目录

数论

(ex)CRT((扩展)中国剩余定理)

```cpp
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
#include <numeric>
using namespace std;
#define int __int128
// 用于存储 __int128 的字符串表示
std::string to_string(__int128 value) {
    std::string result;
    bool isNegative = value < 0;
    value = isNegative ? -value : value;

    do {
        result.push_back(static_cast<char>(value % 10) + '0');
        value /= 10;
    } while (value > 0);

    if (isNegative) {
        result.push_back('-');
    }

    std::reverse(result.begin(), result.end());
    return result;
}

// 从字符串转换为 __int128
__int128 to_int128(const std::string& str) {
    __int128 result = 0;
    bool isNegative = str[0] == '-';
    size_t start = isNegative ? 1 : 0;
```

```
    for (size_t i = start; i < str.size(); ++i) {
        result = result * 10 + (str[i] - '0');
    }

    return isNegative ? -result : result;
}

// 重载 >> 操作符以支持 __int128 输入
std::istream& operator>>(std::istream& in, __int128& value) {
    std::string str;
    in >> str;
    value = to_int128(str);
    return in;
}

// 重载 << 操作符以支持 __int128 输出
std::ostream& operator<<(std::ostream& out, __int128 value) {
    out << to_string(value);
    return out;
}
class exCRT{

    public:
        exCRT(vector<int> r,vector<int> m){
            this->r=r;
            this->m=m;
            n=r.size();
        }
        vector<int> r,m;
        int x,n;
        int exgcd(int a,int b,int &x,int &y)//扩展欧几里得
        {
            if(b==0)
            {
                x=1;y=0;
                return a;
            }
            int d=exgcd(b,a%b,x,y),t=x;
            x=y;y=t-a/b*y;
            return d;
        }
        int CRT()
        {
            int mul=accumulate(m.begin(),m.end(),1LL,
            [](int a,int b){return a*b;}),ans=0;
            for(int i=0;i<n;i++)
            {
                int M=mul/m[i],b,y;
```

```
            exgcd(M,m[i],b,y);
            ans=(ans+r[i]*M%mul*b%mul+mul)%mul;
        }
        return (ans%mul+mul)%mul;
    }
    int _exCRT()
    {
        int M=m[0],ans=r[0];
        for(int i=1;i<n;i++)
        {
            int a=M,b=m[i];
            int c=((r[i]-ans)%b+b)%b;
            int x,y;
            int gcd=exgcd(a,b,x,y);
            int bg=b/gcd;
            if(c%gcd!=0) return -1;
            x=(x%bg+bg)%bg;
            x=(x*c/gcd%bg+bg)%bg;
            ans+=x*M;
            M*=bg;
            ans=(ans%M+M)%M;
        }
        return (ans%M+M)%M;
    }

};
signed main()
{
    int T_start=clock();
    int n;cin>>n;
    vector<int> r(n),m(n);
    for(int i=0;i<n;i++) cin>>m[i]>>r[i];
    exCRT ex(r,m);
    cout<<ex.CRT()<<endl;
    return 0;
}
//exCRT 求解同余方程组
//形式：x≡a1(mod m1),x≡a2(mod m2),...,x≡ak(mod mk)
//其中m1,m2,...,mk 互质(CRT)
//其中m1,m2,...,mk 不互质(_exCRT)
//时间复杂度：O(nln(amax))
```

乘法逆元

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
using namespace std;
int ModQpow(int a,int b,int m)//快速幂
{
    int ans=1;
    while(b)
    {
        if(b&1) ans=ans*a%m;
        a=a*a%m;b>>=1;
    }
    return ans;
}
int invMod1(int a,int m)//a 在模 m 意义下的逆元（费马小定理，m 为质数），即 a
^(m-2)
{
    return ModQpow(a,m-2,m);
}
int exgcd(int a,int b,int &x,int &y)//扩展欧几里得
{
    if(b==0)
    {
        x=1;y=0;
        return a;
    }
    int d=exgcd(b,a%b,x,y),t=x;
    x=y;y=t-a/b*y;
    return d;
}
int invMod2(int a,int m)//a 在模 m 意义下的逆元（扩展欧几里得）
{
    int x,y;
    exgcd(a,m,x,y);
```

```
    return (x%m+m)%m;
}
int main()
{
    int T_start=clock();

    return 0;
}
```

矩阵快速幂

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
#include <numeric>
using namespace std;
#define int long long
const int mod=1e9+7;
class MaTQpow{
      public:
            vector<vector<int>> mat;
            int _mod;
            MaTQpow(vector<vector<int>> _mat,int mod):mat(_mat),_mod(mo
d){}
            MaTQpow(int n,int mod):_mod(mod)
            {
                  mat.resize(n,vector<int>(n,0));
                  for(int i=0;i<n;i++) mat[i][i]=1;
                  _mod=mod;
            }
            MaTQpow operator*(const MaTQpow& other)const{
                  vector<vector<int>> res(mat.size(),vector<int>(other.
mat[0].size(),0));
                  for(int i=0;i<mat.size();i++)
                  {
                        for(int j=0;j<other.mat[0].size();j++)
                        {
                              for(int k=0;k<other.mat.size();k++)
                              {
                                    res[i][j]=(res[i][j]+mat[i][k]*othe
r.mat[k][j]&_mod)%_mod;
                              }
                        }
                  }
                  return MaTQpow(res,_mod);
```

```
        }
        MaTQpow Qpow(int n){
            MaTQpow res(mat.size(),_mod);
            MaTQpow base(mat,_mod);
            while(n)
            {
                if(n&1) res=res*base;
                base=base*base;
                n>>=1;
            }
            return res;
        }
        vector<vector<int>> get(int n){
            return Qpow(n).mat;
        }
};
signed main()
{
    int T_start=clock();
    //freopen("in.txt","r",stdin);
    int n;cin>>n;
    vector<int> a(n+1);
    for(int i=1;i<=n;i++) cin>>a[i];
    int c,m,k,t;cin>>c>>m>>k>>t;c%=m;
    vector<int> dp(m,0);
    for(int i=1;i<=n;i++)
    {
        vector<int> ndp(m,0);
        ndp[a[i]%m]=1;
        for(int j=0;j<m;j++)
        {
            if(dp[j])
            {
                ndp[(j+a[i])%m]=(ndp[(j+a[i])%m]+dp[j])%mod;
            }
        }
        for(int j=0;j<m;j++) dp[j]=(dp[j]+ndp[j])%mod;
    }
    vector<vector<int>> p(m,vector<int>(m,0));
    for(int i=0;i<m;i++)
    {
        for(int j=0;j<m;j++)
        {
            p[i][(i*j)%m]=(p[i][(i*j)%m]+dp[j])%mod;
        }
    }
    MaTQpow mat(p,mod);
    auto res=mat.Qpow(t);
    cout<<res.mat[c][k]<<endl;
```

```
        return 0;
}
//矩阵快速幂：处理快速形式变换
//时间复杂度：O(n^3logk)
```

筛法求积性函数

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
#include <numeric>
#include <functional>
#include <ranges>
#include <iomanip>
//#define int long long //赫赫 要不要龙龙呢
using namespace std;
vector<int> primes(int n)
{
    //积性函数gcd(a,b)=1,f(ab)=f(a)f(b)
    //当f(p^k),p 为质数时时的函数值可以快速求出,
    //即可以通过递推求出所有积性函数
    vector<int> primes;
    vector<bool>v(n+1,0);
    for(int i=2;i<=n;i++)
    {
        if(!v[i])
        {
            primes.push_back(i);
            //考虑f(p)=...
            //单个质数的情况
        }
        for(int j=0;j<primes.size()&&primes[j]*i<=n;j++)
        {
            v[primes[j]*i]=1;
            int m=primes[j]*i;
            if(i%primes[j]==0)
            {
                //此时p[j]是m 的最小质因子,运用反证法p[j]也是m 的最小质因
子
                //考虑f(m)=...
```

```
                //多个质因子的情况
                break;
            }
            else{
                //此时gcd(i,pj)=1,f(m)=f(i)f(pj)
                //新增质因子的情况
            }
        }
    }
    return primes;
}
signed main()
{
    int T_start=clock();
    //freopen("in.txt","r",stdin);
    //freopen("out.txt","w",stdout);
    //ios::sync_with_stdio(false),cin.tie(0),cout.tie(0);
    return 0;
}
```

数论预处理

```cpp
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
#include <numeric>
#include <functional>
#include <ranges>
#include <iomanip>
#define int long long //赫赫 要不要龙龙呢
using namespace std;
class Pre{
public:
    int n,tag;
    const int mod=1e9+7;
    vector<int> inv,fac,invfac;
    Pre(int n):n(n){}
    Pre(int n,int tag):n(n),tag(tag){
        inv.resize(n+1);
        fac.resize(n+1);
        invfac.resize(n+1);
        preC();
    }
    int ModQpow(int a,int b,int m)//快速幂
    {
        int ans=1;
        while(b)
        {
            if(b&1)ans=ans*a%m;
            a=a*a%m,b>>=1;
        }
        return ans;
    }
    //O(nlnn)求1-n所有数的约数
    vector<vector<int>> divs()
    {
```

```
    vector<vector<int>> ans(n+1);
    for(int i=1;i<=n;i++)
    {
        for(int j=i;j<=n;j+=i)
        {
            ans[j].push_back(i);
        }
    }
    return ans;
}
//O(n)求1-n 所有的质数
vector<int> primes()
{
    vector<int> primes;
    vector<bool>v(n+1,0);
    for(int i=2;i<=n;i++)
    {
        if(!v[i])primes.push_back(i);
        for(int j=0;j<primes.size()&&primes[j]*i<=n;j++)
        {
            v[primes[j]*i]=1;
            if(i%primes[j]==0)break;
        }
    }
    return primes;
}
//O(n)求0-n 的阶乘和阶乘逆元
void preC()
{
    inv[1]=1;
    for(int i=2;i<=n;i++)
    {
        inv[i]=(mod-mod/i)*inv[mod%i]%mod;
    }
    fac[0]=invfac[0]=1;
    for(int i=1;i<=n;i++)
    {
        fac[i]=fac[i-1]*i%mod;
        invfac[i]=invfac[i-1]*inv[i]%mod;
    }
}
//组合数C(n,m) n 在上 m 在下
int C(int n,int m)
{
    if(n<0||m<0||n<m)return 0;
    return fac[n]*invfac[m]%mod*invfac[n-m]%mod;
}
//O(n)求1-n 的欧拉函数
vector<int> euler()
```

```
{
    vector<int> phi(n+1);
    phi[1]=1;
    vector<int> primes;
    vector<bool>v(n+1,0);
    for(int i=2;i<=n;i++)
    {
        if(!v[i])primes.push_back(i),phi[i]=i-1;
        for(int j=0;j<primes.size()&&primes[j]*i<=n;j++)
        {
            int m=primes[j]*i;
            v[m]=1;
            if(i%primes[j]==0)
            {
                phi[m]=phi[i]*primes[j];
                break;
            }
            else phi[m]=phi[i]*(primes[j]-1);
        }
    }
    return phi;
}
//O(n)求1-n的约数个数
vector<int> d()
{
    vector<int> a(n+1),d(n+1);
    vector<int> primes;
    vector<bool>v(n+1,0);
    for(int i=2;i<=n;i++)
    {
        if(!v[i])
        {
            primes.push_back(i);
            a[i]=1,d[i]=2;
        }
        for(int j=0;j<primes.size()&&primes[j]*i<=n;j++)
        {
            int m=primes[j]*i;
            v[m]=1;
            if(i%primes[j]==0)
            {
                a[m]=a[i]+1;
                d[m]=d[i]/(a[i]+1)*(a[m]+1);
                break;
            }
            else
            {
                a[m]=1;
                d[m]=d[i]*2;
```

```
            }
        }
    }
    return d;
}
//O(n)求1-n 的约数和
vector<int> sumd()
{
    vector<int> g(n+1),f(n+1);
    vector<int> primes;
    vector<bool>v(n+1,0);
    g[1]=f[1]=1;
    for(int i=2;i<=n;i++)
    {
        if(!v[i])
        {
            primes.push_back(i);
            f[i]=g[i]=i+1;
        }
        for(int j=0;j<primes.size()&&primes[j]*i<=n;j++)
        {
            int m=primes[j]*i;
            v[m]=1;
            if(i%primes[j]==0)
            {
                g[m]=g[i]*primes[j]+1;
                f[m]=f[i]*g[m]/g[i];
                break;
            }
            else
            {
                g[m]=primes[j]+1;
                f[m]=f[i]*g[m];
            }
        }
    }
    return f;
}
//O(n)求1-n 的莫比乌斯函数
vector<int> mu()
{
    vector<int> mu(n+1);
    mu[1]=1;
    vector<int> primes;
    vector<bool>v(n+1,0);
    for(int i=2;i<=n;i++)
    {
        if(!v[i])primes.push_back(i),mu[i]=-1;
        for(int j=0;j<primes.size()&&primes[j]*i<=n;j++)
```

```
            {
                int m=primes[j]*i;
                v[m]=1;
                if(i%primes[j]==0)
                {
                    mu[m]=0;
                    break;
                }
                else mu[m]=-mu[i];
            }
        }
        return mu;
    }
};
signed main()
{
    int T_start=clock();
    //freopen("in.txt","r",stdin);
    //freopen("out.txt","w",stdout);
    //ios::sync_with_stdio(false),cin.tie(0),cout.tie(0);

    return 0;
}
```

整除分块

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#define int long long
using namespace std;
struct blocknode{
      int l;
      int r;
      int val;
};
//对[l,r]的i,floor(n/i)相等
//n%i=n-i*floor(n/i)
//首项n-l*val 公差-val 项数r-l+1
class divb{
public:
    struct node{
        int l,r;
        int val1,val2;
    };
    int s,e;
    vector<node> a;
    divb(int s,int e):s(s),e(e){}
    void b1(int n,int m){
        for(int l=s,r;l<=e;l=r+1)
        {
            r=min(n/(n/l),m/(m/l));
            a.push_back({l,r,n/l,m/l});
        }
    }
    void b2(int n)
    {
        for(int l=s,r;l<=e;l=r+1)
        {
            r=n/(n/l);
            a.push_back({l,r,n/l,0});
        }
```

```
    }
};
signed main()
{
	int T_start=clock();
	int n,k;cin>>n>>k;
	vector<blocknode>a;
	for(int l=1,r;l<=n;l=r+1)
	{
		blocknode tp;
		r=min(n/(n/l),n);
		tp.l=l;tp.r=r;
		tp.val=n/l;
		a.push_back(tp);
	}
	//for(auto i:a)
	//{
		//cout<<i.l<<' '<<i.r<<' '<<i.val<<endl;
	//}
	return 0;
}
//整除分块：处理连续区间内满足条件的元素个数
//时间复杂度：O(n^(1/2))
```

数据结构

BIT

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
using namespace std;
class BIT{
    private:
        int n;
        vector<int> tree;//tree[i] 是[i-lowbit(i)+1,i]的和,[1,n]存储
        int lowbit(int x){
            return x&(-x);
        }
    public:
        BIT(int n): n(n),tree(n+1,0){}
        void update(int i,int val)//单点修改 a[i]+=val
        {
            while(i<=n){
                tree[i]+=val;
                i+=lowbit(i);//跳到后一个 lowbit(x)的位置
            }
        }
        int query(int l,int r)//区间查询 [l,r]的和
        {
            int res=0;
            while(r>=l){
                res+=tree[r];
                r-=lowbit(r);//跳到前一个 lowbit(x)的位置
            }
            return res;
        }
        int query_diff(int i)//单点查询 a_diff[i] (维护差分数组)=sum[1,i]
        {
            return query(1,i);
        }
```

```
        void update_diff(int l,int r,int val)//区间修改（维护差分数组）a_
diff[l]+=val,a_diff[r+1]-=val
        {
            update(l,val);
            update(r+1,-val);
        }
        void init(vector<int> a)//初始化
        {
            vector<int> presum(a.size()+1,0);
            for(int i=1;i<=a.size();i++)
            {
                presum[i]=presum[i-1]+a[i-1];
                tree[i]=presum[i]-presum[i-lowbit(i)];//按定义
            }
        }
};
int main()
{
    int T_start=clock();
    //test
    int n=10; vector<int> a={1,3,2,4,2,1,5,4,3,2},a_diff={1,2,-1,2,-2,-
1,4,-1,-1,-1};//a_diff[i]=a[i]-a[i-1]
    BIT bit(n),bit_diff(n);
    bit.init(a);bit_diff.init(a_diff);
    cout<<bit.query(1,5)<<endl;
    bit.update(1,5);
    cout<<bit.query(1,5)<<endl;
    bit_diff.update_diff(1,5,2);
    cout<<bit_diff.query_diff(5)<<endl;
    cout<<bit_diff.query_diff(6)<<endl;
    //test end
    return 0;
}
//进阶用法1.维护差分数组
//进阶用法2.把数组离散化后按照值域建树状数组，可以用来求逆序对(第K大)
//e.g.val[1,16,9,10,3]->dis[1,5,3,4,2]->bit[1,1,1,1,1]
//    BIT bit(5);
//    //bit.update(1,1);bit.update(2,1);bit.update(3,1);bit.update(4,1);
bit.update(5,1);
//    val_i[1,3]即更新为[1,0,1,0,1] 9即为第2大 即bit.query(1,3)=2
//求逆序对，how to do? 即[1,r]中比a[r]大的数的个数
```

st 表

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
using namespace std;
class st{
    public:
        vector<vector<int>> dp;//dp[i][j]是[i,i+2^j-1]的min/max
        int inf(int a,int b)
        {
            return max(a,b);
        }
        void init(vector<int>& nums,int siz)
        {
            int len=log2(siz)+1;
            dp.resize(siz);
            for(auto &i:dp) i.resize(len);
            for(int i=0;i<siz;i++)
            {
                dp[i][0]=nums[i];
            }
            for(int j=1;j<=len;j++)
            {
                for(int i=0;i+(1<<j)-1<siz;i++)
                {
                    dp[i][j]=inf(dp[i][j-1],dp[i+(1<<(j-1))][j-1]);
                }
            }
        }
        int query(int l,int r)
        {
            int k=log2(r-l+1);
            return inf(dp[l][k],dp[r-(1<<k)+1][k]);
        }
        st(vector<int>& nums){
```

```
            init(nums,nums.size());
        }
};
int read()
{
    int s=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9')
    {
        if(ch=='-') f=-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9')
    {
        s=(s<<3)+(s<<1)+ch-'0';
        ch=getchar();
    }
    return s*f;
}
inline void write(int x)
{
    static int sta[35];
    int top=0;
    if(x<0&&x!=-2147483648) {putchar('-');x=-x;}
    if(x==-2147483648) {printf("-2147483648");return;}
    do{
      sta[top++]=x%10, x/=10;
      }while(x);
    while(top) putchar(sta[--top]+48);
}
int main()
{
    int T_start=clock();
    int n=read(),m=read();
    vector<int> nums(n);
    for(int i=0;i<n;i++)
    {
        nums[i]=read();
    }
    st s(nums);
    for(int i=0;i<m;i++)
    {
        int l=read(),r=read();
        write(s.query(l-1,r-1));
        putchar('\n');
    }
    return 0;
}
```

