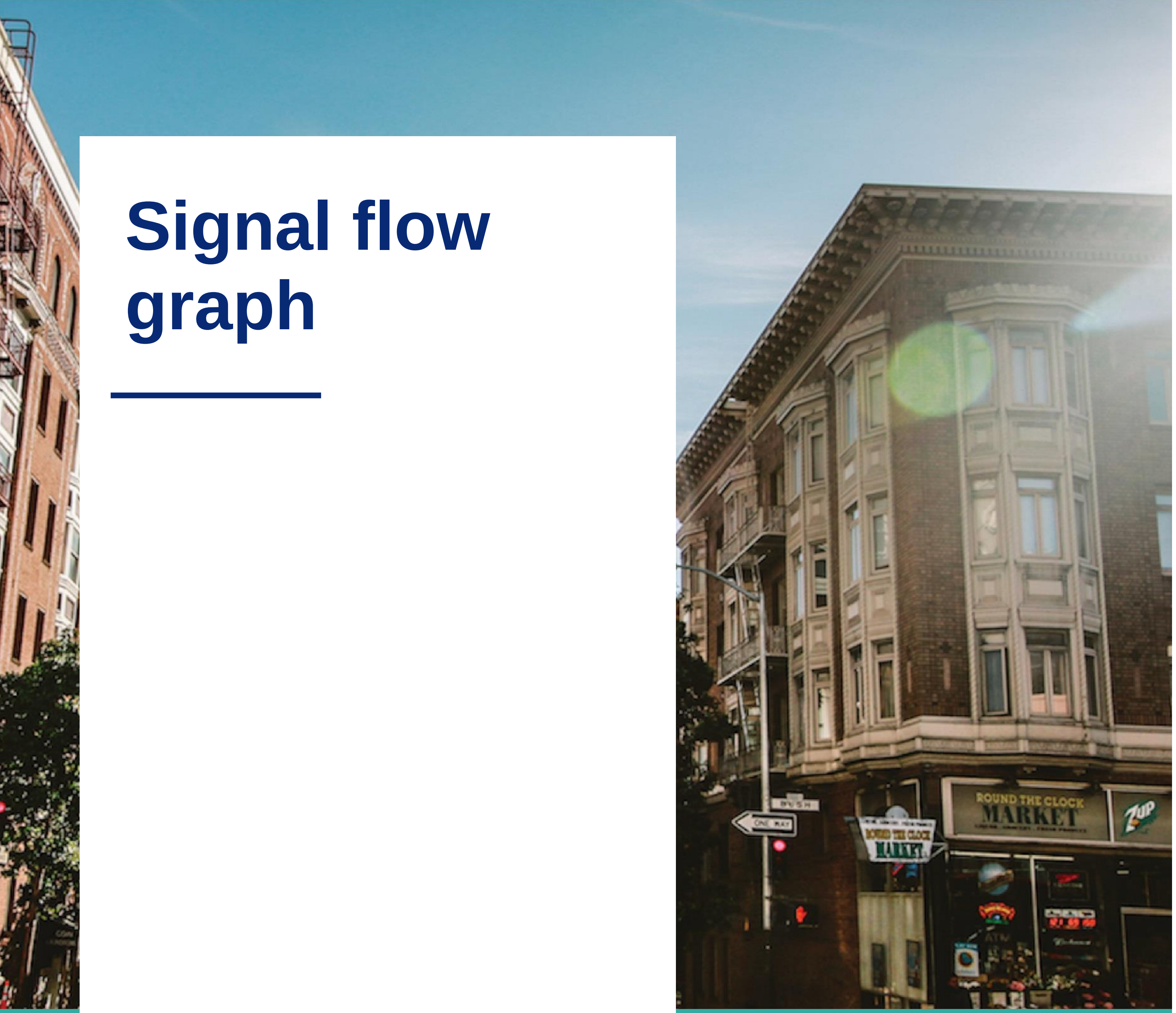


Signal flow graph

JUNE 3

Name : Hamza Hassan Mohammed

ID : 26

Contents

- Statement
- Assumptions
- Algorithms and data structures used
- How it works
- UnitTests
- Libraries Used
- Link to the repository

Statement

Given a graph, the program draws the graph interactively and calculates the overall transfer function given VALID SOURCE NODE and a sink node.

