Homework #7
Introduction to Algorithms
601.433/633
Spring 2020

Due on: Wednesday, April 29th, 12pm

Problem 1 (15 points)

For a directed graph $G = (V, E)$ (source and sink in V denoted by s and t respectively) with capacities $c : E \to \mathbb{R}^+$, and a flow $f : E \to \mathbb{R}^+$, the *support* of the flow f on G is the set of edges $E_{\textsf{supp}} := \{e \in E \mid f(e) > 0\}$, i.e. the edges on which the flow function is positive.

Show that for any directed graph $G = (V, E)$ with non-negative capacities $c : E \to \mathbb{R}^+$ there always exists a maximum flow $f^* : E \to \mathbb{R}^+$ whose *support* has no directed cycle.

Hint: Proof by contradiction?

Problem 2 (20 points)

In a graph $G = (V, E)$, a matching is a subset of the edges $M \subseteq E$ such that no two edges in M share an end-point (i.e. incident on the same vertex).

Write a linear program that, given a bipartite graph $G = (V_1, V_2, E)$, solves the maximum-bipartite-matching problem. I.e. The LP, when solved, should find the largest possible matching on the graph G.

Clearly mention the variables, constraints and the objective function. Prove why the solution to your LP solves the maximum-bipartite-matching problem. You may assume that all variables in the solution to your LP has integer values.

Problem 3 (15 points)

In class you saw that VERTEX-COVER and INDEPENDENT-SET are related problems. Specifically, a graph $G = (V, E)$ has a vertex-cover of size k *if and only if* it has an independent-set of size $|V| - k$.

We also know that there is a polynomial-time 2-approximation algorithm for VERTEX-COVER. Does this relationship imply that there is a polynomial-time approximation algorithm with a constant approximation ratio for INDEPENDENT-SET? Justify your answer.

Problem 4 (10 points)

You are given a biased coin c but you dont know what the bias is. I.e. the coin c outputs *H* with probability $p \in (0, 1)$ (note that it is not inclusive of 0 and 1) and *T* with probability $1 - p$ every time you toss it but you don't know what p is.

Can you simulate a fair coin using by tossing c? I.e. Come up with a "rule" for tossing c (possible several times) and outputting *H* or *T* based on the output of the coin tosses of c such that you will output *H* with probability $1/2$ and *T* with probability $1/2$? Please prove the correctness of your solution.

Hint: i) What if you had two coins c_1, c_2 with the same bias instead of one? Can you toss them together and output *H* or *T* based on the output? ii) You shouldn't be trying to "estimate" p.