Assignment II for AI2615 (Spring 2022)

March 20, 2022

Problem 1 (20 Points)

Given a directed graph $G = (V, E)$ where each vertex can be viewed as a port. Consider that you are a salesman, and you plan to travel the graph. Whenever you reach a port v, it earns you a profit of p_v dollars, and it cost you c_{uv} if you travel from u to v. For any directed cycle in the graph, we can define a profit-to-cost ratio to be

$$r(C) = \frac{\sum_{(u,v)\in C} p_v}{\sum_{(u,v)\in C} c_{uv}}.$$

As a salesman, you want to design an algorithm to find the best cycle to travel with the largest profit-to-cost ratio. Let r^* be the maximum profit-to-cost ratio in the graph.

1. (10 points) If we guess a ratio r, can we determine whether $r^* > r$ or $r^* < r$ efficiently?

   Hint: Construct another graph with appropriate edge lengths and apply the shortest path algorithm.

2. (10 points) Based on the guessing approach, given a desired accuracy $\epsilon > 0$, design an efficient algorithm to output a good-enough cycle, where $r(C) \geq r^* - \epsilon$. Justify the correctness and analyze the running time in terms of $|V|$, ϵ. In the analysis, we can assume $R = \max_{(u,v)\in E}(p_u/c_{uv})$ is a constant.

Problem 2 (20 Points)

An Eulerian path in a directed graph is a path containing each edge exactly once. An Eulerian circuit in a directed graph is a Eulerian path that is a cycle. In the following, let $G = (V, E)$ be a *strongly connected* directed graph.

The in-degree of a vertex in G is the number of incoming edges: $\texttt{deg}_{\texttt{in}}(v) = |\{u \in V : (u, v) \in E\}|$. The out-degree is the number of outgoing edges: $\texttt{deg}_{\texttt{out}}(v) = |\{u \in V : (v, u) \in E\}|$

1. (10 points) Prove that G contains an Eulerian circuits if and only if the in-degree and out-degree of each vertex $v \in V$ are the same. What is the sufficient and necessary condition for the existence of an Eulerian path?

2. (10 points) Give an $O(|E|)$ time algorithm to find an Eulerian circuit in case it exists. You need to clearly specify how your algorithm is implemented using appropriate data structure and prove its time complexity.

Problem 3 (30 Points)

Let $G = (V, E)$ be an undirected connected graph. Let T be a depth-first search tree of G. Suppose that we orient the edges of G as follows:

For each tree edge, the direction is from the parent to the child; for every non-tree (back) edge, the direction is from the descendant to the ancestor. Let G' denote the resulting directed graph.

1. (5 points) Give an example to show that G' is not strongly connected.
2. (5 porints) Prove that if G' is strongly connected, then G satisfies the property that removing any single edge from G will still give a connected graph.
3. (5 points) Prove that if G satisfies the property that removing any single edge from G will still give a connected graph, then G' must be strongly connected.
4. (15 points) Give an $O(|V|+|E|)$ algorithm to find all edges in a given undirected graph such that removing any one of them will make the graph no longer connected.

Problem 4 (30 Points)

We have seen in the class that Dijkstra algorithm fails if the graph contains negatively-weighted edges. Consider the following variant of Dijkstra algorithm. Given a directed weighted graph $G = (V, E, w)$ where $w(u, v)$ may be negative, find an integer W such that $w(u, v) + W > 0$ for each edge $(u, v) \in E$, and define the new weight of (u, v) as $w'(u, v) = w(u, v) + W$. Now, $G' = (V, E, w')$ is a positively weighted graph where the weight of each edge is increased by W. Implement Dijkstra algorithm on $G' = (V, E, w')$, and a shortest path from s to every vertex u in G' is also a shortest path in G.

1. (15 points) Does this algorithm work for directed acyclic graphs? If so, prove it; if not, provide a counterexample.
2. (15 points) A *directed grid* is a directed weighted graphs $G = (V, E, w)$ where the set of vertices is given by $V = \left\{ v_{ij} \,\middle|\, i = 1, \ldots, m; j = 1, \ldots, n \right\}$ ($m, n \in \mathbb{Z}^+$ are two parameters) and the set of directed edges is given by

$$E = \left(\bigcup_{i=1,\ldots,m-1; j=1,\ldots,n} \left\{ (v_{ij}, v_{(i+1)j}) \right\} \right) \cup \left(\bigcup_{i=1,\ldots,m; j=1,\ldots,n-1} \left\{ (v_{ij}, v_{i(j+1)}) \right\} \right).$$

   That is, the vertices form a $m \times n$ grid. There is a *directed* edge *from* every vertex *to* the vertex "right above" it, and there is a *directed* edge *from* every vertex *to* the vertex "to the right of" it (unless the vertex is "on the boundary"). The weight of each edge is specified as input and can be negative.

Does the algorithm work for directed grids? If so, prove it; if not, provide a counterexample.

Problem 5

How long does it take you to finish the assignment (including thinking and discussion)?

Give a rating (1,2,3,4,5) to the difficulty (the higher the more difficult) for each problem.

Do you have any collaborators? Please write down their names here.