CS2700: PROGRAMMING AND DATA STRUCTURES

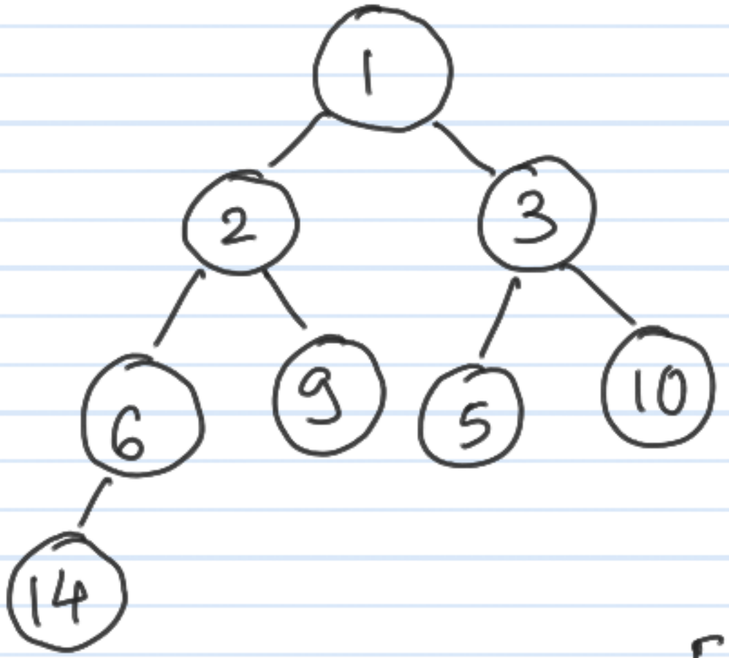
PRIORITY QUEUE

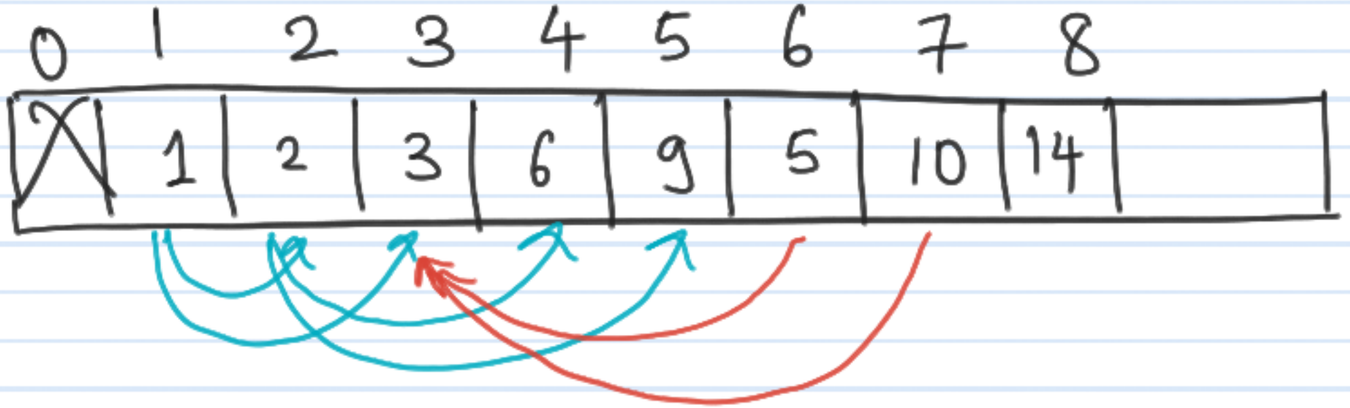
(Binary heap)

Goals:

Implementing a heap using arrays.

