COL106 - Data Structures and Algorithms

Priority Queues & Heaps

HEAP DATASTRUCTURE

HEAP is a binary tree which stores Entry objects in its nodes.

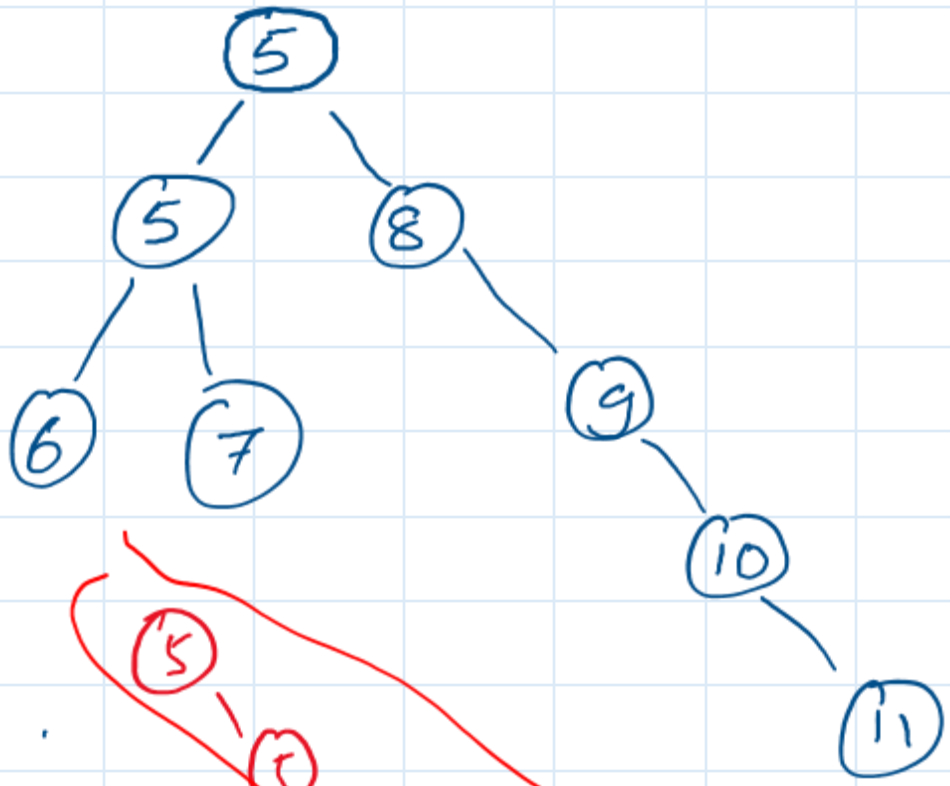
PROPERTY 1 (RELATIONAL PROPERTY)

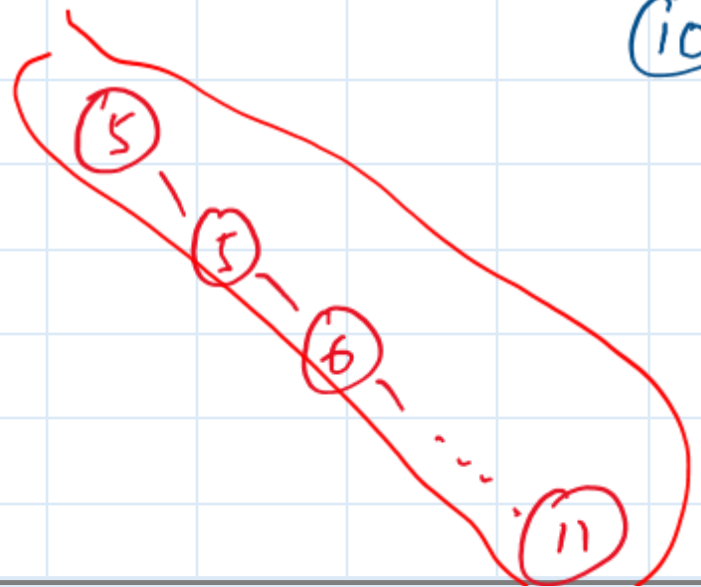
IN A HEAP T, FOR EVERY NODE P, THE KEY STORED AT P IS GREATER THAN OR EQUAL TO THE KEY STORED AT ITS PARENT (EXCEPT ROOT)

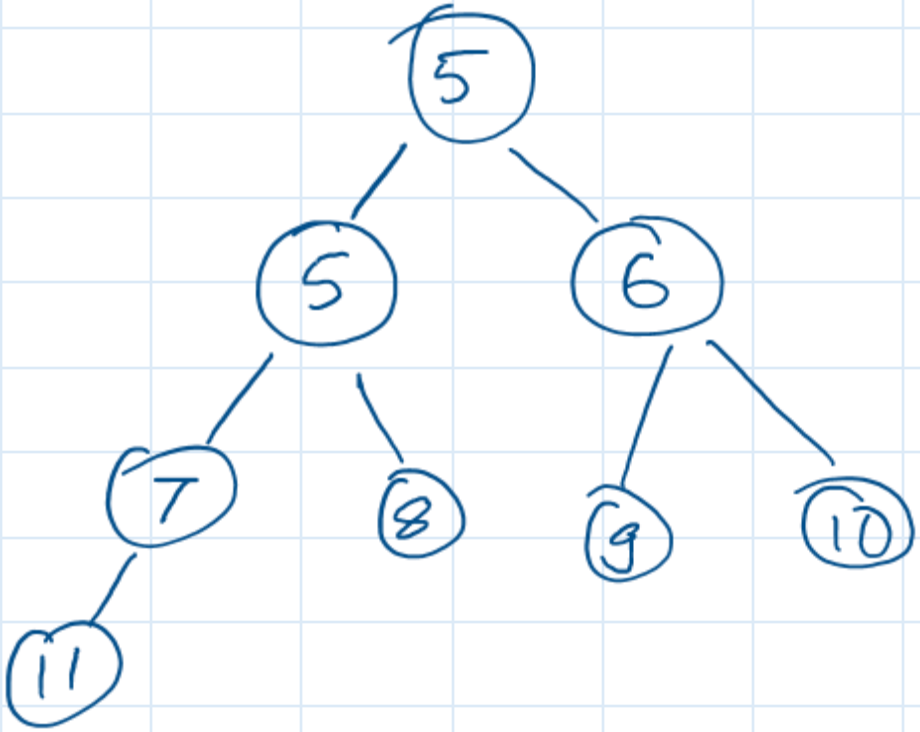
⇒ when we traverse any root - leaf path the keys encountered are strictly non-decreasing

