

# Programming Assignment 4: Paths in Graphs

Revision: June 19, 2018

## Introduction

Welcome to your fourth programming assignment of the Graph Algorithms course! In this assignments we focus on shortest paths in weighted graphs.

## Learning Outcomes

Upon completing this programming assignment you will be able to:

1. compute the minimum cost of a flight from one city to another one;
2. detect anomalies in currency exchange rates;
3. compute optimal way of exchanging the given currency into all other currencies.

## Passing Criteria: 2 out of 3

Passing this programming assignment requires passing at least 2 out of 3 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

<a href="#">1 Computing the Minimum Cost of a Flight</a>	4
<a href="#">2 Detecting Anomalies in Currency Exchange Rates</a>	6
<a href="#">3 Exchanging Money Optimally</a>	7

## Graph Representation in Programming Assignments

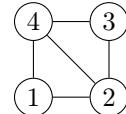
In programming assignments, graphs are given as follows. The first line contains non-negative integers  $n$  and  $m$  — the number of vertices and the number of edges respectively. The vertices are always numbered from 1 to  $n$ . Each of the following  $m$  lines defines an edge in the format  $u \ v$  where  $1 \leq u, v \leq n$  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:

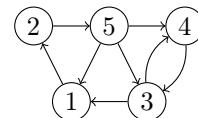
- An undirected graph with four vertices and five edges:

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



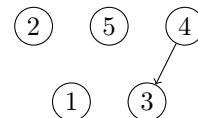
- A directed graph with five vertices and eight edges.

```
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.

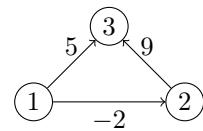
```
5 1  
4 3
```



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 Cost of a Flight

## Problem Introduction

Now, you are interested in minimizing not the number of segments, **but the total cost of a flight**. For this you construct a weighted graph: the weight of an edge from one city to another one is the cost of the corresponding flight.

## Problem Description

**Task.** Given an *directed* graph with positive edge weights and with  $n$  vertices and  $m$  edges as well as two vertices  $u$  and  $v$ , compute the weight of a shortest path between  $u$  and  $v$  (**that is, the minimum total weight of 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.**  $1 \leq n \leq 10^4$ ,  $0 \leq m \leq 10^5$ ,  $u \neq v$ ,  $1 \leq u, v \leq n$ , **edge weights are non-negative integers not exceeding  $10^3$** .

**Output Format.** Output the minimum weight of 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

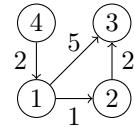
**Sample 1.**

Input:

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

Output:

```
3
```



There is a unique shortest path from vertex 1 to vertex 3 in this graph ( $1 \rightarrow 2 \rightarrow 3$ ), and it has weight 3.

**Sample 2.**

Input:

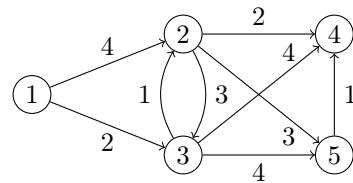
```

5 9
1 2 4
1 3 2
2 3 2
3 2 1
2 4 2
3 5 4
5 4 1
2 5 3
3 4 4
1 5

```

Output:

```
6
```



There are two paths from 1 to 5 of total weight 6:  $1 \rightarrow 3 \rightarrow 5$  and  $1 \rightarrow 3 \rightarrow 2 \rightarrow 5$ .

**Sample 3.**

Input:

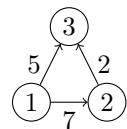
```

3 3
1 2 7
1 3 5
2 3 2
3 2

```

Output:

```
-1
```



There is no path from 3 to 2.

**Need Help?**

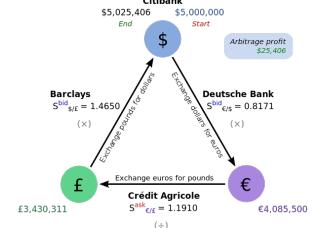
Ask a question or check out the questions asked by other learners at [this forum thread](#).

## 2 Detecting Anomalies in Currency Exchange Rates

### Problem Introduction

You are given a list of currencies  $c_1, c_2, \dots, c_n$  together with a list of exchange rates:  $r_{ij}$  is the number of units of currency  $c_j$  that one gets for one unit of  $c_i$ . You would like to check whether it is possible to start with one unit of some currency, perform a sequence of exchanges, and get more than one unit of the same currency. In other words, you would like to find currencies  $c_{i_1}, c_{i_2}, \dots, c_{i_k}$  such that  $r_{i_1, i_2} \cdot r_{i_2, i_3} \cdots r_{i_{k-1}, i_k} \cdot r_{i_k, i_1} > 1$ . For this, you construct the following graph: vertices are currencies  $c_1, c_2, \dots, c_n$ , the weight of an edge from  $c_i$  to  $c_j$  is equal to  $-\log r_{ij}$ . There it suffices to check whether is a negative cycle in this graph. Indeed, assume that a cycle  $c_i \rightarrow c_j \rightarrow c_k \rightarrow c_i$  has negative weight. This means that  $-(\log c_{ij} + \log c_{jk} + \log c_{ki}) < 0$  and hence  $\log c_{ij} + \log c_{jk} + \log c_{ki} > 0$ . This, in turn, means that

$$r_{ij}r_{jk}r_{ki} = 2^{\log c_{ij}} 2^{\log c_{jk}} 2^{\log c_{ki}} = 2^{\log c_{ij} + \log c_{jk} + \log c_{ki}} > 1.$$



### Problem Description

**Task.** Given an directed graph with possibly negative edge weights and with  $n$  vertices and  $m$  edges, check whether it contains a cycle of negative weight.

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

**Constraints.**  $1 \leq n \leq 10^3$ ,  $0 \leq m \leq 10^4$ , edge weights are integers of absolute value at most  $10^3$ .

**Output Format.** Output 1 if the graph contains a cycle of negative weight and 0 otherwise.

### Time Limits.

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

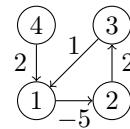
### Sample 1.

Input:

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

Output:

```
1
```



The weight of the cycle  $1 \rightarrow 2 \rightarrow 3 \rightarrow 4 \rightarrow 1$  is equal to  $-2$ , that is, negative.

### Need Help?

Ask a question or check out the questions asked by other learners at [this forum thread](#).

### 3 Exchanging Money Optimally

#### Problem Introduction

Now, you would like to compute an optimal way of exchanging the given currency  $c_i$  into all other currencies. For this, you find shortest paths from the vertex  $c_i$  to all the other vertices.

#### Problem Description

**Task.** Given an directed graph with possibly negative edge weights and with  $n$  vertices and  $m$  edges as well as its vertex  $s$ , compute the length of shortest paths from  $s$  to all other vertices of the graph.

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

**Constraints.**  $1 \leq n \leq 10^3$ ,  $0 \leq m \leq 10^4$ ,  $1 \leq s \leq n$ , edge weights are integers of absolute value at most  $10^9$ .

**Output Format.** For all vertices  $i$  from 1 to  $n$  output the following on a separate line:

- “\*”, if there is no path from  $s$  to  $u$ ;
- “-”, if there is a path from  $s$  to  $u$ , but there is no shortest path from  $s$  to  $u$  (that is, the distance from  $s$  to  $u$  is  $-\infty$ );
- the length of a shortest path otherwise.

#### Time Limits.

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

#### Sample 1.

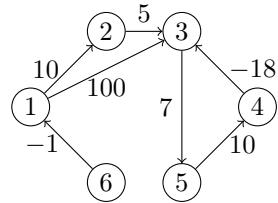
Input:

```
6 7
1 2 10
2 3 5
1 3 100
3 5 7
5 4 10
4 3 -18
6 1 -1
1
```

Output:

```
0
10
-
-
-
*
```

Explanation:



The first line of the output states that the distance from 1 to 1 is equal to 0. The second one shows that the distance from 1 to 2 is 10 (the corresponding path is  $1 \rightarrow 2$ ). The next three lines indicate that the distance from 1 to vertices 3, 4, and 5 is equal to  $-\infty$ : indeed, one first reaches the vertex 3 through edges  $1 \rightarrow 2 \rightarrow 3$  and then makes the length of a path arbitrary small by making sufficiently many walks through the cycle  $3 \rightarrow 5 \rightarrow 4 \rightarrow 3$  of negative weight. The last line of the output shows that there is no path from 1 to 6 in this graph.

### Sample 2.

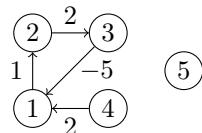
Input:

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

Output:

```
-  
-  
-  
0  
*
```

Explanation:



In this case, the distance from 4 to vertices 1, 2, and 3 is  $-\infty$  since there is a negative cycle  $1 \rightarrow 2 \rightarrow 3 \rightarrow 1$  that is reachable from 4. The distance from 4 to 4 is zero. There is no path from 4 to 5.

### Need Help?

Ask a question or check out the questions asked by other learners at [this forum thread](#).