# Signal Flow Graph

Name 1 : Moustafa Mahmoud

Id 1 : 75

Name 2: Mohamed Ayman

Id2: 61

Programming Language: Java

# 1. Problem Statement:

**Given:**
Signal flow graph representation of the system. Assume that total number of nodes and numeric
branches gains are given.

**Required:**
1- Graphical interface.
2- Draw the signal flow graph showing nodes, branches, gains, …
3- Listing all forward paths, individual loops, all combination of *n* non-touching loops.
4- The values of $\Delta$ , $\Delta_1$ , …, $\Delta_m$ where *m* is number of forward paths.
5- Overall system transfer function.

# 2. Main Features

The Program consists of two Main parts

## a. Signal Flow graph representation:

It allows the addition, deletion and edition of edges in graph with the gain of every edge and represents the Graph graphically to the user allowing ease of use and enhancing the user interface.

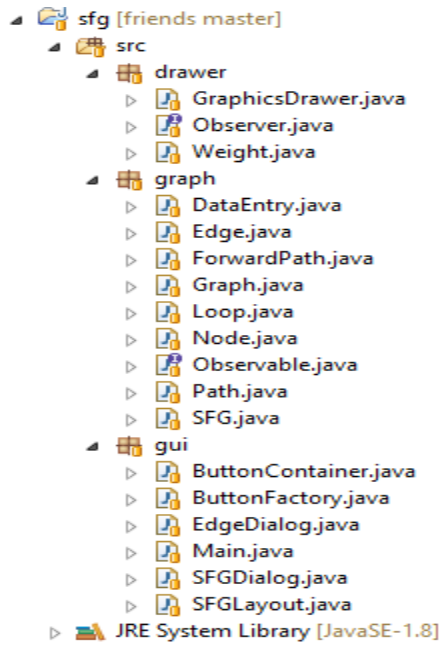
## b. Signal Flow Solver:

It's responsible for:

i. obtaining the number of forward paths from a node to another
ii. obtaining the Gain and Delta of Each path.
iii. obtaining the number of cycles and the gain of Each cycle and the value of the delta
iv. Obtaining all combinations of n non-touching loops.
v. Computing the Overall transfer Function

### 3. Main Modules:

All Modules:




## i. Signal Flow Graph Representation:

The Most important Modules Used for the Graph representation are:

#### a. GraphicsDrawer:

A Module which contains a canvas used for drawing upon the GUI using JavaFX.

This module uses Observer Design pattern where it doesn't draw the new edge added or removed except when it's notified from the SFG Module which implements Observable.

#### b. SFGDialog:

Used to handle the input from the user where it contains the Handlers for the buttons using dialogs.

c. **SFGLayout:**

Contains the main Layout of the program in the GUI dividing the Main Pane into an upper Toolbar and a Canvas used for drawing. The upper toolbar contains the buttons needed for drawing.

d. **Button Factory:**

They are used to create the initial state of the buttons and contain the data needed like the names of the buttons and connect the buttons with the SFGDialog module.

ii. **Signal Flow Solver:**

a. **SFG**

The Main Class which acts upon being a Fascade design pattern is the SFG which is also a singleton as the program supports only one Signal flow Graph.

b. **Graph:**

It's used to represent the Signal Flow graph with its nodes, edges, paths, and loops.

c. **Path:**

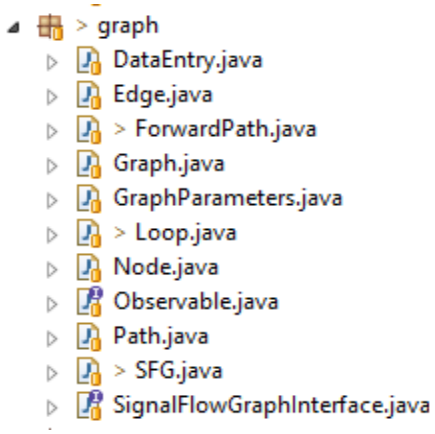
It's used to represent a Forward Path on the Graph with its gain.

d. **Loop:**

It's used to represent a loop on the Graph with its gain.

e. **Graph Parameters:**

Class used to store the Graph parameters required for traversal.




# 4. Data Structure and Algorithms Used:

## a. Data Structure:

### i. ArrayList:

They're dynamically allocated arrays in Java which were used to Contain Several Paths or to represent the Graph where it was represented using Adjacency List where each node has a List of Edges to the next Node.

### ii. TreeSet:

Which is a Red-Black tree based data structure in Java was used to avoid the duplication of loops in the generation process.

## 2. Algorithms:

### a. Depth First Traversal(DFS):

Was used to generate all the forward paths and Loops that exist in the Graph using more of a Brute forcing techniques that tries all possible ways to reach the final node.

### b. Complete Search:

Complete search was used to find all the n non-touching loops and find therefore used in the calculation of the Delta.

