Lecture 24 -
Makeup for ProgTest2

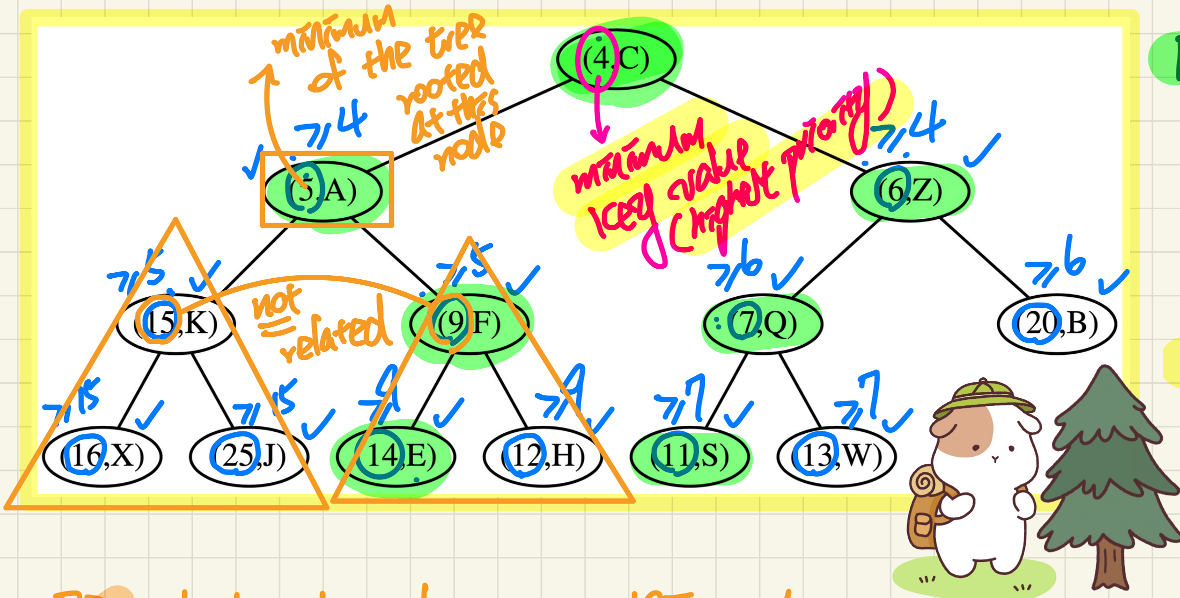
Lecture

Priority Queue

Heaps -
Examples and Properties

Heaps: Relational Properties of Keys

Property: Each non-root node **n** is s.t. **key**(**n**) ≥ **key**(parent(**n**))



P1. Any leaf-to-root path has a sorted seq of keys.

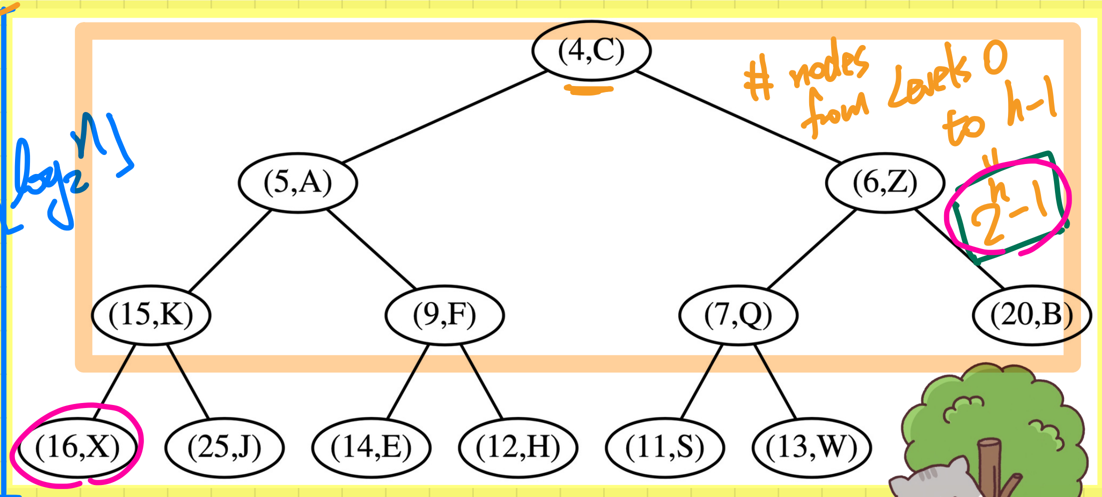
P2. the minimum key exists in the root entry.

