ACM模板

目录

[一 数学、数论 3](#_Toc464286396)

[1. 扩展欧几里得 求逆元 3](#_Toc464286397)

[2. 中国剩余定理 3](#_Toc464286398)

[3. FFT 4](#_Toc464286399)

[4. NTT+cdq分治 6](#_Toc464286400)

[5. 筛法求素数 10](#_Toc464286401)

[6. 原根 10](#_Toc464286402)

[7. 高斯消元 11](#_Toc464286403)

[实数版 11](#_Toc464286404)

[整数 12](#_Toc464286405)

[二 树 16](#_Toc464286406)

[1. splay 16](#_Toc464286407)

[2. treap 20](#_Toc464286408)

[3. 主席树 23](#_Toc464286409)

[4. 树链剖分 25](#_Toc464286410)

[5. 树分治 28](#_Toc464286411)

[三 图 32](#_Toc464286412)

[1. 二分图 32](#_Toc464286413)

[匈牙利 32](#_Toc464286414)

[KM 32](#_Toc464286415)

[2. 欧拉回路 34](#_Toc464286416)

[3. 最大团 34](#_Toc464286417)

[4. Tarjan 35](#_Toc464286418)

[/\* Tarjan for bridge \*/ 35](#_Toc464286419)

[/\* Tarjan for LCA \*/ 37](#_Toc464286420)

[/\* Tarjan for Strongly-Connected Components \*/ 40](#_Toc464286421)

[5. 网络流 42](#_Toc464286422)

[dinic 42](#_Toc464286423)

[费用流 43](#_Toc464286424)

[6. 带花树算法(一般图匹配) 45](#_Toc464286425)

[四 字符串 49](#_Toc464286426)

[1. 后缀数组 49](#_Toc464286427)

[倍增(ologn) 49](#_Toc464286428)

[D3(O(n)) 50](#_Toc464286429)

[2. Manacher 51](#_Toc464286430)

[3. 后缀自动机 53](#_Toc464286431)

[4. AC自动机 55](#_Toc464286432)

[五 奇技淫巧 57](#_Toc464286433)

[1. vim配置文件 57](#_Toc464286434)

[2. pbds用法 57](#_Toc464286435)

[3. rope用法 57](#_Toc464286436)

[4. string用法 57](#_Toc464286437)

[5. 正则表达式(c++11) 57](#_Toc464286438)

# 数学、数论

## 扩展欧几里得 求逆元

int exgcd(int a,int b,int &x,int &y)  
{  
    if(b==0)  
    {  
        x=1;  
        y=0;  
        return a;  
    }  
    int r=exgcd(b,a%b,x,y);  
    int t=x;  
    x=y;  
    y=t-a/b\*y;  
    return r;  
}

## 中国剩余定理

#include <iostream>

using namespace std;

int Extended\_Euclid(int a,int b,int &x,int &y) //扩展欧几里得算法

{

int d;

if(b==0)

{

x=1;y=0;

return a;

}

d=Extended\_Euclid(b,a%b,y,x);

y-=a/b\*x;

return d;

}

int Chinese\_Remainder(int a[],int w[],int len) //中国剩余定理 a[]存放余数 w[]存放两两互质的数

{

int i,d,x,y,m,n,ret;

ret=0;

n=1;

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

n\*=w[i];

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

{

m=n/w[i];

d=Extended\_Euclid(w[i],m,x,y);

ret=(ret+y\*m\*a[i])%n;

}

return (n+ret%n)%n;

}

int main()

{

int n,i;

int w[15],b[15];

while (scanf("%d",&n),n)

{

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

{

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

}

printf("%d/n",Chinese\_Remainder(b,w,n));

}

return 0;

}

## FFT

#define N 1000000

#define int64 long long

#define Del(x,y) memset(x,y,sizeof(x))

const double PI = acos( -1.0 );

struct complex{

double r,i;

complex() { r=i=0 ;}

complex( double \_r, double \_i ) {

r = \_r; i = \_i;

}

inline complex operator+(const complex &c){

return complex( r+c.r , i+c.i );

}

inline complex operator-(const complex &c){

return complex( r-c.r , i-c.i );

}

inline complex operator\*(const complex &c){

return complex( r\*c.r-i\*c.i , r\*c.i+i\*c.r );

}

};

complex a[N],b[N];

int64 ret[N];

char s[N],t[N];

int n,m,k;

void reverse(complex \*y, int L){

int i,j,k; complex t;

for(i=1,j=L>>1; i<L-1; ++i){

if (i < j) {

t = y[i]; y[i] = y[j]; y[j] = t;

}

k = L>>1;

while (j>=k) {

j -= k;

k >>= 1;

}

j += k;

}

}

void FFT(complex \*y, int L, int on){

int i,j,k;

reverse(y,L);

for (i=2; i <=L ; i<<=1) {

complex wn = complex( cos(on\*2\*PI/i), sin(on\*2\*PI/i) );

for( j=0; j<L; j += i){

complex w = complex(1,0);

for(k=j;k<j+i/2;++k){

complex u = y[k];

complex t = w\*y[k+i/2];

y[k] = u + t;

y[k+i/2] = u - t;

w = w \* wn;

}

}

}

if (on == 1)

for(i=0;i<L;i++) y[i].r /= L;

}

void calc(complex \*a, complex \*b, int64 \*ret, int l1, int l2){

int L = 1 , SL = l1+l2-1;

while( L<SL ) L<<=1;

int i;

for(i=l1;i<=L;i++) a[i] = complex(0.0,0.0);

for(i=l2;i<=L;i++) b[i] = complex(0.0,0.0);

FFT(a,L,-1); FFT(b,L,-1);

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

a[i] = a[i]\*b[i];

}

FFT(a,L,1);

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

ret[i] += (int64)floor( a[i].r+0.1);

}

}

## NTT+cdq分治

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;

const int P = 1004535809;

const int N = 6e5, M = 524288;

int pow\_mod(int a,ll b=P-2,int mod=P){

ll x = a, ans = 1;

if (a==1) return 1;

do{

if (b&1) ans=ans\*x%mod;

x=x\*x%mod;

}while(b>>=1);

return ans;

}

int find\_primitive\_number(int p){

int fac[20],m=0;

int q=p-1;

for(int i=2;i<=q;i++)

if (q%i==0){

fac[m++] = i;

while(q%i==0) q/=i;

}

q = p-1;

for(int g=2;;g++){

int flg = 1;

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

if (pow\_mod(g,q/fac[i],p)==1){

flg = 0; break;

}

}

if (flg) return g;

}

return 1;

}

struct F{

int x;

F(int \_x=0):x(\_x){}

F operator\*(const F& tt)const{

return F( (int) ( (ll)x \* tt.x % P ) );

}

F operator+(const F& tt)const{

F ans(x+tt.x);

if (ans.x>=P)

ans.x -= P;

return ans;

}

F operator-(const F& tt)const{

F ans(x-tt.x);

if (ans.x<0)

ans.x += P;

return ans;

}

F ni(){

return F(pow\_mod(x, P-2));

}

};

const int GP = find\_primitive\_number(P);

void ntt(F y[],int len,int on){

for(int i = 1, j = len >> 1; i < len -1; i++){

if (i < j) swap(y[i], y[j]);

int k = len >> 1;

while(j >= k){

j -= k;

k >>= 1;

}

j += k;

}

int gp = pow\_mod(GP, (P-1) / len);

for(int i=2; i <= len; i <<= 1){

F gn = F(pow\_mod(gp, len / i));

if (on == -1) gn = gn.ni();

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

F g = F(1);

for(int k = j; k < j+ (i>>1); k++){

F u = y[k];

F t = g \* y[ k + (i>>1) ];

y[k] = u + t;

y[k + (i>>1)] = u - t;

g = g \* gn;

