Design and Analysis of Algorithms

Assignment No. 3

Group members:

Harsh Meena -IIT2019005

Asha Jyothi Donga -IIT2019006

Sampada Kathar -IIT2019007

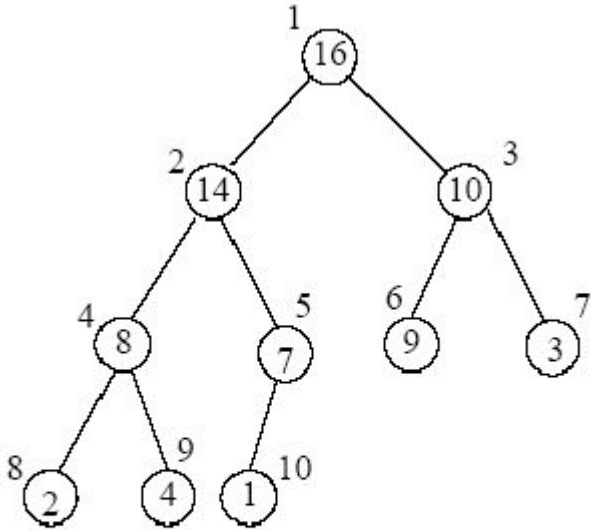
Contents

Find the location of kth largest element in a max heap

Problem:

Design an algorithm to find the location

of kth largest element in a max heap



1	2	3	4	5	6	7	8	9	10
16	14	10	8	7	9	3	2	4	1

(www.cse.in)Taken from reference [4]

Algorithm

1. Create the max heap (priority_queue in stl) pq.
2. Push the root of given max heap array (i.e first element of the Heap array).
3. Repeat the steps for k-1 steps.
    - Pop from the max heap pq (greater element in the max heap pq)
    - Insert the left and right of the popped element if the children exist.
4. The kth greatest element in pq is the kth poped element in the given heap.

To get the left child and right child of the node we need index, so the pair of value and its index is pushed into the priority_queue (max heap).

Pseudo Code :

```
int kthGreatestInMaxHeap(Array heap,Int k)

    priority_queue<Pair(value,index) > pq
    pq.push(Pair(heap[0], 0)

    for( i=0 to k-1)
        poppedIndex= indexOf(pq.top())
        pq.pop()
        leftChild=2*poppedIndex+1;
        rightChild=2*poppedIndex+2;
        if(leftChild exists)
            pq.push(Pair(heap
            [leftChild],leftChild)
        if(rightChild exists)
            pq.push(Pair(heap
            [rightChild],rightChild)

  return indexOf(pq.top())
```

Time Complexity and Space Complexity Analysis

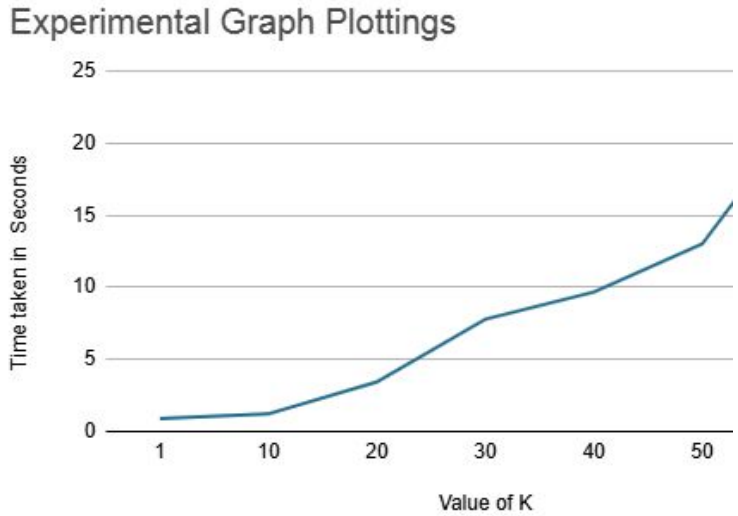
Time Complexity

We need to go through the max heap, k-1 times.

The initial size of the priority queue is one, and it increases by at most one at each of the k – 1 steps. Therefore, there are maximum k elements in the priority queue and the time complexity of the pop and insert operations is O(log.

So total TC of this algorithm becomes : O(k*log(k))

- For best case: k=1 , hence $\Omega(1)$
- Worst Case TC O(k*log(k))



Experimental Graph 1

Space Complexity

In this algorithm we have used priority queue for storing k max elements, hence Space Complexity (Extra Space used) is O(k).

Conclusion

From the analysis of our algorithm we can conclude that the algorithm devised by us is an efficient way of solving our given problem.

It takes a priority queue to get max kth element from given heap which makes it efficient memory wise, and with a worst case time complexity of only O(k*log(k)).

References:

[1]vaibhav29498, 'K-th Greatest Element in a Max-Heap', GeeksforGeeks, 2018. [Online] [Accessed: 1-Feb-2021]

[2]'Heap Data Structures', tutorialspoint. [Online][Accessed: 1-Feb-2021]

[3]GeeksforGeeks, 'Priority Queue — Set 1 (Introduction)', GeeksforGeeks, 2018. [Online][Accessed: 1-Feb-2021]

[4]https://lh3.googleusercontent.com/proxy/mKO_ixx5uUiB93Juy0xjQx7sPvRWmYMZB4EwBpoVn97ZTaED_BMriwLtjjmttoL9_ttq0RbfFRlHeTsMUJEHkWYfqEOjqqK2iB2PR-HdTiNVFGZ0QmX104waXu-HrFEfksEks8OjUjAbrwLy5GADZTU