# Datestruct

## Rmq

/\*

index from 0

first must init\_dex a is value arrest

\*/

void init\_dex(){

dex[0]=-1;

for(int i=1;i<M-10;i++){

if(i&(i-1))dex[i]=dex[i-1];

else dex[i]=dex[i-1]+1;

}

}

struct RMQ{

int tmin[M][20];

void init(int n){

memset(tmin,inf,sizeof(tmin));

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

tmin[i][0]=a[i];

}

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

for(int j=0;(j+(1<<i))-1<n;j++){

int next=j+(1<<(i-1));

tmin[j][i]=min(tmin[j][i-1],tmin[next][i-1]);

}

}

}

int query(int l,int r){

int t1=r-l+1;

t1=dex[t1];

r-=(1<<t1)-1;

int ans=min(tmin[l][t1],tmin[r][t1]);

return ans;

}

}rmq\_tmin;

----------------------------------------------

2-dimension

int dex[M];

int ma[M][M];

void init\_dex(){

dex[0]=-1;

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

if(i&(i-1))dex[i]=dex[i-1];

else dex[i]=dex[i-1]+1;

}

}

struct RMQ{

int tmin[M][M][9][9];

void init(int n){

memset(tmin,inf,sizeof(tmin));

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

for(int j=0;j<n;j++){

tmin[i][j][0][0]=ma[i][j];

}

}

for(int i=0;(1<<i)<=n;i++){

for(int j=0;(1<<j)<=n;j++){

if(i==0&&j==0) continue;

for(int x1=0;x1+(1<<i)-1<n;x1++){

for(int y1=0;y1+(1<<j)-1<n;y1++){

int nextx=x1+(1<<(i-1));

int nexty=y1+(1<<(j-1));

if(i==0){

tmin[x1][y1][i][j]=min(tmin[x1][y1][i][j-1],tmin[x1][nexty][i][j-1]);

}

else if(j==0){

tmin[x1][y1][i][j]=min(tmin[x1][y1][i-1][j],tmin[nextx][y1][i-1][j]);

}

else{

int t1=min(tmin[x1][y1][i-1][j-1],tmin[x1][nexty][i-1][j-1]);

int t2=min(tmin[nextx][y1][i-1][j-1],tmin[nextx][nexty][i-1][j-1]);

tmin[x1][y1][i][j]=min(t1,t2);

}

}

}

}

}

}

int query(int x1,int y1,int x2,int y2){

int p1=x2-x1+1;p1=dex[p1];

int p2=y2-y1+1;p2=dex[p2];

x2-=(1<<p1)-1;

y2-=(1<<p2)-1;

int t1=min(tmin[x1][y1][p1][p2],tmin[x1][y2][p1][p2]);

int t2=min(tmin[x2][y1][p1][p2],tmin[x2][y2][p1][p2]);

return min(t1,t2);

}

}rmq;

------------------------------------------

rmq in tree

struct G{

int head[M],en;

struct E{

int u,v,next;

}e[M<<1];

void init(){

memset(head,-1,sizeof(head));en=0;

}

void add(int u,int v){

e[en].u=u;e[en].v=v;e[en].next=head[u];head[u]=en++;

}

}g1;

int cost[M];

int stk[M],dis[M];

int tmax[M][17],pre[M][17];

void dfs(int u,int leavl){

dis[u]=leavl;

stk[leavl]=u;

tmax[u][0]=cost[u];pre[u][0]=u;

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

int next=stk[leavl-(1<<(i-1))];

tmax[u][i]=max(tmax[u][i-1],tmax[next][i-1]);

pre[u][i]=stk[leavl-(1<<(i))+1];

}

for(int i=g1.head[u];i!=-1;i=g1.e[i].next){

int v=g1.e[i].v;

dfs(v,leavl+1);

}

}

int lowbit(int x){

return x&(-x);

}

int bit(int x){

int pa=-1;

while(x){

pa++;

x=x>>1;

}

return pa;

}

int query(int u,int k){

int tr=u;

int tmax1=0;

while(k){

int t1=lowbit(k);

k-=t1;

t1=bit(t1);

tmax1=max(tmax1,tmax[tr][t1]);

if(k==0) break;

tr=pre[tr][t1];

tr=pre[tr][1];

}

return tmax1;

}

## tree\_divide

#include<cstdio>

#include<cstring>

#define inf 0x3f3f3f3f

#define M 200020

struct G{

int head[M],en;

struct E{

int u,v,cost,next,flag;

}e[M<<2];

void init(){

memset(head,-1,sizeof(head));

en=0;

}

void add(int u,int v,int cost){

e[en].flag=0;

e[en].u=u;e[en].v=v;e[en].cost=cost;e[en].next=head[u];head[u]=en++;

}

}g1;

int mark[M<<4];

int dis[M],fa[M],son[M],mc[M],lv[M],m;

int my\_que[M<<2];

int my\_sta[M<<2];

int find\_root(int pos,int siz){

int qh=0,qt=-1,cen=0;

int top=-1;

fa[pos]=0;

my\_que[++qt]=pos;

while(qt>=qh){

int u=my\_que[qh++];

son[u]=1,mc[u]=0;

for(int i=g1.head[u];~i;i=g1.e[i].next){

int v=g1.e[i].v;

if(v==fa[u]||g1.e[i].flag) continue;

fa[v]=u;

my\_que[++qt]=v;

}

my\_sta[++top]=u;

}

mc[cen]=siz;

while(top>=0){

int u=my\_sta[top--];

son[fa[u]]+=son[u];

if(mc[fa[u]]<son[u]) mc[fa[u]]=son[u];

if(mc[u]<siz-son[u]) mc[u]=siz-son[u];

if(mc[u]<mc[cen]) cen=u;

}

return cen;

}

int stk[M<<2][2];

int ans;

int top;

void dfs(int u,int p,int cost){

dis[u]=dis[p]+cost;

lv[u]=lv[p]+1;

if(lv[u]>ans) return ;

if(dis[u]<=m){

stk[top][0]=dis[u];

stk[top++][1]=lv[u];

if(lv[u]+mark[m-dis[u]]<ans){

ans=lv[u]+mark[m-dis[u]];

}

}

else return ;

for(int i=g1.head[u];~i;i=g1.e[i].next){

int v=g1.e[i].v;

if(v==p||g1.e[i].flag) continue;

dfs(v,u,g1.e[i].cost);

}

}

int change[M<<2],en;

void divide(int u){

dis[u]=0;

lv[u]=0;

mark[0]=0;

en=0;

for(int i=g1.head[u];i!=-1;i=g1.e[i].next){

top=0;

int v=g1.e[i].v;

if(g1.e[i].flag) continue;

dfs(v,u,g1.e[i].cost);

for(int j=0;j<top;j++){

int dis1=stk[j][0];

int cost1=stk[j][1];

if(mark[dis1]>cost1){

mark[dis1]=cost1;

change[en++]=dis1;

}

}

}

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

mark[change[i]]=inf;

}

for(int i=g1.head[u];~i;i=g1.e[i].next){

int v=g1.e[i].v;

if(g1.e[i].flag) continue;

g1.e[i^1].flag=1;

int rt=find\_root(v,son[v]);

divide(rt);

}

}

int main(){

int n;

while(~scanf("%d%d",&n,&m)){

g1.init();

for(int i=0;i<n-1;i++){

int u,v,cost;scanf("%d%d%d",&u,&v,&cost);u++,v++;

g1.add(u,v,cost);g1.add(v,u,cost);

