Antiamuny Code Library

tarjen

目录

第零章

- 安装 `typst`:
  - Linux, macOS, WSL

    ```
    curl -fsSL https://typst.community/typst-install/install.sh | sh
    ```
  - Windows

    ```
    irm https://typst.community/typst-install/install.ps1 | iex
    ```
- 安装 VSCode 插件 `tinymist`:
  - 打开 VSCode
  - 搜索 `tinymist` 安装插件

赛前准备

测样例脚本(python 版)

```python
import os
import sys
import zipfile

c = sys.argv[1]
code
```

测样例脚本(bash 版)

对拍(cpp 版本)

```cpp
int t=10000,j=0;
while(t)
{
    cout<<"test:"<<++j<<"\n";
    t--;
    system("testin.exe > data.txt");
    system("abiaocheng.exe < data.txt > biaoda.txt");
    system("nedtest.exe < data.txt > aatest.txt");
    if(system("fc aatest.txt biaoda.txt")){
         cout<<"error"<<"\n";
         break;
     }
}
if(t==0) cout<<"no error"<<endl;
//system("pause");
return 0;
```

对拍(bat 版本)

```bat
@echo off
setlocal enabledelayedexpansion

set T=0
:loop
if %T% gtr 100000 (
    echo "Finished"
    exit /b
)
set /a T+=1
echo T=!T!
testin.exe > data.txt
abiaocheng.exe < data.txt > biaoda.txt
nedtest.exe < data.txt > aatest.txt

fc aatest.txt biaoda.txt
if errorlevel 1 (
    echo "WA"
    exit /b
)

goto loop
```

杂项

Rand

```cpp
mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());
int myRand(int B) {
  return (unsigned long long)rng() % B;
}
```

Time

```cpp
struct gettime{
    clock_t star,ends;
    gettime(){
        star = clock();
    }
    ~gettime(){
        ends = clock();
        cout <<"Running Time : "<<(double)(ends - star)/ CLOCKS_PER_SEC << endl;
    }
} tim;
int main()
{
    tim.begin();
    tim.end();
    return 0;

}
```

子集枚举

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

    }
}
```

高维前缀和

```cpp
for(int j = 0; j < n; j++)
    for(int i = 0; i < 1 << n; i++)
        if(i >> j & 1) f[i] += f[i ^ (1 << j)];
```

三分

```cpp
double f(double x){
    //something
}
const double eps=1e-8;
double sanfen(double l, double r){
    double mid,midr,ans;
    while (fabs(r-l)>eps) {
        mid=(l+r)/2;
        midr=(mid+r)/2;
         if(f(mid) < f(midr)) l=mid; else r=midr;   //求最大值
     }
     ans=f(l);
     return ans;
 }
```

Bitset

```
template<int LEN>void solve(){
    sz=a.size();
    if (LEN<=b.size()){
        solve<min(N,LEN+10)>();
        return;
    }
    using Bitset=bitset<LEN>;
    Bitset is[sz+5];
    for(int i=0;i<sz;i++)for(int j=0;j<b.size();j++){
         auto&[x,l,r]=a[i];
         auto&[y,L,R]=b[j];
         if(l<=y&&y<=r&&L<=x&&x<=R)is[i][j]=1;
     }
     ll rs=0;
     for(int i=0,x;i<sz;i++)for(int j=i+1;j<sz;j++){
         x=(is[i]&is[j]).count();
         rs+=x*(x-1)/2;
     }
     cout<<rs<<'\n';
 }
```

Bitset 手写

```
const int N=3000;
typedef unsigned long long ull;

int lim=N/64+3;
struct Bitset{
  ull v[N/64+5];
 void init(){
   memset(v,0,sizeof(v));
   return;
   }
   void add(int x){
    v[x>>6]|=(1ull<<(x&63));
    return;
   }
   void shift1(){
    int lst=0;
    for(int i=0;i<=lim;i++){
      int cur=v[i]>>63;
       v[i]<<=1;v[i]|=lst;
       lst=cur;
    }
     return;
   }
   int count(){
    int res=0;
    for(int i=0;i<=lim;i++) res+=__builtin_popcountll(v[i]);
    return res;
   }
   Bitset operator|(const Bitset &x)const{
     Bitset res;
     for(int i=0;i<=lim;i++) res.v[i]=v[i]|x.v[i];
     return res;
   }
  Bitset operator&(const Bitset &x)const{
    Bitset res;
    for(int i=0;i<=lim;i++) res.v[i]=v[i]&x.v[i];
    return res;
  }
  Bitset operator^(const Bitset &x)const{
    Bitset res;
    for(int i=0;i<=lim;i++) res.v[i]=v[i]^x.v[i];
    return res;
  }
  Bitset operator-(const Bitset &x)const{
    Bitset res;ull lst=0;
    for(int i=0;i<=lim;i++){
     ull cur=(v[i]<x.v[i]+lst);
     res.v[i]=v[i]-x.v[i]-lst;
      lst=cur;
   }
    return res;
  }
}
```

Lcslen(n2/w)

```
#include <bits/stdc++.h>
using namespace std;
const int N=3010;
typedef unsigned long long ull;
const int lim=N/64+3;
struct Bitset{
 ull v[N/64+5];
 void init(){
   memset(v,0,sizeof(v));
    return;
  }
  void add(int x){
   v[x>>6]|=(1ull<<(x&63));
   return;
  }
  void shift1(){
   int lst=0;
   for(int i=0;i<=lim;i++){
     int cur=v[i]>>63;
      v[i]<<=1;v[i]|=lst;
      lst=cur;
   }
    return;
  }
  int count(){
   int res=0;
   for(int i=0;i<=lim;i++) res+=__builtin_popcountll(v[i]);
   return res;
  }
  Bitset operator|(const Bitset &x)const{
    Bitset res;
    for(int i=0;i<=lim;i++) res.v[i]=v[i]|x.v[i];
    return res;
  }
  Bitset operator&(const Bitset &x)const{
    Bitset res;
    for(int i=0;i<=lim;i++) res.v[i]=v[i]&x.v[i];
    return res;
  }
  Bitset operator^(const Bitset &x)const{
    Bitset res;
    for(int i=0;i<=lim;i++) res.v[i]=v[i]^x.v[i];
    return res;
  }
  Bitset operator-(const Bitset &x)const{
    Bitset res;ull lst=0;
    for(int i=0;i<=lim;i++){
     ull cur=(v[i]<x.v[i]+lst);
     res.v[i]=v[i]-x.v[i]-lst;
      lst=cur;
   }
    return res;
  }
}ch[26],f,g;
auto getans = [&](int mid){
    int l1=mid,l2=n-mid;
    f.init();g.init();
    for(int i=0;i<26;i++){
        ch[i].init();
    }
    for(int i=mid+1;i<=n;i++)ch[s[i]-'a'].add(i);
    for(int i=1;i<=mid;i++){
        g=f|ch[s[i]-'a'];
        f.shift1();
        f.add(1);
        f=g-f;
        f=f^g;f=f&g;
    }
    return f.count();
};
```

wqs 二分

```
// 这里是 上凸 取 min
// 上凸取 max 二分的时候改变一下 mid 的变化方向
// 下凸取 min 改变 mid 算贡献的符号
//min max 指的是求的是最大值 还是最小值
int solve(int mid){
    k=mid;
    function<void(int,int)> dfs = [&](int x,int h){
        info dp2[3];memset(dp2,0,sizeof(dp2));
        for(auto [it,w]:ve[x])if(it!=h){
             dfs(it,x);
             gmax(dp2[0],dp[x][0]+dp[it][0]);
             gmax(dp2[2],dp[x][0]+dp[it][1]);
             gmax(dp2[2],dp[x][2]+dp[it][0]);
             dp[x][0]=dp2[0];
             dp[x][1]=dp2[1];
             dp[x][2]=dp2[2];
        }
        dp[x][0]=dp2[2];
        dp[x][1]=dp2[0]+info{a[x]+k,1};
    };
    dfs(1,0);
    return dp[1][0].second;
}
signed main()
{
    int l=-sum-1000000000000ll,r=sum+1000000000000ll;
    int ans=1e18;
    while(l<=r){
        int mid=(l+r)/2ll;
        if(solve(mid)>=m){
            ans=dp[1][0].first-m*mid;
            r=mid-1;
        }
        else l=mid+1;
    }
    if(ans!=(int)1e18)cout<<ans<<"\n";
    else cout<<"Impossible\n";
    return 0;
}
```

判断异或方程组是否有解

```
const int maxn=1e2+5;
//每个方程组一定是 xi1^xi2^xi3...=1/0 的形式
bitset<maxn>b[maxn];//1 表示这个方程组存在 Xi  比如 x3^x5=1 就应该是 3  和 5 的地方上是 1
//b[n+1]存方程右边等于 0/1
int sum[maxn];
bool check(int x)
{
    for(int i=1;i<=n;i++)if(b[x][i])return true;
    return !b[x][n+1];
}
bool solve()
{
    for(int i=1,p=1;i<=n;i++,p++)
    {
        if(!b[p][i])
        {
            int pos=0;
            for(int j=p+1; j<=n; j++)
                if(b[j][i])
                {
                    pos=j;
                    break;
                }
            if(pos)swap(b[p],b[pos]);
        }
        int flag=b[p][i];
        for(int j=p+1; j<=n; j++) if(b[j][i]) b[j]^=b[p],flag=1;
        if(!flag) p--;
    }
    for(int i=1; i<=n; i++) if(!check(i)) return false;
    return true;
}
```

可以判断不同或相同的并查集

```cpp
int f[maxn];
int getf(int x){
    if(x<0)return -getf(-x);
    if(x==f[x])return x;
    else return f[x]=getf(f[x]);
}
bool merge(int x,int y){//如果是 x!=y 将 y 取反(x>0 y>0)
    x=getf(x),y=getf(y);
    if(x==-y)return false;
    if(x==y)return true;
    if(x<0)f[-x]=-y;
    else f[x]=y;
    return true;
}
```

Int128

```cpp
#include <bits/stdc++.h>
using namespace std;
inline __int128 read()
{
    __int128 x=0,f=1;
    char ch=getchar();
    while(ch<'0'||ch>'9')
    {
        if(ch=='-')
            f=-1;
        ch=getchar();
    }
    while(ch>='0'&&ch<='9')
    {
        x=x*10+ch-'0';
        ch=getchar();
    }
    return x*f;
}
inline void write(__int128 x)
{
    if(x<0)
    {
        putchar('-');
        x=-x;
    }
    if(x>9)
        write(x/10);
    putchar(x%10+'0');
}

istream& operator >> (istream& in, __int128& num) {
    string s;in>>s;
    num=0;
    for(auto it:s)num=num*10+it-'0';
    return in;
}

ostream& operator << (ostream& out, __int128 num) {
    string s;
    do{
        s.push_back(char(num%10+'0'));
        num/=10;
    }while(num>0);
    reverse(s.begin(),s.end());
    out<<s;
    return out;
}
```

树哈希

```cpp
#include <cctype>
#include <chrono>
#include <cstdio>
#include <random>
#include <set>
#include <vector>

typedef unsigned long long ull;

const ull mask = std::chrono::steady_clock::now().time_since_epoch().count();

ull h(ull x) {
    return x * x * x * 1237123 + 19260817;
}
ull f(ull x) {
    ull cur = h(x & ((1 << 31) - 1)) + h(x >> 31);
    return cur;
}
ull shift(ull x) {
  x ^= mask;
  x ^= x << 13;
  x ^= x >> 7;
  x ^= x << 17;
  x ^= mask;
  return x;
}

const int N = 1e6 + 10;

int n;
ull hash[N];
std::vector<int> edge[N];
std::set<ull> trees;

void getHash(int x, int p) {
  hash[x] = 1;
  for (int i : edge[x]) {
    if (i == p) {
      continue;
    }
    getHash(i, x);
    hash[x] += shift(hash[i]);
  }
  trees.insert(hash[x]);
}
```

```cpp

int main() {
  scanf("%d", &n);
  for (int i = 1; i < n; i++) {
    int u, v;
    scanf("%d%d", &u, &v);
    edge[u].push_back(v);
    edge[v].push_back(u);
  }
  getHash(1, 0);
  printf("%lu", trees.size());
}

```

求 **s** 所有前缀对于 **t** 的所有子串的最长公共子序列长度

```cpp
#include <bits/stdc++.h>
using namespace std;
struct PairLCS {
    vector<vector<int>> ih, iv;
    int n = 0, m = 0;
    PairLCS(string s, string t) : n(s.size()), m(t.size()) {
        ih = iv = vector<vector<int>>(n + 1, vector<int>(m + 1));
        iota(ih[0].begin(),ih[0].end(), 0);
        for (int i = 1; i <= n; i++) {
             for (int j = 1; j <= m; j++) {
                 if (s[i - 1] == t[j - 1]) {
                     ih[i][j] = iv[i][j - 1];
                     iv[i][j] = ih[i - 1][j];
                 } else {
                     ih[i][j] = std::max(ih[i - 1][j], iv[i][j - 1]);
                     iv[i][j] = std::min(ih[i - 1][j], iv[i][j - 1]);
                 }
             }
         }
     }
     int query(int a, int b, int c) const {
         int res = 0;
         for (int i = b + 1; i <= c; i++) res += ih[a][i] <= b;
         return res;
     }  // s[0,a) t[b,c)
};



int cas;

void solution() {
  int q;
  std::string s, t;
  std::cin >> q >> s >> t;

  // int n = s.size(), m = t.size();
  PairLCS solver(s, t);

  for (int _ = 0; _ < q; _++) {
    int a, b, c;
    std::cin >> a >> b >> c;
    std::cout << solver.query(a, b, c) << '\n';
  }
}


int main() {
    ios::sync_with_stdio(false);
    cin.tie(0);
    int T = 1;
    // std::cin >> T;
    for (cas = 1; cas <= T; cas++) solution();

    return 0;
}
```

线性基

```
struct LinearBasis
{
    static const int maxbase = 35;
    bool flag = false;
    ll a[maxbase + 1];
    int tot;
    LinearBasis()
    {
        memset(a, 0, sizeof a);
         tot=0;
     }
     LinearBasis(ll *x, int n)
     {
         LinearBasis();
         build(x, n);
     }
     void build(ll *x, int n)
     {
         for(int i = 1; i <= n; ++i)
             insert(x[i]);
     }
     void clear()
     {
         memset(a, 0, sizeof a);
     }
     bool insert(ll t)
     {
        //暴力插入一个数，维护的是一个上三角型的线性基矩阵，时间复杂度低，当待插入元素能插入时，返
回 true
         for(int i = maxbase; i >= 0; --i)
         {
             if(t & (1ll << i))
             {
                 if(!a[i])
                 {
                     a[i] = t;//这里表示插入成功
                     break;
                 }
                 t ^= a[i];
```

```
            }
        }
        if(t == 0)flag = true;
        return t;
    }
    bool query(ll t)
    {
        // 询问 t 是否可以被当前线性基表示，不插入
        if(t > queryMax())return false;
        if(t == 0)return true;
        for(int i = maxbase; i >= 0; --i)
        {
            if(t & (1ll << i))
            {
                if(!a[i])
                {
                    return false;
                }
                t ^= a[i];
            }
        }
        return true;
    }
    void Insert(ll t)
    {
        //插入一个线性基，利用高斯消元法维护一个对角矩阵
        for(int i = maxbase; i >= 0; --i)
        {
            if(t >> i & 1)
            {
                if(a[i])t ^= a[i];
                else
                {
                    a[i] = t;
                    for(int j = i - 1; j >= 0; --j)if(a[j] && (a[i] >> j & 1))a[i]
^= a[j];
                     for(int j = i + 1; j <= maxbase; ++j)if(a[j] >> i & 1)a[j] ^=
a[i];
                    break;
                }
            }
        }
    }
    LinearBasis merge(const LinearBasis &l1, const LinearBasis &l2)
    {
        // 得到两个线性基的并
        LinearBasis ret = l1;
        for(int i = maxbase; i >= 0; --i)
            if(l2.a[i])
                ret.insert(l2.a[i]);
        return ret;
    }
    LinearBasis intersection(const LinearBasis &l1, const LinearBasis &l2)
    {
        //得到两个线性基的交
        LinearBasis all, ret, full;
```

```cpp
       ret.clear();
       for(int i = maxbase; i >= 0; --i)
       {
           all.a[i] = l1.a[i];
           full.a[i] = 1ll << i;
       }
       for(int i = maxbase; i >= 0; --i)
        {
            if(l2.a[i])
            {
                ll v = l2.a[i], k = 0;
                bool flag = true;
                for(int j = maxbase; j >= 0; --j)
                {
                    if(v & (1ll << j))
                    {
                        if(all.a[j])
                        {
                            v ^= all.a[j];
                            k ^= full.a[j];
                        }
                        else
                        {
                            // l2's basis is not in l1's;
                            flag = false;
                            all.a[j] = v;
                            full.a[j] = k;
                            break;
                        }
                    }
                }
                if(flag)
                {
                    ll v = 0; // get intersection by k;
                    for(int j = maxbase; j >= 0; --j)
                    {
                        if(k & (1ll << j))
                        {
                            v ^= l1.a[j];
                        }
                    }
                    ret.insert(v);
                }
            }
        }
        return ret;
    }
    //询问最值
    ll queryMax()
    {
        ll ret = 0;
        for(int i = maxbase; i >= 0; --i)
            if((ret ^ a[i]) > ret)
                ret ^= a[i];
        return ret;
    }
    ll queryMin()
    {
        for(int i = 0; i <= maxbase; ++i)
            if(a[i])
                return a[i];
        return 0;
    }
    ll Kth_Max(ll k)
    {
        ll res=0;
        for(int i=62;i>=0;i--)
            if (k&(1LL<<i)) res^=a[i];
        return res;
    }
};

```

维护多个二维向量能够表示的范围

```
int gcd(int x,int y){
    if(y==0)return x;
    else return gcd(y,x%y);
}
struct vec{
    int a00,a01,a11;
    void clear(){
        a00=a01=a11=0;
    }
     void insert(int x,int y){
         while(x!=0){
             int t=a00/x;
             a00-=x*t;
             a01-=y*t;
             swap(a00,x);
             swap(a01,y);
         }
         a11=gcd(a11,abs(y));
         if(a11!=0)a01%=a11;
     }
     bool query(int x,int y){
         if(x!=0){
             if(a00==0||x%a00!=0)return false;
             int t=x/a00;
             y-=a01*t;
         }
         if(y==0){
             return true;
         }
         else return a11!=0&&y%a11==0;
     }
};
```

字符串

AC 自动机

```cpp
#include <bits/stdc++.h>
using namespace std;
const int maxn=1e6+10;
int n;
char c[maxn];
struct AC{
    int trie[maxn][26],tot;
    int e[maxn],fail[maxn],old[maxn];
    void init(){
         memset(trie,0,sizeof(trie));
         memset(e,0,sizeof(e));
         memset(fail,0,sizeof(fail));
         memset(old,0,sizeof(old));
         tot=0;
     }
     void insert(char *t){
         int x=0;
         for(int i=1;t[i];i++){
             if(!trie[x][t[i]-'a']){
                 trie[x][t[i]-'a']=++tot;
             }
             x=trie[x][t[i]-'a'];
         }
         e[x]++;
     }
     queue<int> qu;
     void build(){
         for(int i=0;i<26;i++){
             if(trie[0][i])qu.push(trie[0][i]);
         }
         while(!qu.empty()){
             int x=qu.front();
             qu.pop();
             for(int i=0;i<26;i++){
                 if(trie[x][i]){
                     fail[trie[x][i]]=trie[fail[x]][i];
                     qu.push(trie[x][i]);
                 }
                 else trie[x][i]=trie[fail[x]][i];
                      old[trie[x][i]]=e[fail[trie[x][i]]] ? fail[trie[x][i]] :
old[fail[trie[x][i]]];
             }
         }
     }
     int query(char *t){//这里是统计有多少模板串出现在了文本串之中，所以统计到了就要变成-1
         int x=0,res=0;
         for(int i=1;t[i];i++){
             x=trie[x][t[i]-'a'];
             for(int j=x;j&&e[j]!=-1;j=old[j]){
                 res+=e[j];
                 e[j]=-1;
             }
         }
```