P3. key values between LST and RST are not related.

Heaps: Structural Properties of Nodes

Property: The tree is a complete Binary Tree

$h = \lfloor \log_2 n \rfloor$



$n = 13$

$\lfloor \log_2 n \rfloor = \lfloor 3.\cdots \rfloor = 3$

Min # of nodes: $(2^h - 1) + 1$

Max # of nodes: $(2^h - 1) + 2^h = 2^{h+1} - 1$

# of nodes at level h = $n - (2^h - 1)$

Example Heaps

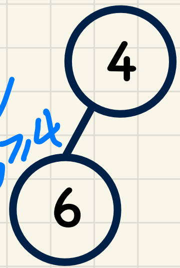
< relational
structural

Example 1

✓

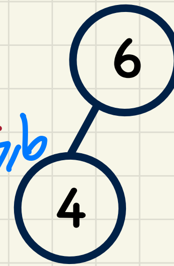
smallest possible one-node heap

Example 2



✓ $6 \geq 4$ ✓

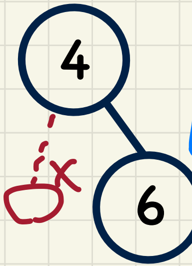
Example 3



✗ $4 \geq 6$ ✗

violating the relational property

Example 4



✓ $6 \geq 4$

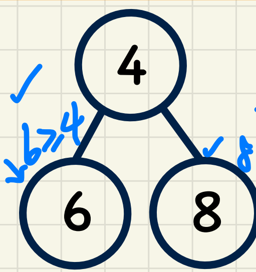
✗

violating structural property

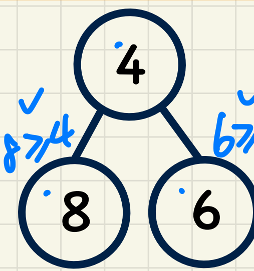
Example 5

Example 6

full BTs
⇒ complete BTs



✓ $6 \geq 4$ ✓ $8 \geq 4$



✓ $8 \geq 4$ ✓ $6 \geq 4$

✓

Lecture

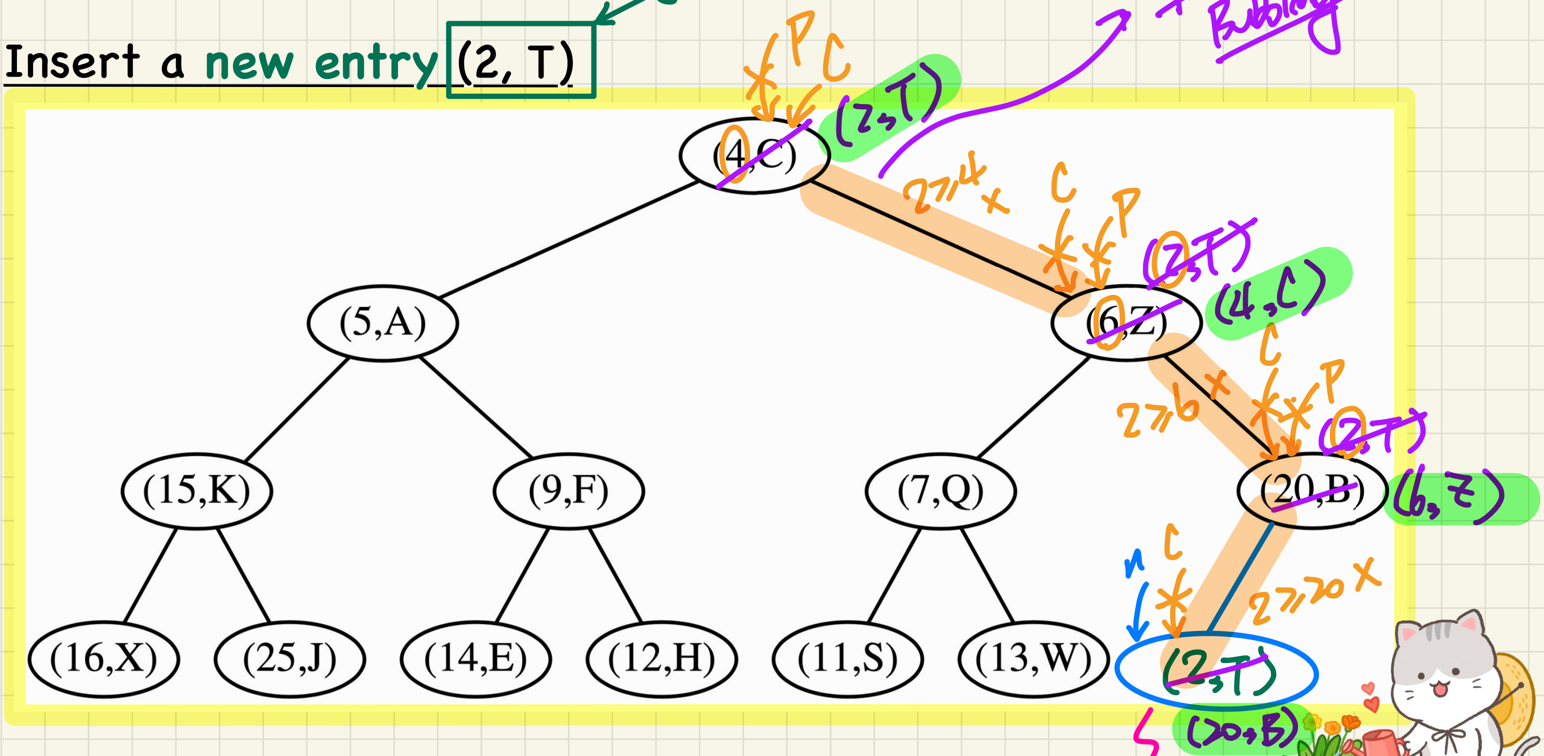
Priority Queue

Heaps - Insertions

Heap Operations: Insertion

Insert a new entry $\boxed{(2,\ T)}$ ← e

Up-Heap Bubbling.



