

**Names: Elsayed Abdelnasser
Elsayed (12)**

Mohamed Samy

Course: Systems Control I

**Assignment: Signal Flow
Graph**

Problem Statement:

Inputs:

1. Total number of nodes.
2. Branches with gains.

Outputs:

1. Signal Flow graph.
2. Forward paths.
3. Individual loops.
4. All non-touching loops.
5. All values of delta.
6. Overall Transfer Function calculated by Mason's Formula.

Main Features:

1. Dynamic graph (e.i. the branch is added to the graph once you enter it)
2. Lists all the calculated gains –forward paths, loops, non-touching loops-
3. You can place the nodes on the graph anywhere for better visibility.
4. All gains are shown on the graph.

Data Structures:

The main-used data structure is Graph.

It is implemented with both Adjacency Matrix and Adjacency List representations to make the most of them.

The adjacency matrix is directed, but the adjacency list is not.

We have used Arrays, ArrayLists, LinkedLists and Stack throughout the program for storing data and implementing algorithms.

Main Modules:

Logic

1. GraphDS: implementing Graph Data Structure and responsible for performing almost all the algorithms used in Logic.
2. Mason: implementing the Mason's Formula and responsible for getting the final results.
3. SFG: initializing GraphStream Library ,which is used to draw the graph, and responsible for linking GraphDS, Mason and GraphStream altogether.
4. Path: holding the path and gain of forward paths or loops.
5. AdjacencyList, JohnsonCycles, SCCResult, StronglyConnectedComponents: implementing Johnson's algorithms and the helper Tarjan algorithms which find the loops and strongly connected components respectively.
6. NonTouchingLoops: Holding the non-touching loops.

Application:

Windows:

1. TotalNumberWindow: The first window appears and lets the user enter the number of nodes.
2. InputWindow: lets the user enter the branches.

3. **ResultWindow**: showing the result.

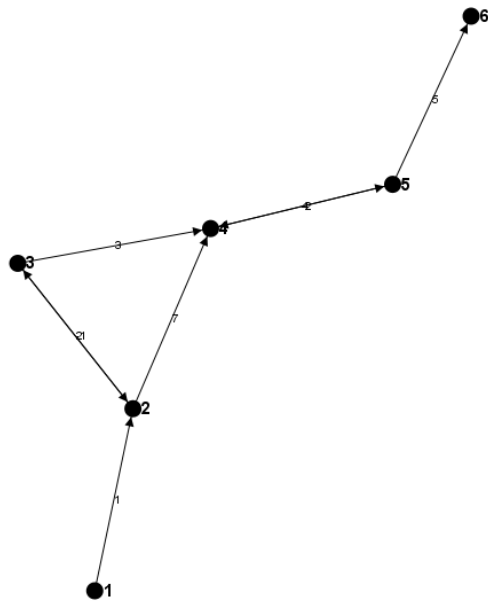
Classes:

1. **Main**: the main class of application.
2. **Controllers**: a class for every window to handle actions which the user performs.

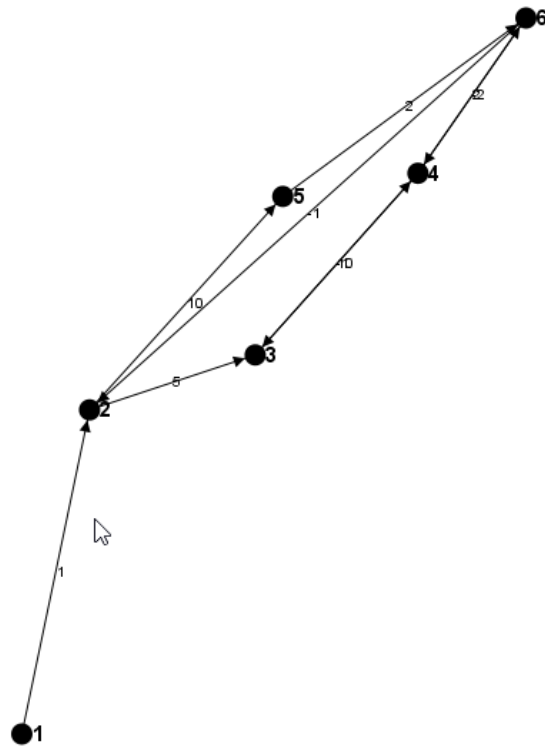
Algorithms:

1. **Modified DFS**: to find all the forward paths of the graph.
2. **Tarjan's**: to find the strongly connected components which are used in finding the loops.
3. **Johnson's**: to find loops.
4. **findTwoNonTouchingLoops**: A method to find all two non-touching loops.
5. **findNonTouchingLoops(int degree)**: A recursive method to find all three or more non-touching loops.