REPRESENTING A HEAP USING an ARRAY



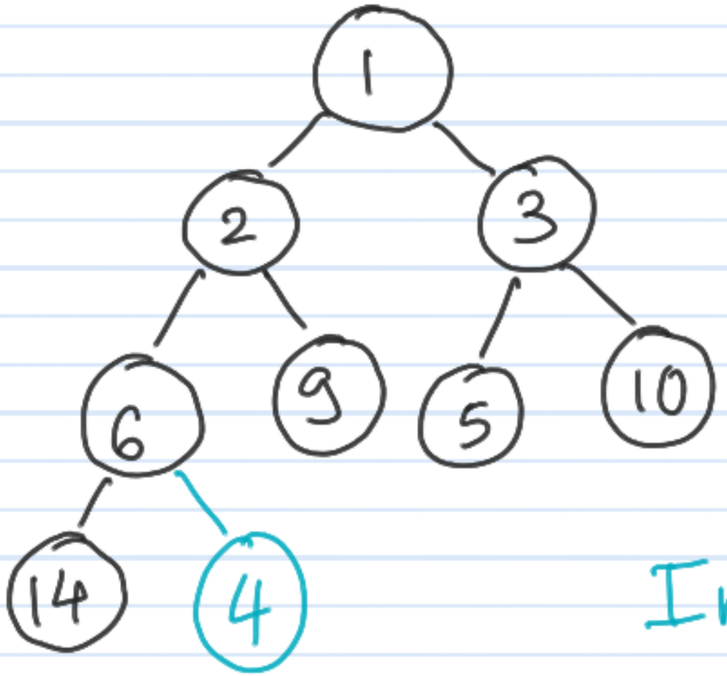


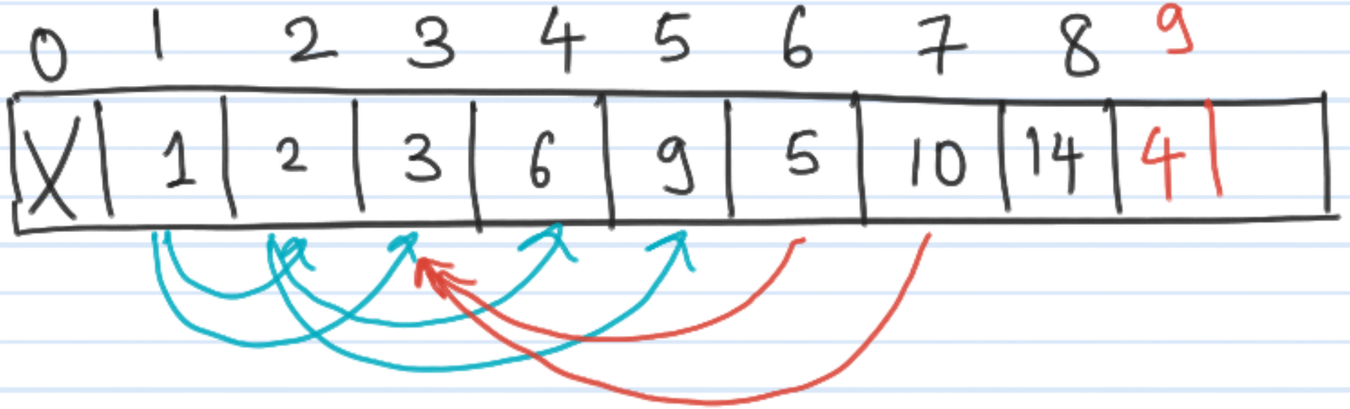
For an element at index i

2 children (if present) : $2i$ $2i+1$

parent (if non root) : $\lfloor i/2 \rfloor$

REPRESENTING A HEAP USING an ARRAY

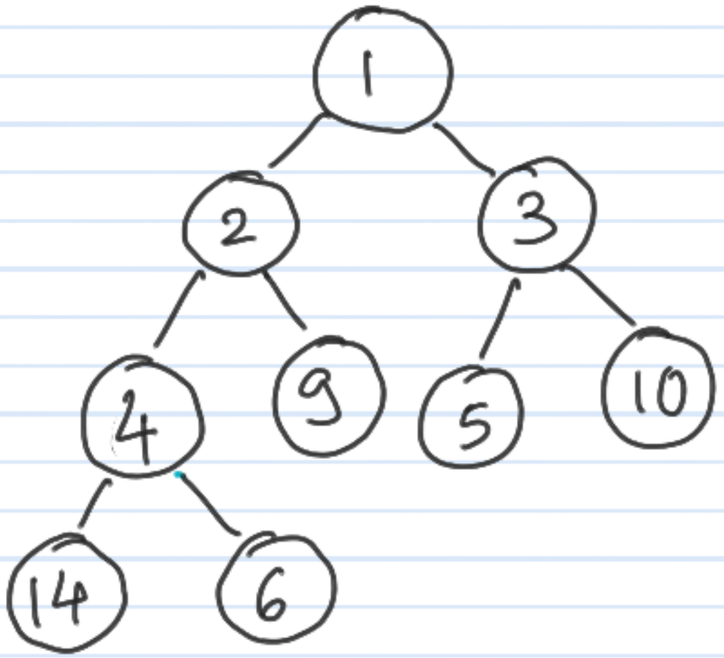




Insert (4)

- Where does this insertion happen in array?
- Percolate up

REPRESENTING A HEAP USING an ARRAY

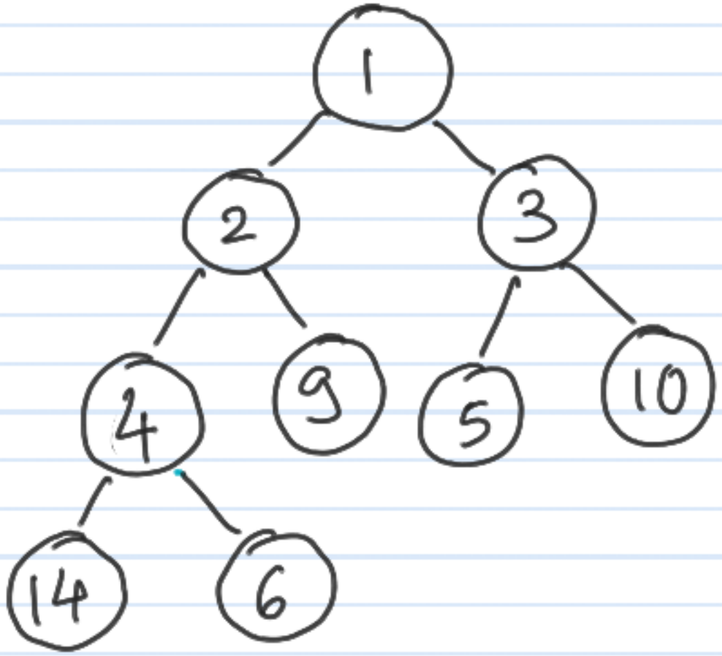


0	1	2	3	4	5	6	7	8	9
X	1	2	3	4	9	5	10	14	6

Insert (4)

- adds one more element to the array
- percolate up is a simple set of swaps.
  - how many ??

