Programming Assignment 3: Paths in Graphs

Revision: June 13, 2018

Introduction

Welcome to your third programming assignment of the Graph Algorithms course! In this and the next programming assignments you will be practicing implementing algorithms for finding shortest paths in graphs. Recall that starting from this programming assignment, the grader will show you only the first few tests.

Learning Outcomes

Upon completing this programming assignment you will be able to:

1. compute the minimum number of flight segments to get from one city to another one;
2. check whether a given graph is bipartite.

Passing Criteria: 1 out of 2

Passing this programming assignment requires passing at least 1 out of 2 programming challenges from this assignment. In turn, passing a programming challenge requires implementing a solution that passes all the tests for this problem in the grader and does so under the time and memory limits specified in the problem statement.

Contents

Graph Representation in Programming Assignments

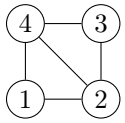
In programming assignments, graphs are given as follows. The first line contains non-negative integers n and m — the number of vertices and the number of edges respectively. The vertices are always numbered from 1 to n. Each of the following m lines defines an edge in the format `u v` where $1 \le u, v \le n$ are endpoints of the edge. If the problem deals with an undirected graph this defines an undirected edge between u and v. In case of a directed graph this defines a directed edge from u to v. If the problem deals with a weighted graph then each edge is given as `u v w` where u and v are vertices and w is a weight.

It is guaranteed that a given graph is simple. That is, it does not contain self-loops (edges going from a vertex to itself) and parallel edges.

Examples:

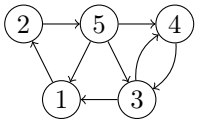
- An undirected graph with four vertices and five edges:

```
4 5
2 1
4 3
1 4
2 4
3 2
```



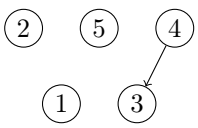
- A directed graph with five vertices and eight edges.

```
5 8
4 3
1 2
3 1
3 4
2 5
5 1
5 4
5 3
```



- A directed graph with five vertices and one edge.

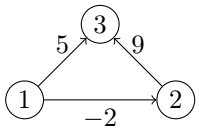
```
5 1
4 3
```



Note that the vertices 1, 2, and 5 are isolated (have no adjacent edges), but they are still present in the graph.

- A weighted directed graph with three vertices and three edges.

```
3 3
2 3 9
1 3 5
1 2 -2
```



1 Computing the Minimum Number of Flight Segments

Problem Introduction

You would like to compute the minimum number of flight segments to get from one city to another one. For this, you construct the following undirected graph: vertices represent cities, there is an edge between two vertices whenever there is a flight between the corresponding two cities. Then, it suffices to find a shortest path from one of the given cities to the other one.

Problem Description

Task. Given an *undirected* graph with n vertices and m edges and two vertices u and v, compute the length of a shortest path between u and v (that is, the minimum number of edges in a path from u to v).

Input Format. A graph is given in the standard format. The next line contains two vertices u and v.

Constraints. $2 \leq n \leq 10^5$, $0 \leq m \leq 10^5$, $u \neq v$, $1 \leq u, v \leq n$.

Output Format. Output the minimum number of edges in a path from u to v, or -1 if there is no path.

Time Limits.

language	`C`	`C++`	`Java`	`Python`	`Haskell`	`JavaScript`	`Scala`
time (sec)	2	2	3	10	4	10	6

Sample 1.

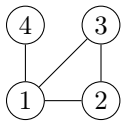
Input:

```
4 4
1 2
4 1
2 3
3 1
2 4
```

Output:

```
2
```

Explanation:



There is a unique shortest path between vertices 2 and 4 in this graph: $2 - 1 - 4$.

Sample 2.

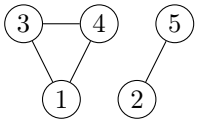
Input:

```
5 4
5 2
1 3
3 4
1 4
3 5
```

Output:

```
-1
```

Explanation:



There is no path between vertices 3 and 5 in this graph.

Need Help?

Ask a question or check out the questions asked by other learners at this forum thread.

2 Checking whether a Graph is Bipartite

Problem Introduction

An undirected graph is called *bipartite* if its vertices can be split into two parts such that each edge of the graph joins to vertices from different parts. Bipartite graphs arise naturally in applications where a graph is used to model connections between objects of two different types (say, boys and girls; or students and dormitories).

An alternative definition is the following: a graph is bipartite if its vertices can be colored with two colors (say, black and white) such that the endpoints of each edge have different colors.

Problem Description

Task. Given an undirected graph with n vertices and m edges, check whether it is bipartite.

Input Format. A graph is given in the standard format.

Constraints. $1 \le n \le 10^5$, $0 \le m \le 10^5$.

Output Format. Output 1 if the graph is bipartite and 0 otherwise.

Time Limits.

language	`C`	`C++`	`Java`	`Python`	`Haskell`	`JavaScript`	`Scala`
time (sec)	2	2	3	10	4	10	6

Sample 1.

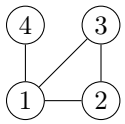
Input:

```
4 4
1 2
4 1
2 3
3 1
```

Output:

```
0
```

Explanation:



This graph is not bipartite. To see this assume that the vertex 1 is colored white. Then the vertices 2 and 3 should be colored black since the graph contains the edges $\{1, 2\}$ and $\{1, 3\}$. But then the edge $\{2, 3\}$ has both endpoints of the same color.

Sample 2.

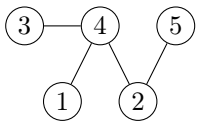
Input:

```
5 4
5 2
4 2
3 4
1 4
```

Output:

```
1
```

Explanation:



This graph is bipartite: assign the vertices 4 and 5 the white color, assign all the remaining vertices the black color.

Need Help?

Ask a question or check out the questions asked by other learners at this forum thread.