⇒ the root contains the minimal key.

(for simplicity we only show keys stored in each node of the tree)







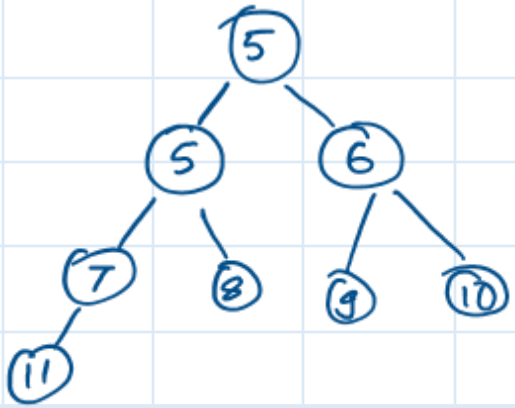
WE SHOULD AVOID BAD TREE CONFIGURATIONS !

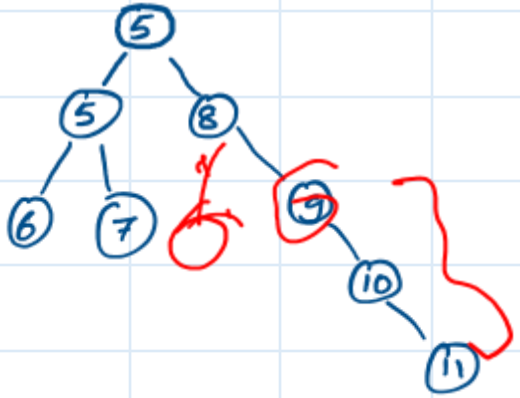
PROPERTY 2 (STRUCTURAL PROPERTY)

A heap T with height h should satisfy **complete** binary tree property.

(i) if levels 0, 1, ..., h-1 of T have maximal number of nodes possible

(ii) remaining nodes at level h reside in the leftmost possible positions at level h.



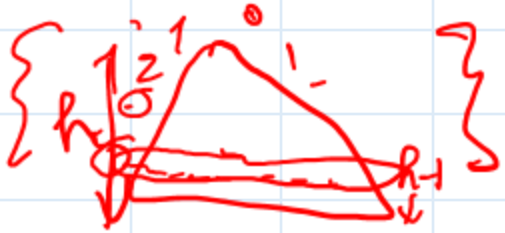


$2^h - 1 \le n \le 2^{h+1} - 1$

A Heap with n entries has height $h = \lfloor \log_2 n \rfloor$

T is a complete binary tree upto $h-1$ levels.

$\Rightarrow$ # of entries $= 1 + 2 + 4 + \cdots + 2^{h-1}$



$= 2^h - 1$

Number of entries in level h is at least 1. and at most 2^h

$\Rightarrow \boxed{n \geq 2^h - 1 + 1} = 2^h$ $\qquad n \leq 2^h - 1 + 2^h = 2^{h+1} - 1$

$\Rightarrow h \leq \log_2 n$ $\qquad h \geq \log(n+1) - 1$

$\Rightarrow h = \lfloor \log n \rfloor$

IF WE CAN IMPLEMENT PRIORITY QUEUE USING HEAPS S.T. ALL OPERATIONS ARE IN TIME PROPORTIONAL TO THE HEIGHT OF HEAP, THEN THEY RUN IN $O(\log n)$ TIME

Finding minimum key entry is $O(1)$

always at the root (why?)