REPRESENTING A HEAP USING an ARRAY

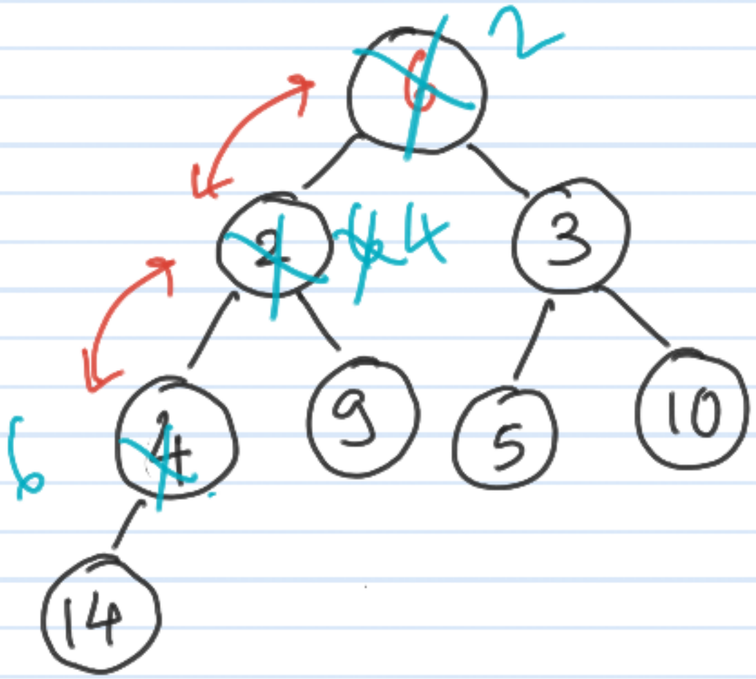


0	1	2	3	4	5	6	7	8	9
X	1	2	3	4	9	5	10	14	6

DeleteMin ()

- Return value at root
- Move last element to root
- Percolate down

REPRESENTING A HEAP USING an ARRAY



0	1	2	3	4	5	6	7	8	9
X	1	2	3	4	9	5	10	14	6

DeleteMin()

- Return value at root
  - element at index 1.
- Move last element to root
  - Swap last elem with index 1
- Percolate down.
  - set of swaps. - how many ??

INITIALIZING A HEAP with n ELEMENTS.

- inserting one element $O(\log n)$ time
- n elements : $O(n \log n)$

Can we do better ?

↳ the best we can hope for is $O(n)$ time.

↳ why?

INITIALIZING A HEAP with n ELEMENTS.

n elements are given for inserts

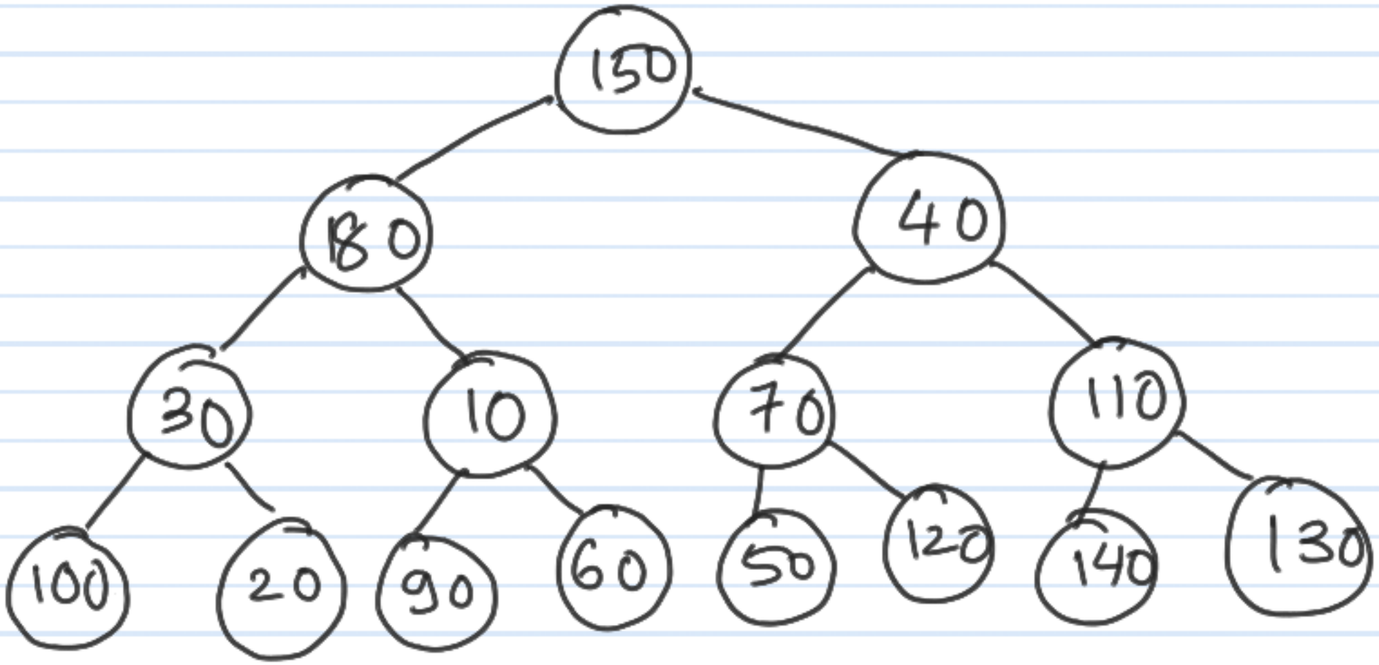
↳ no delete min queries interleaved.

