HW 9 (due Monday, Noon, April 20, 2015)

OLD CS 473: Fundamental Algorithms, Spring 2015 Version: **1.02**

Collaboration Policy: For this homework, Problems 1–2 can be worked in groups of up to three students.

---

1. (40 PTS.) Min cost edge disjoint paths.

The input is a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ with n vertices and m edges, and an integer parameter $k > 0$. The graph G has prices $\text{cost}(\cdot)$ on the edges, which are positive integer numbers, and our purpose here is to compute a minimum-cost set of k edge disjoint paths from s to t in G. The price of the solution Π, is the total cost of the edges of the paths of Π. This can be easily solved using min-cost flow, but our purpose here is to develop an direct algorithm for this purpose.

Here, we are assuming that there is no pair of vertices u and v in G such that both (u, v) and (v, u) are in G.

(A) (5 PTS.) Describe an algorithm, as fast as possible, that computes a set Π_0 of k edge disjoint paths from s to t in G, if such a set of paths exists. What is the running time of your algorithm?

(B) (5 PTS.) Given a set Π of k edge disjoint paths from s to t in G, let G_Π be the ***leftover graph*** $\mathsf{G}_\Pi = (\mathsf{V}, \mathsf{E}_\Pi)$, where $\mathsf{E}_\Pi = \left\{ (u,v) \;\middle|\; (u,v) \in \mathsf{E}(\mathsf{G}) \setminus \mathsf{E}(\Pi) \text{ or } (v,u) \in \Pi \right\}$.

Prove that there is a set Π^{-1} of k edge disjoint paths from t to s in G_Π, and describe how to compute it given Π.

(C) (5 PTS.) Consider two sets Π and Σ of k edge disjoint paths from s to t in G. Consider the set of paths Σ^{-1} (from part (B)), and consider the set of edges $X = \mathsf{E}(\Pi) \cup \mathsf{E}(\Sigma^{-1})$. As long as there is an edge $(u, v) \in X$ such that (v, u) is also in X, we remove both of these edges (i.e., (u, v) and (v, u)) from X (i.e., we are removing all cycles of length 2 from X). Let Y be the resulting set of edges, and we denote this set of edges by $\mathsf{E}(\Pi \oplus \Sigma^{-1})$

Prove that all the edges of Y appear in the graph G_Σ (note, that this is not true for X!).

(D) (5 PTS.) Continuing part (C), consider the graph $\mathsf{H}(\Pi \oplus \Sigma^{-1}) = (\mathsf{V}, Y)$. Prove that this graph can be represented as the union of edge disjoint simple cycles.

(E) (5 PTS.) Consider the following pricing on the edges of G_Σ. For any edge $(u, v) \in \mathsf{E}(\mathsf{G}_\Sigma)$, if it appears in G, then its cost $\text{cost}((u, v))$ is its original cost. However, if $(u, v) \in \mathsf{E}(\mathsf{G}_\Sigma)$ and $(v, u) \in \mathsf{E}(\mathsf{G})$ we set $\text{cost}((u, v)) = -\text{cost}((v, u))$.

Prove that $\text{cost}(\Pi) - \text{cost}(\Sigma) = \text{cost}(\mathsf{E}(\Pi \oplus \Sigma^{-1}))$. Here $\text{cost}(\Pi) = \text{cost}(\mathsf{E}(\Pi)) = \sum_{e \in \mathsf{E}(\Pi)} \text{cost}(e)$.

(F) (5 PTS.) Consider a set Σ of k edge disjoint paths in G from s to t, and assume that there is a cheaper set of k such paths. Prove, that the graph G_Σ contains a simple cycle that the total cost on its edges is negative (such a cycle is a ***negative cycle***). (Hint: Use (E)).

Describe an efficient algorithm for computing this negative cycle.

(G) (5 PTS.) Consider a set Σ of k edge disjoint paths in G, and a negative cycle C in G_Σ. Present an algorithm, as fast as possible, for computing a new set of k edge disjoint paths Π' in G that has cost equal to (or smaller than) $\text{cost}(\Sigma) + \text{cost}(C)$.

(H) (5 PTS.) Let W be the maximum cost of an edge in G. Describe an algorithm, as fast as possible, with running time polynomial in n, k and W, that computes the min-cost set of k edge-disjoint paths from s to t in G. What is the running time of your algorithm. Prove that your algorithm computes the optimal solution.

2. (60 PTS.) We want a proof!

The following question is long, but not very hard, and is intended to make sure you understand the following problems, and the basic concepts needed for proving NP-Completeness.

