**城市建设**

问题描述

　　栋栋居住在一个繁华的C市中，然而，这个城市的道路大都年久失修。市长准备重新修一些路以方便市民，于是找到了栋栋，希望栋栋能帮助他。  
  
　　C市中有n个比较重要的地点，市长希望这些地点重点被考虑。现在可以修一些道路来连接其中的一些地点，每条道路可以连接其中的两个地点。另外由于C市有一条河从中穿过，也可以在其中的一些地点建设码头，所有建了码头的地点可以通过河道连接。  
  
　　栋栋拿到了允许建设的道路的信息，包括每条可以建设的道路的花费，以及哪些地点可以建设码头和建设码头的花费。  
  
　　市长希望栋栋给出一个方案，使得任意两个地点能只通过新修的路或者河道互达，同时花费尽量小。

输入格式

　　输入的第一行包含两个整数n, m，分别表示C市中重要地点的个数和可以建设的道路条数。所有地点从1到n依次编号。  
　　接下来m行，每行三个整数a, b, c，表示可以建设一条从地点a到地点b的道路，花费为c。若c为正，表示建设是花钱的，如果c为负，则表示建设了道路后还可以赚钱（比如建设收费道路）。  
　　接下来一行，包含n个整数w\_1, w\_2, …, w\_n。如果w\_i为正数，则表示在地点i建设码头的花费，如果w\_i为-1，则表示地点i无法建设码头。  
　　输入保证至少存在一个方法使得任意两个地点能只通过新修的路或者河道互达。

输出格式

　　输出一行，包含一个整数，表示使得所有地点通过新修道路或者码头连接的最小花费。如果满足条件的情况下还能赚钱，那么你应该输出一个负数。

样例输入

5 5  
1 2 4  
1 3 -1  
2 3 3  
2 4 5  
4 5 10  
-1 10 10 1 1

样例输出

9

样例说明

　　建设第2、3、4条道路，在地点4、5建设码头，总的花费为9。

数据规模和约定

　　对于20%的数据，1<=n<=10，1<=m<=20，0<=c<=20，w\_i<=20；  
　　对于50%的数据，1<=n<=100，1<=m<=1000，-50<=c<=50，w\_i<=50；  
　　对于70%的数据，1<=n<=1000；  
　　对于100%的数据，1 <= n <= 10000，1 <= m <= 100000，-1000<=c<=1000，-1<=w\_i<=1000，w\_i≠0。

本题的C++参考代码如下：

#include <iostream>

#include <cstdio>

#include <algorithm>

using namespace std;

const int MAX = 11111;

int parent[MAX];

struct Edge

{

int st;

int ed;

int cost;

bool operator<(const Edge &rhs)const

{

return cost < rhs.cost;

}

};

Edge E[222222];

int find(int r)

{

if(parent[r] == r)

return r;

return parent[r] = find(parent[r]);

}

void input(int &x)

{

char ch = getchar();

bool flag = ch == '-';

while(ch<'0' || ch>'9')

{

ch = getchar();

if(ch == '-')

flag = true;

}

x = 0;

while(ch>='0' && ch<='9')

{

x = x \* 10 + ch - '0';

ch = getchar();

}

x = flag?-x:x;

}

int weight;

int num;

int tmp\_cost;

int cnt ;

void kruskal(int n, int m)

{

int i = 0;

for( i=0;i<=n;++i)

parent[i] = i;

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

{

int x = E[i].st;

int y = E[i].ed;

int rx = find(x);

int ry = find(y);

if(rx != ry)

{

parent[rx] = ry;

cnt++;

if(x==0)

{

num++;

tmp\_cost = E[i].cost;

}

weight+=E[i].cost;

}

else if(E[i].cost<0)

weight += E[i].cost;

}

if(num==1)

weight -= tmp\_cost;

}

int main()

{

//freopen("in.txt","r",stdin);

int n,m;

int cost;

input(n);

input(m);

//scanf("%d%d",&n,&m);

int i;

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

{

input(E[i].st);

input(E[i].ed);

input(E[i].cost);

//scanf("%d%d%d",&E[i].st,&E[i].ed,&E[i].cost);

//if(E[i].cost<0)

//weight+=E[i].cost;

}

sort(E,E+m);

kruskal(n,m);

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

{

input(cost);

//scanf("%d",&cost);

if(cost!=-1)

{

E[m].st = 0;

E[m].ed = i;

E[m++].cost = cost;

}

}

if(cnt==n-1)

{

int minWeight = weight;

weight = 0;

sort(E,E+m);

kruskal(n,m);

minWeight = minWeight < weight?minWeight:weight;

cout<<minWeight<<endl;