Assumptions

All data entered by the user is double (or integers), but strings (like H(s)) are not allowed

Algorithms and data structures

The program uses johnson algorithm to get all the SIMPLE cycles of the graph efficiently $O(V+E)*C$

V : number of vertices

E: Number of edges

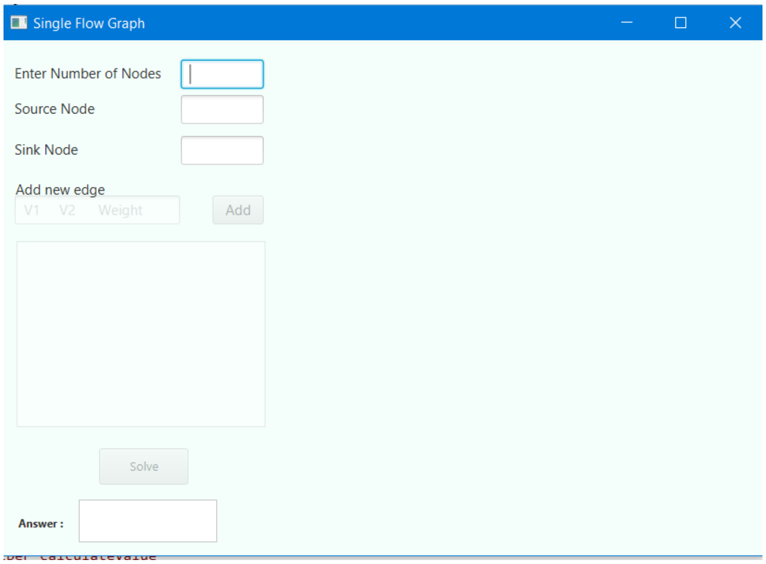
C: number of cycles

How it works

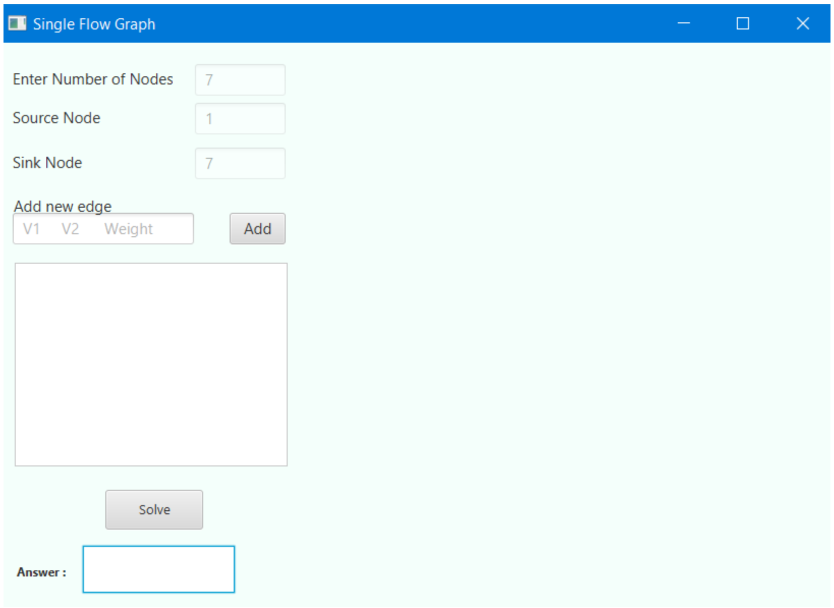
1- At first, the user is required to enter the number of node, the index of the input node and the index of the seek node

Note: the index begins with 1 and ends at end

Warning: if the user entered invalid input the labels in which he could enters the edges will be off and he will not be able to continue until he modifies the input so that it is correct



2- Once the user enters valid input, the label of add new edge will be on and the user will be able to enter the edges of graph.