```
        return res;
    }
};
AC ac;
int main()
{
    scanf("%d",&n);
    for(int i=1;i<=n;i++){
        scanf("%s",c+1);
        ac.insert(c);
    }
    ac.build();
    scanf("%s",c+1);

    cout<<ac.query(c)<<endl;
    return 0;
}
```

Dequehash

```
/*
严格 0base，不用管任何函数里面的东西，用就可以了，不要越界
pair<int,int> first 表示哈希 sum，second 表示当前位置的值
*/
#define int long long
#define sz(a) (int)((a).size())
const int maxn=3e5+10;
const int mod=1e9+7,base=1331;
int fpow(int n, int k, int p = mod) {int r = 1; for (; k; k >>= 1) {if (k & 1) r =
r * n % p; n = n * n % p;} return r;}
void add(int& a, int val, int p = mod) {if ((a = (a + val)) >= p) a -= p;}
void sub(int& a, int val, int p = mod) {if ((a = (a - val)) < 0) a += p;}
int mul(int a, int b, int p = mod) {return (int) a * b % p;}
int inv(int a, int p = mod) {return fpow(a, p - 2, p);}
int p[maxn],ip[maxn];
void init()
{
    p[0] = 1; for(int i=1;i<maxn;i++) p[i] = mul(p[i - 1], base, mod);
    for(int i=0;i<maxn;i++) ip[i] = inv(p[i], mod);
}
struct extendable_sequence {
    deque<pair<int,int>> dq;
    int pow_offset;

    extendable_sequence() {
        pow_offset = 0;
        dq.push_back(make_pair(0, 0));
    }

    int size() {
        return sz(dq) - 1;
    }

    pair<int,int>& operator [] (int i) {
        return dq[i+1];
    }

    void add_back(vector<int> vals) {
        int t = dq.back().first;
        for(int i=0;i<sz(vals);i++) {
            add(t, mul(vals[i], mul(p[sz(dq) - 1], ip[pow_offset], mod), mod), mod);
            dq.push_back(make_pair(t, vals[i]));
        }
    }

    void add_front(vector<int> vals) {
        pow_offset += sz(vals);
        int t = dq.front().first;
        for(int i=sz(vals)-1;i>=0;i--) {
            dq.front().second = vals[i];
            sub(t, mul(vals[i], mul(p[i], ip[pow_offset], mod), mod), mod);
            dq.push_front(make_pair(t, 0));
        }
    }

    int calc(int l, int r) {
        l++, r++;
        if (l > r) return 0;
        int res = dq[r].first;
        sub(res, dq[l - 1].first, mod);
        res = mul(res, ip[l - 1], mod);
        res = mul(res, p[pow_offset], mod);
        return res;
    }
};
//返回(x+y)[l 到 r]的哈希值
int calc(extendable_sequence& x, extendable_sequence& y, int l, int r) {
    int res = x.calc(l, min(r, sz(x) - 1));
    add(res, mul(y.calc(max(0ll, l - sz(x)), r - sz(x)), p[sz(x)], mod), mod);
    return res;
}
//返回(x+y)[i]单个元素的值
int calc(extendable_sequence& x, extendable_sequence& y, int i) {
    if (i < sz(x)) {
        return x[i].second;
    }
    if (i - sz(x) < sz(y)) {
        return y[i - sz(x)].second;
    }
    return -1;
}
```

Exkmp

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1e6 + 1e3;
struct EXKMP
{ // S 里找 T

    char S[maxn], T[maxn];
    int next[maxn], extend[maxn];
    void Get_Next()
     {
        int lenT = strlen(T + 1), p = 1, pos;
        next[1] = lenT; // 对于 next[1] 要特殊考虑
        while (p + 1 <= lenT && T[p] == T[p + 1])
            ++p;
        next[pos = 2] = p - 1; // next[2] 是为了初始化

        for (int i = 3; i <= lenT; i++)
        { // 注意此时 k + 1 = i
            int len = next[i - pos + 1];
            if (len + i < p + 1)
                next[i] = len; // 对应上面第一种情况
            else
            {
                int j = max(p - i + 1, 0); // 找到前面对于 子串 最靠后已经匹配的位置
                while (i + j <= lenT && T[j + 1] == T[i + j])
                    ++j;                         // 第二种需要暴力匹配
                p = i + (next[pos = i] = j) - 1; // 记得更新 p, pos
            }
        }
    }
    void ExKMP()
    {
        int lenS = strlen(S + 1), lenT = strlen(T + 1), p = 1, pos;
        Get_Next();
        while (p <= lenT && S[p] == T[p])
            ++p;
        p = extend[pos = 1] = p - 1; // 初始化 extend[1]

        for (int i = 2; i <= lenS; i++)
        {
            int len = next[i - pos + 1];
            if (len + i < p + 1)
                extend[i] = len;
            else
            {
                int j = max(p - i + 1, 0);
                while (i + j <= lenS && j <= lenT && T[j + 1] == S[i + j])
                    ++j;
                p = i + (extend[pos = i] = j) - 1;
            }
        } // 和上面基本一模一样啦
    }
} sol;
int main()
{
    scanf("%s", sol.S + 1);
    scanf("%s", sol.T + 1);

    sol.ExKMP();
    int len = strlen(sol.S + 1);
    for (int i = 1; i <= len; i++)
        printf("%d%c", sol.extend[i], i == len ? '\n' : ' ');

    return 0;
}
```

Hash

```cpp
const int N=1e6+10;
typedef long long ll;
const ll p1=31,p2=131;
const ll mod1=1e9+7,mod2=1e9+9;
typedef pair<ll,ll> hs;
const hs p = make_pair(p1,p2);
hs &operator+=(hs &a, hs b) {
    a.first=(a.first+b.first)%mod1;
    a.second=(a.second+b.second)%mod2;
     return a;
 }
 hs operator+(hs a, hs b) { return a += b; }
 hs &operator-=(hs &a, hs b) {
     a.first=(a.first-b.first+mod1)%mod1;
     a.second=(a.second-b.second+mod2)%mod2;
     return a;
 }
 hs operator-(hs a, hs b) { return a -= b; }
 hs &operator*=(hs &a, hs b) {
     a.first=(a.first*b.first)%mod1;
     a.second=(a.second*b.second)%mod2;
     return a;
 }
 hs operator*(hs a, hs b) { return a *= b; }
 struct Hash{
     int n;
     vector<hs>has1,has2,Pow;
     void Hash_init(string &s){
         n=(int)s.size();
         Pow.resize(n+2);
         has1.resize(n+2);
         has2.resize(n+2);
         Pow[0]=make_pair(1ll,1ll);
         for(int i=1;i<=n;i++)Pow[i]=Pow[i-1]*p;
         for(int i=1;i<=n;i++)has1[i]=has1[i-1]*p+hs{s[i-1]-'a'+1,s[i-1]-'a'+1};
         for(int i=n;i>=1;i--)has2[i]=has2[i+1]*p+hs{s[i-1]-'a'+1,s[i-1]-'a'+1};
     }
     hs get1(int l,int r){
         return has1[r]-has1[l-1]*Pow[r-l+1];
     }
     hs get2(int l,int r){
         return has2[l]-has2[r+1]*Pow[r-l+1];
     }
 };
```

Kmp

```cpp
#include <bits/stdc++.h>
using namespace std;
const int maxn=1e6+10;
struct KMP{//1base
    int len1,n,nxt[maxn];//nxt 表示以 i 为结尾的前缀与后缀相同的长度
    char s1[maxn],s[maxn];
    void build()
    {
        n=strlen(s+1);
        nxt[1]=0;
        int x=2,now=1;//x 是 s2 当前搜索到的位置，now 是前缀位置
        while(x<=n){
            if(s[x]==s[now]){
                nxt[x]=now;
                now++;
                x++;
            }
            else{
                if(now>1){
                    now=nxt[now-1]+1;
                }
                else{
                    nxt[x]=0;
                    now=1;
                    x++;
                }
            }
        }
    }
    void find(){//s1 1base
        int now=1,tar=1;
        len1=strlen(s1+1);
        while(tar<=len1){
            if(s1[tar]==s[now]){
                tar++;
                now++;
            }
            else{
                if(now>1){
                    now=nxt[now-1]+1;
                }
                else tar++;
            }
            if(now==n+1){
                printf("%d\n",tar-now+1);
            }
        }
    }
};
KMP sol;
int main()
{
    scanf("%s%s",sol.s1+1,sol.s+1);
    sol.build();
    sol.find();
    for(int i=1;i<=sol.n;i++)cout<<sol.nxt[i]<<" ";
    return 0;
}
```

Manacher

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1.1e7 + 5;
struct ST{
 char s[maxn * 2], str[maxn * 2];
int Len[maxn * 2], len;
 void getstr() {//重定义字符串
   int k = 0;
  len = strlen(s);
   str[k++] = '@';//开头加个特殊字符防止越界
    for (int i = 0; i < len; i++) {
      str[k++] = '#';
      str[k++] = s[i];
   }
    str[k++] = '#';
    len = k;
   str[k] = 0;//字符串尾设置为 0，防止越界
 }
  int manacher() {
   int mx = 0, id;//mx 为最右边，id 为中心点
   int maxx = 0;
    for (int i = 1; i < len; i++) {
      if (mx > i) Len[i] = min(mx - i, Len[2 * id - i]);//判断当前点超没超过 mx
      else Len[i] = 1;//超过了就让他等于 1，之后再进行查找
      while (str[i + Len[i]] == str[i - Len[i]]) Len[i]++;//判断当前点是不是最长回文子串，不断的向右扩展
     if (Len[i] + i > mx) {//更新 mx
       mx = Len[i] + i;
       id = i;//更新中间点
       maxx = max(maxx, Len[i]);//最长回文字串长度
      }
    }
    return (maxx - 1);
 }
  void writ(){
   printf("%s\n",str);
      for(int i=0;i<len;i++){
          cout<<Len[i]<<" ";
     }
    cout<<"\n";
  }
};
ST s1,s2;
int main() {
  scanf("%s", s1.s);
 s1.getstr();
 printf("%d\n",s1.manacher());
  return 0;
}
```

倍增 SA

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e6 + 10;//2*strlen
struct Suffix{
  int ht[N],rk[N],sa[N],y[N],c[N];
    int n,m;
    char s[N];
 void init(){
        n=strlen(s+1);
         m=300;
    for(int i=0;i<=m;i++) c[i]=0;
     for(int i=0;i<=2*n;i++) y[i]=0;
     for(int i=1;i<=n;i++) c[rk[i]=s[i]]++;
    for(int i=1;i<=m;i++) c[i]+=c[i-1];
     for(int i=n;i>=1;i--) sa[c[rk[i]]--]=i;
     for(int k=1;k<=n;k<<=1){
      int p=0;
      for(int i=n-k+1;i<=n;i++) y[++p]=i;
       for(int i=1;i<=n;i++){
        if(sa[i]>k){
          y[++p]=sa[i]-k;
         }
       }
      for(int i=0;i<=m;i++) c[i]=0;
       for(int i=1;i<=n;i++) c[rk[i]]++;
       for(int i=1;i<=m;i++) c[i]+=c[i-1];
       for(int i=n;i>=1;i--) sa[c[rk[y[i]]]--]=y[i];
       for(int i=0;i<=n;i++) swap(rk[i],y[i]);
       rk[sa[1]]=p=1;
      for(int i=2;i<=n;i++){
        rk[sa[i]]=(y[sa[i]] == y[sa[i-1]] && y[sa[i]+k] == y[sa[i-1]+k] ? p : ++p);
       }
       if(p>=n) break;
       m=p;
    }
     for(int i=1,k=0;i<=n;i++){
      if(k)k--;
       int j=sa[rk[i]-1];
      while(s[i+k] == s[j+k] )k++;
      ht[rk[i]] = k;
    }
   }
     void writ()
     {
         printf("%s\n",s+1);
         for(int i=1;i<=n;i++)cout<<sa[i]<<" ";;cout<<"\n";
         for(int i=1;i<=n;i++)cout<<ht[i]<<" ";;cout<<"\n";
         for(int i=1;i<=n;i++)cout<<rk[i]<<" ";;cout<<"\n";
     }

};
Suffix suf;
```

后缀自动机 SAM

```
#include <bits/stdc++.h>
using namespace std;
const int N=2e6;
struct SAM {
    struct Node {
        int tr[26];
        int len, fa;
        Node() { memset(tr, 0, sizeof(tr)); len = fa = 0; }
    }ep[N << 1];
     int last, tot, n;
     char base;
     vector<int> edg[N << 1];
    int siz[N << 1];
    void init(int _n) {
        last = tot = 1;
        base = 'a';
        for (int i = 0; i <= 2 * _n; i++) {
            ep[i] = Node();
            edg[i].clear();
            siz[i] = 0;
        }
    }
    void insert(char x) {
        int c = x - base;
        int p = last;
        int np = last = ++tot;
        siz[np] = 1;
        ep[np].len = ep[p].len + 1;
        for (; p && !ep[p].tr[c]; p = ep[p].fa)
            ep[p].tr[c] = np;
        if (!p) ep[np].fa = 1;
        else {
            int q = ep[p].tr[c];
            if (ep[q].len == ep[p].len + 1) ep[np].fa = q;
            else {
                int nq = ++tot;
                ep[nq] = ep[q];
                ep[nq].len = ep[p].len + 1;
                ep[q].fa = ep[np].fa = nq;
                for (; p && ep[p].tr[c] == q; p = ep[p].fa)
                    ep[p].tr[c] = nq;
            }
        }
    }
    void construct() {
        for (int i = 2; i <= tot; i++) {
            edg[ep[i].fa].push_back(i);
        }
    }
    void dfs(int u) {
        for (auto v : edg[u]) {
            dfs(v);
            siz[u] += siz[v];
        }
    }
    void build(string& s) {
        n = s.size();
        init(n);
        for (int i = 0; i < n; i++) {
            insert(s[i]);
        }
        construct();
        dfs(1);
    }

} sam;

```

回文自动机 PAM

```
#include<bits/stdc++.h>
using namespace std;
const int N=2e6+10;
struct PAM_Trie
{
 int ch[26];
 int fail,len,num;
};
struct PAM
 {
   PAM_Trie b[N];
   int n,length,last,cnt;
   char s[N];
   PAM()
   {
     b[0].len = 0; b[1].len = -1;
     b[0].fail = 1; b[1].fail = 0;
    last = 0;
    cnt = 1;
   }
   int get_fail(int x)
   {
    while(s[n-b[x].len-1]!=s[n])
    {
       x=b[x].fail;
     }
     return x;
  }
   void insert()
   {
     int p=get_fail(last);
    if(!b[p].ch[s[n]-'a'])
    {
       b[++cnt].len=b[p].len+2;
       int tmp=get_fail(b[p].fail);
       b[cnt].fail=b[tmp].ch[s[n]-'a'];
       b[cnt].num=b[b[cnt].fail].num+1;
       b[p].ch[s[n]-'a']=cnt;
     }
     last=b[p].ch[s[n]-'a'];
    cout<<last<<"\n";
     //如果要统计出现次数 f[last]++;
  }
   void init()
   {
     length=strlen(s+1);
     for(n=1;n<=length;n++)
    {
       insert();
    }
  }
 }pa;
 int main()
 {
  scanf("%s",pa.s+1);
   pa.init();
   return 0;
 }
```

最小表示

```
int getMin(string s) {
    int i = 0, j = 1, k = 0;
    int len = s.length();
    while(i<len && j<len && k<len) {
        int tmp = s[(i + k) % len] - s[(j + k) % len];
        if(tmp==0) k++;
        else {
            if(tmp>0) i += k + 1;
            else j += k + 1;
             if(i==j) j++;
             k = 0;
         }
     }
     return min(i, j);
}
```

图论

Johnson 全源最短路

```cpp
struct graph {
  vector<vector<pair<int, ll>>> e;
  graph(int n) : e(n + 1) {}
  void adde(int u, int v, ll w) { e[u].push_back({v, w}); }
  vector<ll> h;
  // initialize h(u), return false if there exists a negative cycle
  bool init() {
    int n = e.size();
    h.assign(n, 0);
     queue<int> que;
     for (int u = 1; u < n; u++) que.push(u);
     vector<int> vis(n, 0), cnt(n, n + 1);
     while (que.size()) {
       auto u = que.front();
       que.pop();
       vis[u] = false;
       if (!cnt[u]--) return false;  // exists a negative cycle
       for (auto &[v, w] : e[u])
         if (h[v] > h[u] + w) {
           h[v] = h[u] + w;
           if (!vis[v]) que.push(v), vis[v] = 1;
         }
     }
     return true;
   }
   // single source shortest path from given sink based on h(u)
   vector<ll> query(int s) {
     int n = e.size();
     vector<ll> dis(n, inf);
     priority_queue<pair<ll, int>, vector<pair<ll, int>>,
            greater<pair<ll, int>>>
         que;
     que.push({dis[s] = 0, s});
     while (que.size()) {
       auto [du, u] = que.top();
       que.pop();
       if (dis[u] < du) continue;
       for (auto [v, w] : e[u]) {
         auto dv = du + w + h[u] - h[v];
         if (dis[v] > dv) que.push({dis[v] = dv, v});
       }
     }
     for (int i = 0; i < n; i++) dis[i] += h[i] - h[s];
     return dis;
   }
 };
```

Kosaraju

```cpp
#include <bits/stdc++.h>
using namespace std;
const int maxn = 100010;
vector<int> ve[maxn],ve2[maxn];
vector<int> sta;  // 存第一次dfs1()的结果，即标记点的先后顺序，优先级小的点先进
```

```cpp
int vis[maxn];    // vis[i]标记第一次 dfs1()点 i 是否访问过
int col[maxn];  // col[i]标记点 i 属于第几个强连通分量，同时记录 dfs2()过程中点 i 是否访问过
int cnt;    //cnt 表示强连通分量的个数
void dfs1(int x){
    vis[x] = 1;
    for(auto it:ve[x])if(!vis[it])
        dfs1(it);
    sta.push_back(x); //记录点的先后顺序，按照拓扑排序，优先级大的放在 S 的后面
 }

 void dfs2(int x){
    if(col[x])    return;
    col[x] = cnt;
    for(auto it:ve[x])if(!col[it])
        dfs2(it);
 }
 void Kosaraju(int n) {
    cnt = 0;
    sta.clear();
    memset(vis,0,sizeof(vis));
    memset(col,0,sizeof(col));
    for(int i=1; i<=n; i++) //搜索所有点
        dfs1(i);
    for(int i=n-1; i>=0; i--){
        if(!col[sta[i]]){
            cnt++;
            dfs2(sta[i]);
        }
    }
 }

```

K 短路

```cpp
//复杂度 nlogn +mlogm+klogk
#include <bits/stdc++.h>
#include <queue>

template <class T, class U>
inline bool smin(T &x, const U &y) {
  return y < x ? x = y, 1 : 0;
}
template <class T, class U>
 inline bool smax(T &x, const U &y) {
   return x < y ? x = y, 1 : 0;
 }

 using LL = long long;
 using PII = std::pair<int, int>;

 constexpr int N(2.5e5 + 5);

 using T = LL;
 struct Edge {
   int x, y; T z;
 };
 struct Heap {
   struct Node {
     int ls, rs, h, v;
     T w;
   } t[N * 40];
   int cnt;
   int newNode(int v, T w) {
     t[++cnt] = {0, 0, 1, v, w};
     return cnt;
   }
   int merge(int x, int y) {
     if (!x) return y;
     if (!y) return x;
     if (t[x].w > t[y].w) std::swap(x, y);
     t[++cnt] = t[x], x = cnt;
     t[x].rs = merge(t[x].rs, y);
     if (t[t[x].ls].h < t[t[x].rs].h) std::swap(t[x].ls, t[x].rs);
     t[x].h = t[t[x].rs].h + 1;
     return x;
   }
 } h;


 std::vector<T> kShortestPath(int n, int k, int s, int t, const std::vector<Edge> &e) {
   int m = e.size();
   std::vector<int> deg(n + 1), g(m);
   for (auto &[x, y, z] : e) deg[y]++;
   for (int i = 1; i <= n; i++) deg[i] += deg[i - 1];
   for (int i = 0; i < m; i++) g[--deg[e[i].y]] = i;

   std::vector<T> d(n, -1);
   std::vector<int> fa(n, -1), p;

   using Q = std::pair<T, int>;
   std::priority_queue<Q, std::vector<Q>, std::greater<Q>> q;

   {
     p.reserve(n);
     d[t] = 0, q.push({0, t});

     std::vector<bool> vis(n);
     while (!q.empty()) {
       int x = q.top().second;
       q.pop();
       if (vis[x]) continue;
       vis[x] = true;
       p.push_back(x);
       for (int i = deg[x]; i < deg[x + 1]; i++) {
         auto &[y, _, z] = e[g[i]];
         if (d[y] == -1 || d[y] > d[x] + z) {
           d[y] = d[x] + z, fa[y] = g[i];
           q.push({d[y], y});
         }
       }
     }
   }
```

```cpp

 if (d[s] == -1) std::vector<T>(k, -1);
 std::vector<int> heap(n);
 h.cnt = 0;
 for (int i = 0; i < m; i++) {
   auto &[x, y, z] = e[i];
   if (d[x] != -1 && d[y] != -1 && fa[x] != i) {
     heap[x] = h.merge(heap[x], h.newNode(y, d[y] + z - d[x]));
   }
 }

 for (int x : p) {
   if (x != t) heap[x] = h.merge(heap[x], heap[e[fa[x]].y]);
 }

 if (heap[s]) q.push({d[s] + h.t[heap[s]].w, heap[s]});
 std::vector<T> res = {d[s]};

 for (int i = 1; i < k && !q.empty(); i++) {
   auto [w, o] = q.top();
   q.pop();

    res.push_back(w);

    int j = h.t[o].v;
    if (heap[j]) q.push({w + h.t[heap[j]].w, heap[j]});
    for (auto s : {h.t[o].ls, h.t[o].rs}) {
      if (s) q.push({w + h.t[s].w - h.t[o].w, s});
    }
  }
  res.resize(k, -1);
  return res;
}

int a[N];
void solve() {
  int n, k;
  std::cin >> n >> k;

  std::vector<Edge> e;
  for (int i = 1; i <= n; i++) {
    std::cin >> a[i];
  }
  e.push_back({0, 1, a[1]});
  for (int i = 2; i <= n; i++) {
    if (i - 3 > 0) e.push_back({i - 3, i, a[i]});
    e.push_back({i - 2, i, a[i]});
    e.push_back({i - 1, i, a[i]});
  }
  if (n - 1 >= 1) e.push_back({n - 1, n + 1, 0});
  e.push_back({n, n + 1, 0});

  auto res = kShortestPath(n + 2, k, 0, n + 1, e);

  for (auto x : res) std::cout << x << "\n";
}
```

```cpp

int main() {
  // freopen("t.in", "r", stdin);

  std::ios::sync_with_stdio(false);
  std::cin.tie(nullptr);

  int t = 1;

  // std::cin >> t;

  while (t--) {
    solve();
  }
  return 0;
}
```

Maxflow 只算值版本

```cpp
struct dinic{
    struct E{
        int to,cap,inv;
    };
    vector <E> g[N];
    int dis[N],now[N];
    void addedge(int u,int v,int w){
        g[u].push_back({v,w,(int)g[v].size()});
        g[v].push_back({u,0,(int)g[u].size()-1});
     }
     void bfs(int st){
         queue<int>q;
         memset(dis,0,sizeof dis);
         q.push(st);dis[st]=1;
         while(q.size()){
             int u=q.front();q.pop();
             for(auto &[v,w,inv]:g[u]){
                 if(w&&!dis[v]){
                     dis[v]=dis[u]+1;
                     q.push(v);
                 }
             }
         }
     }
     int dfs(int u,int t,int flow){
         if(u==t)return flow;
         for(int &i=now[u],sz=g[u].size(),d;i<sz;i++){
             auto &[v,w,inv]=g[u][i];
             if(w&&dis[v]>dis[u]){
                 d=dfs(v,t,min(flow,w));
                 if(d>0){
                     w-=d;
                     g[v][inv].cap+=d;
                     return d;
                 }
             }
         }
         return 0;
     }
     int maxflow(int st,int ed){
         for(int flow=0,res;;){
             bfs(st);
             if(!dis[ed])return flow;
             memset(now,0,sizeof now);
             while((res=dfs(st,ed,inf))>0){
                 flow+=res;
             }
         }
     }
 };
```

