ACM/ICPC Template Manaual

QUST

hxk

October 11, 2018

Contents

0 Include

```
//#include <bits/stdc++.h>
#include <algorithm>
#include  <iostream>
#include   <cstring>
#include    <string>
#include    <cstdio>
#include    <vector>
#include     <stack>
#include     <queue>
#include     <cmath>
#include       <set>
#include       <map>
using namespace std;
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define per(i,a,b) for(int i=a;i>=b;i--)
#define clr(a,x) memset(a,x,sizeof(a))
#define pb push_back
#define mp make_pair
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef unsigned long long ull;
typedef long long ll;
typedef vector<int> vi;
typedef pair<int,int> pii;
/************head******************/
int work(){

    return 0;
}
int main(){
#ifdef superkunn
    freopen("input.txt","rt",stdin);
#endif
    work();
    return 0;
}
```

1 Math

1.1 Fast Power

```
typedef long long ll;
void add(ll &a,ll b,ll mod){
    a+=b;
    a%=mod;
}
ll mul_mod(ll a,ll b,ll mod){
    ll res=0;
    while(b){
        if(b&1)add(res,a,mod);
        add(a,a,mod);
        b>>=1;
    }
    return res;
}
/*
ll mul_mod(ll a,ll b,ll mod){
    a%=mod;
    b%=mod;
    ll c=(long double)a*b/mod;
    ll ans=a*b-c*mod;
    if(ans<0)ans+=mod;
    else if(ans>mod)ans-=mod;
    return ans;
}
*/
ll pow_mod(ll a,ll b,ll mod){//a^b
    ll res=1%mod;
    while(b){
        if(b&1)res=mul_mod(res,a,mod);
        a=mul_mod(a,a,mod);
        b>>=1;
    }
    return res;
}
```

1.2 Basic Number Theory

1.2.1 Extended Euclidean

```
typedef long long ll;
//__gcd(a,b);
ll gcd(ll a,ll b){return b==0?a:gcd(b,a%b);}
ll exgcd(ll a,ll b,ll &x,ll &y){
    ll d=a;
    if(b)d=exgcd(b,a%b,y,x),y-=x*(a/b);
    else x=1,y=0;
    return d;
}
```

1.2.2 Multiplicative Inverse Modulo

```
ll inv(ll a,ll m){
    ll x,y;
    ll d=exgcd(a,m,x,y);
    return d==1?(x+m)%m:-1;
}
ll inv(ll a,ll m){
    return pow_mod(a,m-2,m);
}
int p=37;
inv[1]=1;
for(int i=2;i<=40;i++){
    inv[i]=(p-(p/i))*inv[p%i]%p;
}
//fact invfact
int fact[MAXN];
int invfact[MAXN];
ll pow_mod(ll a,ll b){
    ll res=1;
    while(b){
        if(b&1)res=res*a%MOD;
        a=a*a%MOD;
        b>>=1;
    }
    return res;
}
ll fun(ll n,ll m){
    return (1LL*fact[n]*invfact[m])%MOD*invfact[n-m]%MOD;
}
int n=100000;
fact[0]=1;
for(int i=1;i<=n;i++){
    fact[i]=1LL*fact[i-1]*i%MOD;
}
invfact[n]=pow_mod(fact[n],MOD-2);
for(int i=n;i>=1;i--){
    invfact[i-1]=1LL*invfact[i]*i%MOD;
}
```

1.3 Eular phi

1.3.1 Eular

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int MAXN=10000;
int phi[MAXN];
int phi1(int n){
    int res=n;
    for(int i=2;i*i<=n;i++){
        if(n%i==0){
            res=res/i*(i-1);
            for(;n%i==0;n/=i);
        }
    }
    if(n!=1) res=res/n*(n-1);
    return res;
}
void phi2(int n){
    for(int i=0;i<=n;i++) phi[i]=i;
    for(int i=2;i<=n;i++)
        if(phi[i]==i)
```

```
            for(int j=i;j<=n;j+=i) phi[j]=phi[j]/i*(i-1);
}
int main(){
    phi2(100);
    for(int i=1;i<=100;i++)cout<<phi1(i)<<" "<<phi[i]<<endl;
    return 0;
}
```

1.4 Prime

1.4.1 Miller Rabin

```
//using Fast Power
bool Miller_Rabin(ll n, int s){//s is testing frequency . true -> n is prime
    if (n == 2) return 1;
    if (n < 2 || !(n & 1)) return 0;
    int t = 0;
    ll  x, y, u = n - 1;
    while ((u & 1) == 0) t++, u >>= 1;
    for (int i = 0; i < s; i++){
        ll a = rand() % (n - 1) + 1;
        ll x = pow_mod(a, u, n);
        for (int j = 0; j < t; j++){
            ll y = mul_mod(x, x, n);
            if (y == 1 && x != 1 && x != n - 1) return 0;
            x = y;
        }
        if (x != 1) return 0;
    }
    return 1;
}
```

1.4.2 Eratosthenes Sieve

```
#define rep(i,a,b) for(int i=a;i<=b;i++)
const int MAXN=1e5+5;
int prime[MAXN];//1 base
bool is_prime[MAXN];
int sieve(int n){
    int cnt=0;
    rep(i,0,n)is_prime[i]=true;
    is_prime[0]=is_prime[1]=false;
    rep(i,2,n){
        if(is_prime[i]){
            prime[++cnt]=i;
            for(int j=i;j<=n/i;j++)is_prime[i*j]=false;
        }
    }
    return cnt;
}
```

1.4.3 Segment Sieve

```
const int MAXN=1e6+5;
//[a,b)
bool is_prime[MAXN];
bool is_prime_small[MAXN];
```

```
ll prime[MAXN];//1 base
int segment_sieve(ll a,ll b){
    int cnt=0;
    for(int i=0;1LL*i*i<b;i++)is_prime_small[i]=true;
    is_prime_small[0]=is_prime_small[1]=false;
    for(int i=0;i<b-a;i++)is_prime[i]=true;
    if(a==1)is_prime[0]=false;
     for(int i=2;1LL*i*i<b;i++){
        if(is_prime_small[i]){
            for(int j=2*i;1LL*j*j<b;j+=i)is_prime_small[j]=false;//[2,sqrt(b))
            for(ll j=max(2LL,(a+i-1)/i)*i;j<b;j+=i)is_prime[j-a]=false;
        }
    }
    //[a,b) [0,b-a)
    for(ll i=0;i<b-a;i++){
        if(is_prime[i])prime[++cnt]=i+a;
    }
    return cnt;
}
```

1.4.4 primesON

```
const int MAXN=2e5+10;
int v[MAXN],prime[MAXN];
int cnt;
void primes(int n){
    memset(v,0,sizeof(v));
    cnt=0;
    for(int i=2;i<=n;i++){
        if(v[i]==0){
            v[i]=i;
            prime[++cnt]=i;
        }
        for(int j=1;j<=cnt;j++){
            if(prime[j]>v[i]||prime[j]>n/i)break;
            v[i*prime[j]]=prime[j];
        }
    }
}
```

1.4.5 divide

```
// Vijos 1786
const int MAXN=1e5+10;
int cnt;
int num[MAXN];
int p[MAXN];
void divide(int n){
    cnt=0;
    for(int i=2;1LL*i*i<=n;i++){
        if(n%i==0){
            p[++cnt]=i,num[cnt]=0;
        }
        while(n%i==0)n/=i,num[cnt]++;
    }
    if(n>1){
        p[++cnt]=n,num[cnt]=1;
    }
}
int main(){
    int n;
    scanf("%d",&n);
    divide(n);
    printf("%d",p[2]);
    return 0;
}
```

1.4.6 fact

```
int main(){
    int n;
    scanf("%d",&n);
    primes(n);
    for(int i=1;i<=cnt;i++){
        int p=prime[i],c=0;
        for(int j=n;j;j/=p)c+=j/p;
        printf("%d %d\n",p,c);
    }
    return 0;
}
```

1.5 Matrix

```
//hdu 1005
#include <cstdio>
#include <algorithm>
#include <iostream>
using namespace std;
const int MOD = 7;
struct Matrix {
    long long a[2][2];
};
Matrix operator*(const Matrix& lhs, const Matrix& rhs) {
    Matrix ret;
    for (int i = 0; i < 2; ++i) {
        for (int j = 0; j < 2; ++j) {
            ret.a[i][j] = 0;
            for (int k = 0; k < 2; ++k) {
                ret.a[i][j] += lhs.a[i][k] * rhs.a[k][j];
            }
            ret.a[i][j] %= MOD;
        }
    }
    return ret;
}
int main(){
    int a,b,n;
    while(~scanf("%d%d%d",&a,&b,&n)){
        if(a==0&&b==0&&n==0)break;
        Matrix x,y;
        x.a[0][0]=0;
        x.a[0][1]=1;
        x.a[1][0]=b;
        x.a[1][1]=a;
        y.a[0][1]=y.a[1][1]=0;
        y.a[0][0]=y.a[1][0]=1;
        if(n<=2){
            puts("1");
            continue;
        }
        n-=2;
        while(n>0){
            if(n&1)y=x*y;
            x=x*x;
            n>>=1;
        }
        printf("%lld\n",y.a[1][0]%MOD);
    }

    return 0;
}
```

1.5.1 pointchanging

```
#include<bits/stdc++.h>
using namespace std;
const double PI=acos(-1.0);
struct Matrix{
    double a[3][3];
    void init(){
        for(int i=0;i<3;i++){
            for(int j=0;j<3;j++){
                a[i][j]=0;
            }
        }
    }
    void print(){
        for(int i=0;i<3;i++){
            for(int j=0;j<3;j++){
                cout<<a[i][j]<<" ";
            }
            cout<<endl;
        }
        cout<<"-----------"<<endl;
    }
};
Matrix operator*(const Matrix& lhs,const Matrix& rhs){
    Matrix res;
    res.init();
    for(int i=0;i<3;i++){
        for(int j=0;j<3;j++){
            for(int k=0;k<3;k++){
                res.a[i][j]+=lhs.a[i][k]*rhs.a[k][j];
            }
        }
    }
    return res;
}
const int MAXN=1e4+10;
double x[MAXN],y[MAXN];
int main(){

    int n,m;
    scanf("%d%d",&n,&m);
    for(int i=1;i<=n;i++){
        scanf("%lf%lf",&x[i],&y[i]);
    }
    Matrix base;
    base.init();
    base.a[0][0]=base.a[1][1]=base.a[2][2]=1;
    char op[3];
    Matrix now;
    while(m--){
        scanf("%s",op);
        now.init();
        if(op[0]=='X'){
            now.a[0][0]=1;
            now.a[1][1]=-1;
            now.a[2][2]=1;
        }else if(op[0]=='Y'){
            now.a[0][0]=-1;
            now.a[1][1]=1;
            now.a[2][2]=1;
        }else if(op[0]=='M'){
            double p,q;
            scanf("%lf%lf",&p,&q);
            now.a[0][0]=1;
            now.a[1][1]=1;
            now.a[2][2]=1;
            now.a[0][2]=p;
            now.a[1][2]=q;
        }else if(op[0]=='S'){
            double L;
            scanf("%lf",&L);
            now.a[0][0]=L;
            now.a[1][1]=L;
            now.a[2][2]=1;
        }else if(op[0]=='R'){
            double r;
            scanf("%lf",&r);
            r=r/180*PI;
            now.a[0][0]=cos(r);
            now.a[0][1]=-sin(r);
            now.a[1][0]=sin(r);
            now.a[1][1]=cos(r);
            now.a[2][2]=1;
        }
        base=now*base;
    }

    for(int i=1;i<=n;i++){
        Matrix ans;
        ans.init();
        ans.a[0][0]=x[i];
        ans.a[1][0]=y[i];
        ans.a[2][0]=1;
        ans=base*ans;
        printf("%.1f %.1f\n",ans.a[0][0],ans.a[1][0]);
    }
    return 0;
}
```

1.6 Combinatorics

1.6.1 Combination