并查集

```cpp
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
using namespace std;
class DSU{
    public:
        int n;vector<int> fa;
        DSU(int n):n(n)
        {
            srand(time(NULL));
            fa.resize(n+1);
            for(int i=1;i<=n;i++)
            {
                fa[i]=i;
            }
        }
        int find(int u){
            return fa[u]==u?u:fa[u]=find(fa[u]);
        }
        void merge(int a,int b)
        {
            int op=rand()%2;
            if(op==0) fa[find(a)]=find(b);
            else fa[find(b)]=find(a);
        }
};
int main()
{
    int T_start=clock();
    srand(time(NULL));
    return 0;
}
```

FHQTreap

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
using namespace std;
class FHQTreap{
    //无旋Treap：1.满足二叉搜索树性质(val) 2.满足堆性质（优先级）
    //树堆：BST+Heap
    public:
        struct Node{
            int val;
            int size=0;
            int priority;//随机数
            Node *left, *right;
            Node(int val):val(val),priority(rand()),left(NULL),right(NU
LL),size(1){}
            Node(int val,int priority):val(val),priority(priority),left
(NULL),right(NULL),size(1){}
        };
        bool cmp(Node a,Node b){
            return a.val<b.val;
        }
        Node *root;
        FHQTreap():root(NULL){}
        void merge(Node *&root,Node *a,Node *b){
            //val a<=val b(内部满足Treap)
            if(!a)root=b;
            else if(!b)root=a;
            else{
                if(a->priority>b->priority){//a 的优先级大
                    root=a;//a 作为根(为了满足Heap(大))
                    merge(a->right,a->right,b);//b 合并到a 的右子树（为了
满足BST：a 的右子树的所有节点都大于a)
                }
```

```
        else{
            root=b;
            merge(b->left,a,b->left);
        }
    }
    if(root)
    {
        root->size=1;
        if(root->left)root->size+=root->left->size;
        if(root->right)root->size+=root->right->size;
        //cout<<root->val<<' '<<root->size<<endl;
    }
}
void split(Node *root,Node *&a,Node *&b,int val){
    //将 root 按照 val 分割为 a,b 两部分
    //a 的 val 都小于等于 val，b 的 val 都大于 val
    if(!root){
        a=b=NULL;
        return;
    }
    if(root->val<=val){
        a=root;
        split(root->right,a->right,b,val);
    }
    else{
        b=root;
        split(root->left,a,b->left,val);
    }
    if(root)
    {
        root->size=1;
        if(root->left)root->size+=root->left->size;
        if(root->right)root->size+=root->right->size;
        //cout<<root->val<<' '<<root->size<<endl;
    }
}
void insert(int val){
    Node *a,*b;
    split(root,a,b,val);//将 root 按照 val 分割为 a,b 两部分
    merge(a,a,new Node(val));//将 val 插入到 a 中
    merge(root,a,b);//将 a,b 合并为 root
    //偶还能这样
}
void erase(int val){
    Node *a,*b,*c;
    split(root,a,b,val);//将 root 按照 val 分割为 a,b 两部分
    split(a,a,c,val-1);//将 a 按照 val-1 分割为 a,c 两部分
    if(c)
```

```
    {
        merge(a,a,c->right);//将 c 的右子树合并到 a 中(删除一个节点)
        merge(a,a,c->left);//将 c 的左子树合并到 a 中(删除一个节点)
    }
    merge(root,a,b);//将 a,b 合并为 root
}
void print(Node *root){
    if(!root)return;
    print(root->left);
    cout<<root->val<<" ";
    print(root->right);
}
int findMax(Node *root){
    if(!root)return -1;
    while(root->right)root=root->right;
    return root->val;
}
int findMin(Node *root){
    if(!root)return -1;
    while(root->left)root=root->left;
    return root->val;
}
int pre(int val){
    Node *a,*b;
    split(root,a,b,val-1);//将 root 按照 val-1 分割为 a,b 两部分
    int res=findMax(a);
    merge(root,a,b);
    return res;
}
int next(int val){
    Node *a,*b;
    split(root,a,b,val);//将 root 按照 val 分割为 a,b 两部分
    int res=findMin(b);
    merge(root,a,b);
    return res;
}
int rank(int val){
    Node *a,*b;
    split(root,a,b,val-1);//将 root 按照 val-1 分割为 a,b 两部分
    int res=(a?a->size:0)+1;
    merge(root,a,b);
    return res;
}
int QueryKth(int k){
    return KthQuery(root,k);
}
int KthQuery(Node* root,int k){
    if(root==nullptr) return -1;
```

```
            int leftsize=root->left?root->left->size:0;
            if(k<=leftsize) return KthQuery(root->left,k);
            else if(k==leftsize+1) return root->val;
            else return KthQuery(root->right,k-leftsize-1);
        }
        bool find(int val){
            Node *a,*b;
            split(root,a,b,val);//将root按照val分割为a,b两部分
            bool res=a&&findMax(a)==val;
            merge(root,a,b);
            return res;
        }

};
int main()
{
    int T_start=clock();
    FHQTreap treap;
    vector<int> test={1,2,3,4,5,6,7,8,9,10};
    /*for(int i=0;i<test.size();i++){
        treap.insert(test[i]);
    }
    treap.print(treap.root);
    cout<<endl;
    treap.erase(5);
    treap.print(treap.root);
    cout<<endl;
    treap.insert(5);
    treap.insert(5);
    treap.insert(5);
    treap.insert(5);
    treap.print(treap.root);
    cout<<endl;
    treap.erase(5);
    treap.print(treap.root);
    cout<<endl;*/
    cout<<treap.pre(5)<<endl;
    cout<<treap.next(5)<<endl;
    cout<<treap.rank(5)<<endl;
    treap.erase(5);
    treap.print(treap.root);
    cout<<endl;
    treap.insert(5);
    treap.insert(5);
    treap.insert(5);
    treap.print(treap.root);
    cout<<endl;
    cout<<treap.rank(6)<<endl;
    treap.erase(5);
```

```
    treap.print(treap.root);
    cout<<endl;
    cout<<treap.rank(8)<<endl;
    cout<<treap.QueryKth(7)<<endl;
    return 0;
}
```

01trie

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
using namespace std;
class O1Tire{
	public:
		struct node
		{
			node* ch[2];
			int cnt;
			node():ch{nullptr,nullptr},cnt(0){}
		};
		node* root;
		O1Tire():root(new node){}
		void set(int x,int t)//从高到低建树
		{
			node* p=root;
			for(int i=31;i>=0;i--)
			{
				int d=(x>>i)&1;
				if(!p->ch[d])
					p->ch[d]=new node();
				p->ch[d]->cnt+=t;
				p=p->ch[d];
			}
		}
		int findMax(int x)//从高到低找,贪心选择,求解x对tire中所有数
的最大异或值
		{
			node* p=root;
			int res=0;
			for(int i=31;i>=0;i--)
			{
				int d=(x>>i)&1;
```

```
                        if(p->ch[d^1]&&p->ch[d^1]->cnt)
                              p=p->ch[d^1],res+=(1<<i);
                        else
                              p=p->ch[d];
                        if(!p)
                              return res;
                  }
                  return res;
            }

};
int main()
{
      int T_start=clock();
      int t;cin>>t;
      while(t--)
      {
          int n,k;cin>>n>>k;
          vector<int> a(n);
          for(int i=0;i<n;i++)
              cin>>a[i];
            O1Tire tire;
            int ans=0xffffffff;
            for(int i=0,j=0;i<n;i++)
            {
                  tire.set(a[i],1);
                  while(j<=i&&tire.findMax(a[i])>=k)
                  {
                        ans=min(ans,i-j+1);
                        tire.set(a[j],-1);
                        j++;
                  }
            }
            if(ans==0xffffffff) cout<<-1<<endl;
            else cout<<ans<<endl;
      }
      return 0;
}
```

线段树

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
using namespace std;
class SegTree{
    public:
        struct Node
        {
            int sum;
            int s,e;
            int lazy=0;
            Node* lt;
            Node* rt;
            Node(int sum,int s,int e):s(s),e(e),sum(sum),lt(nullptr),rt
(nullptr){}
        };
        Node* root;
        Node* buildtree(vector<int> &nums,int l,int r)
        {
            if(l>r) return nullptr;
            if(l==r) return new Node(nums[l],l,l);
            int mid=(l+r)>>1;
            Node* root=new Node(0,l,r);
            Node* lc=buildtree(nums,l,mid);
            Node* rc=buildtree(nums,mid+1,r);
            if(lc) root->lt=lc,root->sum+=lc->sum;
            if(rc) root->rt=rc,root->sum+=rc->sum;
            return root;
        }
        void init(vector<int>nums)
        {
            root=buildtree(nums,0,nums.size()-1);
            return;
        }
```

```
        void taglazy(Node* root,int val)
        {
            if(root==nullptr) return;
            root->lazy+=val;
            root->sum+=(root->e-root->s+1)*val;
        }
        void pushdown(Node* root)
        {
            if(!root) return ;
            if(root->lazy)
            {
                taglazy(root->lt,root->lazy);
                taglazy(root->rt,root->lazy);
                root->lazy=0;
            }
        }
        void update(Node* root,int l,int r,int val)
        {
            if(!root) return ;
            if(root->s>r||root->e<l) return ;
            if(root->s>=l&&root->e<=r)
            {
                taglazy(root,val);
                return;
            }
            pushdown(root);
            update(root->lt,l,r,val);
            update(root->rt,l,r,val);
            root->sum=((root->lt?root->lt->sum:0)+(root->rt?root->rt->s
um:0));
            return ;
        }
        void update(int l,int r,int val)
        {
            update(root,l,r,val);
            return ;
        }
        int query(Node* root,int l,int r)
        {
            pushdown(root);
            if(!root) return 0;
            if(root->s>r||root->e<l) return 0;
            if(root->s>=l&&root->e<=r) return root->sum;
            return query(root->lt,l,r)+query(root->rt,l,r);
        }
        int query(int l,int r)
        {
            return query(root,l,r);
        }
```

```
};
int main()
{

    return 0;
}
```

case2:max/min

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
using namespace std;
class SegTree{
    public:
        pair<int,int> error={-0x3f3f3f3f,0x3f3f3f3f};
        struct Node{
            int start;int end;
            int high;int low;
            int lazy=0;
            Node* left;
            Node* right;
            Node(int start,int end,int high,int low)
                :start(start),end(end),high(high),
                low(low),left(nullptr),right(nullptr){}
        };
        Node* root;
        Node* Build(vector<int>& nums,int start,int end)
        {
            if(start>end) return nullptr;
            if(start==end) return new Node(start,end,nums[start],nums[s
tart]);
            int mid=(start+end)>>1;
            Node* root=new Node(start,end,-0x3f3f3f3f,0x3f3f3f3f);
            Node* leftchild=Build(nums,start,mid);
            Node* rightchild=Build(nums,mid+1,end);
            if(leftchild)
```

```
    {
        root->left=leftchild;
        root->high=max(root->high,leftchild->high);
        root->low=min(root->low,leftchild->low);
    }
    if(rightchild)
    {
        root->right=rightchild;
        root->high=max(root->high,rightchild->high);
        root->low=min(root->low,rightchild->low);
    }
    return root;
}
void init(vector<int>& nums)
{
    root=Build(nums,0,nums.size()-1);
    return ;
}

void taglazy(Node* root,int val)
{
    if(root==nullptr) return ;
    root->low+=val,root->high+=val;
    root->lazy+=val;
}
void pushdown(Node* root)
{
    if(root==nullptr) return ;
    if(root->lazy!=0)
    {
        taglazy(root->left,root->lazy);
        taglazy(root->right,root->lazy);
        root->lazy=0;
    }
}
void update(Node* root,int l,int r,int val)
{
    if(root==nullptr) return;
    if(root->end<l||root->start>r) return ;
    if(root->start>=l&&root->end<=r)
    {
        taglazy(root,val);
        return;
    }
    pushdown(root);
    update(root->left,l,r,val);
    update(root->right,l,r,val);
    if(root->left)
    {
```

```
                root->high=max(root->left->high,root->high);
                root->low=min(root->left->low,root->low);
            }
            if(root->right)
            {
                root->high=max(root->right->high,root->high);
                root->low=min(root->right->low,root->low);
            }
            return ;
        }
        void update(int l,int r,int val) {update(root,l,r,val);}
        pair<int,int> query(Node* root,int l,int r)
        {
            pushdown(root);
            if(root==nullptr) return error;
            if(root->end<l||root->start>r) return error;
            if(root->start>=l&&root->end<=r) return {root->high,root->l
ow};
            int tpmax=-0x3f3f3f3f,tpmin=0x3f3f3f3f;
            if(root->left)
            {
                pair<int,int> tp1=query(root->left,l,r);
                tpmax=max(tp1.first,tpmax);
                tpmin=min(tp1.second,tpmin);
            }
            if(root->right)
            {
                pair<int,int> tp2=query(root->right,l,r);
                tpmax=max(tp2.first,tpmax);
                tpmin=min(tp2.second,tpmin);
            }
            return {tpmax,tpmin};
        }
        pair<int,int> query(int l,int r) {return query(root,l,r);}
};
int main()
{

    return 0;
}
```

case3:sum+upd+set

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
```

```cpp
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
#define int long long
using namespace std;
class segtree{
    public:
        struct node
        {
            int sum;
            int s;int e;
            node* lt;
            node* rt;
            int lazysum=0;
            int lazycover=0;
            node(int sum,int s,int e):
            sum(sum),s(s),e(e),lt(nullptr),
            rt(nullptr),lazycover(0),lazysum(0){}
        };
        node* root;
        node* build(vector<int>& nums,int l,int r)
        {
            if(l>r) return nullptr;
            if(l==r) return new node(nums[l],l,l);
            int mid=(l+r)>>1;
            node* root=new node(0,l,r);
            node* lc=build(nums,l,mid);
            node* rc=build(nums,mid+1,r);
            root->lt=lc,root->rt=rc;
            root->sum=(lc?lc->sum:0)+(rc?rc->sum:0);
            return root;
        }
        void init(vector<int>& nums) {root=build(nums,0,nums.size()-1);}
        void taglazy(node* root,int val,int op){
            //1:sum,2:cover
            if(!root) return;
            if(op==1)
            {
                root->lazysum+=val;
                root->sum+=(val)*(root->e-root->s+1);
            }
            else if(op==2)
            {
```

```
            root->lazycover=val;
            root->lazysum=0;
            root->sum=val*(root->e-root->s+1);
        }
    }
    void pushdown(node* root)
    {
        if(root->lazycover){
            taglazy(root->lt,root->lazycover,2);
            taglazy(root->rt,root->lazycover,2);
            root->lazycover=0;
        }
        if(root->lazysum){
            taglazy(root->lt,root->lazysum,1);
            taglazy(root->rt,root->lazysum,1);
            root->lazysum=0;
        }
    }
    int query(node* root,int l,int r)
    {
        pushdown(root);
        if(!root) return 0;
        if(root->s>r||root->e<l) return 0;
        if(root->s>=l&&root->e<=r) return root->sum;
        return query(root->lt,l,r)+query(root->rt,l,r);
    }
    int query(int l,int r) {return query(root,l,r);}
    void update(node* root,int l,int r,int val,int op)
    {

        if(!root) return ;
        if(root->s>r||root->e<l) return ;
        if(root->s>=l&&root->e<=r)
        {
            taglazy(root,val,op);
            return ;
        }
        pushdown(root);
        update(root->lt,l,r,val,op);
        update(root->rt,l,r,val,op);
        root->sum=(root->lt?root->lt->sum:0)+(root->rt?root->rt->su
m:0);
        return ;
    }
    void update(int l,int r,int val,int op){
        update(root,l,r,val,op);
    }