A different algorithm

- place n keys in "any order" in heap.
- correct from bottom all the way upto top.

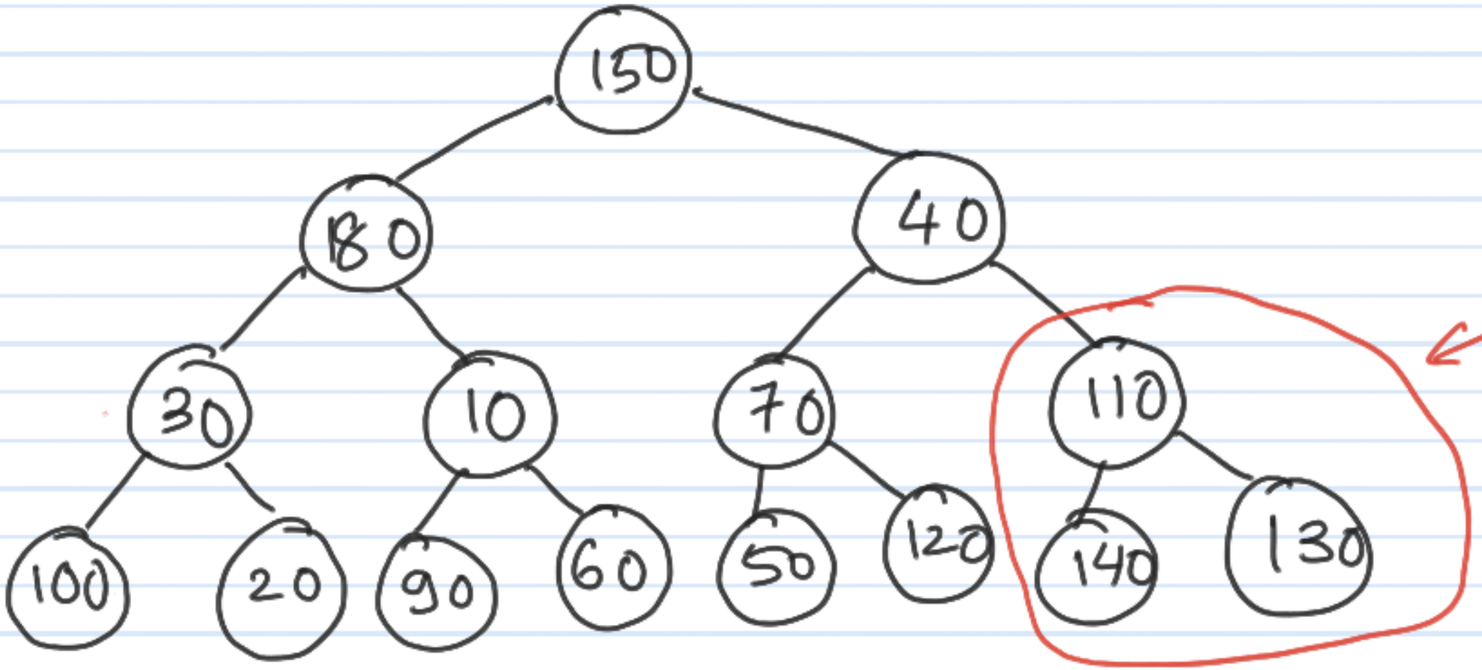
150 , 80 , 40, 30, 10, 70, 110 , 100 20, 90 , 60, 50 , 120,