}

memset(mark,inf,sizeof(mark));

mark[0]=0;

ans=inf;

int rt=find\_root(1,n);

divide(rt);

if(ans==inf){

puts("-1");

}

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

}

return 0;

}

/\*

5 3

0 1 1

1 2 10

1 3 4

2 4 2

\*/

## 链路剖分

/\*

链路剖分 边权型

\*/

#include<cstdio>

#include<cstring>

#include<iostream>

#define M 20000

#define lson (rt<<1)

#define rson (rt<<1|1)

#define inf 0x3f3f3f3f

using namespace std;

struct G{

int head[M],en;

struct E{

int u,v,next,cost;

}e[M<<2];

void init(){

memset(head,-1,sizeof(head));en=0;

}

void add(int u,int v,int cost){

e[en].u=u;e[en].v=v;e[en].cost=cost;e[en].next=head[u];head[u]=en++;

}

}g1;

struct T{

struct E{

int l,r,tmax;

}e[M<<2];

int value[M];

void push\_up(int rt){

e[rt].tmax=max(e[lson].tmax,e[rson].tmax);

}

void build(int l,int r,int rt){

e[rt].l=l;e[rt].r=r;e[rt].tmax=0;

if(l==r){

e[rt].tmax=value[l];

return ;

}

int mid=(l+r)>>1;

build(l,mid,lson);

build(mid+1,r,rson);

push\_up(rt);

}

void update(int ul,int new\_flag,int rt){

if(e[rt].l==ul&&e[rt].r==ul){

e[rt].tmax=new\_flag;

return ;

}

int mid=(e[rt].l+e[rt].r)>>1;

if(ul<=mid) update(ul,new\_flag,lson);

else update(ul,new\_flag,rson);

push\_up(rt);

}

int query(int ul,int ur,int rt){

if(ul<=e[rt].l&&ur>=e[rt].r){

return e[rt].tmax;

}

int mid=(e[rt].l+e[rt].r)>>1;

if(ur<=mid) return query(ul,ur,lson);

else if(ul>mid) return query(ul,ur,rson);

else{

int t1=query(ul,ur,lson);

int t2=query(ul,ur,rson);

return max(t1,t2);

}

}

}tr;

int my\_que[M<<2],top[M],son[M],en,sz[M],fa[M],dep[M],hash[M],value[M];

void build\_tree(){

int l=0,r=-1;

my\_que[++r]=1;

dep[1]=1;

sz[0]=0;

fa[1]=0;

while(l<=r){

int u=my\_que[l++];son[u]=0;sz[u]=1;

for(int i=g1.head[u];i!=-1;i=g1.e[i].next){

int v=g1.e[i].v;

if(v==fa[u]) continue;

fa[v]=u;

value[v]=g1.e[i].cost;

dep[v]=dep[u]+1;

my\_que[++r]=v;

}

}

for(int i=r;i>=0;i--){

int u=my\_que[i];

for(int j=g1.head[u];j!=-1;j=g1.e[j].next){

int v=g1.e[j].v;

if(v==fa[u]) continue;

sz[u]+=sz[v];

if(sz[son[u]]<sz[v]) son[u]=v;

}

}

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

int u=my\_que[i];

if(son[fa[u]]==u) top[u]=top[fa[u]];

else top[u]=u;

}

}

void re\_build(int n){

en=1;

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

if(top[i]!=i) continue;

hash[i]=en;

tr.value[en++]=value[i];

for(int j=i;j;j=son[j]){

if(son[j]==0) continue;

hash[son[j]]=en;

tr.value[en++]=value[son[j]];

}

}

tr.build(1,en,1);

}

void work(int n){

build\_tree();

re\_build(n);

}

void change(int l,int new\_flag){

int u=g1.e[(l<<1)-1].u;

int v=g1.e[(l<<1)-1].v;

if(dep[v]<dep[u]) swap(u,v);

tr.update(hash[v],new\_flag,1);

}

int query(int x,int y){

int ans=-inf;

while(top[x]!=top[y]){

if(dep[top[x]]<dep[top[y]]) swap(x,y);

ans=max(ans,tr.query(hash[top[x]],hash[x],1));

x=fa[top[x]];

}

if(x==y) return ans;

if(dep[y]<dep[x]) swap(x,y);

ans=max(ans,tr.query(hash[son[x]],hash[y],1));

return ans;

}

int main(){

int cas;while(~scanf("%d",&cas)){

while(cas--){

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

g1.init();

for(int i=0;i<n-1;i++){

int u,v,cost;scanf("%d%d%d",&u,&v,&cost);

g1.add(u,v,cost);

g1.add(v,u,cost);

}

work(n);

char str[100];

while(1){

scanf("%s",str);

if(str[0]=='D') break;

if(str[0]=='C'){

int t1,t2;scanf("%d%d",&t1,&t2);

change(t1,t2);

}

if(str[0]=='Q'){

int t1,t2;scanf("%d%d",&t1,&t2);

int ans=query(t1,t2);

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

}

}

}

}

return 0;

}

/\*

1

14

1 2 9

1 3 7

1 4 1

2 5 13

2 6 10

6 11 11

6 12 12

3 7 8

4 8 6

4 9 2

4 10 5

9 13 3

13 14 4

\*/

/\*

链路剖分 点权型

\*/

#include<cstdio>

#include<cstring>

#include<iostream>

#define M 40000

#define lson (rt<<1)

#define rson (rt<<1|1)

#define inf 0x3f3f3f3f

using namespace std;

struct G{

int head[M],en;

struct E{

int u,v,next,cost;

}e[M<<2];

void init(){

memset(head,-1,sizeof(head));en=0;

}

void add(int u,int v,int cost){

e[en].u=u;e[en].v=v;e[en].cost=cost;e[en].next=head[u];head[u]=en++;

}

}g1;

struct A{

int tsum,tmax;

A(){

tsum=0;

tmax=-inf;

}

};

struct T{

struct E{

int l,r,tmax,tsum;

}e[M<<2];

int value[M];

void push\_up(int rt){

e[rt].tmax=max(e[lson].tmax,e[rson].tmax);

e[rt].tsum=e[lson].tsum+e[rson].tsum;

}

void build(int l,int r,int rt){

e[rt].l=l;e[rt].r=r;e[rt].tmax=-inf;e[rt].tsum=0;

if(l==r){

e[rt].tmax=value[l];

e[rt].tsum=value[l];

return ;

}

int mid=(l+r)>>1;

build(l,mid,lson);

build(mid+1,r,rson);

push\_up(rt);

}

void update(int ul,int new\_flag,int rt){

if(e[rt].l==ul&&e[rt].r==ul){

e[rt].tmax=new\_flag;

e[rt].tsum=new\_flag;

return ;

}

int mid=(e[rt].l+e[rt].r)>>1;

if(ul<=mid) update(ul,new\_flag,lson);

else update(ul,new\_flag,rson);

push\_up(rt);

}

A query(int ul,int ur,int rt){

if(ul<=e[rt].l&&ur>=e[rt].r){

A a1;a1.tmax=e[rt].tmax;

a1.tsum=e[rt].tsum;

return a1;

}

int mid=(e[rt].l+e[rt].r)>>1;

if(ur<=mid) return query(ul,ur,lson);

else if(ul>mid) return query(ul,ur,rson);

else{

A a1=query(ul,ur,lson);