must be right-most at level h in order to preserve structural property

Lecture

Priority_Queue

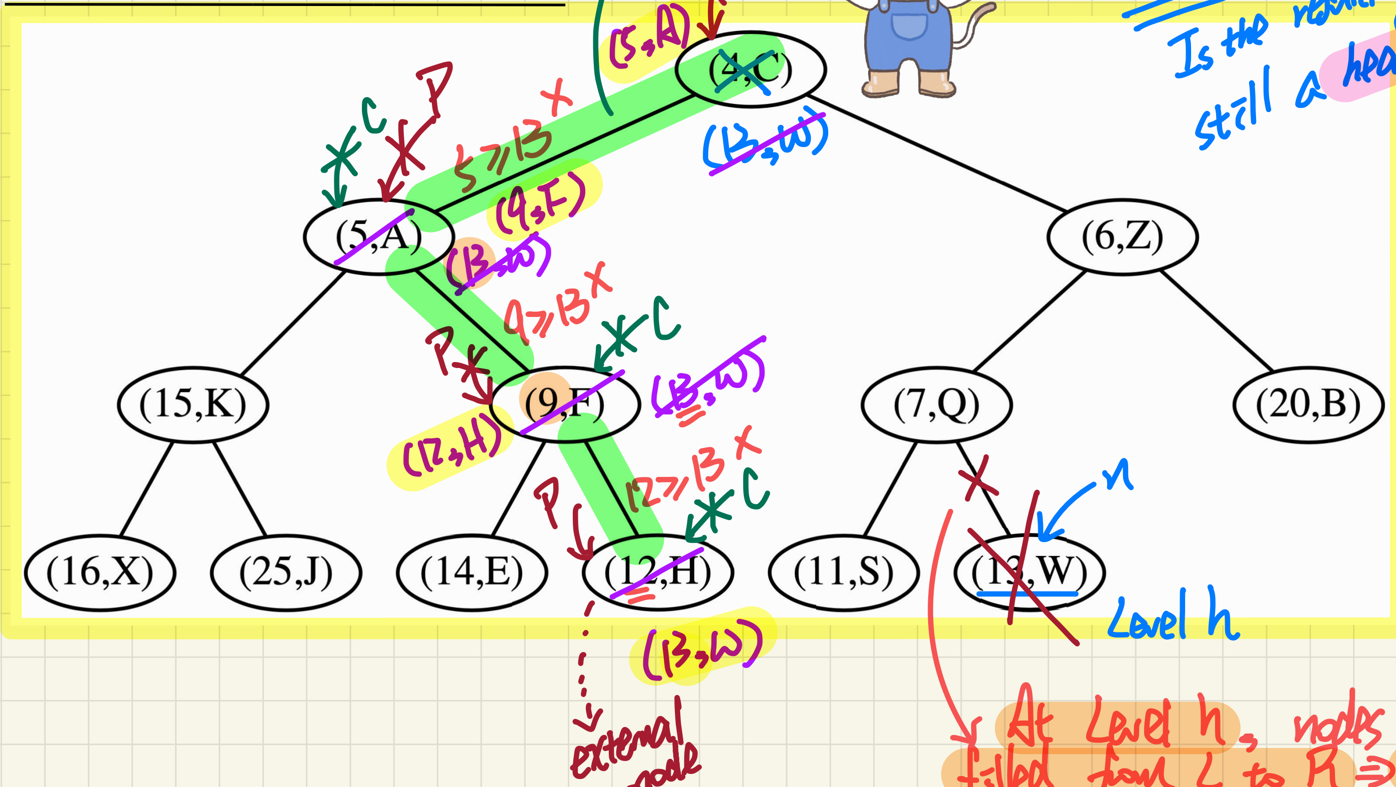
Heaps - Deletions

Heap Operations: Deletion

root-to-leaf path (down-heap bubbling)

Delete the root/minimum

Exercise
Is the resulting tree still a heap?



At Level h, nodes are still filled from L to R ⇒ complete BT

Lecture

Priority Queue

Heaps -
Top-Down Heap Construction

Top-Down Heap Construction

Problem: Build a heap out of ***N*** entires, supplied one at a time.

- Initialize an ***empty heap h***.
- As each new entry **e** = (k, v) is supplied, **insert e** into ***h***.

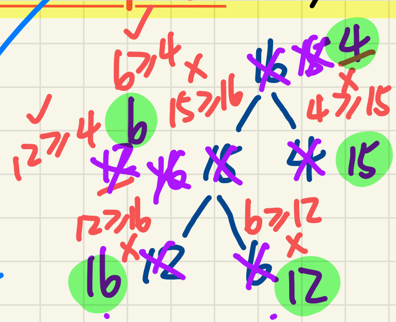
first inserted to level 2

Exercise: Build a heap out of the following 15 keys:

<16, 15, 4, 12, 6, 7, 23, 20, 25, 9, 11, 17, 5, 8, 14>

Assumption: Key values supplied one at a time.

first inserted to level 1

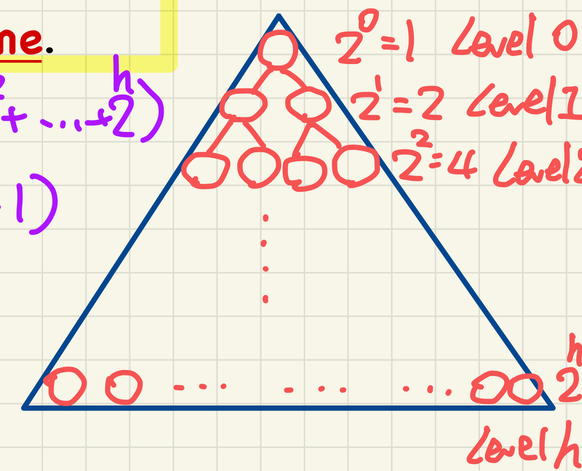


RI: # nodes at level, # upheap bubbling steps

$$1 + 2^1 \cdot 1 \le \log n$$
$$+ 2^2 \cdot 2 \le \log n$$
$$+ \cdots$$
$$+ 2^h \cdot h \quad \log n$$

root

$$* \le 1 + \log_2 n \cdot (2^1 + 2^2 + \cdots + 2^h)$$
$$= 1 + \log_2 n \, (n - 1)$$
$$O(n \cdot \log n)$$



Exercise: Complete inserting the remaining keys to the heap.

Lecture

Priority Queue

Heaps -
Bottom-Up Heap Construction

Bottom-Up Heap Construction

Problem: Build a heap out of N entires, supplied all at once.

- **Assume**: The resulting heap will be ***completely filled*** at **all** levels.
  $\Rightarrow N = 2^{h+1} - 1$ for some ***height*** $h \geq 1$ $\quad [\ h = (\log(N+1)) - 1\ ]$
- Perform the following steps called *Bottom-Up Heap Construction*:

  Step 1: Treat the first $\boxed{\frac{N+1}{2^1}}$ list entries as heap roots.
  $\therefore \frac{N+1}{2^1}$ heaps with height 0 and size $\boxed{2^1 - 1}$ constructed.