Sample runs:



<p>Forward Paths:</p> <p>1) 1 2 3 4 6 $G = 100$</p> <p>2) 1 2 5 6 $G = 20$</p>	<p>Loops:</p> <p>1) 2 3 4 6 2 $G = -100$</p> <p>2) 2 5 6 2 $G = -20$</p> <p>3) 3 4 3 $G = -10$</p> <p>4) 4 6 4 $G = -4$</p>	<p>NonTouchingLoops:</p> <p>2 5 6 2, 3 4 3, $G = 200$</p>
<p>Delta = 335</p> <p>Delta1 = 1</p> <p>Delta2 = 11</p> <p>Overall Transfer Function = 0.9552238805970149</p>		



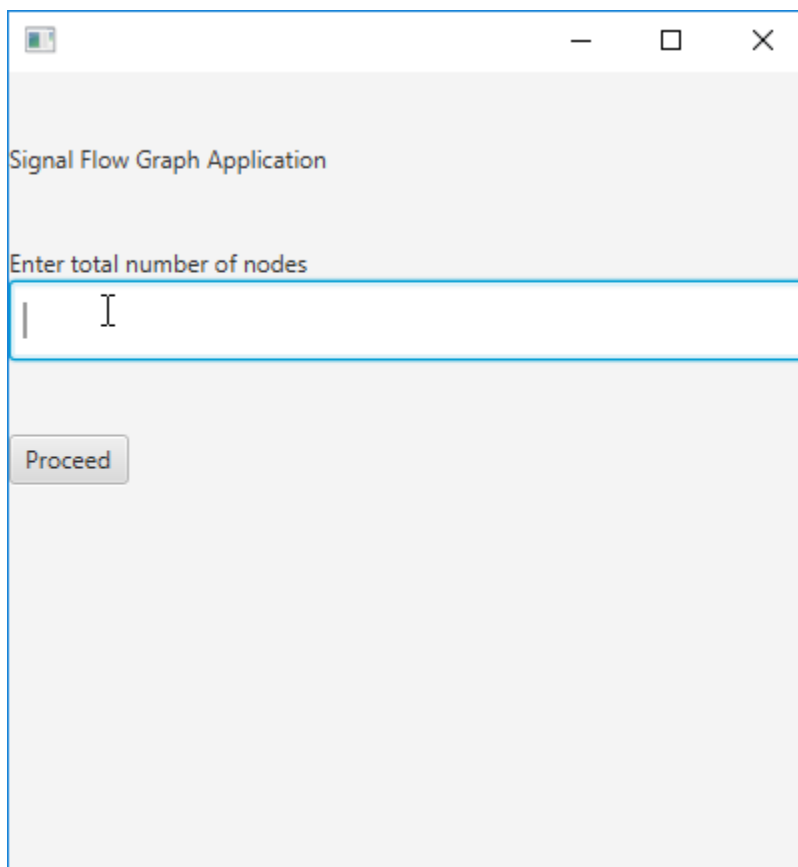
Forward Paths:	Loops:	NonTouchingLoops:
1) 1 2 3 4 5 6 G= 120	1) 2 3 2 G= -2	2 3 2 , 4 5 4 , G= 16
2) 1 2 4 5 6 G= 140	2) 4 5 4 G= -8	
Delta = 27 Delta1 = 1 Delta2 = 1 Overall Transfer Function = 9.62962962962963		

User Guide:

Assumptions:

- **Node 1 is the input node.**
- **Last node is the output node.**

1. Enter the total number of nodes.



The screenshot shows a window titled "Signal Flow Graph Application". Inside the window, there is a text label "Enter total number of nodes" above a text input field. The input field is currently empty, with a vertical cursor line at the beginning. Below the input field is a button labeled "Proceed".

2. Enter each branch and click on Add Branch Button

From: to:

Gain =

3. To show result, click the show result button.