}

}

}

if (on == -1){

F ni( pow\_mod(len, P-2) );

for(int i = 0; i < len; i++) y[i] = y[i]\*ni;

}

}

int a[N], b[N], n;

int f[N], h[N];

ll rev[N],fac[N];

void init(){

fac[0] = 1;

rev[0] = 1;

int n = M;

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

h[i] = pow\_mod(i,i-2)\*(ll)pow\_mod(fac[i-1]) % P;

fac[i] = fac[i-1]\*(ll)i % P;

rev[i] = pow\_mod(i);

}

}

void solve(int l,int r){

if (l==r){

f[l] = (ll)f[l]\*rev[l] % P;

return;

}

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

solve(l, mid);

int len = 1;

while(len <= r-l) len <<= 1;

for(int i=0;i<len;i++)

a[i] = b[i] = 0;

for(int i=l;i<=mid;i++)

a[i-l] = f[i];

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

b[i-1] = h[i];

ntt((F\*)a, len, 1);

ntt((F\*)b, len, 1);

for(int i=0;i<len;i++)

a[i] = (ll)a[i] \* b[i] % P;

ntt((F\*)a, len, -1);

for(int i=mid+1;i<=r;i++)

f[i] = (f[i] + a[i-l-1]) % P;

solve(mid+1, r);

}

int main(){

init();

int T;

scanf("%d",&T);

while(T--){

scanf("%d",&n);

memset(f,0,sizeof(f));

f[0] = 1;

solve(0,n);

ll ans = f[n]\*fac[n]%P;

ans = pow\_mod(2,(ll)n\*(n-1)/2) - ans + P;

ans %= P;

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

}

return 0;

}

## 筛法求素数

int prime[N],primes,phi[N],mob[N];

bool com[N];

void calc(int n){

int i,j;

primes = 0;

mob[1] = 1;

phi[1] = 1;

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

if (!com[i]){

prime[primes++] = i;

phi[i] = i-1;

mob[i] = -1;

}

for(j=0;j<primes&&prime[j]\*i<=n;j++){

com[i\*prime[j]] = true;

if (i%prime[j]){

mob[i\*prime[j]] = -mob[i];

phi[i\*prime[j]] = phi[i]\*(prime[j]-1);

}else{

mob[i\*prime[j]] = 0;

phi[i\*prime[j]] = phi[i]\*prime[j];

break;

}

}

}

}

## 原根

inline int qm(int x,int a,int mod){

LL s=1,b=(LL)x;

while(a){

if (a&1) s = s\*b%mod;

b = b\*b%mod;

a>>=1;

}

return (int)s;

}

int get\_primitive(int p){

if (p==2) return 1;

int tmp = p-1;

l = 0;

int i = 2 ;

while(i<=tmp){

if (tmp%i==0){

a[l++] = i;

while(tmp%i==0&&tmp) tmp/=i;

}

i++;

}

int g=2;

tmp = p-1;

while(1){

for(i=0;i<l;i++) if (qm(g, tmp/a[i] , p )==1) break;

if (i>=l) return g;

g++;

}

}

## 高斯消元

### 实数版

const double eps=1e-8;

int map[N][N];

double a[N][N],exp[M],res[N];//a matrix , res answer

int d[N],n,m;

inline double abs(double x){return x>0?x:-x;}

void Guass(){

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

int s=i;

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

if (abs(a[j][i])>abs(a[s][i])+eps) s=j;

if (s!=i)

for(int j=i;j<=n+1;j++) swap(a[i][j],a[s][j]);

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

for(int k=n+1;k>=i;k--)

a[j][k]=a[i][k]\*a[j][i]/a[i][i]-a[j][k];

}

res[n]=1;

for(int i=n-1;i>=1;i--){

res[i]=a[i][n+1];

for(int j=i+1;j<=n;j++) res[i]-=res[j]\*a[i][j];

res[i]/=a[i][i];

}

}

### 整数

int a[MAXN][MAXN];//增广矩阵

int x[MAXN];//解集

bool free\_x[MAXN];//标记是否是不确定的变元

inline int gcd(int a,int b){int t; while(b!=0) {t=b; b=a%b; a=t; }return a;}

inline int lcm(int a,int b){return a/gcd(a,b)\*b; }//先除后乘防溢出

// 高斯消元法解方程组(Gauss-Jordan elimination).(-2表示有浮点数解，但无整数解，

//-1表示无解，0表示唯一解，大于0表示无穷解，并返回自由变元的个数)

//有equ个方程，var个变元。增广矩阵行数为equ,分别为0到equ-1,列数为var+1,分别为0到var.

int Gauss(int equ,int var)

{

int i,j,k;

int max\_r;// 当前这列绝对值最大的行.

int col;//当前处理的列

int ta,tb;

int LCM;

int temp;

int free\_x\_num;

int free\_index;

for(int i=0;i<=var;i++)

{

x[i]=0;

free\_x[i]=true;

}

//转换为阶梯阵.

col=0; // 当前处理的列

for(k = 0;k < equ && col < var;k++,col++)

{// 枚举当前处理的行.

// 找到该col列元素绝对值最大的那行与第k行交换.(为了在除法时减小误差)

max\_r=k;

for(i=k+1;i<equ;i++)

{

if(abs(a[i][col])>abs(a[max\_r][col])) max\_r=i;

}

if(max\_r!=k)

{// 与第k行交换.

for(j=k;j<var+1;j++) swap(a[k][j],a[max\_r][j]);

}

if(a[k][col]==0)

{// 说明该col列第k行以下全是0了，则处理当前行的下一列.

k--;

continue;

}

for(i=k+1;i<equ;i++)

{// 枚举要删去的行.

if(a[i][col]!=0)

{

LCM = lcm(abs(a[i][col]),abs(a[k][col]));

ta = LCM/abs(a[i][col]);

tb = LCM/abs(a[k][col]);

if(a[i][col]\*a[k][col]<0)tb=-tb;//异号的情况是相加

for(j=col;j<var+1;j++)

{

a[i][j] = a[i][j]\*ta-a[k][j]\*tb;

}

}

}

}

// 1. 无解的情况: 化简的增广阵中存在(0, 0, ..., a)这样的行(a != 0).

for (i = k; i < equ; i++)

{ // 对于无穷解来说，如果要判断哪些是自由变元，那么初等行变换中的交换就会影响，则要记录交换.

if (a[i][col] != 0) return -1;

}

// 2. 无穷解的情况: 在var \* (var + 1)的增广阵中出现(0, 0, ..., 0)这样的行，即说明没有形成严格的上三角阵.

// 且出现的行数即为自由变元的个数.

if (k < var)

{

// 首先，自由变元有var - k个，即不确定的变元至少有var - k个.

for (i = k - 1; i >= 0; i--)

{

// 第i行一定不会是(0, 0, ..., 0)的情况，因为这样的行是在第k行到第equ行.

// 同样，第i行一定不会是(0, 0, ..., a), a != 0的情况，这样的无解的.

free\_x\_num = 0; // 用于判断该行中的不确定的变元的个数，如果超过1个，则无法求解，它们仍然为不确定的变元.

for (j = 0; j < var; j++)

{

if (a[i][j] != 0 && free\_x[j]) free\_x\_num++, free\_index = j;

}

if (free\_x\_num > 1) continue; // 无法求解出确定的变元.

// 说明就只有一个不确定的变元free\_index，那么可以求解出该变元，且该变元是确定的.

temp = a[i][var];

for (j = 0; j < var; j++)

{

if (a[i][j] != 0 && j != free\_index) temp -= a[i][j] \* x[j];

}

x[free\_index] = temp / a[i][free\_index]; // 求出该变元.

free\_x[free\_index] = 0; // 该变元是确定的.

}

return var - k; // 自由变元有var - k个.

