**Project CSE-200**

Name: Fazle Rabby Sourav

Reg. No: 2010331033

Email to login lightOj : [sourav.sust.cse.10@gmail.com](mailto:sourav.sust.cse.10@gmail.com)

Password to Login lightOJ : akilipip

Format I have used:

I have given my solution method’s description and complexity analysis with my .cpp file of solution of particular problem. For convenience I have also given problem statement at the end of the code in .cpp file. And Then I have also created a Doc file. I have skipped precode/template in Code.

**Problem Name: 1184 - Marriage Media**

/\*

OJ: Light OJ

Name Fazle Rabby Sourav

Problem Link: http://www.lightoj.com/volume\_showproblem.php?problem=1184

Problem Name: 1184 - Marriage Media

Rank: 61

\*/

/\*

Solution Method:

This is an elementary bipartite matching problem with some constrains. we should think a man or woman as a single node. and an edge between them

means they can marry/ be a couple. first , we will set up every possible edge between all nodes. for that we will fulfil the given condition and constrains.

if their age difffernce is less than or equal to 5, and if their height difference is less than or equal to 12 inches then we will create a edge between them.

And ant last we simply run a BP algo on the created graph. thus we can get maximum number of matching or marriage.

\*/

/\*

Complexity Analysis:

The Runnig time of DFS is O(E+V) where E is the number of edges and V is is the number of Vertices. Here number of edges are m\*w where m is the

number of man and w is the number of women. and the dfs is called m times, Therefore , the complexity of the algorithm will be O(m\*(m\*w));

or approximately O(n^3)

\*/

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Code\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

struct homoSapiens

**{**

int height**,** age**;**

int divorce**;**

homoSapiens**(**int h**,** int a**,** int d**)**

**{**

height**=**h**;** age**=**a**;** divorce**=**d**;**

**return;**

**}**

**};**

vector **<** homoSapiens **>** man **,** woman**;**

vi adj**[**MAX**];**

int par**[**MAX**];**

bool color**[**MAX**];**

bool dfs**(**int from**)**

**{**

int to**,** i**;**

**if(**color**[**from**])** **return** **false;**

color**[**from**]=**1**;**

**for(**i**=**0**;** i**<**SZ**(**adj**[**from**]);** i**++)**

**{**

to**=** adj**[**from**][**i**];**

**if(**par**[**to**]==-**1 **||** dfs**(**par**[**to**])==**1**)**

**{**

par**[**to**]=**from**;**

**return** **true;**

**}**

**}**

**return** **false;**

**}**

int BPM**()**

**{**

int i**,** j**,** k**,** cnt**=**0**;**

mset**(**par**,** **-**1**);**

**for(**i**=**0**;** i**<**SZ**(**man**);** i**++)**

**{**

CLR**(**color**);**

**if(**dfs**(**i**))**

cnt**++;**

**}**

**return** cnt**;**

**}**

int main**()**

**{**

//READ("in.txt");

//WRITE("out.txt");

int i**,** j**,** k**,** result**,** t**=**0**,** tcase**,** m**,** w**,** h**,** a**,** d**;**

cin**>>**tcase**;**

**while(**tcase**--)**

**{**

man**.**clear**();** woman**.**clear**();**

cin**>>**m**>>**w**;**

**for(**i**=**0**;** i**<**m**;** i**++)**

**{**

scanf**(**"%d %d %d"**,** **&**h**,** **&**a**,** **&**d**);**

man**.**pb**(**homoSapiens**(**h**,** a**,** d**));**

**}**

**for(**i**=**0**;** i**<**w**;** i**++)**

**{**

scanf**(**"%d %d %d"**,** **&**h**,** **&**a**,** **&**d**);**

woman**.**pb**(**homoSapiens**(**h**,** a**,** d**));**

**}**

**for(**i**=**0**;** i**<**m**;** i**++)**

**{**

**for(**k**=**0**;** k**<**w**;** k**++)**

**{**

**if(** **((**abs**(**man**[**i**].**height**-**woman**[**k**].**height**))<=**12**)** **&&** **((**abs**(**man**[**i**].**age**-**woman**[**k**].**age**))<=**5**)** **&&** **(**man**[**i**].**divorce**==**woman**[**k**].**divorce**)** **)**

**{**

adj**[**i**].**pb**(**k**);**

**}**

**}**

**}**

result**=** BPM**();**

csprint

printf**(**"%d\n"**,** result**);**

**for(**i**=**0**;** i**<**MAX**;** i**++)** adj**[**i**].**clear**();**

**}**

**}**

/\*

Problem Statement:

You run a marriage media. You take some profiles for men and women, and your task is to arrange as much marriages as you can. But after reading their bio-data you have found the following criteria.