Maxflow 网络流最大流

```
// 用 givest 定源点汇点
// addedge 一次加了正反两条边
// init 慎用
// S 必须是 0
// 输出方案注意是 head 开头
#include <bits/stdc++.h>
using namespace std;
const int N=2510,M=2510*10;
class Maxflow{
 private:
  int nedge=1,p[2*M],nex[2*M],head[N],c[2*M],cur[2*M];
     int dist[2*N];
     int S,T;
  void Addedge(int a,int b,int v){
    p[++nedge]=b;nex[nedge]=head[a];head[a]=nedge;
    c[nedge]=v;
   }
     bool bfs(){
     queue<int>q;
    for(int i=S;i<=T;i++)dist[i]=-1;
    dist[S]=0;q.push(S);
    while(!q.empty()){
      int now=q.front();q.pop();
      for(int k=head[now];k;k=nex[k])if(dist[p[k]]==-1&&c[k]>0){
        dist[p[k]]=dist[now]+1;
         q.push(p[k]);
       }
     }
     return dist[T]>-1;
  }
   int dfs(int x,int low){
     if(x==T)return low;
     if(low==0)return 0;
     int used=0;
     for(int &k=cur[x];k;k=nex[k])if(dist[p[k]]==dist[x]+1&&c[k]>0){
       int a=dfs(p[k],min(c[k],low-used));
       c[k]-=a;c[k^1]+=a;used+=a;
      if(low==used)break;
     }
     if(used==0)dist[x]=-1;
    return used;
  }
public:
  void init(int s,int t){
    for(int i=S;i<=T;i++)head[i]=0;
    S=s,T=t;
   nedge=1;
 }
    void addedge(int a,int b,int v){
        Addedge(a,b,v);
        Addedge(b,a,0);
 }
  int dinic(){
   int flow=0;
    while(bfs()){
      for(int i=S;i<=T;i++)cur[i]=head[i];
     flow+=dfs(S,1e9);
    }
    return flow;
 }
};
```

Tarjan 缩点

```
stack<int>s;
vector<int>ve[maxn];
int col[maxn],num,dfn[maxn],low[maxn],dfstime;
void tarjan(int u)
{
    s.push(u);
    dfn[u]=low[u]=++dfstime;
    for(auto v:ve[u])
    {
         if(!dfn[v])
         {
             tarjan(v);
             low[u]=min(low[u],low[v]);
         }
         else if(!col[v]) low[u]=min(low[u],dfn[v]);
     }
     if(dfn[u]==low[u])
     {
         col[u]=++num;
         while(s.top()!=u)
         {
             col[s.top()]=num;
             s.pop();
         }
         s.pop();
     }
}
```

二分图匹配

```
#include <bits/stdc++.h>
using namespace std;
int const N = 1510 * 4, M = 75010;
int e[M], ne[M], h[N], idx;
int n, m, k, match[N], backup[N], st[N];
void add(int a, int b) {
    e[idx] = b, ne[idx] = h[a], h[a] = idx++;
}
int find(int x) {
     for (int i = h[x]; ~i; i = ne[i]) {
         int j = e[i];
         if (!st[j]) {
             st[j] = 1;
             if (!match[j] || find(match[j])) {
                 match[j] = x;
                 return 1;
             }
         }
     }
     return 0;
 }
 int main() {
     cin >> n >> m >> k;
     memset(h, -1, sizeof h);
     for (int i = 1, a, b; i <= k; ++i) {
         scanf("%d%d", &a, &b);
         add(a, b + n);
     }
     int maxMatch = 0;
     for (int i = 1; i <= n; ++i) {
         memset(st, 0, sizeof st);
         if (find(i)) maxMatch++;
     }
     cout<<maxMatch<<endl;
     return 0;
 }

```

二分图最优匹配

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=110;
int n, m;
int a[maxn][maxn];
int lx[maxn], ly[maxn], link[maxn];
bool vx[maxn], vy[maxn];
int dfs(int x)
{
    if(x==-1)return 0;
    vx[x] = 1;
    for (int i = 1; i <= n; i++)
    {
        if (!vy[i] && lx[x] + ly[i] == a[x][i])
        {
            vy[i] = 1;
            if (link[i] == -1 || dfs(link[i]))
            {
                link[i] = x;
                return 1;
            }
        }
    }
    return 0;
}
bool deal()
{
    memset(ly, 0, sizeof(ly));
    memset(lx, 0xf7, sizeof(lx));
    memset(link, -1, sizeof(link));
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++)
            lx[i] = max(lx[i], a[i][j]);
    }
    for(int i = 1; i <= n; i++)
    {
        while(1)
        {
            memset(vx, 0, sizeof(vx));
            memset(vy, 0, sizeof(vy));
            if (dfs(i)) break;
            int delta = 0x7f7f7f7f;
            for (int j= 1; j <= n; j++)
            {
                if (vx[j] == 1)
                    for(int k = 1; k <= n; k++)
                        if (vy[k] == 0) delta = min(delta, lx[j]+ ly[k]- a[j]
[k]);
            }
            if (delta == 0x7f7f7f7f)  return 0;
            for (int j= 1; j <= n; j++)
                if (vx[j] == 1) lx[j] -= delta;
            for(int k = 1; k <= n; k++)
                if (vy[k] == 1) ly[k] += delta;
        }
    }
    return 1;
}
int main()
{
       if  (deal() == 1) {
            int ans = 0;
            for(int i = 1; i <= n; i++)
            {
                ans += a[link[i]][i];
            }
            cout << ans <<'\n';//取最小就把所有边权取负再跑
       }
    return 0;
}
```

二分图染色

```
#include <bits/stdc++.h>
using namespace std;
const int maxn=1010;
int n,m;
vector<int>ve[maxn];
int col[maxn][maxn],ans[maxn*2];
```

```cpp
void dfs(int x,int y,int c1,int c2)
{
    if(col[y][c1]){
         dfs(y,col[y][c1],c2,c1);
         col[x][c1]=y;
         col[y][c1]=x;
     }
     else {
         col[x][c1]=y;
         col[y][c1]=x;
         col[y][c2]=0;
     }
 }
 map<pair<int,int>,int>ma;
 int main()
 {
     ios::sync_with_stdio(false);
     cin.tie(0);
     cin>>n>>m;
     int anss=0;
     for(int i=1;i<=m;i++){
         int x,y;
         cin>>x>>y;
         ve[x].push_back(y);
         ma[{x,y}]=i;
         int c1=1,c2=1;
         while(col[x][c1])c1++;
         while(col[y][c2])c2++;
         anss=max({c1,c2,anss});
         if(c1>c2){
             swap(x,y);swap(c1,c2);
         }
         if(c1==c2){
             col[x][c1]=y;
             col[y][c1]=x;
         }
         else{
             dfs(x,y,c1,c2);
         }
     }
     cout<<anss<<"\n";
     for(int i=1;i<=n;i++){
         for(int j=1;j<=anss;j++)if(col[i][j])ans[ma[{i,col[i][j]}]]=j;
     }
     for(int i=1;i<=m;i++)cout<<ans[i]<<"\n";
     return 0;
 }
```

圆方树

```cpp
    vector<vector<int>> e1(n);
    int cnt = n;

    int now = 0;
    vector<int> dfn(n, -1), low(n);
    vector<int> stk;
    function<void(int)> tarjan = [&](int u) {
        stk.push_back(u);
        dfn[u] = low[u] = now++;
         for (auto v : ve[u]) {
            if (dfn[v] == -1) {
                tarjan(v);
                low[u] = min(low[u], low[v]);
                if (low[v] == dfn[u]) {
                    e1.push_back({});
                    int x;
                    do {
                        x = stk.back();
                        stk.pop_back();
                        e1[cnt].push_back(x);
                    } while (x != v);
                    e1[u].push_back(cnt);
                    ++cnt;
                }
            } else {
                low[u] = min(low[u], dfn[v]);
            }
        }
    };
    tarjan(0);
```

基环树

```
#include <bits/stdc++.h>
using namespace std;
const int maxn=2e5+10;
/*
1 init
2 addedge
3 Get
*/
struct Graph{
    vector<int>ve[maxn];
    int base[maxn],id[maxn];
    bool Incircle[maxn];
    vector<int> Circle;
    int len=0;
    int dep[maxn],f[21][maxn];
    int n;
    void init(int _n){
        n=_n;
        for(int i=1;i<=n;i++){
            for(int j=0;j<21;j++)f[j][i]=0;
            ve[i].clear();
            Incircle[i]=false;
            id[i]=-1;
            base[i]=i;
            Circle.clear();
            len=0;
            dep[i]=0;
        }
    }
    void addedge(int x,int y){
        ve[x].push_back(y);
        ve[y].push_back(x);
    }
    void dfs(int x,int fa)
    {
        base[x]=base[fa];
        dep[x]=dep[fa]+1;
        for(int i=0;i<=19;i++)
            f[i+1][x]=f[i][f[i][x]];
        for(auto it:ve[x])
        {
            if(it==fa) continue;
            f[0][it]=x;
            dfs(it,x);
        }
    }
    void Get(){
        vector<int> sta;
        vector<bool> vis(n+1,false);
        function<bool(int,int)> dfs2 = [&](int x,int h){
            vis[x]=true;
            sta.push_back(x);
            for(auto it:ve[x])if(it!=h){
                if(vis[it]){
                    Circle.push_back(it);
                    while(!sta.empty()&&sta.back()!=it){
                        Circle.push_back(sta.back());
                        sta.pop_back();
                    }
                    return true;
                }
                else{
                    if(dfs2(it,x))return true;
                }
            }
            sta.pop_back();
            return false;
        };
        dfs2(1,0);
        len=(int)Circle.size();
        for(auto it:Circle)Incircle[it]=true;
        for(auto it:Circle){
            for(auto it2:ve[it])if(!Incircle[it2]){
                f[0][it2]=it;
                dfs(it2,it);
            }
        }
        for(int i=0;i<len;i++)id[Circle[i]]=i;
    }
    int lca(int x,int y)
    {
        if(dep[x]<dep[y]) swap(x,y);
        for(int i=20;i>=0;i--)
        {
            if(dep[f[i][x]]>=dep[y]) x=f[i][x];
            if(x==y) return x;
        }
        for(int i=20;i>=0;i--)
            if(f[i][x]!=f[i][y])
                x=f[i][x],y=f[i][y];
        return f[0][x];
    }
    int dis(int x,int y){
        if(base[x]==base[y]){
            int l=lca(x,y);
            return dep[x]+dep[y]-2*dep[l];
        }
        else{
            int g=(id[base[x]]-id[base[y]]+len)%len;
             return dep[x]+dep[y]+min(g,len-g);
         }
     }
};
Graph g;
```

带权并查集 dsu

```
const int maxn=1e5+10;
int f[maxn],dis[maxn];
int getf(int x){
    if(x==f[x])return x;
    int z=getf(f[x]);
    dis[x]+=dis[f[x]];
    return f[x]=z;
}
void unit(int i,int j,int len){
     int x=getf(i),y=getf(j);
     f[x]=y;
     //在赋值之前因为 x 是头节点所以 dis 一定等于 0
     dis[x]=dis[j]-dis[i]+len;
 }
```

带花树

```
#include <bits/stdc++.h>
using namespace std;
struct blossom {//0base
    int n, vis_t;
    vector<vector<int>> E;
    vector<int> match, label, org, vis, parent;
    queue<int> Q;
    blossom(int _n) {
        n = _n;
         E = vector<vector<int>>(n, vector<int>());
         match.assign(n, -1);
         label.resize(n);
         org.resize(n);
         iota(org.begin(), org.end(), 0);
         parent.assign(n, -1);
         vis.assign(n, 0);
         vis_t = 0;
     }
     void addEdge(int u, int v) {
         E[u].emplace_back(v);
         E[v].emplace_back(u);
     }
     auto lca(int v, int u) {
         vis_t++;
         while (true) {
             if (v != -1) {
                 if (vis[v] == vis_t) {
                     return v;
                 }
                 vis[v] = vis_t;
                 if (match[v] == -1) {
                     v = -1;
                 } else {
                     v = org[parent[match[v]]];
                 }
             }
             swap(v, u);
         }
     }
     void agument(int v) {
         while (v != -1) {
             auto pv = parent[v];
             auto nxt = match[pv];
             match[v] = pv;
             match[pv] = v;
             v = nxt;
         }
     }
     void flower(int v, int u, int a) {
         while (org[v] != a) {
             parent[v] = u;
             u = match[v];
             if (label[u] == 1) {
                 label[u] = 0;
                 Q.emplace(u);
             }
             org[v] = org[u] = a;
             v = parent[u];
         }
     }
     auto bfs(int root) {
         fill(label.begin(), label.end(), -1);
         iota(org.begin(), org.end(), 0);
         while (!Q.empty()) {
             Q.pop();
         }
         Q.emplace(root);
         label[root] = 0;
         while (!Q.empty()) {
             auto u = Q.front();
             Q.pop();
             for (auto v : E[u]) {
                 if (label[v] == -1) {
                     label[v] = 1;
                     parent[v] = u;
                     if (match[v] == -1) {
                         agument(v);
                         return true;
                     }
                     label[match[v]] = 0;
                     Q.push(match[v]);
                     continue;
                 } else if (label[v] == 0 && org[v] != org[u]) {
                     auto a = lca(org[u], org[v]);
                     flower(v, u, a);
                     flower(u, v, a);
                 }
             }
         }
         return false;
     }
     void solve() {
         for (int i = 0; i < n; ++i) {
             if (match[i] == -1) {
                 bfs(i);
             }
         }
     }
 };
  int main()
  {
      blossom G(n);
      for (int i = 0; i < n; ++i) {
          for (int j = i + 1; j < n; ++j) {
              auto [xi, yi] = stone[i];
              auto [xj, yj] = stone[j];
              if (abs(xi - xj) + abs(yi - yj) <= L) {
                  G.addEdge(i, j);
              }
          }
      }
      G.solve();
      int num = 0;
      for (int i = 0; i < n; ++i) {
          if (G.match[i] != -1) {
              num++;
          }
      }
  }
```

带负环最小费用最大流

```
#include <bits/stdc++.h>
using namespace std;
const int N = 200 + 5, M = 2e4 + N;
struct flow {
  int cnt = 1, hd[N], nxt[M << 1], to[M << 1], limit[M << 1], cst[M << 1];
  void add(int u, int v, int w, int c) {
    nxt[++cnt] = hd[u], hd[u] = cnt, to[cnt] = v, limit[cnt] = w, cst[cnt] = c;
    nxt[++cnt] = hd[v], hd[v] = cnt, to[cnt] = u, limit[cnt] = 0, cst[cnt] = -c;
  }
   int fl[N], fr[N], dis[N], in[N];
   pair<int, int> mincost(int s, int t) {
     int flow = 0, cost = 0;
     while (1) {
       queue<int> q;
       memset(dis, 0x3f, sizeof(dis));
       q.push(s), fl[s] = 1e9, dis[s] = 0;
       while (!q.empty()) {
         int t = q.front();
         q.pop(), in[t] = 0;
         for (int i = hd[t]; i; i = nxt[i]) {
           int it = to[i], d = dis[t] + cst[i];
           if (limit[i] && d < dis[it]) {
             dis[it] = d, fl[it] = min(fl[t], limit[i]), fr[it] = i;
             if (!in[it]) in[it] = 1, q.push(it);
           }
         }
       }
       if (dis[t] > 1e9) return make_pair(flow, cost);
       flow += fl[t], cost += dis[t] * fl[t];
       for (int u = t; u != s; u = to[fr[u] ^ 1])
         limit[fr[u]] -= fl[t], limit[fr[u] ^ 1] += fl[t];
     }
   }
 };
 struct bounded_flow {
   int e, u[M], v[M], lo[M], hi[M], cst[M];
   void add(int _u, int _v, int w, int c) {
     if (c < 0) {
       u[++e] = _u, v[e] = _v, lo[e] = w, hi[e] = w, cst[e] = c;
       u[++e] = _v, v[e] = _u, lo[e] = 0, hi[e] = w, cst[e] = -c;
     } else
       u[++e] = _u, v[e] = _v, lo[e] = 0, hi[e] = w, cst[e] = c;
   }
   flow g;
   pair<int, int> mincost(int n, int s, int t, int ss, int tt) {
     static int w[N];
     memset(w, 0, sizeof(w));
     int flow = 0, cost = 0, tot = 0;
     for (int i = 1; i <= e; i++) {
       w[u[i]] -= lo[i], w[v[i]] += lo[i];
       cost += lo[i] * cst[i];
       g.add(u[i], v[i], hi[i] - lo[i], cst[i]);
     }
     for (int i = 1; i <= n; i++)
       if (w[i] > 0)
         g.add(ss, i, w[i], 0), tot += w[i];
       else if (w[i] < 0)
         g.add(i, tt, -w[i], 0);
     g.add(t, s, 1e9, 0);
     pair<int, int> res = g.mincost(ss, tt);
     cost += res.second;
     flow += g.limit[g.hd[s]];
     g.hd[s] = g.nxt[g.hd[s]], g.hd[t] = g.nxt[g.hd[t]];
     res = g.mincost(s, t);
     return make_pair(flow + res.first, cost + res.second);
   }
 } f;
 int n, m, s, t;
```

```cpp
int main() {
  cin >> n >> m >> s >> t;
  for (int i = 1; i <= m; i++) {
    int u, v, w, c;
    cin >> u >> v >> w >> c, f.add(u, v, w, c);
  }
  pair<int, int> res = f.mincost(n, s, t, 0, n + 1);
  cout << res.first << " " << res.second << endl;
  return 0;
}

```

支配树

```cpp
/*
1base
注意 up 是数组需要外界导入
使用的时候直接 dtree::即可
*/
namespace dtree{
  const int MAXN = 200020;
 vector<int> E[MAXN], RE[MAXN], rdom[MAXN];

   int S[MAXN], RS[MAXN], cs;
  int par[MAXN], val[MAXN], sdom[MAXN], rp[MAXN], dom[MAXN];

   void clear(int n) {
     cs = 0;
     for(int i=0;i<=n;i++) {
       par[i] = val[i] = sdom[i] = rp[i] = dom[i] = S[i] = RS[i] = 0;
      E[i].clear(); RE[i].clear(); rdom[i].clear();
     }
   }
   void add_edge(int x, int y) { E[x].push_back(y); }
  void Union(int x, int y) { par[x] = y; }
  int Find(int x, int c = 0) {
    if(par[x] == x) return c ? -1 : x;
    int p = Find(par[x], 1);
    if(p == -1) return c ? par[x] : val[x];
     if(sdom[val[x]] > sdom[val[par[x]]]) val[x] = val[par[x]];
    par[x] = p;
     return c ? p : val[x];
  }
   void dfs(int x) {
     RS[ S[x] = ++cs ] = x;
    par[cs] = sdom[cs] = val[cs] = cs;
    for(int e : E[x]) {
       if(S[e] == 0) dfs(e), rp[S[e]] = S[x];
      RE[S[e]].push_back(S[x]);
     }
   }
   int solve(int s, int *up) {//s 是起点
     dfs(s);
     for(int i=cs;i;i--) {
       for(int e : RE[i]) sdom[i] = min(sdom[i], sdom[Find(e)]);
       if(i > 1) rdom[sdom[i]].push_back(i);
       for(int e : rdom[i]) {
        int p = Find(e);
        if(sdom[p] == i) dom[e] = i;
        else dom[e] = p;
      }
      if(i > 1) Union(i, rp[i]);
    }
     for(int i=2;i<=cs;i++) if(sdom[i] != dom[i]) dom[i] = dom[dom[i]];
    for(int i=2;i<=cs;i++) up[RS[i]] = RS[dom[i]];
    return cs;
  }
}
```

最小环

```
#include <bits/stdc++.h>
using namespace std;
const int maxn=1e2+10;
const int inf=2e7+10;
int a[maxn][maxn],b[maxn][maxn];
int main()
{
    int n;cin>>n;
    int m;cin>>m;
     for(int i=1;i<=n;i++){
         for(int j=1;j<=n;j++)a[i][j]=b[i][j]=inf;
         a[i][i]=b[i][i]=0;
     }
     while(m--){
         int x,y;cin>>x>>y;
         int w;cin>>w;
         a[x][y]=min(a[x][y],w);
         a[y][x]=min(a[y][x],w);
         b[x][y]=min(b[x][y],w);
         b[y][x]=min(b[y][x],w);
     }
     int ans=inf;
     for(int i=1;i<=n;i++){
         for(int j=1;j<i;j++){
             for(int k=j+1;k<i;k++){
                 ans=min(ans,a[i][j]+a[i][k]+b[j][k]);
             }
         }
         for(int j=1;j<=n;j++){
             for(int k=1;k<=n;k++)b[j][k]=min(b[j][i]+b[i][k],b[j][k]);
         }
     }
     if(ans==inf)cout<<"No solution.";
     else cout<<ans;
}
```

最小费用最大流

```
#include <bits/stdc++.h>
using namespace std;
const int N=1e4,M=1e6;
struct SSP {
    int cnt = 1, hd[N], nxt[M << 1], to[M << 1], limit[M << 1], cst[M << 1];
    void init(){
        memset(hd,0,sizeof(hd));
        cnt=1;
    }
     // w limit c cost
     void add(int u, int v, int w, int c) {
         nxt[++cnt] = hd[u], hd[u] = cnt, to[cnt] = v, limit[cnt] = w, cst[cnt] = c;
         nxt[++cnt] = hd[v], hd[v] = cnt, to[cnt] = u, limit[cnt] = 0, cst[cnt] = -c;
     }

     int fr[N], fl[N], in[N], dis[N];

     pair<int, int> min_cost(int s, int t) {
         int flow = 0, cost = 0;
         while (true) { // SPFA
             queue<int> q;
             memset(dis, 0x3f, sizeof(dis));
             memset(in, 0, sizeof(in));
             fl[s] = 1e9, dis[s] = 0, q.push(s);
             while (!q.empty()) {
                 int cur = q.front();
                 q.pop(), in[cur] = 0;
                 for (int i = hd[cur]; i; i = nxt[i]) {
                     int it = to[i], d = dis[cur] + cst[i];
                     if (limit[i] && d < dis[it]) {
                         fl[it] = min(limit[i], fl[cur]), fr[it] = i, dis[it] = d;
                         if (!in[it]) in[it] = 1, q.push(it);
                     }
                 }
             }
             if (dis[t] > 1e9) return {flow, cost};//改成>0 就是可行流
             flow += fl[t], cost += dis[t] * fl[t];
               for (int u = t; u != s; u = to[fr[u] ^ 1]) limit[fr[u]] -= fl[t], limit[fr[u] ^ 1] += fl[t];
         }
     }
 } Sol;
```

欧拉回路

