Week 9 Exercises Submit

Sandra Batista

1.1-1.2

Exercise 1: BFS Analysis

1. What is the loop invariant of BFS?
2. What is the runtime of BFS?

```
BFS(G,u)
for each vertex v
    pred[v] = nil, d[v] = inf.
Q = new Queue
Q.enqueue(u),  d[u]=0
while Q is not empty
    v = Q.front(); Q.dequeue()
    foreach neighbor, w, of v:
        if pred[w] == nil // w not found
            Q.enqueue(w)
            pred[w] = v,  d[w] = d[v] + 1
```

Exercise 2: DFS Analysis

1. What is the loop invariant of DFS?
2. What is the runtime of DFS?

```
DFS-Visit (G, start_node)
for each vertex u
  u.color = WHITE
  u.pred = nil
st = new Stack
st.push_back(start_node)
while st not empty
  u = st.top(); st.pop()
  if u.color == WHITE
    u.color = GRAY
    foreach vertex v in Adj(u) do
      st.push_back(v)
```

3. Dijkstra’s Algorithm Analysis

What is the run-time of Dijkstra's algorithm?

```
SSSP(G, s)
  PQ = empty PQ
  s.dist = 0;  s.pred = NULL
  PQ.insert(s)
  For all v in vertices
      if v != s then v.dist = inf;
      PQ.insert(v)
  while PQ is not empty
      v = min(); PQ.remove_min()
      for u in neighbors(v)
          w = weight(v,u)
          if(v.dist + w < u.dist)
              u.pred = v
              u.dist = v.dist + w;
              PQ.decreaseKey(u, u.dist)
```

4. Directed Graphs and Adjacency Lists

Starter code:
https://github.com/sandraleeusc/csci104_fall2020_lecture/blob/master/directedGraph.cpp

In this exercise you will implement a directed graph using adjacency lists.

You may assume that all the nodes are labeled with numbers from 1 to n where n is the number of nodes in the graph.

You are also given a struct for a node on the adjacency list.

You must complete the implementation of the DirectedGraph in the starter code.

5. Generating Subsets

Starter code: https://github.com/sandraleeusc/csci104_fall2020_lecture/blob/master/printsubset.cpp

1. Write a function, print_subsets, that is given a vector of strings, stringSet, and integer, k, outputs all subsets of stringSet of size k. You may assume that k is greater than 0 and less than or equal to the size of stringSet. Your implementation should be recursive and you may use helper functions. You may also use loops if absolutely necessary.

Suppose stringSet is the set, {”cat”, ”dog”, ”rabbit”, ”robot”} and k is 3.

Your function should print the following:

```
cat dog rabbit

cat dog robot

cat rabbit robot

dog rabbit robot
```

```
void print_subsets (vector<string> stringSet, int k);
```

2. ***** Trace the recursive call in main() *****