Signal Flow Graph Solver

Project report

Name : Mosab Mohamed Khames ID : 68
Name : Ahmed Ibrahim Khattab ID : 7

Problem statement

Given:

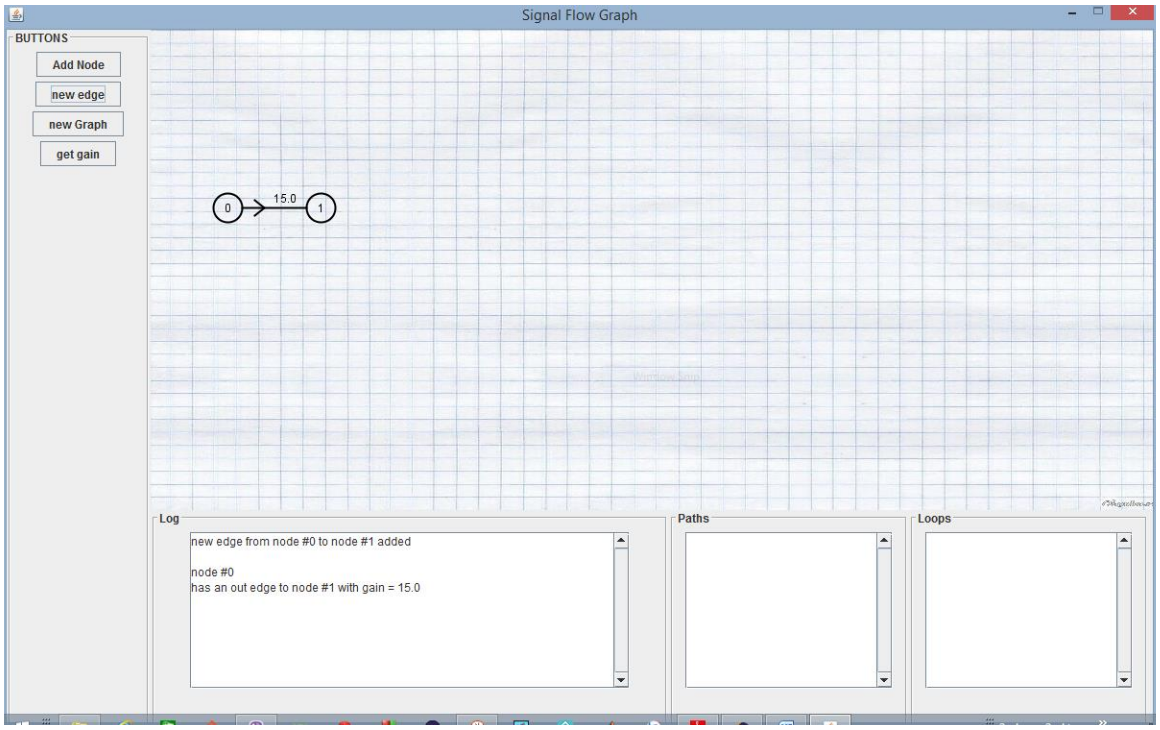
Signal flow graph representation of the system. Assume that total number of nodes and numeric branches gains are given.

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 , , …, where *m* is number of forward paths. □1 □m □
5- Overall system transfer function.

Main features

Program layout :



```
new edge from node #0 to node #1 added

node #0
has an out edge to node #1 with gain = 15.0
```

_ to make the program work in the most efficient way it is preferred to add nodes in an descending order from left to right..

_ the edges that cross above nodes is a forward edge (from left node to right node)

_ the edges that cross under nodes is a fedback edge (from right node to left node)