};
```

```
int read()
{
    int s=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9')
    {
        if(ch=='-') f=-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9')
    {
        s=(s<<3)+(s<<1)+ch-'0';
        ch=getchar();
    }
    return s*f;
}
inline void write(int x)
{
    static int sta[35];
    int top=0;
    if(x<0&&x!=-2147483648) {putchar('-');x=-x;}
    if(x==-2147483648) {printf("-2147483648");return;}
    do{
      sta[top++]=x%10, x/=10;
      }while(x);
    while(top) putchar(sta[--top]+48);
}
signed main()
{

    return 0;
}
```

case4:max/min+upd+set

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
```

```cpp
#include <unordered_map>
using namespace std;
#define int long long
class segtree{
    public:
        const int H=-0xffffffffffffff;
        const int L=0xffffffffffffff;
        pair<int,int> error={H,L};
        struct node{
            int s;int e;
            int lazyadd;
            int lazycover;
            int lazycoveradd;
            int high,low;
            node* lt;
            node* rt;
            node(int s,int e,int high,int low):
            s(s),e(e),lazyadd(0),lazycover(0),
            high(high),low(low),lazycoveradd(-1),
            lt(nullptr),rt(nullptr){}
        };
        node* root;
        void inf(node* a,node* l,node* r)
        {
            if(!a) return ;
            if(!l&&r) a->low=r->low,a->high=r->high;
            if(l&&!r) a->low=l->low,a->high=l->high;
            if(l&&r) a->low=min(l->low,r->low),a->high=max(l->high,r->h
igh);
        }
        node* build(vector<int> &nums,int l,int r)
        {
            if(l>r) return nullptr;
            if(l==r) return new node(l,r,nums[l],nums[l]);
            int mid=(l+r)>>1;
            node* root=new node(l,r,H,L);
            node* lc=build(nums,l,mid);
            node* rc=build(nums,mid+1,r);
            root->lt=lc,root->rt=rc;
            inf(root,lc,rc);
            return root;
        }
        void init(vector<int> &nums){root=build(nums,0,nums.size()-1);}
        void taglazy(node* root,int val,int op)
        {
            //op 1 add,2 cover
            if(!root) return ;
            if(op==1)
            {
```

```
            root->lazyadd+=val;
            root->high+=val;
            root->low+=val;
        }
        if(op==2)
        {
            root->lazycover=val;
            root->lazycoveradd=1;
            root->high=val;
            root->low=val;
            root->lazyadd=0;
        }
    }
    void pushdown(node* root)
    {
        if(!root) return ;
        if(root->lazycoveradd==1)
        {
            taglazy(root->lt,root->lazycover,2);
            taglazy(root->rt,root->lazycover,2);
            root->lazycover=0;
            root->lazycoveradd=-1;
        }
        if(root->lazyadd)
        {
            taglazy(root->lt,root->lazyadd,1);
            taglazy(root->rt,root->lazyadd,1);
            root->lazyadd=0;
        }
    }
    pair<int,int> query(node* root,int l,int r)
    {
        pushdown(root);
        if(!root) return error;
        if(root->s>r||root->e<l) return error;
        if(l<=root->s&&root->e<=r)
        {
            //cout<<"query:"<<root->s<<' '<<root->e<<' '<<root->hig
h<<' '<<root->low<<endl;
            return {root->high,root->low};
        }
        pair<int,int> tplt=query(root->lt,l,r);
        pair<int,int> tprt=query(root->rt,l,r);
        int anshigh=max(tplt.first,tprt.first);
        int anslow=min(tplt.second,tprt.second);
        return {anshigh,anslow};
    }
    pair<int,int> query(int l,int r) {return query(root,l,r);}
    void update(node* root,int l,int r,int val,int op)
```

```
        {
            if(!root) return ;
            if(root->s>r||root->e<l) return ;
            if(l<=root->s&&root->e<=r)
            {
                //cout<<"taglazy:"<<root->s<<' '<<root->e<<' '<<val<<'
'<<op<<endl;
                taglazy(root,val,op);
                return;
            }
            pushdown(root);
            update(root->lt,l,r,val,op);
            update(root->rt,l,r,val,op);
            inf(root,root->lt,root->rt);
            return ;
        }
        void update(int l,int r,int val,int op){
            update(root,l,r,val,op);
        }

};
int read()
{
    int s=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9')
    {
        if(ch=='-') f=-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9')
    {
        s=(s<<3)+(s<<1)+ch-'0';
        ch=getchar();
    }
    return s*f;
}
inline void write(int x)
{
    static int sta[35];
    int top=0;
    if(x<0&&x!=-2147483648) {putchar('-');x=-x;}
    if(x==-2147483648) {printf("-2147483648");return;}
    do{
      sta[top++]=x%10, x/=10;
      }while(x);
    while(top) putchar(sta[--top]+48);
}
signed main()
```

```
{
    int T_start=clock();
    //freopen("in.txt","r",stdin);
    //freopen("out.txt","w",stdout);
    int n=read(),m=read();
    vector<int>a(n);
    for(auto&i:a) i=read();
    segtree st;st.init(a);
    while(m--)
    {
        int op=read(),l=read(),r=read();
        if(op==1)
        {
            int x=read();
            st.update(l-1,r-1,x,2);
        }
        else if(op==2)
        {
            int x=read();
            st.update(l-1,r-1,x,1);
        }
        else printf("%lld\n",st.query(l-1,r-1).first);
        // for(int i=0;i<n;i++)
        // {
        //     cout<<st.query(i,i).first<<' ';
        // }
        // cout<<endl;
    }
    return 0;
}
```

jiangly 线段树

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
#include <numeric>
```

```cpp
#include <functional>
using namespace std;
template<class Info, class Tag>
struct LazySegmentTree {
    const int n;
    std::vector<Info> info;
    std::vector<Tag> tag;
    LazySegmentTree(int n) : n(n), info(4 << std::__lg(n)), tag(4 << st
d::__lg(n)) {}
    LazySegmentTree(std::vector<Info> init) : LazySegmentTree(init.size
()) {
        std::function<void(int, int, int)> build = [&](int p, int l, in
t r) {
            if (r - l == 1) {
                info[p] = init[l];
                return;
            }
            int m = (l + r) / 2;
            build(2 * p, l, m);
            build(2 * p + 1, m, r);
            pull(p);
        };
        build(1, 0, n);
    }
    void pull(int p) {
        info[p] = info[2 * p] + info[2 * p + 1];
    }
    void apply(int p, const Tag &v) {
        info[p].apply(v);
        tag[p].apply(v);
    }
    void push(int p) {
        apply(2 * p, tag[p]);
        apply(2 * p + 1, tag[p]);
        tag[p] = Tag();
    }
    void modify(int p, int l, int r, int x, const Info &v) {
        if (r - l == 1) {
            info[p] = v;
            return;
        }
        int m = (l + r) / 2;
        push(p);
        if (x < m) {
            modify(2 * p, l, m, x, v);
        } else {
            modify(2 * p + 1, m, r, x, v);
        }
        pull(p);
```

```
    }
    void modify(int p, const Info &v) {
        modify(1, 0, n, p, v);
    }
    Info rangeQuery(int p, int l, int r, int x, int y) {
        if (l >= y || r <= x) {
            return Info();
        }
        if (l >= x && r <= y) {
            return info[p];
        }
        int m = (l + r) / 2;
        push(p);
        return rangeQuery(2 * p, l, m, x, y) + rangeQuery(2 * p + 1, m,
r, x, y);
    }
    Info rangeQuery(int l, int r) {
        return rangeQuery(1, 0, n, l, r);
    }
    void rangeApply(int p, int l, int r, int x, int y, const Tag &v) {
        if (l >= y || r <= x) {
            return;
        }
        if (l >= x && r <= y) {
            apply(p, v);
            return;
        }
        int m = (l + r) / 2;
        push(p);
        rangeApply(2 * p, l, m, x, y, v);
        rangeApply(2 * p + 1, m, r, x, y, v);
        pull(p);
    }
    void rangeApply(int l, int r, const Tag &v) {
        return rangeApply(1, 0, n, l, r, v);
    }
    void half(int p, int l, int r) {
        if (info[p].act == 0) {
            return;
        }
        if ((info[p].min + 1) / 2 == (info[p].max + 1) / 2) {
            apply(p, {-(info[p].min + 1) / 2});
            return;
        }
        int m = (l + r) / 2;
        push(p);
        half(2 * p, l, m);
        half(2 * p + 1, m, r);
        pull(p);
```

```
    }
    void half() {
        half(1, 0, n);
    }
};
struct Tag {
    //tag 清空态
    void apply(Tag t) {
        //tag t 下发对 tag 的影响
    }
};
struct Info {
    //维护啥信息
    void apply(Tag t) {
        //tag t 下发对 info 的影响
    }
};
Info operator + (Info a, Info b) {
    Info c;
    //info a 和 info b 合并
    return c;
}
//tip:[l,r)区间->传入[l,r]改为[l,r+1)
int main()
{
    int T_start=clock();

    return 0;
}
```

笛卡尔树

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
using namespace std;
struct DKRTreeNode{
    int val;
    int index;
    DKRTreeNode *left, *right;
    DKRTreeNode(int x,int i):val(x),index(i),left(NULL),right(NULL){}
};
//笛卡尔树具有一下性质:
//1.二叉搜索树，2.堆
//i:index,val:nums[i]
//因为BST的中序遍历一定是原序列，所以新插入的节点一定在右边
//需要调整最右边的纵向位置，使其满足堆的性质，用单调栈维护最右链
//大根堆
DKRTreeNode* buildTree(vector<int> &nums){
    stack<DKRTreeNode*> s;
    DKRTreeNode* root=nullptr;
    for(int i=0;i<nums.size();i++)
    {
        DKRTreeNode* node=new DKRTreeNode(nums[i],i);
        DKRTreeNode* last=nullptr;
        while(!s.empty()&&s.top()->val<node->val)
        {
            last=s.top();
            s.pop();
        }//单调栈维护
        if(!s.empty()) s.top()->right=node;//栈顶元素的右子树为node
        if(last) node->left=last;//栈弹出的元素为node的左子树
        s.push(node);
    }
    while(!s.empty()) root=s.top(),s.pop();//最后一个元素为根节点
    return root;
}
```

```
void printTree(DKRTreeNode* root)
{
    if(root==nullptr) return;
    printTree(root->left);
    cout<<root->val<<" ";
    printTree(root->right);
}
int main()
{
    int T_start=clock();
    vector<int> nums(10,0);
    for(int i=0;i<nums.size();i++) nums[i]=rand()%100;
    for(auto i:nums) cout<<i<<" ";
    cout<<endl;
    DKRTreeNode* root=buildTree(nums);
    printTree(root);
    cout<<root->val<<endl;
    return 0;
}
```

堆

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
using namespace std;
class Heap{//小根堆
    public:
        struct node{
            int val;
        };
        vector<node> heap;
        int n,size=0;
        Heap(int n):n(n){
            heap.resize(n+1);
        }
```

```
        bool cmp(node a,node b){//自定义比较函数
            return a.val<b.val;
        }
        //1...n 的完全二叉树,i 的父亲为 i/2,i 的左孩子为 2i,右孩子为 2i+1
        void push(int val){
            heap[++size].val=val;
            int i=size;
            while(i>1&&cmp(heap[i],heap[i/2])){
                swap(heap[i],heap[i/2]);
                i/=2;//向上调整
            }
        }
        void pop(){
            heap[1]=heap[size--];//将最后一个元素放到第一个位置
            int i=1;
            while(i*2<=size){
                int j=i*2;
                if(j<size&&cmp(heap[j+1],heap[j])) j++;//找到左右孩子中
较大的一个
                if(cmp(heap[i],heap[j])) break;//如果父亲节点比孩子节点大,
则不需要调整
                swap(heap[i],heap[j]);
                i=j;
            }//向下调整
        }
        int top(){
            return heap[1].val;
        }

};
int main()
{
    int T_start=clock();
    int n;
    cin>>n;
    Heap h(n);
    for(int i=0;i<n;i++){
        int x;
        cin>>x;
        h.push(x);
    }
    while(h.size>1){
        cout<<h.top()<<" ";
        h.pop();
    }
    return 0;
}
```

莫队

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <unordered_map>
#include <array>
using namespace std;
class BIT{
    private:
        int n;
        vector<int> tree;//tree[i] 是[i-lowbit(i)+1,i]的和,[1,n]存储
        int lowbit(int x){
            return x&(-x);
        }
    public:
        BIT(int n): n(n),tree(n+1,0){}
        void update(int i,int val)//单点修改 a[i]+=val
        {
            while(i<=n){
                tree[i]+=val;
                i+=lowbit(i);//跳到后一个 lowbit(x)的位置
            }
        }
        int query(int l,int r)//区间查询 [l,r]的和
        {
            int res=0;
            while(r>=l){
                res+=tree[r];
                r-=lowbit(r);//跳到前一个 lowbit(x)的位置
            }
            return res;
        }
        void init(vector<int> a)//初始化
        {
            vector<int> presum(a.size()+1,0);
            for(int i=1;i<=a.size();i++)
```

```
            {
                presum[i]=presum[i-1]+a[i-1];
                tree[i]=presum[i]-presum[i-lowbit(i)];//按定义
            }
        }
};
int read()
{
    int s=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9')
    {
        if(ch=='-') f=-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9')
    {
        s=(s<<3)+(s<<1)+ch-'0';
        ch=getchar();
    }
    return s*f;
}
inline void write(int x)
{
    static int sta[35];
    int top=0;
    if(x<0&&x!=-2147483648) {putchar('-');x=-x;}
    if(x==-2147483648) {printf("-2147483648");return;}
    do{
      sta[top++]=x%10, x/=10;
      }while(x);
    while(top) putchar(sta[--top]+48);
}
unordered_map<int,int> dis(vector<int> a)
{
    sort(a.begin(),a.end());
    unordered_map<int,int> mp;
    for(int i=0;i<a.size();i++)
    {
        mp[a[i]]=i+1;
    }
    return mp;
}
int main()
{
    int t=read();
    while(t--)
    {
        int n=read(),m=read();
```

```
        vector<int> a(n+1);
        for(int i=1;i<=n;i++)
        {
            a[i]=read();
        }
        vector<array<int,4>> q(m);
        for(int i=0;i<m;i++)
        {
            q[i][0]=read();
            q[i][1]=read();
            q[i][2]=read();
            q[i][3]=i;
        }
        int block=static_cast<int>(sqrt(n))+1;//按值域分块
        auto cmp=[block](array<int,4> a,array<int,4> b)
        {
            if(a[0]/block!=b[0]/block) return a[0]/block<b[0]/block;//
按块排序
            else{
                if(a[0]/block%2==0) return a[1]<b[1];//按值排序
                else return a[1]>b[1];//按块排序
            }
        };
        sort(q.begin(),q.end(),cmp);
        vector<int> ans(m,0);
        int l=1,r=0;
        BIT bit(n+1);
        for(auto [ql,qr,x,idx]:q)//暴力
        {
            while(r<qr) bit.update(a[++r],1);
            while(l>ql) bit.update(a[--l],1);
            while(r>qr) bit.update(a[r--],-1);
            while(l<ql) bit.update(a[l++],-1);
            //cout<<l<<" "<<r<<endl;
            ans[idx]=bit.query(1,a[x])+l-1;
        }
        for(auto i:ans) write(i),putchar('\n');
    }
    return 0;
}
```

珂朵莉树

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
#include <numeric>
using namespace std;
class ODT{
    public:
        struct node
        {
            int l,r;
            mutable int val;
            node(int l,int r,int val):l(l),r(r),val(val){}
            bool operator<(const node &o)const {
                return l<o.l;
            }
        };
        set<node> s;
        //0-based
        ODT(vector<int> &a){
            for(int i=0;i<a.size();i++){
                s.insert(node(i,i,a[i]));
            }
        }
        //把[l,r]区间分割成[l,mid)和[mid,r]两个区间
        auto split(int pos){
            auto it=s.lower_bound(node(pos,0,0));
            if(it!=s.end()&&it->l==pos) return it;
            --it;
            int l=it->l,r=it->r,val=it->val;
            s.erase(it);
            s.insert(node(l,pos-1,val));
            return s.insert(node(pos,r,val)).first;
        }
        //把[l,r]区间赋值为val
```

```
        void assign(int l,int r,int val){
            auto itr=split(r+1),itl=split(l);
            s.erase(itl,itr);
            s.insert(node(l,r,val));
        }
        //对区间操作
        void perform(int l,int r)
        {
            auto itr=split(r+1),itl=split(l);
            for(auto it=itl;it!=itr;++it)
            {
                //perform
            }
        }
};
int main()
{
    int T_start=clock();

    return 0;
}
//ODT(珂朵莉树)
//处理区间查询后立即覆盖问题
//修改/查询的一次为O(logn) 一次查询m个区间 覆盖后最多产生3个区间，并减少m左右个区间
//一次操作的代价是随机变量di 那么q次操作的期望是 n乘一个小常数
//所以期望是 均摊O(nlogn)
```

李超线段树

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
#include <numeric>
#include <functional>
#include <ranges>
#include <iomanip>
#define int long long //赫赫 要不要龙龙呢
using namespace std;
class LCSegTree{
public:
    const int L=1,R=1e9;
    const double eps=1e-9;
    struct Line{
        long double a,b;
        int id;
        Line(long double a=0,long double b=/*-1e18*/1e18,int id=0):a(a),
b(b),id(id){}
        long double val(int x)const{return (long double)1.0*a*x+b;}
    };
    struct Node{
        Line l;
        Node *lc,*rc;
        Node():l(),lc(nullptr),rc(nullptr){}
    };
    Node *rt;
    LCSegTree(){
        rt=nullptr;
    }
    bool cmp(long double a,long double b){
           // return a-b>eps;//max
           return b-a>eps;//min
    }
    bool cmp1(Line a,Line b,int x){
```

```
            long double va=a.val(x),vb=b.val(x);
            return cmp(va,vb)||(fabs(va-vb)<eps&&a.id<b.id);
      }
    bool cmp2(pair<long double,int> a,long double b,int cid){
            return cmp(a.first,b)||(fabs(a.first-b)<eps&&a.second<cid);
    }
    void ins(Node*&o,int l,int r,int ql,int qr,Line v){
            if(qr<l||r<ql)return;
            if(!o)o=new Node();
            if(ql<=l&&r<=qr){
                  int mid=(l+r)>>1;
                  bool LB=cmp1(v,o->l,l),MB=cmp1(v,o->l,mid),RB=cmp1(v,
o->l,r);
                  if(MB)swap(o->l,v);
                  if(l==r)return;
                  if(LB!=MB)ins(o->lc,l,mid,ql,qr,v);
                  else ins(o->rc,mid+1,r,ql,qr,v);
                  return;
            }
            int mid=(l+r)>>1;
            ins(o->lc,l,mid,ql,qr,v);
            ins(o->rc,mid+1,r,ql,qr,v);
      }
      pair<long double,int> qry(Node*o,int l,int r,int x){
            if(!o)return {/*-1e18*/1e18,0};
            long double cur=o->l.val(x);
            int cid=o->l.id;
            int mid=(l+r)>>1;
            if(x<=mid){
                  auto res=qry(o->lc,l,mid,x);
                  if(cmp2(res,cur,cid))return res;
            }else{
                  auto res=qry(o->rc,mid+1,r,x);
                  if(cmp2(res,cur,cid))return res;
            }
            return {cur,cid};
      }
      void add(int x1,int y1,int x2,int y2,int id){
            if(x1==x2){
                  int y=max(y1,y2);
                  ins(rt,L,R,x1,x1,Line(0,y,id));
            }else{
                  if(x1>x2)swap(x1,x2),swap(y1,y2);
                  long double a=1.0*(y2-y1)/(x2-x1),b=1.0*y1-a*x1;
                  ins(rt,L,R,x1,x2,Line(a,b,id));
            }
      }
    void add(Line v){
        ins(rt,L,R,L,R,v);
```

```
    }
      pair<long double,int> ask(int x){return qry(rt,L,R,x);}
};
signed main()
{
    int T_start=clock();
    //freopen("in.txt","r",stdin);
    //freopen("out.txt","w",stdout);
    //ios::sync_with_stdio(false),cin.tie(0),cout.tie(0);
    return 0;
}
//李超线段树插入O(logDlogD)查询O(logD)
//解决：插入一条直线，查询某个点的最大/最小值
//插入先划分logn区间，再懒标记下放(logn)
//维护的是直线中点的最大/最小值
//证明，对每个局部，区间中值最大代表着这个局部最优，于是可以遍历获得全局最优
//空间复杂度O(nlogDlogD)
```

