CS 6301 Implementation of Data Structures and Algorithms
Long Project 2: DFS applications

Ver 1.0: Initial description
Due: 11:59 PM, Tue, Mar 21

Max excellence credits: 1.0.

- Submission procedure is the same as that of prior projects.
- For each group, only its last submission is kept, and earlier submissions are discarded.
- Your code must be of good quality and pass all test cases within time limits to earn excellence credits.
- If you deviate from the specification or modify the signature of methods given, your program will fail our testcases, and you will get 0 points.

Project Description

1. Implement the algorithm to find strongly connected components of a directed graph. Add the method to your DFS class from SP 5. Make changes so that all methods share as much of the code as possible.

```
public static DFS stronglyConnectedComponents(Graph g) { ... }
```

2. Implement strategy 1 (stitch sub-tours) discussed in the class to find an Euler tour of a given directed graph. Strategy 1 is based on the algorithm by Hierholzer. See https://en.wikipedia.org/wiki/Eulerian_path
Ensure that your implementation is efficient (do not delete edges) to handle large input size. Your program will be tested with large sized graphs, say, 200,000 nodes. If your code runs slowly, you may not get excellence credit. We will **award you 0.25 EC** if your implementation did not use list, tree, or hashmap to remember the edges traversed. Starter code Euler.java is provided. Do not change the signatures of any of the methods called by the driver.

3. Implement the methods in PERT.java.

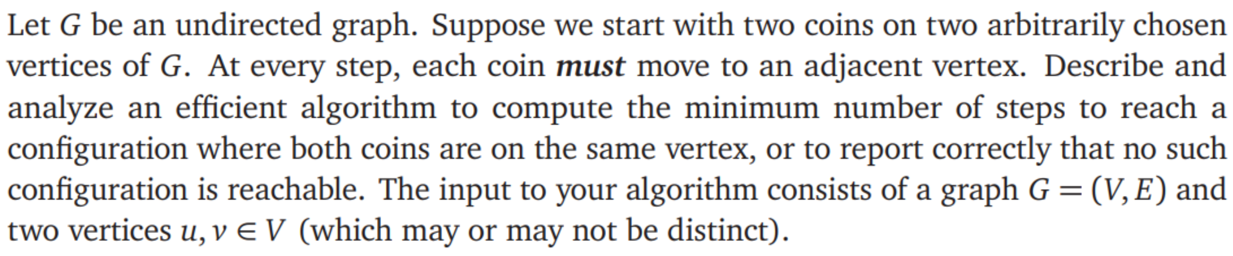
```
public static PERT pert(Graph g, int[] duration);
// Run PERT algorithm on graph g.
// Assume that vertex 1 is s and vertex n is t.
// Add edges from s to all vertices and from all vertices to t.

boolean pert(); // called after calling the constructor
```

The following methods can be called to query the results, after running one of the above methods:

```
public int ec(Vertex u);            // Earliest completion time of u
public int lc(Vertex u);            // Latest completion time of u
public int slack(Vertex u);         // Slack of u
public int criticalPath();          // Length of critical path
public boolean critical(Vertex u);  // Is vertex u on a critical path?
public int numCritical();         // Number of critical nodes in graph
```

EC problem

Write a program to solve the problem stated below. You will get 0.75 EC for this task. To be eligible for EC your algorithm should run in linear time. In the input, the last line will contain two vertex numbers (separated by space) of the two coins.



Let G be an undirected graph. Suppose we start with two coins on two arbitrarily chosen vertices of G. At every step, each coin ***must*** move to an adjacent vertex. Describe and analyze an efficient algorithm to compute the minimum number of steps to reach a configuration where both coins are on the same vertex, or to report correctly that no such configuration is reachable. The input to your algorithm consists of a graph $G = (V, E)$ and two vertices $u, v \in V$ (which may or may not be distinct).