Program 5 pseudocode

Unweighted breadth-first search

Queue<Vertex> q;

for each Vertex v

v.dist = INFINITY;

startingVertex.dist = 0;

q.enqueue(startingVertex);

while (! q.isEmpty() )

Vertex v = q.dequeue();

for each Vertex w adjacent to v

if (w.dist == INFINITY) // if w is not yet found

w.dist = v.dist + 1;

w.path = v;

q.enqueue(w);

**dijkstra’s algorithm is on next page**

dijkstra’s algorithm with priority queue

structs needed:

PathVertex: bool found, int totalDistance, int prevVertex

QueueVertex: int fromVertex, int toVertex, int totalPathCost

Edge: int toIndex, int cost

// follows “basically the BFS algorithm, but with a priority queue and using costs

// priority will be based on total path cost

void shortestPath(int startingVertex, adjList) {

std::priority\_queue<QueueVertex> q;

create pathRepresentation as an array of PathVertex objects

for each PathVertex v in pathRepresentation //could be done in constructor for PathVertex?

totalDistance = max\_int;

found = false;

prevVertex = max\_int;

startingVertex.totalDistance = 0;

startingVertex.prevVertex = -1;

startingVertex.found = true;

numFound = 1;

for each Edge curAdjVertex adjacent to startingVertex

create QueueVertex object with

fromVertex = startingVertex,

toVertex = curAdjacentVertex.destination,

totalCost as startingVertex.totalDistance + curAdjacentVertex.edgeCost

enqueue(QueueVertex object)

while (queue is not empty and current size of found vertices != numVertices )

QueueVertex curQueueVertex = q.dequeue();

if curQueueVertex.toVertex has not been found yet

add to found and update pathRepresentation

increment numFound;

for each Edge curAdjVertex adjacent to curQueueVertex.toVertex

if curAdjVertex.destination is not found

\*\*\*create QueueVertex object with:

fromVertex = curQueueVertex.toVertex,

toVertex = curAdjacentVertex.destination,

totalCost as curQueueVertex.totalPathCost + curAdjacentVertex.edgeCost\*\*\*

enqueue QueueVertexObject