}

else

{

weight = 0;

sort(E,E+m);

kruskal(n,m);

cout<<weight<<endl;

}

}

本题的C参考代码如下：

#include<stdio.h>

#include<string.h>

#include<stdlib.h>

#define INF 0xfffffff

int n,m,k;

int pre[10005];

struct s { int u,v,w; }edge[1000005];

int cmp(const void \*a,const void \*b) { return (\*(struct s \*)a).w-(\*(struct s \*)b).w; }

int find(int x) {

if(x==pre[x]) return x;

return pre[x]=find(pre[x]);

}

void init(int n) {

int i;

for(i=0;i<=n;i++) {

pre[i]=i;

}

}

int ku(int k) {

int sum=0,i;

for(i=0;i<k;i++) {

int u=edge[i].u;

int v=edge[i].v;

int w=edge[i].w;

int fa=find(u);

int fb=find(v);

if(fa!=fb||w<0) {

sum+=w;

pre[fa]=fb;

}

}

return sum;

}

int main() {

//while(scanf("%d%d",&n,&m)!=EOF) {

int i,j;

scanf("%d%d",&n,&m);

init(n);

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

int u,v,w;

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

edge[i].u=u;

edge[i].v=v;

edge[i].w=w;

}

k=m;

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

int cost;

scanf("%d",&cost);

if(cost!=-1) {

edge[k].u=0;

edge[k].v=i;

edge[k++].w=cost;

}

}

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

int fa=find(edge[i].u);

int fb=find(edge[i].v);

if(fa!=fb) pre[fa]=fb;

} for(i=2;i<=n;i++) {

if(find(1)!=find(i)) { break; } }

int temp;

if(i==n+1) {

init(n);

qsort(edge,m,sizeof(edge[0]),cmp);

temp=ku(m);

init(n);

qsort(edge,k,sizeof(edge[0]),cmp);

int ans=ku(k);

if(ans>temp) printf("%d\n",temp);

else printf("%d\n",ans);

//continue;

return 0;

}

init(n);

qsort(edge,k,sizeof(edge[0]),cmp);

int ans=ku(k);

printf("%d\n",ans);

//}

return 0;

}

本题的Java参考代码如下：

import java.io.BufferedInputStream;

import java.io.IOException;

import java.util.Arrays;

import java.util.Comparator;

public class Main {

private static class Edge {

public int src;

public int dest;

public int w;

public Edge(int src, int dest, int w) {

this.src = src;

this.dest = dest;

this.w = w;

}

}

private static BufferedInputStream in = new BufferedInputStream(System.in);

private static Edge[] edge;

private static int[] pre;

private static int readInt() throws IOException {

int i, sum = 0;

boolean flag = false;

while (((i = in.read()) & 48) != 48 || i > 57) {

if (i == 45) {

flag = true;

}

}

for (; (i & 56) == 48 || (i & 62) == 56; i = in.read())

sum = sum \* 10 + (i & 15);

if (flag)

sum = -sum;

return sum;

}

public static void main(String[] args) throws IOException {

int n = readInt();

int m = readInt();

edge = new Edge[n + m];

pre = new int[n + 1];

for (int i = 0; i < m; i++) {

edge[i] = new Edge(readInt(), readInt(), readInt());

}

int a;

int count = 0;

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

a = readInt();

if (a != -1) {

edge[m + count] = new Edge(0, i, a);

count++;

}

}

int sum2 = dfs(n, m);

int sum1 = dfs(n + 1, m + count);

if (sum2 != -1)

sum1 = Math.min(sum1, sum2);

System.out.println(sum1);

}

private static int dfs(int nn, int mm) {

initPre();

Arrays.sort(edge, 0, mm, new Comparator<Edge>() {

public int compare(Edge o1, Edge o2) {

return o1.w - o2.w;

}

});

int count = 0;

int result = 0;

for (int i = 0; i < mm; i++) {

if (union(edge[i].src, edge[i].dest)) {

result += edge[i].w;

count++;

} else if (edge[i].w < 0) {

result += edge[i].w;

}

}

if (count != (nn - 1)) {

return -1;

}

return result;

}

private static void initPre() {

for (int i = 1; i < pre.length; i++) {

pre[i] = i;

}

}

private static int find(int x) {

int root = x;

while (root != pre[root]) {

root = pre[root];

}

int temp;

while (x != pre[x]) {

temp = pre[x];

pre[x] = root;

x = temp;

}

return root;

}

private static boolean union(int x, int y) {

int xx = find(x);

int yy = find(y);

if (xx != yy) {

pre[xx] = yy;

return true;

}

return false;

}

}