```
//2^n-C(0,n)...C(k-1,n)=C(k,n)+...+C(n,n)
//2017 EC A
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int MOD=1000000007;
const int MAXN=1e5+10;
ll cnk[MAXN],inv[MAXN];
ll pow_mod(ll a,ll b){
    ll res=1;
    while(b){
        if(b&1)res=res*a%MOD;
        a=a*a%MOD;
        b>>=1;
    }
    return res;
}
int main(){
    int T;
    scanf("%d",&T);
    int kase=0;
    while(T--){
        int n,k;
        scanf("%d%d",&n,&k);
        ll a=pow_mod(2,n);
        int p=MOD;
        inv[1]=1;
        for(int i=2;i<=k;i++){
            inv[i]=1LL*(p-p/i)*inv[p%i]%p;
        }
        cnk[0]=1;
        ll ans=cnk[0];
        for(int i=1;i<k;i++){
            cnk[i]=cnk[i-1]*(n-i+1)%MOD*inv[i]%MOD;
            ans+=cnk[i];
            if(ans>MOD)ans-=MOD;
        }
        ans=(a-ans+MOD)%MOD;
        printf("Case #%d: %I64d\n",++kase,ans);
    }
    return 0;
}
```

1.7 SumRamainder

```
//cf 616 E
const int MOD=1e9+7;
int main(){
    ll n,k,ans;
    scanf("%lld%lld",&k,&n);
    ans=n%MOD*(k%MOD);
    ans%=MOD;
    ll inv2=MOD-MOD/2;
    for(ll x=1,gx;x<=n;x=gx+1){
        gx=k/x?min(k/(k/x),n):n;
        ans-=((k/x)%MOD*((x+gx)%MOD)%MOD*((gx-x+1)%MOD)%MOD*inv2)%MOD;
        if(ans<0)ans+=MOD;
    }
    printf("%lld",ans);
    return 0;
}
```

2 String Processing

2.1 KMP

```
//MAXN
int nxt[MAXN];
void initkmp(char x[],int m){
    int i=0,j=nxt[0]=-1;
    while(i<m){
        while(j!=-1&&x[i]!=x[j])j=nxt[j];
        nxt[++i]=++j;
    }
}
//x:pa y:tx
int kmp(char x[],int m,char y[],int n){
    int i,j,ans;
    i=j=ans=0;
    initkmp(x,m);
    while(i<n){
        while(j!=-1&&y[i]!=x[j])j=nxt[j];
        i++,j++;
        if(j>=m){
            ans++;
            j=nxt[j];
            //pos:i-m
        }
    }
    return ans;
}
```

2.2 Trie

```
//hihocoder 1014
const int maxnode=2600000+10;
const int sigma_size=26;
struct Trie{
    int ch[maxnode][sigma_size];
    int val[maxnode];
    int sz;
    void init(){sz=0;clr(ch[0],0);}
    int idx(char c){return c-'a';}
    void insert(char *s){
        int u=0,n=strlen(s);
        rep(i,0,n-1){
            int x=idx(s[i]);
            if(!ch[u][x]){
                ++sz;
                clr(ch[sz],0);
                val[sz]=0;
                ch[u][x]=sz;
            }
            u=ch[u][x];
            val[u]++;
        }
    }
    int query(char *s){
        int u=0,n=strlen(s),res=0;
        rep(i,0,n-1){
            int x=idx(s[i]);
            if(!ch[u][x])break;
            u=ch[u][x];
            if(i==n-1)res=val[u];
        }
        return res;
    }
}trie;
char s[30];
int work(){
    trie.init();
    int n,m;
    scanf("%d",&n);
    while(n--){
        scanf("%s",s);
        trie.insert(s);
    }
    scanf("%d",&m);
    while(m--){
        scanf("%s",s);
        printf("%d\n",trie.query(s));
    }
    return 0;
}
```

2.3 Manacher

```
//hihocoder 1032
const int MAXN=2e6+10;//more than 2 times !
char s[MAXN],str[MAXN];
int len1,len2,p[MAXN];
void init(){
    str[0]='$';
    str[1]='#';
    rep(i,0,len1){
        str[i*2+2]=s[i];
        str[i*2+3]='#';
    }
    len2=len1*2+2;
    str[len2]='*';
}
int manacher(){
    int id=0,mx=0,ans=0;
    rep(i,1,len2-1){
        if(mx>i)p[i]=min(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;
        }
        ans=max(ans,p[i]);
    }
    return ans-1;
}
int work(){
    int T;
    scanf("%d",&T);
    while(T--){
        scanf("%s",s);
        len1=strlen(s);
        init();
        printf("%d\n",manacher());
    }
    return 0;
}
```

2.4 SaHash

```
#include<bits/stdc++.h>
using namespace std;
typedef unsigned long long ull;
const int MAXN=3e5+10;
const int P=131;
char s[MAXN];
int len;
ull base[MAXN];
ull f[MAXN];
int sa[MAXN],height[MAXN];
ull H(int l,int r){
    return f[r]-f[l-1]*base[r-l+1];
}
int lcp(int x,int y){
    int l=0,r=min(len-x+1,len-y+1),ans=0;
    while(l<=r){
        int mid=(l+r)>>1;
        if(H(x,x+mid-1)==H(y,y+mid-1)){
            ans=mid;
            l=mid+1;
        }else{
            r=mid-1;
        }
    }
    return ans;
}
bool cmp(int x,int y){
    int d=lcp(x,y);
    return s[x+d]<s[y+d];
}
void calc_height(){
    for(int i=2;i<=len;i++){
        height[i]=lcp(sa[i-1],sa[i]);
    }
}
int main(){
    scanf("%s",s+1);
    len=strlen(s+1);
    base[0]=1;
    for(int i=1;i<=len;i++){
        sa[i]=i;
        base[i]=base[i-1]*P;
        f[i]=f[i-1]*P+(s[i]-'a'+1);
    }
    sort(sa+1,sa+1+len,cmp);
    calc_height();
    for(int i=1;i<=len;i++){
        printf("%d%c",sa[i]-1," \n"[i==len]);
    }
    for(int i=1;i<=len;i++){
        printf("%d%c",height[i]," \n"[i==len]);
    }
    return 0;
}
```

2.5 SA

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<algorithm>
#include<vector>
using namespace std;
const int SIZE = 300010, INF = 1 << 30;
int a[SIZE], sa[SIZE], rk[SIZE], fir[SIZE], sec[SIZE], c[SIZE], h[SIZE];
char str[SIZE];
int len;
bool comp(int i, int j, int k){
    return sec[i] == sec[j] && sec[i + k] == sec[j + k];
}
void sufarr(int n){
    int i, p, l, m = 200;
    for (i = 0;i < m;i++) c[i] = 0;
    for (i = 0;i < n;i++) c[rk[i] = a[i]]++;
    for (i = 1;i < m;i++) c[i] += c[i - 1];
    for (i = n - 1;i >= 0;i--) sa[--c[a[i]]] = i;
    for (l = p = 1;p < n;l *= 2, m = p)
    {
        for (p = 0, i = n - l;i < n;i++) sec[p++] = i;
        for (i = 0;i < n;i++)
            if (sa[i] >= l) sec[p++] = sa[i] - l;
        for (i = 0;i < n;i++) fir[i] = rk[sec[i]];
        for (i = 0;i < m;i++) c[i] = 0;
        for (i = 0;i < n;i++) c[fir[i]]++;
        for (i = 1;i < m;i++) c[i] += c[i - 1];
        for (i = n - 1;i >= 0;i--) sa[--c[fir[i]]] = sec[i];
        memcpy(sec, rk, sizeof(rk));
        rk[sa[0]] = 0;
        for (i = p = 1;i < n;i++)
            rk[sa[i]] = comp(sa[i], sa[i - 1], l) ? p - 1 : p++;
    }
}
void calh(){
    int i, j, k = 0;
    for (i = 1;i <= len;i++) rk[sa[i]] = i;
    for (i = 0;i < len;h[rk[i++]] = k)
        for (k ? k-- : 0, j = sa[rk[i] - 1];a[i + k] == a[j + k];k++);
}
int main(){
    scanf("%s", str);
    len = strlen(str);
    for (int i = 0;i < len;i++) a[i] = str[i];
    a[len] = 0;
    sufarr(len + 1);
    calh();
    for(int i = 1; i <= len; i++) printf("%d ", sa[i]); puts("");
    for(int i = 1; i <= len; i++) printf("%d ", h[i]); puts("");
}
```