A a2=query(ul,ur,rson);

a1.tmax=max(a1.tmax,a2.tmax);

a1.tsum+=a2.tsum;

return a1;

}

}

}tr;

int my\_que[M<<2],top[M],son[M],en,sz[M],fa[M],dep[M],hash[M],value[M];

void build\_tree(){

int l=0,r=-1;

my\_que[++r]=1;

dep[1]=1;

sz[0]=0;

fa[1]=0;

while(l<=r){

int u=my\_que[l++];son[u]=0;sz[u]=1;

for(int i=g1.head[u];i!=-1;i=g1.e[i].next){

int v=g1.e[i].v;

if(v==fa[u]) continue;

fa[v]=u;

dep[v]=dep[u]+1;

my\_que[++r]=v;

}

}

for(int i=r;i>=0;i--){

int u=my\_que[i];

for(int j=g1.head[u];j!=-1;j=g1.e[j].next){

int v=g1.e[j].v;

if(v==fa[u]) continue;

sz[u]+=sz[v];

if(sz[son[u]]<sz[v]) son[u]=v;

}

}

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

int u=my\_que[i];

if(son[fa[u]]==u) top[u]=top[fa[u]];

else top[u]=u;

}

}

void re\_build(int n){

en=1;

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

if(top[i]!=i) continue;

for(int j=i;j;j=son[j]){

hash[j]=en;

tr.value[en++]=value[j];

}

}

tr.build(1,en,1);

}

void work(int n){

build\_tree();

re\_build(n);

}

void change(int l,int new\_flag){

tr.update(hash[l],new\_flag,1);

}

A query(int x,int y){

A ans,a1;;

while(top[x]!=top[y]){

if(dep[top[x]]<dep[top[y]]) swap(x,y);

a1=tr.query(hash[top[x]],hash[x],1);

ans.tmax=max(ans.tmax,a1.tmax);

ans.tsum+=a1.tsum;

x=fa[top[x]];

}

if(dep[y]<dep[x]) swap(x,y);

a1=tr.query(hash[x],hash[y],1);

ans.tmax=max(ans.tmax,a1.tmax);

ans.tsum+=a1.tsum;

return ans;

}

int main(){

int n;while(~scanf("%d",&n)){

g1.init();

for(int i=0;i<n-1;i++){

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

g1.add(u,v,0);

g1.add(v,u,0);

}

for(int i=1;i<=n;i++) scanf("%d",&value[i]);

work(n);

int q;scanf("%d",&q);

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

char str[100];

scanf("%s",str);

if(str[0]=='C'){

int t1,t2;scanf("%d%d",&t1,&t2);

change(t1,t2);

}

else{

int t1,t2;scanf("%d%d",&t1,&t2);

A ans;ans=query(t1,t2);

if(str[1]=='M'){

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

}

else{

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

}

}

}

}

return 0;

}

# DP

MAX increase comment subsequencs

O(n^2)

#include<cstdio>

#include<cstring>

#define M 1080

int dp[M][M];

int a[M];int b[M];

int main(){

int cas;while(~scanf("%d",&cas)){

while(cas--){

int n1;scanf("%d",&n1);

for(int i=1;i<=n1;i++) scanf("%d",&a[i]);

int n2;scanf("%d",&n2);

for(int i=1;i<=n2;i++) scanf("%d",&b[i]);

for(int i=1;i<=n1;i++){

int tmax=0;

for(int j=1;j<=n2;j++){

dp[i][j]=dp[i-1][j];

if(a[i]>b[j]&&dp[i][j]>tmax) tmax=dp[i][j];

if(a[i]==b[j]) dp[i][j]=tmax+1;

}

}

int tmax=0;

for(int i=1;i<=n2;i++){

if(dp[n1][i]>tmax) tmax=dp[n1][i];

}

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

}

}

return 0;

}

## Tree\_dp

对于最大独立集：

dp[i][0] 表示不取i的最大值

dp[i][1]表示取i的最大值

表示不取父亲 dp[fa][0]+=max(dp[son][0],son[son][1]);可以去儿子节点或者不取；

表示取父亲 dp[fa][1]+=dp[son][0] 表示不能取儿子节点

这个题目就是判断唯一性比较复杂用mark[i][0] 标记i 的0 ，1状态是否能取到

对于mark[fa][1]=(mark[son][0]&all) 所有儿子节点的mark[son][0]必须唯一

对于mark[fa][0] 如果有一个儿子dp[son][0]=dp[son][1] 明显为0

如果取某个儿子的某种状态mark[son][?]必须全为1.

----------------------------------

对于最小点覆盖

fa为父亲节点

son为儿子节点

dp[i][1] 表示选了i的最小值

dp[i][0] 表示不选i的最小值

明显 dp[fa][0]+=dp[fa][1];

dp[fa][1]+=min(dp[son][0],dp[son][1])

---------------------------------------------

树形dp 3连发 支配集和覆盖集有不同的定义，覆盖集一定是支配集，但是支配集不一定是覆盖集，覆盖集是每一条边必须有一个点被选择。

但是支配集只要所有点 1包含在集合内 2 和他相连点在集合内。

比如 1->2->3->4->5->6 覆盖就最小为3 但是支配集为2（选取2 和5 就可以了）

3个状态 dp[i][0] 表示不选i时候最小值

dp[i][1] 表示选了i时候最小值

dp[i][2] 表示不选i但是i有至少一个儿子支配

dp[fa][1]=min(dp[son][0],dp[son][1],dp[son][2])+1;

dp[fa][0]=min(dp[son][1],dp[son][2]);

dp[fa][2]=min(dp[son][1],dp[son][2])+max(min(dp[son][1]-dp[son][2],0);(选其一个最优的儿子然后变成1)；

----------------------------------------------

其他树形dp多校训练中能体现出的是，记录最长 次长 次次长，在2次dfs 通过求解

# Graph

## Compone

### 2\_sat\_dfs

/\*

2\_sat (V\*E)

很方便打出字典序最小的解

以hdu 1814 为题

\*/

#include<cstdio>

#include<cstring>

#define M 20000

struct G{

int head[M],en;

struct E{

int u,v,next;

}e[M<<4];

void init(){

memset(head,-1,sizeof(head));

en=0;

}

void add(int u,int v){

e[en].u=u;e[en].v=v;

e[en].next=head[u];

head[u]=en++;

}

}g1;

int mark[M],stk[M],c;

void initt(){

memset(mark,0,sizeof(mark));

}

int dfs(int u){

if(mark[u^1]) return 0;

if(mark[u]) return 1;

mark[u]=1;

mark[u^1]=0;

stk[c++]=u;

for(int i=g1.head[u];i!=-1;i=g1.e[i].next){

int v=g1.e[i].v;

if(!dfs(v)) return 0;

}

return 1;

}

int two\_sat(int n){

initt();

for(int i=0;i<2\*n;i+=2){

if(!mark[i]&&!mark[i^1]){

c=0;

if(!dfs(i)){

while(c>0) mark[stk[--c]]=0;

if(!dfs(i^1)) return 0;

}

}

}

return 1;

}

int main(){

int n,m;

while(~scanf("%d%d",&n,&m)){

g1.init();

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

int t1,t2;scanf("%d%d",&t1,&t2);

t1--;t2--;

g1.add(t1,t2^1);

g1.add(t2,t1^1);

}

int ans=two\_sat(n);

if(!ans){

puts("NIE");

continue;

}

int pflag=0;

for(int i=0;i<2\*n;i+=2){

if(mark[i]){

printf("%d\n",i+1);