1. No man will marry a woman if their height gap is greater than 12 inches.

2. No woman will marry a man if their age gap is greater than 5 years.

3. A couple can be formed if either both are not divorced or both are divorced.

4. Of course, a man can marry a single woman and vice versa.

Now you are given the bio-data of some men and women, you have to arrange the maximum number of marriages considering the given criteria.

Input

Input starts with an integer T (= 200), denoting the number of test cases.

Each case contains two integer m, n (1 = m, n = 50). Each of the next m lines will contain the information for a man, and each of the next n lines will contain the information for a woman. An information will contain three integers denoting the height in inches, age in years and 1 or 0 depending on they are divorced or not respectively. Assume that Height will be between 50 and 80, age will be between 20 and 50.

Output

For each case, print the case number and the maximum number of marriages you can arrange.

Sample Input

Output for Sample Input

2

2 2

70 30 0

60 20 0

71 25 0

71 35 0

1 1

70 30 1

70 30 0

Case 1: 2

Case 2: 0

\*/

**Problem Name: 1206 - Scheduling Taxi Cabs**

/\*

OJ: Light OJ

Name Fazle Rabby Sourav

Problem Link: http://www.lightoj.com/volume\_showproblem.php?problem=1206

Problem Name: 1206 - Scheduling Taxi Cabs

Rank: 20

\*/

/\*

Solution Method:

in this problem we need to find minimum number of taxi neede to give service to all the customer. we will pre calculate the the time need to

give his own service for each taxi. for each taxi we have data of starting time and service time. now we check for all taxi that either it is

possible to complete his own service and reach another customer before the customers starting time.

if it is possible then we will connect both nodes[taxi] with an edge. by this way we get a graph with some nodes and edges.

then we can run DFS from each taxi/nodes to find maximum matching numbers.

in this case maximum matching number is the number of extra customer who are given service by the taxi, which has given one or more service to another customers.

so we will substract the maximum matching number from the number of taxi. that is minumum number of taxi neede to satisfy all the customers

\*/

/\*

Complexity Analysis:

the complexity of the solution is approximately O(n\*(n^2)) or O(n^3)

\*/

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Code\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

***//Have skipped precode***

struct taxi**{**

int h**,** m**,** x1**,** x2**,** y1**,** y2**,** st**,** tym**;**

taxi**(**int \_h**,** int \_m**,** int \_x1**,** int \_y1**,** int \_x2**,** int \_y2**)**

**{**

h**=** \_h**;** m**=** \_m**;** x1**=** \_x1**;** y1**=** \_y1**;** x2**=** \_x2**;** y2**=** \_y2**;**

st**=** **(**h**\***60**)+** **(**m**);**

tym**=** abs**(**x1**-**x2**)+**abs**(**y1**-**y2**);**

**}**

**};**

vector**<** taxi **>** arr**;**

vi edge**[**MAX**];**

int par**[**MAX**];**

bool col**[**MAX**];**

bool dfs**(**int from**)**

**{**

**if(**col**[**from**])** **return** **false;**

col**[**from**]=**1**;**

int i**,** to**;**

REP**(**i**,** SZ**(**edge**[**from**]))**

**{**

to**=** edge**[**from**][**i**];**

**if(**par**[**to**]==-**1 **||** dfs**(**par**[**to**]))**

**{**

par**[**to**]=** from**;**

**return** **true;**

**}**

**}**

**return** **false;**

**}**

int BPM**()**

**{**

int i**,** cnt**=**0**;**

mset**(**par**,** **-**1**);**

REP**(**i**,** SZ**(**arr**))**

**{** CLR**(**col**);**

**if(**dfs**(**i**))**

**{**

cnt**++;**

**}**

**}**

**return** cnt**;**

**}**

int main**()**

**{**

//READ("in.txt");

