**A1 & A2**

This problem can be solved with greedy approach.

Let's look at this problem in a different way - instead of deleting as much edges as possible, we'll try to pick as few as possible (and remove everything else).

We have graph for men and graph for women. Let's call them **M-graph** and **W-graph**. Obviously there is no need to add edge which doesn't connect something in any of graphs. Therefore any edge we take will connect two components in **W-graph**, or in **M-graph**, or in both (this is possible only for edges of 3rd type). You need to make **2N-2** merges (**N-1** in **W-graph** and **N-1** in **M-graph**). As we noticed, every edge will do either 1 merge or 2 merges (as we said before, if can make 2 merges only if it is edge of type 3 and it merges two components in both **W-graph** and **M-graph** at the same time). This leads us to a simple greedy: *let's pick as many edges doing 2 merges as possible*.

Add all edges of 3rd type which are merging distinct components. This can be done using *DSU*. Now you can actually find answer already - if you have used **K** edges of 3rd type, you made **2K** merges already, and you'll need **2N-2-2K** more merges, and it will take *1 edge per merge* now. If you have a task to print edges, not only their number - it can be done by solving **M-graph** and **W-graph** in similar way, but starting with our *partially connected by edges of 3rd type* graphs instead of empty one.

And don't forget to check that original **M-graph** and **W-graph** are connected. Solution without this check will give you only *90 points*.

**Author's Solution**

NA

**Tester's Solution**

#include <bits/stdc++.h>

using namespace std;

int n,m;

int a,b,c,ans;

int w[1<<20];

int get(int x)

{

if (w[x]==x)

return x;

return w[x]=get(w[x]);

}

void merge(int a,int b)

{

w[a]=b;

}

int W[1<<20][3];

int get1(int x,int y)

{

if (W[x][y]==x)

return x;

return get1(W[x][y],y);

}

void merge1(int a,int b,int c)

{

W[a][c]=b;

}

int main(){

ios\_base::sync\_with\_stdio(0);

//cin.tie(0);

cin>>n>>m;

for (int i=1;i<=n;i++)

w[i]=W[i][1]=W[i][2]=i;

for (int i=1;i<=m;i++)

{

cin>>a>>b>>c;

for (int j=1;j<=2;j++)

if (c&j)

{

int ta,tb;

ta=get1(a,j);

tb=get1(b,j);

if (ta==tb)

continue;

merge1(ta,tb,j);

}

if (c!=3)

continue;

a=get(a);

b=get(b);

if (a==b)

continue;

merge(a,b);

++ans;

}

ans=(n-1-ans)\*2+ans;

int cnt=0;

for (int i=1;i<=n;i++)

for (int j=1;j<=2;j++)

if (W[i][j]==i)

++cnt;

if (cnt>2)

{

cout<<-1<<endl;

return 0;

}

cout<<m-ans<<endl;

return 0;}