Main Modules

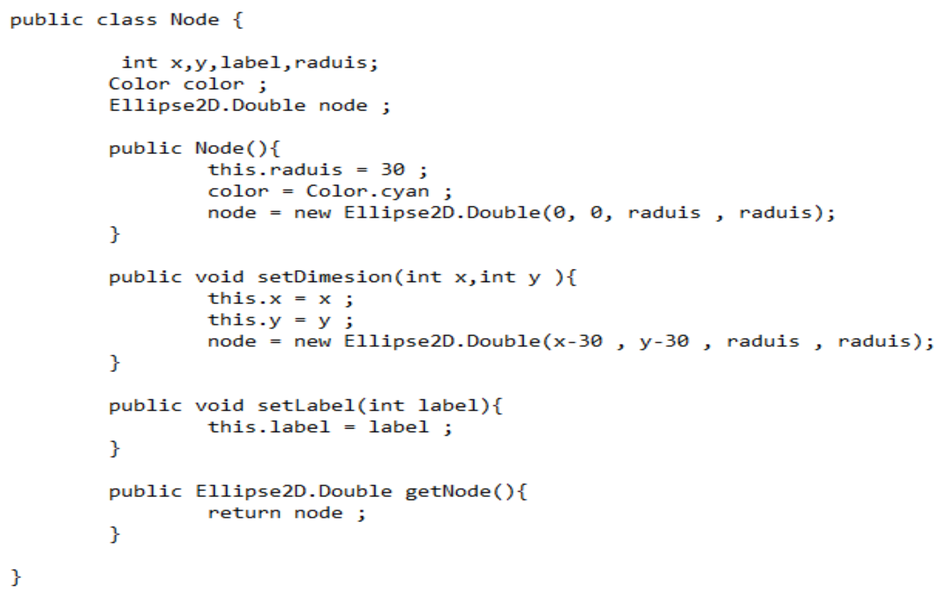
The program contains three modules:

1- GUI ➔ provides a graphical user interface to make the usage of the program easier.
2- Controller ➔ takes actions performed by the user and analysis them to choose the perfect instruction.
3- Transfer_Fun ➔ this is the factory that does everything in order to produce the overall transfer function.

Data Structures

The most data powerful data structures we used is ArrayList that we used to store paths, loops, edges and gains.

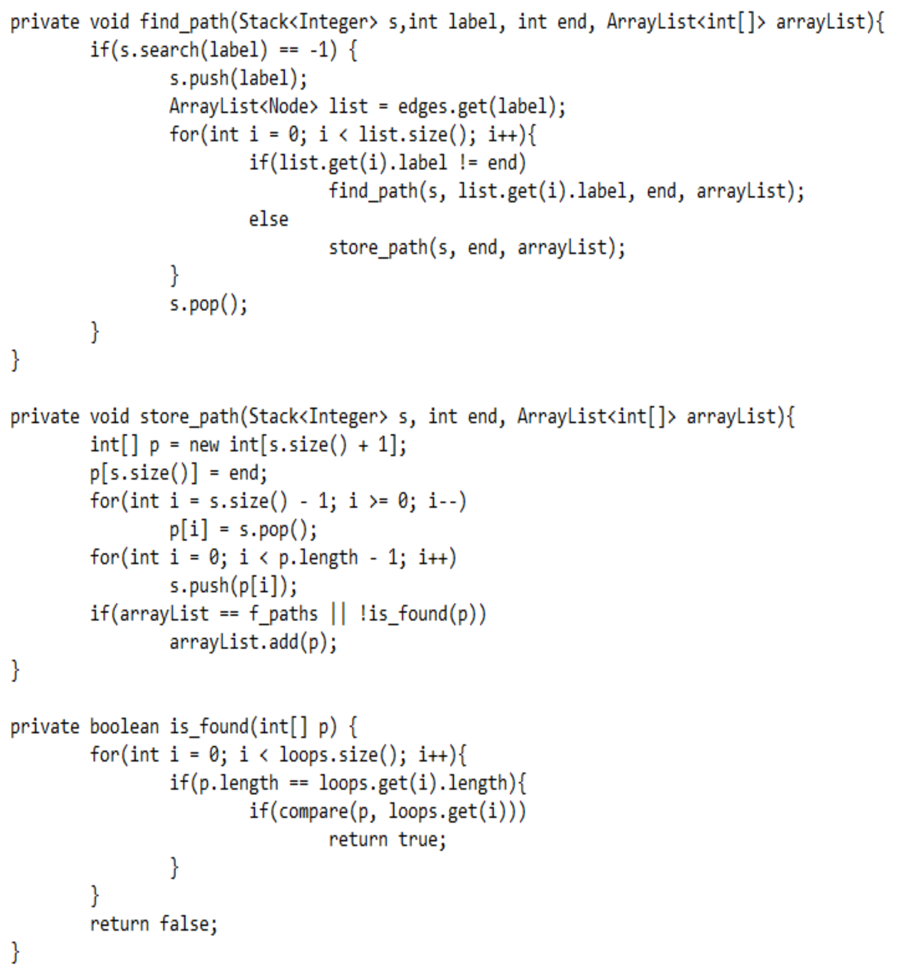
We used Node data structure to store nodes for drawing in GUI and to represent the graph of signal flow. We attach a screen shot of Node data structure.



```
public class Node {

         int x,y,label,raduis;
        Color color ;
        Ellipse2D.Double node ;

        public Node(){
                this.raduis = 30 ;
                color = Color.cyan ;
                node = new Ellipse2D.Double(0, 0, raduis , raduis);
        }

        public void setDimesion(int x,int y ){
                this.x = x ;
                this.y = y ;
                node = new Ellipse2D.Double(x-30 , y-30 , raduis , raduis);
        }

        public void setLabel(int label){
                this.label = label ;
        }

        public Ellipse2D.Double getNode(){
                return node ;
        }

}
```