140, 130



150 , 80 , 40, 30, 10, 70, 110 , 100 20, 90 , 60, 50 , 120,

140, 130



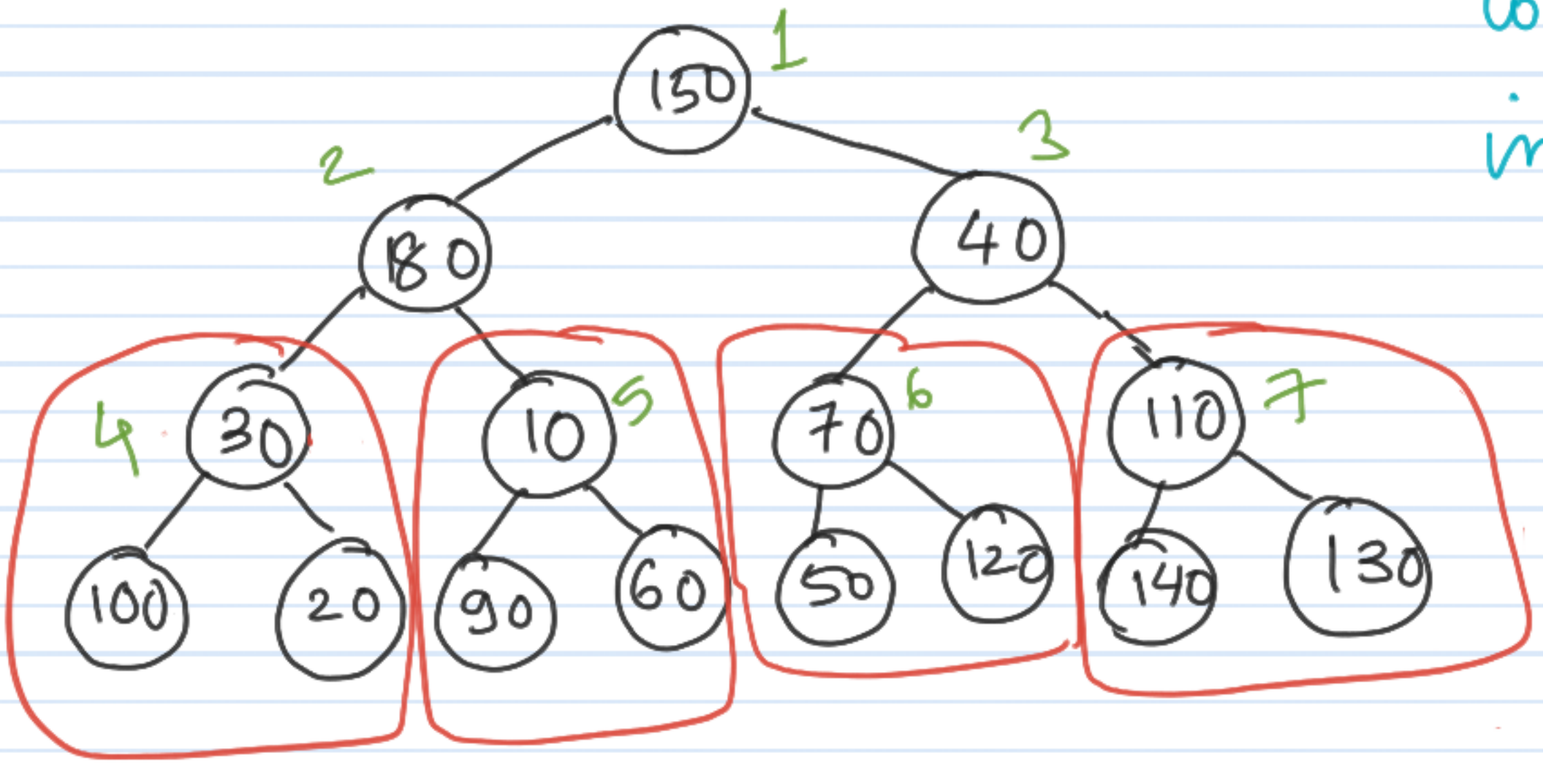
check this subtree.