2.6 HashString

```
//poj 3974
#include<cstdio>
#include<algorithm>
#include<cstring>
using namespace std;
typedef unsigned long long ull;
const int MAXN=1e6+10;
char s[MAXN];
ull a[MAXN];
ull b[MAXN];
ull base[MAXN];
inline ull H(int i, int j) {
    return (a[j] - a[i - 1] * base[j - i + 1]);
}
inline ull H2(int i, int j) {
    return (b[i] - b[j + 1] * base[j - i + 1]);
}
int main(){
    base[0]=1;
    for(int i=1;i<MAXN;i++){
        base[i]=base[i-1]*131;
    }
    int kase=0;
    for(;;){
        scanf("%s",s+1);
        if(s[1]=='E')break;
        int len=strlen(s+1);
        a[0]=b[len+1]=0;
        for(int i=1;i<=len;i++){
            a[i]=a[i-1]*131+s[i]-'a';
        }
        for(int i=len;i>=1;i--){
            b[i]=b[i+1]*131+s[i]-'a';
        }
        int ans=1;
        for(int pos=1;pos<=len;pos++){
            int l=1,r=min(pos-1,len-pos);
            while(l<=r){
                int mid=(l+r)>>1;
                if(H(pos-mid,pos-1)==H2(pos+1,pos+mid)){
                    ans=max(2*mid+1,ans);
                    l=mid+1;
                }else{
                    r=mid-1;
                }
            }
            l=1,r=min(pos-1,len-pos+1);
            while(l<=r){
                int mid=(l+r)>>1;
                if(H(pos-mid,pos-1)==H2(pos,pos+mid-1)){
                    ans=max(2*mid,ans);
                    l=mid+1;
                }else{
                    r=mid-1;
                }
            }
        }
        printf("Case %d: ",++kase);
        printf("%d\n",ans);
    }
    return 0;
}
```

3 Data Structure

3.1 other

3.1.1 QuickSelect

```
anytype QuickSelect(anytype arr[],int l,int r,int k){
    int i=l,j=r,mid=arr[(i+j)>>1];
    while(i<=j){
        while(arr[i]<mid)i++;
        while(arr[j]>mid)j--;
        if(i<=j){
            swap(arr[i],arr[j]);
            i++;
            j--;
        }
    }
    if(l<j&&k<=j)return QuickSelect(arr,l,j,k);
    if(i<r&&k>=i)return QuickSelect(arr,i,r,k);
    return arr[k];
}
```

3.1.2 mergingsort

```
//hdu 1394
const int MAXN=5005;
int n;
vi A;
int x[MAXN];
int merging(vi &a){
    int n=SZ(a);
    if(n<=1)return 0;
    int cnt=0;
    vi b(a.begin(),a.begin()+n/2);
    vi c(a.begin()+n/2,a.end());
    cnt+=merging(b);
    cnt+=merging(c);
    int ai=0,bi=0,ci=0;
    while(ai<n){
        if(bi<SZ(b)&&(ci==SZ(c)||b[bi]<=c[ci])){
            a[ai++]=b[bi++];
        }else{
            cnt+=n/2-bi;
            a[ai++]=c[ci++];
        }
    }
    return cnt;
}
int work(){
    while(~scanf("%d",&n)){
        A.clear();
        rep(i,1,n)scanf("%d",&x[i]),A.pb(x[i]);
        int sum=merging(A);
        int res=sum;
        rep(i,1,n){
            sum=sum-x[i]+(n-1-x[i]);
            res=min(res,sum);
        }
        printf("%d\n",res);
    }
    return 0;
}
```

3.1.3 pbds

```
//cf 1042d
#include<bits/stdc++.h>
#include<ext/pb_ds/assoc_container.hpp>
using namespace std;
using namespace __gnu_pbds;
typedef long long ll;
tree<pair<ll,int>,null_type,less<pair<ll,int> >,rb_tree_tag,
    tree_order_statistics_node_update > rbt;
int main(){
    int n;
    ll t;
    scanf("%d%I64d",&n,&t);
    rbt.insert({0,0});
    ll now=0,ans=0;
    for(int i=1;i<=n;i++){
        ll x;
        scanf("%I64d",&x);
        now+=x;
        ans+=i-rbt.order_of_key({now-t,n+1});
        rbt.insert({now,i});
    }
    printf("%I64d",ans);
    return 0;
}
```

3.1.4 stack

```
//poj 2559
#include<cstdio>
#include<algorithm>
using namespace std;
typedef long long ll;
const int MAXN=1e5+10;
int a[MAXN];
int w[MAXN];
int stk[MAXN];
int top;
int main(){
    int n;
    while(scanf("%d",&n),n){
        ll ans=0;
        top=0;
        stk[top]=0;
        for(int i=1;i<=n+1;i++){
            if(i<=n)scanf("%d",&a[i]);
            else a[i]=0;
            if(a[i]>a[stk[top]]){
                stk[++top]=i;
                w[top]=1;
            }else{
                int width=0;
                while(a[i]<a[stk[top]]){
                    width+=w[top];
                    ans=max(ans,1LL*a[stk[top]]*width);
                    top--;
                }
                stk[++top]=i;
                w[top]=width+1;
            }
        }
        printf("%lld\n",ans);
    }
    return 0;
}
```

3.1.5 queue

```
//ch 1201
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int MAXN=3e5+10;
ll sum[MAXN];
int que[MAXN];
int st,ed;
int main(){
    int n,m;
    scanf("%d%d",&n,&m);
    sum[0]=0;
    st=ed=0;
    que[ed++]=0;
    ll ans=0;
    for(int i=1;i<=n;i++){
        scanf("%lld",&sum[i]);
        sum[i]+=sum[i-1];
        while(i-que[st]>m){
            st++;
        }
        ans=max(ans,sum[i]-sum[que[st]]);
        while(st!=ed&&sum[que[ed-1]]>=sum[i]){
            ed--;
        }
        que[ed++]=i;
    }
    printf("%lld",ans);
    return 0;
}
```

3.2 Binary Indexed Tree

```
//add(pos,a) sum(r)-sum(l-1)
//add(l,a) add(r+1,-a) sum(pos)
const int MAXN=100000;
struct BIT{
    int n,c[MAXN<<1];
    void init(int _n){
        n=_n;
        rep(i,0,n)c[i]=0;
    }
    void update(int i,int v){
        for(;i<=n;i+=i&-i)c[i]+=v;
    }
    int query(int i){
        int s=0;
        for(;i;i-=i&-i)s+=c[i];
        return s;
    }
    int findpos(int v){
        int sum=0;
        int pos=0;
        int i=1;
        for(;i<n;i<<=1);
        for(;i;i>>=1){
            if(pos+i<=n&&sum+c[pos+i]<v){
                sum+=c[pos+i];
                pos+=i;
            }
        }
        return pos+1;
    }
}bit;
```

3.2.1 poj3468

$a_i = \sum_{i=1}^{x} d_i$
$\sum_{i=1}^{x} a_i = \sum_{i=1}^{x} \sum_{j=1}^{i} d_j = \sum_{i=1}^{x} (x-i+1) d_i$
$\sum_{i=1}^{x} a_i = (x+1) \sum_{i=1}^{x} d_i - \sum_{i=1}^{x} d_i \times i$

```
const int MAXN=1e5+5;
int n,q,x,y,z;
long long c1[MAXN],c2[MAXN];
void add(int x,int y){
    for(int i=x;i<=n;i+=i&(-i))c1[i]+=y,c2[i]+=1LL*x*y;
}
ll sum(int x){
    ll ans(0);
    for(int i=x;i;i-=i&(-i))ans+=1LL*(x+1)*c1[i]-c2[i];
    return ans;
}
char op[5];
int work(){
    scanf("%d%d",&n,&q);
    int a1,a2;
    a1=0;
    rep(i,1,n){
        scanf("%d",&a2);
        add(i,a2-a1);
        a1=a2;
    }
    while(q--){
        scanf("%s",op);
        if(op[0]=='Q'){
            scanf("%d%d%d",&x,&y,&z);
            printf("%lld\n",sum(y)-sum(x-1));
        }else{
            scanf("%d%d%d",&x,&y,&z);
            add(x,z);
            add(y+1,-z);
        }
    }
    return 0;
}
```

3.3 Segment Tree

```
#define lson rt<<1
#define rson rt<<1|1
#define le l,m,lson
#define ri m+1,r,rson
#define mid m=(l+r)>>1
```

3.3.1 Single-point Update

```
const int MAXN=5e4+5;
int sum[MAXN<<2];
void push_up(int rt){
    sum[rt]=sum[lson]+sum[rson];
}
void build(int l,int r,int rt){
    if(l==r){
        scanf("%d",&sum[rt]);
        return;
    }
    int mid;
    build(le);
    build(ri);
    push_up(rt);
}
void update(int p,int v,int l,int r,int rt){
    if(l==r){
        sum[rt]+=v;
        return;
    }
    int mid;
    if(p<=m)update(p,v,le);
    else update(p,v,ri);
    push_up(rt);
}
int query(int L,int R,int l,int r,int rt){
    if(L<=l&&r<=R){
        return sum[rt];
    }
    int mid;
    int ret=0;
    if(L<=m)ret+=query(L,R,le);
    if(R>m)ret+=query(L,R,ri);
    return ret;
}
```

3.3.2 Interval Update

```
const int MAXN=1e5+5;
ll lazy[MAXN<<2];
ll tree[MAXN<<2];
void push_up(int rt){
    tree[rt]=tree[lson]+tree[rson];
}
void push_down(int rt,int m){
    ll w=lazy[rt];
    if(w){
        lazy[lson]+=w;
        lazy[rson]+=w;
        tree[lson]+=w*(m-(m>>1));
        tree[rson]+=w*(m>>1);
        lazy[rt]=0;
    }
}
void build(int l,int r,int rt){
    lazy[rt]=0;
    if(l==r){
        scanf("%lld",&tree[rt]);
        return;
    }
    int mid;
    build(le);
    build(ri);
    push_up(rt);
}
void update(int L,int R,int v,int l,int r,int rt){
    if(L<=l&&r<=R){
        lazy[rt]+=v;
        tree[rt]+=1ll*v*(r-l+1);
        return;
    }
    push_down(rt,r-l+1);
    int mid;
    if(L<=m)update(L,R,v,le);
    if(R>m)update(L,R,v,ri);
    push_up(rt);
}
ll query(int L,int R,int l,int r,int rt){
    if(L<=l&&r<=R){
        return tree[rt];
    }
    push_down(rt,r-l+1);
    int mid;
    ll ret=0;
    if(L<=m)ret+=query(L,R,le);
    if(R>m)ret+=query(L,R,ri);
    return ret;
}
```