//WRITE("out.txt");

int i**,** j**,** k**,** result**,** t**=**0**,** tcase**,** m**,** cnt**,** x1**,** y1**,** x2**,** y2**,** minute**,** hour**;**

cin**>>**tcase**;**

**while(**tcase**--)**

**{**

cin**>>**n**;**

REP**(**i**,** n**)**

**{**

scanf**(**"%d:%d %d %d %d %d"**,** **&**hour**,** **&**minute**,** **&**x1**,** **&**y1**,** **&**x2**,** **&**y2**);**

taxi dum**=** taxi**(**hour**,** minute**,** x1**,** y1**,** x2**,** y2**);**

arr**.**pb**(**dum**);**

**}**

REP**(**i**,** SZ**(**arr**))**

**{**

**for(**k**=** i**+**1**;** k**<**SZ**(**arr**);** k**++)**

**{**

int tym\_go**=** **(**abs**(**arr**[**i**].**x2**-**arr**[**k**].**x1**)+**abs**(**arr**[**i**].**y2**-**arr**[**k**].**y1**));**

**if(**arr**[**i**].**st**+**arr**[**i**].**tym **+** tym\_go **<** arr**[**k**].**st**)**

**{**

edge**[**i**].**pb**(**k**);**

**}**

**}**

**}**

result**=** BPM**();**

result**=** n**-**result**;**

printf**(**"Case %d: %d\n"**,** **++**t**,** result**);**

arr**.**clear**();**

//edge clear

REP**(**i**,** MAX**)**

**{**

edge**[**i**].**clear**();**

**}**

**}**

**}**

/\*

Your taxi cab company is one of the best taxi cab networks in town. But running a taxi station is definitely not simple. Apart from the obvious demand for a centralized coordination of the cabs in order to pick up the customers calling to get a cab as soon as possible, there is also a need to schedule all the taxi rides which have been booked in advance. Given a list of all booked taxi rides for the next day, you want to minimize the number of cabs needed to carry out all of the rides.

For simplicity, we model a city as a rectangular grid. An address in the city is denoted by two integers: the street and avenue number. The time needed to get from the address (a, b) to (c, d) by taxi is |a - c| + |b - d| minutes. A cab may carry out a booked ride if it is its first ride of the day, or if it can get to the source address of the new ride from its latest, at least one minute before the new ride's scheduled departure. Note that some rides may end after midnight.

Input

Input starts with an integer T (= 20), denoting the number of test cases.

Each case starts with a line containing an integer M (0 < M < 500), being the number of booked taxi rides. The following M lines contain the rides. Each ride is described by a departure time on the format hh:mm (ranging from 00:00 to 23:59), two integers a b that are the coordinates of the source address and two integers c d that are the coordinates of the destination address. All coordinates are at least 0 and strictly smaller than 200. The booked rides in each scenario are sorted in order of increasing departure time.

Output

For each case, print the case number and the minimum number of cabs required to carry out all the booked taxi rides.

Sample Input

Output for Sample Input

2

2

08:00 10 11 9 16

08:06 9 16 10 11

2

08:00 10 11 9 16

08:07 9 16 10 11

Case 1: 2

Case 2: 1

\*/

**Problem Name: 1209 - Strange Voting**

/\*

OJ: Light OJ

Name Fazle Rabby Sourav

Problem Link: http:http://www.lightoj.com/volume\_showproblem.php?problem=1209

Problem Name: 1209 - Strange Voting

Rank: 41

\*/

/\*

Solution Method:

It seems to me a tricky BPM problem. At the first look, one may think about the Candidates. but here the data of candidates is useless. it is used for

making the problem difficult to dissect. we will concern about the data of male and female voters. at first we make two sets , one is male and another is female, and

save the input data in it. and we assume every person as a single node. then if a male voter prefer i-th candidates and any other female voter wants that i-th cadidates

to be thrown, then we will make an edge between them. after making setting up all edges we will run bipartite matching function. thus we will get maximum matching of that graph.

now we get the maximum matching. but in this case what does it mean by maximum matching??

it is the number of voters whom cant be satisfied.... so the final ans will be number of voters minus the maximum matching.