where do you find it

$n = 15$

150 , 80 , 40, 30, 10, 70, 110 , 100 20, 90 , 60, 50 , 120,

140, 130

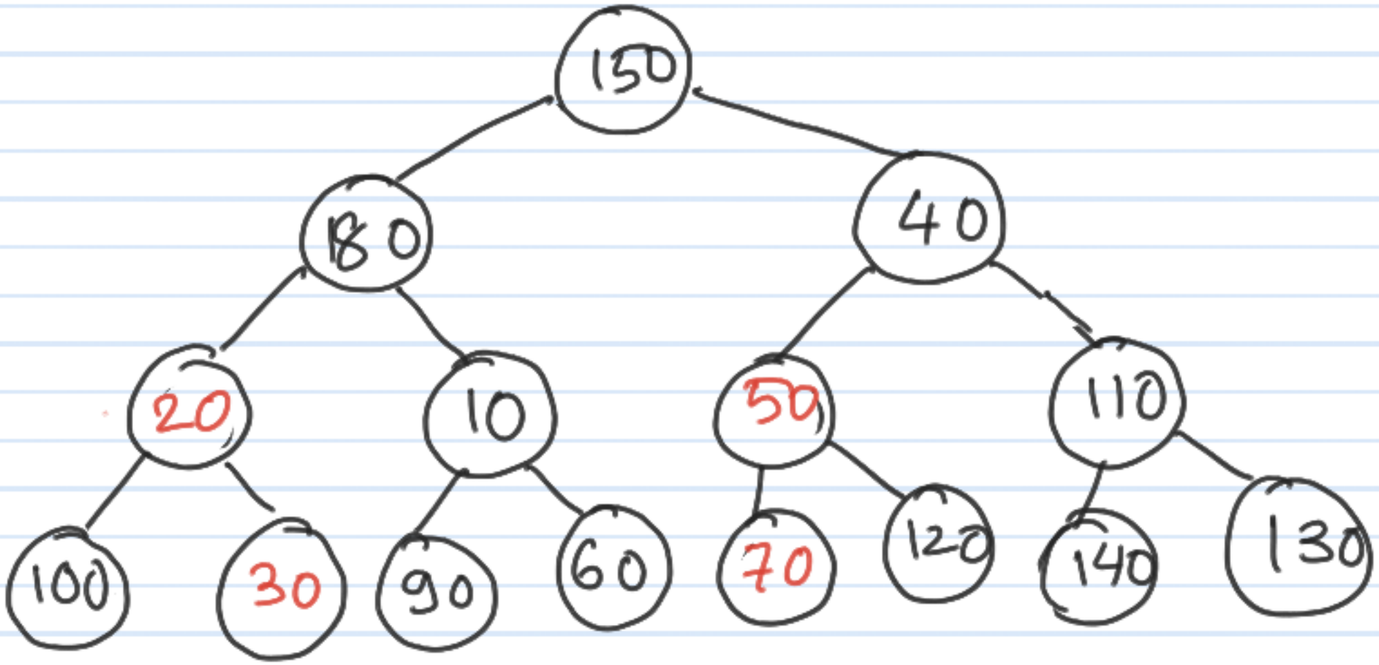


Consider array indices

7, 6, 5 . . . 1

150 , 80 , 40, 30, 10, 70, 110 , 100 20, 90 , 60, 50 , 120,

140, 130

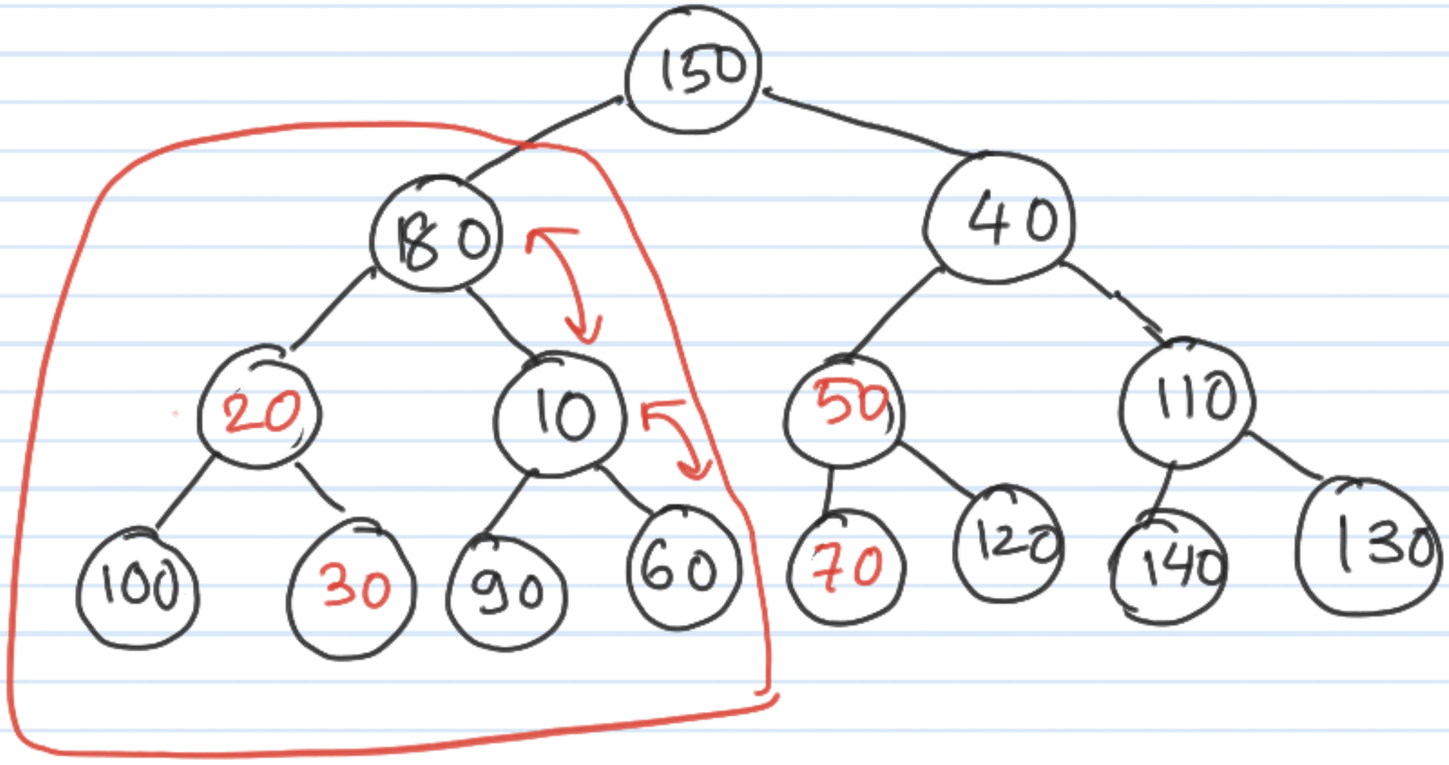


Consider array indices

7, 6, 5 1

150 , 80 , 40, 30, 10, 70, 110 , 100 20, 90 , 60, 50 , 120,

140, 130

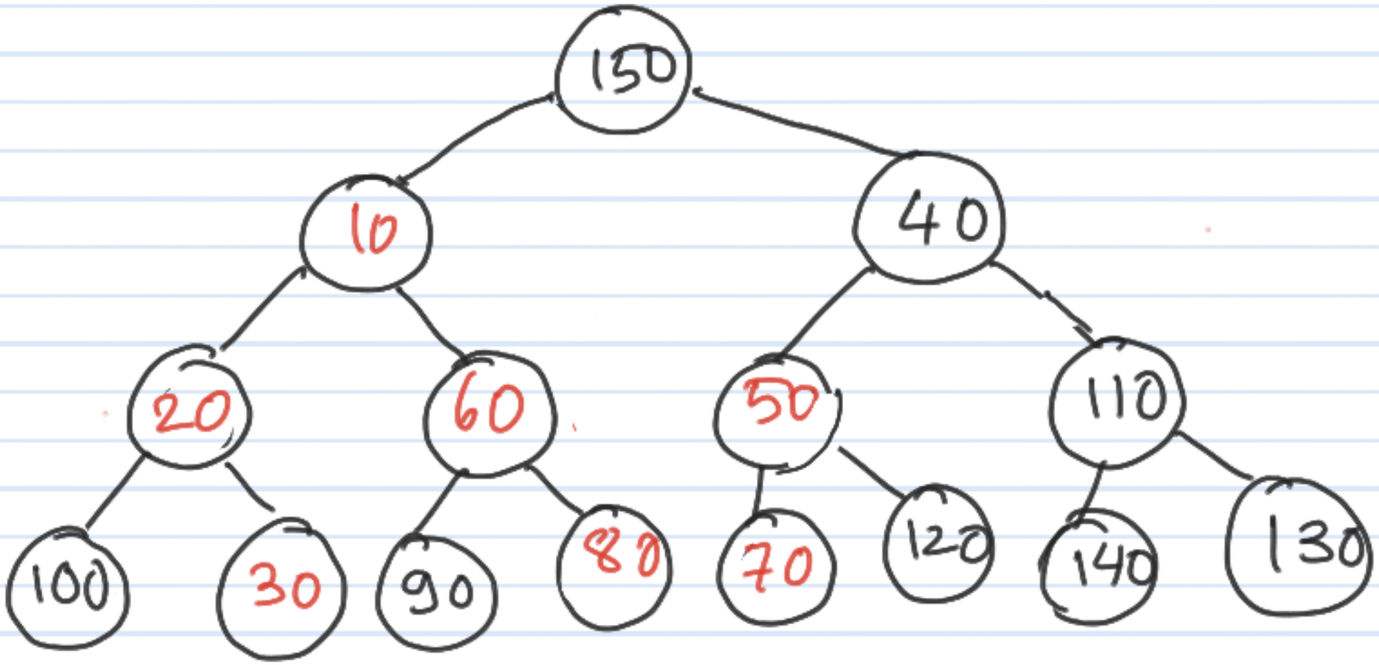


Consider array indices

7, 6, 5 1

150 , 80 , 40, 30, 10, 70, 110 , 100 20, 90 , 60, 50 , 120,

140, 130

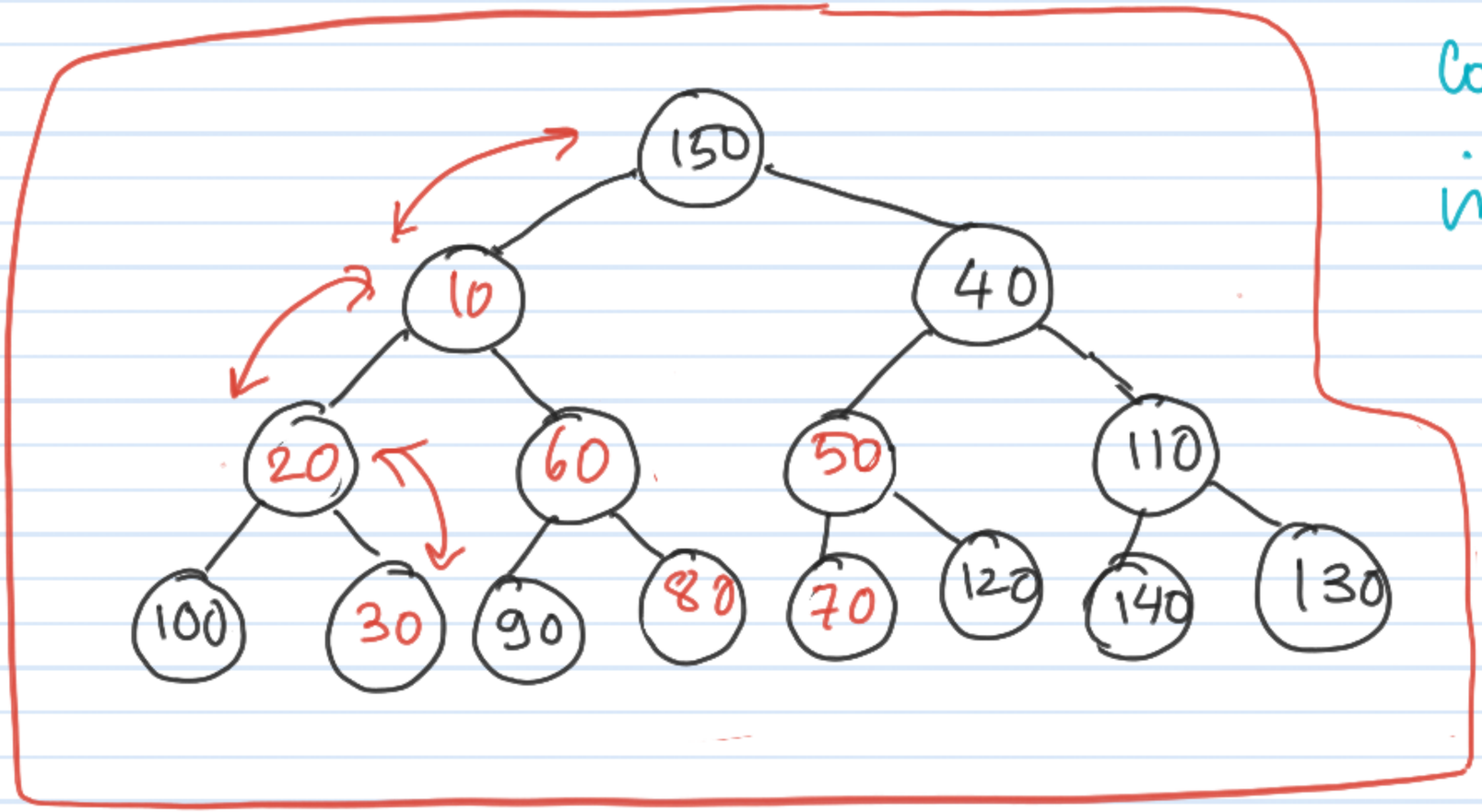


Consider array indices

7, 6, 5 1

150 , 80 , 40, 30, 10, 70, 110 , 100 20, 90 , 60, 50 , 120,

140, 130