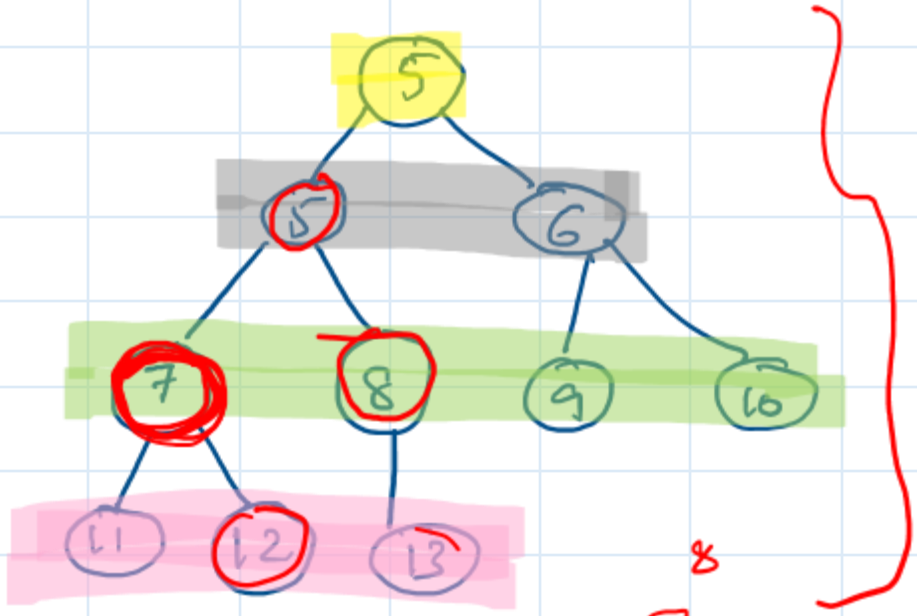
Implementing Heaps

Parent(i): return $\lfloor i/2 \rfloor$

Left(i): return $2i$

Right(i): return $2i+1$

Parent(9) = 4



bin(4) = 100

8

Array A	5	5	6	7	8	9	10	11	12	13
	1	2	3	4	5	6	7	8	9	10
Level	0	1		2				3		

8

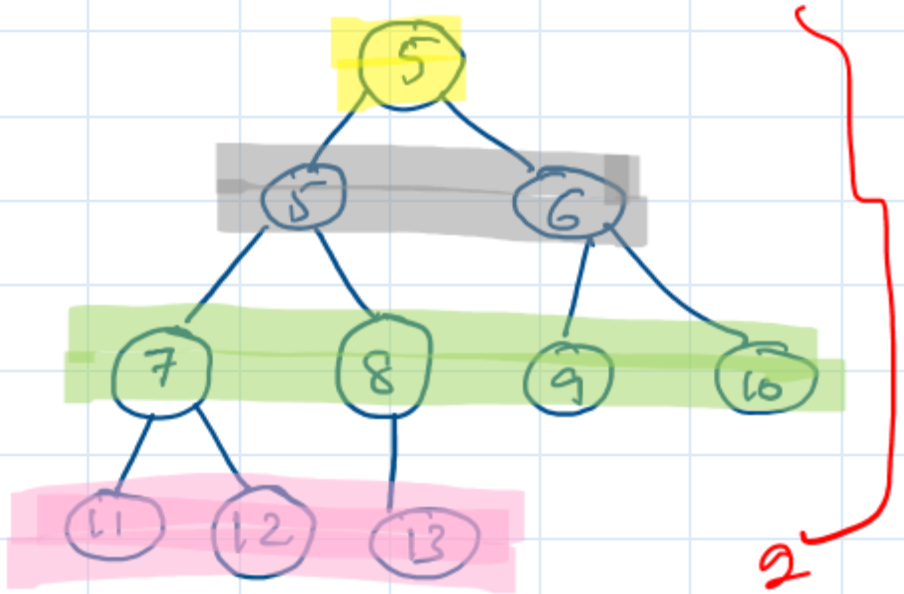
Heap Property: $A[\text{Parent}(i)] \leq A[i]$

Implementing Heaps

Parent(i): return $\lfloor i/2 \rfloor$ right shift bin(i)

Left(i): return $2i$ left shift bin(i)

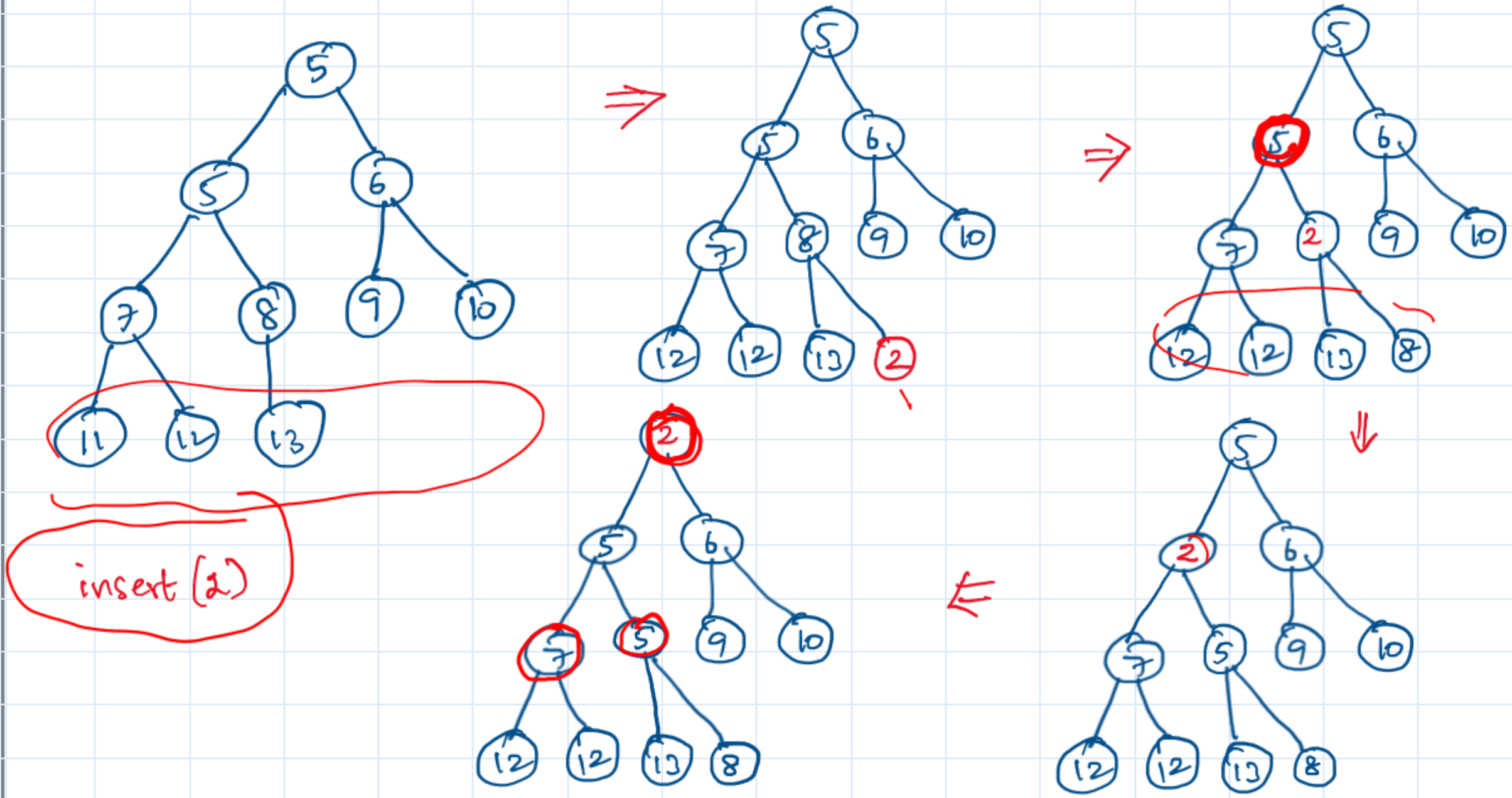
Right(i): return $2i+1$ left shift bin(i) + [lsb(bin(i)) + 1]



