CS/ECE 473 Fall 2020

Homework 3 Problem 3

Ryan Prendergast (ryanp4)
Noah Watson (nwatson3)
Lawson Probasco (lawsonp2)

3.a

Write down a description of randomized quick selection in pseudocode. Show that the expected depth of the recursion of randomized quick selection is $O(logn)$, and that the expected running time is $O(n)$.

Solution: We provide the following pseudocode for randomized quickselect on an array A of size n. It takes the arguments l,r as indices into A: $l \leq n$, $r \leq n$ and $r >= l$. We are looking to find the kth largest element in A. We name the function RQS, and use the standard partition subroutine which, given a left and right subarray, as well as a pivot, partitions elements into the appropriate subarray s.t. elements in the left subarray are less than the pivot, and elements in the right subarray are greater than the pivot.

Algorithm 1 RandomizedQuickSelect(l, r, k)

```
if $l == r$ then
    $return$
end if
$pivot \leftarrow RandInt(l..r)$
partition($l, r, pivot$)
if $pivot == k$ then
    return $A[k]$
else if $k < pivot$ then
    return $RQS(l, pivot - 1, k)$
else
    return $RQS(pivot + 1, r, k)$
end if
```

We first define the recurrence $\bar{T}(n)$ to be the expected value of the amount of work on a problem of size n. We know that the partition function requires $n-1$ comparisons to solve a problem of size n.

$$\bar{T}(n) = n - 1 + max[\bar{T}(k-1), \bar{T}(n-k)]$$

We define a good pivot to be in the middle 50% of elements and a bad one to be in the lower 25 or upper 25. In the event of a good pivot,

$$\bar{T}(n) \leq n - 1 + \bar{T}(3/4 * n)$$

and in the event of a bad pivot

$$\bar{T}(n) \leq n - 1 + \bar{T}(n)$$

Because the pivot is selected uniformly randomly, there is equal probability of 1/2 of both. So,

$$\bar{T}(n) \leq 2n - 2 + .5 * \bar{T}(n) + .5 * \bar{T}(3/4 * n)$$
$$\bar{T}(n) \leq 4n - 4 + \bar{T}(3/4 * n)$$

By the master theorem, we know that the right side recurrence runs in $O(n)$ time, and this serves as an upper bound.

Next we define the recurrence $\bar{D}(n)$ to be the expected value of the depth of the recursion tree of a problem of size n. We know that each call to the function adds one to the recursion tree depth.

$$\bar{D}(n) = 1 + max[\bar{D}(k-1), \bar{D}(n-k)]$$

We will use the same definition of a good and bad pivot as previously: define a good pivot to be in the middle 50% of elements and a bad one to be in the lower 25 or upper 25. In the event of a good pivot,

$$\bar{D}(n) \leq 1 + \bar{D}(3/4 * n)$$

and in the event of a bad pivot

$$\bar{D}(n) \leq 1 + \bar{D}(n)$$

Because the pivot is selected uniformly randomly, there is equal probability of 1/2 of both. So,

$$\bar{D}(n) \leq 2 + .5 * \bar{D}(n) + .5 * \bar{D}(3/4 * n)$$
$$\bar{D}(n) \leq 4 + \bar{D}(3/4 * n)$$

By the master theorem, we know that the right side recurrence runs in $O(logn)$ time, and this serves as an upper bound.

■

CS/ECE 473 Fall 2020	Ryan Prendergast (ryanp4)
Homework 3 Problem 3	Noah Watson (nwatson3)
	Lawson Probasco (lawsonp2)

3.b

Let $A1, A2, ..., Ah$ be h sorted arrays where A_i has n_i elements, and let $n = \sum_{i=i}^{h} n_i$. Assume that the arrays have distinct elements. Describe a randomized algorithm that given integer k finds the k-th smallest element in the combined set of arrays in $O(hlog^2 n)$ expected time.

Solution: We define the following randomized routine:

First we define the array $B[1...h]$ to contain two values at each index that define the boundary indices. Initialize the left boundary $B[i].l \leftarrow 1$ the right boundary $B[i].r \leftarrow n_i$ for all $1 \leq i \leq h$. We also store the local pivot index and initialize it to -1. $B[i].p \leftarrow -1$

Pick a random element in the array of arrays. Check that the random element is within the boundaries of its subarray A_i, and keep generating random elements until we get one that is in the correct bounds. Its value is the pivot. We define $sum_p \leftarrow 0$ to be the sum of the pivot indeces.

Then, for each of the h subarrays A_i, we take only the area within the boundaries: $C = A_i[B[i].l]...A_i[B[i].r]$. We find the index where the pivot would divide the sliced array C into two subarrays such that everything to the left of the index is less than the pivot and everything to the right is greater than (nothing is equal as the problem specifies all elements are unique). We save this local pivot index at $B[i].p$ Also, add this value to the sum: $sum_p + = B[i].p$. Since the subarrays are sorted, each subarray i requires $O(logn_i) \leq O(logn)$ time.

After we have done this partitioning for each class, we compare the sum of the local pivot indeces with the pivot itself. If $sum_p == pivot$, that means there the kth smallest element is at the sum_p index. We find this by finding the last nonempty subarray A_i s.t $B[i].l! = B[i].r$ and return $A_i[B[i].p]$. Requires $O(h)$ time.

If $sum_p < k$, the element we looking for is greater than all to the left of the pivots. We iterate through each subarray A_i and set $B[i].l \leftarrow B[i].p + 1$. This requires $O(h)$ time. Then we recurse, restarting this routine with the updated boundaries in the subarrays.

If $sum_p > k$, the element we looking for is less than all to the right of the pivots. We iterate through each subarray A_i and set $B[i].r \leftarrow B[i].p - 1$. This requires $O(h)$ time. Then we recurse, restarting this routine with the updated boundaries in the subarrays.

Analysis We see that each time we run this routine, we have a loop of size h, and each iteration requires logn time. Adding the other computation, we see that each time we run the above routine it requires $O(hlogn + 2h) = O(hlogn)$ work.

We define the following recurrence for the expected runtime on a problem of size n: k is some value from 1..n that is the random pivot.

$$\bar{T}(n) = hlogn + max[\bar{T}(k-1), \bar{T}(n-k)]$$

We define a good pivot to be in the middle 50% of elements and a bad one to be in the lower 25 or upper 75. In the event of a good pivot,

$$\bar{T}(n) \leq hlogn + \bar{T}(3/4 * n)$$

and in the event of a bad pivot

$$\bar{T}(n) \leq hlogn + \bar{T}(n)$$

Because the pivot is selected uniformly randomly, there is equal probability of 1/2 of both. So,

$$\begin{aligned}\bar{T}(n) &\leq 2hlogn + .5 * \bar{T}(n) + .5 * \bar{T}(3/4 * n)\\ \bar{T}(n) &\leq 4hlogn + \bar{T}(3/4 * n)\end{aligned}$$

By the master theorem, we find that this reccurence runs in $O(hlog^2n)$ time.

■