All graphs in the following have n vertices and m edges.

For each of the following problems, you are given an instance of the problem of size n. Imagine that the answer to this given instance is "yes", and that you need to convince somebody that indeed the answer to

the given instance is `yes`. To this end, describe:

(I) An algorithm for solving the given instance (not necessarily efficient). What is the running time of your algorithm?

(II) The format of the proof that the instance is correct.

(III) A bound on the length of the proof (its have to be of polynomial length in the input size).

(IV) An efficient algorithm (as fast as possible [it has to be polynomial time]) for verifying, given the instance and the proof, that indeed the given instance is indeed `yes`. What is the running time of your algorithm?

We solve the first such question as an example.

(A) (0 PTS.)

Shortest Path

Instance: A weighted undirected graph G, vertices s and t and a threshold w.
Question: Is there a path between s and t in G of length at most w?

Solution:

(I) We seen in class the Dijkstra algorithm for solving the shortest path problem in $O(n \log n + m) = O(n^2)$ time. Given the shortest path, we can just compare its price to w, and return yes/no accordingly.

(II) A "proof" in this case would be a path π in G (i.e., a sequence of at most n vertices) connecting s to t, such that its total weight is at most w.

(III) The proof here is a list of $O(n)$ vertices, and can be encoded as a list of $O(n)$ integers. As such, its length is $O(n)$.

(IV) The verification algorithm for the given solution/proof, would verify that all the edges in the path are indeed in the graph, the path starts at s and ends at t, and that the total weight of the edges of the path is at most w. The proof has length $O(n)$ in this case, and the verification algorithm runs in $O(n^2)$ time, if we assume the graph is given to us using adjacency lists representation.

(B) (5 PTS.)

Independent Set

Instance: A graph G, integer k
Question: Is there an independent set in G of size k?

(C) (5 PTS.)

3Colorable

Instance: A graph G.
Question: Is there a coloring of G using three colors?

(D) (5 PTS.)

TSP

Instance: A weighted undirected graph G, and a threshold w.
Question: Is there a TSP tour of G of weight at most w?

(E) (5 PTS.)

Vertex Cover

Instance: A graph G, integer k
Question: Is there a vertex cover in G of size k?

(F) (5 PTS.)

Subset Sum

Instance: S - set of positive integers,t: - an integer number (target).
Question: Is there a subset $X \subseteq S$ such that $\sum_{x \in X} x = t$?

(G) (5 PTS.)

3DM

Instance: X, Y, Z sets of n elements, and T a set of triples, such that $(a, b, c) \in T \subseteq X \times Y \times Z$.
Question: Is there a subset $S \subseteq T$ of n disjoint triples, s.t. every element of $X \cup Y \cup Z$ is covered exactly once.?

(H) (5 PTS.)

Partition

Instance: A set S of n numbers.
Question: Is there a subset $T \subseteq S$ s.t. $\sum_{t \in T} t = \sum_{s \in S \setminus T} s$?

(I) (5 PTS.)

SET COVER

Instance: $(X, \mathcal{F}, k)$:
X: A set of n elements
$\mathcal{F}$: A family of m subsets of S, s.t. $\bigcup_{X \in \mathcal{F}} X = X$.
k: A positive integer.
Question: Are there k sets $S_1, \ldots, S_k \in \mathcal{F}$ that cover S. Formally, $\bigcup_i S_i = X$?

(J) (5 PTS.)

CYCLE HATER.

Instance: An undirected graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, and an integer $k > 0$.
Question: Is there a subset $X \subseteq \mathsf{V}$ of at most k vertices, such that all cycles in G contain at least one vertices of X.

(K) (5 PTS.)

CYCLE LOVER.

Instance: An undirected graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, and an integer $k > 0$.
Question: Is there a subset $X \subseteq \mathsf{V}$ of at most k vertices, such that all cycles in G contain at least two vertices of X.

(L) (5 PTS.)

Hamiltonian Cycle.

Instance: An undirected graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$.
Question: Is there a simple cycle that visits all the vertices of G?

(M) (5 PTS.)

Many Spiders.

Instance: An undirected graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ and an integer k.
Question: Are there k vertex-disjoint spiders that visits all the vertices of G?

A ***spider*** in a graph G is defined by a vertex v (i.e., the head of the spider), and a collection Π of simple paths all starting in v, that except for v, are vertex disjoint. The vertex set of such a spider is all the vertices that the paths of Π visit.