}

// 3. 唯一解的情况: 在var \* (var + 1)的增广阵中形成严格的上三角阵.

// 计算出Xn-1, Xn-2 ... X0.

for (i = var - 1; i >= 0; i--)

{

temp = a[i][var];

for (j = i + 1; j < var; j++)

{

if (a[i][j] != 0) temp -= a[i][j] \* x[j];

}

if (temp % a[i][i] != 0) return -2; // 说明有浮点数解，但无整数解.

x[i] = temp / a[i][i];

}

return 0;

}

int main(void)

{

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

freopen("out.txt","w",stdout);

int i, j;

int equ,var;

while (scanf("%d %d", &equ, &var) != EOF)

{

memset(a, 0, sizeof(a));

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

{

for (j = 0; j < var + 1; j++)

{

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

}

}

// Debug();

int free\_num = Gauss(equ,var);

if (free\_num == -1) printf("无解!\n");

else if (free\_num == -2) printf("有浮点数解，无整数解!\n");

else if (free\_num > 0)

{

printf("无穷多解! 自由变元个数为%d\n", free\_num);

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

{

if (free\_x[i]) printf("x%d 是不确定的\n", i + 1);

else printf("x%d: %d\n", i + 1, x[i]);

}

}

else

{

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

{

printf("x%d: %d\n", i + 1, x[i]);

}

}

printf("\n");

}

return 0;

}

# 树

## splay

#include <stdio.h>

#include <string.h>

#include <algorithm>

using namespace std;

const int N = 200100;

const int inf = ~0u>>2;

struct node{

int fa,ch[2];

int val,add,min,size,rev;

node(){}

node(int \_val,int \_fa){

val = min = \_val ; fa = \_fa;

ch[0] = ch[1] = add = rev = 0;

size = 1;

}

}t[N];

int n,m,root,tot;

inline void pushdown(int x){

if (!x) return;

if (t[x].rev){

swap(t[x].ch[0],t[x].ch[1]);

if (t[x].ch[0]) t[t[x].ch[0]].rev ^= 1;

if (t[x].ch[1]) t[t[x].ch[1]].rev ^= 1;

t[x].rev = 0;

}

if (t[x].add){

for(int i=0;i<2;i++)

if (t[x].ch[i]) {

t[ t[x].ch[i] ].add += t[x].add;

t[ t[x].ch[i] ].min += t[x].add;

}

t[x].val += t[x].add;

t[x].add = 0;

}

}

inline void pushup(int x){

if (!x) return;

t[x].min = t[x].val;

t[x].size = 1;

for(int i=0;i<2;i++)

if (t[x].ch[i]){

t[x].size += t[ t[x].ch[i] ].size;

t[x].min = min( t[x].min , t[ t[x].ch[i] ].min );

}

t[x].min += t[x].add;

}

inline void rotate(int p){

if (!p) return;

int q = t[p].fa , y = t[q].fa , x = t[q].ch[1] == p;

pushdown(q); pushdown(p);

t[q].ch[x] = t[p].ch[x^1]; t[t[q].ch[x]].fa = q;

t[p].ch[x^1] = q; t[q].fa = p;

t[p].fa = y;

if (y){

if (t[y].ch[0]==q) t[y].ch[0] = p;

else if (t[y].ch[1]==q) t[y].ch[1] = p;

}

pushup(q); pushup(p);

}

inline void splay(int x,int kk=0){

for(int y; (y = t[x].fa)!=kk ; rotate(x))

if (t[y].fa!=kk)

rotate( (x == t[y].ch[0] ) == ( y == t[t[y].fa].ch[0] ) ?y:x );

if (!kk) root = x;

}

inline int find(int x,int k){

pushdown(x);

if (t[ t[x].ch[0] ].size+1 == k) return x;

if (t[ t[x].ch[0] ].size>=k) return find(t[x].ch[0],k);

return find(t[x].ch[1],k-t[ t[x].ch[0] ].size-1);

}

inline void insert(int x,int v){

int k=find(root,x);

t[++tot] = node(v,k);

splay(k);

pushdown(k);

t[tot].ch[1] = t[k].ch[1];

t[ t[k].ch[1] ].fa = tot;

t[k].ch[1] = tot;

pushup(tot); pushup(k);

}

inline void makerange(int l,int r){

splay(find(root,r+2));

splay(find(root,l),root);

}

inline void addr(int l,int r,int a){

makerange(l,r);

t[ t[ t[root].ch[0] ].ch[1] ].add += a;

t[ t[ t[root].ch[0] ].ch[1] ].min += a;

}

inline void del(int x){

makerange(x,x);

t[ t[root].ch[0] ].ch[1] = 0;

t[ t[ t[root].ch[0] ].ch[1] ].size = 0;

t[ t[ t[root].ch[0] ].ch[1] ].fa = 0;

pushup(t[root].ch[0]);

pushup(root);

}

inline void reverse(int l,int r){

makerange(l,r);

t[ t[ t[root].ch[0] ].ch[1] ].rev ^= 1;

}

inline void revolve(int l,int r,int k){

int len = r-l+1;

k = ((k)%len+len)%len;

if (k==0) return;

reverse(l,r-k);

reverse(r-k+1,r);

reverse(l,r);

}

inline int querymin(int l,int r){

makerange(l,r);

return t[ t[ t[root].ch[0] ].ch[1] ].min;

}

int main(){

tot = root = 1;

t[1] = node(inf,0);

insert(1,inf);

int i,x;

scanf("%d",&n);

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

scanf("%d",&x);

insert(i,x);

splay(rand()%tot+1);

}

char op[10];int y,z;

scanf("%d",&m);

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

scanf("%s",op);

switch(op[0]){

case 'A':{

scanf("%d%d%d",&x,&y,&z);

addr(x,y,z);

break;

}

case 'I':{

scanf("%d%d",&x,&y);

insert(x+1,y);

break;

}

case 'D':{

scanf("%d",&x);

del(x);

break;

}

case 'M':{

scanf("%d%d",&x,&y);

printf("%d\n",querymin(x,y));

break;

}

case 'R':{

if (op[3]=='E'){

scanf("%d%d",&x,&y);

reverse(x,y);

}else{

scanf("%d%d%d",&x,&y,&z);

revolve(x,y,z);

}

break;

}

}

}

return 0;

}

## treap

#include<iostream>

#include<stdio.h>

#include<string.h>

#include<cmath>

#include<queue>

#include<stack>

#include<set>

#include<time.h>

#include<map>

#include<algorithm>

#define ll long long

#define eps 1e-5

#define oo 1000000007

#define pi acos(-1.0)

#define MAXN 200005

using namespace std;

// insert(x) : 插入x，如果已经存在返回false，否则返回true

// count(x) : <=x的元素共有多少个

// k\_th(x) : 第k个是几，下标1起，x<=0||x>n均返回-1。

// del(x) : 删除x，成功删除返回true，否则false。

struct node {

int l,r,key,fix,size;

};

struct treap {

node h[MAXN];

int root,num;

void init() { srand(time(0)); num=root=0; }

void rot\_l(int &x) {

int R=h[x].r,L=h[x].l;

h[x].size=h[x].size-h[R].size+h[h[R].l].size;

h[R].size+=h[L].size+1;

h[x].r=h[R].l,h[R].l=x;

x=R;

}

void rot\_r(int &x) {

int L=h[x].l,R=h[x].r;

h[x].size=h[x].size-h[L].size+h[h[L].r].size;

h[L].size+=h[R].size+1;

h[x].l=h[L].r,h[L].r=x;

x=L;

}

bool insert(int key) { return insert(root,key); }

bool insert(int &k,int key) {

if (!k) {

k=++num;

h[k].l=h[k].r=0,h[k].size=1;

h[k].key=key,h[k].fix=rand();

return true;

