IE531: Algorithms for Data Analytics
Spring, 2020
Programming Assignment 3: Randomized-Selection Algorithm with Multiple Pivots
Due Date: March 6, 2020

1 Introduction

The *Recursive Randomized-Selection Algorithm* with a single pivot p, splits the original n-long array into three sub-arrays, where the first sub-array contains all elements of the array that are strictly less than p, the second sub-array contains all elements of the original array that are exactly equal to p, while the third sub-array contains all elements that are strictly greater than p. We are looking for the k-th smallest element in the original array – depending on the value of k and the lengths of the three sub-arrays, we identify one of these three sub-arrays to recurse on for the next round.

We consider the version of the *Recursive Randomized-Selection Algorithm* that uses m-many pivots. Just with the single-pivot case described above, for each pivot we would have a candidate sub-array that contains the element we want to find. In a sense, we have m-many candidate sub-arrays (one from each of the m-many pivots) that contains the element we want. The algorithm then recurses on the sub-array with the smallest-length. From Homework 2, you know that the average running-time $A_m(n)$ of this algorithm (for picking the k-th smallest element in an unordered-array of length n) satisfies the inequality:

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

This suggests that the average-running time can be made smaller by making m larger. But having a large value of m comes incurs a larger cost of comparisons before the appropriate sub-array can be identified for the next round of recursion. This lead us to hypothesize that there is really *no practical benefit* to using multiple pivots. At least, there will be a point beyond which using more pivots will be more computationally time-consuming over the single-pivot case. This programming assignment is about determining the optimal value of m experimentally.

Programming Assignment

You are going to modify the Python Code for the *Random-Selection Algorithm* written by me to create a create a version that uses multiple pivots. To make it easy on you – I have uploaded a hint.py Python Code sample on Compass. You have to write the nitty-gritty details of the function

```
def randomized_select_with_multiple_pivots (current_array, k, no_of_pivots)
```

for this programming assignment.

In the main part of hint.py I implemented the pseudo-code shown in figure 1. If the use of multiple pivots reduces the average run-time, for some (optimal) value of m, then we should see a minimum in the slope of the best-fit regressor of the mean running-time as a function of n.

Figure 2 shows fifteen plots of mean and standard-deviation of the running-time as a function of array-size n. Each of these plots also shows the best-fit linear regressor to the mean- and standard-deviation data, for $1 \leq m \leq 15$ pivots. Figure 3 shows the slope of the best linear regressor for the mean- and standard-deviation for $1 \leq m \leq 15$ pivots. If the slope of the best linear regressor is smaller for some value of m, then the average running-time will be the smallest for this specific choice of m. The data in figure 3 is shown in graphical form in figure 4. The results of my experiments show that there really is nothing to be gained by using multiple pivots.

You may/may-not get the same results as me... just saying!

What I need from you

You can submit your Python code on Compass – make sure it runs without issues. In addition, I would like you to upload a PDF file that shows the plots (cf. figure 2) and a short explanation of why you were able to confirm/refute the theory developed in Homework 2.

---

for $1 \leq m \leq 15$ **do**
    for $100 \leq n \leq 3,900$ in steps of 100 **do**
        for $1 \leq i \leq \#trials$ **do**
            Fill an array of size n with random values.
            Let $k = \lceil \frac{n}{2} \rceil$ (i..e k is almost the median)
            Find the amount of time it took to find the k-th smallest element in the array (for the present value of m, array-size n, and trial i)
        end for
        Compute the mean and standard-deviation of running-time of the $\#trials$-many experiments (for the present value of m, array-size n)
    end for
    Plot the mean-running-time as a function of n (for the specific choice of m, the number of pivots).
    Using polyfit in numpy compute the slope of the best-fit regressors for the mean and standard-deviation of running-time as a function of n (for the given value of m).
end for
Plot the slope of the best-fit regressors for the mean and standard-deviation of running time as a function of m. Check if there is a marked decrease in the slope as m increases (i.e. the average computation-time decreases as we use more pivots).

Figure 1: Experimentally determining if there is any value to using multiple pivots.