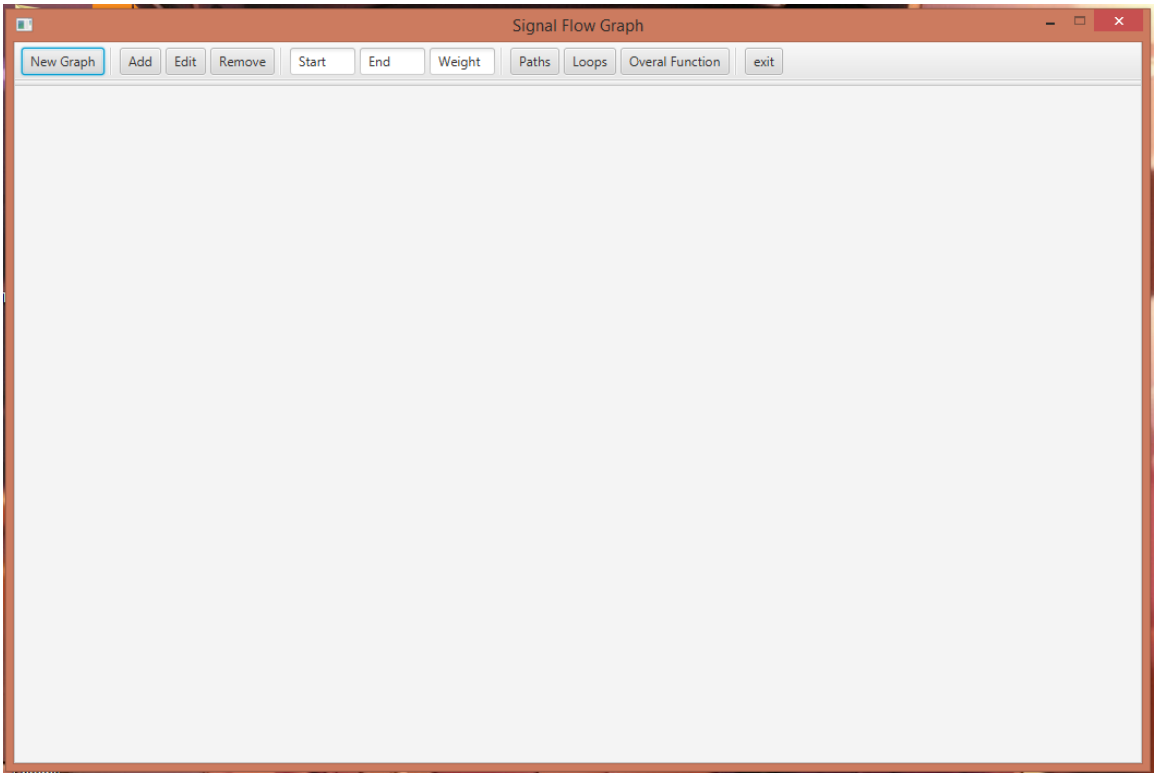
### c. Cycle Detection:

Cycle Detection Algorithm using DFS traversal was used to detect Loops in Graph.

# 5. User Manual:

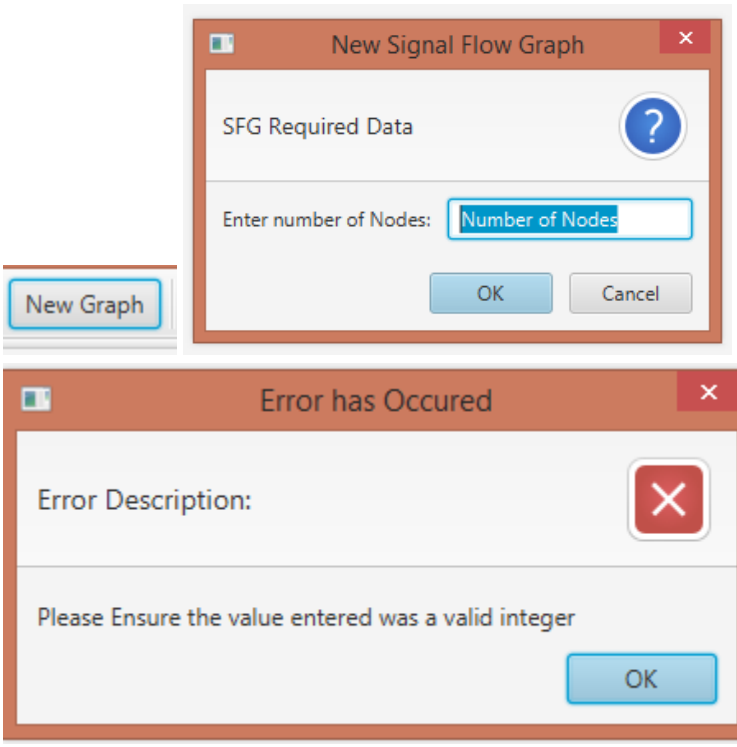
a. First We run the application by Double clicking the Jar file.


Now We'll have a Window opened containing the main Program.



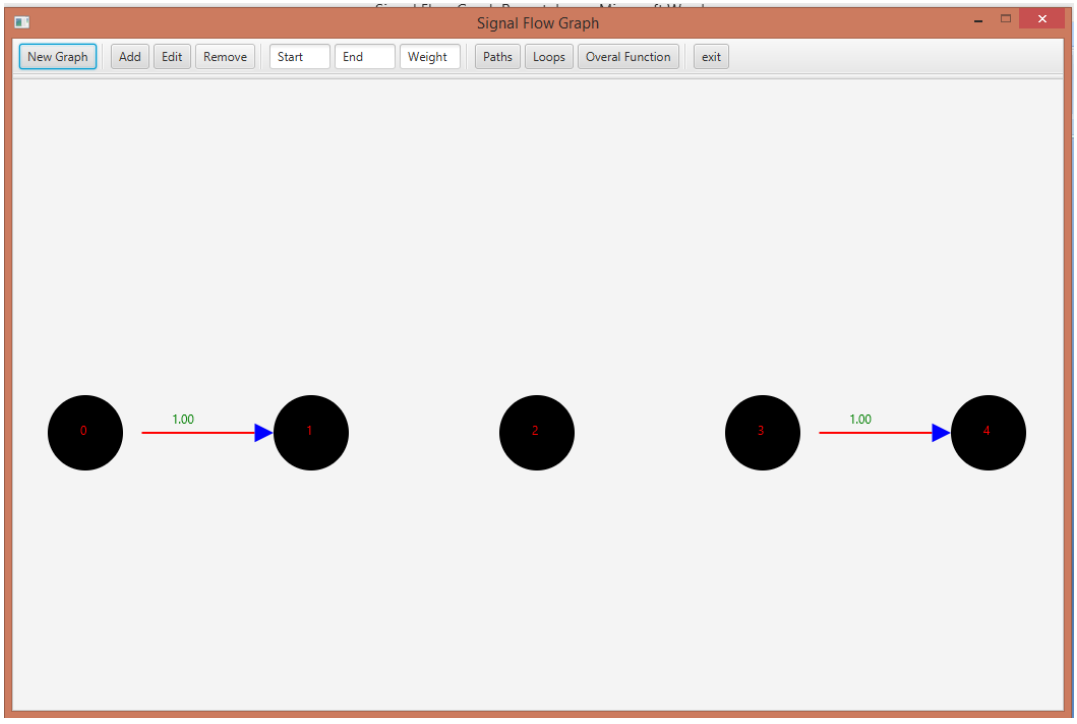

All the interface between the user and the program is through the Toolbar.

b. By Clicking on 'New Graph' Button, A Dialog will be shown to the user asking to specify the number of Nodes.
Of course in case of entering Invalid value or a non numeric value, It'll Throw an Error dialog.