3.4 Splay Tree

```
#define key_value ch[ch[rt][1]][0]
const int MAXN=1e5;
struct Splay{
    int a[MAXN];//0 base
    int sz[MAXN],ch[MAXN][2],fa[MAXN];
    int key[MAXN],rev[MAXN];
    int rt,tot;
    int stk[MAXN],top;
    void push_up(int x){
        sz[x]=sz[ch[x][0]]+sz[ch[x][1]]+1;
    }
    void push_down(int x){
        if(rev[x]){
            swap(ch[x][0],ch[x][1]);
            if(ch[x][0])rev[ch[x][0]]^=1;
            if(ch[x][1])rev[ch[x][1]]^=1;
            rev[x]=0;
        }
    }
    int newnode(int p=0,int k=0){
        int x=top?stk[top--]:++tot;
        fa[x]=p;
        sz[x]=1;
        ch[x][0]=ch[x][1]=0;
        key[x]=k;
        rev[x]=0;
        return x;
    }
    int build(int l,int r,int p){
        if(l>r)return 0;
        int mid=(l+r)>>1;
        int x=newnode(p,a[mid]);
        ch[x][0]=build(l,mid-1,x);
        ch[x][1]=build(mid+1,r,x);
        push_up(x);
        return x;
    }
    void init(int n){
        tot=0,top=0;
        rt=newnode(0,-1);
        ch[rt][1]=newnode(rt,-1);
        rep(i,0,n-1)a[i]=i+1;
        key_value=build(0,n-1,ch[rt][1]);
        push_up(ch[rt][1]);
        push_up(rt);
    }
    void rotate(int x,int d){
        int y=fa[x];
        push_down(y);
        push_down(x);
        ch[y][d^1]=ch[x][d];
        fa[ch[x][d]]=y;
        if(fa[y])ch[fa[y]][ch[fa[y]][1]==y]=x;
        fa[x]=fa[y];
        ch[x][d]=y;
        fa[y]=x;
        push_up(y);
    }
    void splay(int x,int goal=0){
        push_down(x);
        while(fa[x]!=goal){
            if(fa[fa[x]]==goal){
                rotate(x,ch[fa[x]][0]==x);
            }else{
                int y=fa[x];
                int d=ch[fa[y]][0]==y;
                ch[y][d]==x?rotate(x,d^1):rotate(y,d);
                rotate(x,d);
            }
        }
        push_up(x);
        if(goal==0)rt=x;
    }
    int kth(int r,int k){
        push_down(r);
        int t=sz[ch[r][0]]+1;
        if(t==k)return r;
        return t>k?kth(ch[r][0],k):kth(ch[r][1],k-t);
    }
    void select(int l,int r){
        splay(kth(rt,1),0);
        splay(kth(ch[rt][1],r-l+2),rt);
    }
};
```

3.5 Functional Segment Tree

```
//poj 2104
const int MAXN=1e5+6;
int n,m,cnt,x,y,k,root[MAXN],a[MAXN];
struct node{int l,r,sum;}T[MAXN*40];
vi v;
int getid(int x){return lower_bound(all(v),x)-v.begin()+1;}
void update(int l,int r,int &x,int y,int pos){
    x=++cnt;
    T[x]=T[y];
    T[x].sum++;
    if(l==r)return;
    int mid=(l+r)>>1;
    if(mid>=pos)update(l,mid,T[x].l,T[y].l,pos);
    else update(mid+1,r,T[x].r,T[y].r,pos);
}
int query(int l,int r,int x,int y,int k){
    if(l==r)return l;
    int sum=T[T[y].l].sum-T[T[x].l].sum;
    int mid=(l+r)>>1;
    if(sum>=k)return query(l,mid,T[x].l,T[y].l,k);
    else return query(mid+1,r,T[x].r,T[y].r,k-sum);
}
int work(){
    scanf("%d%d",&n,&m);
    v.clear();
    rep(i,1,n)scanf("%d",&a[i]),v.pb(a[i]);
    sort(all(v)),v.erase(unique(all(v)),v.end());
    cnt=0;
    rep(i,1,n)update(1,n,root[i],root[i-1],getid(a[i]));
    rep(i,1,m)scanf("%d%d%d",&x,&y,&k),printf("%d\n",v[query(1,n,root[x-1],root[y],k)
    -1]);
    return 0;
}
```

3.6 Sparse Table

```
//Frequent values UVA - 11235
#include<bits/stdc++.h>
using namespace std;
const int MAXN=1e5+10;
int dp[MAXN][33];
int a[MAXN],b[MAXN],Belong[MAXN];
int rmq(int l,int r){
    int k=31-__builtin_clz(r-l+1);
    return max(dp[l][k],dp[r-(1<<k)+1][k]);
}
int main(){
    int n;
    while(scanf("%d",&n),n){
        int q;
        scanf("%d",&q);
        int index=0;
        int now=-111111;
        for(int i=1;i<=n;i++){
            int x;
            scanf("%d",&x);
            if(now!=x){
                index++;
                now=x;
                a[index]=i;
            }
            Belong[i]=index;
            b[index]=i;
        }
        for(int i=1;i<=index;i++){
            dp[i][0]=b[i]-a[i]+1;
        }
        for (int j = 1; (1 << j) <= index; j++){
            for (int i = 1; i + (1 << j) - 1 <= index; i++){
                dp[i][j] = max(dp[i][j - 1], dp[i + (1 << (j - 1))][j - 1]);
            }
        }
        while(q--){
            int l,r;
            scanf("%d%d",&l,&r);
            if(Belong[l]==Belong[r]){
                printf("%d\n",r-l+1);
            }else{
                int pos1=Belong[l];
                int ans=b[pos1]-l+1;
                int pos2=Belong[r];
                ans=max(ans,r-a[pos2]+1);
                pos1++;
                pos2--;
                if(pos1<=pos2){
                    ans=max(ans,rmq(pos1,pos2));
                }
                printf("%d\n",ans);
            }
        }

    }
    return 0;
}
```

3.7 block

```
//poj 3468
#include <algorithm>
#include  <iostream>
#include   <cstring>
#include    <string>
#include    <cstdio>
#include    <vector>
#include     <stack>
#include     <queue>
#include     <cmath>
#include       <set>
#include       <map>
using namespace std;
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define per(i,a,b) for(int i=a;i>=b;i--)
#define clr(a,x) memset(a,x,sizeof(a))
#define pb push_back
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef unsigned long long ull;
typedef long long ll;
typedef vector<int> vi;
typedef pair<int,int> pii;
/************head******************/
const int MAXN=1e5+10;
int L[MAXN],R[MAXN],pos[MAXN];
ll a[MAXN],b[MAXN],c[MAXN];
int t;
void update(int x,int y,int z){
    int l=pos[x];
    int r=pos[y];
    if(l==r){
        for(int i=x;i<=y;i++){
            a[i]+=z;
        }
        b[l]+=1LL*z*(y-x+1);
    }else{
        for(int i=l+1;i<r;i++){
            c[i]+=z;
        }
        for(int i=x;i<=R[l];i++){
            a[i]+=z;
        }
        b[l]+=1LL*z*(R[l]-x+1);
        for(int i=L[r];i<=y;i++){
            a[i]+=z;
        }
        b[r]+=1LL*z*(y-L[r]+1);
    }
}
ll query(int x,int y){
    ll res=0;
    int l=pos[x];
    int r=pos[y];
    if(l==r){
        for(int i=x;i<=y;i++){
            res+=a[i];
        }
        res+=c[l]*(y-x+1);
    }else{
        for(int i=l+1;i<r;i++){
            res+=c[i]*(R[i]-L[i]+1)+b[i];
        }
        for(int i=x;i<=R[l];i++){
            res+=a[i];
        }
        res+=c[l]*(R[l]-x+1);
        for(int i=L[r];i<=y;i++){
            res+=a[i];
        }
        res+=c[r]*(y-L[r]+1);
    }
    return res;
}
int main(){
    int n,q;
    scanf("%d%d",&n,&q);
    t=sqrt(n);
    for(int i=1;i<=t;i++){
        L[i]=(i-1)*t+1;
        R[i]=i*t;
    }
    if(R[t]<n){
        t++;
        L[t]=R[t-1]+1;
        R[t]=n;
    }
    for(int i=1;i<=n;i++){
        scanf("%lld",&a[i]);
    }
    for(int i=1;i<=t;i++){
        for(int j=L[i];j<=R[i];j++){
            pos[j]=i;
            b[i]+=a[j];
        }
    }
    char op[5];
    while(q--){
        int x,y;
        scanf("%s%d%d",op,&x,&y);
        if(op[0]=='Q'){
            printf("%lld\n",query(x,y));
        }else{
            int z;
            scanf("%d",&z);
            update(x,y,z);
        }
    }
    return 0;
}
```

3.8 Treap

```cpp
#include <algorithm>
#include  <iostream>
#include   <cstring>
#include    <string>
#include    <cstdio>
#include    <vector>
#include     <stack>
#include     <queue>
#include     <cmath>
#include       <set>
#include       <map>
using namespace std;
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define per(i,a,b) for(int i=a;i>=b;i--)
#define clr(a,x) memset(a,x,sizeof(a))
#define pb push_back
#define all(x) (x).begin(),(x).end()
#define fi first
#define se second
#define SZ(x) ((int)(x).size())
typedef unsigned long long ull;
typedef long long ll;
typedef vector<int> vi;
typedef pair<int,int> pii;
/************head******************/
const int SIZE=1e5+10;
struct Treap{
    int l,r;
    int val,dat;
    int cnt,sz;
}a[SIZE];
int tot,root,n,INF=0x7fffffff;
int New(int val){
    a[++tot].val=val;
    a[tot].dat=rand();
    a[tot].cnt=a[tot].sz=1;
    return tot;
}
void Update(int p){
    a[p].sz=a[a[p].l].sz+a[a[p].r].sz+a[p].cnt;
}
void Build(){
    New(-INF);
    New(INF);
    root=1;
    a[1].r=2;
    Update(root);
}
int GetRankByVal(int p,int val){
    if(p==0)return 0;
    if(val==a[p].val)return a[a[p].l].sz+1;
    if(val<a[p].val)return GetRankByVal(a[p].l,val);
    return GetRankByVal(a[p].r,val)+a[a[p].l].sz+a[p].cnt;
}
int GetValByRank(int p,int rk){
    if(p==0)return INF;
    if(a[a[p].l].sz>=rk)return GetValByRank(a[p].l,rk);
    if(a[a[p].l].sz+a[p].cnt>=rk)return a[p].val;
    return GetValByRank(a[p].r,rk-a[a[p].l].sz-a[p].cnt);
```

```
}
void zig(int &p){
    int q=a[p].l;
    a[p].l=a[q].r;
    a[q].r=p;
    p=q;
    Update(a[p].r);
    Update(p);
}
void zag(int &p){
    int q=a[p].r;
    a[p].r=a[q].l;
    a[q].l=p;
    p=q;
    Update(a[p].l);
    Update(p);
}
void Insert(int &p,int val){
    if(p==0){
        p=New(val);
        return;
    }
    if(val==a[p].val){
        a[p].cnt++;
        Update(p);
        return;
    }
    if(val<a[p].val){
        Insert(a[p].l,val);
        if(a[p].dat<a[a[p].l].dat)zig(p);
    }else{
        Insert(a[p].r,val);
        if(a[p].dat<a[a[p].r].dat)zag(p);
    }
    Update(p);
}
int GetPre(int val){
    int ans=1;
    int p=root;
    while(p){
        if(val==a[p].val){
            if(a[p].l>0){
                p=a[p].l;
                while(a[p].r>0)p=a[p].r;
                ans=p;
            }
            break;
        }
        if(a[p].val<val&&a[p].val>a[ans].val)ans=p;
        p=val<a[p].val?a[p].l:a[p].r;
    }
    return a[ans].val;
}
int GetNext(int val){
    int ans=2;
    int p=root;
    while(p){
        if(val==a[p].val){
            if(a[p].r>0){
                p=a[p].r;
                while(a[p].l>0)p=a[p].l;
                ans=p;
            }
            break;
        }
        if(a[p].val>val&&a[p].val<a[ans].val)ans=p;
        p=val<a[p].val?a[p].l:a[p].r;
    }
    return a[ans].val;
}
void Remove(int &p,int val){
    if(p==0)return;
    if(val==a[p].val){
        if(a[p].cnt>1){
            a[p].cnt--;
            Update(p);
            return;
        }
        if(a[p].l||a[p].r){
            if(a[p].r==0||a[a[p].l].dat>a[a[p].r].dat){
                zig(p);
                Remove(a[p].r,val);
            }else{
                zag(p);
                Remove(a[p].l,val);
            }
            Update(p);
        }else{
            p=0;
        }
        return;
    }
    val<a[p].val?Remove(a[p].l,val):Remove(a[p].r,val);
    Update(p);
}
int main(){
    Build();
    int n;
    scanf("%d",&n);
    while(n--){
        int op,x;
        scanf("%d%d",&op,&x);
        switch(op){
        case 1:
            Insert(root,x);
            break;
        case 2:
            Remove(root,x);
            break;
        case 3:
            printf("%d\n",GetRankByVal(root,x)-1);
            break;
        case 4:
            printf("%d\n",GetValByRank(root,x+1));
            break;
        case 5:
            printf("%d\n",GetPre(x));
            break;
        case 6:
            printf("%d\n",GetNext(x));
            break;
        }
    }
    return 0;
}
```