}

if (h[k].key==key) return false;

if (h[k].key>key) {

if (!insert(h[k].l,key)) return false;

h[k].size++;

if (h[h[k].l].fix>h[k].fix) rot\_r(k);

return true;

}

else {

if (!insert(h[k].r,key)) return false;

h[k].size++;

if (h[h[k].r].fix>h[k].fix) rot\_l(k);

return true;

}

}

int count(int key) {

int g=0,k=root;

while (k) {

if (h[k].key>key) k=h[k].l;

else g+=h[h[k].l].size+1,k=h[k].r;

}

return g;

}

int k\_th(int kth) {

int g=0,k=root;

if (h[root].size<kth || kth<=0) return -1;

while (h[h[k].l].size+g+1!=kth)

{

if (h[h[k].l].size+g+1>=kth) k=h[k].l;

else g+=h[h[k].l].size+1,k=h[k].r;

}

return h[k].key;

}

bool del(int key) { return del(root,key); }

bool del(int &k,int key) {

if (!k) return false;

if (h[k].key>key) {

if (!del(h[k].l,key)) return false;

h[k].size--;

}

else if (h[k].key<key) {

if (!del(h[k].r,key)) return false;

h[k].size--;

}

else {

if (!h[k].l && !h[k].r) k=0;

else if (!h[k].l) k=h[k].r;

else if (!h[k].r) k=h[k].l;

else if (h[h[k].l].fix<h[h[k].r].fix) {

rot\_l(k);

if (!del(h[k].l,key)) return false;

h[k].size--;

}

else {

rot\_r(k);

if (!del(h[k].r,key)) return false;

h[k].size--;

}

}

return true;

}

} mytreap;

int main()

{

int m,x;

char c;

mytreap.initial();

scanf("%d",&m);

while (m--)

{

do { c=getchar(); } while (c<'A' || c>'Z');

scanf("%d",&x);

if (c=='I') mytreap.insert(mytreap.root,x); else

if (c=='C') printf("%d\n",mytreap.count(x-1)); else

if (c=='K')

{

x=mytreap.k\_th(x);

if (x==-1) printf("invalid\n");

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

}else

if (c=='D') mytreap.del(mytreap.root,x);

}

return 0;

}

## 主席树

#include <cstdio>

#include <cstring>

#include <algorithm>

using namespace std;

const int N = 100100 , M = 20;

struct node{

int ls,rs,w;

}t[N\*M];

int a[N] , p[N] , b[N] , root[N] ,n, pos;

int cmp(int i,int j){

return a[i]<a[j];

}

void read(){

memset(t,0,sizeof(t));

scanf("%d",&n);

int i;

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

scanf("%d",&a[i]); p[i] = i;

}

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

for(i=1;i<=n;i++) b[p[i]] = i;

for(i=1;i<=n;i++) printf("%d%c",p[i], " \n"[i==n]);

for(i=1;i<=n;i++) printf("%d%c",b[i], " \n"[i==n]);

}

void ins(int &k, int l ,int r, int q ){

t[ ++pos ] = t[k] ; k = pos;

t[k].w ++;

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

if (l==r) return ;

if (q<=mid) ins( t[k].ls ,l, mid, q);

else ins( t[k].rs, mid+1, r , q );

}

int query(int x,int y, int l, int r, int k){

if (l==r) return l;

int tt = t[ t[y].ls ].w - t[ t[x].ls ].w ;

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

if (tt>=k) return query( t[x].ls , t[y].ls , l, mid , k );

else return query( t[x].rs, t[y].rs , mid+1 , r, k-tt );

}

void work(){

int i;

root[0] = 0; pos = 0;

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

root[i] = root[i-1];

ins(root[i],1,n,b[i]);

}

int m,x,y,k,q;

scanf("%d",&m);

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

scanf("%d%d%d",&x,&y,&k);

q = query( root[x-1] , root[y] , 1, n, k);

printf("%d\n",a[ p[ q ] ]);

}

}

int main(){

read();

work();

return 0;

}

## 树链剖分

#include <stdio.h>

#include <string.h>

#include <algorithm>

#pragma comment(linker, "/STACK:102400000,102400000")

using namespace std;

const int N = 200200;

typedef unsigned long long ull;

const ull maxh = 1LL<<62;

struct num{

ull a;

num(){a=1;}

num(ull \_a){a=\_a;}

inline num operator\*(const num tt)const{

num ans;

if (a>=maxh || tt.a>=maxh) ans.a=maxh+1;

else{

if (maxh/a<=tt.a) ans.a=maxh+1;

else {

ans.a=a\*tt.a;

if (ans.a>=maxh) ans.a=maxh+1;

}

}

return ans;

}

}Big(maxh),one(1);

struct edge{int s,e;ull x;}ori[N];

struct edges{int e,next;}e[N<<1];

int head[N],cnt,n,m;

int size[N],son[N],fa[N],dep[N];

int pos,p[N],top[N];

struct SegMent{

num t[N<<2];

void updata(int k,int l,int r,int x,ull a){

if (l==r) t[k]=num(a);

else{

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

if (x<=m) updata( k<<1 , l , m , x , a );

else updata( k<<1|1 , m+1,r , x , a );

t[k] = t[k<<1]\*t[k<<1|1];

}

}

num query(int k,int l,int r,int ll,int rr){

if (ll>rr || ll>r || rr<l) return one;

if (ll<=l && r<=rr) return t[k];

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

if (rr<=m) return query(k<<1,l,m,ll,rr);

if (ll>m) return query(k<<1|1,m+1,r,ll,rr);

return query(k<<1,l,m,ll,rr)\*query(k<<1|1,m+1,r,ll,rr);

}

inline num query(int l,int r){

return query(1,0,pos,l,r);

}

}Seg;

inline void add(int s,int t){

e[cnt]=(edges){t,head[s]};head[s]=cnt++;

e[cnt]=(edges){s,head[t]};head[t]=cnt++;

}

void getsize(int x=1,int pre=0,int depth=0){

size[x]=1;

fa[x]=pre;

dep[x]=depth;

for(int i=head[x];~i;i=e[i].next)

if (e[i].e!=pre){

getsize(e[i].e,x,depth+1);

size[x]+=size[e[i].e];

if (son[x]==-1 || size[son[x]]<size[e[i].e])

son[x] = e[i].e;

}

}

void getorder(int x=1,int st=1){

p[x]=pos++;

top[x] = st;

if (son[x]!=-1){

getorder(son[x],st);

}else return;

for(int i=head[x];~i;i=e[i].next)

if (e[i].e!=fa[x] && e[i].e!=son[x]){

getorder(e[i].e,e[i].e);

}

}

void read(){

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

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

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

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

scanf("%d%d%I64u",&ori[i].s,&ori[i].e,&ori[i].x);

add(ori[i].s,ori[i].e);

}

}

void pre(){

getsize();

pos=0;

getorder();

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

if (dep[ori[i].s]<dep[ori[i].e]) swap(ori[i].s,ori[i].e);

Seg.updata(1,0,pos,p[ori[i].s],ori[i].x);

}

}

num query(int u,int v){

int t1=top[u],t2=top[v];

num ans = one;

while(t1!=t2){

if (dep[t1]<dep[t2]) {

swap(t1,t2); swap(u,v);

}

ans = ans\*Seg.query(p[t1],p[u]);

u=fa[t1]; t1=top[u];

if (ans.a>=maxh) return ans;

}

if (u==v) return ans;

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

return ans\*Seg.query(p[u]+1,p[v]);

}

void work(){

int x,y,op; ull c;

num ans;

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

scanf("%d",&op);

if (op==1){

scanf("%d%d%I64u",&x,&y,&c);

ans = query(x,y);

printf("%I64u\n",c/ans.a);

}else{

scanf("%d%I64u",&x,&c);

Seg.updata(1,0,pos,p[ori[x].s],c);

