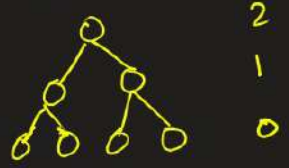
Stack, Queues, Priority Queues & Heaps Lecture 3

Saturday, 10 August 2024 2:05 PM

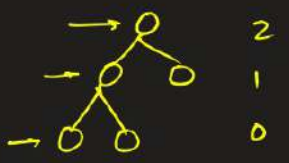
Full & Complete Binary Trees

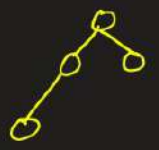


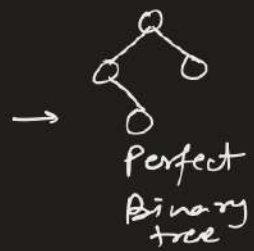
Full binary tree

height = h

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







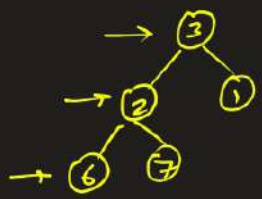
Perfect Binary tree

Complete binary tree

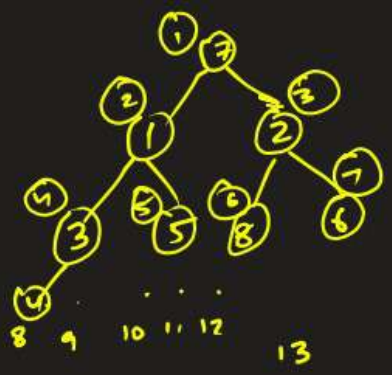
height = h

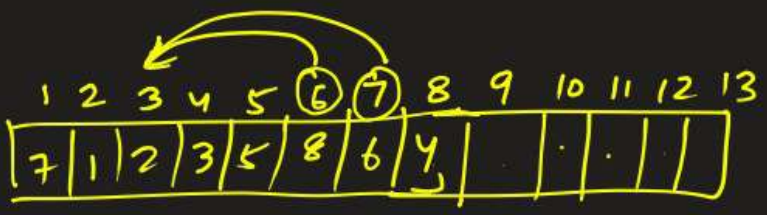
nodes $\in [2^h, 2^{h+1} - 1]$ ✓

Heaps are complete binary trees



array representation of complete binary tree





i → Left child → $2 * i$

right child → $2 * i + 1$

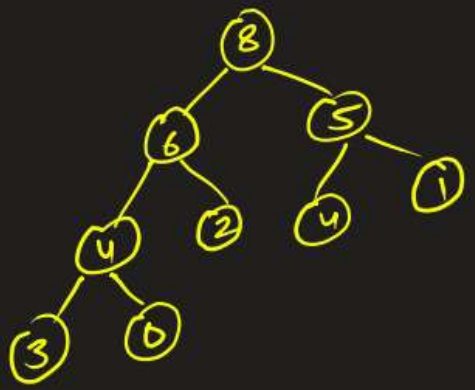
parent [i] = $\lfloor \frac{i}{2} \rfloor$

Heaps

are complete binary trees

Max heap :- Every node ≥ its children

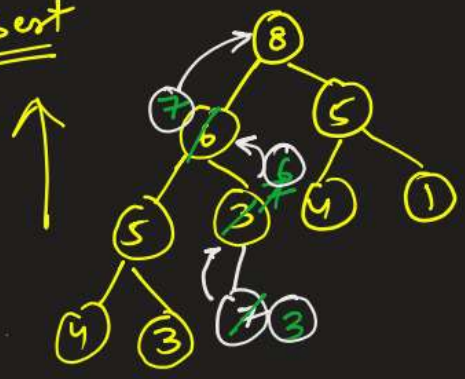
Min heap :- Every node ≤ its children



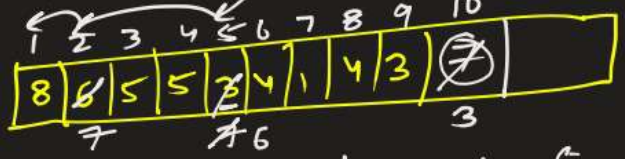
complete binary tree ✓

Max heap ✓

Insert



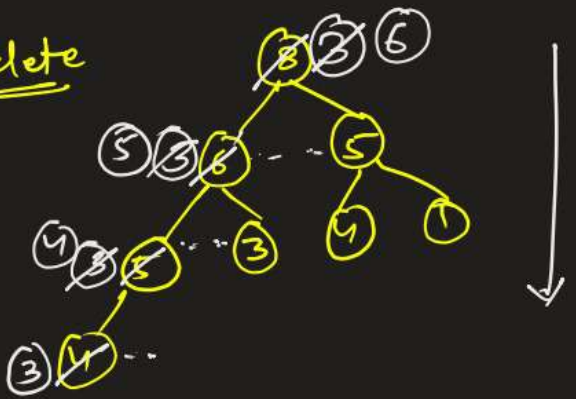
insert (7)

