Homework 4

CS 4104 (Spring 2014)

Assigned on Wednesday, March 5, 2014.
Submit PDF solutions on Scholar
by the beginning of class on Wednesday, March 19, 2014.

Instructions:

- You can pair up with another student to solve the homework. 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.** Please form teams yourselves. Of course, you can ask me for help if you cannot find a team-mate. You may choose to work alone. *Each of you must write down your solution individually, and write down the name of the other member in your team. If you do not have a team-mate, please say so.* **If your solution is largely identical to that of your team-mate or another student, we will return it ungraded.**
- Apart from your team-mate, you are not allowed to consult any 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".* Students can use the LaTeX version of the homework problems to start entering their solutions.
- 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 pseudo-code without an explanation, we will not grade 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.
- Do not describe your algorithms only for a specific example you may have worked out.
- 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.*
- Describe an analysis of your algorithm and state and prove the running time. 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.

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 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 (25 points) You return home for Spring Break all agog with the exciting new ideas you have discovered in the algorithms class. You tell your evil twin sibling 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 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. *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.

Problem 3 (15 points) Consider the version of Dijkstra's algorithm shown below written by someone (maybe the same evil twin sibling) with access to a priority queue data structure that supports *only* the INSERT and EXTRACTMIN operations. Due to this constraint, the difference between this version and the one discussed in class is that instead of the CHANGEKEY$(Q, x, d'(x))$ operation in Step , this version simply inserts the pair $(x, d'(x))$ into Q. The danger with this algorithm is that a node x may occur several times in Q with different values of $d'(x)$. Answer the following questions.

1. (5 points) What is the running time of this algorithm? Just state the bound in terms of the number of nodes n and the number of edges m in G.
2. (5 points) When the algorithm inserts a pair (x, d_1) into Q, suppose the pair (x, d_2) is already in Q. What is the relationship between d_1 and d_2?
3. (5 points) Using this relationship, how will you fix this algorithm? You just have to describe your correction in words, e.g., by saying "I will add the following command after Step X: ..." You do not have to prove the correctness of your algorithm.

Algorithm 1 DIJKSTRA'S ALGORITHM(G, l, s)

```
1: INSERT(Q, s, 0).
2: while S ≠ V do
3:    (v, d'(v)) = EXTRACTMIN(Q)
4:    Add v to S and set d(v) = d'(v)
5:    for every node x ∈ V − S such that (v, x) is an edge in G do
6:       if d(v) + l_(v,x) < d'(x) then
7:          d'(x) = d(v) + l_(v,x)
8:          INSERT(Q, x, d'(x))
9:       end if
10:   end for
11: end while
```

Problem 4 (30 points) You are given a graph $G(V, E)$ and a minimum spanning tree (V, T) for G. You now pick an edge e in E and change its cost from c to c'. Describe an algorithm to update T so that it is a minimum spanning tree for the graph with the new edge weight. Your algorithm must run in time linear in the number of nodes and edges in G. There are multiple cases to consider, depending on whether e is in T or not and whether c is larger or smaller than c'. Describe each case separately. You can assume that all edge costs are distinct, both before and after the change.