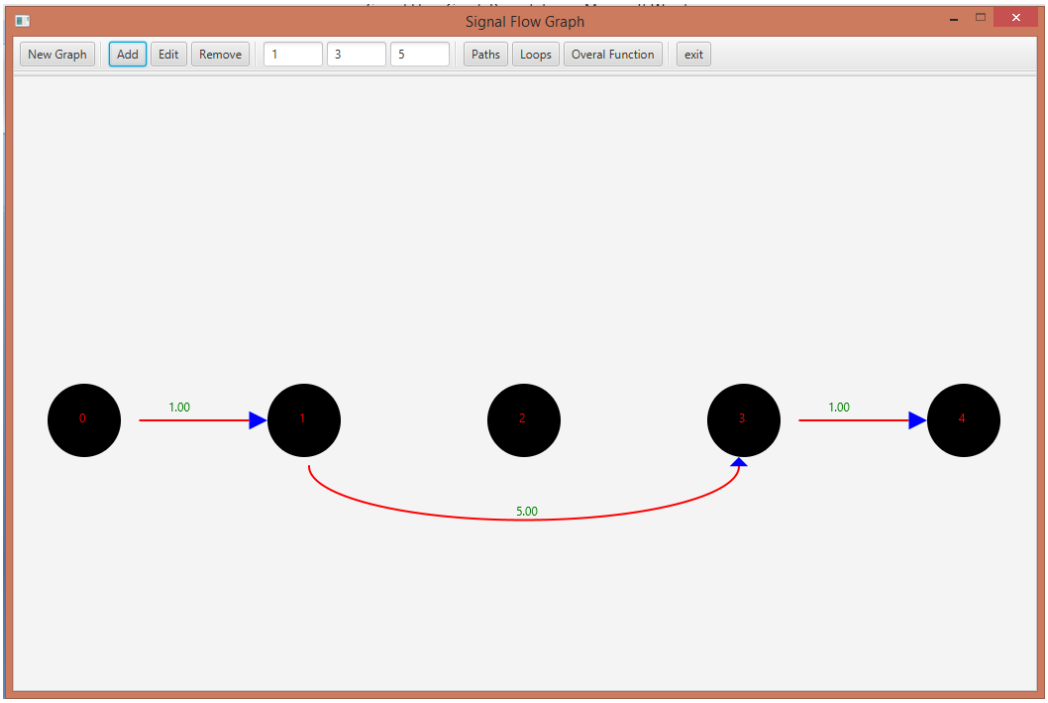

Assuming We entered '5', specifying a 5 nodes flow graph.

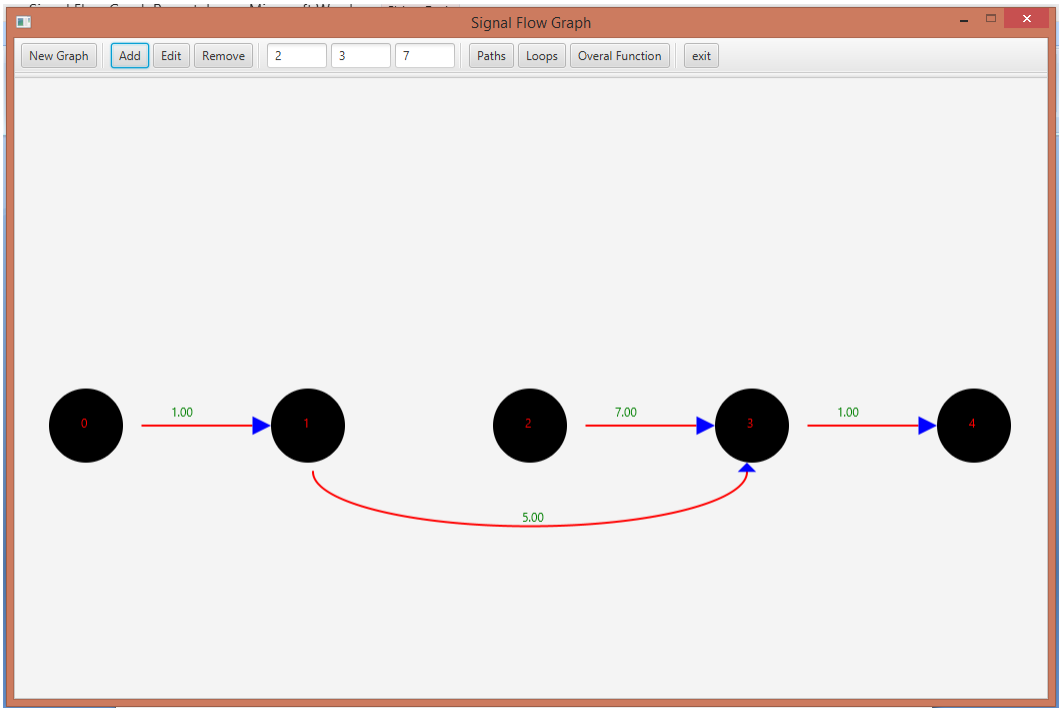
A graph of 5 nodes will be drawn in the main Canvas.

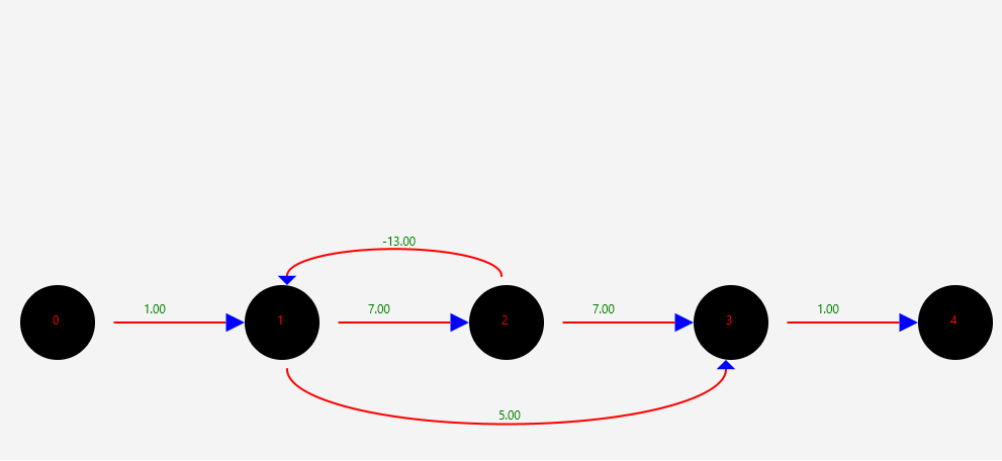



c. To add a new edge to the graph, through the three input boxes, The user can specify the starting node, ending node and the weight of Edge.

