Northeastern University
CS5800 – Algorithms
Fall 2023 Semester, Jonathan Mwaura

Problem Set #4

This Problem Set is due at **11:30PM on Wednesday** 11^{th} Oct, and will be submitted on GRADESCOPE.

This Problem Set will be marked out of 40. There are two problems, each worth 20 points.

Please type (or neatly handwrite) your solutions on standard 8.5×11 paper, with your name at the top of each solution. Ensure that you submit you solutions in one file PDF file on Gradescope. **each problem sets solution should be on in its own individual page, Gradescope will help ensure you submit each solution under its correct problem number**

While a solution must be absolutely perfect to receive full marks, I will be generous in awarding partial marks for incomplete solutions that demonstrate progress.

So that there is no ambiguity, there are two non-negotiable rules. A violation of either rule constitutes plagiarism and will result in you receiving an F for this course.

(a) If you meet with a classmate to discuss any of the Individual Problems, your submission must be an individual activity, done in your own words, away from others. The process of finding a solution might take 3 - 5 iterations or even more BUT you learn from all these attempts and your confidence grows with each iteration.

(b) These problem sets might seem hard on a first look. They are designed to be so. We learn by attempting problems, struggling through them and coming on top. I encourage you to make this learning exercise worth your while. What do I mean? Open the problem sets as early as you get them, then do not look at hints or answers any where (including on the internet and consulting other students for direct answers), give it the best shot you can. If you get stuck come to Professor or TA's office hour and we shall be glad to listen to your rationale and work with you till you are able to tackle the problem sets.

Problem #1

Quicksort is a powerful divide-and-conquer sorting algorithm that can be described in just four lines of pseudocode.

```
QUICKSORT(A, p, r)
1  if $p < r$
2      $q$ = PARTITION(A, p, r)
3      QUICKSORT(A, p, $q - 1$)
4      QUICKSORT(A, $q + 1$, r)
```

The key to Quicksort is the PARTITION(A, p, r) procedure, which inputs elements p to r of array A, and chooses the final element $x = A[r]$ as the pivot element. The output is an array where all elements to the left of x are less than x, and all elements to the right of x are greater than x.

In class, we saw that there are very many techniques to partition the array. One nice technique covered in the book was invented by Nico Lomuto (See page 171 of the class textbook). You can also watch a quick youtube video here: `https://www.youtube.com/watch?v=86WSheyr8cM`. In this question, we will use the Lomuto Partition Method. Please assume that the pivot is *always* the last (right-most) element of the input array.

For example, if $A = [2, 8, 7, 1, 3, 5, 6, \mathbf{4}]$, then the pivot element is $x = A[8] = \mathbf{4}$, and PARTITION$(A, 1, 8)$ returns the array $[2, 1, 3, \mathbf{4}, 7, 5, 6, 8]$. We then run PARTITION on the sub-arrays $[2, 1, 3]$ and $[7, 5, 6, 8]$.

(a) Demonstrate the Quicksort algorithm on the input array $A = [3, 1, 5, 7, 6, 2, 4]$, showing how eventually the algorithm outputs the sorted array $[1, 2, 3, 4, 5, 6, 7]$. Clearly show all of your steps.

(b) When PARTITION is called on an array with n elements, we require $n - 1$ comparisons, since we must compare the pivot element to each of the other $n - 1$ elements.

If the input array is $A = [1, 2, 3, 4, 5, 6, 7]$, show that Quicksort requires a total of 21 comparisons.

(c) Determine an input array with 7 elements for which Quicksort requires the *minimum* number of total comparisons.

Clearly demonstrate why your input array achieves the minimum number of comparisons, and explain why there cannot exist a 7-element array requiring fewer comparisons than your array.

(d) Let A be an array with $n = 2^k - 1$ elements, where k is some positive integer. Determine a formula (in terms of n) for the *minimum* possible number of total comparisons required by Quicksort, as well as a formula for the *maximum* possible number of total comparisons required by Quicksort.

Use your formulas to show that the running time of Quicksort is $O(n \log n)$ in the best case and $O(n^2)$ in the worst case.

Problem #2

There are are about 44 LeetCode problems on Divide and Conquer techniques.

In this question, you will create a **mini-portfolio** consisting of two (two) LeetCode problems on Divide and Conquer, chosen from the following website.

Note that you can work with a partner to code up the problem but each person MUST do their own analysis of the problem

`https://leetcode.com/tag/divide-and-conquer/`

As always, you may code your algorithms in the programming language of your choice.

Here is how your mini-portfolio will be graded.

(i) (5 points) For each of the problems you are including in your mini-portfolio, provide the problem number, problem title, difficulty level, and the screenshot of you getting your solution accepted by LeetCode.

Note that the total points is 5 points for each problem and those points is dependent on the difficult level - for each *easy* problem will be awarded 3 points, each *medium* problem will be awarded 4 points, and each *hard* problem will be awarded 5 points.

You may submit as many problems as you wish but you will receive a <u>maximum</u> of 5 points for each problem and a total of 10 for the two problems.

You will get full credit for *any* correct solution accepted by LeetCode, regardless of how well your runtime and memory usage compares with other LeetCode participants.

(ii) (5 points) For **one** of the problems you are including in your mini-portfolio, provide an analysis of the run time of your algorithm. Be careful to only analyse what you created (i.e. your own code) and not a general solution.

(ii) (5 points) For **one** of the problems you are including in your mini-portfolio, explain the various ways you tried to solve this problem, telling us what worked and what did not work. Describe what insights you had as you eventually found a correct solution. Reflect on what you learned from struggling on this problem, and describe how the struggle itself was valuable for you.

The choice of problems is yours, though you may only include problems that took you a minimum of 30 minutes to solve.