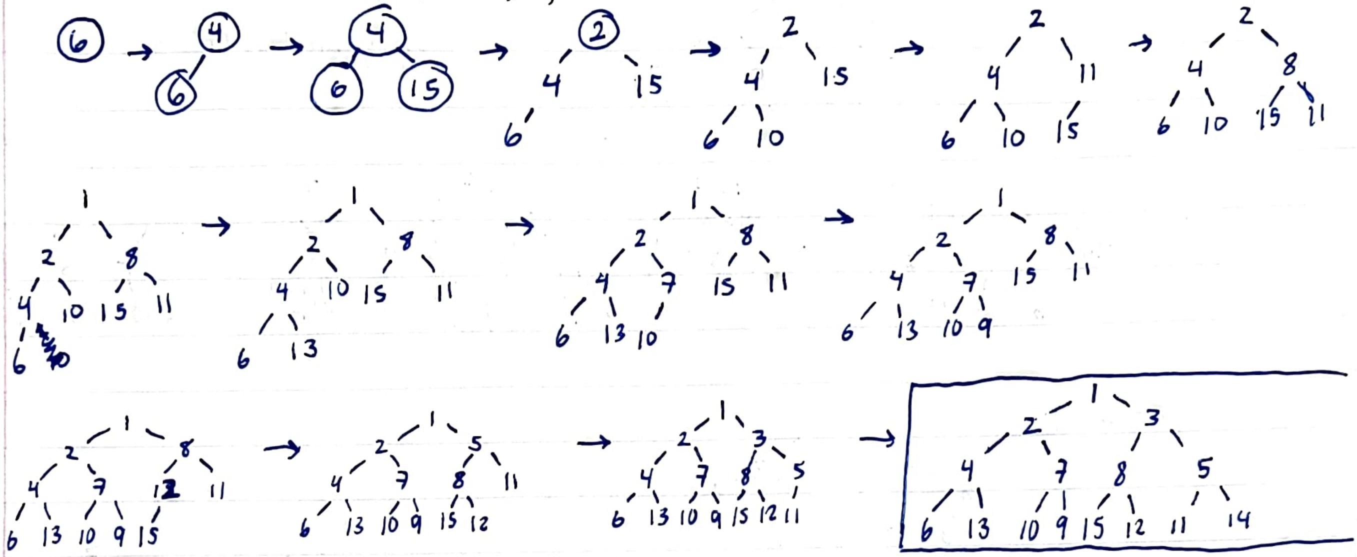
Sophia Evanisko
631007256

PROBLEM SET 6

1

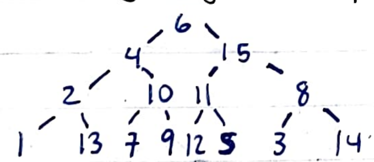
A

6, 4, 15, 2, 10, 11, 8, 1, 13, 7, 9, 12, 5, 3, 14



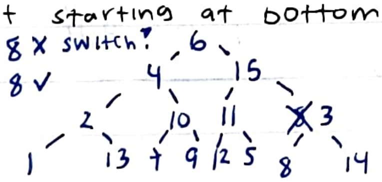
B

throw everything into heap first, sort later

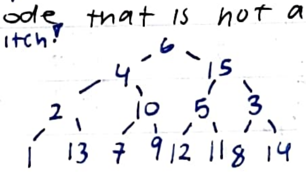


sort starting at bottom-right most node that is not a leaf (N/2 node)

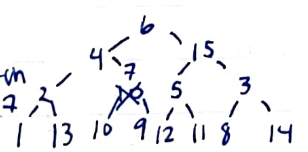
8: 3<8 ✗ switch?
14>8 ✓



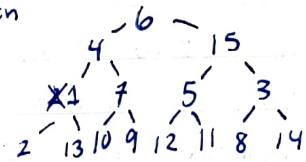
11: 5<11 ✗ switch?
12>11 ✓



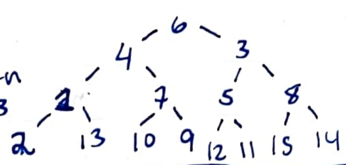
10: 7<10 ✗
9<10 ✗
7<9 ← switch w/ 7



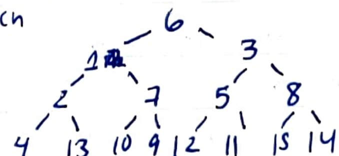
2: 1<2 ✗ switch
13>2 ✓



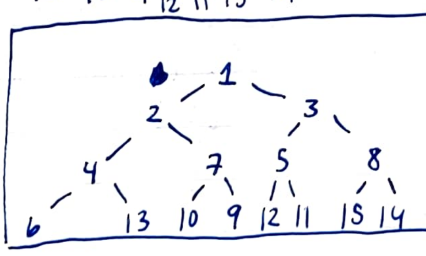
15: 5<15 ✗
3<15 ✗
3<5 ← switch w/ 3
8<15
14<15
8<14 ← switch w/ 8