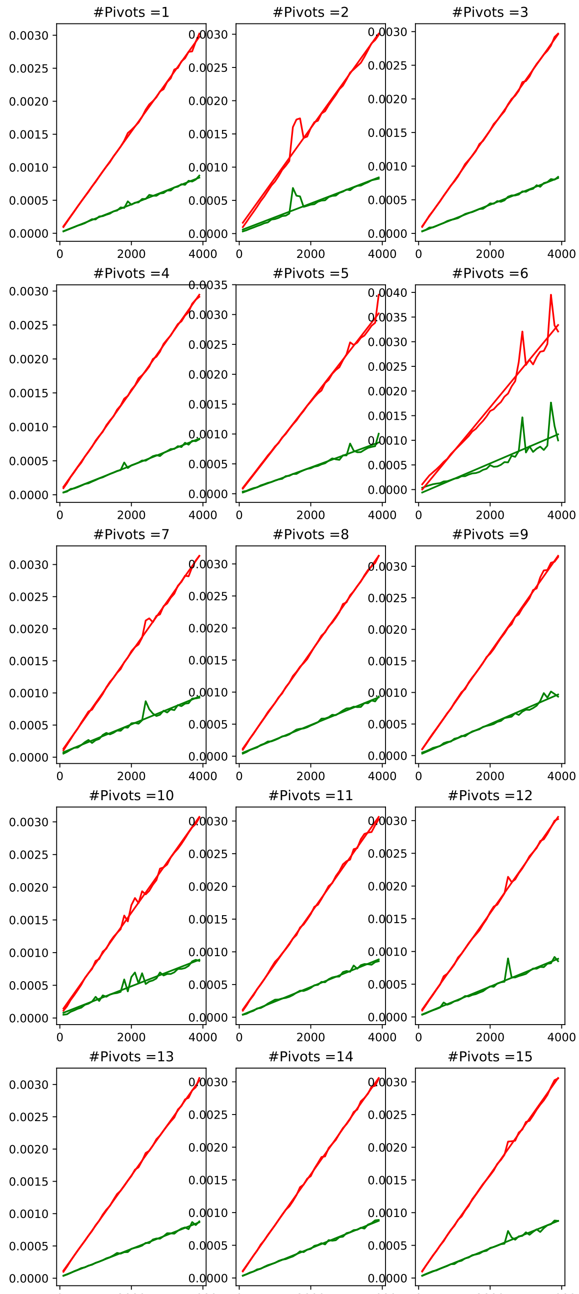

Figure 2: Mean and Standard-Deviation of the running-time (from 1000 trials) vs. n plots for different values of m

```
Randomized Select — -bash — 121×29
MacBook-Air-2:Randomized Select sreenivas$ time ./Randomized_Selection_Multiple_Pivots.py
#Pivots =  1 ; Mean-Regressor's slope =  7.553635158822422e-07 ; Std-Dev-Regressor's slope =  2.136263714167661e-07
#Pivots =  2 ; Mean-Regressor's slope =  7.490841019510275e-07 ; Std-Dev-Regressor's slope =  2.0637121051246537e-07
#Pivots =  3 ; Mean-Regressor's slope =  7.511882205765296e-07 ; Std-Dev-Regressor's slope =  2.0887967095127313e-07
#Pivots =  4 ; Mean-Regressor's slope =  7.449522200391849e-07 ; Std-Dev-Regressor's slope =  2.0831987906394674e-07
#Pivots =  5 ; Mean-Regressor's slope =  7.722711723310227e-07 ; Std-Dev-Regressor's slope =  2.2045976666676173e-07
#Pivots =  6 ; Mean-Regressor's slope =  8.784665852040032e-07 ; Std-Dev-Regressor's slope =  3.1068522558516845e-07
#Pivots =  7 ; Mean-Regressor's slope =  7.89923526800416e-07 ; Std-Dev-Regressor's slope =  2.2821716977473862e-07
#Pivots =  8 ; Mean-Regressor's slope =  7.916552415824208e-07 ; Std-Dev-Regressor's slope =  2.2756403997580943e-07
#Pivots =  9 ; Mean-Regressor's slope =  8.068216543517683e-07 ; Std-Dev-Regressor's slope =  2.464848935164871e-07
#Pivots =  10 ; Mean-Regressor's slope =  7.713694837445222e-07 ; Std-Dev-Regressor's slope =  2.133075844250673e-07
#Pivots =  11 ; Mean-Regressor's slope =  7.751038491776406e-07 ; Std-Dev-Regressor's slope =  2.2194246152261046e-07
#Pivots =  12 ; Mean-Regressor's slope =  7.744529173628707e-07 ; Std-Dev-Regressor's slope =  2.2378712275000414e-07
#Pivots =  13 ; Mean-Regressor's slope =  7.749527395318697e-07 ; Std-Dev-Regressor's slope =  2.174279745649604e-07
#Pivots =  14 ; Mean-Regressor's slope =  7.758206735790067e-07 ; Std-Dev-Regressor's slope =  2.2135048178825972e-07
#Pivots =  15 ; Mean-Regressor's slope =  7.765590819669071e-07 ; Std-Dev-Regressor's slope =  2.2095363444170595e-07
Sensitivity of the Slope of the Linear Regressor of the Mean to the #Pivots    =  1.2743418985864569e-09
Sensitivity of the Slope of the Linear Regressor of the Std-Dev to the #Pivots =  2.6334817011107375e-10

real	19m26.236s
user	15m25.739s
sys	0m3.797s
MacBook-Air-2:Randomized Select sreenivas$
```

Figure 3: Slope of the best-fit linear regressor for the mean- and standard-deviation for different values of m. If the slope is smaller for some value of m, then it means the average running-time is shorter for this choice.

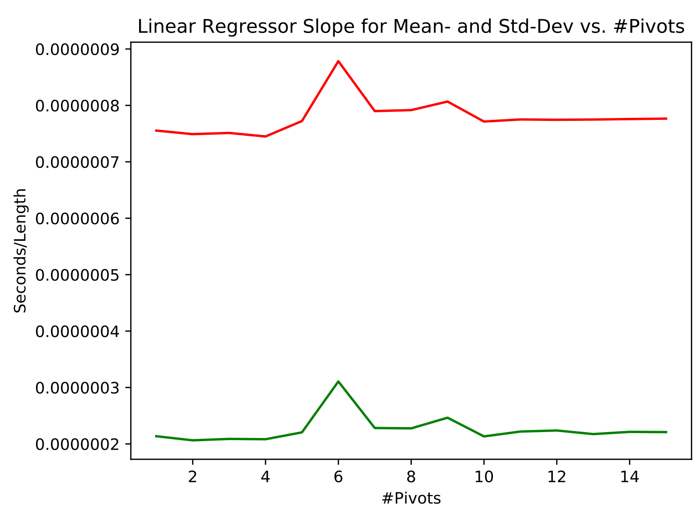

Figure 4: Plots of the slopes of the best-fit linear regressor as a function of m. These slopes are insensitive to the value of m.