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	
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.

First we can sort the students taking CS301 based on their numeric grade in descending order using heapsort: Form a min heap, keep deleting elements from the root and insert the deleted elements in the first available place in the array so that we won't use extra memory. Time complexity would be $O(n \log n)$. The resulting array will be orded (descending) then we can simply access the students with the max grades by using the pop function that will return the element located in the root (largest)

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

When we use heapsort forming the heap would take $O(n)$ time and the complexity of heapifying and accessing the maximum value m time is $O(m \log n)$ So the overall upperbound is $O(m \log n)$

Question 2 [15 points]

Suppose that you have n friends $F = \{f_1, f_2, \ldots, f_n\}$. 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 Türkiye) 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 $F = \{f_1, f_2, \ldots, f_n\}$ 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 your friends $F = \{f_1, f_2, \ldots, f_n\}$ and for each pair of your friends $f_i, f_j \in F$ and whether f_i and f_j know each other or not is there a way to write a message to each of your friend in F such that no two friends who know each other will receive the same message with using m different messages where $m<n$

(Note that decision problems are problems in which the answer is yes/no and the goal is to determine whether a given input satisfies specified conditions.)

Question 3 [30 points]
Mark the following statements as true or false. Give short explanations for your answers (no credits without an explanation).

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

False. Insertion may cause the violation of RBT properties and we need rotations to fix these violations. In the worst case the newly inserted node is the deepest node and all its ancestors are colored red. This violates if a node is red both its children has to be black. To fix this we need to perform series of rotations and fix colorings that may go up to the root. Since the height of RBT is $O(\log n)$ number of rotations can go up to $O(\log n)$ in the worst case.

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

False. Again after an insertion we may violate RBT properties. The worst case occurs if the inserted node is the deepest node and the ancestors are red: this violates that if a node is red all its children has to be black property. To fix this we need to recolor the newly inserted node and the ancestors to black and continue this process up to the root. Since we are going up to the root the node recoloring will be based on the height of the tree and we know that height of RBT is O(logn) there for the worst case insertion requires O(logn) recoloring of nodes.

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

False, in pre-order we visit the current node then recursively the leftsubtree then the right subtree. Since each node will be visited exactly once the time it takes will be equal to the number of nodes in the RBT which is O(n

Question 4 [20 points]

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

NP stands for Nondeterministically Polynomial, that refers to the class of decision problems that can be verified in polynomial time by using deterministic Turing machine such that there exists a polynomial time algorithm to verify the correctness of the solution.

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

A problem is said to be in NP if there exists a polynomial time algorithm to verify whether a given solution is correct or not but it may not be efficiently solved by a deterministic algorithm. So a problem is in NP if we can verify its correctness by a polynomial time algorithm so that we can check if the solution to this problem is correct or not in linear time but if a problem is in NP finding a solution itself may not be done in polynomial time.
(So polynomial time to verify but not necessarily polynomial to solve: a problem in NP.)

Zeynep Kurtuluş
29015