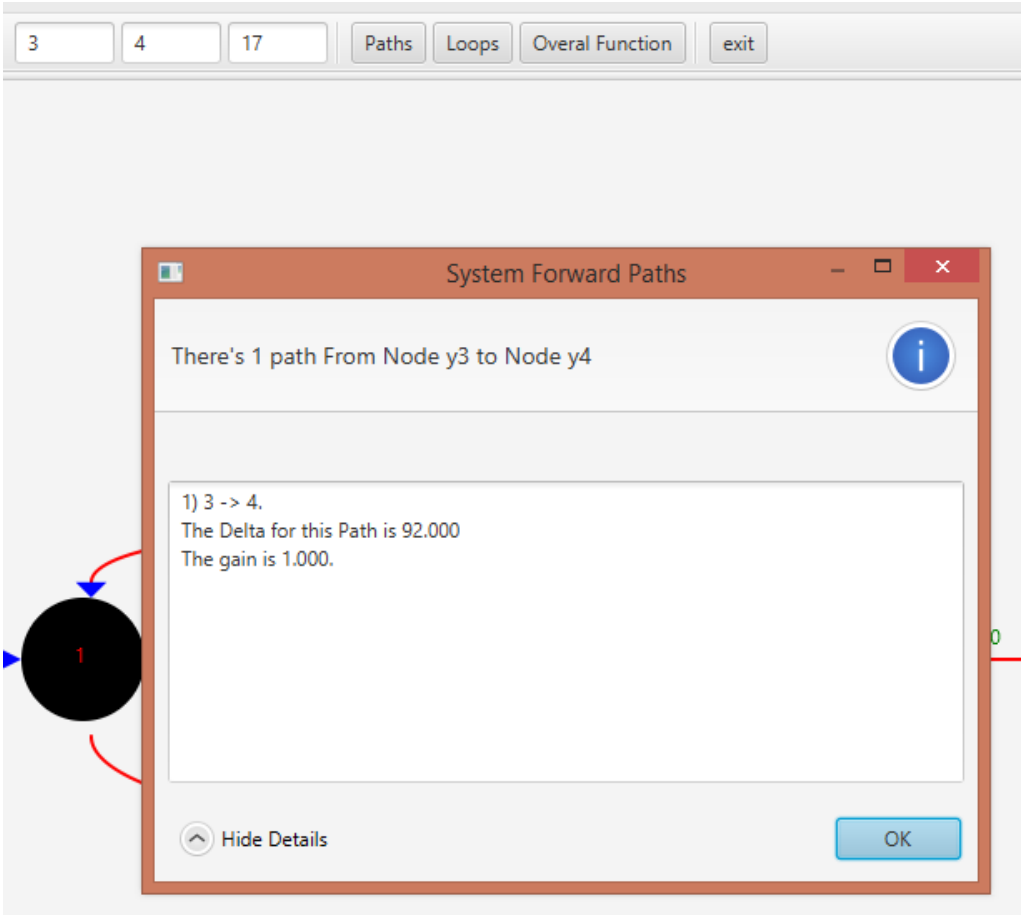
The same process is used on editing an edge or removing an edge except when removing and edge the input box for the weight is ignored.



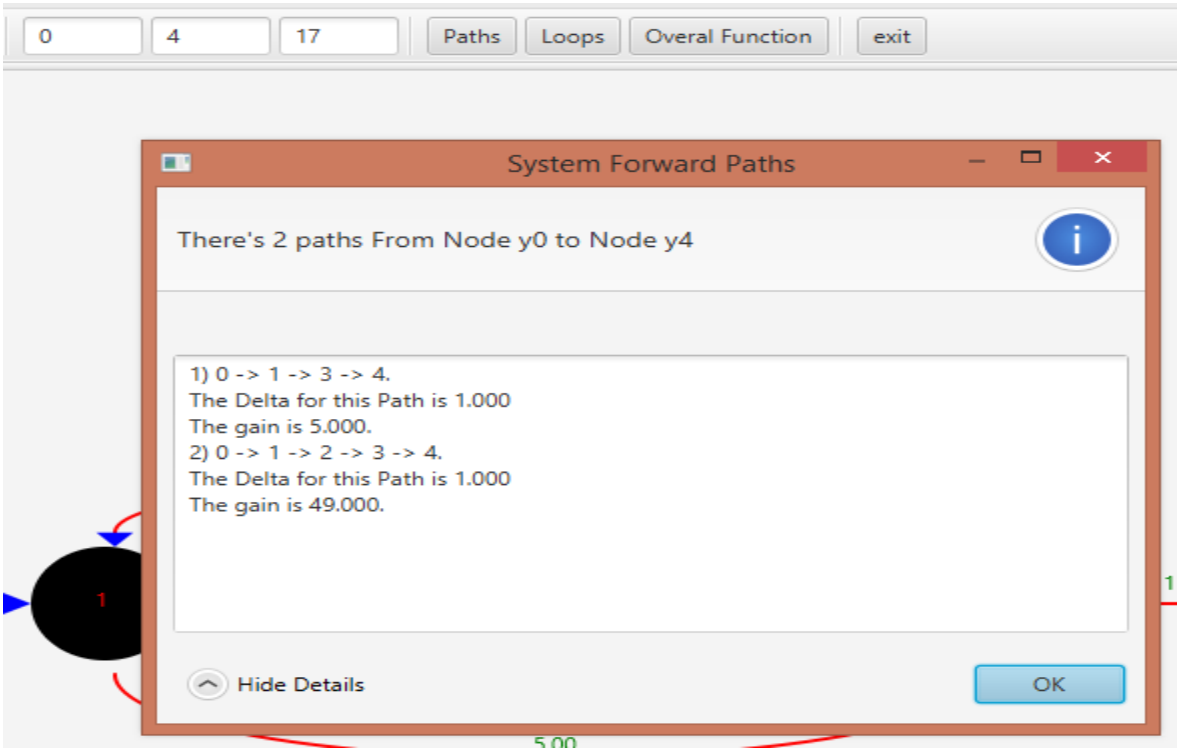



d. To print the Forward paths with each forward path gain and Delta, Click on the 'Paths' Button.
A window will be show printing all the paths from the starting node to the ending node specified in the input boxes.

