ComS 311
Recitation 3, 2:00 Monday
Homework 4

Sean Gordon

October 22, 2019

Algorithm 1 Define G^2 from G using paths of length 2, excluding cycles.

Require: G is provided as an adjacency list'.

```
Assume G is stored in "G"

Create empty adjacency list named "G2"

#For every vertex...
for all list in G do
  start = current vertex
  G2.add(start)

  #For every vertex this points to...
  for all vertex in list do
    innerList = G.get(vertex)

    #For every vertex that that vertex points to...
    for all boof do

      #If this vertex is the start (u == v)
      if vertex == start then
        continue
      end if

      #Add this edge (of length 2) to the new graph
      G2.get(start).add(vertex)
    end for
  end for
end for
```

The runtime of this algorithm is
1st-Loop(V) * 2nd-Loop(E) * 3rd-Loop(V): $O(V^2*E)$

Algorithm 2 Find the number of shortest paths from s to vertex i.

```
Assume G is stored in adjacency list "G"

Create object Pair that stores two Integers
Create an array paths of size V
The array will store path length and count for each vertex in a Pair obj

//Perform breadth first search on the graph ————————————

//Create a queue for BFS that holds depth and the vertex in a Pair
LinkedList<Pair> queue = new LinkedList<Pair>();
boolean visited = new boolean[V];

//Mark the current node as visited, add it to the array, and enqueue it
visited[s] = true;
paths[s] = new Pair(0, 1);
queue.add(new Pair(0, s));

while queue.size() != 0 do
  //Dequeue a vertex
  Pair pair = queue.poll();
  int depth = vertex.depth;
  int vertex = vertex.node;

  Iterator iterator = G[vertex].listIterator();
  while iterator.hasNext() do
    int v = iterator.next();

    if !visited[v] then
      visited[v] = true;
      paths[s] = new Pair(depth+1, 1);
      queue.add(new Pair(depth+1, v));

    else if paths[v].length == depth+1 then
      //If this depth == the one already stored, this is a shortest path
      paths[v].count = paths[v].count + 1;
    end if
  end while
end while
return paths[i].count;
```

Honestly I have no idea how to induction this crap lol
Runtime for above algorithm:
1 while loop through each vertex $\Rightarrow$ O(V)
1 while loop through each edge of each vertex $\Rightarrow$ O(E)
These two combine to become O(V+E)

3)

Shit here...

4)
Goddamnit