CS 4820, Spring 2014	**Homework 5, Problem 1**
Name: Piyush Maheshwari	
NetID: pm489	
Collaborators: Eeshan Wagh	

(1) *(10 points)*

Consider the following generalization of the median-finding problem: given a sequence $A = (a_1, \ldots, a_n)$ and integers $k_1 < k_2 < \cdots < k_t$ between 1 and n, output a sequence $B = (b_1, \ldots, b_t)$ such that b_i is the $(k_i)^{\text{th}}$ largest element among the elements in A for all i from 1 to t. You may assume the elements of A are all distinct.

Show that there exists a randomized algorithm for this problem with expected running time $O(n \log t)$.

Since a sorting-based algorithm can solve this problem in time $O(n \log n)$, the interesting case for the algorithm occurs when $\log t$ is significantly smaller than $\log n$. Accordingly, algorithms that solve the problem by sorting the entire sequence $(a_1, \ldots, a_n)$ will not be considered acceptable solutions to this problem.

Solution

We will use select(k,S) algorithm covered in class as a subroutine to this problem. Initially we have n elements in A and t values of k_i.

The algorithm will be recursive. In each call to the algorithm, we will pick the mid point of the array of numbers from $(k_1....k_t) = k_{t/2}$ and then call select(n,$k_{t/2}$) to find the $k_{t/2}$ largest element in the array from $(a_1, ...a_n)$. Suppose that element is a_x. Store this element in $b_{t/2}$. Now if we split the array into two arrays A_l and A_r such that all elements in A_l are smaller than a_x and all elements in A_r are greater then a_x. Since all the k's are in increasing order, we know that all k_i's before $k_{n/2}$ will be in A_l and all k_i's after $k_{n/2}$ will be in A_r. Hence we can recursively solve these smaller sub problems. We will stop our recurrence when there is only a single value of k passed in the recursive method. In that case we will simple call select(k,S) and output the answer.

Proof of Correctness

We can prove the correctness by induction. We claim that find-median(n,t) correctly outputs the sequence $(b_1, b_2, ..b_n)$.

<u>Base case</u>

Consider find-median(n,1). In this method we have only one value of k. Since in this we will only call select(n,k), this will out the correct answer assuming the correctness of select(n,k).

<u>Inductive Step</u>

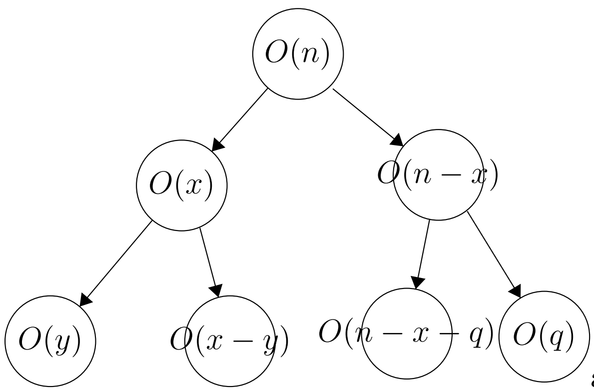
Consider find-median(n,t). We find the mid point among the 't' values of k's and call select(n,$k_{t/2}$). Then we partition the array A into two arrays A_l (some size x) and A_r(size n-x-1) and recursively call find-median(x, (t-1)/2) and find-median(n-x-1, (t-1)/2). Using inductive hypothesis we claim both these recursive calls will correctly compute the array b[]. Using this fact and the fact that select(n,$k_{t/2}$) will correctly compute the largest t/2 element, we will get all the value of b_i. Thus the induction holds.

Running Time

Assume the size of A_l that we compute in any recursive call is x. Then the size of A_r is n - x - 1. From lecture we know that select (n,k) takes O(n) time.

The recurrence can be represented as T(n,t) = T(x,t/2) + T(n-x-1, t/2) + O(n)

If we draw the recursion tree of this recurrence, we can easily see that there is at each level of tree, we do O(n) work. This is because the sum of subproblems that we have at any stage is O(n).



and so on.

We can easily see that the amount of work done on each level is O(n). And since we are dividing the value of t by half at each level, the depth of the tree with be log(t).

Hence the total running time of the algorithm will be O(n logt)