Jordan Allard
CSCI 264-01
Homework 5

HW5 Problem 2 Writeup

a) Verbal Description

This algorithm calculates the smallest number of edges required to turn any given unconnected graph in to a connected one by using a depth-first search.

b) Pseudocode

```
DFS(v):
      visited[v] = true
      For k = 1 to adjList[v].size():
            If visited[adjList[v][k] == false]:
                  DFS(v)
DFS-Main():
      Given a graph with n vertices and m edges:
      Create an adjacency list adjList
      For i = 1 to n: visited[i] = false
      edges = -1
      For i = 1 to n:
            If visited[i] == false:
                  edges += 1
                  DFS(i)
      Return edges
```

c) Proof of Correctness

This algorithm relies on the idea that a depth-first search of a graph, when starting from a certain point, will eventually traverse all the points connected to that initial point. This is because DFS is a recursive algorithm that works by exploring all the points accessible from the previous point, which guarantees that it will visit every possible point conncted to the initial point. If any points remain unvisited after running DFS, we know those points must not be reachable from the first set of points, and that in order to traverse the other points we must run DFS again with an unvisited starting point. Thus, the algorithm can determine the total number of edges required to connect the vertices by simply adding an edge each time DFS is required to pick a new starting point. The for loops within DFS ensure that every vertex will be checked, whether as a part of the DFS chain or as a regular bystander.

d) Running Time Estimate

O(n + m)

e) Running Time Estimate Reasoning

The running time of this algorithm can be broken down into sections:

Building the adjacency list --> O(n) + O(m)
Initializing the adjacency list --> O(n)
DFS --> O(n + m)
Everything else --> O(1)

About the DFS estimate:

To complete DFS, the algorithm is essentially iterating once through every element of the adjacency list, which has a running time of O(deg(v)) for any vertex v. The sum of all the possible degrees of each vertex in a graph is the same as the number of edges in the graph, m. Thus the sum O(deg(v)), repeated for n vertices, becomes O(n + m).

Note: Since this graph is undirected, each edge is represented twice in the adjacentcy list, once for each point it's connected to, so the true cost of iterating through the adjacency list is actually O(n + 2m), which is still linear.

If the graph is completley connected, DFS will include every point in the graph and will have a running time of O(n + m).

If none of the points are connected, DFS will be called only once for each vertex, which altogether will have a running time of O(n).

The final running time of the DFS section of the algorithm is either O(n) or O(n + (2)m), or somewhere in between, which is all equivalent to O(n + m).

Therefore the final running time of this algorithm is O(n + m) + O(n) + O(n + m) + O(1) = O(n + m).