CS 584/684 Algorithm Design and Analysis

Homework 6

Portland State U, Winter 2021 — *02/18/21*

Lecturer: Fang Song — *Due: 03/02/21*

Instructions. This problem set contains 6 pages (including this cover page) and 8 questions. A random subset of problems will be graded.

- Your solutions will be graded on *correctness* and *clarity*. You should only submit work that you believe to be correct, and you will get significantly more partial credit if you clearly identify the gap(s) in your solution. It is good practice to start any long solution with an informal (but accurate) summary that describes the main idea. You may opt for the "I take 15%" option.
- You need to submit a PDF file before the deadline. Either a clear scan of you handwriting or a typeset document is accepted. You will get 5 bonus points for typing in LaTeX (Download and use the accompany TeX file).
- You may collaborate with others on this problem set. However, you must **write up your own solutions** and **list your collaborators and any external sources** for each problem. Be ready to explain your solutions orally to a course staff if asked.
- For problems that require you to provide an algorithm, you must give a precise description of the algorithm, together with a proof of correctness and an analysis of its running time. You may use algorithms from class as subroutines. You may also use any facts that we proved in class or from the book.
- **If you describe a Greedy algorithm, you will get no credit without a formal proof of correctness, even if your algorithm is correct.**

Exercises. Do not turn in.

1. Let f and f' be feasible (s,t)-flows in a flow network G, such that $v(f') > v(f)$. Prove that there is a feasible (s,t)-flow with value $v(f') - v(f)$ in the residual network G_f.
2. Let $u \to v$ be an arbitrary edge in an arbitrary flow network G. Prove that if there is a minimum (s,t)-cut (S,T) such that $u \in S$ and $v \in T$, then there is *no* minimum cut (S',T') such that $u \in T'$ and $v \in S'$.
3. Describe a linear program for the bipartite maximum matching problem. The input is a bipartite graph $G = (U \cup V; E)$, where $E \subseteq U \times V$; the output is the largest matching in G. Your linear program should have one variable for each edge.

Problems to turn in.

1. (10 points) (Stabbing points) Let X be a set of n intervals on the real line. We say that a set P of points *stabs* X if every interval in X contains at least one point in P. Describe an efficient algorithm to compute the smallest set of points that stabs X. Assume that your input consists of two arrays $L[1, \ldots, n]$ and $R[1, \ldots, n]$, representing the left and right endpoints of the intervals in X.

2. (10 points) (Maximum spanning tree) Describe an algorithm to compute the *maximum*-weight spanning tree of a given edge-weighted graph.

3. (10 points) Suppose we are maintaining a data structure under a series of n operations. Let $f(i)$ denote the actual running time of the ith operation. Suppose that $f(i)$ is the largest integer k such that 2^k divides i. Determine the resulting amortized cost of a single operation.

4. (15 points) (Vertex cover) A *vertex cover* of an undirected graph $G = (V, E)$ is a subset of the vertices which touches every edge—that is, a subset $S \subseteq V$ such that for each edge $u, v \in E$, one or both of u, v are in S. Describe and analyze an algorithm, as efficient as you can, to find a minimum vertex cover in a bipartite graph.

5. (Updating max flow) You are given a flow network $G = (V, E)$ with source s and sink t, and integer capacities.

   (a) (10 points) Suppose that you are given a max flow in G. Now we increase the capacity of a single edge $(u, v) \in E$ by 1. Give an $O(m + n)$-time algorithm to update the max flow.

   (b) (10 points) Now suppose all edges have unit capacity and you are given a parameter k. The goal is to delete k edges so as to reduce the maximum $s - t$ flow in G as much as possible. In other words, you should find a set of edges $F \subseteq E$ so that $|F| = k$ and the maximum $s - t$ flow in $G' = (V, E - F)$ is as small as possible subject to this. Describe a polynomial-time algorithm to solve this problem.