图论

拓扑排序

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
using namespace std;
#define MOD 80112002
vector<int> edge[5005];
int _to[5005]={0},_in[5005]={0};
long long ans[5005]={0};queue<int> q;
int read()
{
    int s=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9')
    {
        if(ch=='-') f=-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9')
    {
        s=(s<<3)+(s<<1)+ch-'0';
        ch=getchar();
    }
    return s*f;
}
inline void write(int x)
{
    static int sta[35];
    int top=0;
    do{
      sta[top++]=x%10, x/=10;
      }while(x);
    while(top) putchar(sta[--top]+48);
}
int main()
```

```
{
    int T_start=clock();
    int n=read(),m=read();
    for(int i=0;i<m;i++)
    {
        int u=read(),v=read();
        edge[v].push_back(u);
        _to[u]++;_in[v]++;
    }
    for(int i=1;i<=n;i++)
    {
        if(!_to[i])
        {
            q.push(i);
            ans[i]=1;
        }
    }
    while(!q.empty())
    {
        int temp=q.front();
        //cout<<temp<<endl;
        q.pop();
        for(int i=0;i<edge[temp].size();i++)
        {
            //cout<<temp<<' '<<edge[temp][i]<<' '<<ans[temp]<<' '<<ans
[edge[temp][i]]<<endl;
            ans[edge[temp][i]]=(ans[edge[temp][i]]+ans[temp])%MOD;
            _to[edge[temp][i]]--;
            if(!_to[edge[temp][i]]) q.push(edge[temp][i]);
        }
    }
    long long res=0;
    for(int i=1;i<=n;i++)
    {
        if(!_in[i])
        {
            //cout<<i<<' '<<ans[i]<<endl;
            res=(res+ans[i])%MOD;
        }
    }
    write(res),putchar('\n');
    return 0;
}
```

dfs/bfs

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
using namespace std;
vector <int> edge[100000+5];
queue <int> q;int vis[100000+5]={0},sum=0;
int ans[100000+5];
int read()
{
    int s=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9')
    {
        if(ch=='-') f=-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9')
    {
        s=(s<<3)+(s<<1)+ch-'0';
        ch=getchar();
    }
    return s*f;
}
inline void write(int x)
{
    static int sta[35];
    int top=0;
    do{
      sta[top++]=x%10, x/=10;
      }while(x);
    while(top) putchar(sta[--top]+48);
}
void dfs(int x)
{
    write(x);putchar(' ');vis[x]=1;
    for(int i=0;i<edge[x].size();i++)
```

```
    {
        if(!vis[edge[x][i]]) dfs(edge[x][i]);
    }
    return ;
}
int bfs(int x)
{
    q.push(x);
    while(!q.empty())
    {
        int temp=q.front(); q.pop();
        if(vis[temp]) continue;
        else{
            vis[temp]=1;sum++;
        }
        for(int i=0;i<edge[temp].size();i++)
        {
            q.push(edge[temp][i]);
            if(!vis[edge[temp][i]]) ans[edge[temp][i]]=ans[temp];
        }
        // cout<<q.size()<<endl;
        // for(int i=0;i<=q.size();i++)
        // {
        //     cout<<q.front()<<" ";
        //     q.pop();
        // }
    }
    return sum;
}
int main()
{
    int T_start=clock();
    int n=read(),m=read();
    for(int i=0;i<m;i++)
    {
        int u=read(),v=read();
        edge[v].push_back(u);
    }
    for(int i=1;i<=n;i++)
    {
        ans[i]=i;
    }
    for(int i=n;i>=1;i--)
    {
        bfs(i);
        if(sum==n) break;
    }
    for(int i=1;i<=n;i++)
    {
```

```
        write(ans[i]);putchar(' ');
    }
    putchar('\n');
    return 0;
}
//preview:2024.12.29 23:01
```

最短路（dijkstra）

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
using namespace std;
int read()
{
    int s=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9')
    {
        if(ch=='-') f=-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9')
    {
        s=(s<<3)+(s<<1)+ch-'0';
        ch=getchar();
    }
    return s*f;
}
inline void write(int x)
{
    static int sta[35];
    int top=0;
    if(x<0&&x!=-2147483648) {putchar('-');x=-x;}
    if(x==-2147483648) {printf("-2147483648");return;}
    do{
      sta[top++]=x%10, x/=10;
      }while(x);
    while(top) putchar(sta[--top]+48);
}
vector<int> dijkstra(int n,vector<vector<pair<int,int>>>mp,int s)
{
    vector<int> dis(n+1,0x7fffffff);//初始化距离为无穷大
```

```
    dis[s]=0;//起点到起点的距离为0
    priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,
int>>> pq;
    pq.push({0,s});//将起点加入优先队列
    while(!pq.empty())
    {
        int u=pq.top().second;//取出当前距离最小的点
        int d=pq.top().first;//取出当前距离最小的点的距离
        pq.pop();
        if(d>dis[u]) continue;//u 已经被更新过
        if(mp[u].empty()) continue;
        for(auto it:mp[u])
        {
            int v=it.first;
            int w=it.second;
            if(dis[v]>dis[u]+w)//更新 s->v 的最短距离(min(s->v,s->u->v))
            {
                dis[v]=dis[u]+w;
                pq.push({dis[v],v});
            }
        }
    }
    //正确性证明:
    //假设目前更新 s->t(=3),假设存在 s->u->t(=2),则 s->u<s->t,而 s->u 一定在
之前被更新过，所以 s->u->t 一定在之前被更新过，与假设矛盾。
    //单源最短路(正边权)
    //时间复杂度 O(ElogV),E 为边数,V 为点数(二叉堆)
    //使用斐波那契堆的 Dijkstra 算法的时间复杂度为 O(E+VlogV)。
    //不用堆优化：O(v^2+E)
    //当 E<<v^2 时，使用堆优化
    //当 E~v^2 时，不用堆优化
    return dis;
}
int main()
{
    int T_start=clock();
    //freopen("in.txt","r",stdin);
    int n=read(),m=read(),s=read();
    vector<vector<pair<int,int>>> mp(n+1);
    while(m--)
    {
        int u=read(),v=read(),w=read();
        mp[u].push_back({v,w});
    }
    vector<int> dis=dijkstra(n,mp,s);
    for(int i=1;i<=n;i++)
        write(dis[i]),putchar(' ');
```

```
    int T_end=clock();
    return 0;
}
```

最短路（spfa）

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
using namespace std;
int read()
{
    int s=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9')
    {
        if(ch=='-') f=-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9')
    {
        s=(s<<3)+(s<<1)+ch-'0';
        ch=getchar();
    }
    return s*f;
}
inline void write(int x)
{
    static int sta[35];
    int top=0;
    if(x<0&&x!=-2147483648) {putchar('-');x=-x;}
    if(x==-2147483648) {printf("-2147483648");return;}
    do{
      sta[top++]=x%10, x/=10;
      }while(x);
    while(top) putchar(sta[--top]+48);
}
```

```
vector<int> Bellman_Ford(vector<vector<pair<int,int>>>& mp,int s,int n)
{
    vector<int> dis(n+1,0x7fffffff);
    dis[s]=0;
    for(int i=1;i<=n-1;i++)
    {
        for(int j=1;j<=n;j++)
        {
            for(auto [u,w]:mp[j])
            {
                if(dis[j]!=0x7fffffff&&dis[j]+w<dis[u])
                {
                    dis[u]=dis[j]+w;
                }
            }
        }
    }
    for(int j=1;j<=n;j++)
    {
        for(auto [u,w]:mp[j])
        {
            if(dis[j]!=0x7fffffff&&dis[j]+w<dis[u])
            {
                cout<<"negative cycle!"<<endl;
            }
        }
    }
    return dis;
    //Bellman_Ford 对所有的边进行 n-1 次松弛操作，如果在进行第 n 次松弛操作时，仍然存在边可以松弛，则说明图中存在负权环（从 s 点出发存在负权环）
    //时间复杂度：O(nm),形式上就是暴力）
    //第 i 次循环，我们能找到经历 i 条边到达的点的最短距离
    //所以第 n 次循环，我们能找到经历 n 条边到达的点的最短距离，如果存在负权环，那么一定能在第 n 次循环找到经历 n 条边到达的点的最短距离
}
vector<int> SPFA(vector<vector<pair<int,int>>>& mp,int s,int n)
{
    vector<int> dis(n+1,0x7fffffff);
    vector<int> vis(n+1,0);
    vector<int> cnt(n+1,0);
    dis[s]=0;queue<int> q;
    q.push(s);vis[s]=1;cnt[s]=0;
    while(!q.empty())
    {
        int u=q.front();
        q.pop();vis[u]=0;
        for(auto [v,w]:mp[u])
        {
```

```
            if(dis[v]>dis[u]+w)//松弛
            {
                dis[v]=dis[u]+w;
                cnt[v]=cnt[u]+1;
                if(cnt[v]>n-1)//存在负权环
                {
                    //1-n 的节点，最短路最多经过 n-1 条边，如果经过 n 条边，说明存在负权环
                    cout<<"negative cycle!"<<endl;
                    return dis;
                }
                if(!vis[v])
                {
                    q.push(v);
                    vis[v]=1;
                }
            }
        }
    }
    return dis;
    //SPFA：形式上 Bellman_Ford 是一棵树，很显然，只有上一次被松弛的节点 u，才有可能对 v 进行松弛，所以可以采用 SPFA
    //为啥只有上一次被松弛的节点 u，才有可能对 v 进行松弛？
    //手玩一下就好了（悲
    //考虑简单图，他可以是个递推的过程
}
int main()
{
    int T_start=clock();
    int n=read(),m=read();
    vector<vector<pair<int,int>>> mp(n+1);
    for(int i=0;i<m;i++)
    {
        int u=read(),v=read(),w=read();
        mp[u].push_back({v,w});
    }
    return 0;
}
```

最短路（floyd）

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
```

```cpp
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
using namespace std;
const int MAXN=1e4;
int Graph[MAXN][MAXN];
int dp[MAXN][MAXN];
int main()
{
    int T_start=clock();
    int T=10;
    srand(time(NULL));
    for(int i=1;i<=T;i++)
    {
        for(int j=1;j<=T;j++)
        {
            int op=rand()%3;
            if(false) Graph[i][j]=0;
            else if(op==1) Graph[i][j]=0x3f3f3f3f;
            else Graph[i][j]=(rand()*10+rand())%10;
            //Graph[i][j]=(rand()*10+rand())%10;
        }
        Graph[i][i]=0;
    }


    for(int i=1;i<=T;i++)
    {
        for(int j=1;j<=T;j++)
        {
            dp[i][j]=Graph[i][j];
        }
    }
    for(int i=1;i<=T;i++)
    {
        for(int j=1;j<=T;j++)
        {
            cout<<dp[i][j]<<" ";
        }
        cout<<endl;
    }
    cout<<endl;
    for(int i=1;i<=T;i++)
    {
        for(int j=1;j<=T;j++)
```

```
        {
            for(int k=1;k<=T;k++)
            {
                dp[j][k]=min(dp[j][k],dp[j][i]+dp[i][k]);
            }
        }
    }

    for(int i=1;i<=T;i++)
    {
        for(int j=1;j<=T;j++)
        {
            cout<<dp[i][j]<<" ";
        }
        cout<<endl;
    }
    int T_end=clock();
    return 0;
}

//Floyed 算法
//处理任意两点之间的最短路径(无负环)
//时间复杂度O(n^3)
```

最短路（johnson）

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
#include <numeric>
#include <functional>
using namespace std;
#define int long long
int has_neg=0;
vector<int> SPFA(vector<vector<pair<int,int>>>& mp,int s,int n)
{
```

```
    vector<int> dis(n+1,1e9);
    vector<int> vis(n+1,0);
    vector<int> cnt(n+1,0);
    dis[s]=0;queue<int> q;
    q.push(s);vis[s]=1;cnt[s]=0;
    while(!q.empty())
    {
        int u=q.front();
        q.pop();vis[u]=0;
        for(auto [v,w]:mp[u])
        {
            if(dis[v]>dis[u]+w)//松弛
            {
                dis[v]=dis[u]+w;
                cnt[v]=cnt[u]+1;
                if(cnt[v]>n-1)//存在负权环
                {
                    //1-n 的节点，最短路最多经过 n-1 条边，如果经过 n 条边，说
明存在负权环
                    has_neg=1;
                    return dis;
                }
                if(!vis[v])
                {
                    q.push(v);
                    vis[v]=1;
                }
            }
        }
    }
    return dis;
    //SPFA: 形式上 Bellman_Ford 是一棵树，很显然，只有上一次被松弛的节点 u，
才有可能对 v 进行松弛，所以可以采用 SPFA
    //为啥只有上一次被松弛的节点 u，才有可能对 v 进行松弛?
    //手玩一下就好了（悲
    //考虑简单图，他可以是个递推的过程
}
vector<int> dijkstra(int n,vector<vector<pair<int,int>>>& mp,int s)
{
    vector<int> dis(n+1,1e9);//初始化距离为无穷大
    dis[s]=0;//起点到起点的距离为 0
    priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,
int>>> pq;
    pq.push({0,s});//将起点加入优先队列
    while(!pq.empty())
    {
        int u=pq.top().second;//取出当前距离最小的点
```

```
        int d=pq.top().first;//取出当前距离最小的点的距离
        pq.pop();
        if(d>dis[u]) continue;//u 已经被更新过
        if(mp[u].empty()) continue;
        for(auto it:mp[u])
        {
            int v=it.first;
            int w=it.second;
            if(dis[v]>dis[u]+w)//更新 s->v 的最短距离(min(s->v,s->u->v))
            {
                dis[v]=dis[u]+w;
                pq.push({dis[v],v});
            }
        }
    }
    //正确性证明:
    //假设目前更新 s->t(=3),假设存在 s->u->t(=2),则 s->u<s->t,而 s->u 一定在
之前被更新过，所以 s->u->t 一定在之前被更新过，与假设矛盾。
    //单源最短路(正边权)
    //时间复杂度 O(ElogV),E 为边数,V 为点数(二叉堆)
    //使用斐波那契堆的 Dijkstra 算法的时间复杂度为 O(E+VlogV)。
    //不用堆优化：O(v^2+E)
    //当 E<<v^2 时，使用堆优化
    //当 E~v^2 时，不用堆优化
    return dis;
}
vector<vector<int>> johnson(vector<vector<pair<int,int>>>& mp,int n)
{
    //1.添加一个虚拟节点 0，连接到所有节点，边权为 0
    for(int i=1;i<=n;i++)
    {
        mp[0].push_back({i,0});
    }
    //2.使用 Bellman-Ford 算法计算从虚拟节点 0 到所有节点的最短路径
    vector<int> h=SPFA(mp,0,n+1);
    if(has_neg==1)
    {
        cout<<-1<<endl;
        exit(0);
    }
    //3.删除虚拟节点 0，并更新所有边的权重
    for(int i=1;i<=n;i++)
    {
        for(auto& it:mp[i])
        {
            it.second+=h[i]-h[it.first];
        }
```

```
    }
    //4.对每个节点 i，使用Dijkstra 算法计算从 i 到所有节点的最短路径
    vector<vector<int>> dis(n+1,vector<int>(n+1,1e9));
    for(int i=1;i<=n;i++)
    {
        dis[i]=dijkstra(n,mp,i);
        for(int j=1;j<=n;j++)
        {
            dis[i][j]-=h[i]-h[j];
            if(dis[i][j]>=1e8) dis[i][j]=1e9;
        }
    }
    //5.更新所有边的权重
    for(int i=1;i<=n;i++)
    {
        for(auto& it:mp[i])
        {
            it.second+=h[it.first]-h[i];
        }
    }
    return dis;
}
signed main()
{
    int T_start=clock();
    int n,m;
    cin>>n>>m;
    vector<vector<pair<int,int>>> mp(n+1);
    for(int i=0;i<m;i++)
    {
        int u,v,w;
        cin>>u>>v>>w;
        mp[u].push_back({v,w});
    }
    vector<vector<int>> dis=johnson(mp,n);
    for(int i=1;i<=n;i++)
    {
        int ans=0;
        for(int j=1;j<=n;j++)
        {
            //cout<<dis[i][j]<<" ";
            ans+=j*dis[i][j];
        }
        //cout<<endl;
        cout<<ans<<endl;
    }
    return 0;
}
//johnson 全源最短路算法：O(nmlogm)
```

```
//重新标记边权后,u-v 两点的任意路径一定有 hu-hv 项，最短路不变
//由于三角形不等式，所以重新标记边权，边权一定非负
//所以重新标记边权后，可以使用 Dijkstra 算法求最短路
```