}

}

}

int main(){

read();

pre();

work();

return 0;

}

## 树分治

#include <cstdio>

#include <cstring>

#include <algorithm>

using namespace std;

#define N 10010

#define max(a,b) ((a)>(b)?(a):(b))

#define Del(x,y) memset(x,y,sizeof(x))

#define INF 0x3f3f3f3f

struct edge{

int t,cost,next;

};

edge e[N<<1];

int head[N],size[N],vis[N],f[N],list[N],d[N],pos,n,k,cnt,ans;

void add(int s,int t, int cost){

e[++cnt] = (edge){t,cost,head[s]}; head[s] = cnt;

e[++cnt] = (edge){s,cost,head[t]}; head[t] = cnt;

}

void read(){

Del(head,-1);

Del(size,0);

Del(vis,0);

cnt = -1;

ans = 0;

int i,a,b,c;

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

scanf("%d%d%d",&a,&b,&c);

add(a,b,c);

}

}

int dfs1(int x,int fa){

size[x] = 1;

list [pos++] = x;

f[x] = fa;

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

if (!vis[e[i].t] && e[i].t!=fa){

size[x] += dfs1(e[i].t,x);

}

return size[x];

}

int getroot(int x,int fa){

int tot,max1,max2,i,j,v;

pos = 0;

tot = dfs1(x,fa);

max1 = INF;

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

v = list[i];

max2 = tot-size[ v ];

for(j=head[v];j!=-1;j=e[j].next)

if (e[j].t != f[v]) {

max2 = max(max2,size[e[j].t]);

}

if (max2<max1){

max1 = max2; x = v;

}

}

return x;

}

void dfs2(int x, int cost,int fa){

d[x] = cost;

list[pos++] = x;

int i;

for(i=head[x];i!=-1;i=e[i].next)

if (!vis[e[i].t] && e[i].t!=fa){

dfs2(e[i].t, cost+e[i].cost,x);

}

}

inline int cmp(int a,int b){

return (d[a]<d[b]);

}

int getans(int l,int r){

sort( list+l , list+r , cmp );

int i = l , j = r-1 , res = 0 ;

while(i<j){

while(d[list[i]]+d[list[j]]>k&&i<j) j--;

res += j-i;

i++;

}

return res;

}

void work(int x, int fa){

int root,i,ll;

root = getroot(x,fa);

if (size[x]==1) return ;

vis[root] = 1;

pos = ll = 0;

for (i=head[root];i!=-1;i=e[i].next)

if (!vis[e[i].t]){

dfs2(e[i].t,e[i].cost,root);

ans -= getans(ll,pos);

ll = pos;

}

d[root] = 0; list[pos++] = root;

ans += getans(0,pos);

for (i=head[root];i!=-1;i=e[i].next)

if (!vis[e[i].t]) work(e[i].t,root);

}

int main(){

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

while(scanf("%d%d",&n,&k),n|k){

read();

work(1,0);

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

}

return 0;

}

# 图

## 二分图

### 匈牙利

struct Graph{

int map[N][N],n,m;

int v[N],r[N];

void init(int \_n,int \_m){

memset(map,0,sizeof(map));

n=\_n;m=\_m;

}

inline void add(int s,int t){

map[s][t] = 1;

}

int find(int x){

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

if (!v[i]&&map[x][i]){

v[i] = 1;

if (r[i]==0 || find(r[i])){

r[i] = x; return 1;

}

}

return 0;

}

int cal(){

memset(r,0,sizeof(r));

int ans=0;

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

memset(v,0,sizeof(v));

if (find(i)) ans++;

}

return n+m-ans;

}

}G;

### KM

struct Graph{

int map[N][N];

int fx[N],fy[N], Slack[N],vx[N], vy[N],Match[N];

void read(){

memset(map,0,sizeof(map));

int i,j;

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

for(j=1;j<=m;j++)

{

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

map[i][j] = map[i][j]\*K;

}

}

}

int find(int x){

vx[x] = 1;

int i,dt;

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

if ( !vy[i] ){

dt = fx[x] + fy[i] - map[x][i];

if (dt==0){

vy[i] = 1;

if (Match[i]==-1 || find(Match[i])){

Match[i] = x; return 1;

}

}else if (Slack[i]>dt) Slack[i] = dt;

}

return 0;

}

int KM(){

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

int i,j,k,d;

for(i=1;i<=m;i++) fy[i] = 0;

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

fx[i] = -0x3f3f3f3f;

for(j=1;j<=m;j++)

if (fx[i]<map[i][j]) fx[i] = map[i][j];

}

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

memset(Slack,0x3f,sizeof(Slack));

while(1){

memset(vx,0,sizeof(vx));

memset(vy,0,sizeof(vy));

if (find(k)) break;

d = 0x3f3f3f3f;

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

if (!vy[i]&&d>Slack[i]) d = Slack[i];

for(i=1;i<=n;i++) if (vx[i]) fx[i] -= d;

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

if (vy[i]) fy[i] += d;

else Slack[i] -= d;

}

}

}

int Sum = 0;

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

if (Match[i]!=-1) Sum += map[Match[i]][i];

return Sum;

}

} G;

## 欧拉回路

void dfs(int x){

int& i=sta[x];

for(i;i<ed[x].size();i++)

if (st[ ed[x][i] ]){

st[ ed[x][i] ] = st[ ed[x][i]^1 ] = 0;

dfs( a[ ed[x][i] ] );

}

ans.push\_back(x);

}

## 最大团

struct Graph{

bool map[N][N];

int dp[N];

int mx;

int Stack[N][N];

int d[N];

int ans[N];

void init(){

memset(map,0,sizeof(map));

}

void dfs(int n,int num,int step){

if (num==0){

if (step > mx){

mx = step;

for(int i=1;i<=step;i++) ans[i] = d[i];

}

return ;

}

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

int k = Stack[step][i];

if (step+n-k<=mx) return;

if (step+dp[k]<=mx) return;

int cnt = 0;

for(int j=i+1;j<num;j++)

if ( !map[k][Stack[step][j]] ){

Stack[step+1][cnt++] = Stack[step][j];

}

d[step+1] = k;

dfs(n,cnt,step+1);

}

}

void calc(int n){

mx = 0;

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

int sz = 0;

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

if ( !map[i][j] ) Stack[1][sz++] = j;

d[1] = i;

dfs(n,sz,1);

dp[i]=mx;

}

cout << mx << endl;

//for(int i=1;i<=mx;i++) cout << ans[i] << " \n"[i==mx] ;

}

}G;

## Tarjan

### /\* Tarjan for bridge \*/

#include <cstdio>

#include <cstring>

#include <algorithm>

using namespace std;

template <int maxv,int maxe>

class graph {

public:

struct {

int v,w,next,inv,flag;

} edge[maxe];

int head[maxv],top;

int v,e;

void init() {

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

top=0;

}

int addedge(int x,int y,int w=1) {

edge[top].v=y;

edge[top].w=w;

edge[top].next=head[x];

edge[top].flag=0;

return head[x]=top++;

}

int dfn[maxv],low[maxv],t;

void tarjan(int p,int prev=-1) {

dfn[p]=low[p]=t++;

for(int i=head[p];~i;i=edge[i].next) {

int np=edge[i].v;

if(np==prev) continue;

if(dfn[np])

low[p]=min(low[p],dfn[np]);

else {

tarjan(np,p);

low[p]=min(low[p],low[np]);

if(low[np]>dfn[p]) {

edge[i].flag=1;

edge[edge[i].inv].flag=1;

}

}

}

}

void tarjan() {

memset(dfn,0,sizeof(dfn)); t=1;

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

if(!dfn[i]) tarjan(i);