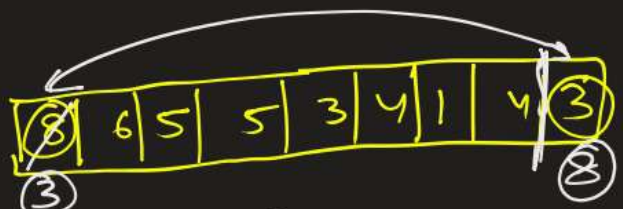


insert at the end and fix the structure bottom to top

Time complexity = $O(\log n)$

Delete



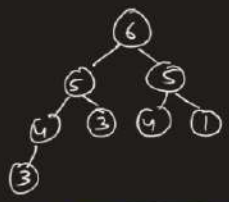


Delete (top)

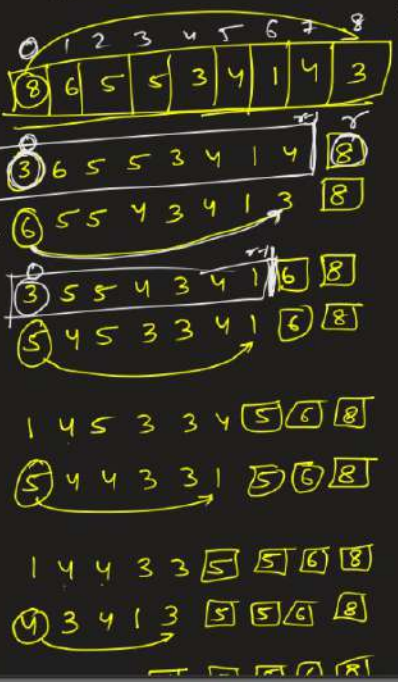
Fix the position of root :-
top to bottom

top to bottom

Time = O(log n)