  Step 2: Treat the next $\boxed{\frac{N+1}{2^2}}$ list entries as heap roots.
  - Each ***root*** sets two heaps from **Step 1** as its ***LST*** and ***RST***.
  - Perform ***down-heap bubbling*** to restore **HOP** if necessary.

  $\therefore \frac{N+1}{2^2}$ heaps, each with height 1 and size $\boxed{2^2 - 1}$ constructed.

  ...

  Step $h+1$: Treat next $\boxed{\frac{N+1}{2^{h+1}}} = \frac{(2^{h+1}-1)+1}{2^{h+1}} = 1$ list entry as heap root.
  - Each ***root*** sets two heaps from **Step h** as its ***LST*** and ***RST***.
  - Perform ***down-heap bubbling*** to restore **HOP** if necessary.

  $\therefore \boxed{\frac{N+1}{2^{h+1}} = 1}$ heap, each with height h and size $\boxed{2^{h+1} - 1}$ constructed.

Exercise: Build a **heap** out of the following 15 keys:

<16, 15, 4, 12, 6, 7, 23, 20, 25, 9, 11, 17, 5, 8, 14>

Assumption: Key values supplied **all at once**.

15 · 3 · 8 · 4 · $\frac{16}{2^{3+1}} = 1$ · 3 · $2^{3+1} - 1 = 15$

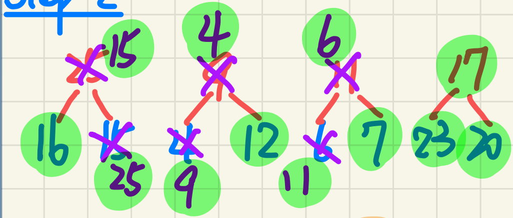
Step 3: ① # heaps: $\frac{N+1}{2^3}$ ② size of each heap: $2^3 - 1 = 7$ ③ height of each heap: 2

