QuickSort Calculations
CS 417

Let $Q(n)$ be the run-time of the quicksort algorithm when the pivot is chosen randomly. Lecture5a.pdf has a clever solution using the recursion

$$Q(n) = \frac{2}{n}\sum_{k=1}^{n-1} Q(k) + n. \tag{1}$$

Here we will do a different solution using this recursion (due to me) and we will do a completely different solution using ideas from probability (based on notes written by Andreas Klappenecker).

1 My solution

Let $f(n) = nQ(n)$.

Then, using (1)

$$f(n) = 2\sum_{k=1}^{n-1} Q(k) + n^2 = 2\sum_{k=1}^{n-1} \frac{f(k)}{k} + n^2.$$

Now, let $F(x)$ be the generating function of $f(n)$, that is,

$$F(x) = \sum_{n=1}^{\infty} f(n)x^n. \tag{2}$$

Using the recursion and interchanging two sums, we get

$$\begin{aligned} F(x) &= \sum_{n=1}^{\infty} 2\sum_{k=1}^{n-1} \frac{f(k)}{k}x^n + \sum_{n=1}^{\infty} n^2x^n \\ &= 2\sum_{k=1}^{\infty} \frac{f(k)}{k} \sum_{n=k+1}^{\infty} x^n + \sum_{n=1}^{\infty} n^2x^n. \end{aligned}$$

Using that

$$\sum_{n=k+1}^{\infty} x^n = x^{k+1}\sum_{n=1}^{\infty} x^n = \frac{x^{k+1}}{1-x} = \frac{x}{1-x}x^k,$$

and

$$\sum_{n=1}^{\infty} n^2x^n = \frac{x^2+x}{(1-x)^3},$$

we get

$$F(x) = \frac{2x}{1-x}\sum_{k=1}^{\infty} \frac{f(k)}{k}x^k + \frac{x^2+x}{(1-x)^3}. \tag{3}$$

Let

$$S(x) = \sum_{k=1}^{\infty} \frac{f(k)}{k}x^k.$$

Then

$$S'(x) = \sum_{k=1}^{\infty} f(k)x^{k-1} = \frac{1}{x}\sum_{k=1}^{\infty} f(k)x^k = \frac{F(x)}{x}. \tag{4}$$

We also have, from (3)

$$S(x) = \frac{1-x}{2x}F(x) - \frac{x+1}{2(1-x)^2}. \tag{5}$$

Let's now take the derivative of (3)

$$F'(x) = \frac{2}{(1-x)^2}S(x) + \frac{2x}{1-x}S'(x) + \frac{1+4x+x^2}{(1-x)^4}. \tag{6}$$

Replacing $S(x)$ using (5) and using $S'(x) = F(x)/x$, we get

$$\begin{aligned} F'(x) &= \frac{F(x)}{(1-x)x} - \frac{x+1}{(1-x)^4} + \frac{2}{1-x}F(x) + \frac{1+4x+x^2}{(1-x)^4} \\ &= \frac{2x+1}{x(1-x)}F(x) + \frac{x^2+3x}{(1-x)^4}. \end{aligned}$$

We get a linear first order differential equation, which we can solve to get

$$F(x) = \frac{Cx - x^2 - 4x\log(1-x)}{(1-x)^3},$$

for some constant C. Note that $F'(0) = f(1) = 1$. But by deriving $F(x)$ we get

$$F'(x) = \frac{C + 2Cx + (2-x)x - 4(1+2x)\log(1-x)}{(1-x)^4},$$

so $F'(0) = C$. Therefore $C = 1$.