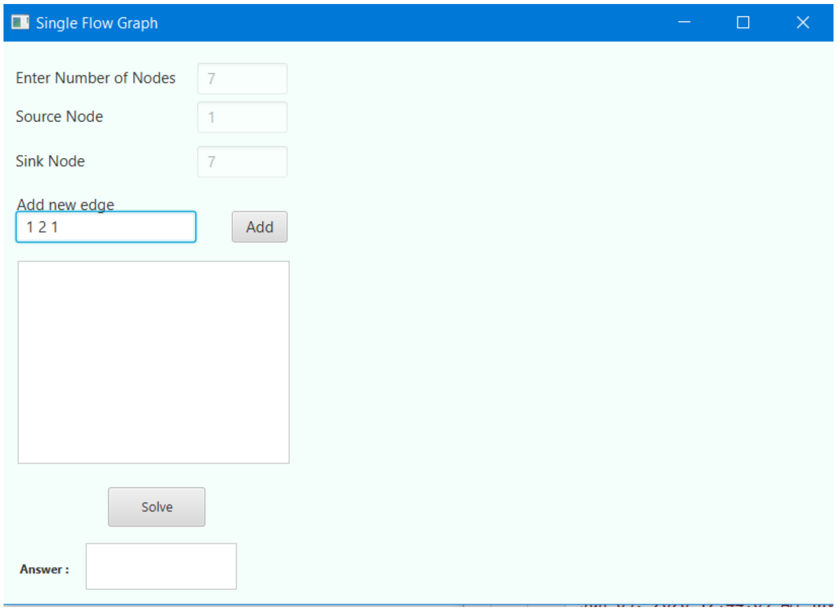


Enter Number of Nodes 7

Source Node 1

Sink Node 7

3- The user enters the edges of the graph in the following format
 ‘V1 V2 weight ‘ and presses add .



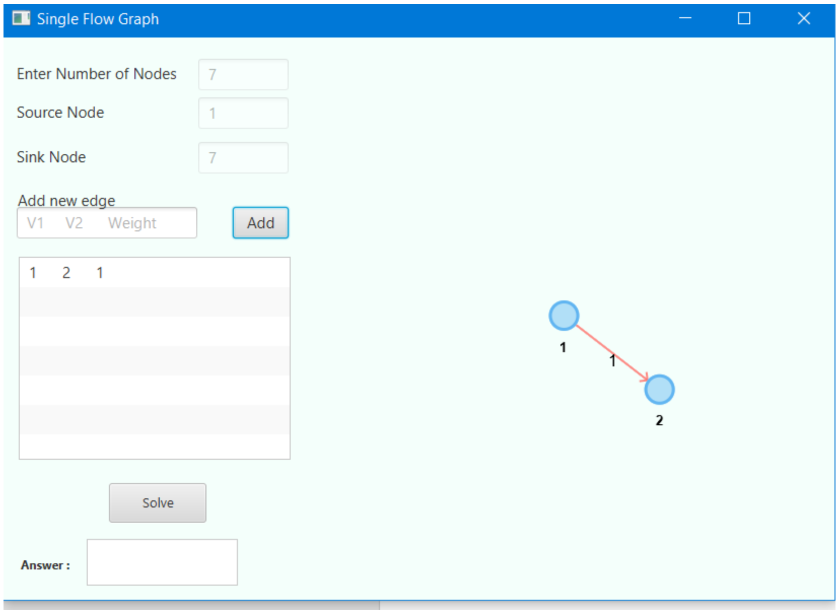
Enter Number of Nodes 7

Source Node 1

Sink Node 7

Add new edge 1 2 1

4- When the user click on the “Add” button, he edge will be added on the text area below and the visualization of the graph appears on the right of the screen



