**算法训练 安慰奶牛**

时间限制：1.0s   内存限制：256.0MB

问题描述

Farmer John变得非常懒，他不想再继续维护供奶牛之间供通行的道路。道路被用来连接N个牧场，牧场被连续地编号为1到N。每一个牧场都是一个奶牛的家。FJ计划除去P条道路中尽可能多的道路，但是还要保持牧场之间 的连通性。你首先要决定那些道路是需要保留的N-1条道路。第j条双向道路连接了牧场Sj和Ej(1 <= Sj <= N; 1 <= Ej <= N; Sj != Ej)，而且走完它需要Lj的时间。没有两个牧场是被一条以上的道路所连接。奶牛们非常伤心，因为她们的交通系统被削减了。你需要到每一个奶牛的住处去安慰她们。每次你到达第i个牧场的时候(即使你已经到过)，你必须花去Ci的时间和奶牛交谈。你每个晚上都会在同一个牧场(这是供你选择的)过夜，直到奶牛们都从悲伤中缓过神来。在早上 起来和晚上回去睡觉的时候，你都需要和在你睡觉的牧场的奶牛交谈一次。这样你才能完成你的 交谈任务。假设Farmer John采纳了你的建议，请计算出使所有奶牛都被安慰的最少时间。

输入格式

第1行包含两个整数N和P。

接下来N行，每行包含一个整数Ci。

接下来P行，每行包含三个整数Sj, Ej和Lj。

输出格式

输出一个整数, 所需要的总时间(包含和在你所在的牧场的奶牛的两次谈话时间)。

样例输入

5 7  
10  
10  
20  
6  
30  
1 2 5  
2 3 5  
2 4 12  
3 4 17  
2 5 15  
3 5 6

样例输出

176

数据规模与约定

5 <= N <= 10000，N-1 <= P <= 100000，0 <= Lj <= 1000，1 <= Ci <= 1,000。

锦囊1

使用最小生成树算法。

锦囊2

将每条边(a, b)的权值Lj改变为2Lj+Ca+Cb，然后使用最小生成树来计算。

本题的C++参考代码如下：

#include<cstdio>

#include<cstdlib>

#include<cstring>

#include<cmath>

#include<ctime>

#include<iostream>

#include<algorithm>

using namespace std;

const int INF=999999999;

struct wjj

{

int x,y,w;

}edge[100005];

int v[10005],N,P,father[10005],Min=INF;

int \_getfather(int x)

{

if(x!=father[x]) father[x]=\_getfather(father[x]);

return father[x];

}

void \_in(int &x)

{

char t=getchar();

while(t<'0'||'9'<t) t=getchar();

for(x=t-'0',t=getchar();'0'<=t&&t<='9';x=x\*10+t-'0',t=getchar());

}

void \_init()

{

\_in(N);\_in(P);

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

{

father[i]=i;

\_in(v[i]);

Min=min(Min,v[i]);

}

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

{

\_in(edge[i].x);\_in(edge[i].y);\_in(edge[i].w);

edge[i].w<<=1;

edge[i].w+=v[edge[i].x]+v[edge[i].y]; //重新计算边的权值！！！

}

}

void \_qst\_w(int l,int r)

{

int i=l,j=r,mw=edge[(i+j)>>1].w;

while(i<=j)

{

while(edge[i].w<mw) i++;

while(edge[j].w>mw) j--;

if(i<=j)

{

swap(edge[i],edge[j]);

i++;j--;

}

}

if(l<j) \_qst\_w(l,j);

if(i<r) \_qst\_w(i,r);

}

void \_solve()//Kruskal

{

\_qst\_w(1,P);

int fx,fy,k,cnt,tot;

k=cnt=tot=0;

while(cnt<N-1)

{

k++;

fx=\_getfather(edge[k].x);

fy=\_getfather(edge[k].y);

if(fx!=fy)

{

father[fx]=fy;

tot+=edge[k].w;

cnt++;

}

}

printf("%d\n",tot+Min); //不要忘记最后加上起点的权值（点的最小权值）

}

int main()

{

\_init();

\_solve();

return 0;

}

本题的C参考代码如下：

# include <stdio.h>

# include <stdlib.h>

# define M 100000

typedef struct Node

{

int u;

int v;

int w;

