Sabancı University
Faculty of Engineering and Natural Sciences

CS301 – Algorithms

Homework 2

Due: April 13, 2023 @ 23.55 (Upload to SUCourse)

PLEASE NOTE:

Provide only the requested information and nothing more. Unreadable, unintelligible and irrelevant answers will not be considered.

You can collaborate with your TA/INSTRUCTOR ONLY and discuss the solutions of the problems. However you have to write down the solutions on your own.

Plagiarism will not be tolerated.

Late Submission Policy:

Your homework grade will be decided by multiplying what you normally get from your answers by a “submission time factor (STF)”.

If you submit on time (i.e. before the deadline), your STF is 1. So, you don’t lose anything.

If you submit late, you will lose 0.01 of your STF for every 5 mins of delay.

We will not accept any homework later than 500 mins after the deadline.

SUCourse+’s timestamp will be used for STF computation.

If you submit multiple times, the last submission time will be used.

Question	Points	Score
1	35	
2	15	
3	30	

please proceed to the next page –→

4	20	
Total:	100	

Question 1 [35 points]

Every year, we send some of CS301 students to the "Center of Advanced Algorithms" in Westeros for a scientific visit, where all expenses are paid by the university. We send the most successful students in CS301 to this visit. We measure the success based on overall numeric grade in CS301.

The number of students selected for this visit depends on the budget available, changes from one year to another.

Since the number of students taking CS301 is increasing, we want to decide the students that we will send for this visit automatically by using an algorithm. This algorithm should use the student information (e.g. student id and CS301 overall numeric grade of the student) and the number of students that will be sent to the visit. Let's say there are *n* students in CS301, and we will send *m* of these students to the visit.

The ties that we might have between the students with the same grade will be broken randomly.

(a) [20 points] Please design an algorithm, as efficient as possible, which we can use for this purpose.
Do not write any code (pseudo or actual). Please only explain how you would solve this problem in a couple of sentences.

assume n students with their grades are listed in students array.

First: Before the sorting algorithm, randomly shuffle n elements in students array (randomly break ties): O(n)

Second: I will use QuickSort algorithm with random partitioning. Partition gives 2 subarrays; 1st subarray has students whose grades are more than the pivot, second subarray has students whose grades less than or equal to pivot.

if 1st subarray size is equal to m → we choose those m students

if 1st subarray size is more than m → call QuickSort recursively with 1st subarray until we have m students

if 1st subarray size less than m → choose all students in 1st subarray then call QuickSort recursively with 2nd subarray until it gives remaining (m – 1st subarray size) students.

QuickSort in total: O(nlogn)

(b) [15 points] Give an upper bound for the run time complexity of your algorithm (astight as possible).

First: Shuffle n students → O(n) time.

Second: QuickSort algorithm with randomized partitioning gives O(nlogn)

This algorithm results in O(nlogn) upper bound.

please proceed to the next page –→

Question 2 [15 points]

Suppose that you have n friends $F = \{f_1, f_2, ..., f_n\}$. Some of your friends know each other. If your friends f_i and f_j know each other, they follow each other on social media and they can see the messages posted for each other. So, when you send a message to your friend f_i, if f_j is a friend of f_i, f_j will also see this message coming from you to f_i.

On April 23 (National Sovereignty and Children's Day in Turkiye) you want send a message to all your friends. However, if you send the same message to all your friends, those friends of yours f_i and f_j who are also friends of each other, will see that you are sending the same message to them. Hence they would not feel special, getting such a general message from you.

However, sending each friend a different message would mean writing n different messages, which is not easy.

Then you consider the following. If you write the same message to two of your friends f_i and f_j who don't know each other, then since they would not see this same message, they would still feel special. Using this idea, you can reduce the number of different messages that you need to write.

We can state this problem as an optimization problem as follows:

Problem Definition (optimization version):
Given your friends $F = \{f_1, f_2, ..., f_n\}$ and for each pair of your friends $f_i, f_j \in F$, whether f_i and f_j know each other or not, what is the minimum number of different messages that you need to write, so that no two of your friends who know each other will get the same message?

Please state the same problem as a decision problem:

Given an integer k and your friends F={f1, f2,…, fn} and for each pair of your friends fi, fj ∈ F, whether fi and fj know each other or not, determine if it is possible to find a way to write k or less different messages, so that no two of your friends who know each other will get the same message?

Question 3 [30 points]

Mark the following statements as true or false. Give short explanations for your answers (no credits without an explanation).

please proceed to the next page –→

(a) [10 points] A red-black tree insertion requires $O(1)$ rotations in the worst case.

FALSE

Worst case insertion of a red-black tree requires O(logn) rotations.

To fix red-black tree violations in the worst case: newly inserted node has to be rotated up for height of the tree which makes O(logn) rotations.

(b) [10 points] A red-black tree insertion requires $O(1)$ node recoloring in the worst case.

FALSE

Worst case insertion of a red-black tree requires O(logn) node recoloring.

Again, to fix red-black tree violations in the worst case: newly inserted node has to be recolored up for height of the tree which makes O(logn) recolorings.

(c) [10 points] Walking a red-black tree with n nodes in pre-order takes $\Theta(n\lg n)$.

FALSE

Walking a red-black tree with n nodes in pre-order: Θ(n)

First root node, then left subtree recursively, then right subtree recursively visited (1 time for each node)

please proceed to the next page –→

Question 4 [20 points]

(a) [10 points] What does NP stand for?

NP stands for nondeterministic polynomial time.

It is a class with decision problems, to verify NP problems in polynomial time a proposed solution should be given. Also there is not an algorithm (not known/proven yet) which solves all NP problems with polynomial time.

(b) [10 points] When do we say a problem is in NP?

If a problem (decision) can be verified with a nondeterministic Turing machine: problem is in NP.

A polynomial time algorithm needs to check if solution is correct or not.

– THE END –