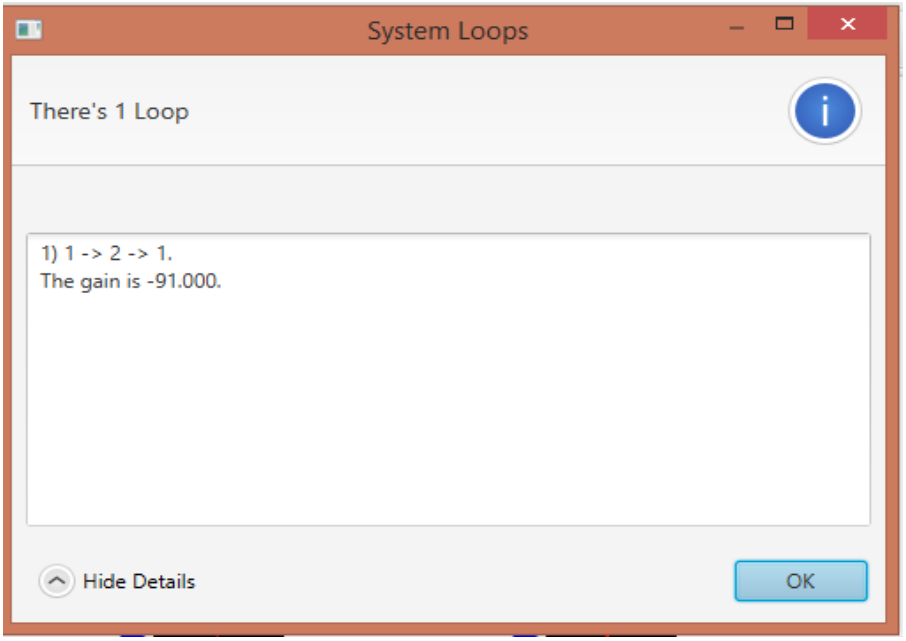



For the previous example there's only 1 Path from node Y2 to node Y3.

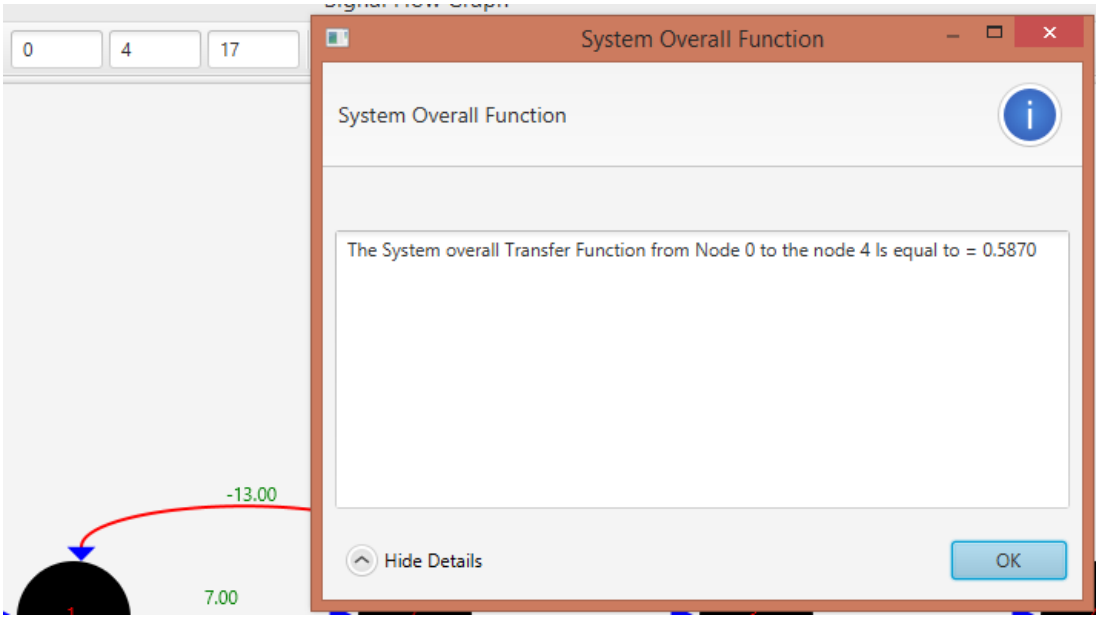
But upon changing the starting and ending node to 0 and 4, We get two forward Paths.



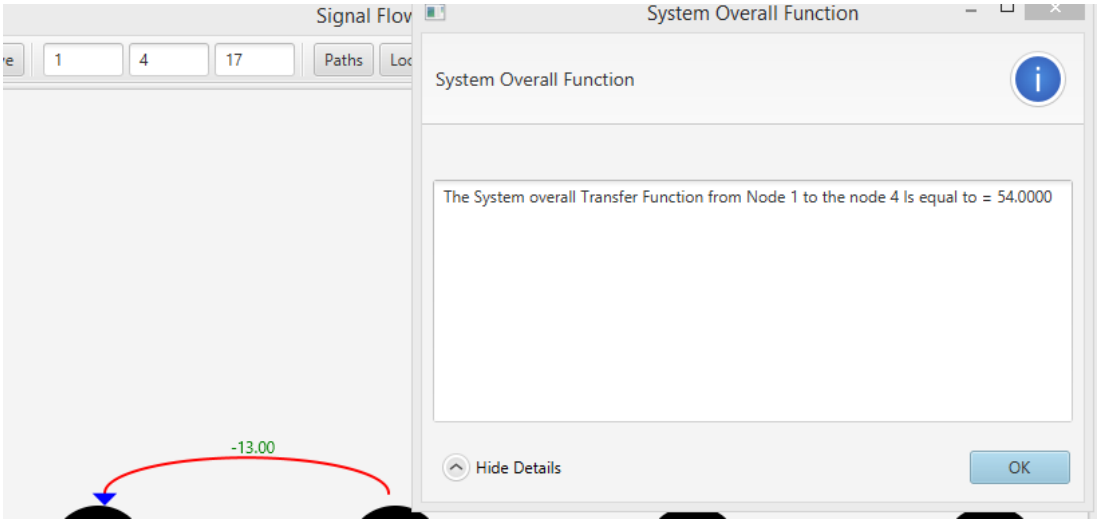

e. To generate all the n non-touching loops, Click on the Loops Button.




f. To obtain the Overall Function, Click on the 'Overall Function' Button, Specifying the nodes of input and output as the next two example.
The first example from input to the output or sink node.