4: 1<4 ✗ switch
7>4 ✓
2<4 ✗ switch
13>4 ✓

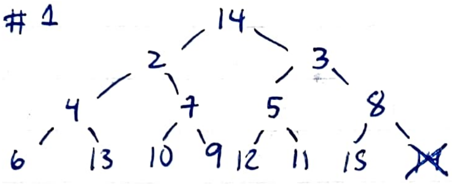


6: 1<6 ✗
3<6 ✗
1<3 ← switch w/ 1



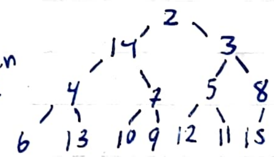
NOT THE SAME AS 1A MIN HEAP

1C DELETE MIN # 1

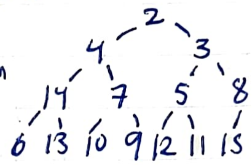


then percolate downward until everything sorted?

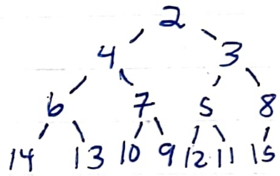
14 > 2
14 > 3
2 < 3 → switch w/2



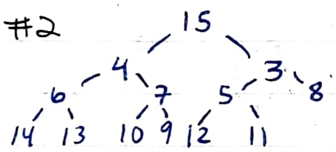
14 > 4
14 > 7
4 < 7 → switch w/4



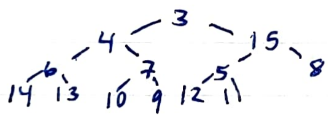
14 > 6
14 > 13
6 < 13
switch w/6



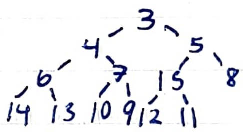
DELETE MIN #2



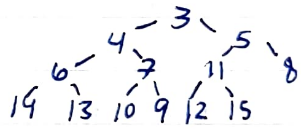
15 > 3
15 > 4
3 < 4
switch w/3



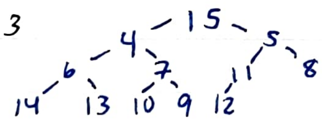
15 > 5
15 > 8
5 < 8



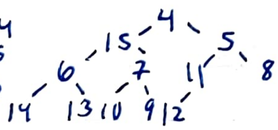
15 > 12
15 > 11
11 < 12



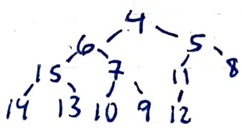
DELETE MIN #3



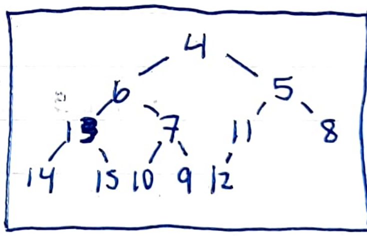
15 > 4
15 > 5
4 < 5



15 > 6
15 > 7
6 < 7



15 > 14
15 > 13
13 < 14



2 PROVE HEAPIFY DOES A MAX OF 2N−2 COMPARISONS

since we only do comparisons on n/2 of the tree

we know on the first run:

h =

$\frac{n+1}{2}$ nodes → don't move

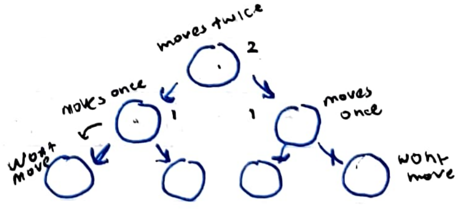
$\frac{n+1}{4}$ nodes → move 1 level

$\frac{n+1}{8}$ nodes → move 2 levels

etc.

total levels and moves is $\frac{n+1}{2^{h+1}} \cdot l$

~~total~~ moves



4 moves × 2 comparisons ← first run

+ 2 moves × 2 comparisons ← second run

$h=2$

so for example on the first run we have:

n	h	~~max~~ moves
1	1	0
3	2	1
7	3	1×2 + 2×1 = 4
15	4	1×3 + 2×2 + 4×1 = 11
31	5	1×4 + 2×3 + 4×2 + 8×1 = 26
63	6	n−h = 63−6 = 57
127	7	n−h = 127−7 = 120

etc.