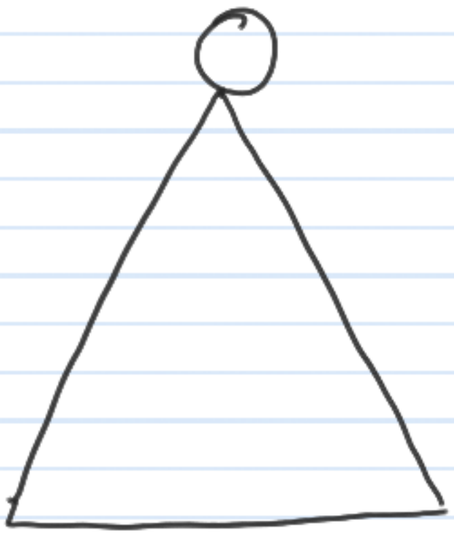


Consider array indices

7, 6, 5 1

INITIALIZING A HEAP WITH n elements.

- insert without order – easy part O(n) time
- Bottom up correction – how do we pay for it?

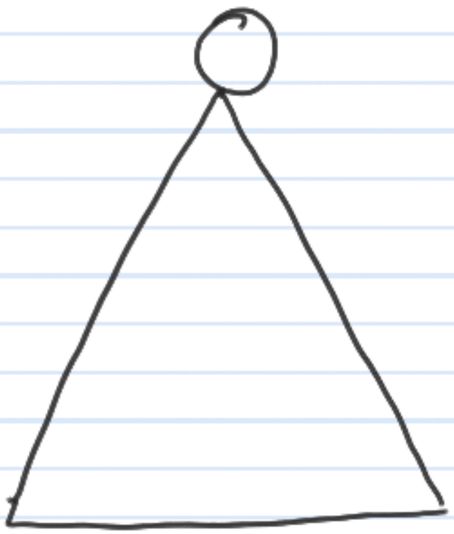


percolate down (i)

- not necessarily constant # of swaps.
- height many swaps at most.

INITIALIZING A HEAP WITH n elements.

- insert without order - easy part O(n) time
- Bottom up correction - how do we pay for it?



percolate down (i)

- not necessarily constant # of swaps.
- height many swaps at most.
- can we bound sum of height of all nodes.

ANALYSIS FOR A COMPLETE BINARY TREE

assume height h

height h : ___ node

height $h-1$: ___ nodes

height $h-2$: ___ nodes

height $h-i$: ___ nodes

$$S = \sum_{i=0}^{h} \_\_ (h-i)$$

ANALYSIS FOR A COMPLETE BINARY TREE

assume height h

height h : ___ node

height $h-1$: ___ nodes

height $h-2$: ___ nodes

height $h-i$: ___ nodes

$$S = h + 2(h-1) + 4(h-2) + \cdots + 2^{h-1}(1)$$

$$2S = 2h + 4(h-1) + 8(h-2) + \cdots + 2^{h}(1)$$

$$2S - S =$$

ANALYSIS FOR A COMPLETE BINARY TREE

assume height h

height h : __ node

height $h-1$: __ nodes

height $h-2$: __ nodes

height $h-i$: ___ nodes

$$S = h + 2(h-1) + 4(h-2) + \cdots + 2^{h-1}(1)$$

$$2S = 2h + 4(h-1) + 8(h-2) + \cdots + 2^{h}(1)$$

$$2S - S = -h + 2 + 4 + \cdots + 2^{h-1} + 2^{h} = (2^{h+1} - 1) - (h+1)$$

$$\simeq O(2^{h+1})$$