**RANKFRAUD PROBLEM**

**Assignment Report**

Submitted By-

Angshuman Das (IMT2015008)

Ankit Gahlawat (IMT2015010)

**PROBLEM STATEMENT** : For ranking the ministers based on their ability to play TT the ministers are to be organised in a list such that every minister had beaten the minister the next minister in the list.

For instance, suppose that the ministers were numbered 1, 2, 3, 4 and 5 and the results of the competition were as follows:

* Minister 1 beat ministers 2 and 4.
* Minister 2 beat minister 5.
* Minister 3 beat ministers 1 and 2.
* Minister 4 beat ministers 2, 3 and 5.
* Minister 5 beat ministers 1 and 3.

Then, the prime minister could present the following list to the king,

       4   3   1   2   5

**SOURCES REFERRED**: We have used the following link “<https://pdfs.semanticscholar.org/4f96/d6c69c13b45510b0cc5ec61e6f4768403124.pdf>” for learning about how to effectively find a hamiltonian path in a tournament graph.

We have implemented the method which is similar to insertion sort.

**DATA STRUCTURES WE USED** : We used a 2-d vector to store the input as a graph.

For the final answer we have used a vector of the size twice that of input. We started adding from the middle of the vector, so that if something is to be added in the beginning of the vector we can do it without disturbing the entire vector.

**BRIEF DESCRIPTION** :

We have used Adjacency Matrix to store the ministers as vertices. This turns out to be a tournament graph as there is a relation between one vertex and every other vertices.

Now the edge between vertices V1 to V2 exists if V1 has beaten V2 in the match played between them. So finally the problem is finding a simple Hamiltonian path from the graph created. As we have a tournament graph, it can be solved in O(n^2) time.

To store the result, we have used a vector.

**PSEUDO CODE** :

The pseudo code for algorithm is:

store the input in a 2-d vector;

start the answer sequence with 1 (or 2, if 1 has not beaten any minister);

for(i = 1 to n){

check if

i can be put in the beginning of the sequence and if yes, put it;

else if

check if it can be put at the end and if yes, put it;

else

for(j=1 to n){

check for appropriate j where we can insert i;

break;

}

}

**SIDE-EFFECTS**: The problem with this question is that it can have multiple correct sequences. So although it is running very well for all of our input cases, it is not getting a score of more than 64 in the website.

The only reason we could deduce was that they may not have stored all different output combinations which are possible causing our output to not match with theirs. The sequence that our code generates is absolutely correct.

**Instructions on how to compile and run the code**:

Go to terminal and type makefile

It will run the rankfraud.cpp file and wait for the input.

We have used cin and cout functions so that everything is in control in the terminal.

So, copy paste any input there and it will print out the result.

**INDIVIDUAL CONTRIBUTIONS** :

Ankit Gahlawat (IMT2015010) - Finding the correct algorithm and making input test-cases.

Writing the code for the algorithm.

Angshuman Das (IMT2015008) - Finding the correct algorithm and writing the documentation.

Writing the code for getting the input.