}else{

printf("%d\n",i+2);

}

}

}

return 0;

}

### s\_sat\_tarjan\_topsort

/\*

2-sat tarjan topsort

Êä³öÈÎÒâ½â

O(E)hit 1917

\*/

#include<cstdio>

#include<cstring>

#include<stack>

#include<queue>

#define M 20000

using namespace std;

struct G{

int head[M],en;

struct E{

int u,v,next;

}e[M<<4];

void init(){

memset(head,-1,sizeof(head));

en=0;

}

void add(int u,int v){

e[en].u=u;e[en].v=v;e[en].next=head[u];

head[u]=en++;

}

}g1,g2;

int subid[M],subnum,dfs\_clock,pre[M],low[M],last[M];

stack<int > stk1;

void initt(){

memset(subid,0,sizeof(subid));

memset(pre,0,sizeof(pre));

memset(low,0,sizeof(low));

subnum=dfs\_clock=0;

}

void dfs(int u){

stk1.push(u);

pre[u]=low[u]=++dfs\_clock;

for(int i=g1.head[u];i!=-1;i=g1.e[i].next){

int v=g1.e[i].v;

if(!pre[v]){

dfs(v);

low[u]=min(low[u],low[v]);

}

else if(!subid[v]){

low[u]=min(low[u],pre[v]);

}

}

if(pre[u]==low[u]){

++subnum;

while(1){

int t1=stk1.top();stk1.pop();

subid[t1]=subnum;

if(t1==u){

last[subnum]=t1;

break;

}

}

}

}

int tarjan(int n){

initt();

for(int i=0;i<2\*n;i++){

if(!pre[i]) dfs(i);

}

for(int i=0;i<2\*n;i+=2){

if(subid[i]==subid[i^1]) return 0;

}

return 1;

}

int indgree[M],color[M];

queue<int> q1;

void init\_top(){

memset(indgree,0,sizeof(indgree));

memset(color,-1,sizeof(color));

g2.init();

while(!q1.empty()) q1.pop();

for(int i=0;i<g1.en;i++){

int t1=g1.e[i].u;

int t2=g1.e[i].v;

t1=subid[t1];t2=subid[t2];

if(t1!=t2){

g2.add(t2,t1);

indgree[t1]++;

}

}

}

void dfs2(int u){

color[u]=0;

for(int i=g2.head[u];i!=-1;i=g2.e[i].next){

int v=g2.e[i].v;

if(color[v]==-1){

dfs2(v);

}

}

}

void top\_sort(int n){

init\_top();

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

if(indgree[i]==0){

q1.push(i);

}

}

while(!q1.empty()){

int u=q1.front();q1.pop();

for(int i=g2.head[u];i!=-1;i=g2.e[i].next){

int v=g2.e[i].v;

indgree[v]--;

if(indgree[v]==0) q1.push(v);

}

if(color[u]==-1){

color[u]=1;

int v=last[u];

v=subid[v^1];

dfs2(v);

}

}

}

int main(){

int n,m;

while(~scanf("%d%d",&n,&m)){

g1.init();

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

int t1,t2;scanf("%d%d",&t1,&t2);

t1--;t2--;

g1.add(t1,t2^1);

g1.add(t2,t1^1);

}

int ans=tarjan(n);

if(!ans){

puts("NIE");

}

else{

top\_sort(subnum);

for(int i=0;i<2\*n;i+=2){

if(color[subid[i]]==1){

printf("%d\n",i+1);

}

else{

printf("%d\n",i+2);

}

}

}

}

return 0;

}

/\*

3 6

1 4

1 3

2 4

2 4

2 6

2 6

\*/

### lca\_tarjan

#define M 60000

using namespace std;

/\*

init(n) query(n) off\_line

must add 2 query

\*/

struct G{

int head[M],en;

struct E{

int u,v,next;

}e[M<<3];

void init(){

memset(head,-1,sizeof(head));en=0;

}

void add(int u,int v){

e[en].u=u;e[en].v=v;e[en].next=head[u];head[u]=en++;

}

}g1,query;

int ans[M],fa[M],mo[M];

int flag[M];

bool vis[M];

void init\_lca(int n){

for(int i=0;i<=n;i++) fa[i]=i;

memset(vis,0,sizeof(vis));

}

int find\_fa(int u){

if(u==fa[u]) return u;

return fa[u]=find\_fa(fa[u]);

}

void un(int u,int v){

int fa1=find\_fa(u);

int fa2=find\_fa(v);

fa[fa2]=fa1;

}

void dfs(int u,int p){

fa[u]=mo[u]=u;

for(int i=g1.head[u];i!=-1;i=g1.e[i].next){

int v=g1.e[i].v;

if(v==p) continue;

dfs(v,p);

un(u,v);

}

vis[u]=1;

for(int i=query.head[u];i!=-1;i=query.e[i].next){

int v=query.e[i].v;

if(vis[v]){

int t2=mo[find\_fa(v)];

ans[i/2]=t2;

}

}

}

### lca\_rmq

#define M 3000

#define inf 0x3f3f3f3f

/\*

init(n\*(log(n)) query(1)

first init\_dex;

before dfs en=0

\*/

using namespace std;

struct G{

int head[M],en;

struct E{

int u,v,next;

}e[M<<2];

void init(){

memset(head,-1,sizeof(head));en=0;

}

void add(int u,int v){

e[en].u=u;e[en].v=v;e[en].next=head[u];head[u]=en++;

}

}g1;

int lv[M];

int aa[M],bb[M],en;

void dfs(int u,int level){

lv[u]=level;

aa[en]=u;

bb[en++]=level;

for(int i=g1.head[u];i!=-1;i=g1.e[i].next){

int v=g1.e[i].v;

dfs(v,level+1);

aa[en]=u;bb[en++]=level;

}

}

int dex[M];

void init\_dex(){

dex[0]=-1;

for(int i=1;i<M-10;i++){

if(i&(i-1))dex[i]=dex[i-1];

else dex[i]=dex[i-1]+1;

}

}

struct R{

int raa[M][20],rbb[M][20];

int head[M];

void init(){

memset(head,-1,sizeof(head));

memset(rbb,inf,sizeof(inf));

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

int t1=aa[i];

if(head[t1]==-1) head[t1]=i;

}

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

rbb[i][0]=bb[i];

raa[i][0]=aa[i];

}

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

for(int j=0;j+(1<<i)<en;j++){

int next=j+(1<<(i-1));

if(rbb[j][i-1]<=rbb[next][i-1]){

rbb[j][i]=rbb[j][i-1];

raa[j][i]=raa[j][i-1];

}

else{

rbb[j][i]=rbb[next][i-1];

raa[j][i]=raa[next][i-1];

}

}

}

}

int query(int l,int r){

l=head[l];r=head[r];

if(l>r) swap(l,r);

int p2=dex[r-l+1];

r-=(1<<p2)-1;

if(rbb[l][p2]<=rbb[r][p2]) return raa[l][p2];

else return raa[r][p2];

}

}rmq;

### tarjan\_bridge

/\*

找桥，双边联通分量 割点 可重边

\*/

#include<cstdio>

#include<cstring>

#define M 15000

using namespace std;

struct G{

int head[M],en;

struct E{

int u,v,next;

}e[M<<2];

void init(){

memset(head,-1,sizeof(head));

en=0;

}

void add(int u,int v){

e[en].u=u;e[en].v=v;e[en].next=head[u];head[u]=en++;