From this we conclude that

$$F(x) = \frac{x - x^2 - 4x\log(1-x)}{(1-x)^3}. \tag{7}$$

Now, using

$$\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n$$

we can deduce (by derivating twice) that

$$\frac{2}{(1-x)^3} = \sum_{n=0}^{\infty}(n+2)(n+1)x^n.$$

We can then deduce that

$$\frac{x}{(1-x)^3} = \sum_{n=0}^{\infty} \frac{(n+1)n}{2}x^n, \tag{8}$$

and

$$\frac{-x^2}{(1-x)^3} = \sum_{n=1}^{\infty} \frac{-n(n-1)}{2}x^n \tag{9}$$

The last piece is harder to estimate. To do it, we will need the Taylor expansion of $\log(1-x)$. Namely,

$$\log(1-x) = -\sum_{k=1}^{\infty} \frac{x^k}{k}.$$

Therefore, by using (8) and the Taylor series expansion, we get

$$\frac{-4x\log(1-x)}{(1-x)^3} = -\sum_{m=0}^{\infty} 2(m+1)mx^m \left(-\sum_{k=1}^{\infty}\frac{x^k}{k}\right) = \sum_{m=0}^{\infty}\sum_{k=1}^{\infty}\frac{2m^2+2m}{k}x^{m+k}.$$

Now let $n = m+k$. Then we get

$$\frac{-4x\log(1-x)}{(1-x)^3} = \sum_{n=2}^{\infty}\left(\sum_{k=1}^{n}\frac{2(n-k)^2+2(n-k)}{k}\right)x^n. \tag{10}$$

Combining (8), (9), and (10), we can find $f(n)$ (for $n \geq 2$) as it should be the coefficient in front of x^n. Therefore, we have

$$f(n) = \frac{(n+1)n}{2} - \frac{n(n-1)}{2} + \sum_{k=1}^{n} \frac{2n^2 - 4nk + 2k^2 + 2n - 2k}{k}$$
$$= n + (2n^2 + 2n)\left(\sum_{k=1}^{n} \frac{1}{k}\right) - 4n^2 + n(n+1) - 2n.$$

Using that $\sum_{k=1}^{n} \frac{1}{k} \approx \log n$, we get

$$f(n) \approx 2n^2 \log n,$$

so

$$Q(n) \approx 2n \log n = \Theta(n \log n).$$

However, we can be even more precise and use

$$\sum_{k=1}^{n} = \log n + \gamma + \frac{1}{2n} + O\left(\frac{1}{n^2}\right),$$

where γ is the Euler-Mascheroni constant ($\gamma = 0.577215\ldots$), to get

$$f(n) = 2n^2 \log n + (2\gamma - 3)n^2 + 2n \log n + n(1 + 2\gamma) + O(1).$$

Therefore

$$Q(n) = 2n \log n + (2\gamma - 3)n + 2 \log n + (1 + 2\gamma) + O\left(\frac{1}{n}\right).$$

The advantage of this long method is that we get a precise formula (granted, the method in Lecture5a.pdf can also lead to a precise formula).

2 Andreas Klappernecker's solution

This solution does not need the recursion (1). The idea is to note that $Q(n)$ is the number of comparisons, and two elements can compare once or not-compare (in quicksort, you only need to compare once, and most pairs are not compared between themselves). Suppose A is the array with values $a_1, a_2, \ldots, a_n$ and that $s_1, s_2, \ldots, s_n$ is the sorted version. Given s_i, s_j with $i < j$, consider the subset

$$S_{ij} = \{s_i, s_{i+1}, \ldots, s_j\}.$$

For A to be sorted, at some point one element of S_{ij} must be a pivot (otherwise, we can't be certain these elements are sorted). Let p be the first pivot from S. If $p \in \{s_i, s_j\}$, then s_i and s_j get compared because one of them is the pivot and all of S must be on the same side. If $p \notin \{s_i, s_j\}$, then s_i and s_j get sorted into different sides and will never be compared. Since out of the $j - i + 1$ elements in S only two choices lead to s_i and s_j to be compared, the probability they end up compared is $\frac{2}{j-i+1}$. Then the expected number of comparisons between s_i and s_j is $\frac{2}{j-i+1}$.

By linearity of expectation

$$Q(n) = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} \approx 2 \sum_{i=1} \log(n - i + 1) = 2 \log n! \approx 2n \log n.$$