

## Lab / Homework 12 – Graphs

### *Background: Representing Graphs*

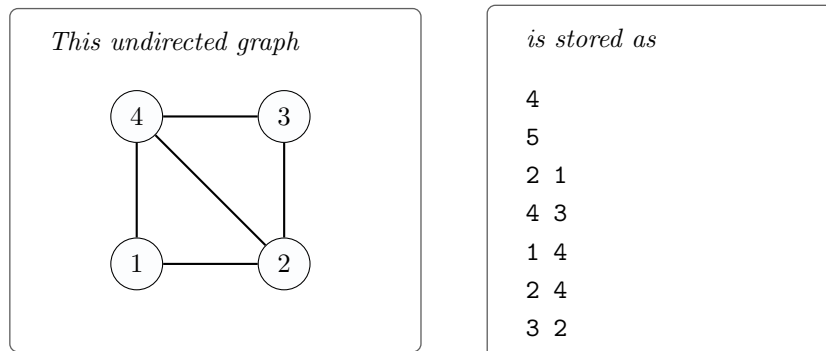
In problems 1, 2, and 3, you will work with undirected graphs. The edges in these graphs have no weights associated with them. In problem 4, you will work with a weighted directed graph.

This section explains how the graphs are stored in text files.

*How are the undirected graphs stored?*

- Graph data is stored in a text file. The first two lines of the file are integers  $n$  and  $m$  – the number of vertices and the number of edges of the graph, respectively.
- The vertices are always numbered from 1 to  $n$ , and each of the  $m$  lines that follow the first two lines defines an edge in the format  $(u, v)$  where  $1 \leq u, v \leq n$ .

*Example*



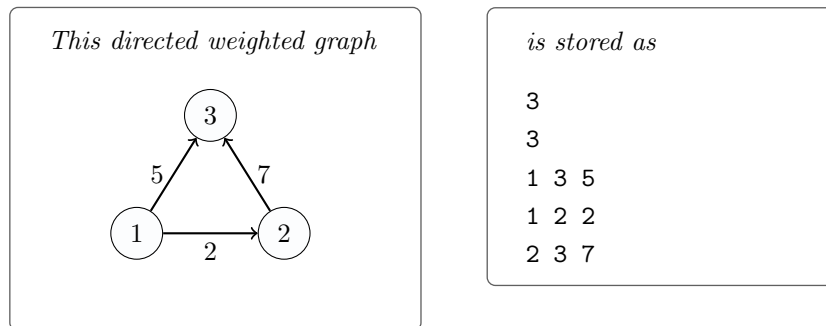
*Explanation:* The graph has 4 vertices and 5 edges, and so 4 and 5 are the first two lines of the input text file. The remaining lines list the edges.

- For each of the problems that follow, we include a helper Java class that reads graph data from a text file and converts it into the adjacency list representation of the graph.

*How are the directed weighted graphs stored?*

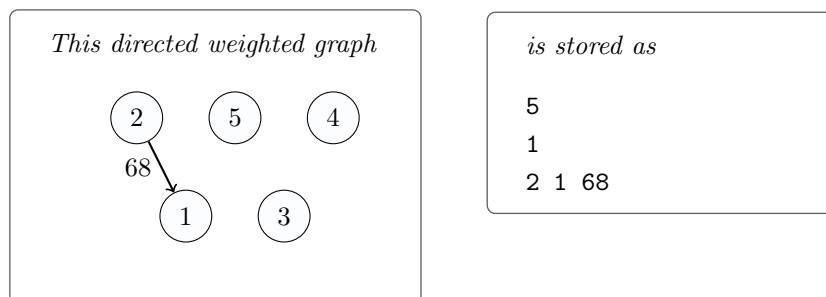
- Problem 4 deals with a weighted directed graph. Each edge in such a graph is given as  $(u, v, w)$  where  $u$  and  $v$  are vertices and  $w$  is the weight of the edge between  $u$  and  $v$ .
- The first two lines of the input file, as with the undirected graphs, contain the numbers  $n$  and  $m$ , which stand for the number of vertices and the number of edges, respectively.

*Example*



*Explanation:* The first two lines say that this directed weighted graph has 3 vertices and 3 edges. The next lines list the edges together with their weights. For example, the third line says that the edge from vertex 1 to vertex 3 has weight 5.

*Example*

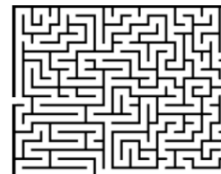


*Explanation:* This directed graph has five vertices and one edge. The edge is from vertex 1 to vertex 2 and has the weight of 68.

**Problem 1: Finding an Exit from a Maze (5 pts)**

A maze is a rectangular grid of cells with walls between some of the adjacent cells. Your goal is to check whether there is a path from a given cell to a given exit from a maze.

