Homework 1: Exercise 1.16

Maryam Bahrani (`mbahrani`)

Dylan Mavrides

5/5

We will compute the expected number of times subarrays of size 0, 1, and 2 are encountered separately and add the results up. Let these sequences be denoted by A, B and C respectively.

In all three case, the same recurrence holds for large enough N, differing only in the base cases:

$$A_N = \frac{2}{N}\sum_{j=0}^{N-1} A_j, \quad A_0 = 1,\ A_1 = 0,\ A_2 = 1$$

$$B_N = \frac{2}{N}\sum_{j=0}^{N-1} B_j, \quad B_0 = 0,\ B_1 = 1,\ B_2 = 1$$

$$C_N = \frac{2}{N}\sum_{j=0}^{N-1} C_j, \quad C_0 = 0,\ C_1 = 0,\ C_2 = 1,\ C_3 = \frac{2}{3}$$

The base case calculations are done by manual brute force, which should continue until current index still satisfies the recursive expression. This ensures that we can subtract the previous index from the current index up until that point and simplify recursively.

We outline the base case calculations for A_N, noting that B and C are similar.

First, note that $A_0 = 1$ by definition.

Furthermore, $A_1 = 0$, since the recurrence stops when $N = 1$ without any calls to subarrays of size zero; however, A_1 does not satisfy the recurrence relation: $A_1 \neq \frac{2}{1}A_0 = 2$.

When $N = 2$, we observe that an array of size 2 is always partitioned into a pivot, a subarray of size 0, a subarray of size 1. That's *one* encounter of a subarray of size 0. Therefore, $A_2 = 1$. Additionally, A_2 *does* satisfy the recurrence relation: $\frac{2}{2}(A_0 + A_1) = 1$. This means that we can stop our manual computation of base cases.

We can now solve each recurrence, using the same methods as Exercise 1.14:

$$\begin{aligned}
A_N &= \frac{2}{N}\sum_{j=0}^{N-1} A_j \\
NA_N &= 2\sum_{j=0}^{N-1} A_j \\
NA_N - (N-1)A_{N-1} &= 2A_{N-1} \\
NA_N &= (N+1)A_{N-1} \\
\frac{A_N}{N+1} &= \frac{A_{N-1}}{N} = \cdots = \frac{A_2}{3} \\
A_N &= \frac{N+1}{3}.
\end{aligned}$$

The calculations for B are identical, and we have $B_N = \frac{N+1}{3}$.

The case of C reduces to

$$\begin{aligned}
\frac{C_N}{N+1} &= \frac{C_{N-1}}{N} = \cdots = \frac{C_3}{4} = \frac{1}{6} \\
C_N &= \frac{N+1}{6}.
\end{aligned}$$

In total, the number of calls to subarrays of size at most 2 is therefore $\frac{N+1}{3} + \frac{N+1}{3} + \frac{N+1}{6}$, i.e. $\frac{5(N+1)}{6}$.

Finally, we can make sure that this makes sense by observing that the number of times subarrays of size $k > 0$ are encountered is given by $\frac{N+1}{\frac{(k+1)(k+2)}{2}}$, where the denominator is the $(k+1)$th triangular number. Summing up over all none-zero k, and adding 1 for the initial call to an array of size N, the total number of calls is given by

$$\begin{aligned}
1 + \sum_{k=1}^{N-1} \frac{N+1}{\frac{(k+1)(k+2)}{2}} &= 1 + 2(N+1)\sum_{k=1}^{N-1} \frac{1}{(k+1)(k+2)} \\
&= 1 + 2(N+1)\left[\sum_{k=0}^{N-1} \frac{1}{(k+1)(k+2)} - \frac{1}{2}\right] \\
&= 1 + 2(N+1)\left[\frac{N}{N+1} - \frac{1}{2}\right] \\
&= 1 + 2N - (N+1) = N.
\end{aligned}$$

To get from the second line to the third, we have used the following identity (you can see solution to exercise 1.14 for an inductive proof of this): $\sum_{k=1}^{N} \frac{1}{k(k+1)} = \frac{N}{N+1}$.

By linearity of expectation, we expected the sum above to equal the expected number of calls to `quicksort` on non-empty subarrays computed in Exercise 1.14, which is indeed the case.