Thus we can get the maximum number of voters to be satisfied.

\*/

/\*

Complexity Analysis:

if n is the number of voters. complexity of DFS is O(V+E). here edge/nodes can be at most n. the complexity of this solution will be O(n\*(n+n)). or O(n^2);

\*/

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Code\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

// Have skipped precode

int m**,** f**,** v**,** taken**[**MAX**],** par**[**MAX**];**

vi edge**[**MAX**];**

vii male**,** fem**;**

bool dfs**(**int from**)**

**{**

**if(**taken**[**from**]==**1**)** **return** **false;**

int i**,** to**;**

taken**[**from**]=**1**;**

REP**(**i**,** SZ**(**edge**[**from**]))**

**{**

to**=** edge**[**from**][**i**];**

**if(**par**[**to**]==-**1 **||** dfs**(**par**[**to**]))**

**{**

taken**[**to**]=**1**;**

par**[**to**]=** from**;**

**return** **true;**

**}**

**}**

**return** **false;**

**}**

int matching**()**

**{**

int i**,** j**,** k**,** cnt**=**0**;**

mset**(**par**,** **-**1**);**

**for(**i**=**0**;** i**<**SZ**(**male**);** i**++)**

**{**

CLR**(**taken**);**

**if(**dfs**(**i**))**

cnt**++;**

**}**

**return** cnt**;**

**}**

int main**()**

**{**

int i**,** j**,** k**,** result**,** t**=**0**,** tcase**,** n**,** cnt**,** x**,** y**;**

string str1**,** str2**;**

cin**>>**tcase**;**

**while(**tcase**--)**

**{**

cin**>>**m**>>**f**>>**v**;**

male**.**clear**();**

fem**.**clear**();**

REP**(**i**,** v**)**

**{**

cin**>>**str1**>>**str2**;**

**if(**str1**[**0**]==**'M'**)**

**{**

str1**=** str1**.**substr**(**1**);**

str2**=** str2**.**substr**(**1**);**

stringstream strm**;**

strm**<<**str1**;** strm**>>**x**;**

strm**.**clear**();**

strm**<<**str2**;** strm**>>**y**;**

male**.**pb**(**MP**(**x**,** y**));**

**}**

**else**

**{**

str1**=** str1**.**substr**(**1**);**

str2**=** str2**.**substr**(**1**);**

stringstream strm**;**

strm**<<**str1**;** strm**>>**x**;**

strm**.**clear**();**

strm**<<**str2**;** strm**>>**y**;**

fem**.**pb**(**MP**(**y**,** x**));**

**}**

**}**

REP**(**i**,** SZ**(**male**))**

**{**

REP**(**k**,** SZ**(**fem**))**

**{**

**if(**male**[**i**].**fs**==**fem**[**k**].**fs**)**

**{**

edge**[**i**].**pb**(**k**+**500**);** edge**[**k**+**500**].**pb**(**i**);**

**}**

**else** **if(**male**[**i**].**sc**==**fem**[**k**].**sc**)**

**{**

edge**[**i**].**pb**(**k**+**500**);** edge**[**k**+**500**].**pb**(**i**);**

**}**

**}**

**}**

result**=** matching**();**

printf**(**"Case %d: %d\n"**,** **++**t**,** v**-**result**);**

REP**(**i**,** MAX**)**

**{**

edge**[**i**].**clear**();**

**}**

**}**

**}**

/\*

problem Statement:

Its year 3000, and the voting system in Ajobdesh has changed to a new era. Instead of the boring old style voting, the new style voting is applied as follows:

1. Initially there are m male candidates and f female candidates for the parliament. For simplicity the male candidates are numbered as 'M1', 'M2' ... 'Mm' and the female candidates are numbered as 'F1', 'F2' ... 'Ff'.

2. There are v voters, and each of them can vote like 'P Q', which means, he wants to see P in the parliament and he wants Q to be thrown out of the parliament. For example, if a person voted like, 'M3 F7', that means he wants M3 to be elected and F7 to be thrown out.