}

}g1;

int pre[M],low[M],subid[M],dfs\_clock,sub\_num,is\_cut[M],is\_brige[M];

int stk[M],top;

void init(){

memset(pre,-1,sizeof(pre));

memset(is\_cut,0,sizeof(is\_cut));

memset(is\_brige,0,sizeof(is\_brige));

dfs\_clock=0;sub\_num=0;

top=-1;

}

void dfs(int u,int id){

pre[u]=low[u]=++dfs\_clock;

stk[++top]=u;

int child=0;

for(int i=g1.head[u];i!=-1;i=g1.e[i].next){

if((i^1)==id) continue;

int v=g1.e[i].v;

child++;

if(pre[v]==-1){

dfs(v,i);

if(low[v]<low[u]) low[u]=low[v];

if(low[v]>pre[u]){

is\_brige[i]=1;

}

if(low[v]>=pre[u]){

is\_cut[u]++;

}

}

else{

if(pre[v]<low[u]) low[u]=pre[v];

}

}

if(pre[u]==low[u]){

sub\_num++;

while(top>=0){

int t1=stk[top--];

subid[t1]=sub\_num;

if(t1==u) break;

}

}

if(child==1&&id==-1) is\_cut[u]=0;

}

int tarjan(int n){

init();

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

if(pre[i]==-1){

dfs(i,-1);

}

}

return sub\_num;

}

### tarjan\_cut\_2point

#include<cstdio>

#include<cstring>

#define M 1010

using namespace std;

struct G{

int head[M<<2],en;

struct E{

int u,v,next;

}e[M<<10];

void init(){

memset(head,-1,sizeof(head));

en=0;

}

void add(int u,int v){

e[en].u=u;e[en].v=v;e[en].next=head[u];head[u]=en++;

}

}g1,g2;

int stk[M<<10],pre[M],dfs\_clock,sub\_num,used[M],low[M],is\_cut[M];

int vis[M],top,color[M];

void init(){

top=-1;

g2.init();

memset(vis,0,sizeof(vis));

memset(pre,0,sizeof(pre));

memset(used,0,sizeof(used));

memset(is\_cut,0,sizeof(is\_cut));

dfs\_clock=sub\_num=0;

}

void dfs(int u,int p){

int child=0;

pre[u]=low[u]=++dfs\_clock;

for(int i=g1.head[u];i!=-1;i=g1.e[i].next){

int v=g1.e[i].v;

if(v==p) continue;

if(!pre[v]){

stk[++top]=i;

child++;

dfs(v,u);

if(low[v]<low[u]){

low[u]=low[v];

}

if(low[v]>=pre[u]){

is\_cut[u]=1;

++sub\_num;

int tsum=0;

while(1){

int t1=stk[top--];

int u1=g1.e[t1].u;int v1=g1.e[t1].v;

if(vis[u1]!=sub\_num) tsum++,vis[u1]=sub\_num,g2.add(sub\_num,u1);

if(vis[v1]!=sub\_num) tsum++,vis[v1]=sub\_num,g2.add(sub\_num,v1);

if(t1==i) break;

}

}

}

else{

if(pre[v]<low[u]) {

stk[++top]=i;//mark this is trick

low[u]=pre[v];

}

}

}

if(p==-1&&child<2) is\_cut[u]=0;

}

void tarjan(int n){

init();

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

if(!pre[i])dfs(i,-1);

}

}

## Flow

### sap\_god\_class

#include<iostream>

#include<cstdio>

#include<cstring>

#include<cmath>

#include<algorithm>

#include<climits>

#define inf 0x3f3f3f3f

using namespace std;

const int NV = 158000;

const int NE = 450002;

struct ISAP {

int n, size;

int head[NV];

int dis[NV], gap[NV], pre[NV], cur[NV];

int maxflow;

struct Edge {

int v, w, next;

Edge () {}

Edge (int V, int NEXT, int W = 0) : v(V), next(NEXT), w(W) { }

}E[NE];

void init(int x) {

n = x, size = 0;

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

head[i] = -1;

}

}

inline void insert(int u, int v, int w = 0) {

E[size] = Edge(v, head[u], w);

head[u] = size++;

E[size] = Edge(u, head[v], 0);

head[v] = size++;

}

int MaxFlow(int src, int des) {

maxflow = 0;

gap[0] = n;

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

dis[i] = gap[i] = 0;

}

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

cur[i] = head[i];

}

int u = pre[src] = src;

int aug = -1;

while (dis[src] < n) {//结束条件1

loop:

for (int &i = cur[u]; i != -1; i = E[i].next) {

int v = E[i].v;

if (E[i].w && dis[u] == dis[v] + 1) {

aug = min(aug, E[i].w);

pre[v] = u;

u = v;

if (v == des) {//找到一条增广路，更新

maxflow += aug;

// //修改残余网络

for (u = pre[u]; v != src; v = u, u = pre[u]) {

E[cur[u]].w -= aug;//正向边

E[cur[u]^1].w += aug;//反向边

}

aug = INT\_MAX;

}//if

goto loop;

}//for

}//for

//寻找最小的距离标号，并修改当前点 为最小的标号+1

int mdis = n;

for (int i = head[u]; i != -1; i = E[i].next) {

int v = E[i].v;

if (E[i].w && mdis > dis[v]) {

cur[u] = i;

mdis = dis[v];

}

}//for

//GAP 优化 断层则跳出 结束条件2

if ((--gap[dis[u]]) == 0) {

break;

}

gap[dis[u] = mdis + 1]++;//将拥有该标号的数量加1

u = pre[u];//当前节点 迁移一个

}//while

return maxflow;

}//ISAP

}g1;

int main() {

int sp,tp;

int n,m;

while(~scanf("%d%d",&n,&m)){

g1.init(n+m+10);

sp=0;tp=n+m+1;

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

int t1;scanf("%d",&t1);

g1.insert(i+m,tp,t1);

}

int tsum=0;

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

int t1,t2,t3;scanf("%d%d%d",&t1,&t2,&t3);

tsum+=t3;

g1.insert(sp,i,t3);

g1.insert(i,t1+m,inf);

g1.insert(i,t2+m,inf);

}

int ans=g1.MaxFlow(0,n+m+1);

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

}

return 0;

}

### Dinic\_myself

#include<cstdio>

#include<queue>

#include<iostream>

#include<cstring>

#define M 1000000

#define N 58080

#define inf 0x3f3f3f3f

using namespace std;

int min(int x,int y){

if(x>y) return y;return x;

}

struct G{

int head[N],en;

struct E{

int u,v,cap,next;

}e[M];

void init(){

memset(head,-1,sizeof(head));en=0;

}

void add(int u,int v,int cap){

e[en].u=u;e[en].v=v;e[en].cap=cap;e[en].next=head[u];head[u]=en++;

e[en].u=v;e[en].v=u;e[en].cap=0 ;e[en].next=head[v];head[v]=en++;

}

}g1;

int dis[N],dinic\_sp,dinic\_tp;

int bfs(){

queue<int> q1;

q1.push(dinic\_sp);

memset(dis,-1,sizeof(dis));

dis[dinic\_sp]=0;

while(!q1.empty()){

int u=q1.front();q1.pop();

for(int i=g1.head[u];i!=-1;i=g1.e[i].next){

int v=g1.e[i].v;

if(dis[v]==-1&&g1.e[i].cap>0){

dis[v]=dis[u]+1;

q1.push(v);