```
#include<bits/stdc++.h>
using namespace std;
const int M=2333,N=666;
struct edge
{
 int nxt,to;
}e[M<<1];
int head[N],tot=1;
int cut[M<<1];
 void add(int u,int v)
 {
   e[++tot]=(edge){head[u],v},head[u]=tot;
   e[++tot]=(edge){head[v],u},head[v]=tot;
 }
 vector<int> st;
 void dfs(int u)//欧拉回路
 {
     for(int i=head[u];i!=0;i=e[i].nxt)
     {
         if(cut[i]) continue;
         int v=e[i].to;
         cut[i]=cut[i^1]=1;
         dfs(v);
     }
     st.push(u);
 }
 int main()
 {


   return 0;
 }

```

数据结构

SegmentSet

```
/*
every pair of pair<int,int> will not intersect
if i is true then it will be in the set
*/
struct SegmentSet{
    set<pair<int,int>>s;
    void write(){
        for(auto [x,y]:s)cout<<x<<"/"<<y<<" ";;cout<<"\n";
    }
     //make l...r to be true
     void insert(int l,int r)
     {
         int L=l,R=r;
         auto it=s.lower_bound(make_pair(L,(int)-2e9));
         while(it!=s.end()&&it->first<=R+1){
             R=max(it->second,R);
             it=s.erase(it);
         }
         if(it!=s.begin()){
             it--;
             if(it->second+1>=L){
                 L=min(L,it->first);
                 R=max(R,it->second);
                 s.erase(it);
             }
         }
         s.insert(make_pair(L,R));
     }
     // if l...r all true return false
     bool query_no_full(int l,int r){
         auto it=s.lower_bound(make_pair(l,(int)-2e9));
         if(it!=s.end()){
             if(l==it->first&&r<=it->second)return false;
         }
         if(it!=s.begin()){
             it--;
             if(it->second>=r)return false;
         }
         return true;
     }
     //make l...r to be false
     void del(int l,int r){
         auto it=s.lower_bound(make_pair(l,(int)-2e9));
         while(it!=s.end()&&it->first<=r){
             if(it->second<=r){
                 it=s.erase(it);continue;
             }
             int R=it->second;
             s.erase(it);
             s.insert(make_pair(r+1,R));
             break;
         }
         it=s.lower_bound(make_pair(l,(int)-2e9));
         if(it!=s.begin()){
             it--;
             int L=it->first,R=it->second;
             if(R>=l){
                 s.erase(it);
                 if(L<=l-1)s.insert(make_pair(L,l-1));
                 if(R>=r+1)s.insert(make_pair(r+1,R));
             }
         }
     }
     // if l...r all false return false
     bool query_at_least_one(int l,int r){
         auto it=s.lower_bound(make_pair(l,(int)-2e9));
         if(it!=s.end()){
             if(it->first<=r)return true;
         }
         if(it!=s.begin()){
             it--;
             if(it->second>=l)return true;
         }
         return false;
     }
 };
```

SegmentTree

```
//?????????? ???????>=??sum??????
#include <bits/stdc++.h>
using namespace std;
const int maxn=1e6+10;
struct Node{
    int l,r,res,tag;
};
struct SegmentTree{
    Node a[maxn*4];
    void tag_init(int i){
        a[i].tag=0;
    }
    void tag_union(int fa,int i){
        a[i].tag+=a[fa].tag;
    }
    void tag_cal(int i){
        a[i].res+=a[i].tag*(a[i].r-a[i].l+1);
    }
    void pushdown(int i){
        tag_cal(i);
        if(a[i].l!=a[i].r){
            tag_union(i,i*2);
            tag_union(i,i*2+1);
        }
        tag_init(i);
    }
    void pushup(int i){
        if(a[i].l==a[i].r)return;
        pushdown(i*2);
        pushdown(i*2+1);
        a[i].res=a[i*2].res+a[i*2+1].res;
    }
    void build(int i,int l,int r){
        a[i].l=l,a[i].r=r;tag_init(i);
        if(l>=r)return;
        int mid=(l+r)/2;
        build(i*2,l,mid);
        build(i*2+1,mid+1,r);
    }
    void update(int i,int l,int r,int w){
        pushdown(i);
        if(a[i].r<l||a[i].l>r||l>r)return;
        if(a[i].l>=l&&a[i].r<=r){
            a[i].tag=w;
            return;
        }
        update(i*2,l,r,w);
        update(i*2+1,l,r,w);
        pushup(i);
    }
    int query(int i,int l,int r){
        pushdown(i);
        if(a[i].r<l||a[i].l>r||l>r)return 0;
        if(a[i].l>=l&&a[i].r<=r){
            return a[i].res;
        }
        return query(i*2,l,r)+query(i*2+1,l,r);
    }
    int min_right(int qL, int& nowsum,int querysum, int i) {//???????>=sum???
        pushdown(i);
        if (a[i].r < qL)return -1;
        if (qL <= a[i].l) {
            int ss = nowsum+a[i].res;
            if (ss<querysum) {
                nowsum = ss;
                return -1;
            }
            if (a[i].l == a[i].r)return a[i].l;
        }
        int pos = min_right(qL, nowsum,querysum,i*2);
        if (pos != -1)return pos;
        return min_right(qL, nowsum,querysum,2*i+1);
    }
    int max_left(int qR,int &nowsum,int querysum,int i){//???????>=sum???
        pushdown(i);
        if(a[i].l > qR)return -1;
        if(qR>=a[i].r){
            int ss=nowsum+a[i].res;
            if(ss<querysum){
                nowsum=ss;
                return -1;
            }
            if(a[i].l==a[i].r)return a[i].r;
        }
        int pos=max_left(qR,nowsum,querysum,i*2+1);
        if(pos!=-1)return pos;
        return max_left(qR,nowsum,querysum,i*2);
```

```cpp
    }
};
SegmentTree tri;
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    int n,q;cin>>n>>q;
    tri.build(1,1,n);
    for(int i=1;i<=n;i++){
        int x;cin>>x;tri.update(1,i,i,x);
    }
     while(q--){
         int ops;cin>>ops;
         if(ops==1){
             int l,r,x;cin>>l>>r>>x;
             tri.update(1,l,r,x);
         }
         if(ops==2){
             int l,r;cin>>l>>r;
             cout<<tri.query(1,l,r)<<"\n";
         }
         if(ops==3){
             int x,sum;cin>>x>>sum;
             int nowsum=0;
             cout<<tri.min_right(x,nowsum,sum,1)<<"\n";
         }
         if(ops==4){
             int x,sum;cin>>x>>sum;
             int nowsum=0;
             cout<<tri.max_left(x,nowsum,sum,1)<<"\n";
         }
     }
 }

```

三维偏序 **cdq**

```cpp
#include <bits/stdc++.h>
using namespace std;
const int maxn=4e5+10;
struct Treearray{
    int c[maxn],n;
    void set_n(int _n){
        n=_n;
        for(int i=1;i<=n;i++)c[i]=0;
    }
     int lowbit(int x){
         return x&(-x);
     }
     void update(int i,int k){    //在 i 位置加上 k
         while(i <= n){
             c[i] += k;
             i += lowbit(i);
         }
     }
     int getsum(int i){        //求 A[1 - i]的和
```

```
        int res = 0;
        while(i > 0){
            res += c[i];
            i -= lowbit(i);
        }
        return res;
    }
};
Treearray tri;
int a[maxn],b[maxn],c[maxn];
int n,k;
int ans[maxn],id[maxn],cnt[maxn],t[maxn];
void subdiv(int l,int r){
    if(l==r)return;
    int mid=(l+r)/2;
    subdiv(l,mid);
    subdiv(mid+1,r);
    sort(id+l,id+mid+1,[&](int x,int y){
        if(b[x]==b[y])return c[x]<c[y];
        return b[x]<b[y];
    });
    sort(id+mid+1,id+r+1,[&](int x,int y){
        if(b[x]==b[y])return c[x]<c[y];
        return b[x]<b[y];
    });

    // for(int i=l;i<=r;i++)cout<<id[i]<<" \n"[i==r];
    assert(tri.getsum(tri.n)==0);
    for(int i=l,j=mid+1;i<=mid||j<=r;){
        if(j!=r+1&&(i==mid+1||b[id[i]]>b[id[j]])){
            // cout<<"id="<<id[j]<<" +"<<tri.getsum(c[id[j]])<<"\n";
            ans[id[j]]+=tri.getsum(c[id[j]]);
            j++;
        }
        else{
            tri.update(c[id[i]],t[id[i]]);
            i++;
        }
    }
    for(int i=l;i<=mid;i++)tri.update(c[id[i]],-t[id[i]]);
}
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin>>n>>k;
    tri.set_n(200000);
    vector<tuple<int,int,int>> v(n);
    for(int i=0;i<n;i++){
        int x,y,z;cin>>x>>y>>z;v[i]=make_tuple(x,y,z);
    }
    sort(v.begin(),v.end());
    int g=0;
    for(int i=0;i<n;i++){
        int j=i;
        while(j+1<n&&v[i]==v[j+1])j++;
        g++;
        a[g]=get<0>(v[i]);
        b[g]=get<1>(v[i]);
        c[g]=get<2>(v[i]);
        t[g]=j-i+1;
        i=j;
        // cout<<"g="<<g<<" "<<a[g]<<" "<<b[g]<<" "<<c[g]<<"\n";
    }
    for(int i=1;i<=g;i++)id[i]=i;
    subdiv(1,g);
    for(int i=1;i<=g;i++){
        cnt[ans[i]+t[i]-1]+=t[i];
        // cout<<"i="<<i<<" ans="<<ans[i]<<"\n";
    }
    for(int i=0;i<n;i++)cout<<cnt[i]<<"\n";
    return 0;
}

```

主席树

```
//注意 Sum 和 cnt 的区别
#include <bits/stdc++.h>
using namespace std;
const int maxn = 1e5; // 数据范围
int n, m;
struct Persistent_SegmentTree
{
    int sum[(maxn << 5) + 10], root[maxn + 10], lch[(maxn << 5) + 10],
        rch[(maxn << 5) + 10], cnt[(maxn<<5)+10];
    int tot = 0;
    void init()
    {
        tot = 0;
    }
    int a[maxn + 10];
    void update(int &rot, int pr, int L, int R, int k)
    { // 插入操作
        rot = ++tot;
        lch[rot] = lch[pr];
        rch[rot] = rch[pr];
        sum[rot] = sum[pr] + k;
        cnt[rot] = cnt[pr] + 1;
        if (L == R)
            return;
        int mid = (L + R) >> 1;
        if (k <= mid)
            update(lch[rot], lch[pr],  L, mid,k);
        else
            update(rch[rot], rch[pr],  mid + 1, R,k);
    }
    int getcnt(int s, int t, int L, int R, int l, int r) // s,t 为 root[l],root[r]的
根节点 中所有大小在[l,r]之间数字出现次数
    {
        if (l <= L && R <= r)
            return cnt[t] - cnt[s];
        int res = 0;
        int mid = (L + R) >> 1;
        if (l <= mid)
            res += getcnt(lch[s], lch[t], L, mid, l, r);
        if (r > mid)
            res += getcnt(rch[s], rch[t], mid + 1, R, l, r);
        return res;
    }

    int getsum(int s, int t, int L, int R, int l, int r) // s,t 为 root[l],root[r]的根节点 中所有大小在[l,r]之间数字的和
    {
        if (l <= L && R <= r)
            return sum[t] - sum[s];
        int res = 0;
        int mid = (L + R) >> 1;
        if (l <= mid)
            res += getsum(lch[s], lch[t], L, mid, l, r);
        if (r > mid)
            res += getsum(rch[s], rch[t], mid + 1, R, l, r);
        return res;
    }
    int get_Kth_min_Sum(int s,int t,int l,int r,int k,int &nowsum,int &ans){//return 第 k 小的值
        int ss = nowsum+cnt[t]-cnt[s];
        if (ss<k) {
            nowsum = ss;
            ans+=sum[t]-sum[s];
            return -1;
        }
        if (l == r){
            ans+=(k-nowsum)*l;
            return l;
        }
        int mid=(l+r)/2;
        int pos = get_Kth_min_Sum(lch[s],lch[t],l,mid,k,nowsum,ans);
        if (pos != -1)return pos;
        return get_Kth_min_Sum(rch[s],rch[t],mid+1,r,k,nowsum,ans);
    }
    int get_Kth_max_Sum(int s,int t,int l,int r,int k,int &nowsum,int &ans){//return 第 k 大的值
        int ss = nowsum+cnt[t]-cnt[s];
        if (ss<k) {
            nowsum = ss;
            ans+=sum[t]-sum[s];
            return -1;
        }
        if (l == r){
            ans+=(k-nowsum)*l;
            return l;
        }
        int mid=(l+r)/2;
        int pos = get_Kth_max_Sum(rch[s],rch[t],mid+1,r,k,nowsum,ans);
        if (pos != -1)return pos;
        return get_Kth_max_Sum(lch[s],lch[t],l,mid,k,nowsum,ans);
    }
    int get_upper(int s,int t,int l,int r,int x){//第一个大于等于 x 的数
```

```
        if (r < x)return -1;
        if (x <= l) {
            int ss = cnt[t]-cnt[s];
            if (ss==0) {
                return -1;
            }
            if (l == r)return l;
        }
        int mid=(l+r)/2;
        int pos = get_upper(lch[s],lch[t],l,mid,x);
        if (pos != -1)return pos;
         return get_upper(rch[s],rch[t],mid+1,r,x);
    }
    int get_lower(int s,int t,int l,int r,int x){//第一个小于等于 x 的数
        if(l > x)return -1;
        if(x>=r){
            int ss=cnt[t]-cnt[s];
            if(ss==0){
                return -1;
            }
            if(l==r)return r;
        }
        int mid=(l+r)/2;
        int pos=get_lower(rch[s],rch[t],mid+1,r,x);
        if(pos!=-1)return pos;
        return get_lower(lch[s],lch[t],l,mid,x);
    }
};
Persistent_SegmentTree tri;
int main()
{
    int n,q;
    cin >> n >> q;

    for (int i = 1; i <= n; i++)
    {
        int x;
        cin >> x;
        tri.update(tri.root[i], tri.root[i - 1], 1, n, x);
        assert(1<=x&&x<=n);
    }
    while (q--)
    {
        int ops;cin>>ops;
        int l,r,k,L,R;
        if(ops==1){
            cin>>l>>r>>L>>R;
            cout<<tri.getsum(tri.root[l-1],tri.root[r],1,n,L,R)<<"\n";
        }
        if(ops==2){// get kth max no output -1
            cin>>l>>r>>k;
            int nowsum=0,ans=0;
cout<<tri.get_Kth_max_Sum(tri.root[l-1],tri.root[r],1,n,k,nowsum,ans)<<"\n";
        }
        if(ops==3){// get kth max sum siz<k return allsum
```

```
            cin>>l>>r>>k;
            int nowsum=0,ans=0;
            tri.get_Kth_max_Sum(tri.root[l-1],tri.root[r],1,n,k,nowsum,ans);
            cout<<ans<<"\n";
        }
        if(ops==4){// get kth min no output -1
            cin>>l>>r>>k;
            int nowsum=0,ans=0;
cout<<tri.get_Kth_min_Sum(tri.root[l-1],tri.root[r],1,n,k,nowsum,ans)<<"\n";
        }
        if(ops==5){// get kth min sum siz<k return allsum
            cin>>l>>r>>k;
            int nowsum=0,ans=0;
            tri.get_Kth_min_Sum(tri.root[l-1],tri.root[r],1,n,k,nowsum,ans);
            cout<<ans<<"\n";
        }
        if(ops==6){// get the min element >= k no return -1
            cin>>l>>r>>k;
            cout<<tri.get_upper(tri.root[l-1],tri.root[r],1,n,k)<<"\n";
        }
        if(ops==7){// get the max element <= k no return -1
            cin>>l>>r>>k;
            cout<<tri.get_lower(tri.root[l-1],tri.root[r],1,n,k)<<"\n";
        }
    }
}
```

二维树状数组

```
const int maxn=1010;
struct treearray{
    int mkp1[maxn][maxn],mkp2[maxn][maxn],mkp3[maxn][maxn],mkp4[maxn][maxn];
    inline int lowbit(int x)
    {
        return x&(-x);
    }
    inline void Update(int x,int y,int k)
    {
         for(int i=x;i<=n;i+=lowbit(i))
         {
             for(int j=y;j<=m;j+=lowbit(j))
             {
                 mkp1[i][j]+=k;
                 mkp2[i][j]+=k*x;
                 mkp3[i][j]+=k*y;
                 mkp4[i][j]+=k*x*y;
             }
         }
     }
     inline void update(int a,int b,int x,int y,int k)
     {
         Update(a,b,k);
         Update(a,y+1,-k);
         Update(x+1,b,-k);
         Update(x+1,y+1,k);
     }
    inline int Query(int x,int y)
    {
        int ans=0;
        for(int i=x;i>=1;i-=lowbit(i))
        {
            for(int j=y;j>=1;j-=lowbit(j))
            {
                ans+=(x+1)*(y+1)*mkp1[i][j]
                -(y+1)*mkp2[i][j]-(x+1)*mkp3[i][j]+mkp4[i][j];
            }
        }
        return ans;
    }
    inline int query(int a,int b,int x,int y)
    {
        return Query(x,y)+Query(a-1,b-1)-Query(x,b-1)-Query(a-1,y);
    }
};
treearray tri;
int main()
{

}

```

动态开点线段树

```
// root 表示整棵线段树的根结点; cnt 表示当前结点个数
const int maxn=1e5+10;
int n, cnt, root;
int sum[maxn*20], ls[maxn*20], rs[maxn*20];

// 用法: update(root, 1, n, x, f); 其中 x 为待修改节点的编号
void update(int& p, int L, int R, int x, int f) {  // 引用传参
    if (!p) p = ++cnt;  // 当结点为空时, 创建一个新的结点
    if (L == R) {
        sum[p] += f;
        return;
    }
    int m = L + ((R - L) >> 1);
    if (x <= m)
        update(ls[p], L, m, x, f);
    else
        update(rs[p], m + 1, R, x, f);
    sum[p] = sum[ls[p]] + sum[rs[p]];  // pushup
}
// 用法: query(root, 1, n, l, r);
int query(int p, int L, int R, int l, int r) {
    if (!p) return 0;  // 如果结点为空, 返回 0
    if (L >= l && R <= r) return sum[p];
    int m = L + ((R - L) >> 1), ans = 0;
    if (l <= m) ans += query(ls[p], L, m, l, r);
    if (r > m) ans += query(rs[p], m + 1, R, l, r);
    return ans;
}
int merge(int a, int b, int l, int r) {
    if (!a) return b;
    if (!b) return a;
    if (l == r) {
        sum[a]+=sum[b];
        return a;
    }
    int mid = (l + r) >> 1;
    ls[a] = merge(ls[a], ls[b], l, mid);
    rs[a] = merge(rs[a], rs[b], mid + 1, r);
    sum[a]=sum[ls[a]]+sum[rs[a]];
    return a;
}
void split(int &p, int &q, int L, int R, int l, int r) {//p 原树 q 新树
    if (R < l || r < L) return;
    if (!p) return;
    if (l <= L && R <= r) {
        q = p;
        p = 0;
        return;
    }
    if (!q) q = ++cnt;
    int m = L + R >> 1;
    if (l <= m) split(ls[p], ls[q], L, m, l, r);
    if (m < r) split(rs[p], rs[q], m + 1, R, l, r);
    sum[p]=sum[ls[p]]+sum[rs[p]];
    sum[q]=sum[ls[q]]+sum[rs[q]];
}
int query1(int& Sum,int Up, int rt, int l, int r) {//min_x of f(x)+x>Up
    // cout<<"Sum="<<Sum<<" Up="<<Up<<" rt="<<rt<<" l="<<l<<" r="<<r<<"\n";
    if(!rt){
        if(r+Sum<=Up)return -1;
        else return r-(r+Sum-Up)+1;
    }
    if (true) {
        int ss = Sum+sum[rt];
        if (ss+r<=Up) {
            Sum = ss;
            return -1;
        }
        if (l == r)return l;
    }
    int mid = (l + r) / 2;
    int pos = query1(Sum,Up,ls[rt], l, mid);
    if (pos != -1)return pos;
    return query1(Sum,Up,rs[rt], mid + 1, r);
}
```

可持久化 01trie

```
const int maxn=1e5+10;
struct Persistent_Trie
{
    int ch[maxn][2],tot,sum[maxn];
    void clear()
    {
        tot=0;
        return;
    }
    void cpy(int from,int to)
    {
        ch[to][0]=ch[from][0];
        ch[to][1]=ch[from][1];
        sum[to]=sum[from];
        return;
    }
    void insert(int &root,int old,int num,int nowbit)
    {
        root=++tot;
        cpy(old,root);
        sum[root]++;
        if(nowbit==-1)return;
        if(num&(1<<nowbit))
        {
            insert(ch[root][1],ch[root][1],num,nowbit-1);
        }
        else
        {
            insert(ch[root][0],ch[root][0],num,nowbit-1);
        }
    }
    int query(int s,int t,int x)
    {
        int ans=0;
        for(int i=30;i>=0;--i)
        {
            if(sum[ch[t][!(x&(1<<i))]]-sum[ch[s][!(x&(1<<i))]])
            {
                t=ch[t][!(x&(1<<i))];
                s=ch[s][!(x&(1<<i))];
                ans|=(1<<i);
            }
            else
            {
                t=ch[t][!!(x&(1<<i))];
                s=ch[s][!!(x&(1<<i))];
            }
        }
        return ans;
    }
};
```

最近公共祖先 LCA

```
const int maxn=1e5+10;
vector<int>ve[maxn];
int dep[maxn],f[21][maxn];
void dfs(int x,int fa)
{
  dep[x]=dep[fa]+1;
  for(int i=0;i<=19;i++)
   f[i+1][x]=f[i][f[i][x]];
    for(auto it:ve[x])
   {
     if(it==fa) continue;
    f[0][it]=x;
    dfs(it,x);
 }
}
int lca(int x,int y)
{
 if(dep[x]<dep[y]) swap(x,y);
 for(int i=20;i>=0;i--)
 {
    if(dep[f[i][x]]>=dep[y]) x=f[i][x];
    if(x==y) return x;
 }
  for(int i=20;i>=0;i--)
   if(f[i][x]!=f[i][y])
     x=f[i][x],y=f[i][y];
 return f[0][x];
}
```

Lca(o1)