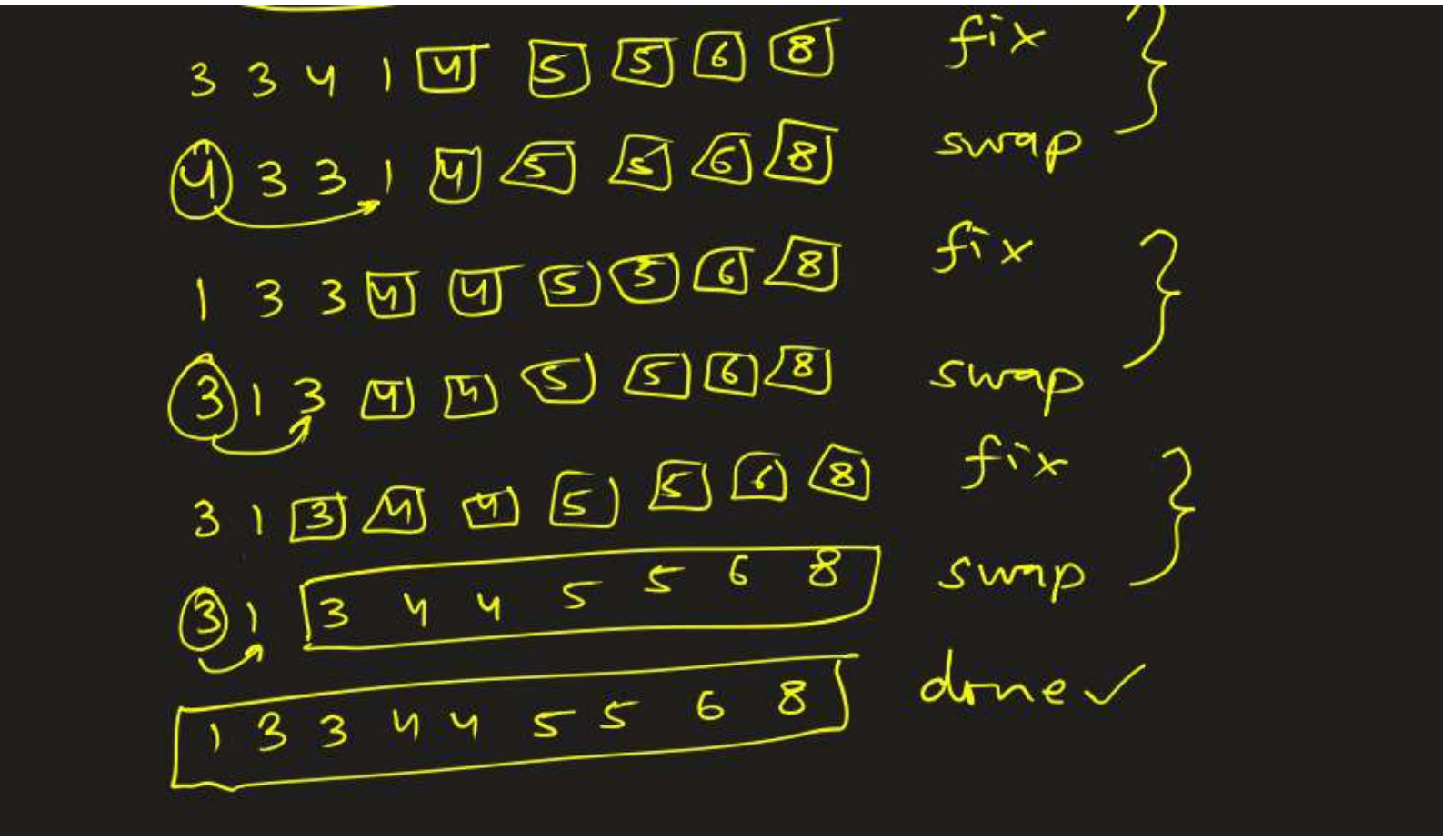


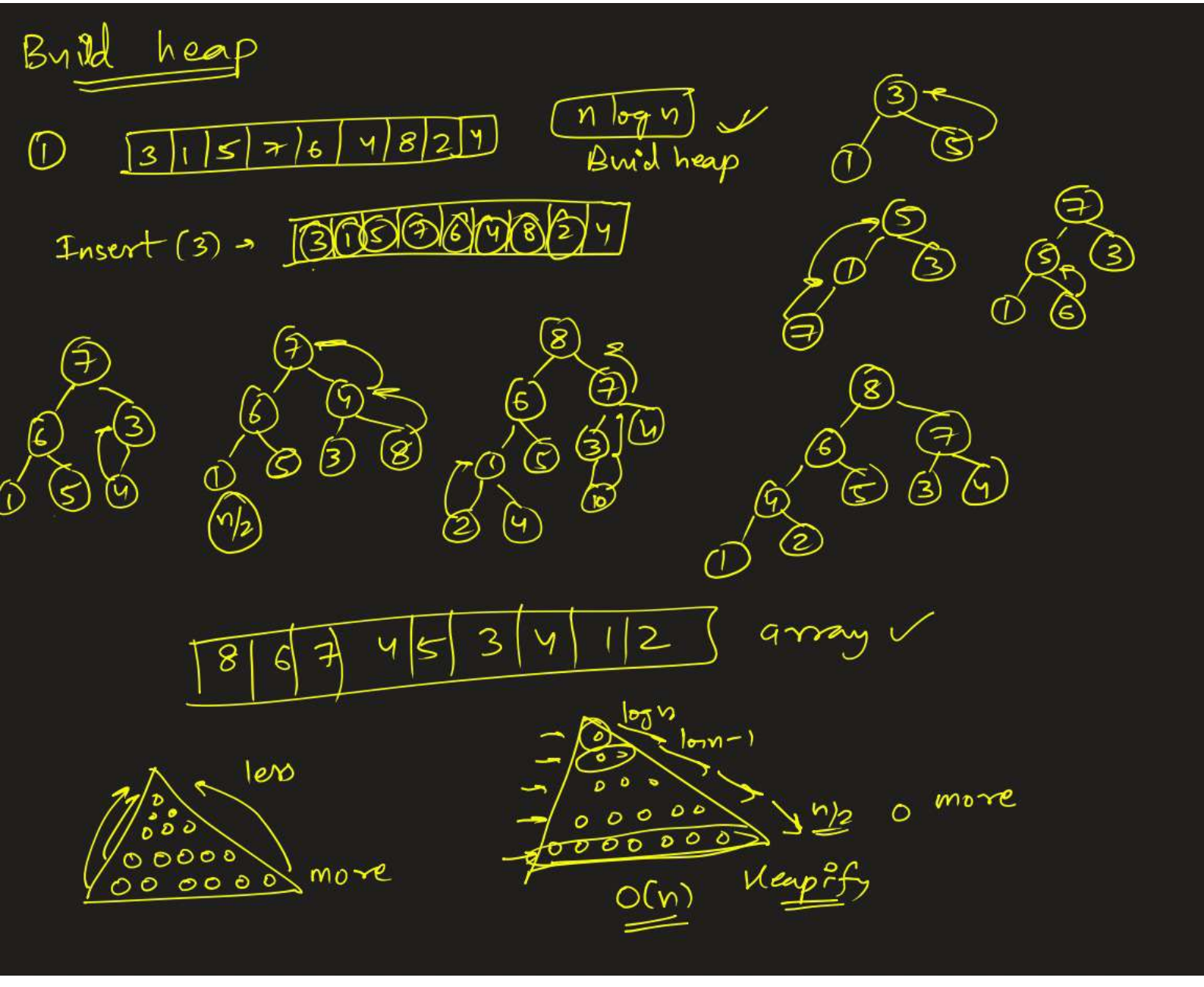
n=9



Heap Sort

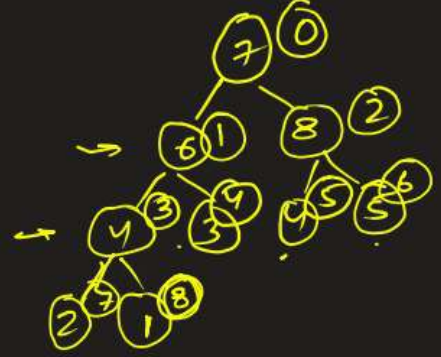
Time = O(n log n) ✓


3 3 4 1 4 5 5 6 8
fix
4 3 3 1 4 5 5 6 8
swap
1 3 3 4 4 5 5 6 8
fix
3 1 3 4 4 5 5 6 8
swap
3 1 3 4 4 5 5 6 8
fix
3 1 3 4 4 5 5 6 8
swap
1 3 3 4 4 5 5 6 8
done ✓


Build heap
① 3 1 5 7 6 4 8 2 4
n log n
Build heap
Insert (3) → 3 1 5 7 6 4 8 2 4
8 6 7 4 5 3 4 1 2
array
less
more
log n
log n-1
n/2
more
O(n)
Heapify

(2)

3	1	5	7	6	4	8	2	4



top to bottom

Build heap → O(n)

Delete & swap → O(n log n)

∴ Heap sort = O(n log n)

$\frac{n}{2}$ $(n-1)/2$

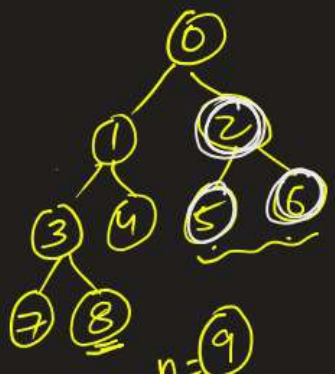
https://leetcode.com/problems/sort-an-array/

0 indexed :-

i → LC → 2i+1

i → RC → 2i+2

i → Parent → $\lfloor \frac{(i-1)}{2} \rfloor$



n = 9

n

n = 8

3	1	8	5	6	2	4	
0	1	2	3	4	5	6	7

8 $\frac{n}{2}-1$

fix_down(_, 3, 7) → l = 3, lc = 7, rc = 8

7 ≤ 7 ✓ 8 ≤ 7 ×

fix_down(_, 2, 7) (p, n) l = 2, lc = 5, rc = 6

l != p ↓

5 ≤ 7 ✓ l = 5

6 ≤ 7 ✓ 6 < 2 ✗

fix_down(_, 5, 7) l = 5, lc = 11, rc = 12

```cpp
class Solution {
public:
    void fix_down(vector<int> &nums, int p, int n) {
        int l = p, lc = 2*p+1, rc = 2*p+2;