3. The parliament will be formed in such a way that the maximum number of votes is satisfied. A voter, who voted 'P Q', is said to be satisfied if P is in the parliament and Q is not. For example, let the parliament be {M1, M3, F3}, then voter who voted 'M1 F4' is satisfied but neither of 'M1 F3', 'F3 M3' or 'M2 F3' is satisfied.

Since Men don't want to see any Woman in the parliament, each man always votes like 'Mx Fy'. And Women don't want Men in the parliament, so each woman always votes like 'Fy Mx'. Assume that only Men and Women are eligible for voting.

Since you are the leading programmer in Ajobdesh, you have to form the parliament such that maximum number of voters is satisfied. Just report the maximum number of satisfied voters.

Input

Input starts with an integer T (= 25), denoting the number of test cases.

Each case starts with a line containing three integers: m, f, v (1 = m, f = 100, 0 = v = 500). Each of the next v lines contains a vote either in the form 'Mx Fy' or 'Fy Mx' (1 = x = m, 1 = y = f).

Output

For each case, print the case number and the maximum number of satisfied voters.

Sample Input

Output for Sample Input

2

1 1 2

M1 F1

F1 M1

1 2 5

M1 F1

M1 F1

M1 F2

F2 M1

F1 M1

Case 1: 1

Case 2: 3

\*/

**Problem Name: 1403 - Air Raid**

/\*

OJ: Light OJ

Name Fazle Rabby Sourav

Problem Link: http://www.lightoj.com/volume\_showproblem.php?problem=1403

Problem Name: 1403 - Air Raid

Rank: 37

\*/

/\*

Solution Method:

there are many cities and road between them. we need to find minimum troop number so that every city is reached by only one troop.

In the statement , it is mentioned that there is no cycle. we assume each intersection as a node. and a road between two intersection means

an edge between corresponding nodes. then run DFS to find maximum number of matching. which stands for this fact that number of maximum matching

is the number of nodes which can be reached by a troop landed another node/city.

so the minimum number of troop will be number of city - maximum matchig number.

\*/

/\*

Complexity Analysis:

The Runnig time of DFS is O(E+V) where E is the number of edges and V is is the number of Vertices. here E= m and V=n.

so the complexity will be O(n\*(n+m)) or approximately O(n^2);

\*/

///\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Code\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*///

int num\_node**,** num\_edge**,** par**[**MAX**],** x**;**

bool col**[**MAX**];**

vi edge**[**MAX**];**

int dfs**(**int from**)**

**{**

int i**,**j**,** k**,** to**;**

REP**(**i**,** SZ**(**edge**[**from**]))**

**{**

to**=** edge**[**from**][**i**];**

**if(**col**[**to**])** **continue;**

col**[**to**]=** 1**;**

**if(**par**[**to**]==-**1 **||** dfs**(**par**[**to**]))**

**{**

par**[**to**]=** from**;**

**return** **true;**

**}**

**}**

**return** **false;**

**}**

int BPM**()**

**{**

int i**,** j**,** k**,** cnt**=**0**;**

mset**(**par**,** **-**1**);**

**for(**i**=**1**;** i**<=**num\_node**;** i**++)**

**{**

CLR**(**col**);**

**if(**dfs**(**i**))**

cnt**++;**

**}**

**return** cnt**;**

**}**

int main**()**

**{**

int i**,** j**,** k**,** result**,** t**=**0**,** tcase**,** n**,** m**,** u**,** v**,** cnt**;**

cin**>>**tcase**;**

**while(**tcase**--)**

**{**

scanf**(**"%d %d"**,** **&**num\_node**,** **&**num\_edge**);**

REP**(**i**,** num\_edge**)**

**{**

scanf**(**"%d %d"**,** **&**u**,** **&**v**);**

edge**[**u**].**pb**(**v**);**

**}**

result**=** BPM**();**

// bug(result);

printf**(**"Case %d: %d\n"**,** **++**t**,** **(**num\_node**-**result**));**

REP**(**i**,** num\_node**+**5**)**

**{**

edge**[**i**].**clear**();**

**}**

**}**

**}**