```
#define int long long
const int maxn=1e5+10;//注意开两倍大小的空间 在 dp 上
vector<pair<int,int>>ve[maxn];
int dep[maxn];
pair<int,int>dp[21][maxn*3];
int red[maxn],d[maxn];
int Dep[maxn];
int dfn[maxn];
void dfs(int x,int fa,int l,int dis)
{
    if(red[x])dis=0;
    d[x]=dis;
 dep[x]=dep[fa]+1;
  Dep[x]=Dep[fa]+l;
    for(auto [it,len]:ve[x])
 {
    if(it==fa) continue;
   dfs(it,x,len,dis+len);
 }
}
vector<int> sp;
void dfs2(int u, int fa)
{

    dfn[u] = sp.size();
    sp.push_back(u);
    for (auto& e : ve[u])if(e.first!=fa)
    {
        int& v = e.first;
        dfs2(v, u);
        sp.push_back(u);
    }
}
void initrmq()
{
    int n = sp.size();
    for (int i = 0; i < n; i++) dp[0][i] = {dfn[sp[i]], sp[i]};
    for (int i = 1; (1 << i) <= n; i++)
```

```
        for (int j = 0; j + (1 << i) - 1 < n; j++)
            dp[i][j] = min(dp[i - 1][j], dp[i - 1][j + (1 << (i - 1))]);
}
int lca(int u, int v)
{
    int l = dfn[u], r = dfn[v];
    if (l > r) swap(l, r);
    int k = __lg(r-l+1);
    return min(dp[k][l], dp[k][r - (1 << k) + 1]).second;
}
```

点分治

```
#include <bits/stdc++.h>
using namespace std;
const int maxn = 100010;
const int inf =2e9;
int n, siz[maxn], dist[maxn], vis[maxn],maxx[maxn];
vector<pair<int, int>> ve[maxn];

void add_edge(int x, int y, int z)
{
    ve[x].emplace_back(y, z);
}

void calcsiz(int x, int fa, int sum, int &rt)
{
    siz[x] = 1;
    maxx[x] = 0;
    for (auto &[to, len] : ve[x])
        if (to != fa && !vis[to])
        {
            calcsiz(to, x, sum, rt);
            maxx[x] = max(maxx[x], siz[to]);
            siz[x] += siz[to];
        }
    maxx[x] = max(maxx[x], sum - siz[x]);
    if (maxx[x] < maxx[rt])
        rt = x;
}


void calcdist(int x, int fa)
{
    for (auto &[to, len] : ve[x])
        if (to != fa && !vis[to])
        {
            dist[to] = dist[x] + len, calcdist(to, x);
        }
}
void dfs(int x, int fa)
{
    vis[x] = true;
    for (auto &[to, len] : ve[x])
        if (to != fa && !vis[to])
        {
            dist[to] = len;
            calcdist(to, x);
        }


    for (auto &[to, len] : ve[x])
        if (to != fa && !vis[to])
        {
            int sum = siz[to];
            int rt = 0;
            calcsiz(to, x, sum, rt);
            calcsiz(rt, -1, sum, rt);
            dfs(rt, x);
        }
}

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    cin >> n ;
    for (int i = 1; i < n; i++)
    {
        int a, b,c;
        cin >> a >> b>>c, add_edge(a, b, c), add_edge(b, a, c);
    }
    int rt = 0;
    maxx[rt] = inf;
    int sum = n;
    calcsiz(1, -1, sum, rt);
    calcsiz(rt, -1, sum, rt);
    dfs(rt, -1);
    return 0;
}

```

珂朵莉树

```
#include <bits/stdc++.h>
using namespace std;
struct SegmentMap{
    map<int,int> ma;
    int sum=0;
    int n;
    SegmentMap(int _n=0){
        n=_n;
        ma[1]=0;
         ma[n+1]=-1;
     }
     void del(int l,int r,int x){//减去当前 (l,r,x) 的贡献
         sum-=(r-l+1)*x;
     }
     void add(int l,int r,int x){//加上当前 (l,r,x) 的贡献
         sum+=(r-l+1)*x;
     }
     void split(int x){
         auto t=prev(ma.upper_bound(x))->second;
         ma[x]=t;
```

```cpp
    }
    void update(int l,int r,auto&& T){
        split(l);split(r+1);
        auto it=prev(ma.upper_bound(l));
        int pr=-1;
        while(it->first<=r){
            int nowl=it->first,nowr=next(it)->first-1;
            int x=it->second;
            del(nowl,nowr,x);
            add(nowl,nowr,T(x));
            if(l!=nowl&&pr==T(x))it=ma.erase(it);
            else {
                it->second=T(x);
                it=next(it);
            }
            pr=T(x);
        }
        if(it->first!=n+1&&it->second==pr)ma.erase(it);
        if(l!=1){
            it=ma.lower_bound(l);
            if(prev(it)->second==it->second)ma.erase(it);
        }
    }
};
int main()
{
    ios::sync_with_stdio(false);
    cin.tie(0);
    int n,q;cin>>n>>q;
    SegmentMap sol(n);
    while(q--){
        int op;cin>>op;
        if(op==1){
            int l,r;cin>>l>>r;
            sol.update(l,r,[](int x){
                return (int)sqrt(x);
            });
        }
        else{
            int l,r,x;cin>>l>>r>>x;
            sol.update(l,r,[x](int y){
                return x;
            });
        }
        cout<<sol.sum<<"\n";
    }
}
```

笛卡尔树

```cpp
const int maxn=2e5+10;
vector<int> ve[maxn];
int a[maxn];
int n;
int build(){
    int top=0;
    vector<int> Stack(n+1,0);
    for(int i=0;i<=n;i++){
        ve[i].clear();
         ve[i].resize(2,-1);
     }
  Stack[++top]=1;
   for(int i=2;i<=n;i++){
    while(top&&a[Stack[top]]>=a[i])top--;
     if(!top)ve[i][0]=Stack[top+1];
    else ve[i][0]=ve[Stack[top]][1],ve[Stack[top]][1]=i;
    Stack[++top]=i;
   }
     return Stack[1];
}
```

线段树区间加区间历史最小值

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int maxn=1e6+10;
const int inf =2e9;
struct Node{
    int l,r,res,tag,tag2,res2;
};
struct SegmentTree{
     Node a[maxn*4];
     void tag_init(int i){
         a[i].tag=a[i].tag2=0;
     }
     void tag_union(int fa,int i){
         if(a[fa].tag2<0)a[i].tag2=min(a[i].tag2,a[i].tag+a[fa].tag2);
         a[i].tag+=a[fa].tag;
     }
     void tag_cal(int i){
         if(a[i].tag2<0)a[i].res2=min(a[i].res2,a[i].res+a[i].tag2);
         a[i].res+=a[i].tag;
     }
     void pushdown(int i){
         tag_cal(i);
         if(a[i].l!=a[i].r){
             tag_union(i,i*2);
             tag_union(i,i*2+1);
         }
         tag_init(i);
     }
     void pushup(int i){
         if(a[i].l==a[i].r)return;
         pushdown(i*2);
         pushdown(i*2+1);
         a[i].res=min(a[i*2].res,a[i*2+1].res);
         a[i].res2=min(a[i*2].res2,a[i*2+1].res2);
     }
     void build(int i,int l,int r){
         a[i].l=l,a[i].r=r;tag_init(i);
         if(l>=r)return;
         int mid=(l+r)/2;
         build(i*2,l,mid);
```

```
        build(i*2+1,mid+1,r);
    }
    void update(int i,int l,int r,int w){
        if(a[i].r<l||a[i].l>r||l>r)return;
        pushdown(i);
        if(a[i].l>=l&&a[i].r<=r){
            a[i].tag+=w;
            a[i].tag2=min(a[i].tag2,a[i].tag);
            return;
        }
        update(i*2,l,r,w);
        update(i*2+1,l,r,w);
        pushup(i);
    }
    int query(int i,int l,int r){
        pushdown(i);
        if(a[i].r<l||a[i].l>r||l>r)return inf;
        if(a[i].l>=l&&a[i].r<=r){
            return a[i].res2;
        }
        return min(query(i*2,l,r),query(i*2+1,l,r));
    }
};
SegmentTree tri;
signed main()
{
    int n,m;
    cin>>n>>m;
    tri.build(1,1,n);
    for(int i=1;i<=m;i++){
        int ops;cin>>ops;
        if(ops==1){
            int l,r,x;cin>>l>>r>>x;
            tri.update(1,l,r,x);
        }
        else{
            int l,r;cin>>l>>r;
            cout<<tri.query(1,l,r)<<"\n";
        }
    }
    return 0;
}
```

莫队

```
#include<bits/stdc++.h>
using namespace std;
const int N=1e6+5;
int n,m,a[N],block,ans[N],now,cnt[N];
struct node
{
    int l,r,i;
    friend bool operator<(const node a,const node b)
    {
        if((a.l/block)==(b.l/block)) return ((a.l/block%2)?a.r>b.r:a.r<b.r);
        else return a.l<b.l;
     }
```

```cpp
}q[N];
void add(int x)
{
}
void del(int x)
{
}
int main()
{
    block=max(1,n/sqrt(m));
    sort(q+1,q+1+m);
    int l=1,r=0;
    for(int i=1;i<=m;i++)
    {
        while(l<q[i].l) del(l++);
        while(l>q[i].l) add(--l);
        while(r<q[i].r) add(++r);
        while(r>q[i].r) del(r--);
        ans[q[i].i]=now;
    }
    for(int i=1;i<=m;i++)
        printf("%d\n",ans[i]);
    return 0;
}

```

虚树

```cpp
//带边权的虚树
//ve2 是虚树,ve 是原树
//注意使用时 mlen 的意义需要修改 getlen 和 dfs 预处理的时候都要修改
//用的时候就 addintoxs(x)丢进去，每次 init 就全部删除
//add 是加原边，ADD 不用管它
//solve 里面是必备的几步，不能去除
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int maxn=1e5+10;
const int inf =1e18;
int n,m;
struct XS{
    struct kkk{
        int to,len;
    };
    int dep[maxn],f[maxn][21],id[maxn];
    int mlen[maxn][21],ans[maxn][21];
    int state[maxn];
    int sum[maxn],all;
    vector<kkk> ve[maxn],ve2[maxn];
    int now;//统计 dfs 序
    void dfs(int u,int fa)
    {
        id[u]=now++;
        dep[u]=dep[fa]+1;
        for(int i=0;i<=19;i++){
            f[u][i+1]=f[f[u][i]][i];
            mlen[u][i+1]=mlen[f[u][i]][i]+mlen[u][i];
        }
        for(auto v:ve[u])
        {
            if(v.to==fa) continue;
            f[v.to][0]=u;
            mlen[v.to][0]=v.len;
            dfs(v.to,u);
        }
    }
    int lca(int x,int y)
    {
        if(dep[x]<dep[y]) swap(x,y);
        for(int i=20;i>=0;i--)
        {
            if(dep[f[x][i]]>=dep[y]) x=f[x][i];
            if(x==y) return x;
        }
        for(int i=20;i>=0;i--)
            if(f[x][i]!=f[y][i])
                x=f[x][i],y=f[y][i];
        return f[x][0];
    }
    int getlen(int x,int y)
    {
        if(x==y)return 0;
        if(dep[x]<dep[y]) swap(x,y);
        int res=0;
        for(int i=20;i>=0;i--){
            if(dep[f[x][i]]>=dep[y]){
                res=res+mlen[x][i];
                x=f[x][i];
            }
            if(x==y)return res;
        }
        return inf;
    }
    void add(int x,int y,int len){//加边
        ve[x].push_back({y,len});
        ve[y].push_back({x,len});
    }
    void ADD(int x,int y,int len){//虚树加边
        ve2[x].push_back({y,len});
        ve2[y].push_back({x,len});
    }
    vector<int> vis;//b vis 这两个用来初始化
    bool b[maxn];
    vector<int> v1;//用来表示加入虚树的点
    int sta[maxn];
    void init()//删除虚树
    {
        for(auto it:vis){
            ve2[it].clear();
            b[it]=false;
            sum[it]=0;
        }
        vis.clear();
```

```cpp
        v1.clear();
    }
    void makexs()
    {
        sort(v1.begin(),v1.end(),[&](int x,int y){
            return id[x]<id[y];
        });
        sta[0]=0;
        int top=1;
        sta[top]=1;
        vis.push_back(1);
        for(auto it:v1){
            int l=lca(it,sta[top]);
            if(l!=sta[top]){
                while(id[l]<id[sta[top-1]]){
                    int w=getlen(sta[top],sta[top-1]);
                    ADD(sta[top-1],sta[top],w);
                    vis.push_back(sta[top--]);
                }
                if(id[l]>id[sta[top-1]]){
                    int w=getlen(sta[top],l);
                    ADD(l,sta[top],w);
                    vis.push_back(sta[top]);
                    sta[top]=l;
                }
                else {
                    int w=getlen(sta[top],sta[top-1]);
                    ADD(sta[top],sta[top-1],w);
                    vis.push_back(sta[top--]);
                }
            }
            sta[++top]=it;
        }
        for(int i=1;i<top;i++){
            int w=getlen(sta[i],sta[i+1]);
            ADD(sta[i],sta[i+1],w);
            vis.push_back(sta[i+1]);
        }
    }
    void addintoxs(int x){
        if(x==1)return;
        v1.push_back(x);
        b[x]=true;
    }
    void solve(){
        now=0;
        dfs(1,0);
        for(int i=1;i<=n;i++){
            makexs();
            init();
        }
    }
};
XS xs;
```

LCT 维护联通性

```cpp
#include <bits/stdc++.h>
using namespace std;
const int maxn = 10010;

struct LCT {
    int ch[maxn][2], fa[maxn], tag[maxn];

    void clear(int x) { ch[x][0] = ch[x][1] = fa[x] = tag[x] = 0; }

     int getch(int x) { return ch[fa[x]][1] == x; }

     int isroot(int x) { return ch[fa[x]][0] != x && ch[fa[x]][1] != x; }

     void pushdown(int x) {
         if (tag[x]) {
         if (ch[x][0]) swap(ch[ch[x][0]][0], ch[ch[x][0]][1]), tag[ch[x][0]] ^= 1;
         if (ch[x][1]) swap(ch[ch[x][1]][0], ch[ch[x][1]][1]), tag[ch[x][1]] ^= 1;
         tag[x] = 0;
         }
     }

     void update(int x) {
         if (!isroot(x)) update(fa[x]);
         pushdown(x);
     }

     void rotate(int x) {
         int y = fa[x], z = fa[y], chx = getch(x), chy = getch(y);
         fa[x] = z;
         if (!isroot(y)) ch[z][chy] = x;
         ch[y][chx] = ch[x][chx ^ 1];
         fa[ch[x][chx ^ 1]] = y;
         ch[x][chx ^ 1] = y;
         fa[y] = x;
     }

     void splay(int x) {
         update(x);
         for (int f = fa[x]; f = fa[x], !isroot(x); rotate(x))
         if (!isroot(f)) rotate(getch(x) == getch(f) ? f : x);
     }

     void access(int x) {
         for (int f = 0; x; f = x, x = fa[x]) splay(x), ch[x][1] = f;
     }

     void makeroot(int x) {
         access(x);
         splay(x);
         swap(ch[x][0], ch[x][1]);
         tag[x] ^= 1;
     }

     int find(int x) {
         access(x);
         splay(x);
```

```
        while (ch[x][0]) x = ch[x][0];
        splay(x);
        return x;
    }
/*------------------------------------------------------*/
    bool query(int x,int y){//查询是否为同一颗树
        return find(x)==find(y);
    }
    void addedge(int x,int y){
        if (find(x) != find(y)) makeroot(x), fa[x] = y;
    }
    void deledge(int x,int y){
        makeroot(x);
        access(y);
        splay(y);
        if (ch[y][0] == x && !ch[x][1]) ch[y][0] = fa[x] = 0;
    }
} st;

int n, q, x, y;
char op[maxn];

int main() {
    scanf("%d%d", &n, &q);
    while (q--) {
        scanf("%s%d%d", op, &x, &y);
        if (op[0] == 'Q') {
            if (st.query(x,y))
            printf("Yes\n");
        else
            printf("No\n");
        }
        if (op[0] == 'C')st.addedge(x,y);
        if (op[0] == 'D')st.deledge(x,y);
    }
    return 0;
}
```

树链剖分

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int maxn=1e5+10;
int mod;
void add(int &x,int y){if((x+=y)>=mod)x-=mod;}
struct Node{
    int l,r,res,tag;
};
struct SegmentTree{
    Node a[maxn*4];
    void tag_init(int i){
        a[i].tag=0;
    }
    void tag_union(int fa,int i){
        add(a[i].tag,a[fa].tag);
    }
    void tag_cal(int i){
        add(a[i].res,(ll)a[i].tag*(a[i].r-a[i].l+1)%mod);
    }
    void pushdown(int i){
        tag_cal(i);
        if(a[i].l!=a[i].r){
            tag_union(i,i*2);
            tag_union(i,i*2+1);
        }
        tag_init(i);
    }
    void pushup(int i){
        if(a[i].l==a[i].r)return;
        pushdown(i*2);
        pushdown(i*2+1);
        a[i].res=(a[i*2].res+a[i*2+1].res)%mod;
    }
    void build(int i,int l,int r){
        a[i].l=l,a[i].r=r;tag_init(i);a[i].res=0;
        if(l>=r)return;
        int mid=(l+r)/2;
        build(i*2,l,mid);
        build(i*2+1,mid+1,r);
    }
    void update(int i,int l,int r,int w){
        pushdown(i);
        if(a[i].r<l||a[i].l>r||l>r)return;
        if(a[i].l>=l&&a[i].r<=r){
            a[i].tag=w;
            return;
        }
        update(i*2,l,r,w);
        update(i*2+1,l,r,w);
        pushup(i);
    }
    int query(int i,int l,int r){
        pushdown(i);
        if(a[i].r<l||a[i].l>r||l>r)return 0;
        if(a[i].l>=l&&a[i].r<=r){
            return a[i].res;
        }
        return (query(i*2,l,r)+query(i*2+1,l,r))%mod;
    }
};
SegmentTree tri;
vector<int> ve[maxn];
int L[maxn],R[maxn],tot,root=1;
int siz[maxn],top[maxn],dep[maxn],fa[maxn],dfn[maxn];
void dfs_sz(int x,int h){
    if(x!=root)ve[x].erase(find(ve[x].begin(),ve[x].end(),h));
    siz[x] = 1;
    dep[x]=dep[h]+1;
    fa[x]=h;
    for(auto &it:ve[x]) {
        dfs_sz(it,x);
        siz[x] += siz[it];
        if(siz[it] > siz[ve[x][0]]) {
            swap(it, ve[x][0]);
        }
    }
}
void dfs_hld(int x) {
    L[x] =++tot;
    dfn[tot]=x;
    for(auto it: ve[x]) {
        top[it] = (it == ve[x][0] ? top[x] : it);
        dfs_hld(it);
    }
    R[x] =tot;
}
void chain_add(int x, int y, int w) {// chain add w
    while(top[x] != top[y]) {
    if(dep[top[x]] < dep[top[y]]) swap(x, y);
    tri.update(1, L[top[x]], L[x], w);
        x = fa[top[x]];
 }
  if(L[x] > L[y]) swap(x, y);
  tri.update(1, L[x], L[y], w);
}
int chain_sum(int x, int y) {// query the length of chain
  int sum=0;
    while(top[x] != top[y]) {
    if(dep[top[x]] < dep[top[y]]) swap(x, y);
     add(sum,tri.query(1, L[top[x]], L[x]));
         x = fa[top[x]];
   }
   if(L[x] > L[y]) swap(x, y);
   add(sum,tri.query(1, L[x], L[y]));
     return sum;
 }
 int main()
 {
     ios::sync_with_stdio(false);
     cin.tie(0);
     int n,q;cin>>n>>q>>root>>mod;
     vector<int> a(n+1);
     for(int i=1;i<=n;i++)cin>>a[i],a[i]%=mod;
     for(int i=1;i<n;i++){
         int x,y;cin>>x>>y;
         ve[x].push_back(y);
         ve[y].push_back(x);
     }
     top[root]=root;
     tri.build(1,1,n);dfs_sz(root,0);dfs_hld(root);
     for(int i=1;i<=n;i++){
         tri.update(1,L[i],L[i],a[i]);
     }
     while(q--){
         int op;cin>>op;
         if(op==1){
             int x,y,w;cin>>x>>y>>w;
             chain_add(x,y,w);
         }
         if(op==2){
             int x,y;cin>>x>>y;
             cout<<chain_sum(x,y)<<"\n";
         }
         if(op==3){//update the subtree of root x +=y
             int x,y;cin>>x>>y;
             tri.update(1,L[x],R[x],y);
         }
         if(op==4){
             int x;cin>>x;
             cout<<tri.query(1,L[x],R[x])<<"\n";
         }
     }

 }
```

数学

组合数

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
const int maxn=100010;
const int mod=1e9+7;
int ksm(int x,int k){
    int res=1;
    while(k){
        if(k&1)res=res*x%mod;
        x=x*x%mod;
        k/=2;
    }
    return res;
}
int ny(int x){
    return ksm(x,mod-2);
}
void add(int &x,int y){
    if((x+=y)>=mod)x-=mod;
}
void del(int &x,int y){
    if((x-=y)<0)x+=mod;
}
int inv[maxn],fac[maxn];
int C(int n,int m){return n==0?1:fac[n]*inv[n-m]%mod*inv[m]%mod;}
int A(int n,int m){return n==0?1:fac[n]*inv[n-m]%mod;}
void init(){
    inv[0]=fac[0]=1;
    inv[1]=1;
    for(int i=1;i<maxn;i++){
        fac[i]=fac[i-1]*i%mod;
    }
    inv[1]=1;
    for(int i=2;i<maxn;i++){
        inv[i]=(int)(mod-mod/i)*inv[mod%i]%mod;
    }
    inv[0]=1;
    for(int i=1;i<maxn;i++){
        inv[i]=inv[i-1]*inv[i]%mod;
    }
}
signed main()
{
    init();
    if(mod==(int)(1e9+7))assert(C(2000,1000)==72475738);
    if(mod==9982444353)assert(C(2000,1000)==472799582);
}
```

BSGS 指数方程余数问题(求 a^x=b%p)

```
#include <cstdio>
#include <cstring>
#include <cmath>
#include <algorithm>
```

```
#include <unordered_map>

using namespace std;

typedef long long LL;

const int INF = 0x3f3f3f3f;

int a, b, p;
unordered_map<int, int> hs;

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

int BSGS(int a, int b, int p) {
    if (1 % p == b % p) return 0;
    int k = sqrt(p) + 1;
    hs.clear();
    for (int y = 0, r = b % p; y < k; y++) {
        hs[r] = y;
        r = (LL)r * a % p;
    }
    int ak = 1;
    for (int i = 1; i <= k; i++) ak = (LL)ak * a % p;
    for (int x = 1, l = ak; x <= k; x++) {
        if (hs.count(l)) return k * x - hs[l];
        l = (LL)l * ak % p;
    }
    return -INF;
}

int exBSGS(int a, int b, int p) {
    b = (b % p + p) % p;
    if (1 % p == b % p) return 0;
    int x, y;
    int d = exgcd(a, p, x, y);
    if (d > 1) {
        if (b % d) return -INF;
        exgcd(a / d, p / d, x, y);
        return exBSGS(a, (LL)b / d * x % (p / d), p / d) + 1;
    }
    return BSGS(a, b, p);
}

int main() {
    while (~scanf("%d%d%d", &a, &p, &b), a || b || p) {
        int res = exBSGS(a, b, p);
        if (res < 0) puts("No Solution");
        else printf("%d\n", res);
    }
    return 0;
}
```

