Homework 2 COM S 311

Alec Meyer

September 17, 2020

Question 1

```
v.explored = true; v.layer = 0;
initialize empty queue Q;
add v to Q;
while queue is not empty
{
        pop u from the head of Q;
        for all neighbors w of node u
        {
                if(w.explored == false)
                {
                        add w to the tail of Q;
                        w.explored = true;
                        w.layer = u.layer+1;
                        if(w.layer is even)
                                add w to group a;
                        else
                                add w to group b
                }
        }
}
for each node u in the graph
{
        if(u % 2 == u.next % 2)
                return false;
}
```

This is algorithm has a modified version of BSF. The only difference is that we are adding each node to a group a or b depending on if it is odd or even, this process is completed at O(1) time so it will not affect the time complexity of BSF. The code shown after BSF traverses the graph and checks checks if any adjacent nodes are both odd or both even. This check is in O(1) time and it takes O(V) to traverse each vertex. If you combine these two sections you will get O(V + E) + O(V) which simplifies to a total runtime of O(V + E).

Question 2

```
search(v, visited[], parent)
{
        add v to the visited array;
        for each neighbor u of v
        {
                if(u == parent && u is visited)
                        return true;
                if(u is not visited)
                        if(search(u, visited, v)) //recur this function
                                return true;
        }
        return false;
}
nodes = array of nodes
for each unvisited node v in nodes
{
        if(search(v, nodes, v.parent))
                return true;
}
```

Similar to BSF this algorithm preforms a traversal of each neighbor of v which results in a O(V) time. As for the recursive part of the algorithm, this will at worst traverse through every edge in the graph resulting in a O(E). the code outside of this recursive function will result in a O(2V + E). In total this runtime is O(2V + E) + O(V + E) whuch can be simplified to a runtime of O(V + E).

Question 3

```
arbitrary starting element v;

node x = BFS(v) //to find the furthest element from v
node u = BFS(x) //furthest node from x
BFS(x, u) //use this BFS to calculate the distance
```

This algorithm just preforms an un-modified version of BFS three times. Therefore, the runtime of this algorithm will be O(V + E)

Question 4

```
Distance d;
const L;
for each v in village
{
        BSF from DoctorResidence to v
                keep track of d;
        BSF from v to DoctorResidence
                keep track of d;
}
return d * L;
```

This algorithm preforms a nested BSF twice so we start with $O(V+E)$. The outer for loop is going to iterate for all V in the graph. Since the function $|V|+|E|$ will iterate V times, the final resulting runtime will be $O(V^2+E)$.

Question 5

```
s.explored = true;
initialize empty queue Q
add s to Q
while queue is not empty
{
        pop u from the head of Q;
        if(u has 0 neighbors)
                then v = node u;
        for all neighbors w of node u
        {
                if(w.explored == false)
                {
                        add w to the tail of Q;
                        w.explored = true;
                }
        }
}
```

This algorithm is a modified version of BFS where the only difference is checking if node **u** has 0 neighbors, which can be done at O(1) time. Once the algorithm finds a layer with only one node then it will set **v** equal to that node. Worst case, node **v** will be $V - 1$ from starting node **s**. That being said, the runtime for this algorithm will be O(V + E).