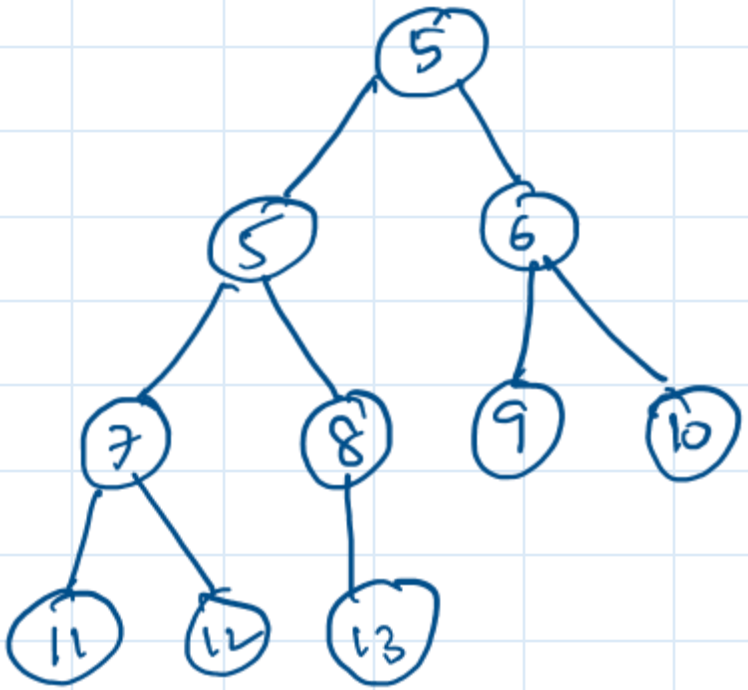
Array A	5	5	6	7	8	9	10	11	12	13
	1	2	3	4	5	6	7	8	9	10
Level	0	1		2				3		