An exit is a cell that lies on the border of the maze (the maze shown to the right has two exits: one on the left border and one on the right border).



We can represent the maze as an undirected graph whose vertices are the cells of the maze. Two vertices are connected by an undirected edge if they are adjacent *and* there is no wall between them.

So, to check whether there is a path between two given cells in the maze, all you have to do is check whether there is a path between the corresponding two vertices in the graph.

*What am I asked to do?*

Given an undirected graph and two distinct vertices  $u$  and  $v$ , your job is to check whether there is a path between  $u$  and  $v$ .

**Constraints and Input/Output Formats**

- ◇ *Input Format.* An undirected graph with  $n$  vertices and  $m$  edges. The graph is stored in the format described on the previous page.

*The last line of the input lists the start and end vertices of the path,  $u$  and  $v$ .*

- ◇ *Constraints.*  $2 \leq n \leq 10^3$ ;  $1 \leq m \leq 10^3$ ;  $1 \leq u, v \leq n$ ;  $u \neq v$ .

*Time limit:* 1.5 sec. *Memory limit:* 512 MB.

- ◇ *Output Format.* Output 1 if there is a path between  $u$  and  $v$  and 0 otherwise.

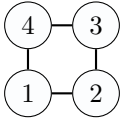
**Sample Run 1**

*input:*

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

*output:*

```
1
```

*Explanation:* The input represents this graph: , which has two paths between vertices 1 and 4. The paths are 1-4 and 1-2-3-4.

Note that the last line of the input is 1 4, which means that we are seeking a path from 1 to 4.

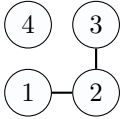
### Sample Run 2

*input:*

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

*output:*

```
0
```

*Explanation:* The input represents this graph: . There is no path between vertices 1 and 4.

### Starter files

The starter solutions for this problem read the input data from a file, pass it to a blank procedure, and then write the result to the standard output. You are asked to implement your algorithm in this blank procedure.

A folder with the starter files is [here](#).

- [Reachabliliy.java](#) is the only file that you need to edit and submit for this problem.
- [In.java](#) helps you read the input. You've used it before, and there is no need to edit it.
- [G1-1.txt](#), [G1-2.txt](#), and [G1-3.txt](#) are text files containing graph data. (The first two are the inputs from sample runs 1 and 2 shown above). At a minimum, make sure you test your solution on these graphs.

### What to do

Determine which graph search procedure discussed in class is appropriate for solving this problem and implement it. You'll find further hints in the comments in [Reachabliliy.java](#).

**Problem 2: Adding Exits to a Maze (5 pts)**

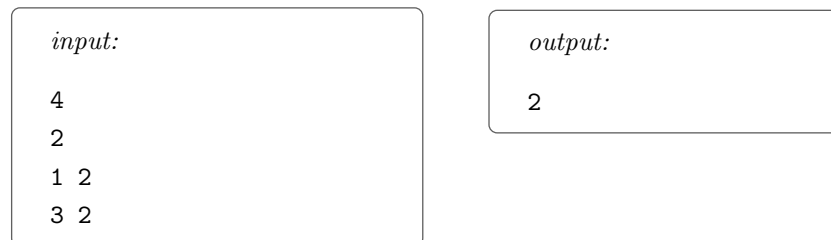
Now you decide to make sure that there are no dead zones in a maze, that is, that at least one exit is reachable from each cell. To do this, you find connected components of the corresponding undirected graph and ensure that each component contains an exit cell.

*What am I asked to do?*

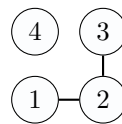
For an undirected graph with  $n$  vertices and  $m$  edges, compute the number of its connected components.

**Constraints and Input/Output Formats**

- ◇ *Input Format.* Same as in the previous problem.
- ◇ *Constraints.*  $1 \leq n \leq 10^3; 1 \leq m \leq 10^3$ .  
*Time limit:* 1.5 sec. *Memory limit:* 512 MB.
- ◇ *Output Format.* Output the number of connected components.

**A Sample Run**

*Explanation:* The input represents this graph



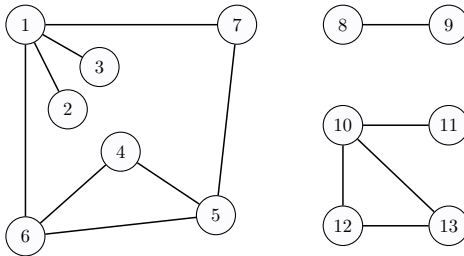
$\{1, 2, 3\}$  and  $\{4\}$ .

### *Starter files*

A folder with the starter files is [here](#).