4 Graph Theory

4.1 Union-Find Set

```cpp
const int MAXN=1e6+5;
struct DSU{
    int p[MAXN];
    void init(int n){rep(i,0,n)p[i]=i;}
    int findp(int x){return x==p[x]?x:p[x]=findp(p[x]);}
    void unite(int x,int y){x=findp(x);y=findp(y);if(x==y)return;p[y]=x;}
    bool same(int x,int y){return findp(x)==findp(y);}
}dsu;
```

4.2 Minimal Spanning Tree

4.2.1 Kruskal

```cpp
//poj 1258
#include<cstdio>
#include<algorithm>
using namespace std;
const int MAXE=1e5+5;
const int MAXN=1e5+5;
struct DSU{
    int p[MAXN];
    void init(int n){for(int i=0;i<=n;i++)p[i]=i;}
    int findp(int x){return x==p[x]?x:p[x]=findp(p[x]);}
    void unite(int x,int y){x=findp(x);y=findp(y);if(x==y)return;p[y]=x;}
    bool same(int x,int y){return findp(x)==findp(y);}
}dsu;
struct edge{int u,v,cost;}es[MAXE];
bool cmp(const edge &x,const edge &y){return x.cost<y.cost;}
int V,E;
int kruskal(){
    sort(es,es+E,cmp);
    dsu.init(V);
    int res=0;
    for(int i=0;i<E;i++){
        if(!dsu.same(es[i].u,es[i].v)){
            dsu.unite(es[i].u,es[i].v);
            res+=es[i].cost;
        }
    }
    return res;
}
int main(){
    while(~scanf("%d",&V)){
        E=0;
        for(int i=1;i<=V;i++){
            for(int j=1;j<=V;j++){
                int w;
                scanf("%d",&w);
                if(i==j)continue;
                es[E].u=i;
                es[E].v=j;
                es[E].cost=w;
                E++;
            }
        }
        printf("%d\n",kruskal());
    }
    return 0;
}
```

4.3 Shortest Path

4.3.1 Dijkstra

```
#include<bits/stdc++.h>
using namespace std;
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define clr(a,x) memset(a,x,sizeof(a))
#define mp make_pair
const int MAXV=2e6;
const int MAXE=5e6+10;
typedef long long anytype;
typedef pair<anytype,int> P;
int tot=0;
int head[MAXV];
struct Edge{
    int v,c,nxt;
    Edge(){}
    Edge(int v,int c,int nxt):v(v),c(c),nxt(nxt){}
}edge[MAXE];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v,int c){
    edge[tot]=Edge(v,c,head[u]);
    head[u]=tot++;
}
anytype d[MAXV];
void dij(int s){
    priority_queue<P,vector<P>,greater<P> > que;
    clr(d,-1);
    d[s]=0;
    que.push(P(0,s));
    while(!que.empty()){
        P t=que.top();
        que.pop();
        int v=t.second;
        if(d[v]!=-1&&d[v]<t.first)continue;
        for(int i=head[v];~i;i=edge[i].nxt){
            Edge e=edge[i];
            if(d[e.v]==-1||d[e.v]>d[v]+e.c){
                d[e.v]=d[v]+e.c;
                que.push(mp(d[e.v],e.v));
            }
        }
    }
}
int main(){
    int T;
    scanf("%d",&T);
    while(T--){
        int n,m,k;
        scanf("%d%d%d",&n,&m,&k);
        init();
        rep(i,1,m){
            int u,v,c;
            scanf("%d%d%d",&u,&v,&c);
            rep(j,0,k){
                add_edge(u+j*n,v+j*n,c);
                if(j!=k)add_edge(u+j*n,v+(j+1)*n,0);
            }
        }
        dij(1);
        printf("%lld\n",d[n+k*n]);
    }
    return 0;
}
```

4.3.2 Spfa

```
//hdu3592
const int MAXN=1e3+5;
const int MAXE=3e4+5;
const int INF=0x3f3f3f3f;
int N,X,Y;
int tot;
int head[MAXN];
struct Edge{
    int v,w,nxt;
    Edge(){}
    Edge(int v,int w,int nxt):v(v),w(w),nxt(nxt){}
}edge[MAXE];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v,int w){
    edge[tot]=Edge(v,w,head[u]);
    head[u]=tot++;
}
queue<int> que;
bool inq[MAXN];
int qtime[MAXN];
int d[MAXN];
int spfa(){
    while(!que.empty())que.pop();
    clr(qtime,0);
    clr(inq,0);
    rep(i,1,N)d[i]=INF;
    d[1]=0;
    que.push(1);
    inq[1]=1;
    qtime[1]++;
    while(!que.empty()){
        int u=que.front();
        que.pop();
        inq[u]=0;
        for(int i=head[u];i!=-1;i=edge[i].nxt){
            int v=edge[i].v;
            int w=edge[i].w;
            if(d[v]>d[u]+w){
                d[v]=d[u]+w;
                if(!inq[v]){
                    que.push(v);
                    inq[v]=1;
                    qtime[v]++;
                    if(qtime[v]>N)return -1;
                }
            }
        }
    }
    if(d[N]==INF)return -2;
    else return d[N];
}
int work(){
    int T;
    scanf("%d",&T);
    while(T--){
        scanf("%d%d%d",&N,&X,&Y);
        init();
        rep(i,1,N-1){
            add_edge(i+1,i,0);
        }
        while(X--){
            int x,y,z;
            scanf("%d%d%d",&x,&y,&z);
            add_edge(x,y,z);
        }
        while(Y--){
            int x,y,z;
            scanf("%d%d%d",&x,&y,&z);
            add_edge(y,x,-z);
        }
        printf("%d\n",spfa());
    }
    return 0;
}
```

4.3.3 kth-p

```
#include<bits/stdc++.h>
using namespace std;
#define INF 0xffffff
#define MAXN 100010
struct node{
    int to;
    int val;
    int next;
};
struct node2{
    int to;
    int g,f;
    bool operator<(const node2 &r ) const  {
        if(r.f==f)
            return r.g<g;
        return r.f<f;
    }
};
```

```
node edge[MAXN],edge2[MAXN];
int n,m,s,t,k,cnt,cnt2,ans;
int dis[1010],visit[1010],head[1010],head2[1010];
void init(){
    memset(head,-1,sizeof(head));
    memset(head2,-1,sizeof(head2));
    cnt=cnt2=1;
}
void addedge(int from,int to,int val){
    edge[cnt].to=to;
    edge[cnt].val=val;
    edge[cnt].next=head[from];
    head[from]=cnt++;
}
void addedge2(int from,int to,int val){
    edge2[cnt2].to=to;
    edge2[cnt2].val=val;
    edge2[cnt2].next=head2[from];
    head2[from]=cnt2++;
}
bool spfa(int s,int n,int head[],node edge[],int dist[])  {
    queue<int>Q1;
    int inq[1010];
    for(int i=0;i<=n;i++)  {
        dis[i]=INF;
        inq[i]=0;
    }
    dis[s]=0;
    Q1.push(s);
    inq[s]++;
    while(!Q1.empty())  {
        int q=Q1.front();
        Q1.pop();
        inq[q]--;
        if(inq[q]>n)
            return false;
        int k=head[q];
        while(k>=0)  {
            if(dist[edge[k].to]>dist[q]+edge[k].val)  {
                dist[edge[k].to]=edge[k].val+dist[q];
                if(!inq[edge[k].to])  {
                    inq[edge[k].to]++;
                    Q1.push(edge[k].to);
                }
            }
            k=edge[k].next;
        }
    }
    return true;
}
int A_star(int s,int t,int n,int k,int head[],node edge[],int dist[]) {
    node2 e,ne;
    int cnt=0;
    priority_queue<node2>Q;
    if(s==t)
        k++;
    if(dis[s]==INF)
        return -1;
    e.to=s;
    e.g=0;
    e.f=e.g+dis[e.to];
    Q.push(e);

    while(!Q.empty())  {
        e=Q.top();
        Q.pop();
        if(e.to==t)//□□□□□□□□
        {
            cnt++;
        }
        if(cnt==k)//□□k□□
        {
            return e.g;
        }
        for(int i=head[e.to]; i!=-1; i=edge[i].next)  {
            ne.to=edge[i].to;
            ne.g=e.g+edge[i].val;
            ne.f=ne.g+dis[ne.to];
            Q.push(ne);
        }
    }
    return -1;
}
int main(){
    while(~scanf("%d%d",&n,&m)){
        init();
        for(int i=1;i<=m;i++){
            int a,b,c;
            scanf("%d%d%d",&a,&b,&c);
            addedge(a,b,c);
            addedge2(b,a,c);
        }
        scanf("%d%d%d",&s,&t,&k);
        spfa(t,n,head2,edge2,dis);
        ans=A_star(s,t,n,k,head,edge,dis);
        printf("%d\n",ans);
    }
    return 0;
}
```

