# 1 UVA Problem 11506: Angry Programmer

**Background**

You, a programmer of an important software house, have been fired because you didn't solve an important problem that was assigned to you. You are very furious and want to take revenge on your boss, breaking the communication between his computer and the central server.

The computer of your boss and the central server are in the same network, which is composed of many machines (computers) and wires linking pairs of those machines. There is at most one wire between any pair of machines and there can be pairs of machines without a wire between them.

To accomplish your objective, you can destroy machines and wires, but you can't destroy neither the computer of your boss nor the central server, because those machines are monitored by security cameras. You have estimated the cost of blowing up each machine and the cost of cutting each wire in the network.

You want to determine the minimum cost of interrupting the communication between your boss' computer and the central server. Two computers $A$ and $B$ can communicate if there is a sequence of undestroyed machines $x_1, ..., x_n$ such that $x_1 = A, x_n = B$ and $x_i$ is linked with $x_{i+1}$ with an uncut wire (for each $1 \leq i \leq n-1$).

**Input**

The input consists of several test cases. Each test case is represented as follows:

- A line with two integers $M$ and $W$ ($2 \leq M \leq 50, 0 \leq W \leq 1000$), representing (respectively) the number of machines and the number of wires in the network.
- **M-2** lines, one per machine (different from the boss' machine and the central server), containing the following information seperated by spaces:
  - An integer $i$ ($2 \leq i \leq M-1$) with the identifier of the machine. Assume that the boss' machine has id 1 and that the central server has id $M$.
  - An integer $c$ ($0 \leq c \leq 100000$) specifying the cost of destroying the machine.
- **W** lines, one per wire, containing the following information separated by spaces:
  - Two integers $j$ and $k$ ($1 \leq j<k \leq M$) specifying the identifiers of the machines linked by the wire. remember that the wire is bidirectional.
  - An integer $d(0 \leq d \leq 100000)$ specifying the cost of cutting the wire.

The end of the input is specified by a line with the string "0 0".
Suppose that the machines have distinct identifiers.

## 1.1 Mathematical Formulation

Given an input of $M$ machines, $M-2$ of which are potential intermediate nodes between the boss' computer and the server, and $W$ wires connecting computers $m_i$ and $m_j$ where $0 \leq i<j \leq M$, determine the minimum cost to destroy (m + w) machines and wires where $(0 \leq m \leq M-2, 0 \leq w \leq W)$.

## 1.2 Solution

Important Confusing Data Structures:

- LinkedList<FlowEdge>[ ] **adj** : An adjacency list of all FlowEdges where the specified node is the originating node.

The main functionality of this algorithm is to implement the Ford Fulkerson min cut algoritm by first building a flow network and then executingthe algorithm on it. We represent the network as each machine (index) has its own linked list of Flow Edges. Each computer has 2 nodes, one as its input and the other as its output. The only edge that flows out of the input and into the output is the edge connecting the two such that the input points to the output with edgeweight equivalent to the cost of destroying that computer; this is the MAX_INTEGER for the boss' computer and the server since we cannot destroy these. Since we are given the boss' computer as "1" and the server as "M" we just let the input for every computer be index = i-1 and their output as index = (i-1)+M for $i \in 1..M$ as described below.

**Algorithm 1** Determine index of Computers

```
procedure detIndex(int index, boolean out)
    if out then return (index - 1) + m
    return (index - 1)
```

Once we do this we have to ensure that the bi-directional rule is accounted for. So, for each wire connecting computers A and B with cost C, we will connect the output for A to the input for B with weight C and the output of B to the input of A with cost C. Once we have built up the entirety of our network we will simply use the Ford Fulkerson algorithm on the built network and print that out. (As outlined below)

