Data Structure and Algorithms
--HW4

Name: Xiaoyi Tang
NetId: xt59

Q1:

- Code:
  - “Graph.java”
- Answer:
  - from the result we know that the graph provided is not acyclic.

Q2:

- Code:
  - “PRIM.java”(for prim’s algorithm)
  - “MST.java”(for kruskal’s algorithm)
- Answer:
  - I implemented the two algorithms and tested the running time of them separately.
  - The running time for Kruskal’s algorithm is 94245950 (average value for 5 times) nanotime and the time for Prim’s algorithm is 72960369 (average value for 5 times).
  - Therefore, it is obvious that the performance of prim’s algorithm is better than kruskal’s algorithm.

Q3:

1. Shortest Path:
   I will take the case (start from 0) as an instance:

V	distTo[v]	edgeTo[v]
0	0.0	-
1		
2	0.26	0 -> 2
3		
4		
5		

6		
7		

1st: choose source vertex 0
2nd: relax all edges incident from 0
3rd: relax all edges incident from 2, we can find that there is no edge incident from 2.

For the source vertex is not specified in this question, so I calculated all shortest path to all vertex from 0 – 7 (shown as below):

	0	1	2	3	4	5	6	7
0	-	-	0.26	-	-	-	-	-
1	1.39	-	1.02	0.29	1.74	-	0.81	0.68
2	-	-	-	-	-	-	-	-
3	1.10	-	0.73	-	1.45	-	0.52	0.39
4	0.38	-	0.64	-	-	-	-	0.37
5	0.61	0.32	0.62	0.61	0.35	-	1.13	0.28
6	0.58	-	0.40	-	0.93	-	-	1.30
7	-	-	0.34	-	-	-	-	-

2. Longest path:
   For the longest path, the solution is similar as in last question.

	0	1	2	3	4	5	6	7
0	-	-	0.26	-	-	-	-	-
1	2.12	-	2.45	0.29	1.74	-	0.81	2.11
2	-	-	-	-	-	-	-	-
3	1.83	-	2.16	-	1.45	-	0.52	1.82
4	0.38	-	0.71	-	-	-	-	0.37
5	2.44	0.32	2.77	0.61	2.06	-	1.13	2.43
6	1.31	-	1.64	-	0.93	-	-	1.30
7	-	-	0.34	-	-	-	-	-

Q4:

(a).
in the first pass:

v	distTo[v]	edgeTo[v]
0	0	-
1	1.05	5 -> 1

2	0.26	0 -> 2
3	0.99	7 -> 3
4	0.26	6 -> 4
5	0.73	4 -> 5
6	1.51	3 -> 6
7	0.60	2 -> 7

In the 2nd pass:

v	distTo[v]	edgeTo[v]
0	0	-
1	0.93	5 -> 1
2	0.26	0 -> 2
3	0.99	7 -> 3
4	0.26	6 -> 4
5	0.61	4 -> 5
6	1.51	3 -> 6
7	0.60	2 -> 7

Pass 3, 4, 5,… has no further changes

(b).
in the 1st pass:

v	distTo[v]	edgeTo[v]
0	0	-
1	1.05	5 -> 1
2	0.26	0 -> 2
3	0.99	7 -> 3
4	0.07	5 -> 4
5	0.73	4 -> 5
6	1.51	3 -> 6
7	0.60	2 -> 7

In the 2nd pass:

v	distTo[v]	edgeTo[v]
0	0	-
1	0.74	5 -> 1
2	0.26	0 -> 2
3	0.83	7 -> 3
4	-0.59	5 -> 4
5	0.42	4 -> 5
6	1.35	3 -> 6
7	0.44	4 -> 7

From the result of 2nd pass we can find that any vertex v is updated in phase V, therefore, there exists a negative cycle in this graph.

Q5.

- Code:
  - “BFS.java”
  - “dfs.java”

Q6.

- Code:
  - “Dijkstra.java”
- Answer:
  - I implemented the Dijkstra’s algorithm to get the shortest path among vertices.
  - Comparing to the result from 4(a) and 4(b), we can find that the results from the code are both wrong. I think the reason is that Dijkstra’s algorithm does not work with negative edge weights. But the data from 4(a) and 4(b) both has negative weights.