CSCI 3104 PS8b (Extra Credit)

Luna Mcbride

TOTAL POINTS

15 / 18

QUESTION 1

18 pts

1.1 14 / 16

✓ **+ 2 pts** **Randomly shuffles array of range n**

✓ **+ 2 pts** **Graph/Table showing comparison**

✓ **+ 2 pts** **Correctly counts comparisons**

✓ **+ 3 pts** **Correct implementation of Quicksort**

✓ **+ 4 pts** **Correct implementation of QuickSelect**

✓ **+ 3 pts** **Correct implementation of partition algorithm**

✓ **- 2 pts** **Split percentage not updated correctly for recursive calls.**

Split value needs to be updated to reflect the partition that 'k' value being searched will have in the new subarray. For example, In an array of range 1...10, the split for k=7 will be 70/30 in the first call of quickselect, and considering the split to be 1...5 and 5...10, it will be (7-5)/(10-5)=20/50 in the second call on the subarray 5...10.

- 1 pts Not counting comparison outside of partition - in quickselect and quicksort. A call of quickselect will call partition multiple times and you miss out on many comparisons.

- 4 pts Does not honour quickselect signature asked in question, and uses order statistic k instead of split.

- 2 pts Not sorted in place

+ 0 pts No submission

+ 0 pts Code not running or running with error

1.2 1 / 2

+ 2 pts Correct

✓ **+ 1 pts** **Partially correct**

+ 0 pts No answer

+ 0 pts Incorrect

The correct recurrence would be T(n) = T(n/4) + T(3n/4) + O(n). The first two terms refer to the 25/75 split recurrence and the last refers to the running time of quickselect.

Name: Luna McBride
ID: 107607144

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 8b (18 Extra credit points) **Fall 2019, CU-Boulder**

Advice 1: For every problem in this class, you must justify your answer: show how you arrived at it and why it is correct. If there are assumptions you need to make along the way, state those clearly.

Advice 2: Verbal reasoning is typically insufficient for full credit. Instead, write a logical argument, in the style of a mathematical proof.

Instructions for submitting your solution:

- The solutions **should be typed** and we cannot accept hand-written solutions. Here's a short intro to Latex.
- You should submit your work through **Gradescope** only.
- If you don't have an account on it, sign up for one using your CU email. You should have gotten an email to sign up. If your name based CU email doesn't work, try the identikey@colorado.edu version.
- Gradescope will only accept **.pdf** files (except for code files that should be submitted separately on Gradescope if a problem set has them) and **try to fit your work in the box provided**.
- You cannot submit a pdf which has less pages than what we provided you as Gradescope won't allow it.

Name: Luna McBride
ID: 107607144

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 8b (18 Extra credit points) **Fall 2019, CU-Boulder**

Important:

1. This assignment has 1 (Q1) coding question.
   - You need to submit 1 python file.
   - The .py file should run for you to get points and name the file as following - If Q1 asks for a python code, please submit it with the following naming convention - `Lastname-Firstname-PS8b-Q1.py`.
   - You need to submit the code via Canvas but the table/plot/result should be on the main .pdf.
2. This assignment is all Extra Credit.
3. **This doesn't have any late submissions.**

Name: Luna McBride
ID: 107607144

CSCI 3104, Algorithms **Profs. Hoenigman & Agrawal**
Problem Set 8b (18 Extra credit points) **Fall 2019, CU-Boulder**

1. (18 pts total) In recitation, you learned about about *Quickselect* algorithm, which can find the k^{th} order statistic (or the k^{th} smallest number) of a given unsorted array in $O(n)$ average time. You will implement *QuickSelect* and use it to find a *Pivot* that gives you a desirable split every time you perform *Partition* in *QuickSort*.

   - Implement $QuickSelect(A, p, r, split)$ **to find a suitable pivot** at each round such that it splits the subarray in consideration in two parts of size roughly 1/4 and 3/4 after partition (or 0.25-0.75 split). This implementation doesn't need the order statistic k but only the smaller split fraction which in this case is $split = 0.25$. **'split' is not the order statistic 'k'**.
   - Swap and put this pivot (which will achieve the desired 1/4-3/4 split) at the end and perform $QuickSort(A, p, r)$ as usual.
   - Count the total number of comparisons in one run of the QuickSort. Note that the number of comparisons in QuickSort is proportional to the runtime of Quicksort. This will help you verify the correctness of your plot.
   - Your implementation should be in-place (should not use an extra array to do QuickSort).
   - There will be some points associated with comments and good programming practices for code and result presentation.
   - Code for the experiment should also be present to get any points for the results.

   (a) (16 pts) Write a single Python code, which takes as input a positive integer n, **randomly shuffles an array of size n** with elements $[1, \ldots, n]$ and performs QuickSort on that shuffled array following the above conditions and counts the total comparisons.
   Experiment - Run your code on a bunch of n values from $[2, 2^2, 2^3, ...2^{10}]$ and present your result in a table with one column as the value of n and another as the number of comparisons. Alternatively, you can present your table in form of a labeled plot with the 2 columns forming the 2 axes.
   Also, print the shuffled array and the sorted output for $n = 5, 10, 20$ showing the state of the array A after each recursive call for each of these three n values.
   (All the result and prints should be on the pdf and the code should be submitted on Canvas.)

   (b) (2 pts) Assume the average runtime for QuickSelect i.e. $O(n)$, write the recurrence relation for your QuickSort implementation that selects pivot using your

QuickSelect (include lower order terms for the sake of clarity). Explain the recurrence briefly.

Solution.
Possible relations:
Best Case: T(n)=2T($\frac{n}{2}$)+n
Worst Case: T(n)=2T(n-1)+n

As it is, since quickselect is $O(n)$, it does not increase the overall cases that QuickSort gives. In terms of my code, the worst case always came up just a bit over n^2. This is because although the worst case is n^2 for a QuickSort, Big O does not account for the smaller terms that make it that much bigger (in a relatively small way, of course). However, because the jump above n^2 is so small relative to each individual quickselect that is done, it does relatively nothing to the overall count, and as such, worst case remains at n^2, and as such, remains at the speed of a QuickSort in the grand scheme of the program.

Solution.
Outputs for n=5,10,20:
[5 1 3 4 2]
[1 2 3 4 5]
[1 2 3 4 5]
[1 2 3 4 5]
[1 2 3 4 5]
[10 7 3 1 4 5 6 9 8 2]
[1 2 3 4 5 6 7 8 9 10]
[1 2 3 4 5 6 7 8 9 10]
[1 2 3 4 5 6 7 8 9 10]
[1 2 3 4 5 6 7 8 9 10]
[1 2 3 4 5 6 7 8 9 10]
[1 2 3 4 5 6 7 8 9 10]
[1 2 3 4 5 6 7 8 9 10]
[1 2 3 4 5 6 7 8 9 10]
[1 2 3 4 5 6 7 8 9 10]
[1 2 3 4 5 6 7 8 9 10]
[8 6 12 3 2 17 11 5 10 15 13 14 7 9 1 20 16 18 4 19]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]
[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]

Name: Luna McBride
ID: 107607144

[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]

N Conditionals
—- ———
2 2
4 20
8 44
16 169
32 578
64 1352
128 13793
256 63456
512 243304
1024 877421

(Extra space)

(Extra space)

(Extra space)