**Algorithm 2** Build the Flow Network from input

```
procedure MAIN(Scanner input)
    while true do
        $m, w \leftarrow$ from input
        if m == 0 && w == 0 then break;
        if w == 0 then
            PRINT(0)
            for m-2 times do input.NEXTLINE( )
        $intnumV, serverIn \leftarrow$ 2*m, m-1 respectively
        $adj[numV] \leftarrow$ initialized
        // create computers
        adj[0].ADD(FlowEdge(0, m, Integer.MAX_VALUE))
        adj[serverIn].ADD(FlowEdge(serverIn, numV-1, Integer.MAX_VALUE))
        for length of m-2 do
            $inti, c \leftarrow$ index of machine and cost to destroy
            $FlowEdgee \leftarrow$ new FlowEdge(i, i+m, c)
            adj[i].ADD(e)
        // connect computers to eachother
        for length of W do
            $inta, b, c \leftarrow$ comp1, comp2, cost to cut wire
            $outA, inA \leftarrow$ detIndex(a, true), detIndex(a, false)
            $outB, inB \leftarrow$ detIndex(b, true), detIndex(b, false)
            $e1, e2 \leftarrow$ FlowEdge(outA, inB, c), FlowEdge(outB, inB)
        PRINT(calcMinCut())
```

## 1.3 Correctness

**Proposition 1.**
*This is the correct flow Network to build which will yeild the min cut.*

*Proof.*
This is correct because we are using the "flow" of each edge to be the cost of destroying either a wire or a computer. We account for the fact that if we destroy a computer, then we will also sever any connection that used this computer as an intermediate node. We have done this by seperating each computer into an input and an output node, between which $\exists$ only one connection which has a flow of the cost of destroying a computer. The bi-directional aspects has us connect the network such that, for each wire connecting computers A and B with cost C, we will connect the output for A to the input for B with weight C and the output of B to the input of A with cost C. Therefore if we sever inA to outA (destroying computer A) B would no longer be able to use this as an intermedite node since inA no longer has any out edges and outA is unreachable since its only in edge was just severed. □

## 1.4 Analysis

For the following analysis, we will say that we have built a flow network graph with $V = 2 \cdot M$ and $E = M + 2 \cdot W$. Our Time and Space analysis will be done with these variables but can be converted to V ≈ M and E ≈ (M+W)

**Proposition 2.** *The space complexity of this algorithm is* ***O(V+E)***

*Proof.*
This is due to the fact that all of our data is stored in data structures:

- adj[V]: Our Flow NetWork $\implies V + E$
- edgeTo[V]: Used in Ford Fulkerson implementation

Giving us a space complexity of **O(V+E)**

□

**Proposition 3.** *The time complexity of this algorithm is* ***O(V+E+MaxFlowValue·E)***

*Proof.*
This is the case because our algorithm first builds our flow network which takes V + E time and then implements the Ford Fulkerson Min Cut algorithm which is synonomous to the Max Flow. This implementation takes MinCutValue·E time.

Giving us a time complexity of **O(V+E+MaxFlowValue·E)**

□

## 1.5 An Example

**input**
4 4
3 5
2 2
1 2 3
1 3 3
2 4 1
3 4 3
0 0

Now when we run this through our Algorithm we will build first the nodes for the computers and connect their ”ins” to their ”outs”

$$\begin{aligned}\text{(boss)} &\implies 0 \to 4 \text{ with cap} = 2.147483647 \cdot 10^9\\ \text{(server)} &\implies 3 \to 7 \text{ with cap} = 2.147483647 \cdot 10^9\\ \text{(3 5)} &\implies 2 \to 6 \text{ with cap} = 5.0\\ \text{(2 2)} &\implies 1 \to 5 \text{ with cap} = 2.0\end{aligned}$$

Next we build the wire connections:

$$\begin{aligned}
(1\ 2\ 3) &\implies 4 \to 1 \text{ with cap} = 3.0 \\
\text{convex} &\implies 5 \to 0 \text{ with cap} = 3.0 \\
\\
(1\ 3\ 3) &\implies 4 \to 2 \text{ with cap} = 3.0 \\
\text{convex} &\implies 6 \to 0 \text{ with cap} = 3.0 \\
\\
(2\ 4\ 1) &\implies 5 \to 3 \text{ with cap} = 1.0 \\
\text{convex} &\implies 7 \to 1 \text{ with cap} = 1.0 \\
\\
(3\ 4\ 3) &\implies 6 \to 3 \text{ with cap} = 3.0 \\
\text{convex} &\implies 7 \to 2 \text{ with cap} = 3.0
\end{aligned}$$

Then when we calculate the mincut value we get 4