EXGCD

```
int exgcd(int a, int b, int &x, int &y){//求 ax+by=gcd(a,b)  !(a==0&&b==0)
    if(b==0){
        x=1;
        y=0;
        return a;
    }
    int d=exgcd(b,a%b,x,y);
    int t=x;
    x=y;
     y=t-(a/b)*y;
     return d;
}
```

FFT

```
//当 vector 用就可以了
#include <bits/stdc++.h>
#define fp(i, a, b) for (int i = (a), i##_ = (b) + 1; i < i##_; ++i)
#define fd(i, a, b) for (int i = (a), i##_ = (b) - 1; i > i##_; --i)
using namespace std;
using ll = int64_t;
using db = double;
/*---------------------------------------------------------------------------*/
struct cp {
     db x, y;
     cp(db real = 0, db imag = 0) : x(real), y(imag){};
     cp operator+(cp b) const { return {x + b.x, y + b.y}; }
     cp operator-(cp b) const { return {x - b.x, y - b.y}; }
     cp operator*(cp b) const { return {x * b.x - y * b.y, x * b.y + y * b.x}; }
 };
 using vcp = vector<cp>;
 using Poly = vector<int>;
 namespace FFT {
     const db pi = acos(-1);
     vcp Omega(int L) {
         vcp w(L); w[1] = 1;
         for (int i = 2; i < L; i <<= 1) {
             auto w0 = w.begin() + i / 2, w1 = w.begin() + i;
             cp wn(cos(pi / i), sin(pi / i));
             for (int j = 0; j < i; j += 2)
                 w1[j] = w0[j >> 1], w1[j + 1] = w1[j] * wn;
         }
         return w;
     }
     auto W = Omega(1 << 21); // NOLINT
     void DIF(cp *a, int n) {
         cp x, y;
         for (int k = n >> 1; k; k >>= 1)
             for (int i = 0; i < n; i += k << 1)
                 for (int j = 0; j < k; ++j)
                     x = a[i + j], y = a[i + j + k],
                     a[i + j + k] = (a[i + j] - y) *  W[k + j], a[i + j] = x + y;
```

```cpp
    }
    void IDIT(cp *a, int n) {
        cp x, y;
        for (int k = 1; k < n; k <<= 1)
            for (int i = 0; i < n; i += k << 1)
                for (int j = 0; j < k; ++j)
                    x = a[i + j], y = a[i + j + k] * W[k + j],
                    a[i + j + k] = x - y, a[i + j] = x + y;
        const db Inv = 1. / n;
        fp(i, 0, n - 1) a[i].x *= Inv, a[i].y *= Inv;
        reverse(a + 1, a + n);
    }
}

namespace Polynomial {
    // basic operator
    void DFT(vcp &a) { FFT::DIF(a.data(), a.size()); }
    void IDFT(vcp &a) { FFT::IDIT(a.data(), a.size()); }
    int norm(int n) { return 1 << (__lg(n - 1) + 1); }

    // Poly mul
    vcp &dot(vcp &a, vcp &b) { fp(i, 0, a.size() - 1) a[i] = a[i] * b[i]; return
a; }
    Poly operator+(Poly a, Poly b) {
        int maxlen = max(a.size(), b.size());
        Poly ans(maxlen + 1);
        a.resize(maxlen + 1), b.resize(maxlen + 1);
        for (int i = 0; i < maxlen;i++)
            ans[i] = a[i] + b[i];
        return ans;
    }
    Poly operator*(ll k, Poly a) {
        Poly ans;
        for(auto i:a)
            ans.push_back(k * i);
        return ans;
    }
    Poly operator*(Poly a, Poly b) {
        int n = a.size() + b.size() - 1;
        vcp c(norm(n));
        fp(i, 0, a.size() - 1) c[i].x = a[i];
        fp(i, 0, b.size() - 1) c[i].y = b[i];
        DFT(c), dot(c, c), IDFT(c), a.resize(n);
        fp(i, 0, n - 1) a[i] = int(c[i].y * .5 + .5);
        return a;
    }
}
/*---------------------------------------------------------------------------*/
using namespace Polynomial;
```

FWT

```cpp
#include <bits/stdc++.h>
using namespace std;
const int mod = 998244353;
void add(int &x, int y) {
  if ((x += y) >= mod) x -= mod;
```

```cpp
}
void del(int &x, int y) {
  if ((x -= y) < 0) x += mod;
}
void fwtor(int a[], int m, int opt)  //(1,-1)
{
  for (int len = 2; len <= m; len <<= 1)
    for (int p = len >> 1, i = 0; i < m; i += len)
      for (int j = i; j < i + p; j++)
        if (opt > 0)
          add(a[j + p], a[j]);
        else
          del(a[j + p], a[j]);
}
void fwtand(int a[], int m, int opt)  //(1,-1)
{
  for (int len = 2; len <= m; len <<= 1)
    for (int p = len >> 1, i = 0; i < m; i += len)
      for (int j = i; j < i + p; j++)
        if (opt > 0)
          add(a[j], a[j + p]);
        else
          del(a[j], a[j + p]);
}
void fwtxor(int a[], int m, int opt)  //(1,1/2)
{
  for (int len = 2; len <= m; len <<= 1)
    for (int p = len >> 1, i = 0; i < m; i += len)
      for (int j = i; j < i + p; j++) {
        add(a[j], a[j + p]);
        a[j + p] = (a[j] - 2ll * a[j + p] % mod + mod) % mod;
        a[j] = 1ll * a[j] * opt % mod;
        a[j + p] = 1ll * a[j + p] * opt % mod;
      }
}
int a[1 << 17], b[1 << 17], c[1 << 17];
void mul(int a[], int b[], int c[], int m) {
  for (int i = 0; i < m; i++) c[i] = 1ll * a[i] * b[i] % mod;
}
void print(int a[], int m) {
  for (int i = 0; i < m; i++) cout << a[i] << " \n"[i == m - 1];
}
int main() {
  int n;
  cin >> n;
  int m = 1 << n;
  for (int i = 0; i < m; i++) cin >> a[i];
  for (int i = 0; i < m; i++) cin >> b[i];

  fwtor(a, m, 1), fwtor(b, m, 1), mul(a, b, c, m);
  fwtor(a, m, -1), fwtor(b, m, -1), fwtor(c, m, -1), print(c, m);

  fwtand(a, m, 1), fwtand(b, m, 1), mul(a, b, c, m);
  fwtand(a, m, -1), fwtand(b, m, -1), fwtand(c, m, -1), print(c, m);

  fwtxor(a, m, 1), fwtxor(b, m, 1), mul(a, b, c, m);
  fwtxor(c, m, (mod + 1) / 2), print(c, m);
}
```

莫比乌斯反演

```
#include <bits/stdc++.h>
using namespace std;
const int maxn=1e5+10;
int pr[maxn],tot,mul[maxn],phi[maxn];
bool vis[maxn];
void init(int n)
{
    mul[1]=phi[1]=1;
    for(int i=2;i<=n;i++)
     {
         if(!vis[i])
         {
             mul[i]=-1;
             pr[++tot]=i;
             phi[i]=i-1;
         }
         for(int j=1;j<=tot && (long long)pr[j]*i<=n;j++)
         {
             int num=pr[j]*i;
             vis[num]=1;
             mul[num]=-mul[i];
             phi[num]=phi[i]*phi[pr[j]];
             if(i%pr[j]==0)
             {
                 phi[num]=pr[j]*phi[i];
                 mul[num]=0;
                 break;
             }
         }
     }
}
```

NTT

```
#include <bits/stdc++.h>

#define fp(i, a, b) for (int i = (a), i##_ = (b) + 1; i < i##_; ++i)
#define fd(i, a, b) for (int i = (a), i##_ = (b) - 1; i > i##_; --i)
#define file(s) freopen(s".in","r",stdin),freopen(s".out","w",stdout)
using namespace std;
const int maxn = 2e5 + 5, P = 998244353;
using arr = int[maxn];
using ll = int64_t;
 /*---------------------------------------------------------------------------*/
 class Cipolla {
     int P, I2{};
     using pll = pair<ll, ll>;
 #define X first
 #define Y second
     ll mul(ll a, ll b) const { return a * b % P; }
      pll mul(pll a, pll b) const { return {(a.X * b.X + I2 * a.Y % P * b.Y) % P,
(a.X * b.Y + a.Y * b.X) % P}; }
     template<class T> T POW(T a, int b, T x) { for (; b; b >>= 1, a = mul(a, a)) if
```

```cpp
(b & 1) x = mul(x, a); return x; }
public:
    Cipolla(int p = 0) : P(p) {}
    pair<int, int> sqrt(int n) {
        int a = rand(), x;
        if (!(n %= P))return {0, 0};
        if (POW(n, (P - 1) >> 1, (int)1) == P - 1) return {-1, -1};
        while (POW(I2 = ((ll) a * a - n + P) % P, (P - 1) >> 1, (int)1) == 1) a = rand();
        x = (int) POW(pll{a, 1}, (P + 1) >> 1, {1, 0}).X;
        if (2 * x > P) x = P - x;
        return {x, P - x};
    }
#undef X
#undef Y
};
/*--------------------------------------------------------------------------*/
#define ADD(a, b) (((a) += (b)) >= P ? (a) -=P : 0) // (a += b) %= P
#define SUB(a, b) (((a) -= (b)) < 0 ? (a) += P: 0)  // ((a -= b) += P) %= P
#define MUL(a, b) ((ll) (a) * (b) % P)
//vector<int> getInv(int L) {
//    vector<int> inv(L); inv[1] = 1;
//    fp(i, 1, L - 1) inv[i] = MUL((P - P / i), inv[P % i]);
//    return inv;
//}
//auto inv = getInv(maxn); // NOLINT
int POW(ll a, int b = P - 2, ll x = 1) { for (; b; b >>= 1, a = a * a % P) if (b & 1) x = x * a % P; return x; }
//int INV(int a) { return a < maxn ? inv[a] : POW(a); }
namespace NTT {
    const int g = 3;
    vector<int> Omega(int L) {
        int wn = POW(g, P / L);
        vector<int> w(L); w[L >> 1] = 1;
        fp(i, L / 2 + 1, L - 1) w[i] = MUL(w[i - 1], wn);
        fd(i, L / 2 - 1, 1) w[i] = w[i << 1];
        return w;
    }
    auto W = Omega(1 << 21); // NOLINT
    void DIF(int *a, int n) {
        for (int k = n >> 1; k; k >>= 1)
            for (int i = 0, y; i < n; i += k << 1)
                fp(j, 0, k - 1)
                     y = a[i + j + k], a[i + j + k] = MUL(a[i + j] - y + P, W[k + j]), ADD(a[i + j], y);
    }
    void IDIT(int *a, int n) {
        for (int k = 1; k < n; k <<= 1)
            for (int i = 0, x, y; i < n; i += k << 1)
                fp(j, 0, k - 1)
                    x = a[i + j], y = MUL(a[i + j + k], W[k + j]),
                    a[i + j + k] = x - y < 0 ? x - y + P : x - y, ADD(a[i + j], y);
        int Inv = P - (P - 1) / n;
        fp(i, 0, n - 1) a[i] = MUL(a[i], Inv);
        reverse(a + 1, a + n);
    }
```

```
}
namespace Polynomial {
    using Poly = std::vector<int>;

    // mul/div int
    Poly &operator*=(Poly &a, int b) { for (auto &x : a) x = MUL(x, b); return a; }
    Poly operator*(Poly a, int b) { return a *= b; }
    Poly operator*(int a, Poly b) { return b * a; }
    Poly &operator/=(Poly &a, int b) { return a *= POW(b); }
    Poly operator/(Poly a, int b) { return a /= b; }

    // Poly add/sub
    Poly &operator+=(Poly &a, Poly b) {
        a.resize(max(a.size(), b.size()));
        fp(i, 0, b.size() - 1) ADD(a[i], b[i]);
        return a;
    }
    Poly operator+(Poly a, Poly b) { return a += b; }
    Poly &operator-=(Poly &a, Poly b) {
        a.resize(max(a.size(), b.size()));
        fp(i, 0, b.size() - 1) SUB(a[i], b[i]);
        return a;
    }
    Poly operator-(Poly a, Poly b) { return a -= b; }

    // Poly mul
    void DFT(Poly &a) { NTT::DIF(a.data(), a.size()); }
    void IDFT(Poly &a) { NTT::IDIT(a.data(), a.size()); }
    int norm(int n) { return 1 << (32 - __builtin_clz(n - 1)); }
     void norm(Poly &a) { if (!a.empty()) a.resize(norm(a.size()), 0); }
     Poly &dot(Poly &a, Poly &b) {
         fp(i, 0, a.size() - 1) a[i] = MUL(a[i], b[i]);
         return a;
     }
     Poly operator*(Poly a, Poly b) {
         int n = a.size() + b.size() - 1, L = norm(n);
         if (a.size() <= 8 || b.size() <= 8) {
             Poly c(n);
             fp(i, 0, a.size() - 1) fp(j, 0, b.size() - 1)
                 c[i + j] = (c[i + j] + (ll) a[i] * b[j]) % P;
             return c;
         }
         a.resize(L), b.resize(L);
         DFT(a), DFT(b), dot(a, b), IDFT(a);
         return a.resize(n), a;
     }

     // Poly inv
     Poly Inv2k(Poly a) { // a.size() = 2^k
         int n = a.size(), m = n >> 1;
         if (n == 1) return {POW(a[0])};
         Poly b = Inv2k(Poly(a.begin(), a.begin() + m)), c = b;
         b.resize(n), DFT(a), DFT(b), dot(a, b), IDFT(a);
         fp(i, 0, n - 1) a[i] = i < m ? 0 : P - a[i];
         DFT(a), dot(a, b), IDFT(a);
         return move(c.begin(), c.end(), a.begin()), a;
```

```cpp
    }
    Poly Inv(Poly a) {
        int n = a.size();
        norm(a), a = Inv2k(a);
        return a.resize(n), a;
    }

    // Poly div/mod
    Poly operator/(Poly a,Poly b){
        int k = a.size() - b.size() + 1;
        if (k < 0) return {0};
        reverse(a.begin(), a.end());
        reverse(b.begin(), b.end());
        b.resize(k), a = a * Inv(b);
        a.resize(k), reverse(a.begin(), a.end());
        return a;
    }
    pair<Poly, Poly> operator%(Poly a, const Poly& b) {
        Poly c = a / b;
        a -= b * c, a.resize(b.size() - 1);
        return {c, a};
    }

    // Poly sqrt
    Poly Sqrt(Poly a) {
        int n = a.size(), k = norm(n);
        Poly b = {(new Cipolla(P))->sqrt(a[0]).first}, c;
        a.resize(k * 2, 0);
        for (int L = 2; L <= k; L <<= 1) {
            b.resize(2 * L, 0), c = Poly(a.begin(), a.begin() + L) * Inv(b);
            fp(i, 0, 2 * L - 1) b[i] = MUL(b[i] + c[i], (P + 1) / 2);
        }
        return b.resize(n), b;
    }

    // Poly calculus
    void Derivative(Poly &a) {
        fp(i, 1, a.size() - 1) a[i - 1] = MUL(i, a[i]);
        a.pop_back();
    }
}
```

任意模数 NTT

```cpp
const long long mod =1e18;
namespace polynomial {
    typedef complex<long double> cplx;
    const long double pi = acos((long double)-1.0);
    const int len = 15, mask = (1 << len) - 1;
    struct UnitRoot {
        static vector<cplx> w;
        static vector<cplx> get_root(int n) {
            n = 1 << 32 - __builtin_clz(n);
             if (n > w.size()) {
                w.resize(n);
                for (int i = 0; i < n; i++)
                    w[i] = cplx(cos(2 * i * pi / n), sin(2 * i * pi / n));
            }
            int m = w.size() / n;
            vector<cplx> res(n);
            for (int i = 0, j = 0; i < n; i++, j += m) res[i] = w[j];
            return res;
        }
    };
    vector<cplx> UnitRoot::w;

    void fft(vector<cplx> &p, const vector<cplx> &w) {
        int n = w.size();
        for (int i = 1, j = 0; i < n - 1; ++i) {
            int s = n;
            do {
                s >>= 1;
                j ^= s;
            } while (~j & s);
            if (i < j) {
                swap(p[i], p[j]);
            }
        }
        for (int d = 0; (1 << d) < n; ++d) {
            int m = 1 << d, m2 = m * 2, rm = n >> (d + 1);
            for (int i = 0; i < n; i += m2) {
                for (int j = 0; j < m; ++j) {
                    auto &p1 = p[i + j + m], &p2 = p[i + j];
                    auto t = w[rm * j] * p1;
                    p1 = p2 - t;
                    p2 = p2 + t;
                }
            }
        }
    }
    vector<long long> conv(const vector<long long> &a,const vector<long long> &b) {
        vector<cplx> w = UnitRoot::get_root(a.size() + b.size() - 1);
        int n = w.size();
        vector<cplx> A(n), B(n), C(n), D(n);
        for (int i = 0; i < a.size(); ++i) A[i] = cplx(a[i] >> len, a[i] & mask);
        for (int i = 0; i < b.size(); ++i) B[i] = cplx(b[i] >> len, b[i] & mask);
        fft(A, w), fft(B, w);
        for (int i = 0; i < n; ++i) {
            int j = (n - i) % n;
            cplx da = (A[i] - conj(A[j])) * cplx(0, -0.5),db = (A[i] + conj(A[j])) * cplx(0.5, 0),dc = (B[i] - conj(B[j])) * cplx(0, -0.5),dd = (B[i] + conj(B[j])) * cplx(0.5, 0);
            C[j] = da * dd + da * dc * cplx(0, 1);D[j] = db * dd + db * dc * cplx(0, 1);
        }
        fft(C, w), fft(D, w);
        vector<long long> res(a.size() + b.size() - 1);
        for (int i = 0; i < res.size(); ++i) {
            long long da = (long long)(C[i].imag() / n + 0.5) % mod,db = (long long)(C[i].real() / n + 0.5) % mod,dc = (long long)(D[i].imag() / n + 0.5) % mod,dd = (long long)(D[i].real() / n + 0.5) % mod;
            res[i] = ((dd << (len * 2)) + ((db + dc) << len) + da) % mod;
        }
        return res;
    }
};
using namespace polynomial;
```

Pollard_Rho

```
typedef long long ll;
map<ll, bool>P;
mt19937_64 rnd(time(0));
namespace Pollard_Rho
{
#define ldb long double
ll mul(ll x, ll y, ll mod)
{
  return ((x * y - (ll)((ldb)x / mod * y) * mod) + mod) % mod;
}
ll gcd(ll a, ll b)
{
  return (b == 0 ? a : gcd(b, a % b));
}
ll ksm(ll a, ll b, ll mod)
{
  ll ans = 1; a %= mod;
  while (b) {if (b & 1)ans = mul(ans, a, mod); b >>= 1; a = mul(a, a, mod);}
 return ans;
}
int pr[15] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
bool Miller_Rabin(ll n)
{
 if (n == 2 || n == 3)return 1;
 if (n % 2 == 0 || n == 1)return 0;
 ll d = n - 1;
  int s = 0;
 while (d % 2 == 0)s ++, d >>= 1;
 for (int i = 0; i <= 11; i ++)
 {
    if (pr[i] >= n)break;
    ll a = pr[i];
    ll x = ksm(a, d, n);
   ll y = 0;
    for (int j = 0; j <= s - 1; j ++)
    {
      y = mul(x, x, n);
      if (y == 1 && x != 1 && x != (n - 1))return 0;
     x = y;
   }
    if (y != 1)return 0;
 }
  return 1;
}
ll Pollard_Rho(ll n)
{
 ll now, pre, g;
  while (true)
 {
    now = pre = rnd() % (n - 1) + 1;
   g = 1;
   ll c = rnd() % (n - 1) + 1;
    for (int i = 1, fst = 1 ;; i ++)
   {
      now = (mul(now, now, n) + c) % n;
      g = mul(g, abs(now - pre), n);
     if (now == pre || !g)break;
      if (!(i & 127) || i == fst)
      {
        g = gcd(g, n);
       if (g > 1)return g;
        if (i == fst)pre = now, fst <<= 1;
     }
    }
  }
}
void Find(ll n)
{
  if (n == 1)return ;
  if (Miller_Rabin(n))
 {
    P[n] = 1;
    return ;
 }
  ll p = Pollard_Rho(n);
 int c = 0;
 while (!(n % p))
 {
    n /= p, c ++;
  }
  Find(p);
 Find(n);
}
}
void solve(int x,set<int> &s){
    Pollard_Rho :: Find(x);
    for (auto [x, _] : P)s.insert(x);
    P.clear();
}
```

扩展中国剩余定理

```
#define int long long
int mul(int a,int b,int mod){//O(1)取模快速乘，不会爆 long long
  return (a*b-(int)((long double)a/mod*b)*mod+mod)%mod;
}
int exgcd(int a, int b, int& x, int& y){
 if(!b){
    x = 1, y = 0;
    return a;
  }
   int d = exgcd(b,a%b,y,x);
   y -= a/b*x;
   return d;
 }
 int solve(int n,vector<int>&mo,vector<int>&res){
```

```
 int a1,m1;
    a1=res[0],m1=mo[0];
 bool ok = 1;
 for(int i=1;i<n;i++){
    int a2,m2,k1,k2;
        m2=mo[i],a2=res[i];
   int d = exgcd(m1,m2,k1,k2);
    if((a2-a1)%d) ok = 0;
    else{
      k1=mul(k1,(a2-a1)/d,m2/d);//这个地方必须要用取模快速乘
      a1=a1+k1*m1;
     m1=abs(m1/d*m2);
   }
  }
  if(ok)return (a1%m1+m1)%m1;
    else return -1;
}
```