however, there is also a second round of movements equal to h where you must go through the heap and do final switches

second round = h moves * 2 comparisons per move

$\underbrace{2[2^h]-2-2h}_{\text{first round of comparisons}} + \underbrace{[h*2]}_{\text{second round of comparisons}} = 2[2^h]-2+0$

simplifies to $\boxed{2n-2 \text{ comparison}}$

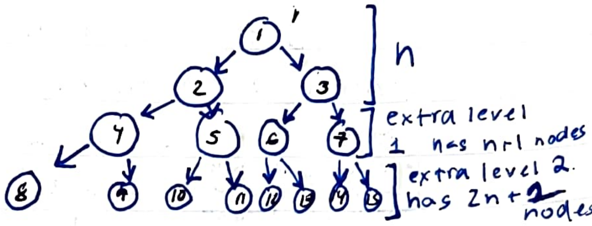
so # of moves on first run is $n-h$ ~~where~~ or 2^h-1-h

so # of comparisons on first run is $2(2^h-1-h)$ because you have 2 comps per move

(one comparison to see which of two children is larger, another to see if child + parent switch)

SOPHIA EVANISKO
631007256

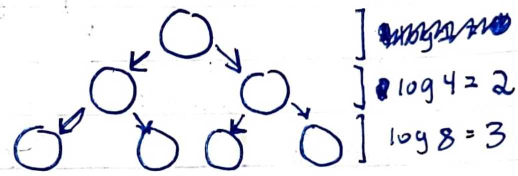
3 A two extra levels



every time you add an extra level to a binary search you are adding the # of nodes on the tree + 1 ← because of the root

- so, extra level one should add $n+1$ extra nodes, making the tree a size of $n + (n+1) = 2n+1$
- thus the second extra level will add $(2n+1)+1$ nodes, making the tree a size of $n + (n+1) + (2n+2) = \boxed{4n+3}$

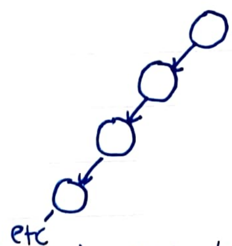
B deepest node is at depth $2 \log n$



by logarithm rules $2\log n = \log n^2$
height of bst is ~ $\log n$
if the height of this tree is n^2 there must be n^2 nodes in the tree
note: height is depth but opposite, in complete bst height of tree = depth of root
thus, the array must be able to hold at least $\boxed{n^2}$ elements

C deepest node is at depth $4.1 \log n$
we can use the same logic as in part B
log rules show $4.1 \log n = \log n^{4.1}$
height of bst is ~$\log n$ so if height is $\log n^{4.1}$ there must be $n^{4.1}$ nodes in the tree
array must have size of $\boxed{n^{4.1}}$

D worst-case binary tree



this is a binary tree thats basically just a straight line
in this case height $= n-1$
the max number of nodes in a bst of height h is $2^{h+1}-1$
so if we plug our height of $n-1$ we get an

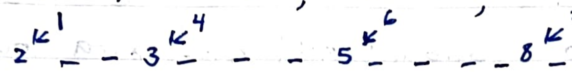
array size of $2^{(n-1)+1} - 1$
$= 2^n - 1$

4 Show 8 element heap only needs 8 comparisons

elements: 1 through 8 store unknown values + must be heapified

compare: 1 with 2, 3 with 4, 5 with 6, 7 with 8 = 4 comparisons
4 total

lets assume we get this result (though it doesn't matter + will be 8 comparisons either way): 1 > 2, 4 > 3, 6 > 5, 7 > 8

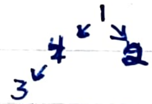


next compare: 1 with 4, 6 with 7 = 2 comparisons
6 total

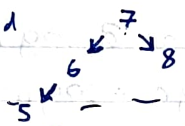
lets assume the result is: 1 > 4, 7 > 6

by transitive we know: 1 > 4 > 3 and 7 > 6 > 5

heap tree so far



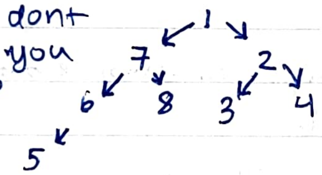
and



next compare: 1 with 7, 2 with 4 = 2 comparisons
8 total

assume 1 > 7, 2 > 4

again, the values dont actually matter, you can build this heap regardless of your starting inputs, this was just an example of how



in 8 total comparisons