Heap Property: $A[\text{Parent}(i)] \leq A[i]$

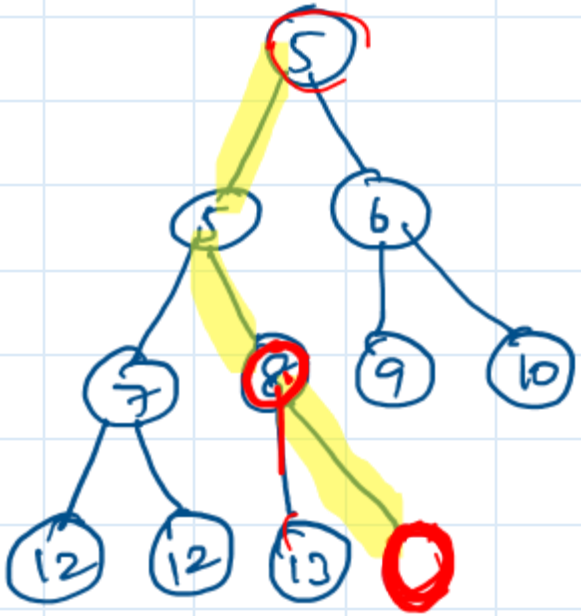
Inserting an element



Inserting an element

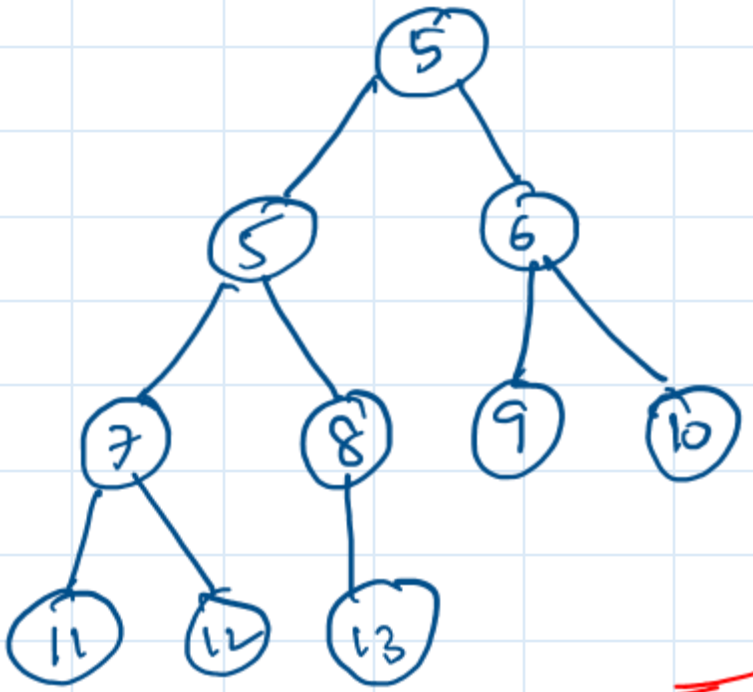


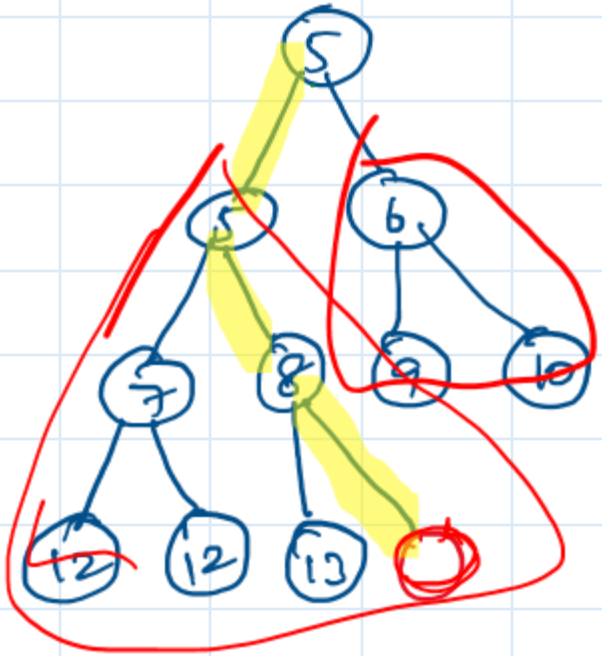
insert (2)



1. Enlarge heap
2. Consider path from root to inserted node
3. Find topmost element on path with higher priority than inserted element
4. Insert new element by shifting down other elements on the path

Inserting an element





1. Enlarge heap
2. Consider path from root to inserted node
3. Find topmost element on path with higher priority than inserted element
4. Insert new element by shifting down other elements on the path

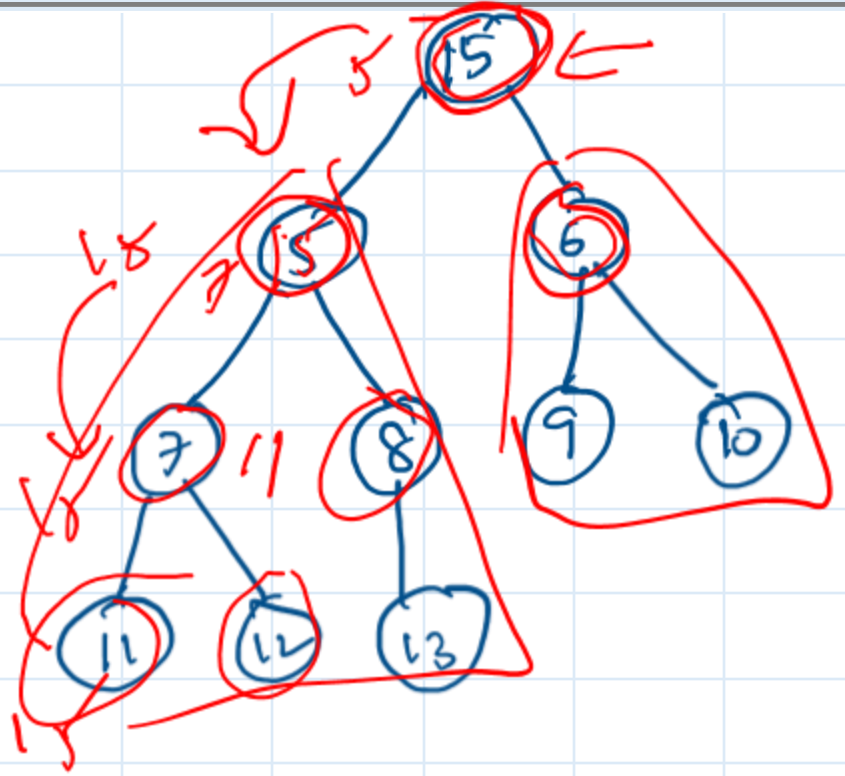
Correctness?

insert (2)

Claim: Only nodes whose contents change are the ones on the path

Heapify(i)