Algorithms

The most important algorithm is DFS which we use to find both forward paths and loops. We also use it to find non-touch loops. I attach a screen shot of one of these algorithm from our source code that finds paths and loops.



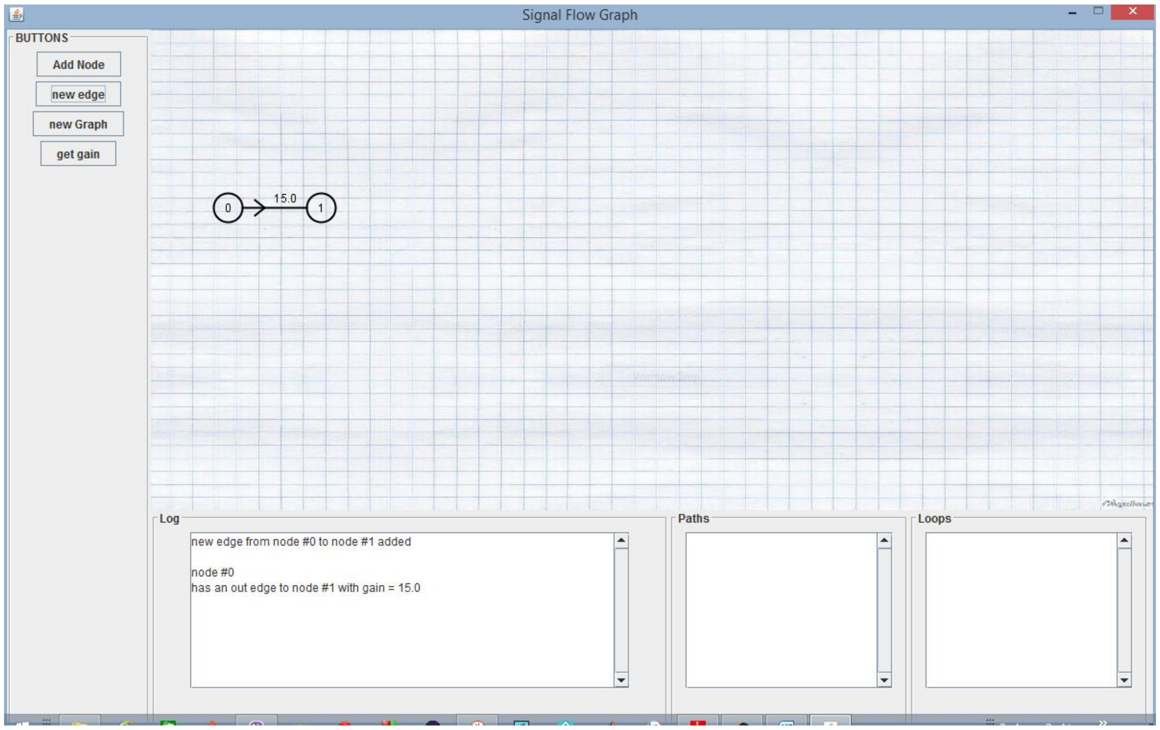
```java
private void find_path(Stack<Integer> s,int label, int end, ArrayList<int[]> arrayList){
        if(s.search(label) == -1) {
                s.push(label);
                ArrayList<Node> list = edges.get(label);
                for(int i = 0; i < list.size(); i++){
                        if(list.get(i).label != end)
                                find_path(s, list.get(i).label, end, arrayList);
                        else
                                store_path(s, end, arrayList);
                }
                s.pop();
        }
}

private void store_path(Stack<Integer> s, int end, ArrayList<int[]> arrayList){
        int[] p = new int[s.size() + 1];
        p[s.size()] = end;
        for(int i = s.size() - 1; i >= 0; i--)
                p[i] = s.pop();
        for(int i = 0; i < p.length - 1; i++)
                s.push(p[i]);
        if(arrayList == f_paths || !is_found(p))
                arrayList.add(p);
}

private boolean is_found(int[] p) {
        for(int i = 0; i < loops.size(); i++){
                if(p.length == loops.get(i).length){
                        if(compare(p, loops.get(i)))
                                return true;
                }
        }
        return false;
}
```