LCA（最近公共祖先,倍增）

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
using namespace std;
const int MAXN=5e5+5;
const int LOG=25;//MAXN<=2^LOG
vector<int> tree[MAXN];
int dep[MAXN],st[MAXN][LOG];//节点深度，st 表，st[i][j]=i 的 2^j 级祖先
int read()
{
	int s=0,f=1;
	char ch=getchar();
	while(ch<'0'||ch>'9')
	{
		if(ch=='-') f=-1;
		ch=getchar();
	}
	while(ch>='0'&&ch<='9')
	{
		s=(s<<3)+(s<<1)+ch-'0';
		ch=getchar();
	}
	return s*f;
}
inline void write(int x)
{
	static int sta[35];
	int top=0;
	if(x<0&&x!=-2147483648) {putchar('-');x=-x;}
	if(x==-2147483648) {printf("-2147483648");return;}
	do{
	  sta[top++]=x%10, x/=10;
	  }while(x);
	while(top) putchar(sta[--top]+48);
}
void init(int node,int parent)//用 dfs 预处理 dep 和 st
```

```
{
	dep[node]=(parent==-1)?0:dep[parent]+1;
	st[node][0]=parent;//一级祖先为自身
	for(int i=1;i<LOG;i++)//更新node 的祖先表
	{
		if(st[node][i-1]!=-1)
		{
			st[node][i]=st[st[node][i-1]][i-1];
			//node 的2^j 级祖先为node 的2^j-1 祖先的2^j-1 祖先
		}
		else st[node][i]=-1;//你的码的码没了，你还有码？（可删吗？）
	}
	for(auto child:tree[node])
	{
		if(child!=parent)
		{
			init(child,node);//从父节点向下dfs
		}
	}
}
int lca(int u,int v)
{
	if(dep[u]<dep[v]) swap(u,v);//确保u 比v 深
	int diff=dep[u]-dep[v];
	for(int i=0;i<LOG;i++)
	{
		if((diff>>i)&1)
		{
			u=st[u][i];//u 向上跳转2^i,其中i 为diff 的二进制表示中第
i 位为一
		}
	}
	if(u==v) return u;//深度相等，可能找到
	//不相等，假设他们与lca(u,v)的距离为diff
	//注意到5=4+1，5-4=1
	//7=4+2+1,7-4-2=1
	//6=4+2,6-4-1=1
	//12=8+4,12-8-2-1
	//做以下操作总能使diff=1
	// for(int i=LOG-1;i>=0;i--)
	// {
	//	if(st[u][i]!=st[v][i])
	//	{
	//		u=st[u][i];
	//		v=st[v][i];
	//	}
	// }
```

```
        // return st[u][0];
        //优化版
        for(int i=LOG-1;i>=0;i--)
        {
                if(st[u][i]!=st[v][i])
                {
                        u=st[u][i];
                        v=st[v][i];
                }
        }
        return st[u][0];
}
int main()
{
        int T_start=clock();
        int n=read(),m=read(),s=read();//n 个点，n-1 条边,m 个询问，s 为根
        for(int i=0;i<n-1;i++)
        {
                int u=read(),v=read();
                tree[u].push_back(v);
                tree[v].push_back(u);//存树
        }
        for(int i=1;i<=n;i++)
        {
                dep[i]=-1;
                for(int j=0;j<LOG;j++)
                {
                        st[i][j]=-1;
                }
        }
        init(s,-1);
        while(m--)
        {
                write(lca(read(),read()));
                putchar('\n');
        }
        return 0;
}
```

LCA（最近公共祖先,离线）

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
```

```cpp
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <unordered_map>
using namespace std;
const int MaxN=5e5+5;
vector<int> tree[MaxN];
vector<pair<int,int>> q[MaxN];
int vis[MaxN],ans[MaxN];
int fa[MaxN];
void prepare_tree(int n)
{
	for(register int i=1;i<=n;i++)
	{
		fa[i]=i;
	}
}
int find(int G)
{
	if(G==fa[G]) return G;
	else
	{
		fa[G]=find(fa[G]);
		return fa[G];
	}
	//return G==fa[G]? G:(fa[G]=find(fa[G]));
}
void merge(int a,int b)//合并
{
	fa[find(a)]=find(b);//有时路径压缩可能破坏rank'(rank->树深)
	/*register int x=find(a),y=find(b);
	Rank[x]<=Rank[y]?fa[x]=y:fa[y]=x;
	if(Rank[x]==Rank[y]&&x!=y) Rank[y]++;*/

}
int read()
{
	int s=0,f=1;
	char ch=getchar();
	while(ch<'0'||ch>'9')
	{
		if(ch=='-') f=-1;
		ch=getchar();
	}
	while(ch>='0'&&ch<='9')
	{
```

```
            s=(s<<3)+(s<<1)+ch-'0';
            ch=getchar();
        }
        return s*f;
}
inline void write(int x)
{
        static int sta[35];
        int top=0;
        if(x<0&&x!=-2147483648) {putchar('-');x=-x;}
        if(x==-2147483648) {printf("-2147483648");return;}
        do{
          sta[top++]=x%10, x/=10;
          }while(x);
        while(top) putchar(sta[--top]+48);
}
void dfs(int node)
{
        vis[node]=1;
        for(auto child:tree[node])
        {
            if(!vis[child])
            {
                dfs(child);
                fa[child]=node;//调换顺序会使路径压缩到 child 的父节点，
此时子树还没遍历完
            }
        }
        for(auto i:q[node])
        {
            if(vis[i.first])//node 及其子树已经 dfs 完了,如果此时 i 已经搜到，
显然，根据 dfs 原则，find(i)是 lca(i,node)
            {
                ans[i.second]=find(i.first);
            }
        }
}
int main()
{
        int T_start=clock();
        int n=read(),m=read(),s=read();
        for(int i=0;i<n-1;i++)
        {
            int u=read(),v=read();
            tree[v].push_back(u);
            tree[u].push_back(v);
        }
        for(int i=0;i<m;i++)
        {
```

```
            int u=read(),v=read();
            q[v].push_back(make_pair(u,i));
            q[u].push_back(make_pair(v,i));
        }
        prepare_tree(n);
        for(int i=1;i<=n;i++)
        {
            vis[i]=0;
        }
        dfs(s);
        for(int i=0;i<m;i++)
        {
            write(ans[i]);
            putchar('\n');
        }
        return 0;
}
```

最小生成树（Kruskal）

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
using namespace std;
class BSU{
    public:
        int n;vector<int> fa;
        BSU(int n):n(n)
        {
            fa.resize(n+1);
            for(int i=1;i<=n;i++)
            {
                fa[i]=i;
            }
        }
        int find(int u){
            return fa[u]==u?u:fa[u]=find(fa[u]);
        }
        void merge(int a,int b)
        {
            int op=rand()%2;
            if(op==0) fa[find(a)]=find(b);
            else fa[find(b)]=find(a);
        }
};
int read()
{
    int s=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9')
    {
        if(ch=='-') f=-1;
        ch=getchar();
    }
```

```
    while(ch>='0'&&ch<='9')
    {
        s=(s<<3)+(s<<1)+ch-'0';
        ch=getchar();
    }
    return s*f;
}
inline void write(int x)
{
    static int sta[35];
    int top=0;
    if(x<0&&x!=-2147483648) {putchar('-');x=-x;}
    if(x==-2147483648) {printf("-2147483648");return;}
    do{
      sta[top++]=x%10, x/=10;
      }while(x);
    while(top) putchar(sta[--top]+48);
}
int kruskal(vector<array<int,3>> &edge,int m,int n)
{
    sort(edge.begin(),edge.end(),[](auto a,auto b)->bool{
        return a[2]<b[2];
    });
    BSU bsu(n);int cnt=0;int ans=0;
    for(auto [u,v,w]:edge)
    {
        if(bsu.find(u)!=bsu.find(v))
        {
            bsu.merge(u,v);
            cnt++;ans+=w;
            //cout<<u<<" "<<v<<endl;
        }
        if(cnt==n-1) break;
    }
    return cnt==n-1?ans:-1;
}
//时间复杂度O(mlogm)，证明同prim
int main()
{
    int T_start=clock();
    srand(time(NULL));
    //freopen("in.txt","r",stdin);
    int n=read(),m=read();
    vector<array<int,3>> edge(m);
    for(int i=0;i<m;i++)
    {
        int u=read(),v=read(),w=read();
        edge[i]={u,v,w};
    }
```

```
    int ans=kruskal(edge,m,n);
    if(ans==-1) puts("orz");
    else cout<<ans<<endl;
    return 0;
}
```

最小生成树（Prim）

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
using namespace std;
int read()
{
    int s=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9')
    {
        if(ch=='-') f=-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9')
    {
        s=(s<<3)+(s<<1)+ch-'0';
        ch=getchar();
    }
    return s*f;
}
inline void write(int x)
{
    static int sta[35];
    int top=0;
    if(x<0&&x!=-2147483648) {putchar('-');x=-x;}
    if(x==-2147483648) {printf("-2147483648");return;}
    do{
      sta[top++]=x%10, x/=10;
      }while(x);
```

```
    while(top) putchar(sta[--top]+48);
}
vector<int> prim(vector<vector<pair<int,int>>> &mp,int s,int n)
{
    vector<int> dis(n+1,0x7fffffff);//点离当前生成树的距离
    vector<int> in(n+1,0);//点是否在生成树中
    priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,
int>>> pq;
    dis[s]=0;pq.push({0,s});
    while(!pq.empty())
    {
        auto [_,u]=pq.top();//找到最小生成树连的边中未加入生成树的边权最小
的边
        pq.pop();
        if(in[u]) continue;
        in[u]=true;//进入最小生成树
        for(auto [v,w]:mp[u])
        {
            if(dis[v]>w&&!in[v])//更新不在当前生成树中的点离生成树的距离
            {
                dis[v]=w;
                pq.push({dis[v],v});
            }
        }
    }
    return dis;
}
//和dij 一样，时间复杂度O((n+m)logn),暴力prim 时间复杂度O(n^2),看看稀疏图
和稠密图哪个更快
//正确性证明：反证法：假设prim 生成的不是最小生成树
// 1).设prim 生成的树为G0
// 2).假设存在Gmin 使得cost(Gmin)<cost(G0)    则在Gmin 中存在<u,v>不属于G0
// 3).将<u,v>加入G0 中可得一个环，且<u,v>不是该环的最长边(这是因为<u,v>∈Gmi
n)
// 4).这与prim 每次生成最短边矛盾
// 5).故假设不成立，命题得证.
int main()
{
    int T_start=clock();
    int n=read(),m=read();
    vector<vector<pair<int,int>>> mp(n+1);
    for(int i=1;i<=m;i++)
    {
        int u=read(),v=read(),w=read();
        mp[u].push_back({v,w});
        mp[v].push_back({u,w});
    }
```

```
    vector<int> ans=prim(mp,1,n);
    int sum=0;
    for(int i=1;i<=n;i++)
    {
        if(ans[i]==0x7fffffff)
        {
            puts("不连通！");
            return 0;
        }
        sum+=ans[i];
    }
    cout<<sum<<endl;
    return 0;
}
```

强连通分量（SCC)

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
using namespace std;
pair<vector<int>,int> tarjan(vector<vector<int>> &mp,int n)
{
      vector<int> bel(n+1,-1);//bel[i]:i 属于哪个强连通分量
    vector<int> dfn(n+1,-1),low(n+1,-1);
    stack<int> st;int cnt=0,scc_cnt=0;
    auto dfs=[&](auto dfs,int u)->void{
        dfn[u]=low[u]=++cnt; //时间戳+1
        st.push(u); //inst[u]=1; //入栈
        for(int v:mp[u])
        {
            if(dfn[v]==-1)//case1:u 的邻接点 v 未被访问过
            {
                dfs(dfs,v);
                low[u]=min(low[u],low[v]);
            }
                  else if(bel[v]==-1)//v 所属的强连通分量还未被确定（等价于
case2）
                  {
                        low[u]=min(low[u],dfn[v]);
                  }
            //case3:u 的邻接点 v 不在栈中,且访问过
            //说明 v 已经确定在某个强连通分量中，所以 u 的 low 不需要更新
        }
        if(dfn[u]==low[u])
        {
                  scc_cnt++;
            while(true)
                  {
                        int v=st.top();
```

```
                    st.pop();
                    bel[v]=scc_cnt;
                    if(v==u) break;
                }
            }
        };
        //图有可能不是强联通的
        for(int i=1;i<=n;i++)
        {
            if(dfn[i]==-1)
            {
                dfs(dfs,i);
            }
        }
          return {bel,scc_cnt};
}
int main()
{
    int T_start=clock();
    int n,m;cin>>n>>m;
    vector<vector<int>> mp(n+1);
      vector<int> val(n+1);
    for(int i=1;i<=n;i++) cin>>val[i];
    for(int i=0;i<m;i++)
    {
        int u,v;cin>>u>>v;
        mp[u].push_back(v);
    }
    auto [bel,cnt]=tarjan(mp,n);
      vector<vector<int>> mp2(cnt+1);
      vector<int> val2(cnt+1,0);
      vector<int> in(cnt+1,0);
      vector<int> dp(cnt+1,0);
      for(int i=1;i<=n;i++)
      {
            val2[bel[i]]+=val[i];
      }
    for(int i=1;i<=n;i++)
    {
        for(int v:mp[i])
        {
            if(bel[i]!=bel[v])
            {
                mp2[bel[i]].push_back(bel[v]);
                    in[bel[v]]++;
            }
        }
    }
      queue<int> q;
```

```
	for(int i=1;i<=cnt;i++)
	{
		if(in[i]==0) q.push(i),dp[i]=val2[i];
	}
	while(!q.empty())
	{
		int u=q.front();q.pop();
		for(int v:mp2[u])
		{
			dp[v]=max(dp[v],dp[u]+val2[v]);
			in[v]--;
			if(in[v]==0) q.push(v);
		}
	}
	cout<<*max_element(dp.begin()+1,dp.end())<<endl;
    return 0;
}
```

点双联通分量（BCC）

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
using namespace std;
vector<vector<int>> tarjan(vector<vector<int>> &mp,int n)
{
	//点双连通分量：无割点，且任意两点间至少有两条路径
	vector<int> low(n+1,-1),dfn(n+1,-1);
	//low:从当前点出发能到达的最早时间戳
	//dfn:当前点的时间戳
	vector<vector<int>> bccs;
	stack<int> st;
	int cnt=0;
	auto dfs=[&](auto dfs,int u,int fa)->void{
		int ch=0; //儿子数
		dfn[u]=low[u]=++cnt;
```

```
            st.push(u);
            for(auto v:mp[u])
            {
                  if(dfn[v]==-1)//case1:未访问
                  {
                        ch++;
                        dfs(dfs,v,u);
                        low[u]=min(low[u],low[v]);//更新low[u]
                        if((fa==-1&&ch>1)||(fa!=-1&&low[v]>=dfn[u]))//
是割点,v 以及他的被处理过的子树是一个bcc
                        {
                              vector<int> bcc;
                              while(1)
                              {
                                    int x=st.top();st.pop();
                                    bcc.push_back(x);
                                    if(x==v)break;//处理到v
                              }
                              bcc.push_back(u);//把割点也加入bcc:割点有可
能在多个bcc 中
                              bccs.push_back(bcc);
                        }
                  }
                  else if(v!=fa)//case2:已访问且不是父节点
                  {
                        low[u]=min(low[u],dfn[v]);//更新low[u]
                  }
            }
            // if(fa==-1&&ch==0) {
            //    bccs.push_back({u});
            // }
      };
      for(int i=1;i<=n;i++)
      {
            if(dfn[i]==-1)
            {
                  dfs(dfs,i,-1);
                  //处理剩下的bcc
                  vector<int> bcc;
                  while(!st.empty())
                  {
                        int x=st.top();st.pop();
                        bcc.push_back(x);
                  }
                  if(!bcc.empty()) bccs.push_back(bcc);
            }
      }
      return bccs;
```

```
}
//无向图中割点：删除该点后，图的 bcc 数增加
//一个图中割点的判断
//1.对于某个顶点 u，如果存在至少一个顶点 v（u 的儿子），使得 low[v]>=dfn[u]，
即只能回到祖先（到不了 dfn 更早的点），那么 u 点为割点。
//2.对于搜索的起始点，如果它的儿子数大于等于 2，那么它就是割点。
int main()
{
      int T_start=clock();
      int n,m;cin>>n>>m;
    vector<vector<int>> mp(n+1);
      vector<int> val(n+1);
    for(int i=0;i<m;i++)
    {
        int u,v;cin>>u>>v;
        mp[u].push_back(v);
            mp[v].push_back(u);
    }
      vector<vector<int>> bccs=tarjan(mp,n);
      cout<<bccs.size()<<endl;
      for(auto bcc: bccs)
      {
            cout<<bcc.size()<<' ';
            for(auto x: bcc)
            {
                  cout<<x<<" ";
            }
            cout<<endl;
      }
      return 0;
}
```

