A Probabilistic Analysis of Quicksort

You are assumed to be familiar with Quicksort. In each iteration this sorting algorithm chooses a pivot and then, by performing $n-1$ comparisions with the pivot, splits the remainder of the input into those elements less than the pivot and those elements greater than the pivot. It then recurses on two subproblems, sorting the elements less than the pivot and those greater than the pivot separately.

In the worst case, the pivot is always the largest of the elements to be sorted, the size of the problem decreases by one in each iteration, and there are $\binom{n}{2}$ comparisions. In the best case, the algorithm always split the input into two sets whose size differs by at most one, and the algorithm takes fewer than $nlogn$ comparisions. As you will be asked to show in the assignment, actually in the best case the algorithm takes at least $(1+o(1))n\ log\ n$ comparisions. We are interested in the variant of quicksort in which each pivot is chosen uniformly at random. I.e. if in each subproblem, each element in the list currently being sorted is equally likely to be the pivot. As discussed above, the random variable X which counts the number of comparisions made by this algorithm can take values as as large as $\binom{n}{2}$ and smaller than $nlog\ n$. We are interested in computing $E(X)$, the expected number of comparisions taken. .

Our approach exploits the fact that X is actually equal to the sum of $\binom{n}{2}$ simple random variables. Spefically, if for $1 \leq i < j \leq n$, we let $X_{i,j}$ be the number of comparisions the algorithm makes between the ith smallest element z_i and jth smallest element z_j of the input then

$$X = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{i,j}.$$

Now, the expectation of the sum is the sum of the expectations, so

$$E(X) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E(X_{i,j}).$$

We note that we compare two elements of the list to be sorted in an iteration, precisely if one of them is the pivot. In this case, the pivot is not compared in any future iteration and the two elements are compared exactly once. Thus each $X_{i,j}$ is either 0 or 1 and $E(X_{i,j})$ is the probability that z_i and z_j are compared.

We note further that while the pivot is neither equal to nor lies between z_i and z_j, all of $z_i, z_{i+1}, ..., z_j$ remain in the same subproblem. Thus, if no z_k with $i < k < j$ is chosen as a pivot before either of z_i or z_j then they will be compared. On the other hand if some z_k with $i < k < j$ is chosen as a pivot before either of z_i or z_j then they will not be compared.

In the first iteration, we can choose the pivot uniformly as follows. We first expose whether the pivot is one of $z_i, z_{i+1}, ..., z_j$ or not (it is with probability $\frac{j-i+1}{n}$). If so, we choose it to be z_k for each $i \leq k \leq j$ with probability $\frac{1}{j-i+1}$. Otherwise we chose it to be each z_k with $k < i$ or $k > j$ with probablitiy $\frac{1}{n-j+i-1}$. In the latter case, we use the same procedure in the subproblem containing z_i and z_k and the elements between them. In the analysis, n is replaced by n' where n' is the number of elements to be sorted in this subproblem. We continue in this manner until we use z_k as a pivot for some $i \leq k \leq j$. We see that the probability that $k = i$ or $k = j$ is exactly $\frac{2}{j-i+1}$ regardless of the size of the subproblem and so the probability we compare z_i and z_j is exactly $\frac{2}{j-i+1}$.

Thus,

$$\begin{aligned} E(X) &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \\ &= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} \\ &\leq \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{2}{k+1} \\ &\leq \sum_{i=1}^{n} \sum_{k=1}^{n} \frac{2}{k} \\ &= n \sum_{k=1}^{n} \frac{2}{k} \end{aligned}$$

Now , it is well known that $\sum_{k=1}^{n} \frac{1}{k}$is $log\ n + O(1)$ we see that $E(X) \leq 2nlog\ n + O(n)$. This means that on average quicksort performs no more than $2 + O(1)$ times as many comparisions as it performs in the best case.

Randomized Algorithms for Selection

The ith order statistic of a set of distinct is its ith smallest element. Thus the minimum is the first order statistic and the maximum of a set of n elements is the nth order statistic. if n is odd then the *median* of a set of n elements is its $\frac{n+1}{2}$ order statistic while if n is even then a set of N elements has two medians. Its $\frac{n}{2}$ order statistic is the *lower median* and its $\frac{n+2}{2}$ order statistic is the *higher median*. The *rank* of the ith order statistic is i.

The selection problem takes as input a set A of (distinct) elements and an integer r between 1 and $n = |A|$. Its output is the rth order statistic(r here stands for rank).

QuickSelection

One way of solving selection is to sort the input set using Quicksort and to then simply traverse the ordered set to find the desired element. However, this is much too much work, as in each iteration we really need only recurse on one of the two subproblems not both. We now describe an algorithm QUICKSELECT which chooses pivots and splits the problem into two parts in each iteration just like QUICKSORT, but only recurses on at most one of the subproblems. Specifically if the pivot is the jth order statistic then if $j = r$ we simply return the pivot, if $j > r$ we simply return the rth order statistic in the set of elements which are less than the pivot, and if $j < r$ we simply return the $r - jth$ order statistic in the set of elements which are greater than the pivot.