拉格朗日插值

```
#define int long long
const int N=1e6+10,mod=998244353;
int  ksm(int  a,int  n,int  m=mod){int  s=1;while(n){if(n&1)  s=s*a%m;a=a*a%m;n>>=1;}
return s;}
int fac[N+5],facinv[N+5],inv[N+5];
struct LR{
  int Inv(int n){return ksm(n,mod-2);}
 void init(){  //预处理阶乘和阶乘逆元,逆元.
   fac[0]=inv[0]=inv[1]=1;
        for(int i=1;i<=N;i++)
             fac[i]=fac[i-1]*i%mod;
     facinv[N]=Inv(fac[N]);
    for(int i=N-1;~i;i--)
             facinv[i]=facinv[i+1]*(i+1)%mod;
     for(int i=2;i<N+5;i++)
      inv[i]=(mod-mod/i)*inv[mod%i]%mod;
  }
   int cal(vector<int>&x,vector<int>&y,int k){ //离散点 n 个点[0,n-1] x[i],y[i] 插 f(k)
         int n=x.size();
     int s=0;
         for(int i=0;i<n;i++)if(x[i]==k)return y[i];
    for(int i=0;i<n;i++){
       int p=y[i]%mod,q=1;
       for(int j=0;j<n;j++){
         if(i==j) continue;
        p=p*((k-x[j])%mod+mod)%mod;
         q=q*((x[i]-x[j])%mod+mod)%mod;
      }
       s=(s+p*Inv(q)%mod)%mod;
     }return (s%mod+mod)%mod;
  }
   int inpo(vector<int>&f,int x){  //给定 连续 i 属于[0,n] f(i) 拉插 f(x)
         int n=f.size()-1;
         if(x>=0&&x<=n)return f[x];
    int p,s=0;
         vector<int>pre(n+1),suf(n+1);
         pre[0]=x-0;
```

```cpp
         for(int i=1;i<=n;i++)pre[i]=pre[i-1]*(x-i)%mod;
         suf[n]=x-n;
         for(int i=n-1;i>=0;i--)suf[i]=suf[i+1]*(x-i)%mod;
    for(int i=0;i<=n;i++){
      p=facinv[n-i]%mod*facinv[i]%mod;
             if(i>0)p=p*pre[i-1]%mod;
             if(i<n)p=p*suf[i+1]%mod;
      if((n-i)&1)  s=(s-p*f[i]%mod+mod)%mod;
      else s=(s+p*f[i]%mod)%mod;
    }
     return (s%mod+mod)%mod;
   }
 }sol;
```

拉格朗日插值没有模数

```cpp
struct LR{
 int inter(std::vector <int> vec, int x){
    int n = vec.size() - 1;
    int ans = 0;
    for (int i = 0; i <= n; ++ i){
        int div = 1;
        for (int j = 0; j <= n; ++ j){
            if (i != j) div *= (i - j);
        }
         bool flag = div < 0;
         div = std::abs(div);
         int prod = vec[i];
         for (int j = 0; j <= n; ++ j){
             if (i == j) continue;
             int gcd = std::abs(std::__gcd(x - j, div));
             prod *= (x - j) / gcd;
             div /= gcd;
         }
         ans += flag ? -prod : prod;
     }
     return ans;
 }
```

杜教筛

```cpp
const int maxn=3e6+10;
int sumf[maxn];
int Sum(int n){// 这是 f * g 的 n 项前缀和

}
int Sumg(int n){// g 的 n 项前缀和

}
map<int,int> f;
 int F (int n) {
  if (n <= 3000'000) return sumf[n]; // 预处理出 n 较小时的前缀和
  if (f.find(n)!=f.end()) return f[n]; // 记忆化，如果求过这个值，就不需要再递归一遍了
  int ans = Sum(n);
  for (int l = 2, r; l <= n; l = r + 1) // 整除分块
     r = n / (n / l), ans -= (Sumg(r) - Sumg(l-1)) * F (n / l);
     // [l,r] 的 F (n / l) 是一样的，对 g(x) 求个和即可
  return f[n] = ans / Sumg(1); // 别忘了除上 g(1)
}

```

线性筛质数

```cpp
#include <bits/stdc++.h>
using namespace std;
const int maxn=1e6+10;
bool v[maxn];
int n,pr;
vector<int> p;
void init()
{
    v[1]=true;
    for(int i=2;i<maxn;i++)
    {
     if(!v[i])p.push_back(i);
     for(int j=0;j<p.size()&&i*p[j]<maxn;++j){v[i*p[j]]=true;if(i%p[j]==0)break;}
    }
}

```

线性递推

```cpp
#include<bits/stdc++.h>
using namespace std;
#define rep(i,a,n) for (int i=a;i<n;i++)
#define per(i,a,n) for (int i=n-1;i>=a;i--)
#define all(x) (x).begin(),(x).end()
#define siz(x) ((int)(x).size())
typedef vector<int> VI;
typedef long long ll;
typedef pair<int,int> PII;
const ll mod=1000000007;
ll powmod(ll a,ll b) {ll res=1;a%=mod; assert(b>=0); for(;b;b>>=1){if(b&1)res=res*a%mod;a=a*a%mod;}return res;}
ll n;
namespace linear_seq {
    const int N=10010;
    ll res[N],base[N],_c[N],_md[N];

    vector<int> Md;
    void mul(ll *a,ll *b,int k) {
        rep(i,0,k+k) _c[i]=0;
        rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
        for (int i=k+k-1;i>=k;i--) if (_c[i])
            rep(j,0,siz(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
        rep(i,0,k) a[i]=_c[i];
    }
    int solve(ll n,VI a,VI b) {
        ll ans=0,pnt=0;
        int k=siz(a);
        assert(siz(a)==siz(b));
        rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
        Md.clear();
        rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
        rep(i,0,k) res[i]=base[i]=0;
        res[0]=1;
        while ((1ll<<pnt)<=n) pnt++;
        for (int p=pnt;p>=0;p--) {
            mul(res,res,k);
            if ((n>>p)&1) {
                for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
                rep(j,0,siz(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
            }
        }
        rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
        if (ans<0) ans+=mod;
        return ans;
    }
    VI BM(VI s) {
        VI C(1,1),B(1,1);
        int L=0,m=1,b=1;
        rep(n,0,siz(s)) {
            ll d=0;
            rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
            if (d==0) ++m;
            else if (2*L<=n) {
                VI T=C;
                ll c=mod-d*powmod(b,mod-2)%mod;
                while (siz(C)<siz(B)+m) C.push_back(0);
                rep(i,0,siz(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                L=n+1-L; B=T; b=d; m=1;
            } else {
                ll c=mod-d*powmod(b,mod-2)%mod;
                while (siz(C)<siz(B)+m) C.push_back(0);
                rep(i,0,siz(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
                ++m;
            }
        }
        return C;
    }
    int gao(VI a,ll n) {
        VI c=BM(a);
        c.erase(c.begin());
        rep(i,0,siz(c)) c[i]=(mod-c[i])%mod;
        return solve(n,c,VI(a.begin(),a.begin()+siz(c)));
    }
};

int main() {
    vector<int>v;
    v.push_back(2);
    v.push_back(24);
    v.push_back(96);
    v.push_back(416);
    v.push_back(1536);
    v.push_back(5504);
    v.push_back(18944);
    v.push_back(64000);
    v.push_back(212992);
    v.push_back(702464);
  scanf("%lld", &n);
    printf("%lld\n",1LL * linear_seq::gao(v,n-1) % mod);
```

```
}

```

辛普森积分

```
double simpson(double l, double r) {
  double mid = (l + r) / 2;
  return (r - l) * (f(l) + 4 * f(mid) + f(r)) / 6;  // 辛普森公式
}

double asr(double l, double r, double eps, double ans, int step) {//step 是递归的下限
  double mid = (l + r) / 2;
  double fl = simpson(l, mid), fr = simpson(mid, r);
  if (abs(fl + fr - ans) <= 15 * eps && step < 0)
     return fl + fr + (fl + fr - ans) / 15;  // 足够相似的话就直接返回
   return asr(l, mid, eps / 2, fl, step - 1) +
         asr(mid, r, eps / 2, fr, step - 1);  // 否则分割成两段递归求解
 }

 double calc(double l, double r, double eps) {
   return asr(l, r, eps, simpson(l, r), 12);
 }

```

高斯消元(模意义)

```
#define int long long
const int eps=0;
const int maxn=220;
const int mod=1e6+3;
int ksm(int x,int k){
    int res=1;
    while(k){
        if(k&1)res=res*x%mod;
        x=x*x%mod;
         k/=2;
     }
     return res;
 }
 int ny(int x){
     return ksm(x,mod-2);
 }
 void add(int &x,int y){
     if((x+=y)>=mod)x-=mod;
 }
 void del(int &x,int y){
     if((x-=y)<0)x+=mod;
 }
 int a[maxn][maxn],x[maxn];//方程左边的矩阵和方程右边的值，求解之后 x 存的就是结果
 int Gauss(int equ,int var){//equ 方程数 var 未知数个数  return 1 表示有解
     int i,j,k,col,max_r;
     for(k=0,col=0;k<equ&&col<var;k++,col++){
         max_r=k;
         for(i=k+1;i<equ;i++)
             if((a[i][col])>(a[max_r][col]))
                 max_r=i;
         if((a[max_r][col])==0)return 0;
         if(k!=max_r){
             for(j=col;j<var;j++)
                 swap(a[k][j],a[max_r][j]);
             swap(x[k],x[max_r]);
         }
         x[k]=x[k]*ny(a[k][col])%mod;
         for(j=col+1;j<var;j++)a[k][j]=a[k][j]*ny(a[k][col])%mod;
         a[k][col]=1;
         for(i=0;i<equ;i++)
             if(i!=k){
                 del(x[i],x[k]*a[i][col]%mod);
                 for(j=col+1;j<var;j++)del(a[i][j],a[k][j]*a[i][col]%mod);
                 a[i][col]=0;
             }
     }
     return 1;
 }
```

高斯消元(浮点数)

```
#define lf double
//0 base
const lf eps=1e-9;
const int maxn=220;
lf a[maxn][maxn],x[maxn];//方程左边的矩阵和方程右边的值，求解之后 x 存的就是结果
int Gauss(int equ,int var){//equ 方程数 var 未知数个数  return 1 表示有解
    int i,j,k,col,max_r;
    for(k=0,col=0;k<equ&&col<var;k++,col++){
        max_r=k;
        for(i=k+1;i<equ;i++)
            if(fabs(a[i][col])>fabs(a[max_r][col]))
                max_r=i;
        if(fabs(a[max_r][col])<eps)return 0;
        if(k!=max_r){
            for(j=col;j<var;j++)
                swap(a[k][j],a[max_r][j]);
            swap(x[k],x[max_r]);
        }
        x[k]/=a[k][col];
        for(j=col+1;j<var;j++)a[k][j]/=a[k][col];
        a[k][col]=1;
        for(i=0;i<equ;i++)
            if(i!=k){
                x[i]-=x[k]*a[i][col];
                for(j=col+1;j<var;j++)a[i][j]-=a[k][j]*a[i][col];
                a[i][col]=0;
            }
    }
    return 1;
}
```

计算几何

开头

```cpp
#include <bits/stdc++.h>
using namespace std;

using point_t=long double;  //全局数据类型，可修改为 long long 等

constexpr point_t eps=1e-8;
constexpr long double PI=3.1415926535897932384l;

```

点与向量

```cpp
// 点与向量
template<typename T> struct point
{
    T x,y;

    bool operator==(const point &a) const {return (abs(x-a.x)<=eps && abs(y-a.y)<=eps);}
    bool operator<(const point &a) const {if (abs(x-a.x)<=eps) return y<a.y-eps; return x<a.x-eps;}
    bool operator>(const point &a) const {return !(*this<a || *this==a);}
    point operator+(const point &a) const {return {x+a.x,y+a.y};}
    point operator-(const point &a) const {return {x-a.x,y-a.y};}
    point operator-() const {return {-x,-y};}
    point operator*(const T k) const {return {k*x,k*y};}
    point operator/(const T k) const {return {x/k,y/k};}
    T operator*(const point &a) const {return x*a.x+y*a.y;}  // 点积
    T operator^(const point &a) const {return x*a.y-y*a.x;}  // 叉积，注意优先级
    int toleft(const point &a) const {const auto t=(*this)^a; return (t>eps)-(t<-eps);}  // to-left 测试
    T len2() const {return (*this)*(*this);}  // 向量长度的平方
    T dis2(const point &a) const {return (a-(*this)).len2();}  // 两点距离的平方

    // 涉及浮点数
    long double len() const {return sqrtl(len2());}  // 向量长度
    long double dis(const point &a) const {return sqrtl(dis2(a));}  // 两点距离
    long double ang(const point &a) const {return acosl(max(-1.0l,min(1.0l,((*this)*a)/(len()*a.len()))));}  // 向量夹角
    point rot(const long double rad) const {return {x*cos(rad)-y*sin(rad),x*sin(rad)+y*cos(rad)};}  // 逆时针旋转（给定角度）
    point rot(const long double cosr,const long double sinr) const {return {x*cosr-y*sinr,x*sinr+y*cosr};}  // 逆时针旋转（给定角度的正弦与余弦）
};

using Point=point<point_t>;
```

极角排序

```cpp
// 极角排序
struct argcmp
{
    bool operator()(const Point &a,const Point &b) const
    {
        const auto quad=[](const Point &a)
```

```
       {
           if (a.y<-eps) return 1;
           if (a.y>eps) return 4;
            if (a.x<-eps) return 5;
            if (a.x>eps) return 3;
            return 2;
        };
        const int qa=quad(a),qb=quad(b);
        if (qa!=qb) return qa<qb;
        const auto t=a^b;
        // if (abs(t)<=eps) return a*a<b*b-eps;  // 不同长度的向量需要分开
        return t>eps;
    }
};
```

直线

```
// 直线
template<typename T> struct line
{
    point<T> p,v;  // p 为直线上一点，v 为方向向量

     bool operator==(const line &a) const {return v.toleft(a.v)==0 && v.toleft(p-a.p)==0;}
    int toleft(const point<T> &a) const {return v.toleft(a-p);}  // to-left 测试
    bool operator<(const line &a) const  // 半平面交算法定义的排序
    {
         if (abs(v^a.v)<=eps && v*a.v>=-eps) return toleft(a.p)==-1;
         return argcmp()(v,a.v);
     }

     // 涉及浮点数
     point<T> inter(const line &a) const {return p+v*((a.v^(p-a.p))/(v^a.v));}  // 直线交点
     long double dis(const point<T> &a) const {return abs(v^(a-p))/v.len();}  // 点到直线距离
     point<T> proj(const point<T> &a) const {return p+v*((v*(a-p))/(v*v));}  // 点在直线上的投影
 };

 using Line=line<point_t>;
```

线段

```
//线段
template<typename T> struct segment
{
    point<T> a,b;

    bool operator<(const segment &s) const {return make_pair(a,b)<make_pair(s.a,s.b);}

    // 判定性函数建议在整数域使用

     // 判断点是否在线段上
     // -1 点在线段端点 | 0 点不在线段上 | 1 点严格在线段上
     int is_on(const point<T> &p) const
     {
         if (p==a || p==b) return -1;
```

```
        return (p-a).toleft(p-b)==0 && (p-a)*(p-b)<-eps;
    }

    // 判断线段直线是否相交
    // -1 直线经过线段端点 | 0 线段和直线不相交 | 1 线段和直线严格相交
    int is_inter(const line<T> &l) const
    {
        if (l.toleft(a)==0 || l.toleft(b)==0) return -1;
        return l.toleft(a)!=l.toleft(b);
    }

    // 判断两线段是否相交
    // -1 在某一线段端点处相交 | 0 两线段不相交 | 1 两线段严格相交
    int is_inter(const segment<T> &s) const
    {
        if (is_on(s.a) || is_on(s.b) || s.is_on(a) || s.is_on(b)) return -1;
        const line<T> l{a,b-a},ls{s.a,s.b-s.a};
        return l.toleft(s.a)*l.toleft(s.b)==-1 && ls.toleft(a)*ls.toleft(b)==-1;
    }

    // 点到线段距离
    long double dis(const point<T> &p) const
    {
        if ((p-a)*(b-a)<-eps || (p-b)*(a-b)<-eps) return min(p.dis(a),p.dis(b));
        const line<T> l{a,b-a};
        return l.dis(p);
    }

    // 两线段间距离
    long double dis(const segment<T> &s) const
    {
        if (is_inter(s)) return 0;
        return min({dis(s.a),dis(s.b),s.dis(a),s.dis(b)});
    }
};

using Segment=segment<point_t>;
```

多边形

```
// 多边形
template<typename T> struct polygon
{
    vector<point<T>> p;  // 以逆时针顺序存储

    size_t nxt(const size_t i) const {return i==p.size()-1?0:i+1;}
    size_t pre(const size_t i) const {return i==0?p.size()-1:i-1;}

    // 回转数
     // 返回值第一项表示点是否在多边形边上
     // 对于狭义多边形，回转数为 0 表示点在多边形外，否则点在多边形内
     pair<bool,int> winding(const point<T> &a) const
     {
         int cnt=0;
         for (size_t i=0;i<p.size();i++)
         {
             const point<T> u=p[i],v=p[nxt(i)];
```

```cpp
            if (abs((a-u)^(a-v))<=eps && (a-u)*(a-v)<=eps) return {true,0};
            if (abs(u.y-v.y)<=eps) continue;
            const Line uv={u,v-u};
            if (u.y<v.y-eps && uv.toleft(a)<=0) continue;
            if (u.y>v.y+eps && uv.toleft(a)>=0) continue;
            if (u.y<a.y-eps && v.y>=a.y-eps) cnt++;
            if (u.y>=a.y-eps && v.y<a.y-eps) cnt--;
        }
        return {false,cnt};
    }

    // 多边形面积的两倍
    // 可用于判断点的存储顺序是顺时针或逆时针
    T area() const
    {
        T sum=0;
        for (size_t i=0;i<p.size();i++) sum+=p[i]^p[nxt(i)];
        return sum;
    }

    // 多边形的周长
    long double circ() const
    {
        long double sum=0;
        for (size_t i=0;i<p.size();i++) sum+=p[i].dis(p[nxt(i)]);
        return sum;
    }
};

using Polygon=polygon<point_t>;
```

凸多边形

```cpp
//凸多边形
template<typename T> struct convex: polygon<T>
{
    // 闵可夫斯基和
    convex operator+(const convex &c) const
    {
        const auto &p=this->p;
        vector<Segment> e1(p.size()),e2(c.p.size()),edge(p.size()+c.p.size());
        vector<point<T>> res; res.reserve(p.size()+c.p.size());
         const auto cmp=[](const Segment &u,const Segment &v) {return argcmp()(u.b-
u.a,v.b-v.a);};
        for (size_t i=0;i<p.size();i++) e1[i]={p[i],p[this->nxt(i)]};
        for (size_t i=0;i<c.p.size();i++) e2[i]={c.p[i],c.p[c.nxt(i)]};
        rotate(e1.begin(),min_element(e1.begin(),e1.end(),cmp),e1.end());
        rotate(e2.begin(),min_element(e2.begin(),e2.end(),cmp),e2.end());
        merge(e1.begin(),e1.end(),e2.begin(),e2.end(),edge.begin(),cmp);
        const auto check=[](const vector<point<T>> &res,const point<T> &u)
        {
            const auto back1=res.back(),back2=*prev(res.end(),2);
             return (back1-back2).toleft(u-back1)==0 && (back1-back2)*(u-back1)>=-
eps;
        };
        auto u=e1[0].a+e2[0].a;
        for (const auto &v:edge)
        {
            while (res.size()>1 && check(res,u)) res.pop_back();
            res.push_back(u);
            u=u+v.b-v.a;
        }
        if (res.size()>1 && check(res,res[0])) res.pop_back();
        return {res};
    }

    // 旋转卡壳
    // func 为更新答案的函数，可以根据题目调整位置
    template<typename F> void rotcaliper(const F &func) const
    {
        const auto &p=this->p;
         const auto area=[](const point<T> &u,const point<T> &v,const point<T> &w)
{return (w-u)^(w-v);};
        for (size_t i=0,j=1;i<p.size();i++)
        {
            const auto nxti=this->nxt(i);
            func(p[i],p[nxti],p[j]);
            while (area(p[this->nxt(j)],p[i],p[nxti])>=area(p[j],p[i],p[nxti]))
            {
                j=this->nxt(j);
                func(p[i],p[nxti],p[j]);
            }
        }
    }

    // 凸多边形的直径的平方
    T diameter2() const
    {
        const auto &p=this->p;
        if (p.size()==1) return 0;
        if (p.size()==2) return p[0].dis2(p[1]);
        T ans=0;
            auto func=[&](const point<T> &u,const point<T> &v,const point<T> &w)
{ans=max({ans,w.dis2(u),w.dis2(v)});};
        rotcaliper(func);
        return ans;
    }

    // 判断点是否在凸多边形内
    // 复杂度 O(logn)
    // -1 点在多边形边上 | 0 点在多边形外 | 1 点在多边形内
    int is_in(const point<T> &a) const
    {
        const auto &p=this->p;
        if (p.size()==1) return a==p[0]?-1:0;
        if (p.size()==2) return segment<T>{p[0],p[1]}.is_on(a)?-1:0;
        if (a==p[0]) return -1;
          if ((p[1]-p[0]).toleft(a-p[0])==-1 || (p.back()-p[0]).toleft(a-p[0])==1)
return 0;
       const auto cmp=[&](const Point &u,const Point &v){return (u-p[0]).toleft(v-
p[0])==1;};
        const size_t i=lower_bound(p.begin()+1,p.end(),a,cmp)-p.begin();
        if (i==1) return segment<T>{p[0],p[i]}.is_on(a)?-1:0;
```

```cpp
        if (i==p.size()-1 && segment<T>{p[0],p[i]}.is_on(a)) return -1;
        if (segment<T>{p[i-1],p[i]}.is_on(a)) return -1;
        return (p[i]-p[i-1]).toleft(a-p[i-1])>0;
    }

    // 凸多边形关于某一方向的极点
    // 复杂度 O(logn)
    // 参考资料：https://codeforces.com/blog/entry/48868
    template<typename F> size_t extreme(const F &dir) const
    {
        const auto &p=this->p;
       const auto check=[&](const size_t i){return dir(p[i]).toleft(p[this->nxt(i)]-
p[i])>=0;};
        const auto dir0=dir(p[0]); const auto check0=check(0);
        if (!check0 && check(p.size()-1)) return 0;
        const auto cmp=[&](const Point &v)
        {
            const size_t vi=&v-p.data();
            if (vi==0) return 1;
            const auto checkv=check(vi);
            const auto t=dir0.toleft(v-p[0]);
            if (vi==1 && checkv==check0 && t==0) return 1;
            return checkv^(checkv==check0 && t<=0);
        };
        return partition_point(p.begin(),p.end(),cmp)-p.begin();
    }

     // 过凸多边形外一点求凸多边形的切线，返回切点下标
     // 复杂度 O(logn)
     // 必须保证点在多边形外
     pair<size_t,size_t> tangent(const point<T> &a) const
     {
         const size_t i=extreme([&](const point<T> &u){return u-a;});
         const size_t j=extreme([&](const point<T> &u){return a-u;});
         return {i,j};
     }

     // 求平行于给定直线的凸多边形的切线，返回切点下标
     // 复杂度 O(logn)
     pair<size_t,size_t> tangent(const line<T> &a) const
     {
         const size_t i=extreme([&](...){return a.v;});
         const size_t j=extreme([&](...){return -a.v;});
         return {i,j};
     }
};

using Convex=convex<point_t>;
```