}

};

graph<1200,2400> g1;

int nn=1,degree[1200],scc[1200];

int dfs(int p) {

for(int i=g1.head[p];~i;i=g1.edge[i].next) {

int np=g1.edge[i].v;

if(scc[np]) continue;

if(g1.edge[i].flag) {

scc[np]=++nn;

++degree[nn];

++degree[scc[p]];

}

else scc[np]=scc[p];

dfs(np);

}

}

int main() {

g1.init();

scanf("%d%d",&g1.v,&g1.e);

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

int x,y;

scanf("%d%d",&x,&y);

int xx=g1.addedge(x,y),yy=g1.addedge(y,x);

g1.edge[xx].inv=yy;

g1.edge[yy].inv=xx;

}

g1.tarjan();

scc[1]=1;

dfs(1);

int c=0;

for(int i=1;i<=nn;++i) c+=degree[i]==1;

printf("%d\n",(c+1)/2);

return 0;

}

### /\* Tarjan for LCA \*/

#include <stdio.h>

#include <cstring>

#include <algorithm>

using namespace std;

const int maxv=42000,maxe=2\*maxv;

const int maxq=700;

struct {

int v,w,next;

} edge[maxe];

int head[maxv],top;

int n,m;

int ans[maxq];

int dist[maxv];

int qp[maxv][2];

void init() {

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

top=0;

}

void addedge(int x,int y,int w) {

edge[top].v=y;

edge[top].w=w;

edge[top].next=head[x];

head[x]=top++;

}

struct {

int x,id,next;

} query[maxq];

int qhead[maxv],qtop;

void initq() {

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

qtop=0;

}

void addquery(int x,int y,int id) {

query[qtop].x=y; query[qtop].id=id;

query[qtop].next=qhead[x]; qhead[x]=qtop++;

query[qtop].x=x; query[qtop].id=id;

query[qtop].next=qhead[y]; qhead[y]=qtop++;

}

int fa[maxv];

int find(int x) {

return x!=fa[x]?(fa[x]=find(fa[x])):x;

}

void uni(int x,int y) {

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

}

void tarjan(int p,int pre) {

fa[p]=p;

for(int i=head[p];~i;i=edge[i].next) {

int np=edge[i].v;

if(np==pre) continue;

dist[np]=dist[p]+edge[i].w; // not for tarjan.

tarjan(np,p);

uni(p,np);

}

for(int i=qhead[p];~i;i=query[i].next) {

int np=query[i].x,id=query[i].id;

if(fa[np]&&!ans[id])

ans[id]=find(np);

}

}

int main() {

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

while(T--) {

init(); initq();

memset(fa,0,sizeof(fa));

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

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

int x,y,w;

scanf("%d%d%d",&x,&y,&w);

addedge(x,y,w); addedge(y,x,w);

}

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

int x,y;

scanf("%d%d",&x,&y);

addquery(x,y,i);

ans[i]=0;

qp[i][0]=x; qp[i][1]=y;

}

memset(dist,0,sizeof(dist));

tarjan(1,-1);

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

printf("%d\n",dist[qp[i][0]]+dist[qp[i][1]]-2\*dist[ans[i]]);

}

return 0;

}

### /\* Tarjan for Strongly-Connected Components \*/

#include <stdio.h>

#include <stack>

#include <cstring>

using namespace std;

class graph {

public:

struct {

int v,w,next;

} edge[60000];

int head[12000],top;

int n,m,t;

void init() {

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

top=0;

}

void addedge(int x,int y,int w=1) {

edge[top].v=y;

edge[top].w=w;

edge[top].next=head[x];

head[x]=top++;

}

/\* Tarjan for Strongly-Connected Components. \*/

void tarjan() {

t=0; sccnum=0;

memset(dfn,0,sizeof(dfn));

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

if(!dfn[i]) tarjan(i);

}

int dfn[12000],low[12000],instack[12000];

int scc[12000],sccnum,sccsize[12000];

stack<int> s;

void tarjan(int p) {

dfn[p]=low[p]=++t;

instack[p]=1; s.push(p);

for(int i=head[p];~i;i=edge[i].next) {

int np=edge[i].v;

if(!dfn[np]) {

tarjan(np);

low[p]=min(low[p],low[np]);

}

else if(instack[np])

low[p]=min(low[p],dfn[np]);

}

if(dfn[p]==low[p]) {

int cp; ++sccnum;

sccsize[sccnum]=0;

do {

cp=s.top();

s.pop(); instack[cp]=0;

scc[cp]=sccnum;

++sccsize[sccnum];

} while(cp!=p);

}

}

int degree[12000];

int work() {

memset(degree,0,sizeof(degree));

memset(dfn,0,sizeof(dfn));

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

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

int ans=0;

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

if(!degree[i]) {

if(ans) {

ans=0;

break;

}

else ans=sccsize[i];

}

return ans;

}

/\* DAG Construction \*/

void dfs(int p) {

dfn[p]=1;

for(int i=head[p];~i;i=edge[i].next) {

int np=edge[i].v;

if(scc[np]!=scc[p])

++degree[scc[p]];

if(!dfn[np]) dfs(np);

}

}

} g;

## 网络流

### dinic

/\*

p1698

from net

\*/

#include <cstdio>

#include <cstring>

#include <iostream>

#include <algorithm>

#include <vector>

using namespace std;

#define N 400

#define INF 0x3f3f3f3f

struct Dinic {

int c[N][N], n, s, t, l[N], e[N];

int flow(int maxf = INF) {

int left = maxf;

while (build()) left -= push(s, left);

return maxf - left;

}

int push(int x, int f) {

if (x == t) return f;

int &y = e[x], sum = f;

for (; y<n; y++)

if (c[x][y] > 0 && l[x]+1==l[y]) {

int cnt = push(y, min(sum, c[x][y]));

c[x][y] -= cnt;

c[y][x] += cnt;

sum -= cnt;

if (!sum) return f;

}

return f-sum;

}

bool build() {

int m = 0;

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

l[e[m++]=s] = 0;

for (int i=0; i<m; i++) for (int y=0; y<n; y++)

if (c[e[i]][y] > 0 && l[y]<0) l[e[m++]=y] = l[e[i]] + 1;

memset(e, 0, sizeof(e));

return l[t] >= 0;

}

} net;

int main() {

int T, n, f[8], w, d, mw, ds;

scanf("%d", &T);

while (T--) {

scanf("%d", &n);

memset(net.c, 0, sizeof(net.c));

ds = mw = 0;

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

for (int j=0; j<7; j++) scanf("%d", &f[j]);

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

ds += d;

mw = max(w, mw);

net.c[0][i+1] = d;

for (int j=0; j<w; j++)

for (int k=0; k<7; k++)

if (f[k]) net.c[i+1][8+j\*7+k] = 1;

}

mw = 7\*mw + 7;

for (int j=8; j<=mw; j++) net.c[j][mw+1] = 1;

net.t = mw + 1, net.n = mw + 2;

if (ds == net.flow()) puts("Yes");

else puts("No");

}

return 0;

}

### 费用流

int spfa(){

int x , v, i;

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

Del(visited , false);

Del(dis,0x3f);

Del(pre,0xff);

dis[0] = -2; visited[0] = true;

q.push(0);

while( !q.empty() ){

x = q.front();

if (x != n+1)

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

v = e[ ed[x][i] ].t;

if ( e[ ed[x][i] ].c>0 && dis[v]>dis[ x ] + e[ ed[x][i] ].f ){

dis[ v ] = dis[x] + e[ ed[x][i] ].f;

pre[ v ] = x;

cc[ v ] = ed[x][i];

if (!visited[v]){

visited[v] = true;

q.push( v );

}

}

}

q.pop();

visited[x] = false;

}

if (dis[ n+1 ] < 1e13) return 1; else return 0;