Enter Number of Nodes 7

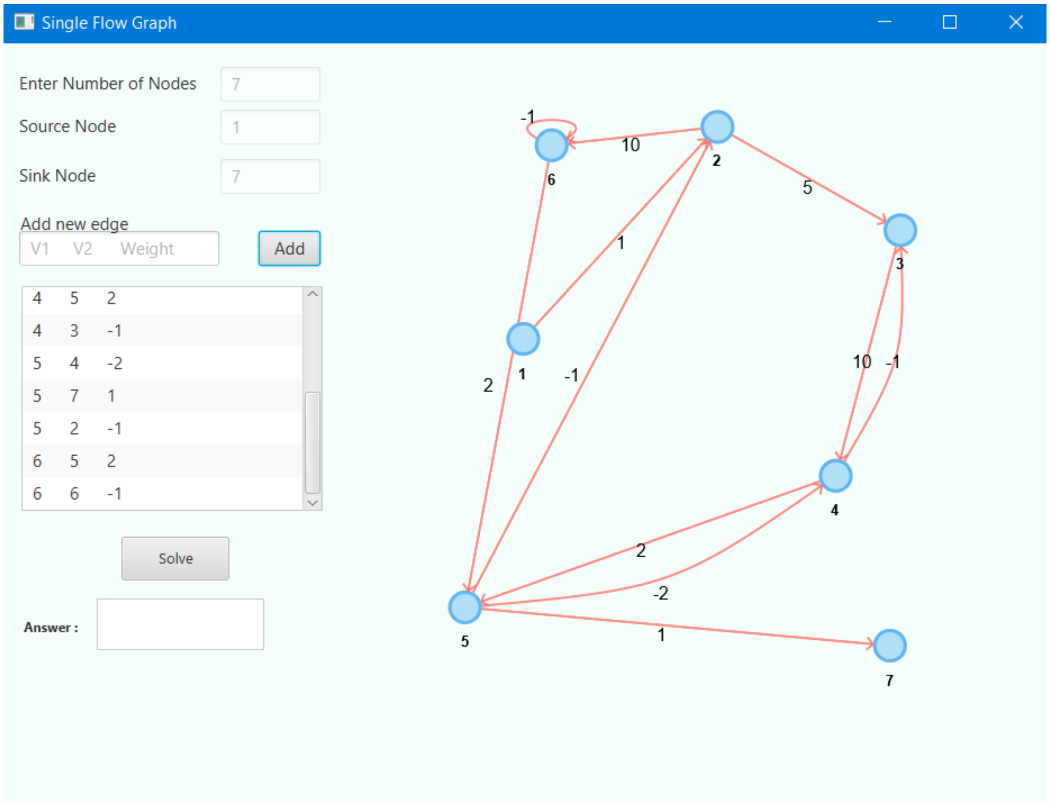
Source Node 1

Sink Node 7

Add new edge

```
1   2   1
```

5- as we see, we added bunch of different edges in the graph



Single Flow Graph

Enter Number of Nodes 7

Source Node 1

Sink Node 7

Add new edge

V1 V2 Weight

Add

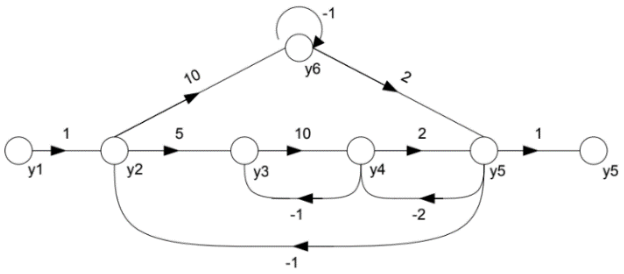
4	5	2
4	3	-1
5	4	-2
5	7	1
5	2	-1
6	5	2
6	6	-1