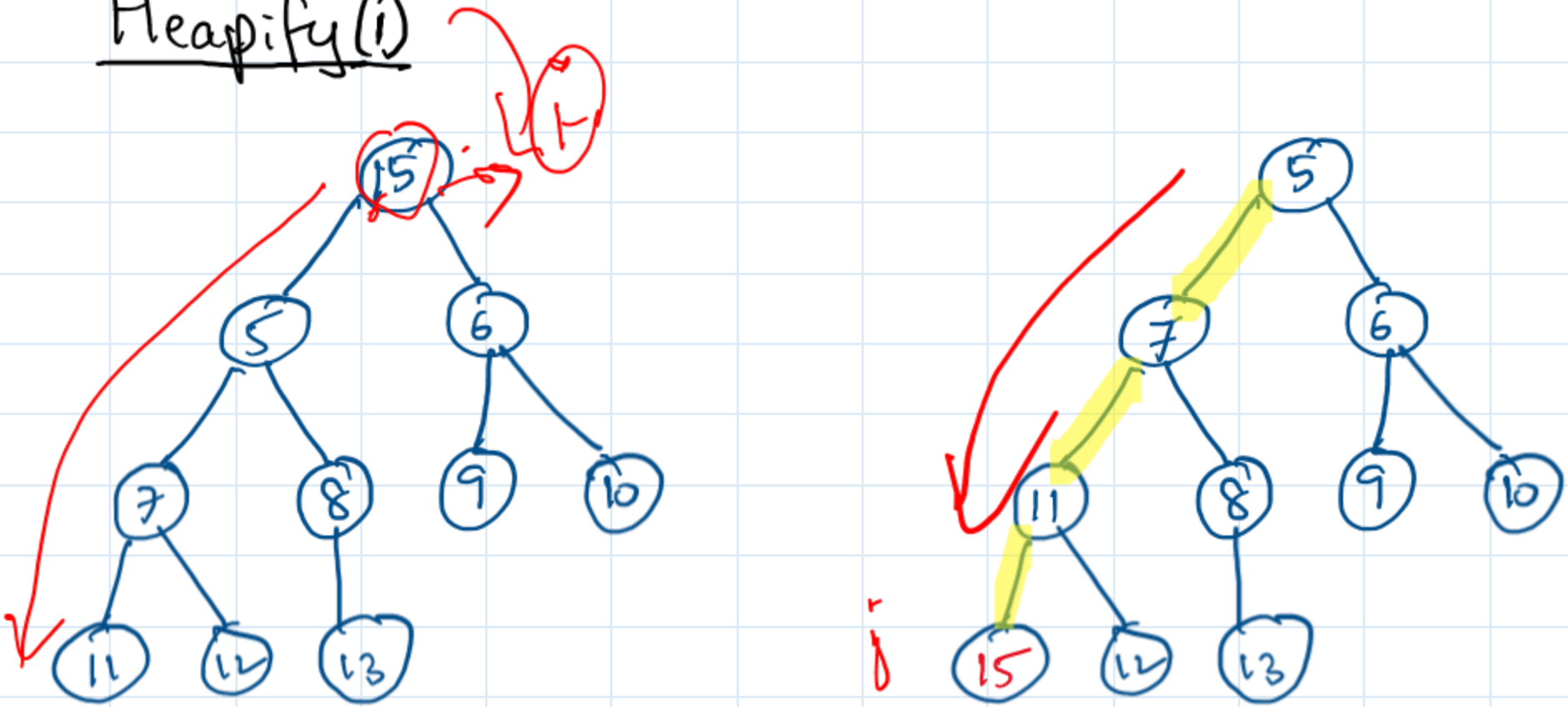
Given array A, index i with A[i] violating heap property (but binary trees rooted at Left(i) & Right(i) are heaps) Heapify(i) should make binary tree rooted at i a heap. by moving A[i] down.



Heapify(1)

<u>Idea:</u> Swap A[i] with smaller among its 2 children

Heapify(i)



Complexity:

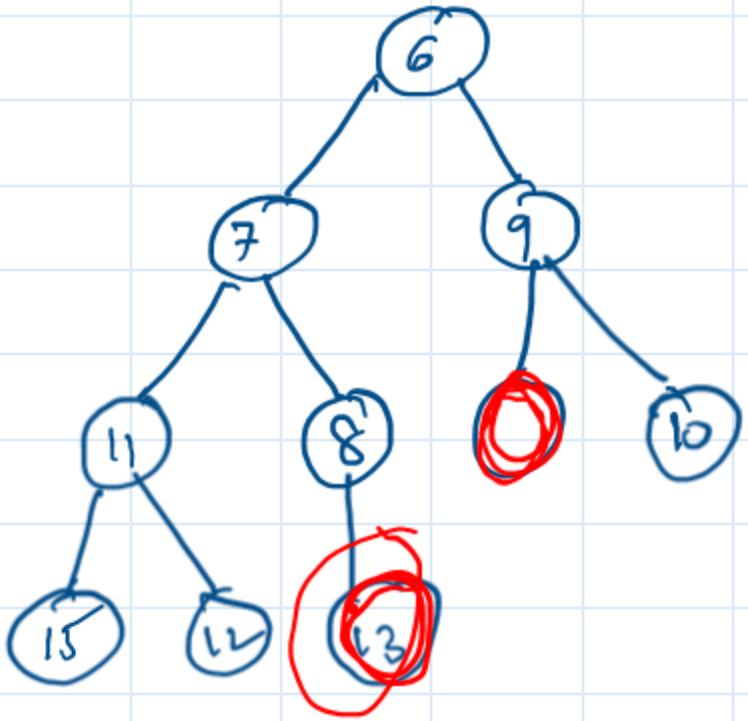
$O(\log n)$

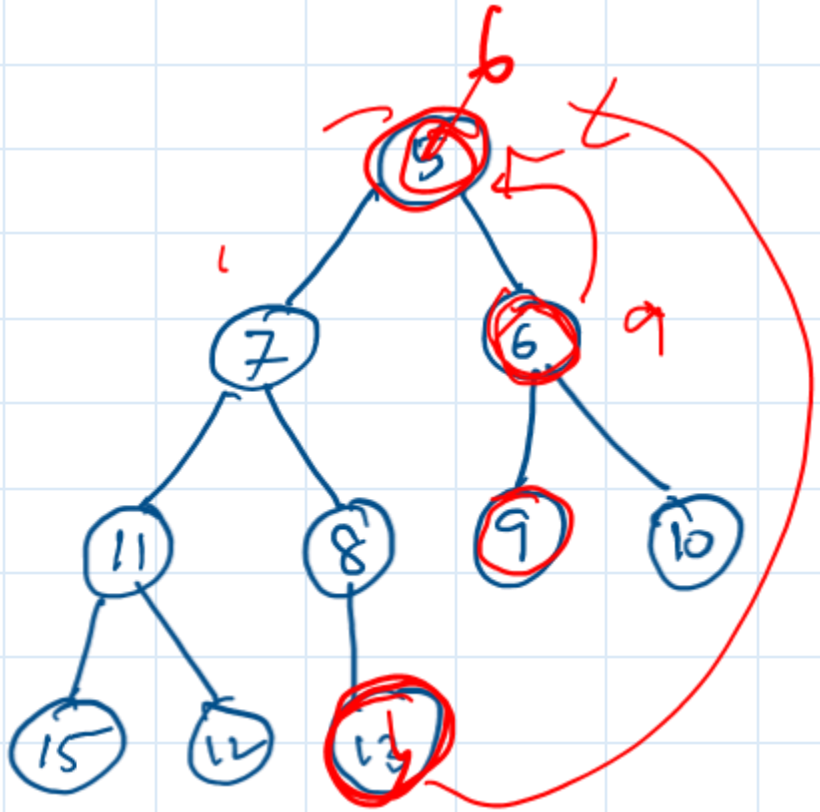
$A[left(j)], A[right(j)] \geq A[i]$

- Last node on path (say j) has both $A[left(j)]$, $A[right(j)]$ larger than $A[i]$
- All elements on path have lower priority than their siblings
- All elements on this path are moved up & $A[i]$ went to location j.

Remove minimum element in a heap

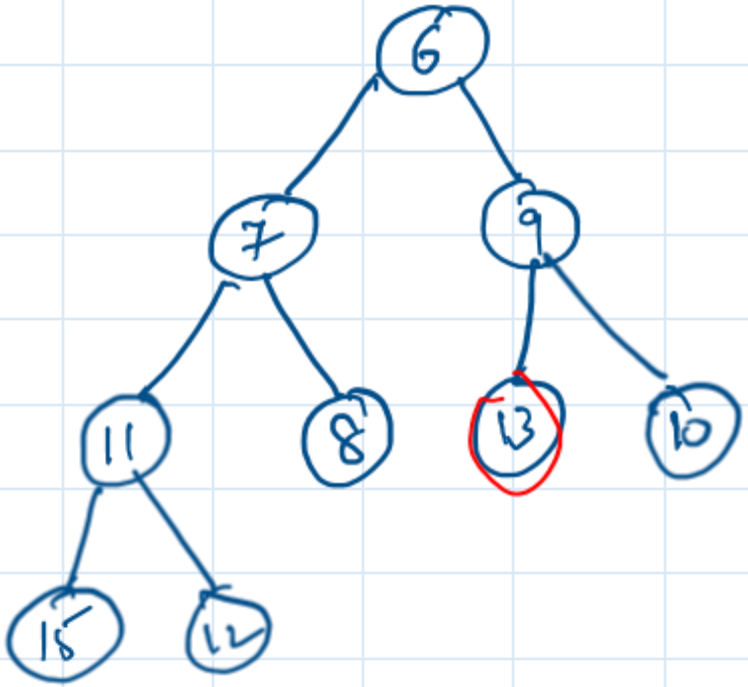
- Can find min in $O(1)$ time
- Delete & move up one of the children
- Empty location moves down the tree