}

}

}

if(dis[dinic\_tp]==-1) return 0;

return 1;

}

int dfs(int u,int tmax){

if(u==dinic\_tp){

return tmax;

}

int tsum=0;

for(int i=g1.head[u];i!=-1;i=g1.e[i].next){

int v=g1.e[i].v;

if(dis[v]==dis[u]+1&&g1.e[i].cap>0){

int t1=min(g1.e[i].cap,tmax-tsum);

t1=dfs(v,t1);

g1.e[i].cap-=t1;

g1.e[i^1].cap+=t1;

tsum+=t1;

}

}

return tsum;

}

int dinic(int sp,int tp){

dinic\_sp=sp;dinic\_tp=tp;

int tsum=0;

while(bfs()){

tsum+=dfs(sp,inf);

}

return tsum;

}

int main(){

int n,m;

int tsum=0;

int sp,tp;

while(~scanf("%d%d",&n,&m)){

sp=0;tp=n+m+1;

g1.init();

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

int t1;scanf("%d",&t1);

g1.add(i+m,tp,t1);

}

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

int t1,t2,t3;scanf("%d%d%d",&t1,&t2,&t3);

g1.add(sp,i,t3);

g1.add(i,t1+m,inf);

g1.add(i,t2+m,inf);

tsum+=t3;

}

int ans=dinic(sp,tp);

tsum=tsum-ans;

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

}

return 0;

}

/\*

2 2

1 2 2

1 2 3

9 6

1 2 5

1 3 7

1 4 4

2 3 1

2 6 3

3 4 2

3 5 5

3 6 4

4 5 4

\*/

### mincost\_maxflow

#define inf 0x3f3f3f3f

using namespace std;

/\*

sumflow is max\_flow

\*/

const int MAXN=2300;

const int MAXM=100200;

const int INF=inf;

int sumflow;

struct Edge

{

int u,v,cap,cost,ednxt;

}edge[MAXM<<2];

int edn,head[MAXN],dist[MAXN],pp[MAXN];

bool vis[MAXN];

void init()

{

edn=0,memset(head,-1,sizeof(head));

}

void addedge(int u,int v,int cap,int cost)

{

edge[edn].u=u,edge[edn].v=v,edge[edn].cap=cap,edge[edn].cost=cost, edge[edn].ednxt=head[u],head[u]=edn++;

edge[edn].u=v,edge[edn].v=u,edge[edn].cap=0, edge[edn].cost=-cost,edge[edn].ednxt=head[v],head[v]=edn++;

}

int my\_queue[MAXN<<3];

int tp,hp;

bool spfa(int s,int t,int n)

{

int i,u,v;

hp=0;tp=-1;

memset(vis,false,sizeof(vis));

memset(pp,-1,sizeof(pp));

for(i=0;i<=n;i++) dist[i]=INF;

vis[s]=true,dist[s]=0;

my\_queue[++tp]=s;

while(tp>=hp)

{

u=my\_queue[tp--];vis[u]=false;

for(i=head[u];i!=-1;i=edge[i].ednxt)

{

v=edge[i].v;

if(edge[i].cap&&dist[v]>dist[u]+edge[i].cost)

{

dist[v]=dist[u]+edge[i].cost;

pp[v]=i;

if(!vis[v])

{

my\_queue[++tp]=v;

vis[v]=true;

}

}

}

}

if(dist[t]==INF)return false;

return true ;

}

int mcmf(int s,int t,int n)

{

int flow=0;

int i,minflow,mincost;

mincost=0;

while(spfa(s,t,n))

{

minflow=INF+1;

for(i=pp[t];i!=-1;i=pp[edge[i].u])

if(edge[i].cap<minflow)

minflow=edge[i].cap;

flow+=minflow;

for(i=pp[t];i!=-1;i=pp[edge[i].u])

{

edge[i].cap-=minflow;

edge[i^1].cap+=minflow;

}

mincost+=dist[t]\*minflow;

}

sumflow=flow;

return mincost;

}

### mincost\_zkw

/\*

ÊÊÓÃÓÚ³íÃÜÍ¼ÕýÁ÷

\*/

#include <iostream>

#include <cstring>

#include <cstdio>

#define MAXNODE 3000

#define MAXEDGE MAXNODE\*5

#define MIN(a,b) ((a)<(b)?(a):(b))

#define OPPOSITE(x) (((x)&1)?((x)+1):((x)-1))

#define INFINIT ~0U>>1

using namespace std;

int n,m;

int N,S,T;

int begin[MAXNODE+1],end[MAXEDGE+1],next[MAXEDGE+1],c[MAXEDGE+1],cost[MAXEDGE+1],d[MAXNODE+1],cur[MAXNODE+1];

bool hash[MAXNODE+1];

int Count = 0;

int aug(int u,int f){

if (u == T) return f;

hash[u] = true;

for (int now = cur[u]; now; now = next[now])

if (c[now]&&!hash[end[now]]&&d[u] == d[end[now]]+cost[now])

if (int tmp = aug(end[now],MIN(f,c[now])))

return c[now] -= tmp,c[OPPOSITE(now)] += tmp,cur[u] = now,tmp;

return 0;

}

bool modlabel(){

int tmp = INFINIT;

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

if (hash[i])

for (int now = begin[i]; now; now = next[now])

if (c[now]&&!hash[end[now]])

tmp = MIN(tmp,d[end[now]]+cost[now]-d[i]);

if (tmp == INFINIT)

return true;

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

if (hash[i])

hash[i] = false,d[i] += tmp;

return false;

}

int CostFlow(){

int costflow = 0,tmp;

while (true){

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

cur[i] = begin[i];

while (tmp = aug(S,~0U>>1)){

costflow += tmp\*d[S];

memset(hash,0,sizeof(hash));

}

if (modlabel())

break;

}

return costflow;

}

void AddEdge(int a,int b,int flow, int v){

Count++; next[Count] = begin[a]; begin[a] = Count; end[Count] = b; c[Count] = flow; cost[Count] = v;

Count++; next[Count] = begin[b]; begin[b] = Count; end[Count] = a; c[Count] = 0; cost[Count] = -v;

}

int main(){

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

// freopen("costflow.out","w",stdout);

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

while (m--){

int t1,t2,t3,t4;

scanf("%d%d%d%d",&t1,&t2,&t3,&t4);

AddEdge(t1,t2,t3,t4);

}

S = 1,T = N = n;

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

return 0;

}

### 带花树一般图匹配

/\*

dex form 0 to n-1 ans is double of

\*/

#include <cstdio>

#include <cstring>

#include <iostream>

#include <queue>

using namespace std;

const int N = 250;

// 并查集维护

int belong[N];

int findb(int x) {

return belong[x] == x ? x : belong[x] = findb(belong[x]);

}

void unit(int a, int b) {

a = findb(a);

b = findb(b);

if (a != b) belong[a] = b;

}

int match[N],n;

vector<int> e[N];

int Q[N], rear;

int next[N], mark[N], vis[N];

// 朴素算法求某阶段中搜索树上两点x, y的最近公共祖先r

int LCA(int x, int y) {

static int t = 0; t++;

while (true) {

if (x != -1) {

x = findb(x); // 点要对应到对应的花上去

if (vis[x] == t) return x;

vis[x] = t;

if (match[x] != -1) x = next[match[x]];

else x = -1;

}

swap(x, y);

}

}

void group(int a, int p) {

while (a != p) {

int b = match[a], c = next[b];