Solve

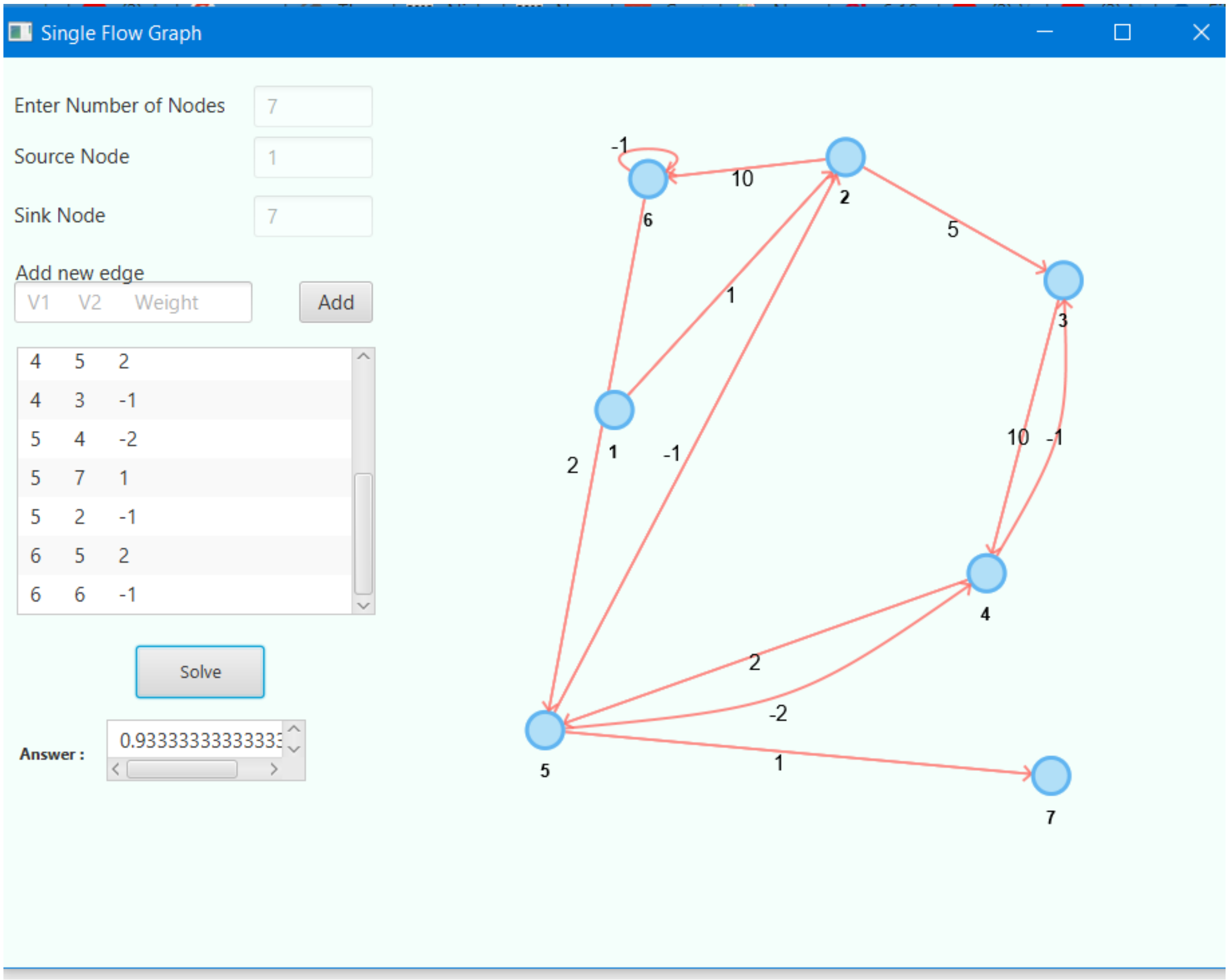
Answer :

Note : this example is the same with exercise number 3 in the sheet

[3]. Find the gains $\frac{y_5}{y_1}$ for the signal flow graph in the following graph.



6- when the user finished entering all the edges he presses the “Solve” button. as we can see, the result appeared in the solve label.