4.4 Topo Sort

```
//cf 915D
const int MAXN=505;
const int MAXM=1e5+5;
int n,m;
int tot;
int head[MAXN],cur[MAXN],idec[MAXN];
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXM];
void init(){
    tot=0;
    clr(head,-1);
}
```

```
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    head[u]=tot++;
}
int que[MAXN];
int st,ed;
bool topsort(int x){
    int nst=1,ned=0;
    rep(i,1,n)cur[i]=idec[i];
    cur[x]--;
    que[++ned]=x;
    while(nst<=ned){
        int u=que[nst++];
        for(int i=head[u];i!=-1;i=edge[i].nxt){
            int v=edge[i].v;
            if(--cur[v]==0)que[++ned]=v;
        }
    }
    if(ned+ed==n)return true;
    else return false;
}
int work(){
    scanf("%d%d",&n,&m);
    init();
    while(m--){
        int u,v;
        scanf("%d%d",&u,&v);
        add_edge(u,v);
        idec[v]++;
    }
    st=1,ed=0;
    rep(i,1,n){
        if(idec[i]==0)que[++ed]=i;
    }
    while(st<=ed){
        int u=que[st++];
        for(int i=head[u];i!=-1;i=edge[i].nxt){
            int v=edge[i].v;
            if(--idec[v]==0)que[++ed]=v;
        }
    }
    if(ed==n){
        puts("YES");
        return 0;
    }
    rep(i,1,n){
        if(idec[i]==1){
            if(topsort(i)){
                puts("YES");
                return 0;
            }
        }
    }
    puts("NO");
    return 0;
}
```

4.5 LCA

4.5.1 LCA

```
//hdu 2586
const int MAXV=1e5+100;
int tot;
int head[MAXV];
struct Edge{
    int v,w,nxt;
    Edge(){}
    Edge(int v,int w,int nxt):v(v),w(w),nxt(nxt){}
}edge[MAXV<<1];
void init(){
    tot=0;
    memset(head,-1,sizeof(head));
}
void add_edge(int u,int v,int w){
    edge[tot]=Edge(v,w,head[u]);
    head[u]=tot++;
}
int t,f[MAXV][22],d[MAXV];
ll dist[MAXV];
void bfs(){
    queue<int> que;
    que.push(1);
    d[1]=1;
    while(!que.empty()){
        int u=que.front();
        que.pop();
        for(int i=head[u];~i;i=edge[i].nxt){
            int v=edge[i].v;
            if(d[v])continue;
            d[v]=d[u]+1;
            dist[v]=dist[u]+edge[i].w;
            f[v][0]=u;
            for(int j=1;j<=t;j++){
                    f[v][j]=f[f[v][j-1]][j-1];
            }
            que.push(v);
        }
    }
}
int lca(int x,int y){
    if(d[x]>d[y])swap(x,y);
    for(int i=t;i>=0;i--){
        if(d[f[y][i]]>=d[x])y=f[y][i];
    }
    if(x==y)return x;
    for(int i=t;i>=0;i--){
        if(f[x][i]!=f[y][i]){
            x=f[x][i];
            y=f[y][i];
        }
    }
    return f[x][0];
}
int main() {
    int T;
    cin>>T;
    while (T--) {
        int n,m;
        cin >> n >> m;
        t = (int)(log(n) / log(2)) + 1;
        init();
        memset(d,0,sizeof(d));
        for (int i = 1; i < n; i++) {
            int x, y, z;
            scanf("%d%d%d", &x, &y, &z);
            add_edge(x, y, z), add_edge(y, x, z);
        }
        bfs();
        for (int i = 1; i <= m; i++) {
            int x, y;
            scanf("%d%d", &x, &y);
            printf("%lld\n", dist[x] + dist[y] - 2 * dist[lca(x, y)]);
        }
    }
    return 0;
}
```

4.6 Depth-First Traversal

```
vector<int> G[MAXN];
int vis[MAXN];
void dfs(int u){
    vis[u]=1;
    PREVISIT(u);
    for(auto v:G[u]){
        if(!vis[v])dfs(v);
    }
    POSTVISIT(u);
}
```

4.6.1 Biconnected-Component

```
//UVALive - 3523
#include<bits/stdc++.h>
using namespace std;
#define clr(a,x) memset(a,x,sizeof(a))
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define mp make_pair
#define fi first
#define se second
#define pb push_back
typedef pair<int,int> pii;
typedef vector<int> vi;
const int MAXV=1e3+10;
const int MAXE=1e6+10;
int tot;
int head[MAXV];
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXE<<1];
```

```
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    head[u]=tot++;
}
int pre[MAXV],is_cut[MAXV],bccno[MAXV],dfs_clock,bcc_cnt;
vi bcc[MAXV];
stack<pii > st;
int dfs(int u,int fa){
    int lowu=pre[u]=++dfs_clock;
    int child=0;
    for(int i=head[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        pii e=mp(u,v);
        if(!pre[v]){
            st.push(e);
            child++;
            int lowv=dfs(v,u);
            lowu=min(lowu,lowv);
            if(lowv>=pre[u]){
                is_cut[u]=1;
                bcc_cnt++;
                bcc[bcc_cnt].clear();
                for(;;){
                    pii x=st.top();
                    st.pop();
                    if(bccno[x.fi]!=bcc_cnt){
                        bcc[bcc_cnt].pb(x.fi);
                        bccno[x.fi]=bcc_cnt;
                    }
                    if(bccno[x.se]!=bcc_cnt){
                        bcc[bcc_cnt].pb(x.se);
                        bccno[x.se]=bcc_cnt;
                    }
                    if(x.fi==u&&x.se==v)break;
                }
            }
        }else if(pre[v]<pre[u]&&v!=fa){
            st.push(e);
            lowu=min(lowu,pre[v]);
        }
    }
    if(fa<0&&child==1)is_cut[u]=0;
    return lowu;
}
void find_bcc(int n){
    clr(pre,0);
    clr(is_cut,0);
    clr(bccno,0);
    dfs_clock=bcc_cnt=0;
    rep(i,1,n){
        if(!pre[i])dfs(i,-1);
    }
}
int odd[MAXV],color[MAXV];
bool bipartite(int u,int b){
    for(int i=head[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        if(bccno[v]!=b)continue;
        if(color[v]==color[u])return false;
        if(!color[v]){
            color[v]=3-color[u];
            if(!bipartite(v,b))return false;
        }
    }
    return true;
}
bool mmp[MAXV][MAXV];
int main(){
    int n,m;
    while(scanf("%d%d",&n,&m),n+m){
        clr(mmp,0);
        rep(i,1,m){
            int x,y;
            scanf("%d%d",&x,&y);
            mmp[x][y]=1;
            mmp[y][x]=1;
        }
        init();
        rep(i,1,n){
            rep(j,i+1,n){
                if(!mmp[i][j]){
                    add_edge(i,j);
                    add_edge(j,i);
                }
            }
        }
        find_bcc(n);
        clr(odd,0);
        for(int i=1;i<=bcc_cnt;i++){
            clr(color,0);
            for(int j=0;j<bcc[i].size();j++){
                bccno[bcc[i][j]]=i;
            }
            int u=bcc[i][0];
            color[u]=1;
            if(!bipartite(u,i)){
                for(int j=0;j<bcc[i].size();j++){
                    odd[bcc[i][j]]=1;
                }
            }
        }
        int ans=n;
        rep(i,1,n)if(odd[i])ans--;
        printf("%d\n",ans);
    }
    return 0;
}
```

4.6.2 Strongly Connected Component

```
const int MAXV=1e4+10;
const int MAXE=1e5+10;
int tot,head[MAXV];
int low[MAXV],dfn[MAXV],stk[MAXV],Belong[MAXV];
int idx,top,scc;
bool instk[MAXV];
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXE];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    head[u]=tot++;
}
void Tarjan(int u){
    int v;
    low[u]=dfn[u]=++idx;
    stk[top++]=u;
    instk[u]=true;
    for(int i=head[u];~i;i=edge[i].nxt){
        v=edge[i].v;
        if(!dfn[v]){
            Tarjan(v);
            if(low[u]>low[v])low[u]=low[v];
        }else if(instk[v]&&low[u]>dfn[v])low[u]=dfn[v];
    }
    if(low[u]==dfn[u]){
        scc++;
        do{
            v=stk[--top];
            instk[v]=false;
            Belong[v]=scc;
        }while(v!=u);
    }
}
void tscc(int N){
    clr(dfn,0);
    clr(instk,0);
    idx=scc=top=0;
    rep(i,1,N)if(!dfn[i])Tarjan(i);
}
```

4.6.3 Kosaraju

```
const int MAXV=2e4+10;
const int MAXE=5e4+10;
int tot,scc,head[MAXV],rhead[MAXV],Belong[MAXV];
bool vis[MAXV];
int stk[MAXV],top;
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXE],redge[MAXE];
void init(){
    tot=0;
    clr(head,-1);
    clr(rhead,-1);
}
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    redge[tot]=Edge(u,rhead[v]);
    head[u]=rhead[v]=tot++;
}
void dfs(int u){
    vis[u]=true;
    for(int i=head[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        if(!vis[v])dfs(v);
    }
    stk[++top]=u;
}
void rdfs(int u,int k){
    vis[u]=true;
    Belong[u]=k;
    for(int i=rhead[u];~i;i=redge[i].nxt){
        int v=redge[i].v;
        if(!vis[v])rdfs(v,k);
    }
}
void kscc(int V){
    scc=top=0;
    clr(vis,0);
    rep(i,1,V)if(!vis[i])dfs(i);
    clr(vis,0);
    per(i,top,1){
        int v=stk[i];
        if(!vis[v])rdfs(v,++scc);
    }
}
```

