**Algorithm HW9**

Approach 1:

1. Build a Graph class to maintain adjacency lists.

abstract class Graph {

private int V;

private AdjacencyList<Integer, ArrayList<Integer>> adj;

Graph(int[][] edges); // constructor to construct the adjacency list with O(n) time.

void addEdge(int v, int w); // O(1)

ArrayList<Integer> adj(int v); // O(1)

int V();

}

abstract class Graph {

private int V;

private AdjacencyList<Integer, ArrayList<Integer>> adj;

Graph(int[][] edges); // constructor to construct the adjacency list with O(n) time.

void addEdge(int v, int w); // O(1)

ArrayList<Integer> adj(int v); // O(1)

int V();

}

1. Preprocess the graph to recognize the connectivity and divide into different components, so build a ConnectedComponent class to find connected components with DFS.

abstract class ConnectedComponent {

private ArrayList<Boolean> marked; // space complexity: O(V)

private ArrayList<Integer> id; // space complexity: O(V)

private int count;

ConnectedComponent();

boolean connected(int v, int w); // time complexity: O(1)

void dfs(Graph G, int v); // time complexity: O(E)

}

1. Create ShortestDistance class to find the shortest distance between two vertices.

abstract class ShortestDistance {

private ArrayList<Boolean> marked;

private ArrayList<Integer> distTo;

ShortestDistance();

void bfs(Graph G, int s);

int distTo(int v);

}

Find the largest shortest distance between two vertices in a graph by doing the following code.

For choosing a vertex in the outer loop {

For choosing another vertex in the inner loop {

If (they are connected) {

find the shortest distance between these two vertices in the component

}

}

}

O(V^2)

**Self-defined classes:**

/\* Data Structure Adjacency List to store a vertex index and its corresponding adjacency list. \*/

class IndexAdjacentVertices<E> {

E index;

ArrayList<E> adjList = new ArrayList<E>();

IndexAdjacentVertices(E index) {

this.index = index;

}

E getIndex() {

return index;

}

ArrayList<E> getAdjVertices() {

return adjList;

}

void addAdjVertex(E adjVertex) {

adjList.add(adjVertex);

}

}

class AdjacencyList<E> extends ArrayList<IndexAdjacentVertices<E>> {

AdjacencyList<E> adjLists;

AdjacencyList() {

adjLists = new AdjacencyList<E>();

}

boolean containsKey(E key) {

for (IndexAdjacentVertices<E> adjList: this.adjLists) {

if (key == adjList.getIndex()) {

return true;

}

}

return false;

}

void addAdjVertexAtIndex(E index, E adjVertex) {

}

}