Single Flow Graph

Enter Number of Nodes 7

Source Node 1

Sink Node 7

Add new edge

V1 V2 Weight

Add

4	5	2
4	3	-1
5	4	-2
5	7	1
5	2	-1
6	5	2
6	6	-1

Solve

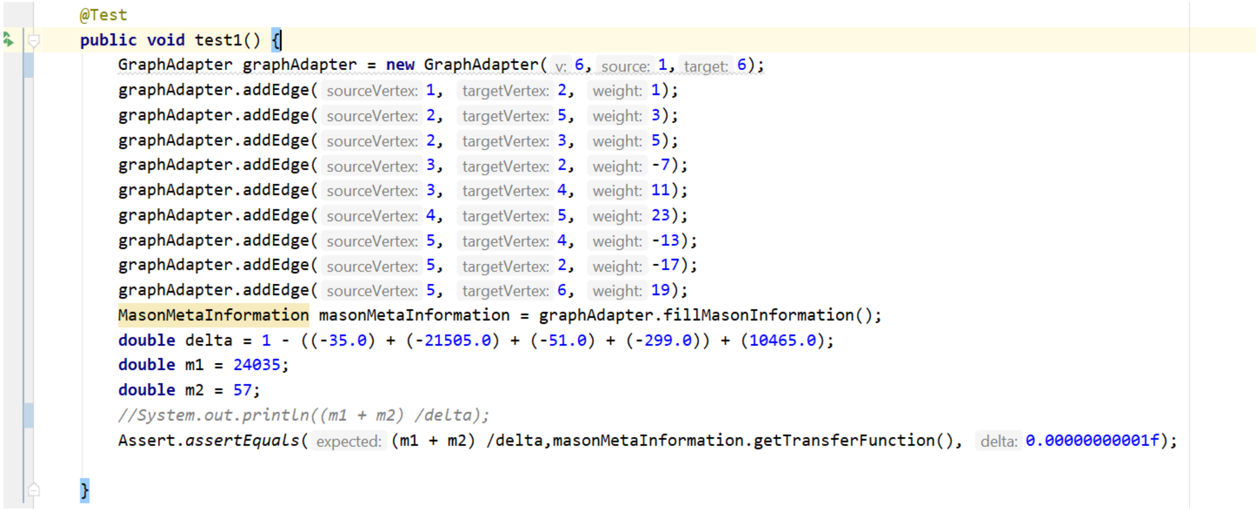
Answer : 0.9333333333333

Notes:

- The user can move the locations of vertices and edges by pressing and drag them
- If something went wrong with calculations, as error prompt will appear to the user
- In case that the edge the user entered is invalid, it will not be added toe the list (e.g. the user entered the weight as string)