4.6.4 TwoSAT

```
//poj3683
//0 base !
//if (x V (!y))then add_clause(1,x,0,y)
//if  x then add_var(1,x)
const int MAXV=1e5;
const int MAXE=3e6+5;
int tot,scc,head[MAXV],rhead[MAXV],Belong[MAXV];
bool vis[MAXV];
int stk[MAXV],top;
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXE],redge[MAXE];
void init(){
    tot=0;
    clr(head,-1);
    clr(rhead,-1);
}
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    redge[tot]=Edge(u,rhead[v]);
    head[u]=rhead[v]=tot++;
}
void dfs(int u){
    vis[u]=true;
    for(int i=head[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        if(!vis[v])dfs(v);
    }
    stk[++top]=u;
}
void rdfs(int u,int k){
    vis[u]=true;
    Belong[u]=k;
    for(int i=rhead[u];~i;i=redge[i].nxt){
        int v=redge[i].v;
        if(!vis[v])rdfs(v,k);
    }
}
void kscc(int V){
    scc=top=0;
    clr(vis,0);
    rep(i,0,V-1)if(!vis[i])dfs(i);
    clr(vis,0);
    per(i,top,1){
        int v=stk[i];
        if(!vis[v])rdfs(v,++scc);
    }
}
void add_clause(int xv,int x,int yv,int y){
    x=x<<1|xv;
    y=y<<1|yv;
    add_edge(x^1,y);
    add_edge(y^1,x);
}
void add_var(int xv,int x){
    x=x<<1|xv;
    add_edge(x^1,x);
}
int st[MAXV],ed[MAXV],d[MAXV];
char tm[10];
int fun(){
    int res=0;
    int h=(tm[0]-'0')*10+tm[1]-'0';
    res=h*60;
    res+=(tm[3]-'0')*10+tm[4]-'0';
    return res;
}
int work(){
    int n;
    scanf("%d",&n);
    rep(i,0,n-1){
        scanf("%s",tm);
        st[i]=fun();
        scanf("%s",tm);
        ed[i]=fun();
        scanf("%d",&d[i]);
    }
    init();
    rep(i,0,n-1){
        rep(j,0,i-1){
            if(min(st[i]+d[i],st[j]+d[j])>max(st[i],st[j])){
                add_clause(0,i,0,j);
            }
            if(min(st[i]+d[i],ed[j])>max(st[i],ed[j]-d[j])){
                add_clause(0,i,1,j);
            }
            if(min(ed[i],st[j]+d[j])>max(ed[i]-d[i],st[j])){
                add_clause(1,i,0,j);
            }
            if(min(ed[i],ed[j])>max(ed[i]-d[i],ed[j]-d[j])){
                add_clause(1,i,1,j);
            }
        }
    }
    kscc(2*n);
    rep(i,0,n-1){
        if(Belong[i<<1]==Belong[i<<1|1]){
            puts("NO");
            return 0;
        }
    }
    puts("YES");
    rep(i,0,n-1){
        if(Belong[i<<1|1]>Belong[i<<1]){
            printf("%02d:%02d %02d:%02d\n",st[i]/60,st[i]%60,(st[i]+d[i])/60,(st[i]+d[i
    ])%60);
        }else{
            printf("%02d:%02d %02d:%02d\n",(ed[i]-d[i])/60,(ed[i]-d[i])%60,ed[i]/60,ed[
    i]%60);
        }
    }
    return 0;
}
```

4.6.5 cut-vertex

```
//poj 1144
#include<cstdio>
#include<cstring>
#include<algorithm>
using namespace std;
#define rep(i,a,b) for(int i=a;i<=b;i++)
#define clr(a,x) memset(a,x,sizeof(a))
const int MAXV=105;
const int MAXE=1e5;
int tot;
int head[MAXV];
struct Edge{
    int v,nxt;
    Edge(){}
    Edge(int v,int nxt):v(v),nxt(nxt){}
}edge[MAXE<<1];
void init(){
    tot=0;
    clr(head,-1);
}
```

```
void add_edge(int u,int v){
    edge[tot]=Edge(v,head[u]);
    head[u]=tot++;
}
int n;
bool is_cut[MAXV];
int low[MAXV],pre[MAXV];
int dfs_clock;
int dfs(int u,int fa){
    int lowu=pre[u]=++dfs_clock;
    int child=0;
    for(int i=head[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        if(!pre[v]){
            child++;
            int lowv=dfs(v,u);
            lowu=min(lowu,lowv);
            if(lowv>=pre[u]){
                is_cut[u]=true;
            }
        }else if(pre[v]<pre[u]&&v!=fa){
            lowu=min(lowu,pre[v]);
        }
    }
    if(fa<0&&child==1)is_cut[u]=false;
    low[u]=lowu;
    return lowu;
}
int main(){
    while(scanf("%d",&n),n){
        init();
        int x;
        while(scanf("%d",&x),x){
            int y;
            while(getchar()!='\n'){
                scanf("%d",&y);
                add_edge(x,y);
                add_edge(y,x);
            }
        }
        clr(is_cut,0);
        clr(low,0);
        clr(pre,0);
        dfs_clock=0;
        int cnt=0;
        dfs(1,-1);
        for(int i=1;i<=n;i++){
            if(is_cut[i])cnt++;
        }
        printf("%d\n",cnt);
    }
    return 0;
}
```

4.7 Bipartite Graph Matching

4.7.1 Hungry

```
//poj3041
```

```
const int MAXV=1e3+5;
struct BM{
    int V;
    vi G[MAXV];
    int match[MAXV];
    bool vis[MAXV];
    void init(int x){
        V=x;
        rep(i,1,V)G[i].clear();
    }
    void add_edge(int u,int v){
        G[u].pb(v);
        G[v].pb(u);
    }
    bool dfs(int u){
        vis[u]=true;
        for(int i=0;i<(int)G[u].size();i++){
            int v=G[u][i];
            int w=match[v];
            if(w==-1||(!vis[w]&&dfs(w))){
                match[u]=v;
                match[v]=u;
                return true;
            }
        }
        return false;
    }
    int matching(){
        int ret=0;
        clr(match,-1);
        rep(i,1,V){
            if(match[i]==-1){
                clr(vis,0);
                if(dfs(i))ret++;
            }
        }
        return ret;
    }
}bm;
int work(){
    int n,k;
    scanf("%d%d",&n,&k);
    bm.init(2*n);
    while(k--){
        int u,v;
        scanf("%d%d",&u,&v);
        bm.add_edge(u,n+v);
    }
    printf("%d",bm.matching());
    return 0;
}
```

4.8 Network Flow

4.8.1 Dinic

```
//poj 3281
#include<cstdio>
#include<iostream>
```

```
#include<algorithm>
#include<cstring>
#include<queue>
using namespace std;
#define clr(a,x) memset(a,x,sizeof(a))
const int MAXV=400+5;
const int MAXE=1e5+5;
const int INF=0x3f3f3f3f;
int tot;
int head[MAXV],level[MAXV],iter[MAXV];
struct Edge{
    int v,cap,nxt;
    Edge(){}
    Edge(int v,int cap,int nxt):v(v),cap(cap),nxt(nxt){}
}edge[MAXE<<1];
void init(){
    tot=0;
    clr(head,-1);
}
void add_edge(int u,int v,int c){
    edge[tot]=Edge(v,c,head[u]);
    head[u]=tot++;
    edge[tot]=Edge(u,0,head[v]);
    head[v]=tot++;
}
void bfs(int s){
    clr(level,-1);
    level[s]=0;
    queue<int> que;
    que.push(s);
    while(!que.empty()){
        int u=que.front();
        que.pop();
        for(int i=head[u];~i;i=edge[i].nxt){
            int v=edge[i].v;
            int c=edge[i].cap;
            if(c>0&&level[v]<0){
                level[v]=level[u]+1;
                que.push(v);
            }
        }
    }
}
int dfs(int u,int t,int f){
    if(u==t)return f;
    for(int &i=iter[u];~i;i=edge[i].nxt){
        int v=edge[i].v;
        int c=edge[i].cap;
        if(c>0&&level[u]<level[v]){
            int d=dfs(v,t,min(f,c));
            if(d>0){
                edge[i].cap-=d;
                edge[i^1].cap+=d;
                return d;
            }
        }
    }
    return 0;
}
```

```
int max_flow(int s,int t){
    int flow=0;
    while(1){
        bfs(s);
        if(level[t]<0)return flow;
        int f;
        memcpy(iter,head,sizeof(head));
        while(f=dfs(s,t,INF))flow+=f;
    }
}
int main(){
    int n,f,d;
    scanf("%d%d%d",&n,&f,&d);
    int s=0,t=2*n+f+d;
    init();
    for(int i=1;i<=f;i++){
        add_edge(s,2*n+i,1);
    }
    for(int i=1;i<=d;i++){
        add_edge(2*n+f+i,t,1);
    }
    for(int i=1;i<=n;i++){
        add_edge(i,n+i,1);
        int ff,dd;
        scanf("%d%d",&ff,&dd);
        while(ff--){
            int x;
            scanf("%d",&x);
            add_edge(2*n+x,i,1);
        }
        while(dd--){
            int x;
            scanf("%d",&x);
            add_edge(n+i,2*n+f+x,1);
        }
    }
    printf("%d",max_flow(s,t));
    return 0;
}
```

4.8.2 MinCost MaxFlow

```
// poj2135
#include<cstdio>
#include<vector>
#include<algorithm>
#include<queue>
using namespace std;
const int MAXV=1005;
const int MAXE=50000;
const int INF=100000000;
typedef pair<int,int> P;
struct edge{int to,cap,cost,rev;};
int dist[MAXV],h[MAXV],prevv[MAXV],preve[MAXV];
int V;
vector<edge> G[MAXV];
void add_edge(int from,int to,int cap,int cost){
    G[from].push_back((edge){to,cap,cost,G[to].size()});
    G[to].push_back((edge){from,0,-cost,G[from].size()-1});
}
int min_cost_flow(int s,int t,int f){
    int res=0;
    fill(h,h+V,0);
    while(f>0){
        priority_queue<P,vector<P>,greater<P> >que;
        fill(dist,dist+V,INF);
        dist[s]=0;
        que.push(P(0,s));
        while(!que.empty()){
            P p=que.top(); que.pop();
            int v=p.second;
            if(dist[v]<p.first) continue;
            for(int i=0;i<G[v].size();i++){
                edge &e=G[v][i];
                if(e.cap>0&&dist[e.to]>dist[v]+e.cost+h[v]-h[e.to]){
                    dist[e.to]=dist[v]+e.cost+h[v]-h[e.to];
                    prevv[e.to]=v;
                    preve[e.to]=i;
                    que.push(P(dist[e.to],e.to));
                }
            }
        }
        if(dist[t]==INF){
            return -1;
        }
        for(int v=0;v<V;v++) h[v]+=dist[v];
        int d=f;
        for(int v=t;v!=s;v=prevv[v]){
            d=min(d,G[prevv[v]][preve[v]].cap);
        }
        f-=d;
        res+=d*h[t];
        for(int v=t;v!=s;v=prevv[v]){
            edge &e=G[prevv[v]][preve[v]];
            e.cap-=d;
            G[v][e.rev].cap+=d;
        }
    }
    return res;
}
int main(){
    int N,M;
    scanf("%d%d",&N,&M);
    V=N;
    for(int i=1;i<=M;i++){
        int x,y,z;
        scanf("%d%d%d",&x,&y,&z);
        add_edge(x-1,y-1,1,z);
        add_edge(y-1,x-1,1,z);
    }
    printf("%d",min_cost_flow(0,N-1,2));
    return 0;
}
```