User Guide

To use our program for solving signal flow graphs, you should do the following procedures:

1- Draw the nodes.
2- Draw the edges.
3- Associate the gains for each edge.
4- Order to compute the transfer function.

These procedures can easily be done with our GUI, in the following picture you can see Add Node Button on the left. This button is used for drawing nodes. The New Edge button is used to add edges between nodes. The New Graph is for drawing a new graph.



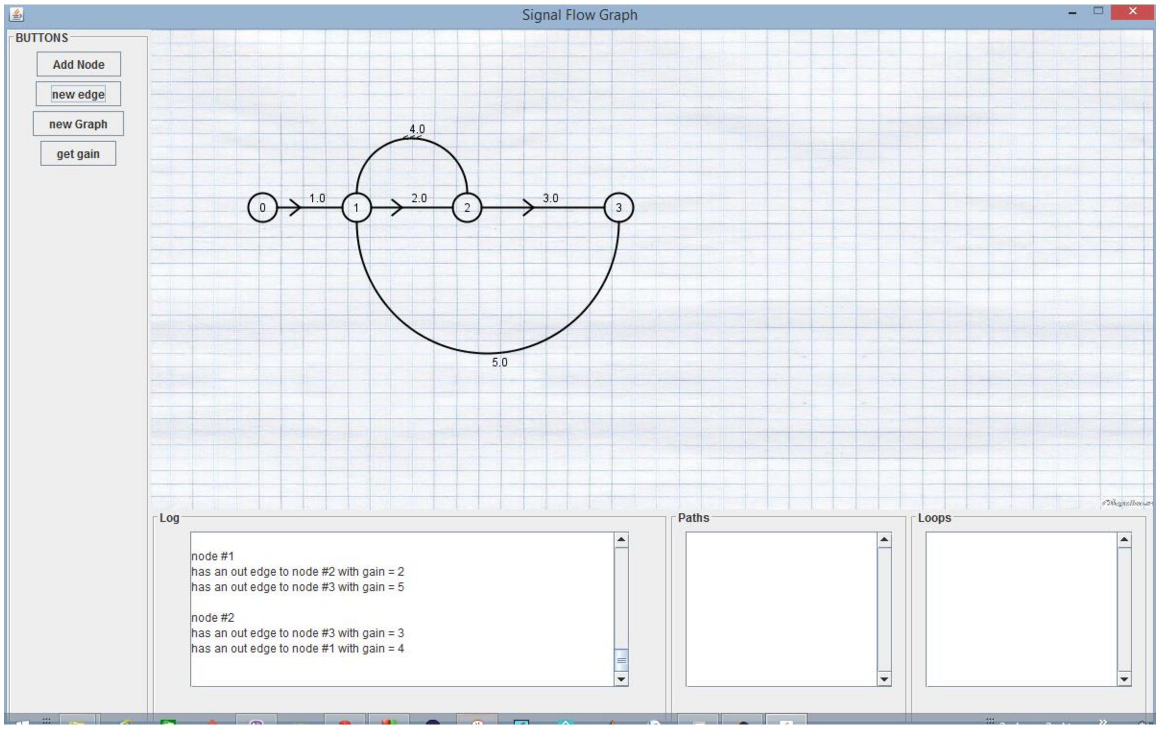
```
new edge from node #0 to node #1 added

node #0
has an out edge to node #1 with gain = 15.0
```

Note that if you need to draw a traversal edge, you will have above edge to indicate an edge from right to left and below edge to indicate an edge from left to right.