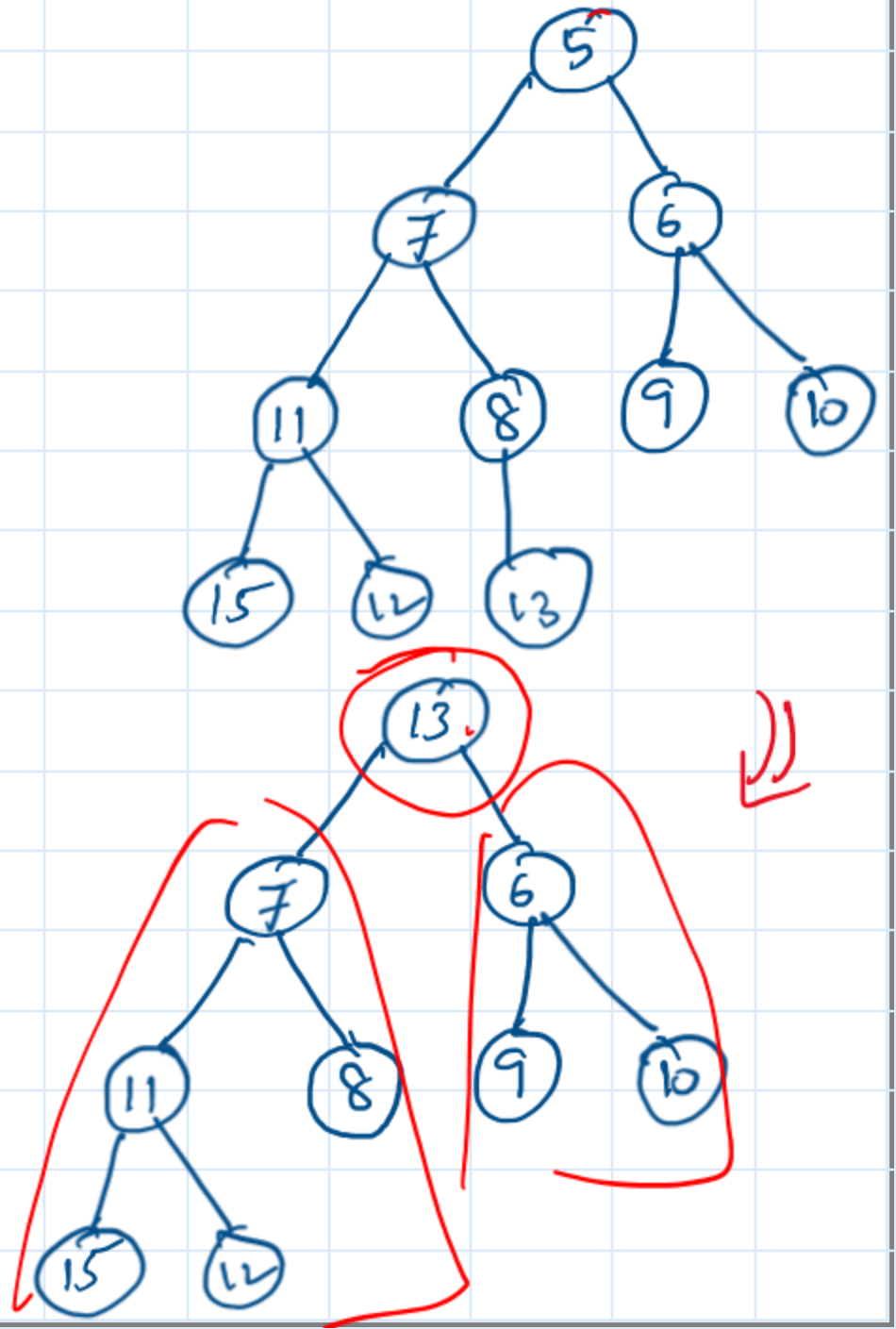


Removing min in a heap

Idea: Heap is going to have one element less. Why not fill the root with last element of heap.



Heapify(1)



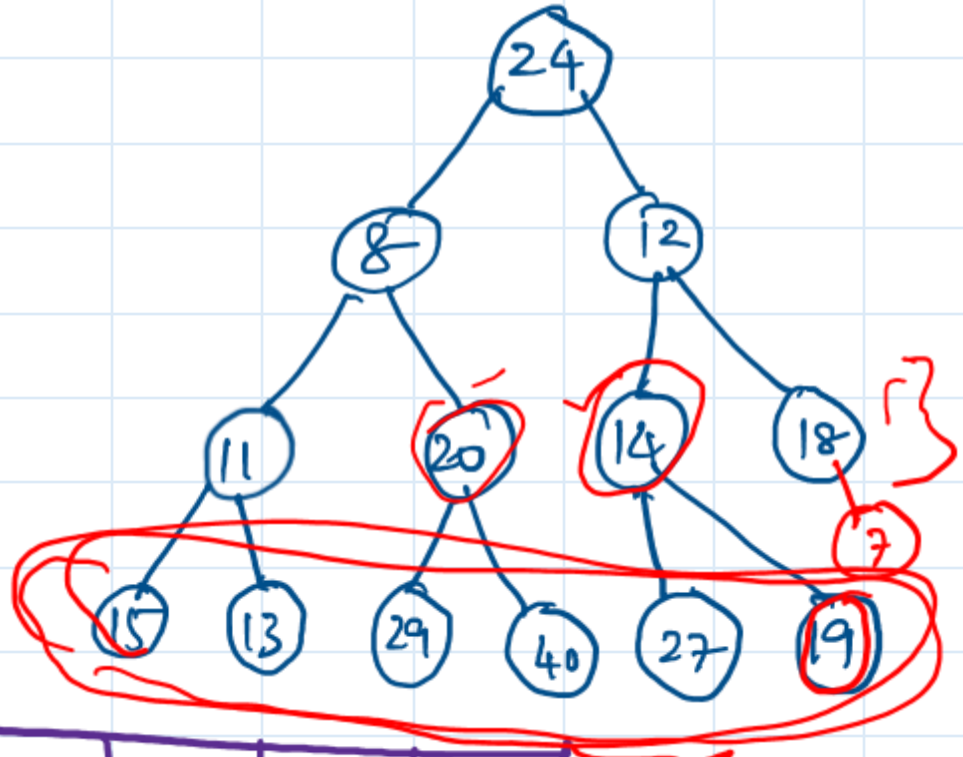
Building a heap

- Repeatedly insert an element in to the heap. $\sum_{i=1}^{n} O(\log i) = O(n \log n)$
- Bottom up construction: leaves are heaps, so start from there.

BUILD-HEAP (A)

for $i \leftarrow \lfloor \frac{n}{2} \rfloor$ down to 1

HEAPIFY(i)



24	8	12	11	20	14	18	15	13	29	40	27	19	7
1	2	3	4	5	6	7	8	9	10	11	12	13	

Building a Heap

- **Correctness:** induction on i, all trees rooted at $j > i$ are heaps.
- **Analysis:** Idea: Time taken by Heapify(i) = O(height of subtree rooted at i)
  - For $n/2$ nodes at height 1, heapify() requires ≤ 1 swap each
  - For $n/4$ " " 2, " " ≤ 2 "

    ⋮

    $n/2^i$ - - - i, - - - - - $\leq i$

$$\text{Total swaps} = O\left(\left(\sum_{i=1}^{\log n} \frac{i}{2^i}\right) \cdot n + 1\right) = O(n)$$

<u>claim:</u> $\sum_{i=1}^{n} \frac{i}{2^i} \leq 2$

<u>Proof:</u> $\sum_{i=0}^{\infty} x^i = \frac{1}{1-x}$ if $|x|<1$

$$\sum_{i=0}^{\infty} i x^{i-1} = \frac{1}{(1-x)^2}$$

$$\Rightarrow \sum_{i=0}^{\infty} i x^i = \frac{x}{(1-x)^2}$$

plug in $x = 1/2$