EECS 325 – Fall 2023
Analysis of Algorithms

Homework 2

Instructions:

- This homework is due by 11:59PM Pacific Time on **Thursday, 10/19**. You will receive one grace period allowing you to submit a homework assignment up to two days late for the quarter. Subsequent assignments submitted up to two days late will incur a 20% penalty. Assignments submitted more than two days late will not be graded and will receive a score of 0. Your lowest homework score of the quarter (after applying late penalties) will be dropped.
- You must submit your written solutions as a single .pdf file on Canvas with your name at the top. Typesetting your written solutions using LaTeX is strongly encouraged. You must write your programming solutions as a single .py file. We have provided a skeleton file.
- You are welcome and encouraged to discuss the problems in groups of up to three people, but **you must write up and submit your own solutions and code**. You must also write the names of everyone in your group on the top of your submission. Students in the honors and non-honors section may collaborate.
- The primary resources for this class are the lectures, lecture slides, the CLRS and Erickson algorithms textbooks, the teaching staff, your (up to two) collaborators, and the class Piazza forum. We strongly encourage you only to use these resources. If you do use another resource, make sure to cite it and explain why you needed it.
- There are 4 questions, worth a total of 40 points.
- You must justify all of your answers unless specifically stated otherwise.

Questions

Question 1. (Structure of DFS forests, 7 points.) Exercise 22.3-11 (CLRS 3rd ed.)/Exercise 20.3-10 (CLRS 4th ed.).

(**Hint**: Note that this question crucially uses the fact that the graph is directed.)

Question 2. (Space complexity of BFS versus DFS, 9 points.) Recall that the *space complexity* of an algorithm is the maximum number of memory cells that it uses at a given point in time. For this question, the space complexity of BFS is the maximum number of vertices stored in its queue, and the space complexity of recursive DFS is the maximum depth of its call stack.[1]

a. (6 points.) Give an example of a graph $G = (V, E)$ on which DFS has lower space complexity than BFS, regardless of which vertex the algorithms start on. Make sure to justify your answer.

b. (3 points, ♣.) What about a graph G determines the space complexity of DFS?

Question 3. (Maximum Subarray Problem continued, 11 points.) In this question, you will continue to study the Maximum Subarray Problem, which is defined as follows.

Maximum Subarray Problem
Input: An array $A[0, \ldots, n-1]$ of n integers.
Output: The maximum sum of all elements in some (possibly empty) contiguous subarray of $A[0, \ldots, n-1]$. Formally,

$$\max\left(0, \max_{i \le j} \sum_{k=i}^{j} A[k]\right) ,$$

where 0 corresponds to the value of the empty array.

You will implement the third algorithm described in Prof. Borradaile's "TCS crash course" (see page 11).[2] You will then run your code on inputs of different sizes, and will report on the performance of the algorithm.

a. (7 points.) Implement the $O(n \log n)$-time "simplification and delegation" (i.e., divide-and-conquer) algorithm.

b. (4 points.) Measure the runtime of the algorithm from Item a on each of the 10 provided input arrays (which are of size $n = 500i$ for $i = 1, 2, \ldots, 10$).

Make a scatter plot of the runtimes for the "simplification and delegation" algorithm to your scatter plot. The x-axis should be the size n of the input array, and the y-axis should be the runtime of the algorithm in milliseconds. (Making a scatter plot of this form should be easy in Excel, Google Sheets, etc.)

How do the runtimes compare to the algorithms you implemented in Homework 1?

[1] As we discussed in class, this is equivalent to the maximum number of gray vertices during a run of BFS or DFS, using the pseudocode from the textbook. Note that Item a considers the *exact* complexity of the algorithms on a G, i.e., the space complexity will be a number like 5 or 7 rather than an asymptotic expression like $O(n)$.

[2] Available at https://web.engr.oregonstate.edu/~glencora/wiki/uploads/tcscrashcourse.pdf.

Question 4. (Counting inversions, 13 points.) Do CLRS Question 2-4.

(**Hint**: For part (c), note that the pseudocode for insertion sort is earlier in the chapter (on p.18 in CLRS, third edition). For part (d), think about how to count the number of inversions in an array $A = A[1, \ldots, n]$ when each half of the array is sorted, i.e., $A[1, \ldots, \lfloor n/2 \rfloor]$ and $A[\lfloor n/2 \rfloor + 1, \ldots, n]]$ are sorted, where $\lfloor \cdot \rfloor$ is the floor function.)