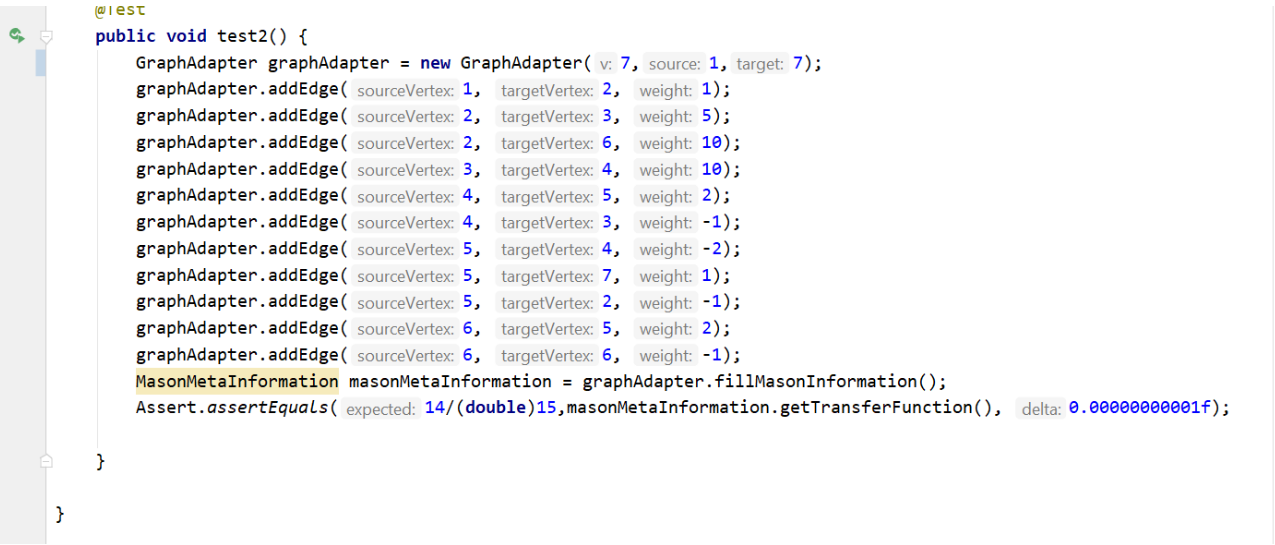
Unit Testing

Mainly, the program is provided with a UnitTest class which has two tests from the sheet no.3(one of the examples have no numbers so substituted for suitable numbers)



```
@Test
public void test1() {
    GraphAdapter graphAdapter = new GraphAdapter( v: 6, source: 1, target: 6);
    graphAdapter.addEdge( sourceVertex: 1,  targetVertex: 2,  weight: 1);
    graphAdapter.addEdge( sourceVertex: 2,  targetVertex: 5,  weight: 3);
    graphAdapter.addEdge( sourceVertex: 2,  targetVertex: 3,  weight: 5);
    graphAdapter.addEdge( sourceVertex: 3,  targetVertex: 2,  weight: -7);
    graphAdapter.addEdge( sourceVertex: 3,  targetVertex: 4,  weight: 11);
    graphAdapter.addEdge( sourceVertex: 4,  targetVertex: 5,  weight: 23);
    graphAdapter.addEdge( sourceVertex: 5,  targetVertex: 4,  weight: -13);
    graphAdapter.addEdge( sourceVertex: 5,  targetVertex: 2,  weight: -17);
    graphAdapter.addEdge( sourceVertex: 5,  targetVertex: 6,  weight: 19);
    MasonMetaInformation masonMetaInformation = graphAdapter.fillMasonInformation();
    double delta = 1 - ((-35.0) + (-21505.0) + (-51.0) + (-299.0)) + (10465.0);
    double m1 = 24035;
    double m2 = 57;
    //System.out.println((m1 + m2) /delta);
    Assert.assertEquals( expected: (m1 + m2) /delta,masonMetaInformation.getTransferFunction(),  delta: 0.00000000001f);

}
```



```
    @Test
    public void test2() {
        GraphAdapter graphAdapter = new GraphAdapter( v: 7, source: 1, target: 7);
        graphAdapter.addEdge( sourceVertex: 1,  targetVertex: 2,  weight: 1);
        graphAdapter.addEdge( sourceVertex: 2,  targetVertex: 3,  weight: 5);
        graphAdapter.addEdge( sourceVertex: 2,  targetVertex: 6,  weight: 10);
        graphAdapter.addEdge( sourceVertex: 3,  targetVertex: 4,  weight: 10);
        graphAdapter.addEdge( sourceVertex: 4,  targetVertex: 5,  weight: 2);
        graphAdapter.addEdge( sourceVertex: 4,  targetVertex: 3,  weight: -1);
        graphAdapter.addEdge( sourceVertex: 5,  targetVertex: 4,  weight: -2);
        graphAdapter.addEdge( sourceVertex: 5,  targetVertex: 7,  weight: 1);
        graphAdapter.addEdge( sourceVertex: 5,  targetVertex: 2,  weight: -1);
        graphAdapter.addEdge( sourceVertex: 6,  targetVertex: 5,  weight: 2);
        graphAdapter.addEdge( sourceVertex: 6,  targetVertex: 6,  weight: -1);
        MasonMetaInformation masonMetaInformation = graphAdapter.fillMasonInformation();
        Assert.assertEquals( expected: 14/(double)15,masonMetaInformation.getTransferFunction(),  delta: 0.00000000001f);

    }

}
```

Libraries Used

1- **Jgrapht**
   - It is used to help get the cycles using johnson algorithm.
   - Implementing such algorithm from scratch might be time consuming
   - If I just used *depth first search* to get the answer, its complexity is going to be high and can not be used in case of large input.

2- **javaFxSmartGraph**
   it is used to draw the vertices and the edges interactively

Link to the repository

https://github.com/Hamzawy63/TransferFunction/