- `ConnectedComponents.java` is the only file that you need to modify and submit.
- `In.java` helps you read the input. You've used it before, and there is no need to edit it.
- `G2-1.txt`, `G2-2.txt`, and `G2-3.txt` are text files containing graph data. (The first one is the input from the sample run shown above). At a minimum, make sure you test your solution on these graphs.

For example, `G2-3.txt` represents the following graph:



This graph has three connected components, so your solution should output '3' when given this graph as input.

### *What to do*

Modify depth-first search so that it counts the number of connected components.

Once you are satisfied that your solution works, determine how many connected components [this graph with 250 vertices and 1273 edges](#) has.

### ***Problem 3: Minimum Number of Flight Segments (5 pts)***

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

*What am I asked to do?*

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  $v$ ).

#### *Constraints and Input/Output Formats*

- ◇ *Input Format.* Same as in the previous problem.
- ◇ *Constraints.*  $2 \leq n \leq 10^5$ ;  $0 \leq m \leq 10^5$ ;  $u \neq v$ ,  $1 \leq u, v \leq n$ .  
*Time limit:* 3 sec. *Memory limit:* 512 MB.
- ◇ *Output Format.* Output the minimum number of edges in a path from  $u$  to  $v$ , or  $-1$  if there is no path.

#### *Sample Run 1*

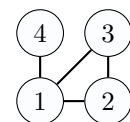
*input:*

```
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



. The

path is  $2 \rightarrow 1 \rightarrow 4$ .

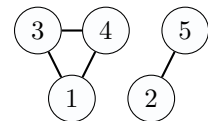
*Sample Run 2**input:*

```
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:

*Starter files*

A folder with the starter files is [here](#).

- [BFS.java](#) is the only file that you need to modify and submit.
- [In.java](#) helps you read the input. You've used it before, and there is no need to edit it.
- [G3-1.txt](#), [G3-2.txt](#), and [G3-3.txt](#) are text files containing graph data. (The first two are the inputs from sample runs 1 and 2 shown above). At a minimum, make sure you test your solution on these graphs.

*What to do*

Implement carefully the shortest paths algorithm covered in class (review breadth-first search).



### Problem 4: Computing the Minimum Cost of a Flight (5 pts)

Now, you are no longer interested in minimizing the number of segments from  $u$  to  $v$ . Instead, you would like to know the total cost of a flight from  $u$  to  $v$ . To do this, you construct a weighted graph: the weight of an edge from one city to another one is the cost of the corresponding flight.

*What am I asked to do?*

You are given a *directed* graph with positive edge weights and  $n$  vertices and  $m$  edges. You are also given two vertices  $u$  and  $v$ .

You are asked to compute the weight of a shortest path between  $u$  and  $v$  (that is, the minimum total weight of a path from  $u$  to  $v$ ).

#### Constraints and Input/Output Formats

- ◇ *Input Format.* A weighted graph in the standard format as described on page 1 of this assignment.
- ◇ *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  $10^3$ .
- Time limit:* 3 sec. *Memory limit:* 512 MB.
- ◇ *Output Format.* Output the minimum weight of a path from  $u$  to  $v$ , or  $-1$  if there is no path.

#### Sample Run 1

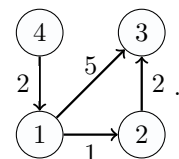
*input:*

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

*output:*

3

*Explanation:* There is a unique shortest path from vertex 1 to vertex 3 in this graph:



The path is  $1 \rightarrow 2 \rightarrow 3$ , and it has weight 3.

### Sample Run 2

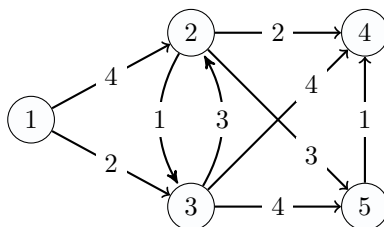
*input:*

```
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

*Explanation:* The input graph is this:



This graph has two paths from 1 to 5 of total weight 6:  $1 \rightarrow 3 \rightarrow 5$  and  $1 \rightarrow 3 \rightarrow 2 \rightarrow 5$ .

### Sample Run 3

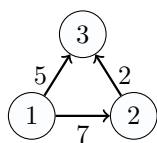
*input:*

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

*output:*

-1

*Explanation:* There is no path from 3 to 2 in this graph:



*Starter files*

A folder with the starter files is [here](#).

- [Dijkstra.java](#) is the only file that you need to modify and submit.
- [In.java](#) helps you read the input. You've used it before, and there is no need to edit it.
- [G4-1.txt](#), [G4-2.txt](#), and [G4-3.txt](#) are text files containing graph data. These files correspond to the inputs in the sample runs shown above. At a minimum, make sure you test your solution on these graphs.

*What to do*

Implement carefully the corresponding algorithm covered in class (review Dijkstra's algorithm).