==

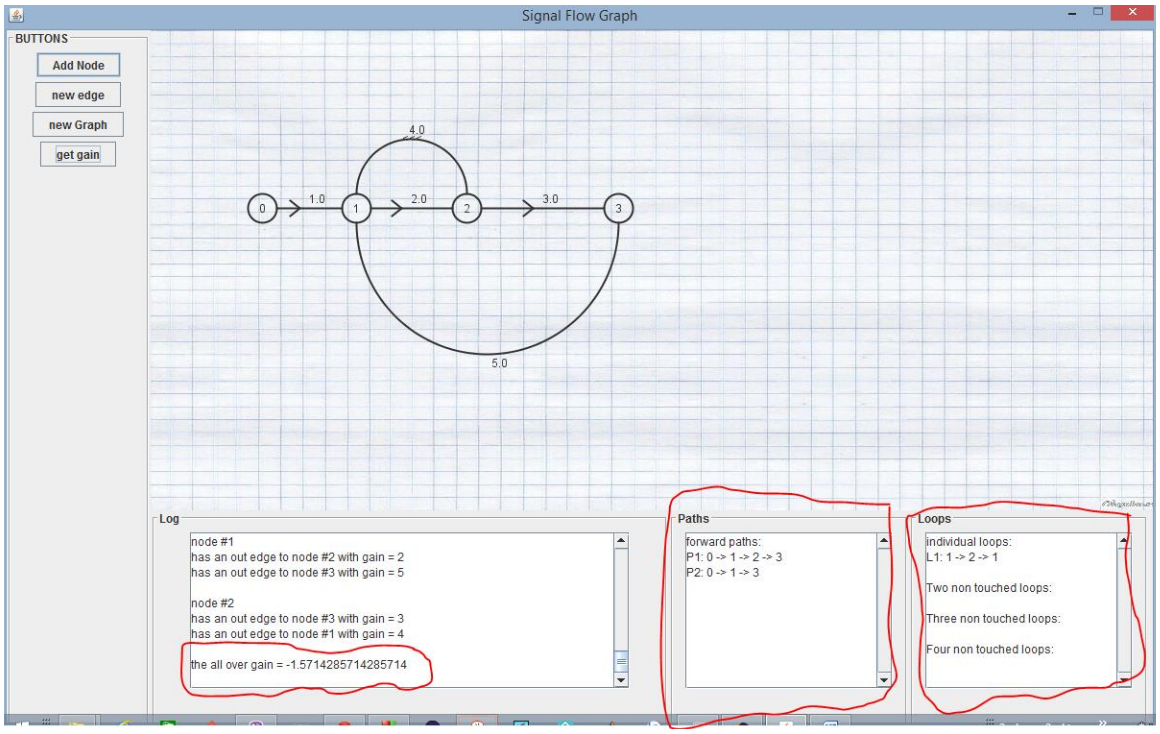
Sample runs:

Before calculating the gain..



```
node #1
has an out edge to node #2 with gain = 2
has an out edge to node #3 with gain = 5

node #2
has an out edge to node #3 with gain = 3
has an out edge to node #1 with gain = 4
```

After calculating the gain…



```
node #1
has an out edge to node #2 with gain = 2
has an out edge to node #3 with gain = 5

node #2
has an out edge to node #3 with gain = 3
has an out edge to node #1 with gain = 4

the all over gain = -1.5714285714285714
```

```
forward paths:
P1: 0 -> 1 -> 2 -> 3
P2: 0 -> 1 -> 3
```

```
individual loops:
L1: 1 -> 2 -> 1

Two non touched loops:

Three non touched loops:

Four non touched loops:
```