Heaps 1
Monday, 10 July 2023 9:10 PM

→ CBT: Complete Binary Tree : Data to be filled level by level L→R

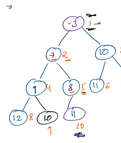
Present 10
Size 16

Constant CBT using arrays
→ Root node at index 1
→ Any node at index : i
Parent: i/2
Child nodes = 2i, 2i+1
1 based indexing

→ Index i
child node: 2i+1, 2i+2
root node (i−1)/2

Property
CBT + Node data <= (both child nodes) : Minheap
CBT + Node data >= (both child nodes) : Maxheap

→ Minheap



→ TC: Height of CBT
↳ $\log N$

```
void Insert(vector<int> arr, int x){
    arr.push_back(x);
    int find = arr.size()-1
    while(find>1 && arr[find/2] > arr[find]){
        swap(arr[find/2], arr[find])
        find = find/2
    }
}
```

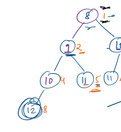
We should initialize arr with a dummy element
arr[0] = INT_MIN

< For max heap

→ Heaps : [Searching / Deleting data at some random position] is waste, not these.

Minheap: Root node minval
Maxheap: Root node maxval } → O(1)

Delete min in minheap
max in maxheap } Delete
1) Swap root node data to last node & delete
2) Node > [Child nodes] or Repeat until we reach leaf
Swap root node with min of its child node




```
void deleteNode(vector<int> arr){
    int n = arr.size()-1;
    arr[1] = arr[n-1];
    arr.pop_back();
    drilldown(arr, 1);
}
```

→ Delete min:
TC: $\log N$

```
void drilldown(vector<int> arr, int i){
    int find = 1, n = arr.size()-1
    while(true)
        min = i;
        if(2i <= n && arr[2i] < arr[i]){
            min = 2i;
        }
        if(2i+1 <= n && arr[2i+1] < arr[min]){
            min = 2i+1;
        }
        if(min == i){ break; }
        swap(arr[i], arr[min])
        i = min;
}
```

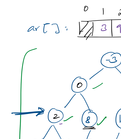
to check if left child present
to check if right child is present

$\log N$

TC =
$\frac{N}{2} \times [1 + N/4 [1] + N/16 [2] + \dots 1 [\log N]]$

[Summation of this series = O(N)]

	0	1	2	3	4	5	6	7	8	9	10
ar[]		3	9	10	6	11	8	-3	2	0	8



$\frac{N}{2}$

```
void Heapify(vector<int> arr){
    int n = arr.size()-1;
    for(int i=n; i>=1; i--){
        drilldown(arr, i);
    }
}
```

→ Heapify
SC: 1
TC: O(N)

Sol: Take elements by elements & insert in new heap
TC: $N \log N$ SC: O(N)
Not an inplace heapify
without extra space

→ Delete min, → Heapify : Inplace sorting algorithm
↳ $O(N + N\log N)$, SC: O(1)
Sorted order

→ Heapify:
1) Apply Heapify on vector
2) At every step of Delete min → Keep it at last of vector → Take our own size variable & update it
Decreasing order

size = 8
Initially
0 1 2 3 4 5 6 7 8
hole
size--

	Heaps	BBST
Insert()	O(log N)	O(log N)
Get Min/Max	O(1)	O(log N)
Deletion/Delete max	O(log N)	O(log N)
		Order

For a given question if require only above operations, Heaps > BBST

→ Use:
priority_queue <int> PQ; } maxheap
priority_queue <int, vector<int>, greater<int>> PQ; } minheap
vector
comparator
Provide comparator

PQ.push(x)
PQ.pop()
PQ.top()
PQ.size()

Java
PriorityQueue<Integer> PQ = new PriorityQueue<>();
PriorityQueue<Integer> PQ = new PriorityQueue<>(Collections.reverseOrder)
- add(x)
- peek()
- remove()
- size()

Q1) Given N sticks combine 2 sticks down until you get a single stick. [Cost to combine 2 sticks is their length of both]

Eg: arr[]: 3 2 7 5

Steps	Cost
2+7 : 9	{3, 9, 5}
9−5 : 14	{3, 14}
3/14 : 17	{17}
9+14+17	
= 40	

Steps
2, 3 : 5 {5, 7, 5}
5, 5 : 10 {7, 10}
7/10 : 17 {17}
32

arr[]: 3 3 4 5 8

Step:
3 3 : 6 {6, 4, 5, 8}
4 5 : 9 {6, 9, 8}
6 8 : 14 {14, 9}
14 9 : 23
52

1) Given list of numbers
→ Pick smallest 2
→ Delete them from list
→ update cost.
→ Insert back combined length of both

minheap pq
Insert all elements to pq
int cost = 0
while(pq.size() >= 2){
    int min1 = pq.top(); → O(1)
    pq.pop(); → O(log N)
    int min2 = pq.top(); → O(1)
    pq.pop(); → O(log N)
    cost += (min1 + min2)
    pq.push(min1+min2) → O(log N)
}

	Heaps	BBST
	O(1)	O(log N)
	O(log N)	O(log N)
	O(1)	O(log N)
	O(log N)	O(log N)
	O(log N)	O(log N)

Faster

Q2) Given N array elements find K smallest elements → Note: using quick select in O(N), difficult plean that.

N = 15, 9, 6, 20, 25, -3, 13, 5, 25

K = 4

Solutions:
1) Insert all elements in minheap pop K elements
TC: $N\log N + K\log N$ SC: O(N)
2) Sort + first K elements
$N\log N + K$

N = 15 9 6 20 25 -3 13 5 19
Insert -3 delete max
Insert 13 delete max fail
Insert 5 Delete max fail

K = 4

25 > (max fail)	
-3 < 20	
13 < 15 (max fail)	
5 < 13 (max fail)	
19 > 9 (max fail) ✗	

K Elements

observations
1) list size is always K
2) new element comes
if(new element < max of list){
    Insert new ele to list
    delete max ele from list
}
else{
    // new element > (max of list)
    It cannot be a answer
}

w. minheap

→ Heap size is at max K.
1) $K\log K + (N-K)(\log K + \log K)$
Insert first K Elements
Extract
Delete max
Insert new element

→ $K\log K + (N-K)(\log K)$
overall = $N\log K$ SC: O(K)