Algorithms I

Tutorial 8 Solution Hints

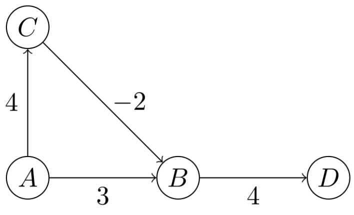
November 4, 2016

Problem 1

Instead of using a min-heap in each iteration of Prim's algorithm, we can do a linear search on the vertices to find such a vertex. So, each iteration now takes $O(V)$ time and there are V such iterations. Hence, the total time complexity is $O(V^2)$.

Problem 2

The following graph will give incorrect distances for B and D if A as the source vertex.



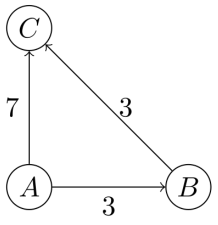
Problem 3

Assume array *dist* and array *parent* are the outputs of the professor's program. Also, let $n = |V|$ and $m = |E|$. We should check the following to check the correctness of the algorithm:

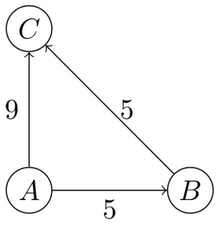
- For each edge from u to v, we should have $dist[u] + w(u, v) \geq dist[v]$. We can check this property in $O(m)$.
- For each vertex v except the source we should have $parent[v] \neq v$ and $dist[parent[v]] + w(parent[v], v) = dist[v]$ and for the source (s) we should have $parent[s] = null$. We can check this property in $O(n)$.
- $dist[s] = 0$, where s is the source. We can check this property in $O(1)$.
- We have to check whether array parent represents a spanning tree or not. In order to check this property we can run a DFS algorithm on the shortest- path tree (stored in array parent) from source s. At the end of the DFS algorithm all vertices should be visited. Note that we can check this prop- erty by one DFS algorithm on the shortest-path tree with at most $n1$ edges in $O(n)$.

Problem 4

The answer is no and here is a counter example. The shortest-path tree of this graph has edges (A, B) and (B, C).



For $c = 2$, the new graph would be the following. The shortest-path tree of this graph has edges (A, B) and (A, C)



Problem 5

Let $G = (V, E)$ be a weighted undirected graph. Let $s, t \in V$ and $s \neq t$. Design an $O(E \log V)$ algorithm to find all vertices v such that v lies on at least one of the shortest paths between s and t.

We find shortest distance of all the vertices from s and t. Now, a vertex v lies on at least one of the shortest path between s and t if $\mathit{mindist}(s, v) + \mathit{mindist}(t, v) = \mathit{mindist}(s, t)$.

Problem 6

We can use the same trick as in Problem 1.

Problem 7

We run the algorithm on the graph. After the completion, we check if any diagonal entry is negative. If yes, there is a negative weighted cycle, otherwise there is no such cycle.

Problem 8

We represent the graph in adjacency matrix M where $M_{i,j}$ is the cost of the edge (i, j) if the edge (i, j) is present in the graph, otherwise ∞. Thus, if there is no self loop, $M_{i,i} = \infty$ for all vertices i, otherwise there is a cycle of length 1. Now, we run Floyd-Warshall algorithm. After i^{th} iteration of the algorithm, we check if $M_{j,j} \neq \infty$ for j. If we find such a j, we have found a cycle of length $i + 1$ which contains vertex j. *Note*: This algorithm works only for directed graph.

Problem 9

If in the case where $d_{ij}^{(k-1)} = d_{ik}^{(k-1)} + d_{kj}^{(k-1)}$, we set $\pi_{ij}^{(k)}$ to be $\pi_{kj}^{(k-1)}$, instead of $\pi_{ij}^{(k-1)}$, the predecessor matrix will still be correct. This is because in case of equality it doesn't matter which of the two paths will be taken, as the weight of both paths is the same. Essentially it means we would go from i to j through k instead of not through k, but in this case both are valid shortest paths, meaning the predecessor matrix would still hold shortest paths.