边双连通分量

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
```

```cpp
#include <unordered_map>
using namespace std;
vector<vector<int>> tarjan(vector<vector<pair<int,int>>> &mp,int n)
{
      //边双联通分量：无向图中，边双联通分量是指一个极大子图，删除该子图中的任意一条边，该子图仍然连通
    //连接边双联通分量的边称为桥
      vector<int> low(n+1,-1),dfn(n+1,-1);
      //low:从当前点出发能到达的最早时间戳
      //dfn:当前点的时间戳
      vector<vector<int>> dccs;
      stack<int> st; int cnt=0;
      auto dfs=[&](auto dfs,int u,int fre)->void{
        //fre:来时的边
            dfn[u]=low[u]=++cnt;
            st.push(u);
            for(auto [v,rev]:mp[u])
            {
                  if(dfn[v]==-1)//case1:未访问
                  {
                        dfs(dfs,v,rev);
                        low[u]=min(low[u],low[v]);//更新 low[u]
                  }
                  else if(rev!=(fre^1))//case2:已访问且不是该边的反边
                  {
                  //阻断向父节点更新的可能
                  //多重边可能有一种特殊的组合让 v!=fa 失效
                  //eg.1-2,1-2,2-3,2-3
                        low[u]=min(low[u],dfn[v]);//更新 low[u]
                  }
            }
        if(dfn[u]==low[u])//case3:u 的子树中不存在能到达 u 的祖先的边
        //u 的子树全为 dcc
            {
                  vector<int> dcc;
                  while(true){
                        int t=st.top();st.pop();
                        dcc.push_back(t);
                        if(t==u) break;
            }
                  dccs.push_back(dcc);
        }
      };
      for(int i=1;i<=n;i++)
      {
            if(dfn[i]==-1)
            {
```

```
                dfs(dfs,i,-1);
            }
        }
        return dccs;
}
int main()
{
        int T_start=clock();
        int n,m;cin>>n>>m;
    vector<vector<pair<int,int>>> mp(n+1);
        vector<int> val(n+1);
    int tot=0;
    for(int i=0;i<m;i++)
    {
        int u,v;cin>>u>>v;
        if(u==v) continue;
        mp[u].push_back({v,tot+1});
            mp[v].push_back({u,tot});
        tot+=2;//存各自的边的编号
    }
        vector<vector<int>> bccs=tarjan(mp,n);
        cout<<bccs.size()<<endl;
        for(auto bcc: bccs)
        {
            cout<<bcc.size()<<' ';
            for(auto x: bcc)
            {
                cout<<x<<" ";
            }
            cout<<endl;
        }
        return 0;
}
```

树链剖分(+线段树)

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
using namespace std;
#define int long long
const int N=1e5+5;
int dep[N],fa[N],hson[N],top[N],siz[N],dfn[N],rk[N],val[N];
int mod;
class SegTree{
    public:
        struct Node
        {
            int sum;
            int s,e;
            int lazy=0;
            Node* lt;
            Node* rt;
            Node(int sum,int s,int e):s(s),e(e),sum(sum),lt(nullptr),rt
(nullptr){}
        };
        Node* root;
        Node* buildtree(vector<int> &nums,int l,int r)
        {
            if(l>r) return nullptr;
            if(l==r) return new Node(nums[l],l,l);
            int mid=(l+r)>>1;
            Node* root=new Node(0,l,r);
            Node* lc=buildtree(nums,l,mid);
            Node* rc=buildtree(nums,mid+1,r);
            if(lc) root->lt=lc,root->sum=(root->sum+lc->sum)%mod;
            if(rc) root->rt=rc,root->sum=(root->sum+rc->sum)%mod;
            return root;
        }
        void init(vector<int>nums)
```

```
        {
            root=buildtree(nums,0,nums.size()-1);
            return;
        }
        void taglazy(Node* root,int val)
        {
            if(root==nullptr) return;
                  val%=mod;
            root->lazy=(root->lazy+val)%mod;
            root->sum=(root->sum+(root->e-root->s+1)%mod*val)%mod;
        }
        void pushdown(Node* root)
        {
            if(!root) return ;
            if(root->lazy)
            {
                taglazy(root->lt,root->lazy);
                taglazy(root->rt,root->lazy);
                root->lazy=0;
            }
        }
        void update(Node* root,int l,int r,int val)
        {
            if(!root) return ;
            if(root->s>r||root->e<l) return ;
            if(root->s>=l&&root->e<=r)
            {
                taglazy(root,val);
                return;
            }
            pushdown(root);
            update(root->lt,l,r,val);
            update(root->rt,l,r,val);
            root->sum=((root->lt?root->lt->sum:0)+(root->rt?root->rt->s
um:0))%mod;
            return ;
        }
        void update(int l,int r,int val)
        {
            update(root,l,r,val);
            return ;
        }
        int query(Node* root,int l,int r)
        {
            pushdown(root);
            if(!root) return 0;
            if(root->s>r||root->e<l) return 0;
            if(root->s>=l&&root->e<=r) return root->sum;
            return query(root->lt,l,r)+query(root->rt,l,r);
```

```
        }
        int query(int l,int r)
        {
            return query(root,l,r);
        }
};
class cutTree
{
      //树链剖分，把树剖分成若干条链，每条链上维护一个线段树
      //可以支持链上修改和查询，也可以支持树上修改和查询
      //还可以求 lca
      //重链剖分有一个重要的性质：对于节点数为 n 的树，从任意节点向上走到根节点，经过的轻边数量不超过 logn。这是因为，如果一个节点连向父节点的边是轻边，
      //就必然存在子树不小于它的兄弟节点，那么父节点对应子树的大小一定超过该节点的两倍(由 dfs1 可得)。每经过一条轻边，子树大小就翻倍，所以最多只能经过 log n 条。
      public:
            int n,tot,s;
            //s:根节点
            vector<vector<int>> tree;
            //dep:树深,fa:父节点,hson:i 的重儿子,top:重链顶端,siz:子树大小,dfn:dfs 序,rk:dfs 序对应的节点
            SegTree seg;
            void dfs1(int u,int f)
            {
                  //cntt++;cout<<cntt<<endl;
                  dep[u]=dep[f]+1;//更新树深
                  fa[u]=f;siz[u]=1;
                  for(auto v:tree[u])
                  {
                        if(v==f)continue;
                        dfs1(v,u);
                        siz[u]+=siz[v];
                        if(hson[u]==-1||siz[v]>siz[hson[u]]) hson[u]=v;
                        //u 的重儿子是所有子树大小最大的儿子
                  }
            }
            void dfs2(int u)
            {
                  dfn[u]=++tot;rk[tot]=u;
                  //优先访问重儿子,保证重链顶端的 dfn 最小
                  if(hson[u]!=-1)
                  {
                        top[hson[u]]=top[u];
                        //重儿子的 top 是它所在重链的顶端
                        dfs2(hson[u]);
                  }
```

```
	for(auto v:tree[u])
	{
		if(v==fa[u]||v==hson[u])//跳过父节点和重儿子
			continue;
		top[v]=v;//轻儿子的top是自己
		dfs2(v);
	}
}
void init()
{
	tot=0;
	dfs1(s,0);
	dfs2(s);
}
int lca(int u,int v)
{
	while(top[u]!=top[v])//不在同一条重链上
	{
		if(dep[top[u]]<dep[top[v]])swap(u,v);
		u=fa[top[u]];
		//链头深度大的往上跳
	}
	return dep[u]<dep[v]?u:v;
}
int queryPath(int u,int v)
{
	int ans=0;
	while(top[u]!=top[v])//遍历所有的边
	{
		if(dep[top[u]]<dep[top[v]])swap(u,v);
		ans=(ans+seg.query(dfn[top[u]],dfn[u]))%mod;
		u=fa[top[u]];
	}
	if(dep[u]>dep[v])swap(u,v);
	ans=(ans+seg.query(dfn[u],dfn[v]))%mod;
	return ans;
}
void updatePath(int u,int v,int val)
{
	while(top[u]!=top[v])
	{
		if(dep[top[u]]<dep[top[v]])swap(u,v);
		seg.update(dfn[top[u]],dfn[u],val);
		u=fa[top[u]];
	}
	if(dep[u]>dep[v])swap(u,v);
	seg.update(dfn[u],dfn[v],val);
}
void updateSub(int u,int val)
```

```
        {
            //子树的dfn 一定是连续的
            seg.update(dfn[u],dfn[u]+siz[u]-1,val);
        }
        int querySub(int u)
        {
            return seg.query(dfn[u],dfn[u]+siz[u]-1);
        }
    cutTree(int n,vector<vector<int>> tree,int s):n(n),tree(tree),s(s)
    {
        for(int i=0;i<=n;i++)
        {
            dep[i]=0;fa[i]=-1;hson[i]=-1;top[i]=-1;
            siz[i]=0;dfn[i]=-1;rk[i]=-1;
        }
        top[s]=s;init(); vector<int> inf(n+1,0);
        for(int i=1;i<=n;i++)inf[dfn[i]]=val[i]%mod;
        seg.init(inf);
    }
};
signed main()
{
    int T_start=clock();
    ios::sync_with_stdio(false);
    cin.tie(0);
    //freopen("in.txt","r",stdin);
    //freopen("out.txt","w",stdout);
    int n,m,s;
    cin>>n>>m>>s>>mod;
    vector<vector<int>> tree(n+1);
    for(int i=1;i<=n;i++)
    {
        cin>>val[i];
    }
    for(int i=1;i<n;i++)
    {
        int u,v;
        cin>>u>>v;
        tree[u].push_back(v);
        tree[v].push_back(u);
    }
    cutTree ct(n,tree,s);
    while(m--)
    {
        int op,x,y,z;
        cin>>op;
        if(op==1)
        {
            cin>>x>>y>>z;
```

```
                ct.updatePath(x,y,z);
            }
            else if(op==2)
            {
                cin>>x>>y;
                cout<<ct.queryPath(x,y)%mod<<endl;
            }
            else if(op==3)
            {
                cin>>x>>y;
                ct.updateSub(x,y);
            }
            else if(op==4)
            {
                cin>>x;
                cout<<ct.querySub(x)%mod<<endl;
            }
        }
        int T_end=clock();
        //cout<<"time: "<<(double)(T_end-T_start)/CLOCKS_PER_SEC<<"s"<<en
dl;
        return 0;
}
```

分层图

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
using namespace std;
int read()
{
    int s=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9')
    {
        if(ch=='-') f=-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9')
    {
        s=(s<<3)+(s<<1)+ch-'0';
        ch=getchar();
    }
    return s*f;
}
inline void write(int x)
{
    static int sta[35];
    int top=0;
    if(x<0&&x!=-2147483648) {putchar('-');x=-x;}
    if(x==-2147483648) {printf("-2147483648");return;}
    do{
      sta[top++]=x%10, x/=10;
      }while(x);
    while(top) putchar(sta[--top]+48);
}
int dij(vector<vector<pair<int,int>>>& mp,int s,int n,int t,int kk)
{
    vector<int> vis((kk+1)*n+1,0x7fffffff);
```

```
    vis[s]=0;
    priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,
int>>> pq;
    pq.push({0,s});
    while(!pq.empty())
    {
        auto [val,k]=pq.top();
        pq.pop();
        if(val>vis[k]) continue;
        for(auto i:mp[k])
        {
            auto [v,w]=i;
            if(vis[v]>vis[k]+w)
            {
                vis[v]=vis[k]+w;
                pq.push({vis[v],v});
            }
        }
    }
    int ans=0x7fffffff;
    for(int i=0;i<=kk;i++)
    {
        //i 表示免费次数
        ans=min(ans,vis[i*n+t]);
    }
    return ans;
}
int main()
{
    //分层图：解决 k 次免费（有代价）最短路问题
    int T_start=clock();
    int n=read(),m=read(),k=read();
    int s=read(),t=read();
    vector<vector<pair<int,int>>> mp((k+1)*n+1);
    while(m--)
    {
        int u,v,w;
        u=read(),v=read(),w=read();
        for(int i=0;i<=k;i++)
        {
            mp[i*n+u].push_back({i*n+v,w});
            mp[i*n+v].push_back({i*n+u,w});
            if(i!=k)
            {
                mp[i*n+u].push_back({(i+1)*n+v,0});
                mp[i*n+v].push_back({(i+1)*n+u,0});//分层图连边
            }
        }
    }
```

```
    cout<<dij(mp,s,n,t,k)<<endl;
    return 0;
}
```

2-sat

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
#include <numeric>
using namespace std;
pair<vector<int>,int> tarjan(vector<vector<int>> &mp,int n)
{
      vector<int> bel(n+1,-1);//bel[i]:i 属于哪个强连通分量
    vector<int> dfn(n+1,-1),low(n+1,-1);
    stack<int> st;int cnt=0,scc_cnt=0;
    auto dfs=[&](auto dfs,int u)->void{
        dfn[u]=low[u]=++cnt; //时间戳+1
        st.push(u); //inst[u]=1; //入栈
        for(int v:mp[u])
        {
            if(dfn[v]==-1)//case1:u 的邻接点 v 未被访问过
            {
                dfs(dfs,v);
                low[u]=min(low[u],low[v]);
            }
                  else if(bel[v]==-1)//v 所属的强连通分量还未被确定（等价于
case2）
                  {
                        low[u]=min(low[u],dfn[v]);
                  }
            //case3:u 的邻接点 v 不在栈中,且访问过
            //说明 v 已经确定在某个强连通分量中，所以 u 的 low 不需要更新
        }
        if(dfn[u]==low[u])
        {
                  scc_cnt++;
            while(true)
                  {
```

```
                    int v=st.top();
                    st.pop();
                    bel[v]=scc_cnt;
                    if(v==u) break;
                }
            }
        };
        //图有可能不是强联通的
        for(int i=1;i<=n;i++)
        {
            if(dfn[i]==-1)
            {
                dfs(dfs,i);
            }
        }
        return {bel,scc_cnt};
}
int main()
{
        int T_start=clock();
        //freopen("in.txt","r",stdin);
        int n,m;cin>>n>>m;
        vector<vector<int>> mp(2*n+1);
        for(int i=0;i<m;i++)
        {
            int u,b1,v,b2;
            cin>>u>>b1>>v>>b2;
            //u=b1 or v=b2
            //=>u!=b1->v=b2 and v!=b2->u=b1
            mp[u+(!b1)*n].push_back(v+b2*n);
            mp[v+(!b2)*n].push_back(u+b1*n);
            //u=b1-> u+b1*n x
        }
        auto [bel,scc_cnt]=tarjan(mp,2*n);
        vector<int> ans(n+1,-1);
        int flag=1;
        for(int i=1;i<=n;i++)
        {
            if(bel[i]==bel[i+n]) {flag=0;break;}
            else ans[i]=bel[i]>bel[i+n];
        }
        //此处处理的是i的正确性
        //当bel[u==0]>bel[u==1]时,u==0的拓扑序小,i应当被赋值为false
        //因为i->!i为永真式的前提为i=0
        //此处i的含义是命题变元i的取值=0
        //所以ans[i]=1
        if(flag)
        {
```

```
            cout<<"POSSIBLE"<<endl;
            for(int i=1;i<=n;i++)
            {
                  cout<<ans[i]<<" ";
            }
            cout<<endl;
      }
      else cout<<"IMPOSSIBLE"<<endl;
      return 0;
}
//2-sat
//处理n个命题变元的赋值问题，形式上判断形如(p->q) and (!p->q)是否可永真赋值
//即判断是否存在一种赋值使得p->q 和!p->q 同时为真
//很显然若p->q,q->p 均成立,则p,q 在一个scc 里
```

差分约束

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
using namespace std;
int flag=0;
int read()
{
    int s=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9')
    {
        if(ch=='-') f=-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9')
    {
        s=(s<<3)+(s<<1)+ch-'0';
        ch=getchar();
    }
    return s*f;
}
inline void write(int x)
{
    static int sta[35];
    int top=0;
    if(x<0&&x!=-2147483648) {putchar('-');x=-x;}
    if(x==-2147483648) {printf("-2147483648");return;}
    do{
      sta[top++]=x%10, x/=10;
      }while(x);
    while(top) putchar(sta[--top]+48);
}
vector<int> SPFA(vector<vector<pair<int,int>>>& mp,int s,int n)
{
```

```
    vector<int> dis(n+1,0x7fffffff);
    vector<int> vis(n+1,0);
    vector<int> cnt(n+1,0);
    dis[s]=0;queue<int> q;
    q.push(s);vis[s]=1;cnt[s]=1;
    while(!q.empty())
    {
        int u=q.front();
        q.pop();vis[u]=0;
        for(auto [v,w]:mp[u])
        {
            if(dis[v]>dis[u]+w)//松弛
            {
                dis[v]=dis[u]+w;
                cnt[v]=cnt[u]+1;
                if(cnt[v]>=n+1)
                {
                    flag=1;
                    return vector<int>(n+1,-1);
                }
                if(!vis[v])
                {
                    q.push(v);
                    vis[v]=1;
                }
            }
        }
    }
    return dis;
}
int main()
{
    int T_start=clock();
    int n=read(),m=read();
    vector<vector<pair<int,int>>> mp(n+1);
    for(int i=1;i<=m;i++)
    {
        int v=read(),u=read(),w=read();
        mp[u].push_back(make_pair(v,w));
    }
    for(int i=1;i<=n;i++)
    {
        mp[0].push_back(make_pair(i,0));
    }
    vector<int>ans=SPFA(mp,0,n);
    if(flag==1)
    {
        printf("NO\n");
    }
```