50% Step 1

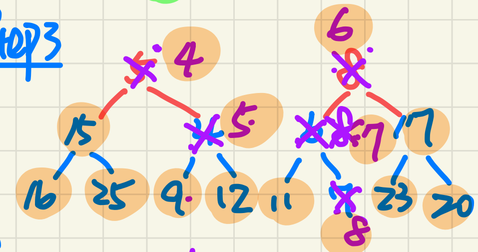
8 heaps, size 1, height 0

16 15 4 12 6 7 23 20

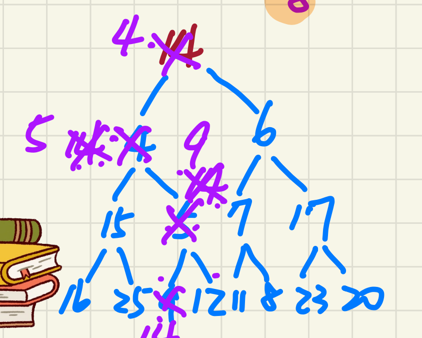
25% Step 2



12.5% Step 3



Step



<u>**Lecture**</u>

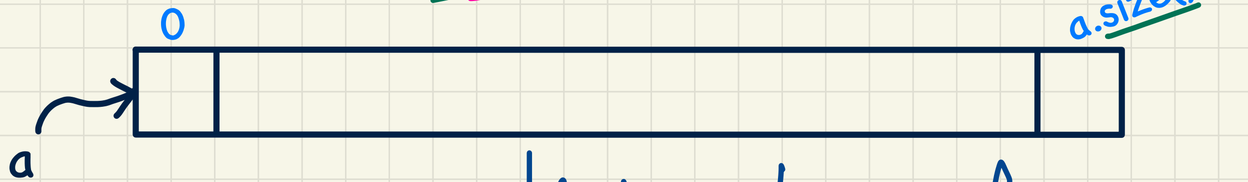
<u>**Priority Queue**</u>

Heaps -
Heap Sort Algorithm

Heap Sort: Ideas

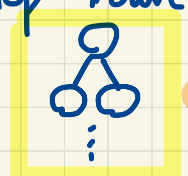
$O(N \cdot \log N)$

N entries



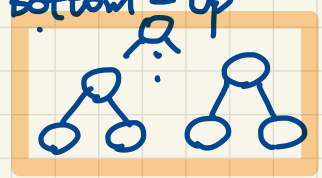
Construct a heap out of N entries

(A) Top-Down



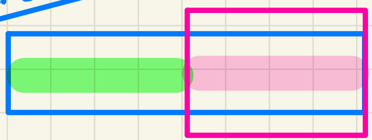
$O(N \cdot \log N)$

(B) Bottom-Up



$O(N)$

Selection Sort

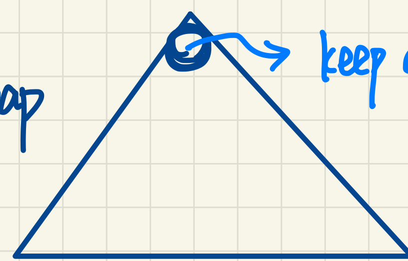


select the min from unsorted portion & put it to the front/end of the list

≈ Selection Sort

exploit the HOP (relational property): root stores entry with min key

heap



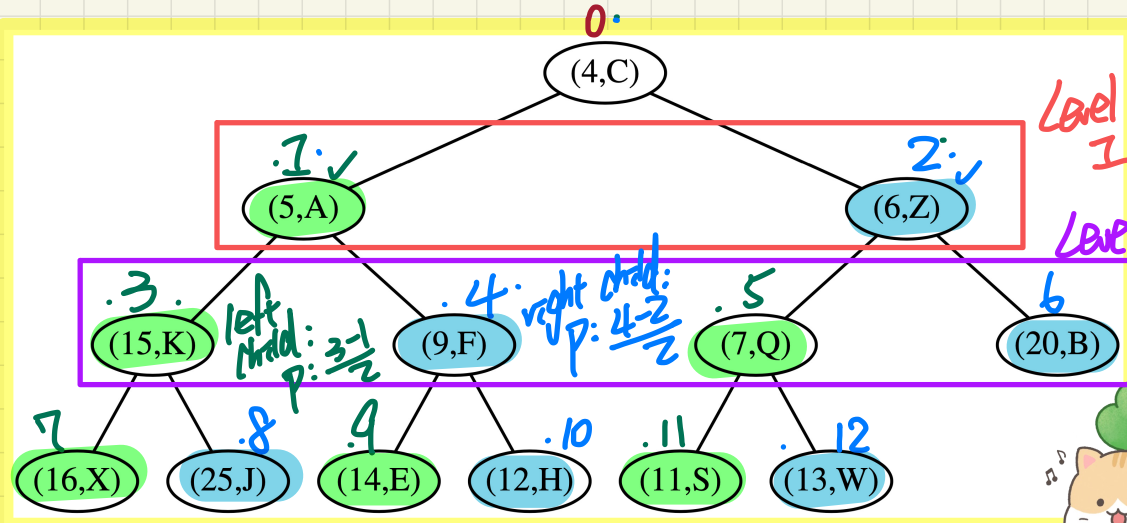
keep deleting the root until the heap is empty. N deletions, each $O(\log N) \Rightarrow O(N \cdot \log N)$

Lecture

Priority Queue

Heaps -
Array-Based Implementation

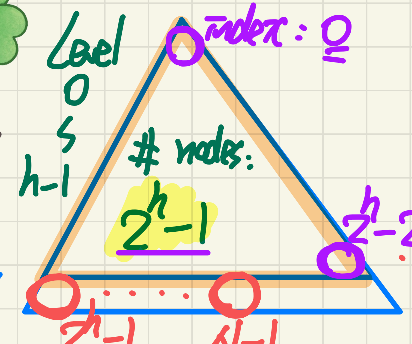
Array-Based Representation of a Complete BT



Exercise

What if the BT is not complete?

(bad for space util.)



$$
index(x) = \begin{cases} 0 & \text{if } x \text{ is the root} \\ 2 \cdot index(\,parent(x)\,) + 1 & \text{if } x \text{ is a left child} \\ 2 \cdot index(\,parent(x)\,) + 2 & \text{if } x \text{ is a right child} \end{cases}
$$

L1 L2

0	1	2	3	4	5	6	7	8	9	10	11	12
(4, C)	(5, A)	(6, Z)	(15, K)	(9, F)	(7, Q)	(20, B)	(16, X)	(25, J)	(14, E)	(12, H)	(11, S)	(13, W)