Roads in HackerLand

105/563 challenges solved

Rank: 19918 | Points: 2504.09

You have successfully solved Roads in HackerLand Share Tweet

Try the next challenge | Try a Random Challenge

Problem Submissions Leaderboard Editorial

Editorial by nikasvanidze

We use a little long arithmetic and *depth-first search* (DFS) to solve this challenge. Consider the following equation:

$$2^0 + 2^1 + 2^2 + 2^3 + \ldots + 2^{x-1} = 2^x - 1$$

We can say that:

$$2^{a_1} + 2^{a_2} + 2^{a_3} + \ldots + 2^{a_{k-1}} < 2^{a_k}, \text{where } 0 \leq a_x < a_{x+1}$$

This means it's best to move where the maximum size of the edge is as small as possible.

We have to make a Minimum Spanning Tree (MST) and move through it. Then, for each node, calculate the number of ways which travel passes through it. If the tree is bisected at an edge such that there are Y nodes in one half on the tree and $N-Y$ nodes in the second half, then the number of ways which travel passes through this edge is $Y \cdot (N-Y)$.

The author's solution runs in $O(M \times \log M)$, but it can be solved in $O(N+M)$ if you sort the edges using radix sort. Take some time to review the code below!

Set by nikasvanidze

Problem Setter's code:

```
#include <bits/stdc++.h>
#define F first
#define S second
using namespace std;
const int N=100000;
const int M=200000;

int n,m;
int d[N+5],bm;
long long b[M+70];
pair <int,pair <int,int> > a[M+5];
vector <pair <int,int> > v[N+5];

void input(){
    scanf("%d%d",&n,&m);
    for (int i=0;i<m;i++)
    scanf("%d%d%d",&a[i].S.F,&a[i].S.S,&a[i].F);
}

//DSU
int fin(int x){
    if (d[x]==x) return x;
    return d[x]=fin(d[x]);
}

int dfs(int x,int fr){
```

```
    int R=1;
    for (int i=0;i<v[x].size();i++)
    if (v[x][i].F!=fr){
        int K=dfs(v[x][i].F,x);
        b[v[x][i].S]+=1ll*K*(n-K);
        R+=K;
    }
    return R;
}

void sol(){
    sort(a,a+m); // this is O(M log M) but using index sort we can do this in O(M)
    //MST
    for (int i=1;i<=n;i++) d[i]=i;
    for (int i=0;i<m;i++)
    if (fin(a[i].S.F)!=fin(a[i].S.S)){
        d[fin(a[i].S.F)]=fin(a[i].S.S);
        v[a[i].S.F].push_back({a[i].S.S,a[i].F});
        v[a[i].S.S].push_back({a[i].S.F,a[i].F});
        bm=a[i].F;
    }
    dfs(1,1);
    for (int i=0;i<bm;i++)
        b[i+1]+=b[i]/2,
        b[i]%=2;
    while (b[bm]>1){
        b[bm+1]=b[bm]/2;
        b[bm]%=2;
        bm++;
    }
    for (int i=bm;i>=0;i--) putchar('0'+b[i]);
    putchar('\n');
}

int main() {
    input();
    sol();
    return 0;
}
```

Tested by gorbunovdv

Problem Tester's code:

```
n, m = map(int, raw_input().split())
e = []
for i in range(m):
    u, v, c = map(int, raw_input().split())
    e.append((c, u - 1, v - 1))

parent = [i for i in range(n)]


def find_set(v):
    global parent
    if parent[v] != v:
        parent[v] = find_set(parent[v])
    return parent[v]


e.sort()

g = [[] for i in range(n)]

for (c, u, v) in e:
    a, b = find_set(u), find_set(v)
    if a != b:
        parent[a] = b
        g[u].append((v, c))

        g[v].append((u, c))
```

```
ans = [0] * (m * 2)


def dfs(v, p=-1):
    global ans
    cur_size = 1
    for (to, c) in g[v]:
        if to != p:
            sz = dfs(to, v)
            ans[c] += sz * (n - sz)
            cur_size += sz
    return cur_size

dfs(0)

for i in range(len(ans) - 1):
    ans[i + 1] += ans[i] / 2
    ans[i] %= 2

flag = False
real_ans = []
for i in range(len(ans) - 1, -1, -1):
    if flag or ans[i] > 0:
        flag = True
        real_ans.append(ans[i])
print "".join(map(str, real_ans))
```

Feedback

Was this editorial helpful?

Yes | No