```
    else
    {
        //printf("YES\n");
        for(int i=1;i<=n;i++)
        {
            printf("%d ",ans[i]);
        }
        printf("\n");
    }
    return 0;
}
//对一个差分约束系统，判断是否存在一组解，使得所有约束条件都成立。
//ex. x1-x2<=3
//    x2-x3<=-2
//    x1-x3<=1
//将 xn 看作超级源点（到所有点的权值为 w=0）到 n 的最短路
//那么第一个式子的意义就是 x1<=x2+3,0 到 1 的最短路<=3+0 到 2 的最短路
//在图上的意义就是建 2->1 的边权为 3 的边，0->1,0->2 的边权为 0 的边
//0->1,0->2 的边权为 0 的边也是添加了以下条件
//x1-x0<=0
//x2-x0<=0
//x0=0
//那么整个系统就转化为了一张图
//求 xn 即求 0 到 n 的最短路，如果存在负环，则无解，否则有解
//负环还原的形式为
//x1-x2<=-1...1
//x2-x3<=-4...2
//x3-x1<=-5...3
//1+2+3->0<=-10,不成立
//还有结论，设定 w 即求 x1,x2..xn<=w 的最大解
//如果差分约束系统换换不等号，求最长路，spfa 改一下即可

//结论形式证明
//假设 X0 是定死的；X1 到 Xn 在满足所有约束的情况下可以取到的最大值分别为 M1、M2、……、Mn（当然我们不知道它们的值是多少）；解出的源点到每个点的最短路径长度为 D1、D2、……、Dn。
//基本的 Bellman-Ford 算法是一开始初始化 D1 到 Dn 都是无穷大。然后检查所有的边对应的三角形不等式，一但发现有不满足三角形不等式的情况，则更新对应的 D 值。最后求出来的 D1 到 Dn 就是源点到每个点的最短路径长度。
//如果我们一开始初始化 D1、D2、……、Dn 的值分别为 M1、M2、……、Mn，则由于它们全都满足三角形不等式（我们刚才已经假设 M1 到 Mn 是一组合法的解），则 Bellman-Ford 算法不会再更新任合 D 值，则最后得出的解就是 M1、M2、……、Mn。
//好了，现在知道了，初始值无穷大时，算出来的是 D1、D2、……、Dn；初始值比较小的时候算出来的则是 M1、M2、……、Mn。大家用的是同样的算法，同样的计算过程，总不可能初始值大的算出来的结果反而小吧。所以 D1、D2、……、Dn 就是 M1、M2、……、Mn。
```

点分治

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
#include <unordered_set>
#include <numeric>
using namespace std;

int main()
{
    int T_start=clock();
    //freopen("in.txt","r",stdin);
    //freopen("out.txt","w",stdout);
    int n,m;cin>>n>>m;
    vector<vector<pair<int,int>>> tr(n+1);
    for(int i=1;i<n;i++)
    {
        int u,v,w;
        cin>>u>>v>>w;
        tr[u].push_back({v,w});
        tr[v].push_back({u,w});
    }
    vector<int> siz(n+1,0),q(m+1),vis(n+1,0),ans(m+1,0);
    for(int i=1;i<=m;i++) cin>>q[i];
    auto getsz=[&](auto getsz,int u,int p=-1)->int
    {
        siz[u]=1;
        for(auto [v,w]:tr[u])
        {
            if(v==p||vis[v])continue;
            siz[u]+=getsz(getsz,v,u);
        }
        return siz[u];
    };//统计以 u 为根的子树大小
    auto getrt=[&](auto getrt,int u,int p=-1,int sizrt)->int
```

```
    {
        for(auto [v,w]:tr[u])
        {
            if(v==p||vis[v])continue;
            if(siz[v]>sizrt/2)return getrt(getrt,v,u,sizrt);
        }
        return u;
    };//寻找重心
    //重心：对于一棵树，如果存在一个顶点，其子树中最大的子树大小不超过整棵树大小的一半，则称该顶点为这棵树的重心。
    auto clac=[&](auto clac,int uu,int dis,int p=-1,vector<int>& tpd)->void
    {
        tpd.push_back(dis);
        for(auto [vv,ww]:tr[uu])
        {
            if(vv==p||vis[vv])continue;
            clac(clac,vv,dis+ww,uu,tpd);
        }
    };
    auto div=[&](auto div,int u)->void{
        vis[u]=1;
        unordered_set<int> s{0};
        for(auto [v,w]:tr[u])
        {
            if(vis[v])continue;
            vector<int> tpd;
            clac(clac,v,w,u,tpd);
            for(auto d:tpd)
            {
                for(int i=1;i<=m;i++)
                {
                    if(!ans[i]&&d<=q[i]&&s.find(q[i]-d)!=s.end())
                    {
                        ans[i]=1;
                    }
                }
            }
            for(auto d:tpd)s.insert(d);
        }
        for(auto [v,w]:tr[u])
        {
            //用重心划分 u 的子树
            if(vis[v])continue;
            getsz(getsz,v);
            int subrt=getrt(getrt,v,-1,siz[v]);
            div(div,subrt);
        }
    };//处理以 u 为根的子树
```

```
    getsz(getsz,1);
    int rt=getrt(getrt,1,-1,siz[1]);
    div(div,rt);
    for(int i=1;i<=m;i++)
    {
        if(ans[i])cout<<"AYE\n";
        else cout<<"NAY\n";
    }
    return 0;
}
//淀粉质：把树上路径问题转化为子树分治问题
//把树按重心划分，那么树高（或树的大小）不超过 n/2，递归深度不超过 logn(最坏：
退化为链），于是可以暴力处理子树
//根据实现方式的不同，时间复杂度可以做到 O(nlogn)或 O(nlog^2n)
```

二分图染色

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
using namespace std;
struct node{
    int v;
    int w;
};
vector<node> mp[20005];
bool vis[20005]={false};
int dyed[20005]={0};
int Data[100005];
int read()
{
    int s=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9')
    {
        if(ch=='-') f=-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9')
    {
        s=(s<<3)+(s<<1)+ch-'0';
        ch=getchar();
    }
    return s*f;
}
inline void write(int x)
{
    static int sta[35];
    int top=0;
    if(x<0&&x!=-2147483648) {putchar('-');x=-x;}
    if(x==-2147483648) {printf("-2147483648");return;}
    do{
      sta[top++]=x%10, x/=10;
```

```
      }while(x);
    while(top) putchar(sta[--top]+48);
}
bool dye(int start,int mid)
{
    queue<int> q;
    q.push(start);
    vis[start]=1;dyed[start]=1;
    while(!q.empty())
    {
        int temp=q.front();
        q.pop();
        for(auto i:mp[temp])
        {
            if(i.w>=mid)
            {
                if(!vis[i.v])
                {
                    q.push(i.v);
                    vis[i.v]=true;
                    dyed[i.v]=3-dyed[temp];
                }
                else if(dyed[i.v]==dyed[temp]) return false;
            }
        }
    }
    return true;
}
bool isBinGraph(int n,int mid)
{
    memset(vis,0,sizeof(vis));
    memset(dyed,0,sizeof(dyed));
    for(int i=1;i<=n;i++)
    {
        if(!vis[i])
        {
            if(!dye(i,mid)) return false;
        }
    }
    return true;
}
int main()
{
    int T_start=clock();
    freopen("in.txt","r",stdin);
    // freopen("out.txt","w",stdout);
    int n=read(),m=read();
    for(int i=0;i<m;i++)
    {
```

```
        int u=read(),v=read(),w=read();
        mp[u].push_back({v,w});
        mp[v].push_back({u,w});
        Data[i]=w;
    }
    sort(Data,Data+m);
    // for(int i=0;i<m;i++)
    // {
    //     cout<<Data[i]<<endl;
    // }
    if(isBinGraph(n,0))
    {
        cout<<"0"<<endl;
    }
    else
    {
        int l=0,r=m;
        while(l<=r)
        {
            int mid=(l+r)>>1;
            if(!isBinGraph(n,Data[mid])) l=mid+1;
            else r=mid-1;
        }
        cout<<Data[r]<<endl;
    }
    return 0;
}
```

最大流

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
#include <numeric>
#include <functional>
using namespace std;
#define int long long
class maxflow{
    public:
        struct node
        {
            int to,cap,id;
        };
        vector<vector<node>> mp;
        vector<int> dep,cur;//dep:层次图, cur:当前弧优化
        int n,m,s,t;
        maxflow(int n,int m,int s,int t,vector<array<int,3>>& eds):
        mp(n+1),n(n),m(m),s(s),t(t),dep(n+1),cur(n+1){
            //u->v capacity
            for(auto [u,v,cap]:eds){
                int uid=mp[u].size();
                int vid=mp[v].size();
                mp[u].push_back({v,cap,vid});
                mp[v].push_back({u,0,uid});
                //建反边
            }
        }
        bool bfs(){
            fill(dep.begin(),dep.end(),-1);
            fill(cur.begin(),cur.end(),0);
            queue<int> q;
            q.push(s);
            dep[s]=0;
            while(!q.empty()){
```

```
                int u=q.front();
                q.pop();
                for(auto [v,cap,id]:mp[u]){
                    if(cap>0&&dep[v]==-1){
                        dep[v]=dep[u]+1;
                        q.push(v);
                    }
                }
            }
            return dep[t]!=-1;
        }
        int dfs(int u,int lim)//到u点的最大流量lim
        {
            if(u==t) return lim;
            int sum=0;//u点流出的流量
            for(int &i=cur[u];i<mp[u].size();i++){
                //当前弧优化,考虑u->v有重边,那么这个优化会使
                //被榨干过的v的出边不再被访问
                auto [v,cap,id]=mp[u][i];
                if(cap>0&&dep[v]==dep[u]+1){
                    int f=dfs(v,min(lim,cap));
                    mp[u][i].cap-=f;
                    mp[v][id].cap+=f;
                    sum+=f;
                    lim-=f;
                    if(lim==0) break;
                }
            }
            if(sum==0) dep[u]=-1;//无增广路
            return sum;
        }
        int dinic(){
            int res=0;
            while(bfs()){
                res+=dfs(s,INT_MAX);
            }
            return res;
        }
};
signed main()
{
    int T_start=clock();
    int n,m,s,t;
    cin>>n>>m>>s>>t;
    vector<array<int,3>> eds(m);
    for(auto &[u,v,cap]:eds){
        cin>>u>>v>>cap;
    }
    maxflow mf(n,m,s,t,eds);
```

```
    cout<<mf.dinic()<<endl;
    return 0;
}
//最大流，解决从有向图源点到汇点的最大流量问题(假定源点流量无限)
//dinic 算法，时间复杂度O(n^2*m)
//增广路：是从源点到汇点的路径，其上所有边的残余容量均大于0
//初级思路：贪心选择所有增广路，然后更新边权，引入反向边进行反悔贪心
//基本思路：每次bfs 把图变成一个带层数的DAG(限制 dfs 深度)
//然后找到极大增广流量，更新图，重复上述过程
```

最小费用最大流(dinic)

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
#include <numeric>
#include <functional>
using namespace std;
#define int long long
class Mcmf{
public:
    struct node{
        int to;
        int cap;
        int cost;
        int rev;
    };
    int n,s,t;
    int maxf=0,minc=0;
    const int INF=1e18;
    vector<vector<node>> mp;
    vector<int> dis,cur,inq,vis;
    Mcmf(int n,int s,int t,vector<array<int,4>>& eds):
    n(n),s(s),t(t),mp(n+1),dis(n+1),
    cur(n+1),inq(n+1,0),vis(n+1,0){
        for(auto [u,v,cap,w]:eds){
            int uid=mp[u].size();
            int vid=mp[v].size();
            mp[u].push_back({v,cap,w,vid});
            mp[v].push_back({u,0,-w,uid});
            //反边的费用是负的
        }
    }

    bool spfa(){
        fill(dis.begin(),dis.end(),INF);
```

```
    fill(inq.begin(),inq.end(),0);
    deque<int> q;dis[s]=0,inq[s]=1;
    q.push_back(s);
    while(!q.empty()){
        int u=q.front();q.pop_front();
        inq[u]=0;
        for(auto [v,cap,w,rev]:mp[u]){
            if(cap>0&&dis[u]+w<dis[v]){
                dis[v]=dis[u]+w;
                if(!inq[v]){
                    if(!q.empty()&&dis[v]<dis[q.front()]){
                        q.push_front(v);
                    }else{
                        q.push_back(v);
                    }
                    inq[v]=1;
                }
            }
        }
    }
    return dis[t]!=INF;
}

int dfs(int u,int f){
    if(u==t)return f;
    vis[u]=1;
    int res=0;
    for(int &i=cur[u];i<mp[u].size();i++){
        auto [v,cap,w,rev]=mp[u][i];
        if(!vis[v]&&cap>0&&dis[u]+w==dis[v]){
            int tmp=dfs(v,min(f,cap));
            f-=tmp;
            res+=tmp;
            mp[u][i].cap-=tmp;
            mp[v][rev].cap+=tmp;
            minc+=tmp*w;
            if(!f)break;
        }
    }
    vis[u]=0;
    return res;
}

void dinic(){
    while(spfa()){
        fill(vis.begin(),vis.end(),0);
        fill(cur.begin(),cur.end(),0);
        maxf+=dfs(s,INF);
    }
```

```
    }

};
signed main()
{
    int T_start=clock();
    int n,m,s,t;
    cin>>n>>m>>s>>t;
    vector<array<int,4>> eds;
    for(int i=0;i<m;i++){
        int u,v,cap,w;
        cin>>u>>v>>cap>>w;
        eds.push_back({u,v,cap,w});
    }
    Mcmf mcmf(n,s,t,eds);
    mcmf.dinic();
    cout<<mcmf.maxf<<" "<<mcmf.minc<<endl;
    return 0;
}
//最小费用最大流，O(nmf)
//基本思路：找到最短增广路，然后增广，直到找不到为止
//最短增广路：spfa(slf 优化)，每次找到最短路径，然后更新，直到找不到为止
//增广：用 dinic 思路在最短路上多路增广
```

最小费用最大流(dinic，浮点)

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
#include <unordered_map>
#include <numeric>
#include <functional>
#include <iomanip>
using namespace std;
#define int long long
struct ta{
    int u,v;
```

```
    int cap;
    double w;
};
class Mcmf{
public:
    struct node{
        int to;
        int cap;
        double cost;
        int rev;
    };
    int n,s,t;
    int maxf=0;double minc=0;
    const int INF=1e18;
    vector<vector<node>> mp;
    vector<int> cur,inq,vis;
    vector<double> dis;
    Mcmf(int n,int s,int t,vector<ta>& eds):
    n(n),s(s),t(t),mp(n+1),dis(n+1),
    cur(n+1),inq(n+1,0),vis(n+1,0){
        for(auto [u,v,cap,w]:eds){
            int uid=mp[u].size();
            int vid=mp[v].size();
            mp[u].push_back({v,cap,w,vid});
            mp[v].push_back({u,0,-w,uid});
            //反边的费用是负的
        }
    }

    bool spfa(){
        fill(dis.begin(),dis.end(),INF);
        fill(inq.begin(),inq.end(),0);
        deque<int> q;dis[s]=0,inq[s]=1;
        q.push_back(s);
        while(!q.empty()){
            int u=q.front();q.pop_front();
            inq[u]=0;
            for(auto [v,cap,w,rev]:mp[u]){
                if(cap>0&&dis[u]+w+1e-10<dis[v]){
                    dis[v]=dis[u]+w;
                    if(!inq[v]){
                        if(!q.empty()&&dis[v]+1e-10<dis[q.front()]){
                            q.push_front(v);
                        }else{
                            q.push_back(v);
                        }
                        inq[v]=1;
                    }
                }
```

```
            }
        }
        return dis[t]!=INF;
    }

    int dfs(int u,int f){
        if(u==t)return f;
        vis[u]=1;
        int res=0;
        for(int &i=cur[u];i<mp[u].size();i++){
            auto [v,cap,w,rev]=mp[u][i];
            if(!vis[v]&&cap>0&&abs(dis[u]+w-dis[v])<1e-10){
                int tmp=dfs(v,min(f,cap));
                f-=tmp;
                res+=tmp;
                mp[u][i].cap-=tmp;
                mp[v][rev].cap+=tmp;
                minc+=1.0*tmp*w;
                if(!f)break;
            }
        }
        vis[u]=0;
        return res;
    }

    void dinic(){
        while(spfa()){
            fill(vis.begin(),vis.end(),0);
            fill(cur.begin(),cur.end(),0);
            maxf+=dfs(s,INF);
        }
    }

};
signed main()
{
    int T_start=clock();
    int n;cin>>n;
    vector<array<int,2>> xy(n+1);
    for(int i=1;i<=n;i++){
        cin>>xy[i][0]>>xy[i][1];
    }
    int s=0,t=2*n+1;
    vector<ta> eds;
    for(int i=1;i<=n;i++)
    {
        eds.push_back({s,i,2,0});
    }
    for(int i=1;i<=n;i++)
```

```
    {
        eds.push_back({i+n,t,1,0});
    }
    auto dis=[&](int u,int v)->double{
        return sqrt(1.0*(xy[u][0]-xy[v][0])*(xy[u][0]-xy[v][0])+1.0*(xy
[u][1]-xy[v][1])*(xy[u][1]-xy[v][1]));
    };
    for(int u=1;u<=n;u++)
    {
        for(int v=1;v<=n;v++)
        {
            if(xy[u][1]>xy[v][1])
            {
                eds.push_back({u,v+n,1,dis(u,v)});
            }
        }
    }
    Mcmf mc(2*n+2,s,t,eds);
    mc.dinic();
    if(mc.maxf==n-1) cout<<fixed<<setprecision(10)<<mc.minc<<endl;
    else cout<<-1<<endl;
    return 0;
}
//最小费用最大流，O(nmf)
//基本思路：找到最短增广路，然后增广，直到找不到为止
//最短增广路：spfa(slf 优化)，每次找到最短路径，然后更新，直到找不到为止
//增广：用dinic 思路在最短路上多路增广
//浮点数比较
//a==b->abs(a-b)<1e-10
//a<b->a+eps<b
//a>b->a>b+eps
```