}

Node;

Node e[100002];

int fa[100002];

int c[100002];

int rank[100002];

int sum = 0;

int n, m;

int cmp(const void \*a, const void \*b)

{

Node \*c = (Node \*)a;

Node \*d = (Node \*)b;

return c->w-d->w;

}

int find(int x)

{

int i, k, r;

r = x;

while (fa[r]>=0)

r = fa[r];

k = x;

while (k != r)

{

i = fa[k];

fa[k] = r;

k = i;

}

return r;

/\* if (x != fa[x])

fa[x] = find(fa[x]);

return fa[x];\*/

}

void Union(int u, int v) {

/\* if (rank[u] > rank[v])

fa[v] = u;

else

{

if (rank[u] == rank[v])

rank[v]++;

fa[u] = v;

}\*/

int r1,r2;

int num;

r1=find(u);

r2=find(v);

num=fa[r1]+fa[r2];

if(fa[r1]<fa[r2])

{

fa[r2]=r1;

fa[r1]=num;

}

else

{

fa[r1]=r2;

fa[r2]=num;

}

}

int Kruskal()

{

int i;

int u,v;

int sumweight=0,count=0;

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

fa[i]=-1;

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

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

{

u=e[i].u;

v=e[i].v;

if(find(u)!=find(v))

{

sumweight+=e[i].w;

Union(u,v);

count++;

if(count>=n-1)

break;

}

}

return sumweight;

}

int main ()

{

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

int i, j, min = M;

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

{

scanf ("%d", &c[i]);

if (c[i] < min)

min = c[i];

}

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

{

int u, v, w;

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

e[i].u=u-1;

e[i].v=v-1;

e[i].w=w\*2+c[u-1]+c[v-1];

}

printf ("%d\n", min+Kruskal());

return 0;

}

本题的Java参考代码如下：

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.util.ArrayList;

import java.util.Collections;

import java.util.StringTokenizer;

class Reader3{

static BufferedReader reader;

static StringTokenizer tokenizer;

static void init(InputStream input){

reader=new BufferedReader(new InputStreamReader(input));

tokenizer=new StringTokenizer("");

}

static String next() throws IOException{

while (!tokenizer.hasMoreElements()) {

tokenizer = new StringTokenizer(reader.readLine(

));

}

return tokenizer.nextToken();

}

static int nextInt() throws IOException{

return Integer.parseInt(next());

}

static double nextDouble() throws IOException{

return Double.parseDouble(next());

}

}

class KruskalDui{

int a,b,l;

}

public class Main{

/\*\*

\* @param args

\* @throws IOException

\*/

static int father[]=new int[100000];

static ArrayList<KruskalDui> path =new ArrayList<KruskalDui>();

public static int getfather(int x) {

if (x!=father[x]) {

father[x]=getfather(father[x]);

}

return father[x];

}

public static void \_qst\_w(int l,int r) {

int i=l,j=r,mw=path.get((i+j)/2).l;

while(i<=j){

while(path.get(i).l<mw) i++;

while(path.get(j).l>mw) j--;

if(i<=j){

Collections.swap(path,i,j);

i++;j--;

}

}

if(l<j) \_qst\_w(l,j);

if(i<r) \_qst\_w(i,r);

}

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

// TODO Auto-generated method stub

Reader3.init(System.in);

int n=Reader3.nextInt();

int p=Reader3.nextInt();

int d[]=new int [n+1];

int minD=Integer.MAX\_VALUE;

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

d[i]=Reader3.nextInt();

father[i]=i;

if (d[i]<minD) {

minD=d[i];

}

}

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

KruskalDui k=new KruskalDui();

k.a=Reader3.nextInt();

k.b=Reader3.nextInt();

k.l=Reader3.nextInt();

k.l=k.l\*2+d[k.a]+d[k.b];

path.add(k);

}

\_qst\_w(0,p-1);

int fx,fy,result=minD,count=0,k=0;

while(count<n-1){

fx=getfather(path.get(k).a);

fy=getfather(path.get(k).b);

if(fx!=fy){

father[fx]=fy;

result+=path.get(k).l;

count++;

}

k++;

}

System.out.println(result);

}

}