The second example from 1st node to the 4th node.

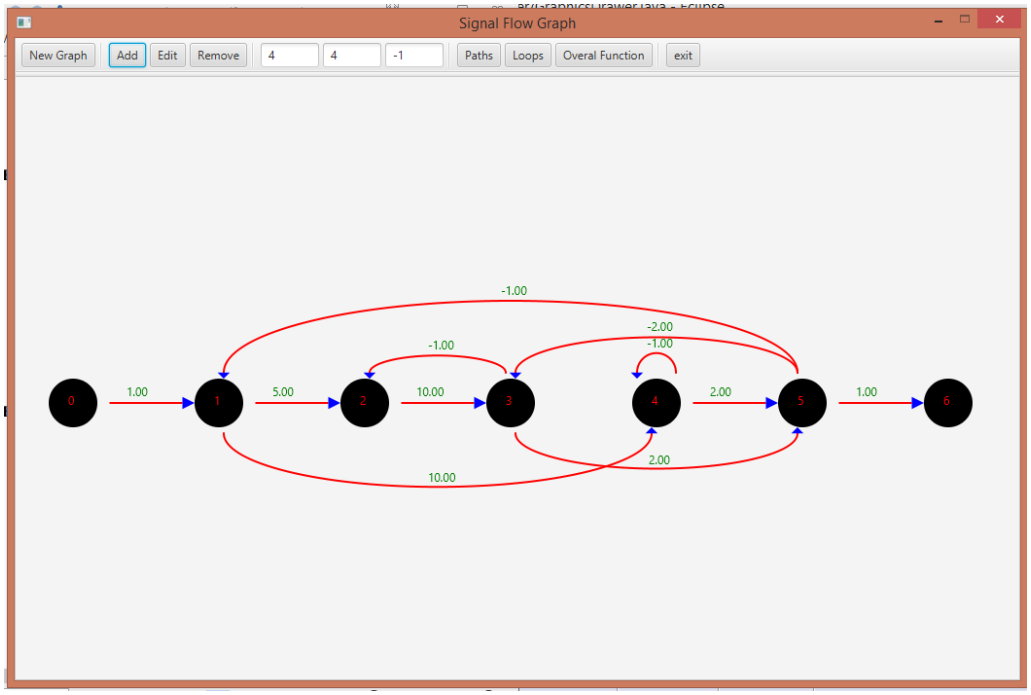


# 6. Sample Runs:

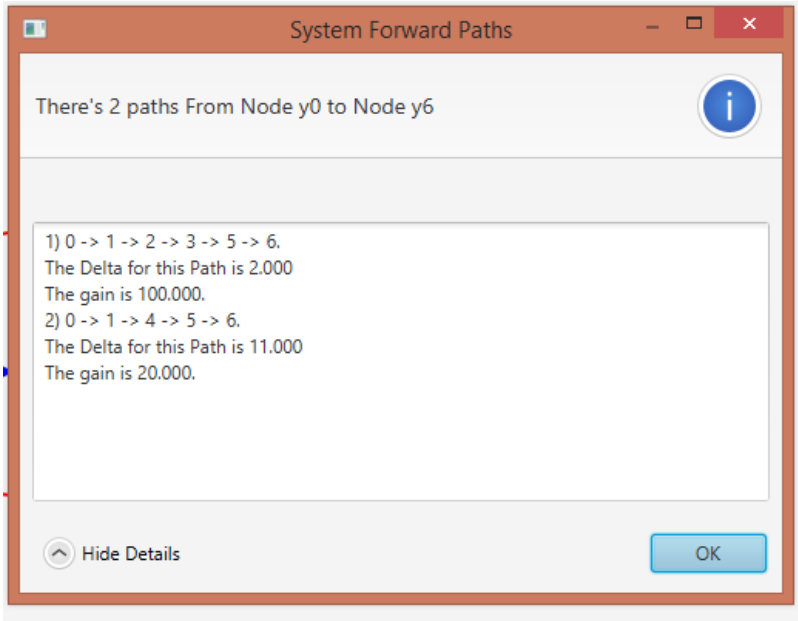
The Sample Runs consist of two Examples:

## a. First

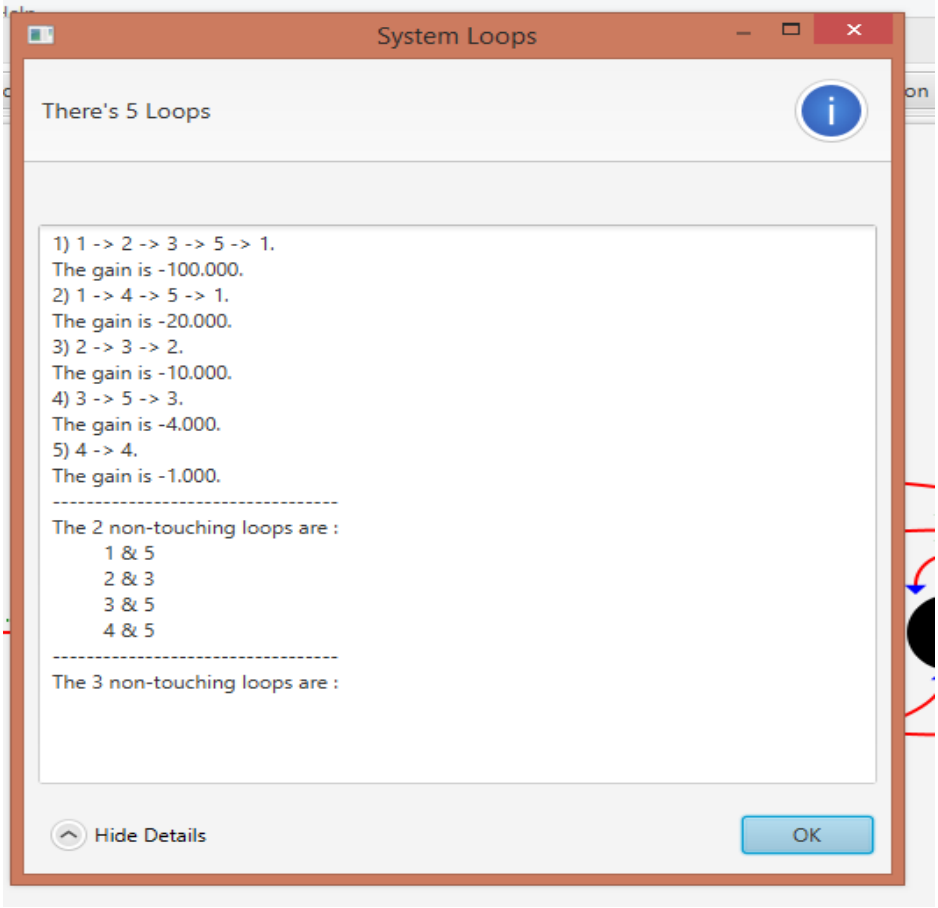
### i. Graph



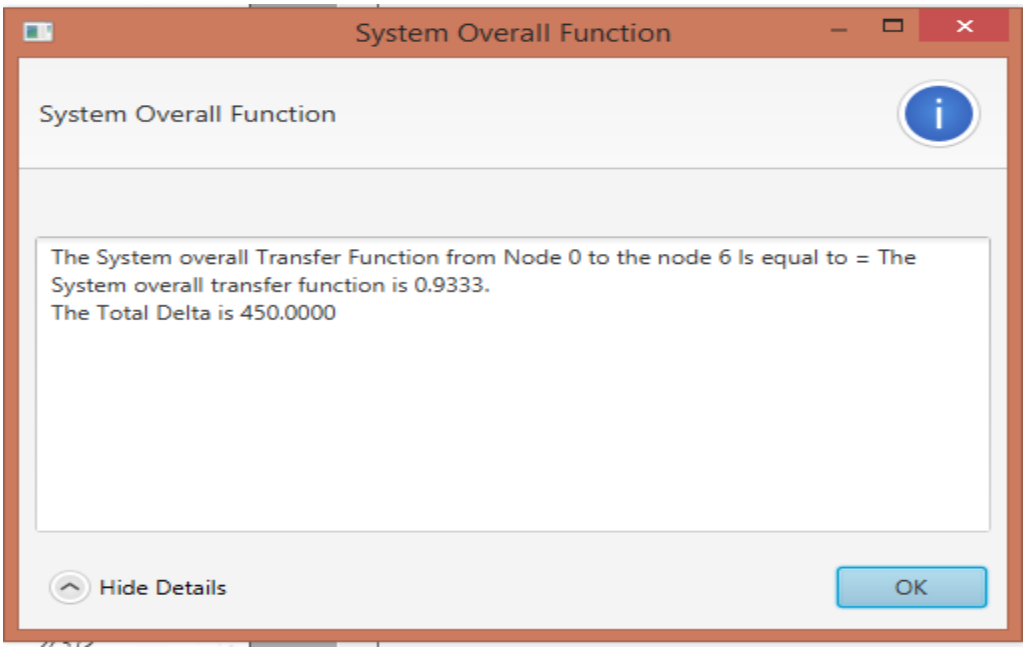

### ii. Forward Paths



## iii. Loops

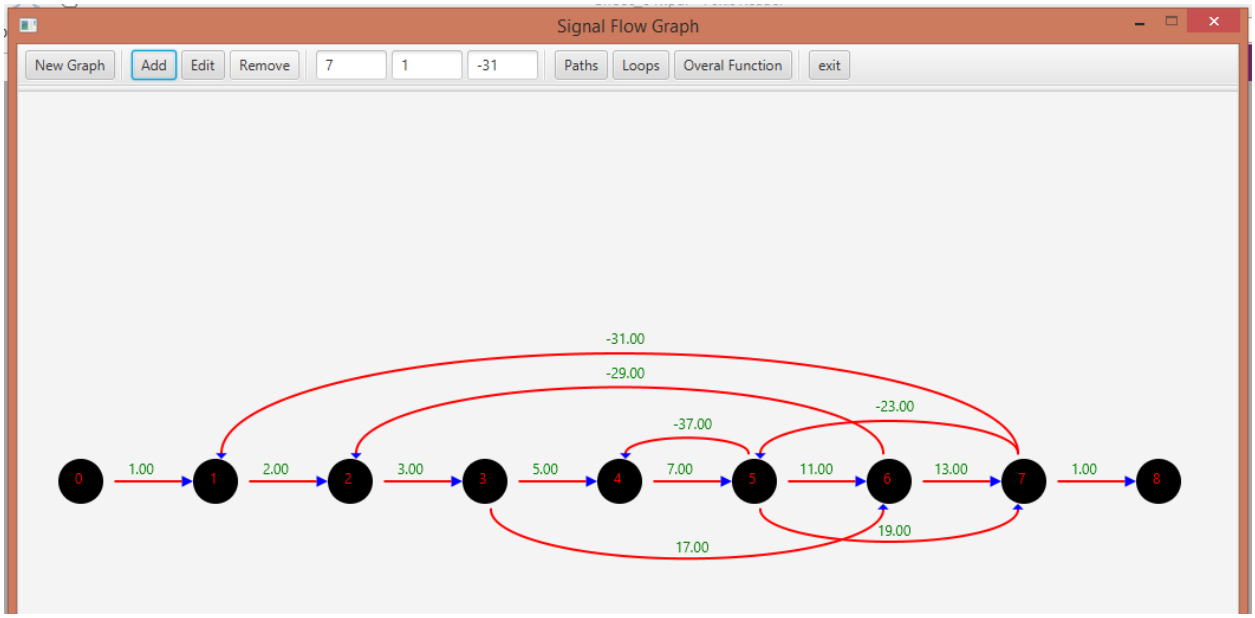



