Hw5: Graph algorithm implementation

Total: 22pts

Due: March 10 11:59 on turnin system

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

In this assignment, you will program on graph depth-first search algorithm. I have provided a code from textbook of data structure I, listing 13.1. This code includes a StackX class, a Vertex Class, a Graph class, and a driver application. Your program can be built upon this code and make some modifications. Specifically, you will do the following.

1. [2pts] 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.
2. [6pts] Add a public method dfsRec() in class Graph to implement the method posted in HW4. This method does the DFS search without using stack. Instead, it is implemented recursively. Note that you will need a private helping method that is recursive. Hint: the signature of the private method can be
dfsRecVisit(Vertex u, int i) //i represents the index of u in the array of vertexList.
3. [2pts] Add a public method displayVisitTime() to class Graph. This method prints out each vertex's label, discovery time, finishing time, and parent in DFS.
4. [1pts] Add statements to test your dfsRec() and displayVisitTime() in the driver application.
5. [2pts] Add an instance variable adjList to class Graph so that the graph can be 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.
6. [2pts] Add statements to Graph method addEdge to update the adjacency list when a new edge is created.
7. [6pts] Add a public method dfsRecList() in class Graph to implement the dfs search when the graph is represented by an adjacency list. Basically, you implement the same algorithm as in 2. The only difference is that the graph is represented by adjacency list.
8. [1pt] Add statements to test your dfsRecList() in the driver application.

The output of your program should look like for the graph in the main method:

Visits:
Vertex A: discovery 1, finishing 10, parent null
Vertex B: discovery 2, finishing 5, parent A
Vertex C: discovery 3, finishing 4, parent B
Vertex D: discovery 6, finishing 9, parent A
Vertex E: discovery 7, finishing 8, parent D

9. Submit your dfs.java through turnin before due time.