        if(lc <= n && nums[l] < nums[lc])
            l = lc;
        if(rc <= n && nums[l] < nums[rc])
            l = rc;
        if(l!=p) {
            swap(nums[l], nums[p]);
            fix_down(nums, l, n);
        }
    }
    vector<int> sortArray(vector<int>& nums) {
        int n = nums.size();
        // Heapify O(n)
        for(int i=n/2-1; i>=0; i--) {
            fix_down(nums, i, n-1);
        }
        // Heap Sort
        for(int r=n-1; r>0; r--) {
            swap(nums[0], nums[r]);
            fix_down(nums, 0, r-1);
        }
        return nums;
    }
};
```

Priority queues

```
priority_queue <int> pq;

pq.push(3);   log(n)
pq.push(7);
pq.push(2);
pq.push(8);

pq.top() → 8        O(1)
pq.pop()            O(logn)
pq.top() → 7
pq.pop()
pq.top() → 3
pq.pop()
pq.top() → 2
pq.pop()
```

(X) `priority_queue < int, vector<int>, greater<int>> pq;`

```
priority_queue <int> pq;

pq.push(3) →      pq.push(-3)
(5) →             (-5)
(7) →             (-7)
(1) →             (-1)

top() → -1 → 1 smallest
```

Number Theory

Array strings LL

```
import heapq

arr = [1, 5, 9, 6, 7, 3]

heapq.heapify(arr)

arr = [_____] → heap

heapq.heappush(arr, 4);

heapq.heappop(arr);
```