}

int dinic(int s, int low){

if (s == n+1) return low;

visited[ s ] = true;

int i,a;

for(i=0 ; i<ed[s].size(); i++)

if ( e[ ed[s][i] ].c>0 && (!visited[ e[ ed[s][i] ].t ]) &&

dis[ e[ ed[s][i] ].t] == dis[s] + e[ ed[s][i] ].f){

if ( a = dinic( e[ ed[s][i] ].t, min(low,e[ ed[s][i] ].c) ) ){

e[ ed[s][i] ].c -= a;

e[ ed[s][i]^1 ].c += a;

return a;

}

}

visited[ s ] = false;

return 0;

}

void work(){

maxflow = 0; ans = -1;

int a;

while( spfa() ){

Del(visited,false);

while( a = dinic( 0 , sum) ) {

ans = (( ans>dis[ n+1 ] )?ans:dis[n+1] );

maxflow += a;

Del(visited,false);

}

}

}

## 带花树算法(一般图匹配)

#define SET(a,b) memset(a,b,sizeof(a))

const int N = 410;

int n,m;

struct Gragh{

deque<int> Q;

bool g[MAXN][MAXN],inque[MAXN],inblossom[MAXN],inpath[MAXN];

int match[MAXN],pre[MAXN],base[MAXN];

//g map

int findancestor(int u,int v)

{

memset(inpath,false,sizeof(inpath));

while(1)

{

u=base[u];

inpath[u]=true;

if(match[u]==-1)break;

u=pre[match[u]];

}

while(1)

{

v=base[v];

if(inpath[v])return v;

v=pre[match[v]];

}

}

void reset(int u,int anc)

{

while(u!=anc)

{

int v=match[u];

inblossom[base[u]]=1;

inblossom[base[v]]=1;

v=pre[v];

if(base[v]!=anc)pre[v]=match[u];

u=v;

}

}

void contract(int u,int v,int n)

{

int anc=findancestor(u,v);

SET(inblossom,0);

reset(u,anc);reset(v,anc);

if(base[u]!=anc)pre[u]=v;

if(base[v]!=anc)pre[v]=u;

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

if(inblossom[base[i]])

{

base[i]=anc;

if(!inque[i])

{

Q.push\_back(i);

inque[i]=1;

}

}

}

bool bfs(int S,int n)

{

for(int i=0;i<=n;i++)pre[i]=-1,inque[i]=0,base[i]=i;

Q.clear();Q.push\_back(S);inque[S]=1;

while(!Q.empty())

{

int u=Q.front();Q.pop\_front();

for(int v=1;v<=n;v++)

{

if(g[u][v]&&base[v]!=base[u]&&match[u]!=v)

{

if(v==S||(match[v]!=-1&&pre[match[v]]!=-1))contract(u,v,n);

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

{

pre[v]=u;

if(match[v]!=-1)Q.push\_back(match[v]),inque[match[v]]=1;

else

{

u=v;

while(u!=-1)

{

v=pre[u];

int w=match[v];

match[u]=v;

match[v]=u;

u=w;

}

return true;

}

}

}

}

}

return false;

}

int solve(int n)

{

SET(match,-1);

int ans=0;

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

if(match[i]==-1&&bfs(i,n))

ans++;

return ans;

}

void Clear(){

SET(g,-0);

}

}G;

int a[N];

set<int> S;

void read(){

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

int i,j,x;

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

S.clear();

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

scanf("%d",&x); S.insert(x);

}

G.Clear();

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

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

if (S.count(a[i]+a[j])) G.g[i][j] = 1;

}

printf("%d\n",G.solve(n));

}

int main(){

int T;

scanf("%d",&T);

while(T--){

read();

}

return 0;

}

# 字符串

## 后缀数组

### 倍增(ologn)

int cmp(int \*r,int a,int b,int l){

return (r[a]==r[b]) && (r[a+l]==r[b+l]);

}

int wa[N],wb[N],WS[N],wv[N],r[N];

int Rank[N],height[N];

char str[N];

int sa[N];

void DA(int \*r,int \*sa,int n,int m){

int i,j,p,\*x=wa,\*y=wb,\*t;

for(i=0;i<m;i++) WS[i]=0;

for(i=0;i<n;i++) WS[x[i]=r[i]]++;

for(i=1;i<m;i++) WS[i]+=WS[i-1];

for(i=n-1;i>=0;i--) sa[--WS[x[i]]]=i;

for(j=1,p=1;p<n;j\*=2,m=p)

{

for(p=0,i=n-j;i<n;i++) y[p++]=i;

for(i=0;i<n;i++) if(sa[i]>=j) y[p++]=sa[i]-j;

for(i=0;i<n;i++) wv[i]=x[y[i]];

for(i=0;i<m;i++) WS[i]=0;

for(i=0;i<n;i++) WS[wv[i]]++;

for(i=1;i<m;i++) WS[i]+=WS[i-1];

for(i=n-1;i>=0;i--) sa[--WS[wv[i]]]=y[i];

for(t=x,x=y,y=t,p=1,x[sa[0]]=0,i=1;i<n;i++)

x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;

}

}

void calheight(int \*r,int \*sa,int n){

int i,j,k=0;

for(i=1;i<=n;i++) Rank[sa[i]]=i;

for(i=0;i<n; height[Rank[i++]] = k )

for(k?k--:0,j=sa[Rank[i]-1]; r[i+k]==r[j+k]; k++);

}

void work(){

DA(r,sa,n+1,256);

calheight(r,sa,n);

}

### D3(O(n))

#include <stdio.h>

#include<string.h>

#define maxn 1000001

#define F(x) ((x)/3+((x)%3==1?0:tb))

#define G(x) ((x)<tb?(x)\*3+1:((x)-tb)\*3+2)

int wa[maxn],wb[maxn],wv[maxn],ws[maxn];

int c0(int \*r,int a,int b)

{return r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2];}

int c12(int k,int \*r,int a,int b)

{if(k==2) return r[a]<r[b]||r[a]==r[b]&&c12(1,r,a+1,b+1);

else return r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1];}

void sort(int \*r,int \*a,int \*b,int n,int m)

{

int i;

for(i=0;i<n;i++) wv[i]=r[a[i]];

for(i=0;i<m;i++) ws[i]=0;

for(i=0;i<n;i++) ws[wv[i]]++;

for(i=1;i<m;i++) ws[i]+=ws[i-1];

for(i=n-1;i>=0;i--) b[--ws[wv[i]]]=a[i];

return;

}

void dc3(int \*r,int \*sa,int n,int m) // r为待匹配数组 n为总长度 m为字符范围

{

int i,j,\*rn=r+n,\*san=sa+n,ta=0,tb=(n+1)/3,tbc=0,p;

r[n]=r[n+1]=0;

for(i=0;i<n;i++) if(i%3!=0) wa[tbc++]=i;

sort(r+2,wa,wb,tbc,m);

sort(r+1,wb,wa,tbc,m);

sort(r,wa,wb,tbc,m);

for(p=1,rn[F(wb[0])]=0,i=1;i<tbc;i++)

rn[F(wb[i])]=c0(r,wb[i-1],wb[i])?p-1:p++;

if(p<tbc) dc3(rn,san,tbc,p);

else for(i=0;i<tbc;i++) san[rn[i]]=i;

for(i=0;i<tbc;i++) if(san[i]<tb) wb[ta++]=san[i]\*3;

if(n%3==1) wb[ta++]=n-1;

sort(r,wb,wa,ta,m);

for(i=0;i<tbc;i++) wv[wb[i]=G(san[i])]=i;

for(i=0,j=0,p=0;i<ta && j<tbc;p++)

sa[p]=c12(wb[j]%3,r,wa[i],wb[j])?wa[i++]:wb[j++];

for(;i<ta;p++) sa[p]=wa[i++];

for(;j<tbc;p++) sa[p]=wb[j++];

return;