Definining X and $X_{i,j}$ as in the analysis of quicksort, we once again have that

$$E(X) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} E(X_{i,j})$$

and that $X_{i,j}$ is the probability $p_{i,j}$ that z_i and z_j are compared.

But the value of $p_{i,j}$ differs from that calculated in the quicksort analysis. Specifically, if the element we are looking for lies on one side of the pivot, and z_i and z_j both lie on the other side, then neither z_i nor z_j remains in the active problem and so they will not be compared. Thus, we need to consider the first time that there is a pivot lying between the minimum $min_{i,j}$ of z_i, z_r, z_j and the maximum $max_{i,j}$ of these 3 values. The probability $p_{i,j}$ is precisely the probability that when this occurs the pivot is one of z_i or z_j. It

follows that $p_{i,j}$ is always at most $\frac{2}{max(|r-i|,|r-j|)+1}$. So in calculating our sum, we reindex using $k_{i,j} = max(|r-i|, |r-j|)$ and note that $p_{i,j} \leq \frac{2}{k_{i,j}+1} < \frac{2}{k_{i,j}}$.

Now, we know every $k_{i,j}$ lies between 1 and n. Furthermore, if $k_{i,j} = k$ then one of i or j is $r-k$ or $r+k$ and the other lies between $r-k$ and $r+k$. Thus there are at most $4k$ choices for such a pair. It follows that:

$$E(X) \leq \sum_{k=1}^{n} 4k\frac{2}{k} \leq 8n$$

.

The Two Pivot Algorithm

The expected number of comparisions made by our next algorithm for randomized selection is $2n+o(n)$. It employs a subroutine to finds two pivots p_{low} and p_{high} with $p_{low} < p_{high}$ using $o(n)$ comparisions. It then use comparisions with the pivots to split $A - p_{low} - p_{high}$ into three sets: $Low = \{z|z < p_{low}\}$, $High = \{z|z > p_{high}\}$, and $Middle = \{z|p_{low} < z < p + high\}$. This can be done with $2n-4$ comparisions, by comparing each non-pivot with each pivot. Now, if z_r is one of the pivots we return it. Otherwise we proceed as follows. If r is less than the rank r_{low} of p_{low} We sort Low, and return its rth order statistic. If r exceeds the rank r_{high} of p_{high} we sort High and return its $(r - r_{high})th$ order statistic. If $r_{low} < r < r_{high}$ we sort Middle and return its $(r - r_{low})th$ order statistic. We claim that the probability p_{fail} of the event that our subroutine does not return pivots such that the element of $\{Low, Middle, High\}$ we need to sort has size at most $\frac{n}{log^2\ n}$ is $o(\frac{1}{n})$.

Now, we can sort a set of m elements in $O(mlog\ m)$ steps. The set we sort has at most n elements. So, the total number of steps we need is $o(n) + 2n - 4 + O((1-p_{fail})\frac{n}{log^2\ n}log(\frac{n}{log^2\ n})) + O(p_{fail}nlog\ n)$. If our claim is true, this is $2n + o(n)$.

To complete the description and analysis of our algorithm, it remains to describe a subroutine which chooses the two pivots in $o(n)$ time for which the claim holds. We do so now. The subroutine chooses a subset A' of $\lceil\sqrt{n}\rceil$ elements of uniformily at random (by repeatedly choosing a uniform element from the unchosen set to add to the chosen set until it has the right size), and sorts A'. It sets i_{low} to be $\lceil\frac{r}{\sqrt{n}} - n^{\frac{3}{8}}\rceil$ and i_{high} to be $\lceil\frac{r}{\sqrt{n}} + n^{\frac{3}{8}}\rceil$.

Provided r is at least $2n^{\frac{7}{8}}$ and at most $n - 2n^{\frac{7}{8}}$ It sets p_{low} to be the ith order statistic of A' and p_{high} to be the $i_{high}th$ order statistic of A'. As will be

discussed in class, well known results then yield that the probability that the event $r_{low} < r < r_{high} < r_{low} + \frac{n}{log^2\ n}$ fails is $o(\frac{1}{n})$. Provided this event holds, we search Middle which has at most $\frac{n}{log^2\ n}$ elements. So the claim holds in this case.

If r is less than $2n^{\frac{7}{8}}$ It sets p_{low} and p_{high} both to be the to be the $i_{high}th$ order statistics. As will be discussed in class, similar well known results then yield that the probability that the event $r < r_{low} < \frac{n}{log^2\ n}$ fails is $o(\frac{1}{n})$. Provided this event holds, we search Low which has at most $\frac{n}{log^2\ n}$ elements. So the claim holds in this case.

If r is more than $n - 2n^{\frac{7}{8}}$ It sets p_{low} and p_{high} both to be the to be the $i_{low}th$ order statistics. A symmetrical argument yields that the probability that the event $r > r_{high} > n - \frac{n}{log^2\ n}$ fails is $o(\frac{1}{n})$. Provided this event holds, we search High which has at most $\frac{n}{log^2\ n}$ elements. So the claim holds in this case.