Hw2: Graph Search Algorithms

Total: 20pts. Due: 3:29PM Feb. 6th on turnin system

You can work on this assignment in a team of two members.

Include your answers to Q1 and Q2 in a file named hw02-part1.txt and submit it through turnin.

1. [3pts] Show the d and π values that result from running breadth-first search on the following undirected graph, using vertex M as the source.

2. [3pts] Show how the depth-first search works on the following graph. Assume that the for loop of lines 6-10 of the DFS procedure considers the vertices in alphabetical order, and assume that each adjacency list is ordered alphabetically. Show the discovery and finishing times for each vertex, and show the classification of each edge.
3. **Implement the graph depth-first search algorithm using recursion.** I have provided a driver program dfs.java and some helping classes for you to start with and test your implementation. The driver program is based on listing 13.1 from textbook of data structure I. This code includes a StackX class, a Vertex Class, a Graph class, and a driver application. You will make the following changes to dfs.java.

   a. [1pt]Add instance variables to class Vertex to represent: a vertex’s color, a vertex’s discovery time, a vertex’s finishing time, and a vertex’s parent.

   b. DFS using adjacency matrix only.

      1. [3pts]Add a public method dfsRec() into class Graph to implement the method in Q2. This method performs DFS recursively without explicitly using stack. Note that you may need a private recursive helping method in addition to the specified public method. **Assume the edges are represented with the adjacency matrix, which is already included in the program.**

      2. [1pt]Add a public method displayAdjMat() into class Graph that prints the adjacency matrix to the console.

   c. DFS using adjacency list only.

      1. [1pts] Add an instance variable adjList into class Graph so that the graph is also represented by adjacency list. Note adjList is an array of linkedList. You can use java.util.LinkedList, instead of implementing the linkedlist yourself. Change the Graph constructor accordingly.

---

```java
DFS(G)
1  for each vertex u ∈ V[G]
2      color[u] ← WHITE
3      π [u] ← NIL
4  end-for
5  time ← 0
6  for each vertex u∈V[G]
7      if color[u] = WHITE
8          DFS-VISIT(u)
9      End-if
10  end-for

DFS-VISIT(U)
1  color[u] ← GRAY //White vertex u has just been discovered
2  time ← time + 1
3  d[u] ← time
4  for each v ∈ adj[u] //Explore edge (u, v).
5      if color[v] = WHITE
6          π[v] ← u
7          DFS-VISIT(v)
8      end-if
9  end-for
10 color[u] ← BLACK //Blacken u; it is finished.
11 f[u] ← time ← time +1
```
2. [1pts] add statements to Graph method addEdge to update the adjacency list when a new edge is created.
3. [3pts] add a public method dfsRecList() in class Graph to implement the dfs search. Assume the edges are represented with adjacency lists only and do not use the adjacency matrix. Basically, you implement the same algorithm as in step 2. The only difference is that the graph is represented by adjacency list instead of adjacency matrix.
4. [1pt] add a public method displayAdjList() in class Graph that prints the adjacency list to the console.

d. [1pts] add a public method displayVisitTime() into class Graph. This method prints out each vertex’s label, discovery time, finishing time, and parent in DFS.

e. [1pt] The driver program dfs.java and sample inputs/outputs have been posted on the course website. DO NOT use package in your program. Your program is supposed to work with the provided main method without making any additional changes. The main method reads the provided sample input files. Your output should match the sample outputs.

f. I assume you will ask me if there is any confusion about this assignment.

g. [1pt] Submit your dfs.java only through turnin before due time; No class files or directories should be submitted.