// next数组是用来标记花朵中的路径的，综合match数组来用，实际上形成了

// 双向链表，如(x, y)是匹配的，next[x]和next[y]就可以指两个方向了。

if (findb(c) != p) next[c] = b;

// 奇环中的点都有机会向环外找到匹配，所以都要标记成S型点加到队列中去，

// 因环内的匹配数已饱和，因此这些点最多只允许匹配成功一个点，在aug中

// 每次匹配到一个点就break终止了当前阶段的搜索，并且下阶段的标记是重

// 新来过的，这样做就是为了保证这一点。

if (mark[b] == 2) mark[Q[rear++] = b] = 1;

if (mark[c] == 2) mark[Q[rear++] = c] = 1;

unit(a, b); unit(b, c);

a = c;

}

}

// 增广

void aug(int s) {

for (int i = 0; i < n; i++) // 每个阶段都要重新标记

next[i] = -1, belong[i] = i, mark[i] = 0, vis[i] = -1;

mark[s] = 1;

Q[0] = s; rear = 1;

for (int front = 0; match[s] == -1 && front < rear; front++) {

int x = Q[front]; // 队列Q中的点都是S型的

for (int i = 0; i < (int)e[x].size(); i++) {

int y = e[x][i];

if (match[x] == y) continue; // x与y已匹配，忽略

if (findb(x) == findb(y)) continue; // x与y同在一朵花，忽略

if (mark[y] == 2) continue; // y是T型点，忽略

if (mark[y] == 1) { // y是S型点，奇环缩点

int r = LCA(x, y); // r为从i和j到s的路径上的第一个公共节点

if (findb(x) != r) next[x] = y; // r和x不在同一个花朵，next标记花朵内路径

if (findb(y) != r) next[y] = x; // r和y不在同一个花朵，next标记花朵内路径

// 将整个r -- x - y --- r的奇环缩成点，r作为这个环的标记节点，相当于论文中的超级节点

group(x, r); // 缩路径r --- x为点

group(y, r); // 缩路径r --- y为点

}

else if (match[y] == -1) { // y自由，可以增广，R12规则处理

next[y] = x;

for (int u = y; u != -1; ) { // 交叉链取反

int v = next[u];

int mv = match[v];

match[v] = u, match[u] = v;

u = mv;

}

break; // 搜索成功，退出循环将进入下一阶段

}

else { // 当前搜索的交叉链+y+match[y]形成新的交叉链，将match[y]加入队列作为待搜节点

next[y] = x;

mark[Q[rear++] = match[y]] = 1; // match[y]也是S型的

mark[y] = 2; // y标记成T型

}

}

}

}

int g[N][N];

int get\_match(){

for(int i=0;i<n;i++) e[i].clear();

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

for(int j=0;j<n;j++){

if(g[i][j]>0) e[i].push\_back(j);

}

}

int ans=0;

memset(match,-1,sizeof(match));

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

if(match[i]==-1) aug(i);

}

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

if(match[i]!=-1) ans++;

}

return ans/2;

}

/\*

4 3

1 3

2 3

2 4

\*/

### 二分匹配

支配集、极小支配集

支配集即一个点集，使得所有其他点至少有一个相邻点在集合里。或者说是一部分的“点”支配了所有“点”。极小支配集(minimal dominating set)：本身为支配集，其真子集都不是。最小支配集(minimum dominating set)：点最少的支配集。支配数(dominating number)：最小支配集的点数。

边支配集、极小边支配集

边支配集即一个边集，使得所有边至少有一条邻接边在集合里。或者说是一部分的“边”支配了所有“边”。极小边支配集(minimal edge dominating set)：本身是边支配集，其真子集都不是。最小边支配集(minimum edge dominating set)：边最少的边支配集。边支配数(edge dominating number)：最小边支配集的边数。

最小路径覆盖

最小路径覆盖(path covering)：是“路径” 覆盖“点”，即用尽量少的不相交简单路径覆盖有向无环图G的所有顶点，即每个顶点严格属于一条路径。路径的长度可能为0(单个点)。

最小路径覆盖数＝G的点数－最小路径覆盖中的边数。应该使得最小路径覆盖中的边数尽量多，但是又不能让两条边在同一个顶点相交。拆点：将每一个顶点i拆成两个顶点Xi和Yi。然后根据原图中边的信息，从X部往Y部引边。所有边的方向都是由X部到Y部。因此，所转化出的二分图的最大匹配数则是原图G中最小路径覆盖上的边数。因此由最小路径覆盖数＝原图G的顶点数－二分图的最大匹配数便可以得解。

匹配

匹配(matching)是一个边集，满足边集中的边两两不邻接。匹配又称边独立集(edge independent set)。

在匹配中的点称为匹配点(matched vertex)或饱和点；反之，称为未匹配点(unmatched vertex)或未饱和点。

交错轨(alternating path)是图的一条简单路径，满足任意相邻的两条边，一条在匹配内，一条不在匹配内。

增广轨(augmenting path)：是一个始点与终点都为未匹配点的交错轨。

最大匹配(maximum matching)是具有最多边的匹配。

匹配数(matching number)是最大匹配的大小。

完美匹配(perfect matching)是匹配了所有点的匹配。

完备匹配(complete matching)是匹配了二分图较小集合（二分图X，Y中小的那个）的所有点的匹配。

增广轨定理：一个匹配是最大匹配当且仅当没有增广轨。

所有匹配算法都是基于增广轨定理：一个匹配是最大匹配当且仅当没有增广轨。这个定理适用于任意图。

二分图的性质

二分图中，点覆盖数是匹配数。

(1) 二分图的最大匹配数等于最小覆盖数，即求最少的点使得每条边都至少和其中的一个点相关联，很显然直接取最大匹配的一段节点即可。

(2) 二分图的独立数等于顶点数减去最大匹配数，很显然的把最大匹配两端的点都从顶点集中去掉这个时候剩余的点是独立集，这是|V|-2\*|M|，同时必然可以从每条匹配边的两端取一个点加入独立集并且保持其独立集性质。

(3) DAG的最小路径覆盖，将每个点拆点后作最大匹配，结果为n-m，求具体路径的时候顺着匹配边走就可以，匹配边i→j',j→k',k→l'....构成一条有向路径。

（4）最大匹配数=左边匹配点+右边未匹配点。因为在最大匹配集中的任意一条边，如果他的左边没标记，右边被标记了，那么我们就可找到一条新的增广路，所以每一条边都至少被一个点覆盖。

（5）最小边覆盖=图中点的个数-最大匹配数=最大独立集。

# Math

## Matrix

int \*\* new\_matrix(int n){

int \*\* p1;

p1=new int \*[n];

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

p1[i]=new int [n];

for(int j=0;j<n;j++) p1[i][j]=0;

}

return p1;

}

int \*\*mul(int \*\*p1,int \*\*p2,int n,int m1,int com){

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

for(int j=0;j<m1;j++){

int tsum=0;

for(int k=0;k<com;k++){

tsum+=mul\_1(p1[i][k],p2[k][j]);

tsum=tsum%m;

}

tmp[i][j]=tsum;

}

}

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

for(int j=0;j<m1;j++){

p1[i][j]=tmp[i][j];

}

}

return p1;

}

int \*\* pow(int \*\*p1,int n,int time){

int \*\*ans=new\_matrix(M);

for(int i=0;i<n;i++)ans[i][i]=1;

int \*\*tmp1=p1;

int i=1;

while(i<=time){

if(time&i){

ans=mul(ans,tmp1,n,n,n);

