IE531: Algorithms for Data Analytics
Spring, 2020
Homework 2: The Randomized-Selection Algorithm
Due Date: February 21, 2020

Instructions

1. You can modify any of the Python code on Compass to solve these problems, if you want. It might help you with honing your programming skills.

2. You will submit a PDF-version of your answers on Compass on-or-before midnight of the due date.

Instructions

For the *Randomized Selection Algorithm* we picked a random member of the array as the *pivot* p (as opposed to p being the *Median-of-Medians*, when we did the *Deterministic Selection Algorithm*). Assume there is only one such p in the array – we have to compare with $(n-1)$ many others to split the original array into three arrays. This takes $c*(n-1)$ time. In the worst case, we will recurse on the longer of two arrays (i.e. the array that contains numbers smaller than p, and the array that contains numbers larger than p[1]. The average value of the number of elements in the longer array cannot be larger than $\frac{3n}{4}$. This means the average running time of the algorithm must satisfy the recursion

$$A(n) \leq c(n-1) + A\left(\frac{3n}{4}\right) \leq cn + A\left(\frac{3n}{4}\right) \Rightarrow A(n) \leq 4cn.$$

That is, the average running time of the *Randomized Selection Algorithm* must be linear in n.

To see the fact that the average value of the longer array is bounded by $\frac{3n}{4}$, we used an analogy of splitting a candy-bar of length L into two pieces by picking a uniformly distributed random variable in the interval $[0, L]$. It is not hard to see that the length of the longer-piece is uniformly distributed in the interval $[L/2, L]$, which means the average-value of the length of the longer-piece is $\frac{3L}{4}$.

In the January 22, 2019 class we discussed the possibility of using multiple pivots, and improving the average running time of the Randomized Selection Algorithm. For example, if we had two (random) pivots p_1 and p_2. Using p_1, after $(n-1)$-many comparisons, we split the original array into three parts; this is followed by identifying the appropriate sub-array that contains the k-th smallest elements (that we wish to find). A similar process can be done with the other pivot p_2 – after another $(n-1)$-many comparisons – we identify yet another sub-array that contains the k-th smallest element (that we wish to find). The logical thing to do, would be to recurse on the smaller of these two sub-arrays that contains the k-th smallest element.

[1] For the worst-case running-time we can glibly ignore the array that contains of elements in the original array that are equal to p , why?

We would like to get an estimate of the average running time of the Randomized Selection Algorithm in the presence of m-many randomly-chosen pivots. We will do this in multiple steps in this homework.

1. (*50 points*) You have m-many candy-bars, each of them has a length L. Each candy bar is broken into two pieces using a uniformly distributed random variable in the interval $x \in [0, L]$ where the two pieces have a length of x and $L - x$, respectively. We throw away the shorter of these two pieces and keep the longer one. In the end, we will have m-many longer-pieces of candy-bars. Show that the average value of the smallest of these m-many longer-pieces of candy-bars is

$$\frac{L(m+2)}{2(m+1)}$$

2. (*50 Points*) Show that the version of the Randomized Selection Algorithm that uses m-many pivots to split the original array into m-many sub-arrays, which is then followed by a recursion on the array of smallest-size that contains the k-th smallest element will have an average running time

$$A_m(n) \leq \frac{2(m+1)}{m} \times c \times n.$$