5 Others

5.1 Matrix

5.1.1 Matrix FastPow

```
typedef vector<ll> vec;
typedef vector<vec> mat;
mat mul(mat& A, mat& B)
{
    mat C(A.size(), vec(B[0].size()));
    for (int i = 0; i < A.size(); i++)
        for (int k = 0; k < B.size(); k++)
            if (A[i][k]) // □□□□□□□□
                for (int j = 0; j < B[0].size(); j++)
                    C[i][j] = (C[i][j] + A[i][k] * B[k][j]) % mod;
    return C;
}
mat Pow(mat A, ll n)
{
    mat B(A.size(), vec(A.size()));
    for (int i = 0; i < A.size(); i++) B[i][i] = 1;
    for (; n; n >>= 1, A = mul(A, A))
        if (n & 1) B = mul(B, A);
    return B;
}
```

5.2 Tricks

5.2.1 Stack-Overflow

```
#pragma comment(linker, "/STACK:1024000000,1024000000")
```

5.2.2 Fast-Scanner

```
template <class T>
inline bool scan_d(T &ret){
    char c;
    int sgn;
    if (c = getchar(), c == EOF) return 0; //EOF
    while (c != '-' && (c < '0' || c > '9')) c = getchar();
    sgn = (c == '-') ? -1 : 1;
    ret = (c == '-') ? 0 : (c - '0');
    while (c = getchar(), c >= '0' && c <= '9') ret = ret * 10 + (c - '0');
    ret *= sgn;
    return 1;
}
inline void out(int x){
    if(x<0){
        putchar('-');
        x=-x;
    }
    if (x > 9) out(x / 10);
    putchar(x % 10 + '0');
}
```

5.2.3 Strok-Sscanf

```
// get some integers in a line
gets(buf);
int v;
char *p = strtok(buf, " ");
while (p){
    sscanf(p, "%d", &v);
    p = strtok(NULL," ");
}
```

5.3 Mo Algorithm

```
//hdu 6333
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;
const int MAXN=1e5+10;
const int MOD=1e9+7;
int block;
struct node{
    int l,r,id;
}no[MAXN];
bool cmp(node x,node y){
    if(x.l/block==y.l/block)return x.r<y.r;
    else return x.l/block<y.l/block;
}
int ans[MAXN];
int fact[MAXN];
int invfact[MAXN];
ll pow_mod(ll a,ll b){
    ll res=1;
    while(b){
        if(b&1)res=res*a%MOD;
        a=a*a%MOD;
        b>>=1;
    }
    return res;
}
ll fun(ll n,ll m){
    return (1LL*fact[n]*invfact[m])%MOD*invfact[n-m]%MOD;
}
int main(){
    int n=100000;
    fact[0]=1;
    for(int i=1;i<=n;i++){
        fact[i]=1LL*fact[i-1]*i%MOD;
    }
    invfact[n]=pow_mod(fact[n],MOD-2);
    for(int i=n;i>=1;i--){
        invfact[i-1]=1LL*invfact[i]*i%MOD;
    }
    int q;
    scanf("%d",&q);
    block=(int)sqrt(100000);
    for(int i=1;i<=q;i++){
        scanf("%d%d",&no[i].r,&no[i].l);
        no[i].id=i;
    }
    sort(no+1,no+1+q,cmp);
    int L=1,R=1;
    ll now=2;
    int inv2=pow_mod(2,MOD-2);
    for(int i=1;i<=q;i++){
        while(R<no[i].r){
            now=(now*2-fun(R,L)+MOD)%MOD;
            R++;
        }
        while(L>no[i].l){
            now=(now-fun(R,L)+MOD)%MOD;
            L--;
        }
        while(R>no[i].r){
            R--;
            now+=fun(R,L);
            now%=MOD;
            now=now*inv2%MOD;
        }
        while(L<no[i].l){
            L++;
            now=(now+fun(R,L))%MOD;
        }
        ans[no[i].id]=now;
    }
    for(int i=1;i<=q;i++){
        printf("%d\n",ans[i]);
    }
    return 0;
}
```

5.4 BigNum

5.4.1 High-precision

```
import java.io.*;
import java.math.*;
import java.util.StringTokenizer;

public class Main{
    public static void main(String[] args){
        InputStream inputStream = System.in;//new FileInputStream("C:\\Users\\xxx\\Downloads\\test.in");
        OutputStream outputStream = System.out;
        InputReader in = new InputReader(inputStream);
        PrintWriter out = new PrintWriter(outputStream);
        Task solver = new Task();
        solver.solve(in, out);
        out.close();
    }
    static class Task {

        public void solve(InputReader in, PrintWriter out) {
            //do sth

        }

    }
    static class InputReader {
        public BufferedReader reader;
        public StringTokenizer tokenizer;

        public InputReader(InputStream stream) {
            reader = new BufferedReader(new InputStreamReader(stream), 32768);
            tokenizer = null;
        }

        public String next() {
            while (tokenizer == null || !tokenizer.hasMoreTokens()) {
                try {
                    tokenizer = new StringTokenizer(reader.readLine());
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }
            return tokenizer.nextToken();
        }

        public int nextInt() {
            return Integer.parseInt(next());
        }

        public long nextLong() {
            return Long.parseLong(next());
        }

        public double nextDouble() {
            return Double.parseDouble(next());
        }

        public char[] nextCharArray() {
            return next().toCharArray();
        }

        public boolean hasNext() {
            try {
                String string = reader.readLine();
                if (string == null) {
                    return false;
                }
                tokenizer = new StringTokenizer(string);
                return tokenizer.hasMoreTokens();
            } catch(IOException e) {
                return false;
            }
        }
        public BigInteger nextBigInteger() {
            return new BigInteger(next());
        }

        public BigDecimal nextBigDecinal() {
            return new BigDecimal(next());
        }
    }
}
```

5.5 VIM

```
syntax on
set nu
set tabstop=4
set expandtab
set autoindent
set cin
set mouse=a

map<F2> :call SetTitle()<CR>
func SetTitle()
let l = 0
let l = l + 1 | call setline(l,'#include <algorithm>')
let l = l + 1 | call setline(l,'#include  <iostream>')
let l = l + 1 | call setline(l,'#include   <cstring>')
let l = l + 1 | call setline(l,'#include    <string>')
let l = l + 1 | call setline(l,'#include    <cstdio>')
let l = l + 1 | call setline(l,'#include    <vector>')
let l = l + 1 | call setline(l,'#include     <stack>')
let l = l + 1 | call setline(l,'#include     <queue>')
let l = l + 1 | call setline(l,'#include     <cmath>')
let l = l + 1 | call setline(l,'#include       <set>')
let l = l + 1 | call setline(l,'#include       <map>')
let l = l + 1 | call setline(l,'using namespace std;')
let l = l + 1 | call setline(l,'#define rep(i,a,b) for(int i=a;i<=b;i++)')
let l = l + 1 | call setline(l,'#define per(i,a,b) for(int i=a;i>=b;i--)')
let l = l + 1 | call setline(l,'#define clr(a,x) memset(a,x,sizeof(a))')
let l = l + 1 | call setline(l,'#define pb push_back')
let l = l + 1 | call setline(l,'#define mp make_pair')
let l = l + 1 | call setline(l,'#define all(x) (x).begin(),(x).end()')
let l = l + 1 | call setline(l,'#define fi first')
let l = l + 1 | call setline(l,'#define se second')
let l = l + 1 | call setline(l,'#define SZ(x) ((int)(x).size())')
let l = l + 1 | call setline(l,'typedef unsigned long long ull;')
let l = l + 1 | call setline(l,'typedef long long ll;')
let l = l + 1 | call setline(l,'typedef vector<int> vi;')
let l = l + 1 | call setline(l,'typedef pair<int,int> pii;')
let l = l + 1 | call setline(l,'/*************head******************/')
let l = l + 1 | call setline(l,'int work(){')
let l = l + 1 | call setline(l,'')
let l = l + 1 | call setline(l,'    return 0;')
let l = l + 1 | call setline(l,'}')
let l = l + 1 | call setline(l,'int main(){')
let l = l + 1 | call setline(l,'#ifdef superkunn')
let l = l + 1 | call setline(l,'    freopen("input.txt","rt",stdin);')
let l = l + 1 | call setline(l,'#endif')
let l = l + 1 | call setline(l,'    work();')
let l = l + 1 | call setline(l,'    return 0;')
let l = l + 1 | call setline(l,'}')
endfunc
```

5.6 BASH

```
g++ -g -Wall -std=c++11 -Dsuperkunn main.cpp
./a.out
```

6 Geometry

```
struct Point{
    double x,y;
    Point(double x=0,double y=0):x(x),y(y){}
};
typedef Point Vector;
Vector operator + (Vector A,Vector B){return Vector(A.x+B.x,A.y+B.y);}
Vector operator - (Point A,Point B){return Vector(A.x-B.x,A.y-B.y);}
Vector operator * (Vector A,double p){return Vector(A.x*p,A.y*p);}
Vector operator / (Vector A,double p){return Vector(A.x/p,A.y/p);}
bool operator < (const Point& a,const Point &b){
    return a.x<b.x||(a.x==b.x&&a.y<b.y);
}
const double eps = 1e-10;
int dcmp(double x){
    if(fabs(x)<eps)return 0;else return x<0?-1:1;
}
bool operator == (const Point& a,const Point &b){
    return dcmp(a.x-b.x)==0&&dcmp(a.y-b.y)==0;
}
//(x,y)->  atan2(y,x)
double Dot(Vector A,Vector B){return A.x*B.x+A.y*B.y;}
double Length(Vector A){return sqrt(Dot(A,A));}
double Angle(Vector A,Vector B){return acos(Dot(A,B)/Length(A)/Length(B));}
double Cross(Vector A,Vector B){return A.x*B.y-A.y*B.x;}
double Area2(Point A,Point B,Point C){return Cross(B-A,C-A);}
Vector Rotate(Vector A,double rad){
    return Vector(A.x*cos(rad)-A.y*sin(rad),A.x*sin(rad)+A.y*cos(rad));
}
Vector Normal(Vector A){
    double L=Length(A);
    return Vector(-A.y/L,A.x/L);
}
```