**ECE 573 HW4 Report**

Zhongze Tang (zt67)

**Q1**

The graph is cyclic, which means it contains at least one circle in it.

The idea of testing whether a graph is acyclic or not is simple. Just do the DFS for all the vertices in the graph. And if the next vertex we are going to visit has been visited, at the same time, this vertex is not where we come from (I call it “parent”), there must be a circle in the graph.

In the codes, what is different from original DFS is that, when we call the DFS function, we also add the vertex v’s parent vertex w to the parameters list. If a vertex u from G.adj(v) has been visited, and it is not equal to w, there must be a circle, and the graph is cyclic.

**Q2**

Both the lazy and eager Prim’s algorithms are faster than Kruskal’s. Run the program five times and calculate the average running time, we can get a table like this:

|  |  |  |
| --- | --- | --- |
| Algorithm | Running Time (ns) | Time complexity |
| Kruskal | 7222169 | ElogE |
| Lazy Prim | 2581351.8 | ElogE |
| eager Prim | 1344047 | ElogV |

The order of growth of the actual cost of Kruskal is E + E0 log E, where E0 is the number of edges whose weight is less than the weight of the MST edge with the highest weight. Despite this advantage, Kruskal’s algorithm is generally slower than Prim’s algorithm because it has to do a connected() operation for each edge, in addition to the priority-queue operations that both algorithms do for each edge processed

In this problem, the graph is a dense one (V < E), so the eager Prim performs better than other two algorithms.

**Q5**

BFS will run properly on the dataset. However, if we use the simple recursive version of DFS, the stack overflow will occur. To solve this, we must maintain a stack by ourselves.

First, we push the first vertex into the stack and mark it as visited. Second, when the stack is not empty, repeat the third step and the forth step. Third, we pop the first vertex in the stack. Last, we visit all the adjacent vertices of this vertex, and push the unvisited vertices into the stack, then mark them as visited.

Both DFS and BFS will print out how may vertices they visit. The value should be 264346.

**Q6**