Homework 5

CS 5114 (Fall 2018)

Assigned on October 22, 2018.
Submit PDF solutions on Canvas by the
beginning of class on October 29, 2018.

Instructions:

- The Graduate Honor Code applies this to homework. In particular, you are not allowed to consult any sources other than your textbook, the slides on the course web page, your own class notes, and the instructor. 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.
- 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 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.
- You will also 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.

Problem 1 (15 points) Suppose you have a potential solution to a maximum-flow problem, i.e., you have a flow network $G = (V, E)$ (with s and t and edge capacities, defined, of course) and an assignment $f : E \to \mathbb{R}^+$ that satisfies the capacity and conservation conditions. Develop an efficient algorithm to determine if f is indeed a maximum flow.

Problem 2 (15 points) Given a flow network G, suppose f is flow with positive value, i.e., $\nu(f) > 0$. Consider the subgraph G' of G that only consists of edges that carry positive flow, i.e., an edge e is in G' if and only if $f(e) > 0$. Prove that there must be a simple s-t path in G'. *Note:* G' is not the residual graph.

Problem 3 (30 points) In this problem, we will consider how to update the maximum flow upon small changes to edge capacities. Let $G = (V, E)$ be a flow network with integer capacities. Suppose you are given the maximim flow in G.

(a) (10 points) If you increase the capacity of an edge (u, v) by 1, give an efficient algorithm to update the maximum flow.

(b) (20 points) If you decrease the capacity of an edge (u, v) by 1, give an efficient algorithm to update the maximum flow. *Hint:* This problem requires careful consideration of the possibilities, e.g., the maximum flow could be the same, the value of the maximum flow could be the same, or the maximum flow may decrease in value.

Problem 4 (40 points) A flow network has m edges. Show that there exists a sequence of at most m augmenting paths that yield the maximum flow. In other words, if you prove this statement, you would have shown that if the augmenting paths were chosen correctly (by magic), the Ford-Fulkerson algorithm will terminate in at most m iterations. Note that this question is not algorithmic: you do not have to demonstrate an algorithm to compute these augmenting paths. *Hint: Assume that you know the maximum flow and use Problem 2.*