Homework 4

CS 4104 (Spring)

Assigned on March 3, 2021.
Submit PDF solutions on Canvas
by 11:59pm on March 10, 2021.

Instructions:

- The Graduate Honor Code applies to this homework.
  - You can pair up with another student to solve the homework. Please form teams yourselves. Of course, you can ask the instructor for help if you cannot find a team-mate. You may choose to work alone.
  - You are allowed to discuss possible algorithms and bounce ideas with your team-mate. **Do not discuss proofs of correctness or running time in detail with your team-mate. You must write down your solution individually and indepedently. Do not send a written solution to your team-mate for any reason whatsoever.**
  - *In your solution, write down the name of the other member in your team. If you do not have a team-mate, please say so.*
  - Apart from your team-mate, you are not allowed to consult sources other than your textbook, the slides on the course web page, your own class notes, the TAs, and the instructor. In particular, do not use a search engine.
- Do not forget to typeset your solutions. *Every mathematical expression must be typeset as a mathematical expression, e.g., the square of n must appear as n^2 and not as "n^2".* You can use the LaTeX version of the homework problems to start entering your solutions.
- Do not make any assumptions not stated in the problem. If you do make any assumptions, state them clearly, and explain why the assumption does not decrease the generality of your solution.
- You must also provide a clear proof that your solution is correct (or a counter-example, where applicable). Type out all the statements you need to complete your proof. *You must convince us that you can write out the complete proof. You will lose points if you work out some details of the proof in your head but do not type them out in your solution.*
- If you are proposing an algorithm as the solution to a problem, keep the following in mind (the strategies are based on mistakes made by students over the years):
  - Describe your algorithms as clearly as possible. The style used in the book is fine, as long as your description is not ambiguous. Explain your algorithm in words. A step-wise description is fine. *However, if you submit detailed code or pseudo-code without an explanation, we will not grade your solutions.*
  - Do not describe your algorithms only for a specific example you may have worked out.
  - Make sure to state and prove the running time of your algorithm. You will only get partial credit if your analysis is not tight, i.e., if the bound you prove for your algorithm is not the best upper bound possible.
  - You will get partial credit if your algorithm is not the most efficient one that is possible to develop for the problem.
- In general for a graph problem, you may assume that the graph is stored in an adjacency list and that the input size is $m + n$, where n is the number of nodes and m is the number of edges in the graph. Therefore, a linear time graph algorithm will run in $O(m + n)$ time.

My team-mate is ____________________

Problem 1 (30 points) Let $G = (V, E)$ be an undirected connected graph and let $c : E \to \mathbb{R}^+$ be a function specifying the costs of the edges, i.e., every edge has a positive cost. Assume that no two edges have the same cost. Given a set $S \subset V$, where S contains at least one element and is not equal to V, let e_S denote the edge in E defined by applying the cut property to S, i.e.,

$$e_S = \arg\min_{e \in cut(S)} c_e.$$

In this definition, the function "arg min" is just like "min" but returns the argument (in this case the edge) that achieves the minimum. Let F be set of all such edges, i.e., $F = \{e_S, S \subset V, S \neq \emptyset\}$. In the definition of F, S ranges over *all* subsets of V other than the empty set and V itself. Answer the following questions, providing proofs for all but the first question.

(i) (5 points) How many distinct cuts does G have? We will use the same definition as in class: a cut is a set of edges whose removal disconnects G into two or more non-empty connected components. Two cuts are distinct if they do not contain exactly the same set of edges. For this question, just provide an upper bound.

(ii) (8 points) Consider the graph induced by the set of edges in F, i.e., the graph $G' = (V, F)$. Is G' connected?

(iii) (7 points) Does G' contain a cycle?

(iv) (5 points) How many edges does F contain?

(v) (5 points) What conclusion can you draw from your answers to the previous statements?

Problem 2 (35 points) You return home for the weekend all agog with the exciting new ideas you have discovered in the algorithms class. You tell your evil twin about the Minimum Spanning Tree (MST) problem and the clever algorithms for computing it. Your sibling pooh poohs your new-found wisdom and proposes the following simple algorithm on to compute the MST of an undirected, connected graph G, assuming that no two edges have the same cost.

1. Maintain a set T of edges. Initially T is empty.
2. Process the edges of E in *any* order.
3. For each edge $e \in E$,
   (a) Add e to T.
   (b) If T contains a cycle, delete e from T.

Something seems fishy. Could this algorithm really compute the MST? Show up your sibling by fixing the algorithm so that T is indeed the MST at the end and prove that the modified algorithm computes the MST of G. What you should include in your solution is both the corrected algorithm and its proof of correctness.

Notes: (a) The algorithm is not the same as Kruskal's algorithm since it processes the edges in *any* order. In contrast Kruskal's algorithm processes the edges in increasing order of cost. (b) In your fix, you decide not to sort the edges by cost or use any data structure such as the priority queue to sort the edges by cost. (c) We are interested only in proving the correctness of this algorithm. We are not interested in its running time. *Most of the points are for a clear and complete proof of correctness.*

Hint: I am providing an elaborate hint here. Your proof of correctness should show that at the end of the algorithm, T is an MST. Therefore, you have to prove all three points implied by the phrase "Minimum Spanning Tree." Prove each of these statements:

(a) T **does not contain a cycle.** This proof should be easy.

(b) T **is spanning, i.e., connects all vertices in** G**.** This part can be challenging. Consider an arbitrary subset S of V. It is enough to prove that at the end of the algorithm, T contains at least one edge in cut(S).[1] As the modified algorithm progresses, what can you show about that edges in cut(S) that are also in T? Informally, I am suggesting that you imagine the algorithm is running in the background while you focus your attention on the edges in cut(S). The algorithm will process these edges in some order. Think about this order to show that at the end of the algorithm, T contains at least one edge in cut(S).

(c) T **is an MST.** If you have proven the first two parts, then you know that T is a spanning tree. How many edges can it contain? If you modified the algorithm correctly, then what can you say about the edges that you did not include in T? Remember that the algorithm has processed every edge in G. Now combine what know so far with what you proved in part (v) of Problem 1.

Problem 3 (35 points) You are given two sets of n points. The first set of points $P = \{p_1, p_2, \ldots, p_n\}$ lies on the line $y = 0$. The other set of points $Q = \{q_1, q_2, \ldots, q_n\}$ lies on the line $y = 1$. None of these point sets is sorted by x-coordinate. Now construct a set of n line segments as follows: for each $1 \leq i \leq n$, connect p_i to q_i. Develop a divide-and-conquer algorithm that computes how many pairs of these line segments intersect. Your algorithm should run in $O(n \log n)$ time. You may assume that all the points in P have different x-coordinates and that all the points in Q also have different x-coordinates. (A point in P may have the same x-coordinate as a point in Q, but this possibility will not affect your algorithm.)

[1] For every subset S, if T contains at least one edge in cut(S), then T is connected. You may assume this fact.