## iv. Output

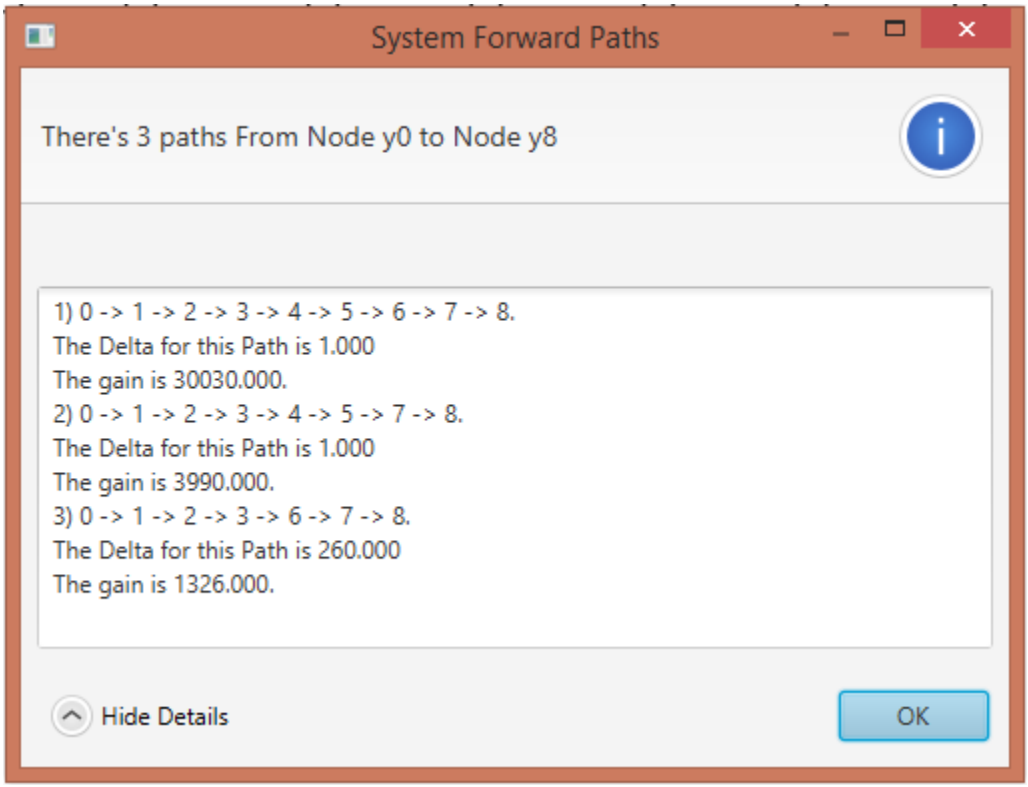


## b. Second

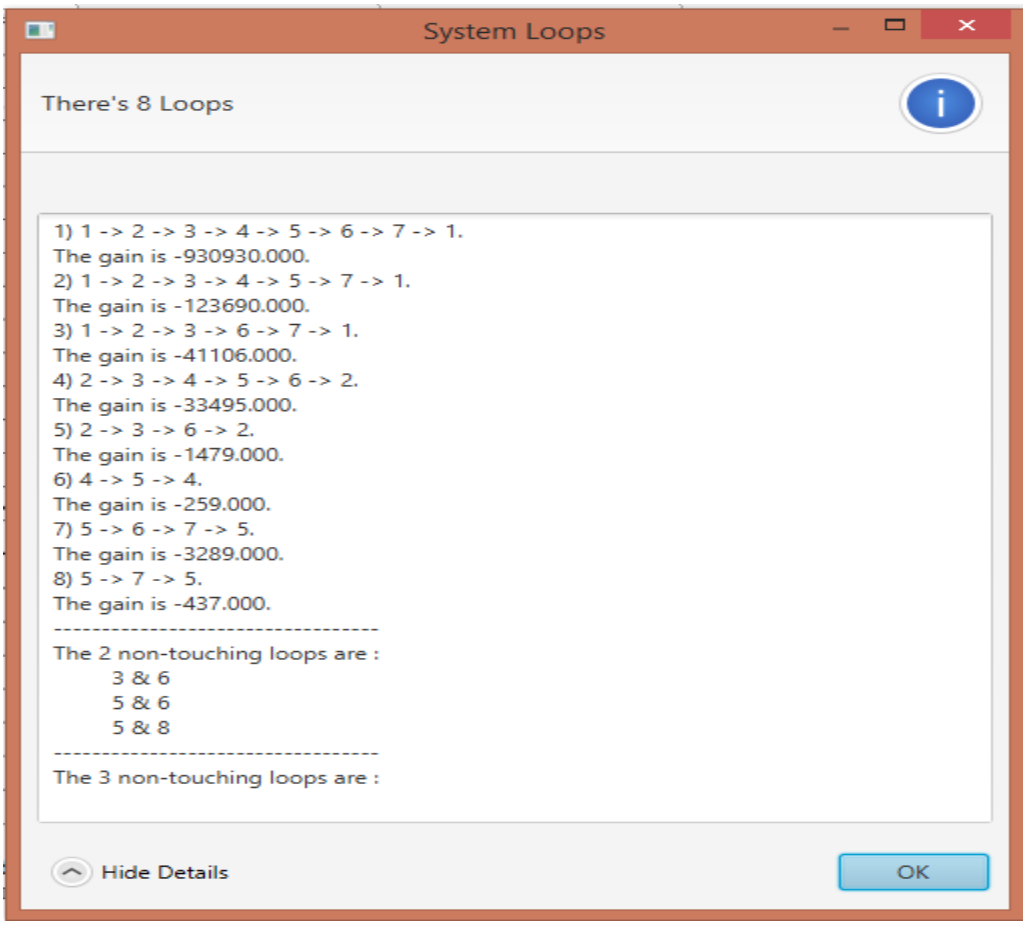
### i. Graph




### ii. Forward Paths



## iii. Loops




## iv. Output