Part II: Applications of Network Models (Spring 2021) **Homework: 04**

Lecture 1: Shortest Path Problem

Lecturer: Junlong Zhang zhangjunlong@mail.tsinghua.edu.cn

Student: Zhenyu Jin jzy20@mails.tsinghua.edu.cn

P1.

Give the conditions under which Dijkstra algorithm can work. Prove that under these conditions Dijkstra algorithm is correct, that is, we can indeed obtain the shortest paths by using Dijkstra algorithm.

Solution:

- The condition that Dijkstra algorithm can work is non negative weight edge exists in the network.
- Prove:
  Given the condition above, then the distance of point to original must be increasing. According to the formulation given below, $d(i) = mind(i), d(j) + l(i, j)$, which point j in the linking set of i. Suppose that we have already sort the points in the set, except origin point, the point are n_0, n_1,$\cdots$, n_k, i.e., $d(n_0) \leq d(n_1) \leq \cdots \leq d(n_m)$. Then we have that $d(n_i) = minl(s, n_i), min_{j<i} d(n_j) + l(n_i, n_j)$.

P2.

Give the conditions under which Floyd algorithm can work. Prove that under these conditions Floyd algorithm is correct, that is, we can indeed obtain the shortest paths by using Floyd algorithm.

Solution:

- The condition that Floyd algorithm can work include: the network does not contain negative weight cycle.
- Prove:
  The Floyd alogrithm can be analogied as dynamic programming, that is, if the shortest path between point u and v pass through the point w, then the path of u to w and w to v should be shortest too.
  Let G be a graph with numbered vertices 1 to N. In the k^{th} step, let shortestPath(i, j, k) be a function that yields the shortest path from i to j that only uses nodes from the set $1, 2, \cdots, k$. In the next step, the algorithm will then have to find the shortest paths between all pairs i,j using only the vertices from $1, 2, \ldots, k, k + 1$.
  This is the idea of dynamic programming. In each iteration, all pairs of nodes are assigned the cost for the shortest path found so far:

  $$d(i, j, k) = mind(i, j, k), d(i, k + 1, k) + d(k + 1, j, k)$$

  Hence, that Floyd algorithm can work under the condition that the network has no negative cycle.

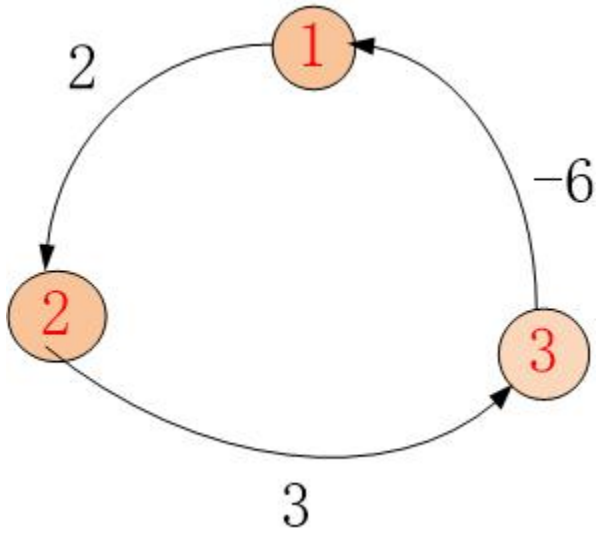

Figure 1: Counterexample of Floyd alogrithm

P3.

Give a counterexample for which Floyd algorithm does not work.

Solution:

- The counterexample is given below, since there exsits a negative weight cycle in the network, then the iteration would never end since the total distance could be decrease by each iteration.

P4.

Analyze the time complexity of Dijkstra algorithm and Bellman-Ford algorithm.

Solution:

- The time complexity of Dijkstra algorithm is $O(V^2)$. As depicated in the book *IntroductiontoAlgorithms*, Dijstra can be expressed as,

```
DIJKSTRA(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  S ← Ø
3  Q ← V[G]        //V*O(1)
4  while Q ≠ Ø
5     do u ← EXTRACT-MIN(Q)        // V*O(V), V*O(lg V)
6        S ← S∪ {u}
7        for each vertex v∈ Adj[u]
```

```
8           do RELAX(u, v, w)
```

- The time complexity of Bellman-Ford algorithm is $O(VE)$. As depicated in the book $Introduction to Algorithms$, Bellman-Ford can be expressed as,

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i 1 to V[G] - 1        //θ(E)
3     do for each edge (u, v)
4        do RELAX(u, v, w)        //(V-1)θ(E)
5  for each edge (u, v)
6     do if d[v] > d[u] + w(u, v)
7        then return FALSE        // O(E)
8  return TRUE
```