圆

```cpp
// 圆
struct Circle
{
    Point c;
    long double r;

    bool operator==(const Circle &a) const {return c==a.c && abs(r-a.r)<=eps;}
    long double circ() const {return 2*PI*r;}  // 周长
    long double area() const {return PI*r*r;}  // 面积

    // 点与圆的关系
    // -1 圆上 | 0 圆外 | 1 圆内
    int is_in(const Point &p) const {const long double d=p.dis(c); return abs(d-r)<=eps?-1:d<r-eps;}

    // 直线与圆关系
    // 0 相离 | 1 相切 | 2 相交
    int relation(const Line &l) const
    {
        const long double d=l.dis(c);
        if (d>r+eps) return 0;
        if (abs(d-r)<=eps) return 1;
        return 2;
    }

    // 圆与圆关系
    // -1 相同 | 0 相离 | 1 外切 | 2 相交 | 3 内切 | 4 内含
    int relation(const Circle &a) const
    {
        if (*this==a) return -1;
        const long double d=c.dis(a.c);
        if (d>r+a.r+eps) return 0;
        if (abs(d-r-a.r)<=eps) return 1;
        if (abs(d-abs(r-a.r))<=eps) return 3;
        if (d<abs(r-a.r)-eps) return 4;
        return 2;
    }

    // 直线与圆的交点
    vector<Point> inter(const Line &l) const
    {
        const long double d=l.dis(c);
        const Point p=l.proj(c);
        const int t=relation(l);
        if (t==0) return vector<Point>();
        if (t==1) return vector<Point>{p};
        const long double k=sqrt(r*r-d*d);
        return vector<Point>{p-(l.v/l.v.len())*k,p+(l.v/l.v.len())*k};
    }

    // 圆与圆交点
    vector<Point> inter(const Circle &a) const
    {
        const long double d=c.dis(a.c);
        const int t=relation(a);
        if (t==-1 || t==0 || t==4) return vector<Point>();
        Point e=a.c-c; e=e/e.len()*r;
        if (t==1 || t==3)
        {
            if (r*r+d*d-a.r*a.r>=-eps) return vector<Point>{c+e};
            return vector<Point>{c-e};
        }
        const long double costh=(r*r+d*d-a.r*a.r)/(2*r*d),sinth=sqrt(1-costh*costh);
        return vector<Point>{c+e.rot(costh,-sinth),c+e.rot(costh,sinth)};
    }

    // 圆与圆交面积
    long double inter_area(const Circle &a) const
    {
        const long double d=c.dis(a.c);
        const int t=relation(a);
        if (t==-1) return area();
        if (t<2) return 0;
        if (t>2) return min(area(),a.area());
        const long double costh1=(r*r+d*d-a.r*a.r)/(2*r*d),costh2=(a.r*a.r+d*d-r*r)/(2*a.r*d);
        const long double sinth1=sqrt(1-costh1*costh1),sinth2=sqrt(1-costh2*costh2);
        const long double th1=acos(costh1),th2=acos(costh2);
        return r*r*(th1-costh1*sinth1)+a.r*a.r*(th2-costh2*sinth2);
    }

    // 过圆外一点圆的切线
    vector<Line> tangent(const Point &a) const
    {
        const int t=is_in(a);
        if (t==1) return vector<Line>();
        if (t==-1)
        {
            const Point v={-(a-c).y,(a-c).x};
            return vector<Line>{{a,v}};
        }
        Point e=a-c; e=e/e.len()*r;
        const long double costh=r/c.dis(a),sinth=sqrt(1-costh*costh);
        const Point t1=c+e.rot(costh,-sinth),t2=c+e.rot(costh,sinth);
        return vector<Line>{{a,t1-a},{a,t2-a}};
    }

    // 两圆的公切线
    vector<Line> tangent(const Circle &a) const
    {
        const int t=relation(a);
        vector<Line> lines;
        if (t==-1 || t==4) return lines;
        if (t==1 || t==3)
        {
            const Point p=inter(a)[0],v={-(a.c-c).y,(a.c-c).x};
            lines.push_back({p,v});
        }
        const long double d=c.dis(a.c);
        const Point e=(a.c-c)/(a.c-c).len();
        if (t<=2)
        {
            const long double costh=(r-a.r)/d,sinth=sqrt(1-costh*costh);
            const Point d1=e.rot(costh,-sinth),d2=e.rot(costh,sinth);
            const Point u1=c+d1*r,u2=c+d2*r,v1=a.c+d1*a.r,v2=a.c+d2*a.r;
            lines.push_back({u1,v1-u1}); lines.push_back({u2,v2-u2});
        }
        if (t==0)
        {
            const long double costh=(r+a.r)/d,sinth=sqrt(1-costh*costh);
            const Point d1=e.rot(costh,-sinth),d2=e.rot(costh,sinth);
            const Point u1=c+d1*r,u2=c+d2*r,v1=a.c-d1*a.r,v2=a.c-d2*a.r;
            lines.push_back({u1,v1-u1}); lines.push_back({u2,v2-u2});
        }
        return lines;
    }

    // 圆的反演
    tuple<int,Circle,Line> inverse(const Line &l) const
    {
        const Circle null_c={{0.0,0.0},0.0};
        const Line null_l={{0.0,0.0},{0.0,0.0}};
        if (l.toleft(c)==0) return {2,null_c,l};
        const Point v=l.toleft(c)==1?Point{l.v.y,-l.v.x}:Point{-l.v.y,l.v.x};
        const long double d=r*r/l.dis(c);
        const Point p=c+v/v.len()*d;
        return {1,{(c+p)/2,d/2},null_l};
    }

    tuple<int,Circle,Line> inverse(const Circle &a) const
    {
        const Circle null_c={{0.0,0.0},0.0};
        const Line null_l={{0.0,0.0},{0.0,0.0}};
        const Point v=a.c-c;
        if (a.is_in(c)==-1)
        {
            const long double d=r*r/(a.r+a.r);
            const Point p=c+v/v.len()*d;
            return {2,null_c,{p,{-v.y,v.x}}};
        }
        if (c==a.c) return {1,{c,r*r/a.r},null_l};
        const long double d1=r*r/(c.dis(a.c)-a.r),d2=r*r/(c.dis(a.c)+a.r);
        const Point p=c+v/v.len()*d1,q=c+v/v.len()*d2;
        return {1,{(p+q)/2,p.dis(q)/2},null_l};
    }
};

// 圆与多边形面积交
long double area_inter(const Circle &circ,const Polygon &poly)
{
    const auto cal=[](const Circle &circ,const Point &a,const Point &b)
    {
        if ((a-circ.c).toleft(b-circ.c)==0) return 0.0l;
        const auto ina=circ.is_in(a),inb=circ.is_in(b);
        const Line ab={a,b-a};
        if (ina && inb) return ((a-circ.c)^(b-circ.c))/2;
        if (ina && !inb)
        {
            const auto t=circ.inter(ab);
            const Point p=t.size()==1?t[0]:t[1];
            const long double ans=((a-circ.c)^(p-circ.c))/2;
            const long double th=(p-circ.c).ang(b-circ.c);
            const long double d=circ.r*circ.r*th/2;
            if ((a-circ.c).toleft(b-circ.c)==1) return ans+d;
            return ans-d;
        }
        if (!ina && inb)
        {
            const Point p=circ.inter(ab)[0];
            const long double ans=((p-circ.c)^(b-circ.c))/2;
            const long double th=(a-circ.c).ang(p-circ.c);
            const long double d=circ.r*circ.r*th/2;
            if ((a-circ.c).toleft(b-circ.c)==1) return ans+d;
            return ans-d;
        }
        const auto p=circ.inter(ab);
        if (p.size()==2 && Segment{a,b}.dis(circ.c)<=circ.r+eps)
        {
            const long double ans=((p[0]-circ.c)^(p[1]-circ.c))/2;
                      const  long  double  th1=(a-circ.c).ang(p[0]-circ.c),th2=(b-
circ.c).ang(p[1]-circ.c);
            const long double d1=circ.r*circ.r*th1/2,d2=circ.r*circ.r*th2/2;
            if ((a-circ.c).toleft(b-circ.c)==1) return ans+d1+d2;
            return ans-d1-d2;
        }
        const long double th=(a-circ.c).ang(b-circ.c);
        if ((a-circ.c).toleft(b-circ.c)==1) return circ.r*circ.r*th/2;
        return -circ.r*circ.r*th/2;
    };

    long double ans=0;
    for (size_t i=0;i<poly.p.size();i++)
    {
        const Point a=poly.p[i],b=poly.p[poly.nxt(i)];
        ans+=cal(circ,a,b);
    }
    return ans;
}
```

判断多条线段是否有交点

```
// 判断多条线段是否有交点
// 扫描线，复杂度 O(nlogn)
bool segs_inter(const vector<Segment> &segs)
{
    if (segs.empty()) return false;
    using seq_t=tuple<point_t,int,Segment>;
    const auto seqcmp=[](const seq_t &u, const seq_t &v)
    {
        const auto [u0,u1,u2]=u;
         const auto [v0,v1,v2]=v;
         if (abs(u0-v0)<=eps) return make_pair(u1,u2)<make_pair(v1,v2);
         return u0<v0-eps;
     };
     vector<seq_t> seq;
     for (auto seg:segs)
     {
         if (seg.a.x>seg.b.x+eps) swap(seg.a,seg.b);
         seq.push_back({seg.a.x,0,seg});
         seq.push_back({seg.b.x,1,seg});
     }
    sort(seq.begin(),seq.end(),seqcmp);
    point_t x_now;
    auto cmp=[&](const Segment &u, const Segment &v)
    {
        if (abs(u.a.x-u.b.x)<=eps || abs(v.a.x-v.b.x)<=eps) return u.a.y<v.a.y-eps;
                 return  ((x_now-u.a.x)*(u.b.y-u.a.y)+u.a.y*(u.b.x-u.a.x))*(v.b.x-v.a.x)<((x_now-v.a.x)*(v.b.y-v.a.y)+v.a.y*(v.b.x-v.a.x))*(u.b.x-u.a.x)-eps;
    };
    multiset<Segment,decltype(cmp)> s{cmp};
    for (const auto [x,o,seg]:seq)
    {
        x_now=x;
        const auto it=s.lower_bound(seg);
        if (o==0)
        {
            if (it!=s.end() && seg.is_inter(*it)) return true;
            if (it!=s.begin() && seg.is_inter(*prev(it))) return true;
            s.insert(seg);
        }
        else
        {
          if (next(it)!=s.end() && it!=s.begin() && (*prev(it)).is_inter(*next(it))) return true;
            s.erase(it);
        }
    }
    return false;
}
```

半平面交

```
// 半平面交
// 排序增量法，复杂度 O(nlogn)
// 输入与返回值都是用直线表示的半平面集合
vector<Line> halfinter(vector<Line> l, const point_t lim=1e9)
{
      const  auto  check=[](const  Line  &a,const  Line  &b,const  Line  &c){return a.toleft(b.inter(c))<0;};
    // 无精度误差的方法，但注意取值范围会扩大到三次方
    /*const auto check=[](const Line &a,const Line &b,const Line &c)
    {
            const  Point  p=a.v*(b.v^c.v),q=b.p*(b.v^c.v)+b.v*(c.v^(b.p-c.p))-a.p*(b.v^c.v);
        return p.toleft(q)<0;
    };*/
    l.push_back({{-lim,0},{0,-1}}); l.push_back({{0,-lim},{1,0}});
    l.push_back({{lim,0},{0,1}}); l.push_back({{0,lim},{-1,0}});
    sort(l.begin(),l.end());
    deque<Line> q;
    for (size_t i=0;i<l.size();i++)
    {
        if (i>0 && l[i-1].v.toleft(l[i].v)==0 && l[i-1].v*l[i].v>eps) continue;
        while (q.size()>1 && check(l[i],q.back(),q[q.size()-2])) q.pop_back();
        while (q.size()>1 && check(l[i],q[0],q[1])) q.pop_front();
        if (!q.empty() && q.back().v.toleft(l[i].v)<=0) return vector<Line>();
        q.push_back(l[i]);
    }
    while (q.size()>1 && check(q[0],q.back(),q[q.size()-2])) q.pop_back();
    while (q.size()>1 && check(q.back(),q[0],q[1])) q.pop_front();
    return vector<Line>(q.begin(),q.end());
}
```

圆面积并

```
// 圆面积并
// 轮廓积分，复杂度 O(n^2logn)
// ans[i] 表示被至少覆盖了 i+1 次的区域的面积
vector<long double> area_union(const vector<Circle> &circs)
{
    const size_t siz=circs.size();
    using arc_t=tuple<Point,long double,long double,long double>;
    vector<vector<arc_t>> arcs(siz);
    const auto eq=[](const arc_t &u,const arc_t &v)
     {
        const auto [u1,u2,u3,u4]=u;
        const auto [v1,v2,v3,v4]=v;
        return u1==v1 && abs(u2-v2)<=eps && abs(u3-v3)<=eps && abs(u4-v4)<=eps;
     };

     auto cut_circ=[&](const Circle &ci,const size_t i)
     {
        vector<pair<long double,int>> evt;
        evt.push_back({-PI,0}); evt.push_back({PI,0});
        int init=0;
        for (size_t j=0;j<circs.size();j++)
        {
            if (i==j) continue;
            const Circle &cj=circs[j];
            if (ci.r<cj.r-eps && ci.relation(cj)>=3) init++;
            const auto inters=ci.inter(cj);
          if (inters.size()==1) evt.push_back({atan2l((inters[0]-ci.c).y,(inters[0]-
ci.c).x),0});
            if (inters.size()==2)
            {
                const Point dl=inters[0]-ci.c,dr=inters[1]-ci.c;
                long double argl=atan2l(dl.y,dl.x),argr=atan2l(dr.y,dr.x);
                if (abs(argl+PI)<=eps) argl=PI;
                if (abs(argr+PI)<=eps) argr=PI;
                if (argl>argr+eps)
                {
                    evt.push_back({argl,1}); evt.push_back({PI,-1});
                    evt.push_back({-PI,1}); evt.push_back({argr,-1});
                }
                else
                {
                    evt.push_back({argl,1});
                    evt.push_back({argr,-1});
                }
            }
        }
        sort(evt.begin(),evt.end());
        int sum=init;
        for (size_t i=0;i<evt.size();i++)
        {
```

```cpp
            sum+=evt[i].second;
            if (abs(evt[i].first-evt[i+1].first)>eps) arcs[sum].push_back({ci.c,ci.r,evt[i].first,evt[i+1].first});
            if (abs(evt[i+1].first-PI)<=eps) break;
        }
    };

    const auto oint=[](const arc_t &arc)
    {
        const auto [cc,cr,l,r]=arc;
        if (abs(r-l-PI-PI)<=eps) return 2.0l*PI*cr*cr;
        return cr*cr*(r-l)+cc.x*cr*(sin(r)-sin(l))-cc.y*cr*(cos(r)-cos(l));
    };

    for (size_t i=0;i<circs.size();i++)
    {
        const auto &ci=circs[i];
        cut_circ(ci,i);
    }
    vector<long double> ans(siz);
    for (size_t i=0;i<siz;i++)
    {
        long double sum=0;
        sort(arcs[i].begin(),arcs[i].end());
        int cnt=0;
        for (size_t j=0;j<arcs[i].size();j++)
        {
          if (j>0 && eq(arcs[i][j],arcs[i][j-1])) arcs[i+(++cnt)].push_back(arcs[i][j]);
            else cnt=0,sum+=oint(arcs[i][j]);
        }
        ans[i]=sum/2;
    }
    return ans;
}
```

多边形面积并

```cpp
// 多边形面积并
// 轮廓积分，复杂度 O(n^2logn)，n 为边数
// ans[i] 表示被至少覆盖了 i+1 次的区域的面积
vector<long double> area_union(const vector<Polygon> &polys)
{
    const size_t siz=polys.size();
    vector<vector<pair<Point,Point>>> segs(siz);
    const auto check=[](const Point &u,const Segment &e){return !((u<e.a && u<e.b) || (u>e.a && u>e.b));};

    auto cut_edge=[&](const Segment &e,const size_t i)
    {
        const Line le{e.a,e.b-e.a};
        vector<pair<Point,int>> evt;
        evt.push_back({e.a,0}); evt.push_back({e.b,0});
        for (size_t j=0;j<polys.size();j++)
        {
            if (i==j) continue;
            const auto &pj=polys[j];
            for (size_t k=0;k<pj.p.size();k++)
            {
                const Segment s={pj.p[k],pj.p[pj.nxt(k)]};
                if (le.toleft(s.a)==0 && le.toleft(s.b)==0)
                {
                    evt.push_back({s.a,0});
                    evt.push_back({s.b,0});
                }
                else if (s.is_inter(le))
                {
                    const Line ls{s.a,s.b-s.a};
                    const Point u=le.inter(ls);
                    if (le.toleft(s.a)<0 && le.toleft(s.b)>=0) evt.push_back({u,-1});
                    else if (le.toleft(s.a)>=0 && le.toleft(s.b)<0) evt.push_back({u,1});
                }
            }
        }
        sort(evt.begin(),evt.end());
        if (e.a>e.b) reverse(evt.begin(),evt.end());
        int sum=0;
        for (size_t i=0;i<evt.size();i++)
        {
            sum+=evt[i].second;
            const Point u=evt[i].first,v=evt[i+1].first;
            if (!(u==v) && check(u,e) && check(v,e)) segs[sum].push_back({u,v});
            if (v==e.b) break;
        }
    };

    for (size_t i=0;i<polys.size();i++)
    {
        const auto &pi=polys[i];
        for (size_t k=0;k<pi.p.size();k++)
        {
            const Segment ei={pi.p[k],pi.p[pi.nxt(k)]};
            cut_edge(ei,i);
        }
    }
    vector<long double> ans(siz);
    for (size_t i=0;i<siz;i++)
    {
        long double sum=0;
        sort(segs[i].begin(),segs[i].end());
        int cnt=0;
        for (size_t j=0;j<segs[i].size();j++)
        {
            if (j>0 && segs[i][j]==segs[i][j-1]) segs[i+(++cnt)].push_back(segs[i][j]);
            else cnt=0,sum+=segs[i][j].first^segs[i][j].second;
        }
        ans[i]=sum/2;
    }
    return ans;
}
```

点集形成的最小最大三角形

```cpp
// 点集形成的最小最大三角形
// 极角序扫描线，复杂度 O(n^2logn)
// 最大三角形问题可以使用凸包与旋转卡壳做到 O(n^2)
pair<point_t,point_t> minmax_triangle(const vector<Point> &vec)
{
    if (vec.size()<=2) return {0,0};
    vector<pair<int,int>> evt;
    evt.reserve(vec.size()*vec.size());
    point_t maxans=0,minans=numeric_limits<point_t>::max();
     for (size_t i=0;i<vec.size();i++)
     {
         for (size_t j=0;j<vec.size();j++)
         {
             if (i==j) continue;
             if (vec[i]==vec[j]) minans=0;
             else evt.push_back({i,j});
         }
     }
     sort(evt.begin(),evt.end(),[&](const pair<int,int> &u,const pair<int,int> &v)
     {
         const Point du=vec[u.second]-vec[u.first],dv=vec[v.second]-vec[v.first];
         return argcmp()({du.y,-du.x},{dv.y,-dv.x});
     });
     vector<size_t> vx(vec.size()),pos(vec.size());
     for (size_t i=0;i<vec.size();i++) vx[i]=i;
     sort(vx.begin(),vx.end(),[&](int x,int y){return vec[x]<vec[y];});
     for (size_t i=0;i<vx.size();i++) pos[vx[i]]=i;
     for (auto [u,v]:evt)
     {
         const size_t i=pos[u],j=pos[v];
         const size_t l=min(i,j),r=max(i,j);
         const Point vecu=vec[u],vecv=vec[v];
         if (l>0) minans=min(minans,abs((vec[vx[l-1]]-vecu)^(vec[vx[l-1]]-vecv)));
        if (r<vx.size()-1) minans=min(minans,abs((vec[vx[r+1]]-vecu)^(vec[vx[r+1]]-
vecv)));
                                maxans=max({maxans,abs((vec[vx[0]]-vecu)^(vec[vx[0]]-
vecv)),abs((vec[vx.back()]-vecu)^(vec[vx.back()]-vecv))});
         if (i<j) swap(vx[i],vx[j]),pos[u]=j,pos[v]=i;
     }
     return {minans,maxans};
 }
```

点集的凸包

```cpp
// 点集的凸包
// Andrew 算法，复杂度 O(nlogn)
Convex convexhull(vector<Point> p)
{
    vector<Point> st;
    if (p.empty()) return Convex{st};
    sort(p.begin(),p.end());
    const auto check=[](const vector<Point> &st,const Point &u)
    {
         const auto back1=st.back(),back2=*prev(st.end(),2);
         return (back1-back2).toleft(u-back1)<=0;
     };
     for (const Point &u:p)
    {
        while (st.size()>1 && check(st,u)) st.pop_back();
        st.push_back(u);
    }
    size_t k=st.size();
    p.pop_back(); reverse(p.begin(),p.end());
    for (const Point &u:p)
    {
        while (st.size()>k && check(st,u)) st.pop_back();
        st.push_back(u);
    }
    st.pop_back();
    return Convex{st};
}
```

给定三点求圆心

```
void cal(Point& a, Point& b, Point& c){ //圆上三点定圆心
    double a1=b.x-a.x, b1=b.y-a.y, c1=(a1*a1+b1*b1)/2;
    double a2=c.x-a.x, b2=c.y-a.y, c2=(a2*a2+b2*b2)/2;
    double d =a1*b2-a2*b1;
    X =a.x+(c1*b2-c2*b1)/d;
    Y =a.y+(a1*c2-a2*c1)/d;
    r=dis2(a);
}

```