最小费用最大流(原始对偶)

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
#include <array>
```

```cpp
#include <unordered_map>
#include <numeric>
#include <functional>
using namespace std;
#define int long long
class Mcmf{
public:
    struct node{
        int to;
        int cap;
        int cost;
        int rev;
    };
    int n,s,t;
    int maxf=0,minc=0;
    const int INF=1e18;
    vector<vector<node>> mp;
    vector<int> dis,h,prev,previd;
    Mcmf(int n,int s,int t,vector<array<int,4>>& eds):
    n(n),s(s),t(t),mp(n+1),dis(n+1),
    h(n+1,INF),prev(n+1),previd(n+1){
        for(auto [u,v,cap,w]:eds){
            int uid=mp[u].size();
            int vid=mp[v].size();
            mp[u].push_back({v,cap,w,vid});
            mp[v].push_back({u,0,-w,uid});
            //反边的费用是负的
        }
    }
    bool dijk(){
        fill(dis.begin(),dis.end(),INF);
        dis[s]=0;
        priority_queue<pair<int,int>,vector<pair<int,int>>,greater<pair
<int,int>>> pq;
        pq.push({0,s});
        while(!pq.empty()){
            auto [d,u]=pq.top();
            pq.pop();
            if(dis[u]<d) continue;
            for(int i=0;i<mp[u].size();i++){
                auto [v,cap,w,rev]=mp[u][i];
                int cost=w+h[u]-h[v];
                if(cap>0&&dis[u]+cost<dis[v]){
                    dis[v]=dis[u]+cost;
                    prev[v]=u;//记录前驱
                    previd[v]=i;//记录当前弧
                    pq.push({dis[v],v});
                }
            }
```

```
    }
    return dis[t]<INF;
}

void SPFA(){
    h[s]=0;
    queue<int> q;
    vector<bool> inq(n+1,0);
    q.push(s),inq[s]=1;
    while(!q.empty()){
        int u=q.front();
        q.pop();
        inq[u]=0;
        for(auto [v,cap,w,rev]:mp[u]){
            if(cap>0&&h[u]+w<h[v]){
                h[v]=h[u]+w;
                if(!inq[v]){
                    q.push(v);
                    inq[v]=1;
                }
            }
        }
    }
}

void PD(){
    SPFA();
    while(dijk())
    {
        //now dis(u-v)=dis(u,v)(real)+h[u]-h[v]
        //when u=0, dis(u,v)=dis(u,v)(real)-h[v]
        int d=INF;
        for(int i=t;i!=s;i=prev[i])
        {
            d=min(d,mp[prev[i]][previd[i]].cap);
        }
        //计算增广路径上的最小流量
        maxf+=d;
        minc+=d*(dis[t]+h[t]);
        for(int i=t;i!=s;i=prev[i])
        {
            mp[prev[i]][previd[i]].cap-=d;
            mp[i][mp[prev[i]][previd[i]].rev].cap+=d;
        }
        //更新残余网络
        for(int i=1;i<=n;i++)
        {
            if(dis[i]<INF)
            {
```

```
                    h[i]+=dis[i];
                }
            }
        }
    }
};
signed main()
{
    int T_start=clock();
    int n,m,s,t;
    cin>>n>>m>>s>>t;
    vector<array<int,4>> eds;
    for(int i=0;i<m;i++){
        int u,v,cap,w;
        cin>>u>>v>>cap>>w;
        eds.push_back({u,v,cap,w});
    }
    Mcmf mcmf(n,s,t,eds);
    mcmf.PD();
    cout<<mcmf.maxf<<" "<<mcmf.minc<<endl;
    return 0;
}
//Primal-Dual 原始对偶算法,O(F*E*logE)
//利用johnson最短路，将每条边的权值加上一个常数，使得每条边的权值非负，从而
可以使用dijkstra算法
```

Hash

双 Hash

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
using namespace std;
const int base1=27,base2=27;
const int hash_mod1=100663319,hash_mod2=402653189;
/*hash mod number 53 97 193 389 769 1543 3079 6151 12289 24593 49157 98
317 196613 393241 786433 1572869 3145739 6291469 12582917 25165843 5033
1653 100663319 402653189 805306457 1610612741 (1e9+7,1e9+9)*/
//char ss[10][10010];
struct Data
{
    long long hash1;
    long long hash2;
}a[10005];
long long make_string_hash1(char s[])
{
    register long long ans=0;
    register int len=strlen(s);
    for(register int i=0;i<len;i++)
    {
        ans=(ans*base1+(long long)(s[i]))%hash_mod1;
    }
    return ans;
}
long long make_string_hash2(char s[])
{
    register long long ans=0;
    register int len=strlen(s);
    for(register int i=0;i<len;i++)
    {
        ans=(ans*base2+(long long)(s[i]))%hash_mod2;
    }
    return ans;
```

```
}//求子串哈希值 hash=((hash[r]-hash[l-1]*mod^(r-l+1))%mod+mod)%mod
int main()
{
    int T_start=clock();

    return 0;
}
```

hash 表

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
using namespace std;
/*hash mod number 53 97 193 389 769 1543 3079 6151 12289 24593 49157 98
317 196613 393241 786433 1572869 3145739 6291469 12582917 25165843 5033
1653 100663319 402653189 805306457 1610612741 (1e9+7,1e9+9)*/
/*prime such as 4k+3 (some)553963 553991 554003 554011 554051 554087 55
4123 554167 554171 554179 554207 554263 554299 554303 554347 554383 554
419 554431 554447 554467 554503 554527 554531 554611 554627 554639 5546
63 554699 554707 554711 554731 554747 554759 554767 554779 554791 55480
3 554839 554843 554887 554891 554899 554923 554927 554951 554959 555043
 555083 555091 555119 555143 555167 555251 555287 555307 555383 555391
555419 555439 555487 555491 555523 555671 555683 555691 555707 555739 5
55743 555767 555823 555827 555871 555931 555967 556007 556027 556043 55
6051 556067 556103 556123 556159 556211 556219 556243 556267 556271 556
279 556327 556331 556343 556351 556399 556403 556459 556483 556487 5565
19 556559 556579 556583 556607 556627 556639 556651 556679 556687 55669
1 556723 556727 556763 556799 556811 556819 556823 556859 556867 556883
 556891 556931 556939 556943 556967 556987 556999 557027 557059 557087
557159 557303 557339 557371 557423 557443 557483 557519 557551 557567 5
57591 557611 557639 557663 557671 557731 557743 557747 557759 557779 55
7803 557831 557863 557891 557899 557903 557927 557987 558007 558067 558
083 558091 558139 558167 558179 558203 558223 558251 558287 558307 5583
19 558343 558427 558431 558479 558491 558499 558539 558563 558583 55858
7 558599 558611 558643 558683 558703 558731 558787 558791 558827 558863
 558931 558947 558979 559051 559067 559099 559123 559183 559211 559219
559231 559243 559259 559319 559343 559367 559451 559459 559483 559511 5
59523 559547 559571 559583 559591 559631 559639 559667 559679 559687 55
9703 559739 559747 559799 559807 559831 559859 559883 559907 559939 559
967 559991 560023 560039 560047 560083 560107 560123 560159 560171 5601
79 560191 560207 560227 560239 560243 560299 560311 560411 560447 56045
9 560471 560479 560491 560503 560531 560543 560551 560639 560683 560719
 560767 560771 560783 560803 560827 560863 560887 560891 560939 561019
561047 561059 561079 561083 561091 561103 561191 561199 561251 561307 5
61343 561347 561359 561367 561419 561439 561551 561559 561599 561607 56
```

1667 561703 561767 561787 561839 561907 561923 561931 561943 561947 561
983 562007 562019 562043 562091 562103 562147 562231 562259 562271 5622
83 562291 562307 562351 562399 562403 562427 562439 562459 562519 56257
9 562591 562607 562631 562651 562663 562691 562699 562703 562711 562739
562759 562763 562831 562871 562931 562943 562963 562967 562979 562987
563011 563039 563047 563051 563099 563119 563131 563183 563219 563263 5
63287 563327 563351 563359 563411 563419 563447 563467 563503 563543 56
3551 563587 563599 563623 563663 563723 563743 563747 563831 563851 563
887 563947 563971 563987 563999 564059 564103 564127 564163 564191 5642
27 564251 564271 564299 564307 564323 564359 564367 564371 564391 56440
7 564419 564463 564467 564491 564523 564607 564643 564667 564671 564679
564703 564779 564827 564871 564899 564919 564923 564959 564979 564983
565039 565111 565127 565163 565171 565183 565207 565247 565259 565283 5
65303 565319 565343 565379 565387 565391 565427 565451 565463 565483 56
5507 565511 565519 565559 565567 565571 565583 565603 565651 565667 565
723 565727 565771 565787 565867 565891 565907 565919 565979 566011 5660
23 566047 566107 566131 566179 566183 566227 566231 566311 566323 56634
7 566387 566431 566443 566539 566543 566551 566563 566567 566639 566659
566707 566719 566723 566759 566767 566791 566851 566879 566911 566939
566947 566963 566971 566987 566999 567011 567031 567059 567067 567107 5
67143 567179 567187 567263 567319 567323 567367 567383 567407 567439 56
7451 567467 567487 567499 567527 567607 567631 567659 567667 567719 567
751 567767 567779 567811 567863 567871 567883 567899 567943 567947 5679
79 567991 568019 568027 568091 568151 568163 568171 568187 568207 56823
1 568279 568303 568363 568367 568387 568391 568439 568471 568523 568619
568627 568643 568679 568691 568699 568723 568751 568783 568787 568807
568823 568831 568891 568903 568907 568963 568979 568987 568991 568999 5
69003 569011 569047 569071 569083 569111 569159 569243 569251 569263 56
9267 569323 569419 569423 569431 569447 569479 569507 569579 569599 569
603 569623 569659 569663 569671 569683 569711 569731 569747 569759 5697
71 569819 569831 569839 569843 569851 569887 569903 569927 569939 56998
3 570043 570047 570071 570079 570083 570091 570107 570131 570139 570191
570359 570379 570391 570403 570407 570419 570463 570467 570487 570491
570499 570511 570527 570539 570547 570587 570643 570659 570667 570671 5
70683 570719 570743 570827 570839 570851 570859 570887 570919 570959 57
0967 570991 571019 571031 571099 571111 571147 571163 571199 571211 571
223 571231 571267 571279 571303 571331 571339 571399 571471 571531 5715
79 571583 571603 571679 571699 571751 571759 571783 571799 571811 57184
7 571867 571871 571903 571939 572023 572027 572051 572059 572063 572087
572107 572179 572183 572207 572239 572251 572303 572311 572323 572387
572399 572419 572423 572471 572479 572491 572519 572567 572587 572599 5
72639 572651 572659 572683 572687 572699 572707 572711 572791 572807 57
2827 572843 572867 572879 572903 572927 572939 572963 573007 573031 573
047 573107 573119 573143 573163 573179 573247 573263 573299 573343 5733
71 573379 573383 573451 573479 573487 573511 573523 573527 573571 57364
7 573679 573691 573719 573739 573763 573787 573791 573847 573851 573863
573871 573883 573887 573899 573967 574003 574031 574051 574099 574127
574159 574163 574183 574219 574279 574283 574307 574363 574367 574423 5
74439 574507 574543 574547 574619 574627 574631 574643 574667 574687 57

```
4699 574703 574711 574723 574727 574799 574859 574907 574939 574963 574
967 575027 575063 575087 575119 575123 575131 575203 575219 575231 5752
43 575251 575303 575359 575371 575431 575479 575503 575551 575579 57559
1 575611 575623 575647 575651 575699 575711 575723 575747 575791 575863
 575867 575903 575923 575959 575963 575987 576019 576031 576119 576131
576151 576167 576179 576203 576211 576223 576227 576287 576299 576319 5
76379 576391 576427 576431 576439 576523 576539 576551 576647 576659 57
6671 576683 576703 576727 576731 576739 576743 576787 576791 576883 576
899 576943 576967 577007 577043 577063 577067 577111 577123 577147 5771
51 577219 577259 577271 577279 577307 577327 577331 577351 577363 57738
7 577399 577427 577463 577471 577483 577523 577531 577547 577559 577627
 577639 577667 577739 577751 577799 577807 577831 577867 577879 577919
577931 577939 577979 578047 578063 578131 578167 578183 578191 578203 5
78251 578267 578299 578311 578327 578363 578371 578399 578407 578419 57
8467 578483 578503 578563 578587 578603 578647 578659 578687 578719 578
779 578803 578819 578827 578839 578843 578923 578959 578971 578999 5790
11 579023 579079 579083 579107 579119 579179 579199 579239 579251 57925
9 579263 579283 579287 579311 579331 579379 579407 579427 579451 579499
 579503 579539 579563 579571 579583 579587 579611 579643 579707 579763
579779 579851 579883 579907 579947 579967 579983 580031 580079 580163 5
80183 580187 580219 580231 580259 580291 580303 580331 580339 580343 58
0379 580471 580487 580607 580627 580631 580639 580663 580687 580691 580
711 580747 580759 580763 580787 580807 580843 580859 580871 580891 5809
19 580927 580939 581047 581071 581099 581143 581171 581183 581227 58123
9 581263 581303 581311 581323 581351 581407 581411 581443 581447 581459
 581491 581527 581551 581599 581639 581663 581683 581687 581699 581731
581743 */
const int MOD1=98317,MOD2=196613;
vector <int> a[MOD1];
vector <int> vis;
bool _find(int x)
{
    if(!a[(x%MOD1+MOD1)%MOD1].size()) return false;
    for(int i=0;i<a[(x%MOD1+MOD1)%MOD1].size();i++)
    {
        if(a[(x%MOD1+MOD1)%MOD1][i]==x) return true;
    }
    return false;
}
void _insert(int x)
{
    if(!_find(x))
    {
        a[(x%MOD1+MOD1)%MOD1].push_back(x);
        vis.push_back((x%MOD1+MOD1)%MOD1);
    }
    return ;
}
int read()
```

```
{
    int s=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9')
    {
        if(ch=='-') f=-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9')
    {
        s=(s<<3)+(s<<1)+ch-'0';
        ch=getchar();
    }
    return s*f;
}
inline void write(int x)
{
    static int sta[35];
    int top=0;
    if(x<0) {putchar('-');x=-x;}
    do{
      sta[top++]=x%10,x/=10;
      }while(x);
    while(top) putchar(sta[--top]+48);
}
int main()
{
    int T_start=clock();
    int t=read();
    while(t--)
    {
        int n=read();
        for(int i=0;i<n;i++)
        {
            int x=read();
            if(!_find(x))
            {
                _insert(x);
                write(x),putchar(' ');
            }
        }
        putchar('\n');
        for(auto i:vis)
        {
            a[i].clear();
        }
        vis.clear();
    }
```

```
    return 0;
}
```

字符串

KMP

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
using namespace std;
vector<int> prefix_init_f(string s) //前缀函数初始化
// 前缀函数就是，子串 s[0..i]最长的相等的真前缀与真后缀的长度。
// 在 kmp 算法中，前缀函数是核心，它决定了模式串（key）在匹配过程若不匹配应该跳转的位置。
// e.g. abcabc 的前缀函数为[0,0,0,1,2,3]
{
	int len=s.length();
	vector<int> dp(len,0);
	dp[0]=0;
	for(int i=1;i<len;i++)
	{
		int j=dp[i-1];
		while(j>0&&s[i]!=s[j]) j=dp[j-1];//如果 s[i]和 s[j]不相同，j 跳到前一个符合的位置
		if(s[i]==s[j]) j++; //如果 s[i]和 s[j]相同，j+1
		dp[i]=j;
	}
	return dp;
}
void kmp(string s,string key)
{
	if(key.length()==0) return;
	vector<int> dp=prefix_init_f(key);
	int i=0,j=0;
	while(i<s.length())
	{
		if(s[i]==key[j]) {i++;j++;} //如果匹配，接着匹配下一个字符
		else if(j>0) j=dp[j-1]; //如果不匹配，j 跳到前一个符合的位置
```

```
            else i++;
            if(j==key.length())
            {
                  /*some operation*/
                  j=dp[j-1];//匹配成功后，j 跳到前一个符合的位置
            }
      }
}
int main()
{
      int T_start=clock();

      return 0;
}
```

tiretree

```
#include <algorithm>
#include <bitset>
#include <cmath>
#include <cstdio>
#include <cstdlib>
#include <cstring>
#include <ctime>
#include <deque>
#include <map>
#include <iostream>
#include <queue>
#include <set>
#include <stack>
#include <vector>
using namespace std;
struct TiretreeNode{
	vector<TiretreeNode*> child;
	bool isEnd;

	TiretreeNode(): child(26,nullptr),isEnd(false) {}
};
void insert(string s,TiretreeNode* root)
{
	for(auto ch:s)
	{
		int idx=ch-'a';
		if(root->child[idx]==nullptr)
		{
			root->child[idx]=new TiretreeNode;
		}
		root=root->child[idx];
	}
	root->isEnd=true;
}
bool iscontain(string s,TiretreeNode* root)
{
	for(auto ch:s)
	{
		int idx=ch-'a';
		if(root->child[idx]==nullptr)
		{
			return false;
		}
		root=root->child[idx];
	}
	return root!=nullptr&&root->isEnd==true;
}
```

```
int main()
{
	int T_start=clock();
	vector<string> allsub={"hello","world","ld"};
	TiretreeNode* root=new TiretreeNode();
	for(auto i:allsub)
	{
		insert(i,root);
	}
	vector<string> test={"hll","l","d","po","world"};
	for(auto i:test)
	{
		cout<<i<<' '<<(iscontain(i,root)?"is in the tree":"is not i
n the tree")<<endl;
	}
	return 0;
}
```