MAXIMUM FLOW-MINIMUM CUT

PROBLEM SOLUTIONS

Ozan CELIK

KOCAELI UNIVERSITY

COMPUTER ENGINEERING

ozancelikinfo@gmail.com

1.Summary

This project calculates the maximum flow from the source to the target in a system with flow. This calculation is made on the basis of inter-node capacities. The user can design this system with the interface and examine the solution. The data of the system is taken from the user through the interface.

A minimum cut solution is applied to cut the flow. This solution aims to cut the flow in the system by removing the fewest and least capacity edges. The result of this solution is displayed to the user on the interface. In the solution, the interface between which node nodes is removed to the user is specified on the interface.

Detailed information about the solutions will be explained in the method section.

2.Entrance

This project finds the max flow of a system. If this flow is desired to be interrupted, flow is interrupted with min cut. Examples of real world applications: Airline scheduling,Commu nication Network Flow, Circulation Demand problem Image segmentetion.

3.Basic Knowledge

The project is coded in Java language. NetBeans 8.2 IDE was used. Swing library of java was used for graphic design.

4.Methods

-public void valvePaint(Graphics g): It is the function that determines where the nodes are placed on the panel.

-public void Edge (Graphicsg): It is the method that draws the edges linking the node and shows the capacities of the edges, Retri eves the data from the Entry, Exit and Capacity fields on the panel.

-int maximumFlow(int graph[][], int source, int target): It is the method that calculates the maximum flow reaching the target within the system.BFS and Edmonds-Karp algorithms are used for this calculation.

BooleanbreadthFirstSearch(int[][]residualGraph, int source, int target, int[] parents) : Algorithm used to detect paths from source to destination

-void minimumCut(int[][] graph, int source, int target): It is the method that detects the edges that should be removed in order to cut the flow with the minimum number of edges and minimum capacity.DFS and Ford-Felkurson algorithm were used for this calculation.

-void depthFirstSearch(int[][] residualGraph, int source, boolean[] visited): Determines the visited and not visited nodes.

5.Pseudo Code

Run

Enter the number of nodes in the addNode field and click on button

```
{  Print nodes on the panel   and  Assign size of graph

 Graph[addNodeValue][ addNodeValue];

}
```

Enter the ID and capacity of the nodes and the amount of the edge in the input-output fields

```
{

Graph[EntryValue][ExitValue] = capacity

}
```

Enter source and target node.Click the max flow calculater button

```
{

 BreathFirstSearch{

 LinkedList <Integer> queueStruct = new LinkedList<>();

Visited[] = false;

 While(queueStruct.size != 0){

node1 = q.poll();

for(int node2 = 0 ; node2 != source ; node2 = parent[node2]){

if(visit==false  &&residualGraph[node1][node2]){

queueStruct.add(node2);

parents[node2] = node1;

visit[node2] = true;


return (visit[target] == true)

}

MaxFlow(){

İnt maxFlow = 0;

While(breathFirstSearch){

PthFlow=Math.min(pathFlow,graph[node1][node2]

Graph[node1][node2]-=pthFlow;

MaxFlow += pthFlow;

}

return maxFlow;

print max flow result

}
```

Enter source and target node.Click the min-cut calculater button

```
DepthFirstSearch(){

finds the visited nodes of the residual graf

visited[source] = true; }

MinimumCut(){

Boolean visited[] = false ;

DepthFirstSearch(graph,source,Visited)

İf(graph[node1][node2]>0 && visited[i] == true && visited[j] == false){

print node ID s of the edges to be removed

}
```

