Analysis of Algorithms I: Randomized Quicksort

Xi Chen

Columbia University

As discussed in the last class, we use randomization to improve the performance of Quicksort against those worst-case instances. We use the following procedure Randomized-Partition to replace Partition. It randomly picks one element in the sequence, swaps it with the first element, and then calls Partition.

Randomized-Partition (A, p, q), which works on $A[p \dots q]$

1. $r = \text{Random}(p, q)$ (pick an integer between p and q uniformly at random) and exchange $A[p]$ with $A[r]$
2. return Partition (A, p, q)

So essentially we randomly pick an element in the sequence and use it as a pivot to partition.

Randomized-Quicksort(A, p, q), which sorts $A[p \ldots q]$

```
1  If $p < q$
2      $s =$ Randomized-Partition$(A, p, q)$
3      Randomized-Quicksort$(A, p, s - 1)$
4      Randomized-Quicksort$(A, s + 1, q)$
```

Use induction to prove its correctness. Given any input instance A, its running time $t(A)$ is now a random variable that depends on the pivots it picks randomly.

Theorem (Main)

Given any input sequence A of length n, Randomized-Quicksort $(A[1 \ldots n])$ has expected running time $O(n \lg n)$.

This of course implies that the worst-case expected running time (defined in the note of the last class) of Randomized-Quicksort is $O(n \lg n)$. Actually, we will see from the analysis that the order of the elements in A at the beginning (almost) does not affect the running time of Randomized-Quicksort at all.

We start with some intuition of why its expected running time is $O(n \lg n)$. This is by no means a proof. So we know Quicksort has worst-case running time $\Omega(n^2)$ because if the input is already sorted, every pivot we pick is the smallest / largest of the sequence, which leads to highly unbalanced subsequences and $\Omega(n^2)$ running time. Now what if we are very (or extremely) lucky and get the median of the sequence as a pivot every time. (Btw, what if we just "find the median of the sequence" and use it as a pivot? But ... how much time it takes to find the median? Will see next class.)

Best case: Always use the median to partition. Recurrence:

$$T(n) = 2T(n/2) + \Theta(n)$$

From now on we usually assume $T(1) = \Theta(1)$ by default. Using Case 2 of the Master Theorem applies, we get $T(n) = \Theta(n \lg n)$.

Now what if we are not that lucky but always partition the sequence into two subsequences of ratio 9 : 1 or 1 : 9. The recurrence becomes:

$$T(n) = T(n/10) + T(9n/10) + \Theta(n).$$

How to solve it? Use recursion tree to make a good guess, and then prove it formally using the substitution method.

The first level: cn
The second level sums to: $c(n/10) + c(9n/10) = cn$
The third level sums to:

$$c(n/100) + c(9n/100) + c(9n/100) + c(81n/100) = cn$$

How many levels are there? The longest path (the rightmost one) is of length $\log_{10/9} n = \Theta(\lg n)$. From the recursion tree, a good guess would be $T(n) = \Theta(n \lg n)$. Use the substitution method to formally prove it.

Note that recursion tree alone sometimes may not serve as a good proof and sometimes may lead to wrong conclusions. In particular, this tree is not a complete tree (the leftmost path is much shorter, though still of the order $\log_{10} n = \Theta(\lg n)$). Many of the levels close to the bottom do not sum to cn. This is very different from the recursion tree for

$$T(n) = aT(n/b) + f(n)$$

which is a complete tree and all nodes on the same level have the same number. So it is always a good idea to verify your guess from irregular recursion trees using the substitution method.

More generally, it can be shown that if every pivot we use satisfies

$$\text{the longer subsequence} \leq \text{the shorter sequence} \times 9$$

then the running time is $O(n \lg n)$ (though the recurrence for this is kind of scary). Also 9 can be replaced by any fixed constant, like 99. It is also easy to calculate that the probability of picking a pivot satisfying the property above is $4/5$ (or $49/50$ if replacing 9 by 99, or even closer to 1 if we use a larger constant).

