CS535 Fall 2022 HW1 Sample Solutions

1. This problem can be solved using 2 parts. First, by finding out the shortest $s - t$ paths in a graph D. Second, by finding out the inclusion-wise maximal edge-disjoint shortest $s - t$ paths which we found in part 1.

   For the first part-1, we can apply BFS twice. One for path $s - t$ and by marking the nodes with the numbers that are equivalent to their levels. Other, by making another graph in a reverse direction doing BFS traversal from $t - s$.

   By adding the levels/numbers obtained in both the graph we can make a third concluding graph which shows us the all the shortest paths possible from $s - t$ (All the nodes which have same numbers/level after addition). Hence, we have now found a set of shortest paths from $s - t$.

   We can use a modified version of DFS to solve the second part by marking/coloring the vertices whenever we are traversing back from s. Whenever we encounter a vertex/node which is already marked/colored or t(in-reverse) we can add the edges of the $s-t$ path to a set while traversing back to s. The one's which have completed $s - t$ paths would be selected as the final paths. Run-time would be $O(m + n)$ for all the traversals.

2. First run Karp's algorithm on D in $O(nm)$ to get the minimum circuit mean μ.

   Now we create a new graph $D_2 = (V, A; l_2)$, where $l_2(a) = l(a) - \mu, \forall a \in A$. By doing this we make sure that there are no negative cycles in D_2, and each minimum circuit has a mean of 0.

   We add a pseudo source vertex s to D_2 and create a new edge $(s, v), \forall v \in V$ with the length $l_2(s, v) = 0$. This makes sure that every original vertex from D is reachable from source s, therefore we are able to define the adjusted edge length on every edge $a \in A$. The construction can be done in linear time.

   Now we run Bellman-Ford starting from s in $O(nm)$, to calculate the shortest distance for s to v, denoted by $d(v), \forall v \in V$. Using $d(v)$ as the potential function, the adjusted length will be non-negative. And since each minimum circuit has mean$= 0$, each edge in the minimum circuit must have the adjusted length to be exactly 0.

   Now if we apply $d(v)$ to the original graph D, we get all adjusted edge length $\geq \mu$, and those of edges in any minimum circuit to be exactly μ. The overall time complexity is $O(nm)$.

3. Firstly, prove **If** D has no $s-t$ path **Then** there exists a nonnegative integer-valued labeling p satisfying the two conditions. Since there is no $s - t$ path, we can construct a labeling p like the following: split all the vertices into two sets S and X, such that S contains all nodes that can be reached by s and X contains all nodes that can not be reached by s; label $p(v) = n$ for $v \in S$, and $p(v) = 0$ for $v \in X$. In this way, for edge $(u, v) \in S$, $p(u) - p(v) = n - n = 0 \leq 1$, for edge $(u, v) \in X$, $p(u) - p(v) = 0 - 0 = 0 \leq 1$. And we cannot find edge from set S to X since there is no $s - t$ path. Therefore, the two requirements have been met.

   Secondly, prove **If** there exists a nonnegative integer-valued labeling p satisfying the two conditions **Then** D has no $s - t$ path. Suppose there exist a $s - t$ path; Since $p(s) = n$, $p(t) = 0$, and for each $(u, v) \in A$, $p(u) - p(v) \leq 1$, any possible path from s to t has at least n length. However, the maximum possible $s - t$ path in D is $n - 1$, which is contradictory. Therefore, D has no $s - t$ path.

4. We know one edge (u, v) has a negative length. We can use Dijkstra's Algorithm with slight modifications to solve this.

   While traversing a node/vertex using Dijkstra's, check if it is possible that a visited neighbour is relaxed to negative value and break the loop as soon as you get your answer as True.

Now, since you have found a neighbour with negative value, we can say that a negative circuit is present. Else, it would be impossible to reach that node/vertex again.

As we are using Dijkstra's algorithm for this, worst-case time complexity would be $O(m + n \log n)$.

5. Let k be the number of iterations performed in the augmented graph. We denote by n_i the number of vertices deleted in the ith iteration, excluding the new vertex v. We claim that the number of edges deleted in iteration i is at least $n_i + 1$. If the claim holds, k iterations in total deletes at least $n + k$ edges, and the total number of edges in the augmented graph is at most $n + m$, therefore $k \leq m$.

   To prove the claim, we first observe that it is true when $n_i = 0$ as in one iteration at least one edge is removed. If $n_i > 0$, there have to be at least 1 incoming and 1 outgoing edge of each deleted vertex also deleted. Since n_i excludes vertex v, at least $n_i + 1$ edges are removed.

6. We create a new graph $D' = (V', A')$ by replacing each vertex $v \in V$ using two vertices v_0 and v_1, and each edge $(u, v) \in A$ is replaced by (u_0, v_1) and (u_1, v_0). D' has twice as many edges and vertices as the original graph, $|V'| = 2n$, $|A'| = 2m$. We run BFS on D' to find the shortest path from s_0 to t_0 and s_0 to t_1. We only need to ignore the subscript of each node and the remaining is the shortest s-t even/odd path in D. The time equals to the time complexity of BFS, which is $O(m + n)$.