6.Big O Complexity

-Max Flow Complexity

M=Number of edges in the system
F=Max Flow value

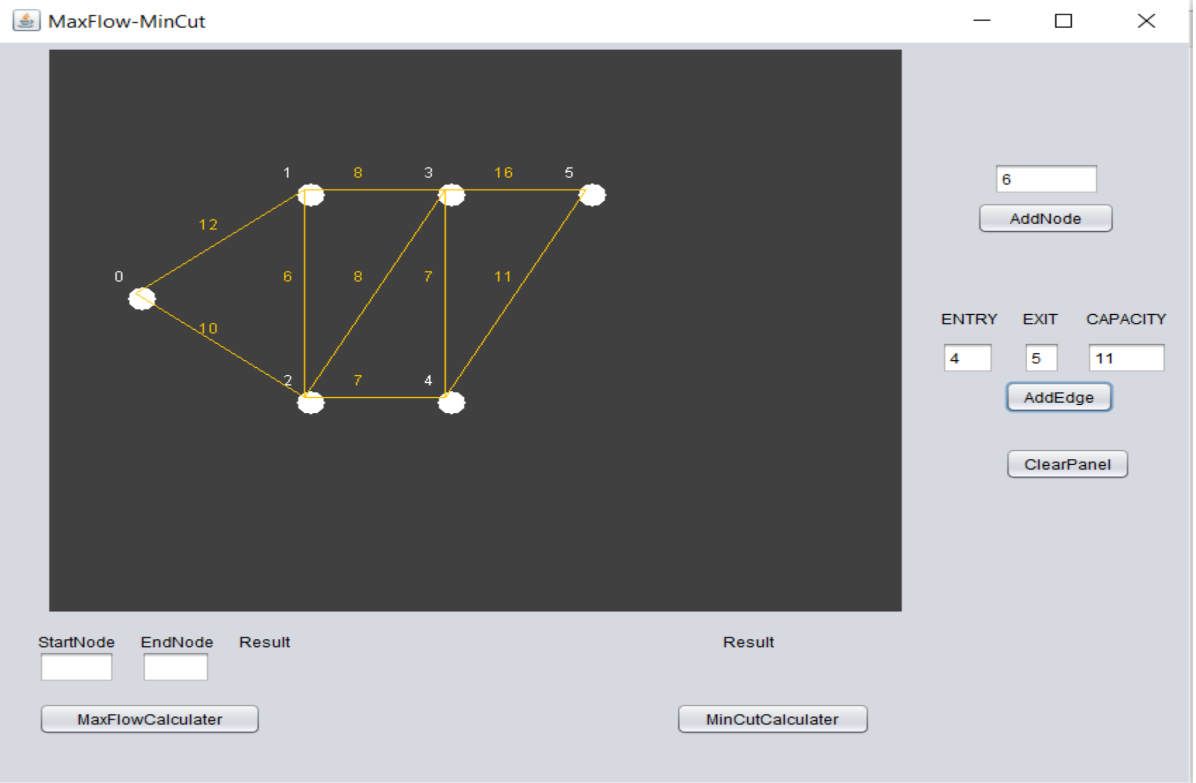
O(O*F) is Max Flow Complexity

-Min Cut Complexity

M = Number of edges in the system
N = Number of node in the system

$O(N*M*M) => O(NM^2)$

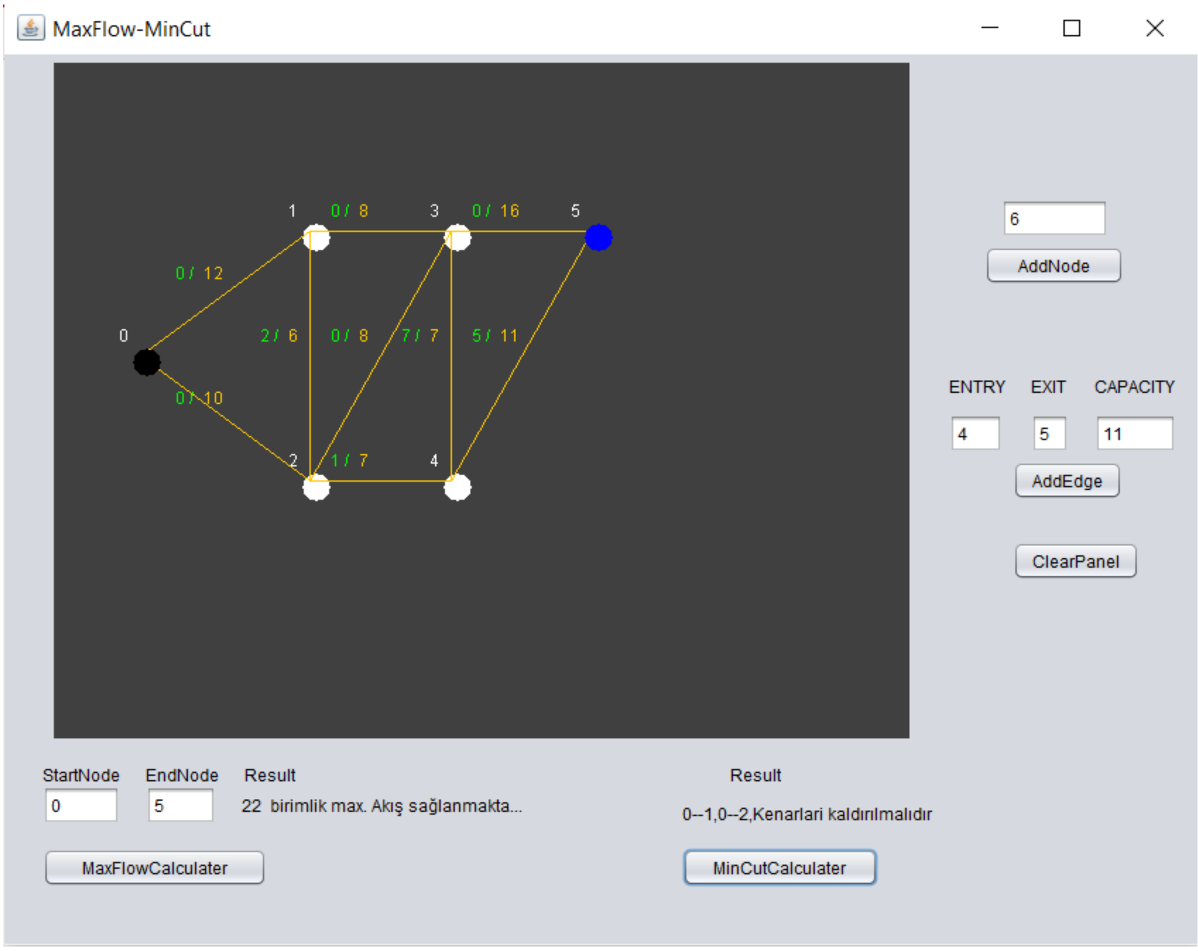
7.Experimental results



MaxFlow-MinCut

6

ENTRY: 4
EXIT: 5
CAPACITY: 11



MaxFlow-MinCut

6

ENTRY: 4
EXIT: 5
CAPACITY: 11

StartNode: 0
EndNode: 5

Result: 22 birimlik max. Akış sağlanmakta...

Result: 0--1,0--2,Kenarlari kaldırılmalıdır

8.Results

In this project, the solution processes of the systems with this problem have been realized by using the targeted max flow and min cut operations. With this project, experience in algorithm analysis has been obtained

9. References

[1]http://bilgisayarkavramlari.sadievrenseker.com/2010/05/22/edmonds-karp-algoritmasi/

[2]https://www.geeksforgeeks.org/ford-fulkerson-algorithm-for-maximum-flow-problem/

[3]https://www.youtube.com/watch?v=GiN3jRdgxU4

[4]https://www.youtube.com/watch?v=Gtyq4_UYIOA