To summarize, if every pivot results in two subsequences of ratio bounded by a constant, then Quicksort takes time $O(n \lg n)$. On the other hand, every time Randomized Quicksort picks a pivot, it satisfies this condition with a probability close to 1. But ... this is still far from a formal proof (though the idea here may lead you to Exercise ?, a proof different from the one we present next, where the difficulty is to solve a hairy recurrence).

Now we start the proof. Randomized-Quicksort spends most of the time in those calls to Randomized-Paritition. (Check that it spends only constant many steps between any two consecutive calls to Randomized-Partition). So to bound its running time, it suffices to bound the time it spends in Randomized-Partition.

Every time we call Randomized-Partition (A, p, q), it takes time

$$\Theta(1) + \Theta(q - p)$$

where $\Theta(1)$ accounts for the time needed to choose a random index and to swap $A[r]$ and $A[p]$. Or equivalently, it takes time

$$\Theta(1) + \Theta(\text{number of comparisons made between two elements})$$

We need the following lemma to suppress the $\Theta(1)$ above.

Lemma

The number of calls to Randomized-Partition is at most n.

Proof.

First, every element is used as a pivot at most once. This is because every time an element is used as a pivot, it is at the right position by the end of this partition operation, and is never examined again, not to mention being used as a pivot again. (Check an example, and prove it using induction.)

On the other hand, every time we call Randomized-Partition, it picks an element as a pivot. Therefore, the number of calls to Randomized-Partition is the same as the number of pivots picked, and is no more than n. □

The time we spend in Randomized-Quicksort:

1. Call 1: $c_1 + c_2 \cdot \#$ of comparisons made in Call 1
2. Call 2: $c_1 + c_2 \cdot \#$ of comparisons made in Call 2
3. $\cdots$
4. Call k: $c_1 + c_2 \cdot \#$ of comparisons made in Call k

where by the previous lemma, $k \le n$ no matter what pivots we use. Therefore, the total time is

$$\begin{aligned} & c_1 \cdot k + c_2 \cdot \text{total number of comparisons} \\ = \; & O(n) + \Theta(\text{total number of comparisons}) \end{aligned}$$

Let X denote the total number of comparisons made in all the calls to Randomized-Quicksort, which is a random variable because it depends on the pivots we randomly pick. From the equation on the last slide, the theorem follows immediately from

$$E[X] = O(n \lg n)$$

However, understanding $\Pr[X = i]$ is very difficult. Instead we use indicator random variables and linearity of expectations.

For simplicity of the proof, we assume all elements in the input are distinct. (The textbook uses a slightly different implementation of Partition, and the proof does not require this assumption though the idea is the same.) With this assumption, let $z_1 < z_2 < \cdots < z_n$ be a permutation of the input sequence, and let $Z_{i,j}$ denote

$$Z_{i,j} = \left\{ z_i, z_{i+1}, \ldots, z_j \right\}, \quad \text{for all } i, j : 1 \leq i < j \leq n$$

Note that $|Z_{i,j}| = j - i + 1$.

Indicator random variables: For all $i, j : 1 \leq i < j \leq n$, $X_{i,j}$ is an indicator variable such that $X_{i,j} = 1$ if we compare z_i and z_j at least once in the execution and $X_{i,j} = 0$ otherwise. Question: Is the following equation correct?

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

In general, it is not correct because $X_{i,j}$ is only an indicator variable. No matter how many times we compare z_i and z_j, whether it is 1 or 2 or 3 or more, it always contributes 1 to the right hand side. But ... if it can shown that we never compare z_i and z_j more than once then this equation always holds.

Lemma

For all $i, j : 1 \leq i < j \leq n$, z_i and z_j is compared at most once.

Proof.

First of all, we compare z_i and z_j if and only if one of them is the pivot of the current partition operation. Without loss of generality, say z_i is the pivot. Then as mentioned earlier, after z_i is done with the role of being a pivot, it is in the right position and never examined again and thus, we will never compare it with z_j again in the subsequent time. $\square$

Now we can use the linearity of expectations:

$$E[X] = E\left[\sum_{i=1}^{n}\sum_{j=i+1}^{n} X_{i,j}\right] = \sum_{i=1}^{n}\sum_{j=i+1}^{n} E[X_{i,j}]$$

From $E[X_{i,j}] = 0 \cdot \Pr(X_{i,j} = 0) + 1 \cdot \Pr(X_{i,j} = 1) = \Pr(X_{i,j} = 1)$, the key is to calculate the probability that we compare z_i and z_j.

