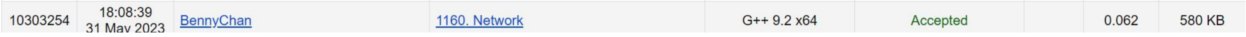
Laboratory work # 8

Student: *Zhanghao Chen*
Student ID: 21321205
Timus Name: *BennyChan*

Mail: 1824362950@qq.com

Problem # 1160 *Network*

Screenshot from Timus:



10303254	18:08:39 31 May 2023	BennyChan	1160. Network	G++ 9.2 x64	Accepted		0.062	580 KB

Explanation of algorithm:

I implement Kruskal's algorithm for finding the minimum spanning tree of a graph. The algorithm works by first sorting the edges by weight, and then selecting edges from smallest to largest. If selecting an edge would create a cycle, it is not selected. The algorithm continues until (*n*-1) edges have been selected. It is worth noting that the algorithm uses a union-find data structure to detect cycles.

Computational complexity of algorithm:

$$O(ElogE + ElogN)$$

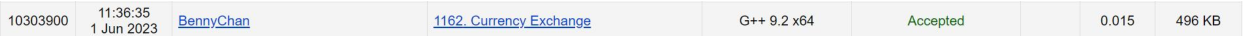
where E is the number of edges and N is the number of nodes

Source code:

```
#include<bits/stdc++.h>
using namespace std;

int fa[1010];

struct node3{
    int x,y,len;
}p[15010];

bool cmp(node3 a,node3 b){
    return a.len<b.len;
}

int find(int x){
    if(fa[x]!=x){
        fa[x]=find(fa[x]);
    }
    return fa[x];
}

int main(){
    int n,m,i,j,s=0,maxz=0;
    cin>>n>>m;

    for(i=1;i<=n;i++){
        fa[i]=i;
    }

    for(i=1;i<=m;i++){
        cin>>p[i].x>>p[i].y>>p[i].len;
    }

    sort(p+1,p+m+1,cmp);

    for(i=1;i<=m&&s<n-1;i++){
        int tx=find(p[i].x);
        int ty=find(p[i].y);
        if(tx!=ty){
            fa[tx]=ty;
            s++;
            maxz=max(maxz,p[i].len);
        }
    }

    cout<<maxz<<endl;
    cout<<i-1<<endl;

    for(j=1;j<i;j++){
        cout<<p[j].x<<" "<<p[j].y<<endl;
    }

    return 0;
}
```

Problem # 1162 *Currency Exchange*

Screenshot from Timus:



10303900	11:36:35 1 Jun 2023	BennyChan	1162. Currency Exchange	G++ 9.2 x64	Accepted		0.015	496 KB

Explanation of algorithm:

I utilize the shortest path faster algorithm (SPFA) to detect whether foreign exchange arbitrage is possible. Given a list of currency exchange rates and transaction fees, the algorithm determines if there exists a positive cycle in the graph of currency exchange rates.

Computational complexity of algorithm:

$$O(N \times M)$$

where M is the number of edges and N is the number of nodes

Source code:

```
#include<bits/stdc++.h>
using namespace std;

#define maxn 1003
#define INF 0x3f3f3f3f
#define eps 1e-7

struct Edge
{
    int fromVertex, toVertex;
    double exchangeRate, transactionFee;
    Edge(int from, int to, double rate, double fee)
    {
        fromVertex = from;
        toVertex = to;
        exchangeRate = rate;
        transactionFee = fee;
    }
};

vector<Edge> ways[maxn];
double dis[maxn];
int cnt[maxn];
bool ever[maxn];
int n, m, s;
double v;

void initialize()
{
    memset(ways, 0, sizeof(ways));
    memset(cnt, 0, sizeof(cnt));
    memset(ever, 0, sizeof(ever));
    for(int i = 1; i <= n; ++i)
    {
        dis[i] = 0;
    }
    for(int i = 0; i < m; ++i)
    {
        int fromVertex, toVertex;
        double toRate1, toFee1, toRate2, toFee2;
        cin >> fromVertex >> toVertex;
        cin >> toRate1 >> toFee1 >> toRate2 >> toFee2;
        ways[fromVertex].push_back(Edge(fromVertex, toVertex, toRate1, toFee1));
        ways[toVertex].push_back(Edge(toVertex, fromVertex, toRate2, toFee2));
    }
}

bool SPFA()
{
    ever[s] = 1;
    cnt[s]++;
    queue<int> q;
    q.push(s);
    dis[s] = v;
    while(!q.empty())
    {
        int cur = q.front();
        q.pop();
        int len = ways[cur].size();
        ever[cur] = 0;
        for(int i = 0; i < len; ++i)
        {
            int y = ways[cur][i].toVertex;
            if((dis[cur] - ways[cur][i].transactionFee)*ways[cur][i].exchangeRate - dis[y] > eps)
            {
                dis[y] = (dis[cur] - ways[cur][i].transactionFee)*ways[cur][i].exchangeRate;
                if(!ever[y])
                {
                    q.push(y);
                    ever[y] = 1;
                    cnt[y]++;
                    if(cnt[y] >= n)
                        return true;
                }
            }
        }
    }
    return false;
}

int main()
{
    cin >> n >> m >> s >> v;
    initialize();
    bool isPositiveCycle = SPFA();
    if(isPositiveCycle)
        printf("YES");
    else
        printf("NO");
    return 0;
}
```