CSE331: Introduction to Algorithms
Notes on Lecture 13: The Selection Problem

Antoine Vigneron

October 18, 2020

Abstract

We give an analysis of randomized QUICKSORT by extending the analysis of the randomized selection algorithm given in Lecture 13.

Notation and terminology. We assume that we are sorting an input array $A[1 \ldots n]$ recording the keys $\{a_1, \ldots, a_n\}$ in no particular order. While running QUICKSORT, we recurse on some subarrays of the form $A[p, r]$. We denote by S the set of all such subarrays, so the elements of S are of the form $I = A[p, r]$ for some integers $p, r \in \{1, \ldots, n\}$. (This set S does not contain all subarrays $A[p, r]$, $1 \leqslant p \leqslant r \leqslant n$, but only those considered while running QUICKSORT once.)

During the course of the algorithm, we say that a_i is in *phase* j if the current subarray $I \in S$ that contains a_i has size $|I|$ satisfying

$$n\left(\frac{3}{4}\right)^{j+1} < |I| \leqslant n\left(\frac{3}{4}\right)^{j}.$$

When randomized QUICKSORT picks a pivot x in subarray I, we say that the pivot is *central* if there are at least $|I|/4$ elements smaller than x in I and there are at least $|I|/4$ greater elements. It occurs with probability $1/2$, as we observed in Lecture 13.

We denote by Y_i the number of different subarrays $I \in S$ that contain a_i during the course of the algorithm. In other words, we have $Y_i = |\{I \in S \mid a_i \in I\}|$. (See Figure 1.)

Analysis. The running time of QUICKSORT is dominated by the execution of the PARTITION procedure on the subarrays being partitioned, that is, the subarrays in S. The time taken by PARTITION on a subarray I is linear, so it is $\Theta(|I|)$. Therefore, the running time $T(n)$ satisfies

$$T(n) = \Theta(Y) \text{ where } Y = \sum_{I \in S} |I|. \tag{1}$$

We now use *double counting* to obtain a different expression of Y. By definition, $Y = \sum_{I \in S} |I|$. Each key a_i contributes Y_i times to this sum, so we have $Y = \sum_{i=1}^{n} Y_i$. (See Figure 1.) It follows by linearity of expectation that

$$E[Y] = \sum_{i=1}^{n} E[Y_i]. \tag{2}$$

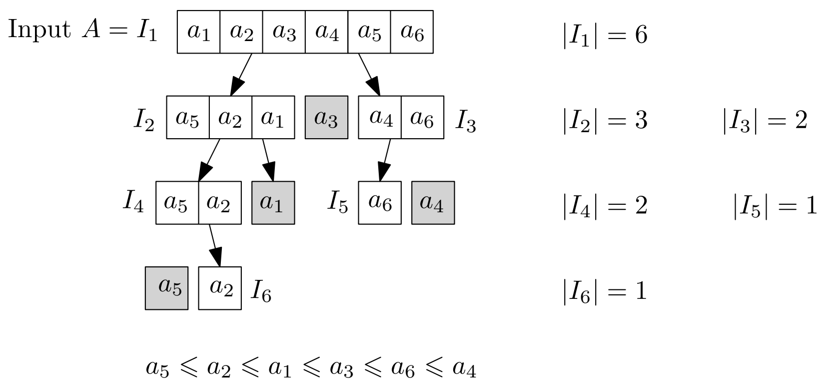

Figure 1: Example of execution of QUICKSORT. The pivots are shaded. The set of subarrays generated during the course of the algorithm is $S = \{I_1, I_2, \ldots, I_6\}$. Element a_2 is contained in subarrays I_1, I_2, I_4 and I_6, so we have $Y_2 = 4$. Elements a_5 and a_6 are contained in 3 subarrays, so we have $Y_5 = Y_6 = 3$. Elements a_1 and a_4 contained in 2 subarrays, so we have $Y_1 = Y_4 = 2$. Element a_3 is only contained in I_1, so $Y_3 = 1$. The double counting argument counts the total size of all the subarrays in two different ways, which in this case gives $\sum_i Y_i = \sum_j |I_j| = 15$.

It remains to bound $E[Y_i]$. To this end, we argue that during the course of the algorithm, the expected number of different arrays in phase j that contain a_i is at most two. Indeed, suppose that a_i is contained in an array I in phase j. Then with probability $1/2$, the random pivot chosen in I is central, and thus the next array containing a_i will have size at most $n\left(\frac{3}{4}\right)^{j+1}$, and a_i moves to phase larger than j. By the waiting time bound, it means that a_i is taken from phase j to phase larger than j after an expected number of at most two random partitions of the interval containing it.

As the the array containing a_i in phase j has size at most $n(3/4)^j$, there are at most $\log_{4/3} n$ phases before the size of this array reaches 1, so there are at most $\log_{4/3} n$ different phases. Since the expected number of arrays that contain a_i in phase j is at most 2, it follows that $E[Y_i] \leqslant 2\log_{4/3} n$. By Equation 2, it implies that $E[Y] \leqslant 2n\log_{4/3} n = O(n \log n)$, and thus by Equation 1

$$E[T(n)] = O(n \log n).$$

We have just proved that the expected running time of randomized QUICKSORT, over all possible random choices that it makes, is $O(n \log n)$.