We start with a few observations: First, as mentioned earlier, we compare z_i and z_j only when one of them is the current pivot used to partition. Second, until the moment we pick a pivot from $Z_{i,j}$, all elements in $Z_{i,j}$ lie in the same subproblem. This is because every time we pick a pivot outside of $Z_{i,j}$, say x, it holds that either all elements in $Z_{i,j}$ are greater than x or all elements in $Z_{i,j}$ are smaller than x. As a result, when using x to partition, all elements in $Z_{i,j}$ go to the same subproblem.

When we pick the first pivot from $Z_{i,j}$ (Is it possible that Randomized-Quicksort never picks any pivot from $Z_{i,j}$?), say $x \in Z_{i,j}$, there are two cases: If $x = z_i$ or z_j, say z_i, then we need to compare $x = z_i$ to every element in the current subsequence because it is the pivot, including z_j. (Why z_j must be in the current subsequence? This is because $x = z_i$ is the first pivot we pick from $Z_{i,j}$ and thus, from the second observation, elements in $Z_{i,j}$ are still in the same subproblem at this moment.)

If the first pivot x from $Z_{i,j}$ is neither z_i nor z_j, then we don't get to compare z_i and z_j in this partition operation (instead we compare x with z_i, and x with z_j). After partitioning using x, z_i and z_j will never be compared again because $z_i < x$ and $x < z_j$ so they will be separated into two different subproblems.

To summarize, we compare z_i with z_j if and only if the first pivot from $Z_{i,j}$ is z_i or z_j. Thus,

$$\begin{aligned} & \Pr[X_{i,j} = 1] \\ = \; & \Pr(\text{first pivot from } Z_{i,j} \text{ is } z_i) + \Pr(\text{first pivot from } Z_{i,j} \text{ is } z_j) \end{aligned}$$

But all elements in $Z_{i,j}$ are equally likely to be the first pivot from $Z_{i,j}$, so both probabilities above are equal to $1/(j-i+1)$. The next experiment may help understand why this is the case.

Consider the following game with n balls inside a bin. $(j - i + 1)$ of the balls are red and the rest are black. One red ball is marked with i, one is marked with $i + 1, \ldots$, and one is marked with j. Then

1. Randomly pick a ball from the bin
2. If the ball we get is black, open the bin and take out a few more black balls; then repeat line 1
3. If the ball we get is red, stop.

Now what is the probability we end up with the red ball marked with i? the one marked with $i+1$? ... the one marked with j? They are all equal because being marked with $i, i+1, \ldots, j$ does not make any of them special, and does not yield any advantage over other red balls. Compare this game with Randomized Quicksort. Conclude that the probability that we end up with the red ball marked with $k : i \leq k \leq j$ is the same as the probability that the first pivot from $Z_{i,j}$ is z_k.

Finally, plugging in $\Pr[X_{i,j} = 1] = 2/(j - i + 1)$, we get

$$\begin{aligned} E[X] &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} 2/(j - i + 1) \\ &= 2 \cdot \sum_{i=1}^{n} \big(1/2 + 1/3 + \cdots + 1/(n - i + 1)\big) \\ &< 2 \cdot \sum_{i=1}^{n} \big(1 + 1/2 + 1/3 + \cdots + 1/n\big) \end{aligned}$$

The sum $1 + 1/2 + \cdots + 1/n$ is the harmonic series (check the appendix of the textbook) and $\ln n$ is a very good approximation:

$$1 + 1/2 + \cdots + 1/n = \ln n + O(1)$$

As a result, we have

$$E[X] = 2 \cdot n \cdot O(\ln n) = O(n \ln n) = O(n \lg n)$$

and ... this finishes the whole proof.