}

int rank[maxn],height[maxn];

void calheight(int \*r,int \*sa,int n) // 求height数组。

{

int i,j,k=0;

for(i=1;i<=n;i++) rank[sa[i]]=i;

for(i=0;i<n;height[rank[i++]]=k)

for(k?k--:0,j=sa[rank[i]-1];r[i+k]==r[j+k];k++);

return;

}

## Manacher

#include <iostream>

#include <string>

#include <cstring>

using namespace std;

void findBMstr(string& str)

{

int \*p = new int[str.size() + 1];

memset(p, 0, sizeof(p));

int mx = 0, id = 0;

for(int i = 1; i <= str.size(); i++)

{

if(mx > i)

{

p[i] = (p[2\*id - i] < (mx - i) ? p[2\*id - i] : (mx - i));

}

else

{

p[i] = 1;

}

while(str[i - p[i]] == str[i + p[i]])

p[i]++;

if(i + p[i] > mx)

{

mx = i + p[i];

id = i;

}

}

int max = 0, ii;

for(int i = 1; i < str.size(); i++)

{

if(p[i] > max)

{

ii = i;

max = p[i];

}

}

max--;

int start = ii - max ;

int end = ii + max;

for(int i = start; i <= end; i++)

{

if(str[i] != '#')

{

cout << str[i];

}

}

cout << endl;

delete p;

}

int main()

{

string str = "12212321";

string str0;

str0 += "$#";

for(int i = 0; i < str.size(); i++)

{

str0 += str[i];

str0 += "#";

}

cout << str0 << endl;

findBMstr(str0);

return 0;

}

## 后缀自动机

#include <bits/stdc++.h>

using namespace std;

const int N = 200000 + 100;

const int C = 27;

typedef long long ll;

const char zz = 'z'+1;

struct State{

State\* go[C];

State\* parent;

int l,len;

State\* extend(int ,int );

};

State statepool[N << 1];

State \*cur, \*last, \*root, \*b[N<<1];

State\* new\_state(int l, int len){

memset(cur->go,0,sizeof(cur->go));

cur->parent = NULL;

cur->l = l;

cur->len = len;

return cur++;

}

State\* State::extend(int c, int len){

State\* p =this;

State\* np=new\_state(p->l+1, len);

while( p && !p->go[c]){

p->go[c] = np;

p = p->parent;

}

if (!p) np->parent = root;

else{

State\* q = p->go[c];

if (q->l == p->l+1) np->parent = q;

else{

State\* nq = new\_state( p->l+1, len);

memcpy( nq->go , q->go , sizeof(q->go));

nq->parent = q->parent;

np->parent = q->parent = nq;

while( p && p->go[c]==q){

p->go[c] = nq;

p = p->parent;

}

}

}

return np;

}

void init(){

cur = statepool;

root = last = new\_state(0, 0);

}

char s[N],t[N];

int pos,n,m;

int cnt[N<<1];

void build(){

init();

int len = 0;

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

len = s[i]==zz ? 0: len+1;

last = last->extend(s[i]-'a', len);

}

}

int calc(char \*s){

State \*p = root;

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

p = p->go[s[i] - 'a'];

if (!p) return 0;

}

return min( p->l , p->len) - p->parent->l;

}

int main(){

int T;

scanf("%d",&T);

for(int cas=1;cas<=T;cas++){

pos = 0;

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

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

scanf("%s",s+pos);

pos += strlen(s+pos);

s[pos++] = zz;

}

build();

printf("Case #%d:\n",cas);

while(m--){

scanf("%s",t);

printf("%d\n",calc(t));

}

}

return 0;

}

## AC自动机

struct Aho\_Corasick{

int next[N][C], fail[N], last[N], length[N];

int cnt, root;

int ans[N];

queue<int> q;

int new\_node(){

cnt++;

for(int i=0;i<C;i++)

next[cnt][i] = -1;

fail[cnt] = last[cnt] = -1;

length[cnt] = 0;

return cnt;

}

void init(){

cnt = -1;

root = new\_node();

}

void push(char \*s){

int m = strlen(s);

int now = root;

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

int c = get(s[i]);

if (next[now][c]==-1)

next[now][c] = new\_node();

now = next[now][c];

}

length[now] = m;

}

void getfail(){

fail[root] = last[root] = root;

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

int &u = next[root][i];

if (u!=-1){

fail[u] = last[u] = root;

q.push(u);

}else

u = root;

}

while(!q.empty()){

int cur = q.front(); q.pop();

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

int &u = next[cur][i];

if (u!=-1){

fail[u] = next[fail[cur]][i];

q.push(u);

last[u] = length[fail[u]] ? fail[u]: last[fail[u]];

}else

u = next[fail[cur]][i];

}

}

}

void solve(char \*s){

int now = root;

for(int i=0;s[i];i++) ans[i] = 0;

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

int c = get(s[i]);

if (c==26) now = now;

else{

now = next[now][c];

int tp = now;

while(tp != root){

if (length[tp]){

ans[i-length[tp]+1] --;

ans[i+1]++;

}

tp = last[tp];

}

}

}

int tans=0;

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

tans += ans[i];

if (tans<0) s[i] = '\*';

}

puts(s);

}

};

# 奇技淫巧

## vim配置文件

syntax on

set nu

set autoindent

set cindent

set tabstop=4

set shiftwidth=4

colo evening

## pbds用法

#include <bits/stdc++.h>

#include <ext/pb\_ds/assoc\_container.hpp>

#include <ext/pb\_ds/tree\_policy.hpp>

using namespace \_\_gnu\_pbds;

using namespace \_\_gnu\_cxx;

using namespace std;

struct node{

int x,id;

node(int x=0,int id=0):x(x),id(id){}

};

struct cmp{

bool operator()(const node &a,const node &b)const{

return a.x<b.x || (a.x == b.x && a.id < b.id);

}

};

// null\_mapped\_type

typedef tree<node, null\_type, cmp, rb\_tree\_tag, tree\_order\_statistics\_node\_update> Set;

typedef tree<node, null\_type, cmp, rb\_tree\_tag, tree\_order\_statistics\_node\_update>::iterator SetIterator;

typedef tree<string, int, greater<string>, rb\_tree\_tag, tree\_order\_statistics\_node\_update> Map;

typedef tree<string, int, greater<string>, rb\_tree\_tag, tree\_order\_statistics\_node\_update>::iterator MapIterator;

void Set\_test(){

Set s;

s.insert(node(1,1));

s.insert(node(2,1));

SetIterator w = s.find\_by\_order(1); // start from 0

cout << w->x << " " << w->id << endl;

cout << s.order\_of\_key(node(3,1)) << endl; // count nodes < K

cout << s.order\_of\_key(node(2,1)) << endl;

}

# define mp make\_pair

void Map\_test(){

Map m;

m.insert(mp("1111",1));

m.insert(mp("2222",1));

MapIterator q = m.find\_by\_order(1);

cout << q->first << " " << q->second << endl;

cout << m.order\_of\_key("12") << endl;

}

int main(){

Set\_test();

Map\_test();

return 0;

}

## rope用法

# include <bits/stdc++.h>

# include <ext/rope>

using namespace std;

using namespace \_\_gnu\_cxx;

int main(){

char s[] = "123456";

rope<char> \*r = new rope<char>(s, 7);

r->replace(3, "7"); // replace

cout << \*r << endl;

r->insert(4,"456465"); // insert

cout << \*r << endl;

r->erase(4,5); // delete st, length

cout << \*r << endl;

rope<char> \*q = new rope<char>(\*r);

cout << \*q << endl;

rope<char> p = \*q + \*r ; // add

cout << p << endl;

return 0;

}

## string用法

## 正则表达式(c++11)

## 热身赛测试内容

1. C++11
2. Pbds
3. Rope
4. 栈大小