Jordan Allard
CSCI 264-01
Homework 5

HW5 Problem 5 Writeup

a) Verbal Description

This algorithm uses a breadth-first search from vertex s to keep track of the number of shortest paths to each other vertex in the graph. It uses a dynamic programming approach to calculate these paths by calculating the number of shortest paths for the first depth of vertices, then uses those results to calculate the number of shortest paths for next depth of vertices, and so on.

The Heart of the Solution is as follows:

```
I) WHAT: p[j] = the number of shortest paths from vertex s to vertex j
II) HOW:
      p[j] = 1          if j = s;
      p[j] = sum(p[i]), where i is all the vertices, with minimum
                  depth, from which j can be discovered
III) WHERE: p[t]
```

b) Pseudocode

```
Given a graph with n vertices and m edges
For i = 1 to n:
      numPaths[i] = 1
      depths[i] = -1
beg = 1
end = 2
queue[1] = s
depth[s] = 0
While (beg < end):
      currVertex = queue[beg]
      For every neighbor u in adjList[currVertex]:
            If depth[u] == -1:
                  queue[end] = u;
                  depth[u] = depth[currVertex] + 1
                  numPaths[u] = numPaths[currVertex]
                  end++
            Else if depth[currVertex] + 1 == depth[u]
                  numPaths[u] += depth[currVertex]
      beg++
Return numPaths[t]
```

c) Proof of Correctness (Explanation of HOW)

A breadth-first search traverses every edge in a graph, so every possible path to the target vertex will be discovered. BFS does not add duplicate vetices to the queue, so if a point is discovered more than once, then it must have been discovered from different vertex. This means that the path through which it was found is unique from all other paths. Thus, each time the point is rediscovered at the same depth, the algorithm has found a brand new set of unique same-length paths. By nature of BFS, the shortest distance will always be the depth at which the vertex is first discovered. Thus, adding every the paths from each time a vertex is rediscovered at the same depth as when it was initially discovered will yield the total number of shortest paths to that vertex.

d) Running Time Estimate

O(n + m)

e) Running Time Estimate Reasoning

The running time of this algorithm can be broken down as follows:

- Initialization --> O(n)
- Building the adjacency list from input --> O(m)
- Initializing BFS --> O(n)
- BFS --> O(m)
- Cleanup --> O(n)
- Everything else --> O(1)

A breadth-first search traverses every edge in the graph one time. The while loop will run exactly n times, since each vertex will be added to the queue only one time. The for loop inside the while loop has a running time of O(deg(head)), so the total running time of the while and for loops together is the sum of deg(v) for all n vertices, which is equivalent to O(m). Thus the total running time of the algorithm is O(n) + O(m) + O(1) = O(n + m).