}

tmp1=mul(tmp1,tmp1,n,n,n);

i=i<<1;

}

return ans;

}

按单位向量（x,y,z）逆向角度θ;运算矩阵矩阵

X^2+(1-X^2)cosθ XY(1-cosθ)-Zsinθ XZ(1-cosθ)+Ysinθ

YX(1-cosθ)+Zsinθ Y^2+(1-Y^2)cosθ YZ(1-cosθ)-Xsinθ

ZX(1-cosθ)-Ysinθ ZY(1-cosθ)+Xsinθ Z^2+(1-Z^2)cosθ

# String

### Ac\_自动机

/\*

statci AC\_Automation M point

must build before point

hdu 2222

\*/

#include<cstdio>

#include<cstring>

#define M 60000

#defien kind 26

using namespace std;

struct AC{

int root,en,my\_que[M];

struct E{

int flag,fail;

int next[kind];

}e[M];

void flash(int id){

e[id].flag=0;

memset(e[id].next,-1,sizeof(e[id].next));

}

void init(){

root=0;en=1;

flash(root);

e[root].fail=-1;

}

void insert(char \*p){

int tr=root;

for(int i=0;p[i];i++){

int dex=p[i]-'a';

if(e[tr].next[dex]==-1){

int t2=en++;flash(t2);

e[tr].next[dex]=t2;

}

tr=e[tr].next[dex];

}

e[tr].flag++;

}

void build(){

int hp=0,tp=-1;

my\_que[++tp]=root;

while(tp>=hp){

int now=my\_que[hp++];

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

if(e[now].next[i]!=-1){

int nxt=e[now].next[i];

int dex=i;

if(now==root){

e[nxt].fail=root;

}

else{

int tmp=e[now].fail;

while(tmp!=-1&&e[tmp].next[dex]==-1) tmp=e[tmp].fail;

if(tmp==-1) tmp=root;

else tmp=e[tmp].next[dex];

e[nxt].fail=tmp;

}

my\_que[++tp]=nxt;

}

}

}

}

int query(char \*p){

int ans=0;

int tr=root;

for(int i=0;p[i];i++){

int dex=p[i]-'a';

while(e[tr].next[dex]==-1&&tr!=root)tr=e[tr].fail;

tr=e[tr].next[dex];

if(tr==-1)tr=root;

int tmp=tr;

while(tmp!=root&&e[tmp].flag!=-1){

ans+=e[tmp].flag;

e[tmp].flag=-1;

tmp=e[tmp].fail;

}

}

return ans;

}

}ac;

char str[M\*2];

int main(){

int cas;while(~scanf("%d",&cas)){

while(cas--){

ac.init();

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

char tmp[300];

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

scanf("%s",tmp);

ac.insert(tmp);

}

ac.build();

scanf("%s",str);

int ans=ac.query(str);

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

}

}

return 0;

}

### string\_hash

const int MAXN = 1000003;

//常用 1

1 SDBMHash

unsigned int SDBMHash(char \*str)

{

unsigned int hash = 0;

while (\*str)

{

// equivalent to: hash = 65599\*hash + (\*str++);

hash = (\*str++) + (hash << 6) + (hash << 16) - hash;

}

return (hash & 0x7FFFFFFF)%MAXN;

}

//常用2

6 BKDR Hash Function

unsigned int BKDRHash(char \*str)

{

unsigned int seed = 131; // 31 131 1313 13131 131313 etc..

unsigned int hash = 0;

while (\*str)

{

hash = hash \* seed + (\*str++);

}

return (hash & 0x7FFFFFFF)%MAXN;

}

// a idea is large ll to cover sort ans unqiue

2 RS Hash Function

unsigned int RSHash(char \*str)

{

unsigned int b = 378551;

unsigned int a = 63689;

unsigned int hash = 0;

while (\*str)

{

hash = hash \* a + (\*str++);

a \*= b;

}

return (hash & 0x7FFFFFFF)%MAXN;

}

3 JS Hash Function

unsigned int JSHash(char \*str)

{

unsigned int hash = 1315423911;

while (\*str)

{

hash ^= ((hash << 5) + (\*str++) + (hash >> 2));

}

return (hash & 0x7FFFFFFF)%MAXN;

}

4 P. J. Weinberger Hash Function

unsigned int PJWHash(char \*str)

{

unsigned int BitsInUnignedInt = (unsigned int)(sizeof(unsigned int) \* 8);

unsigned int ThreeQuarters = (unsigned int)((BitsInUnignedInt \* 3) / 4);

unsigned int OneEighth = (unsigned int)(BitsInUnignedInt / 8);

unsigned int HighBits = (unsigned int)(0xFFFFFFFF) << (BitsInUnignedInt - OneEighth);

unsigned int hash = 0;

unsigned int test = 0;

while (\*str)

{

hash = (hash << OneEighth) + (\*str++);

if ((test = hash & HighBits) != 0)

{

hash = ((hash ^ (test >> ThreeQuarters)) & (~HighBits));

}

}

return (hash & 0x7FFFFFFF)%MAXN;

}

5 ELF Hash Function

unsigned int ELFHash(char \*str)

{

unsigned int hash = 0;

unsigned int x = 0;

while (\*str)

{

hash = (hash << 4) + (\*str++);

if ((x = hash & 0xF0000000L) != 0)

{

hash ^= (x >> 24);

hash &= ~x;

}

}

return (hash & 0x7FFFFFFF)%MAXN;

}

7 DJB Hash Function

unsigned int DJBHash(char \*str)

{

unsigned int hash = 5381;

while (\*str)

{

hash += (hash << 5) + (\*str++);

}

return (hash & 0x7FFFFFFF)%MAXN;

}

8 AP Hash Function

unsigned int APHash(char \*str)

{

unsigned int hash = 0;

int i;

for (i=0; \*str; i++)

{

if ((i & 1) == 0)

{

hash ^= ((hash << 7) ^ (\*str++) ^ (hash >> 3));

}

else

{

hash ^= (~((hash << 11) ^ (\*str++) ^ (hash >> 5)));

}

}

return (hash & 0x7FFFFFFF)%MAXN;

}

# Other

### Mergesort

#define ll long long

template<class T>

class Merge {

public:

T \*a,\*b;

int n;

ll num;

ll sort(T \*a1,T \*b1,int n1){

a=a1;b=b1;n=n1;

num=0;

divide(0,n-1);

return num;

}

void divide(int l,int r){

if(r>l){

int mid=(l+r)>>1;

divide(l,mid);

divide(mid+1,r);

merge(l,r);

}

}

void merge(int l,int r){

int mid=(l+r)>>1;

int lpoint=l,rpoint=mid+1;

int point=l;

while(lpoint<=mid&&rpoint<=r){

if(a[rpoint]<a[lpoint]){

b[point++]=a[rpoint++];

num+=(mid-lpoint+1);

}

else{

b[point++]=a[lpoint++];

}

}

while(lpoint<=mid){

b[point++]=a[lpoint++];

}

while(rpoint<=r){

b[point++]=a[rpoint++];

}

for(int i=l;i<=r;i++){

a[i]=b[i];

}

}

};

### 输入外挂

int get\_int()

{

int res = 0, ch, flag = 0;

if((ch = getchar()) == '-')

flag = 1;

else if(ch >= '0' && ch <= '9')

res = ch - '0';

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

res = (res<<1)+(res<<3)+ ch - '0';

return flag ? -res : res;

}