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
# linux下去掉.exe 和 加./
c = sys.argv[1]
code = c.capitalize()
os.system(f'g++ {c}.cpp -O2 -Wall -o {c}.exe')
try:
    x = zipfile.Zipfile(f'samples-{code}.zip')
    x.extractall(f'samples-{code}')
except:
    pass

for i in range(1,10000):
    if not os.path.isfile(f'samples-{code}/{i}.ans'):
        break
    print(f'Test {i}')
    os.system(f'{c}.exe < samples-{code}/{i}.in > samples-{code}/{i}.out')
    out = open(f'samples-{code}/{i}.out').read().replace('\r\n','\n')
    ans = open(f'samples-{code}/{i}.ans').read().replace('\r\n','\n')
    if(out != ans):
        print(f'WA {i}')
    else:
        print(f'AC {i}')
```

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

```
1  mt19937_64 rng(chrono::steady_clock::now().time_since_epoch().count());
2  int myRand(int B) {
3    return (unsigned long long)rng() % B;
4  }
```

Time

```
1  struct gettime{
2      clock_t star,ends;
3      gettime(){
4          star = clock();
5      }
6      ~gettime(){
7          ends = clock();
8          cout <<"Running Time : "<<(double)(ends - star)/ CLOCKS_PER_SEC << endl;
9      }
10 } tim;
11 int main()
12 {
13     tim.begin();
14     tim.end();
15     return 0;
16
17 }
```

子集枚举

```
1  for(int i=0;i<(1<<n);i++){
2      for(int j=i;j;j=(j-1)&i){
3
4      }
5  }
```

高维前缀和

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

三分

```
1  double f(double x){
2      //something
3  }
4  const double eps=1e-8;
5  double sanfen(double l, double r){
6      double mid,midr,ans;
7      while (fabs(r-l)>eps) {
8          mid=(l+r)/2;
9          midr=(mid+r)/2;
10         if(f(mid) < f(midr)) l=mid; else r=midr;    //求最大值
11     }
12     ans=f(l);
13     return ans;
14 }
```

Bitset

```
1  template<int LEN>void solve(){
2      sz=a.size();
3      if (LEN<=b.size()){
4          solve<min(N,LEN+10)>();
5          return;
6      }
7      using Bitset=bitset<LEN>;
8      Bitset is[sz+5];
9      for(int i=0;i<sz;i++)for(int j=0;j<b.size();j++){
10         auto&[x,l,r]=a[i];
11         auto&[y,L,R]=b[j];
12         if(l<=y&&y<=r&&L<=x&&x<=R)is[i][j]=1;
13     }
14     ll rs=0;
15     for(int i=0,x;i<sz;i++)for(int j=i+1;j<sz;j++){
16         x=(is[i]&is[j]).count();
17         rs+=x*(x-1)/2;
18     }
19     cout<<rs<<'\n';
20 }
```

Bitset 手写

```
1  const int N=3000;
2  typedef unsigned long long ull;
3
4  int lim=N/64+3;
5  struct Bitset{
6    ull v[N/64+5];
7   void init(){
8     memset(v,0,sizeof(v));
9     return;
10   }
11   void add(int x){
12    v[x>>6]|=(1ull<<(x&63));
13    return;
14   }
15   void shift1(){
16    int lst=0;
17    for(int i=0;i<=lim;i++){
18      int cur=v[i]>>63;
19       v[i]<<=1;v[i]|=lst;
20       lst=cur;
21    }
22     return;
23   }
24   int count(){
25    int res=0;
26    for(int i=0;i<=lim;i++) res+=__builtin_popcountll(v[i]);
27    return res;
28   }
29   Bitset operator|(const Bitset &x)const{
30     Bitset res;
31     for(int i=0;i<=lim;i++) res.v[i]=v[i]|x.v[i];
32     return res;
33   }
```

```
34    Bitset operator&(const Bitset &x)const{
35      Bitset res;
36      for(int i=0;i<=lim;i++) res.v[i]=v[i]&x.v[i];
37      return res;
38    }
39    Bitset operator^(const Bitset &x)const{
40      Bitset res;
41      for(int i=0;i<=lim;i++) res.v[i]=v[i]^x.v[i];
42      return res;
43    }
44    Bitset operator-(const Bitset &x)const{
45      Bitset res;ull lst=0;
46      for(int i=0;i<=lim;i++){
47       ull cur=(v[i]<x.v[i]+lst);
48       res.v[i]=v[i]-x.v[i]-lst;
49        lst=cur;
50    }
51      return res;
52    }
53 }
```

Lcslen(n2/w)

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N=3010;
4  typedef unsigned long long ull;
5  const int lim=N/64+3;
6  struct Bitset{
7   ull v[N/64+5];
8   void init(){
9      memset(v,0,sizeof(v));
10      return;
11    }
12    void add(int x){
13     v[x>>6]|=(1ull<<(x&63));
14      return;
15    }
16    void shift1(){
17     int lst=0;
18     for(int i=0;i<=lim;i++){
19       int cur=v[i]>>63;
20        v[i]<<=1;v[i]|=lst;
21        lst=cur;
22    }
23      return;
24    }
25    int count(){
26     int res=0;
27     for(int i=0;i<=lim;i++) res+=__builtin_popcountll(v[i]);
28     return res;
29    }
30    Bitset operator|(const Bitset &x)const{
31      Bitset res;
32      for(int i=0;i<=lim;i++) res.v[i]=v[i]|x.v[i];
33      return res;
34    }
```

```
35    Bitset operator&(const Bitset &x)const{
36      Bitset res;
37      for(int i=0;i<=lim;i++) res.v[i]=v[i]&x.v[i];
38      return res;
39    }
40    Bitset operator^(const Bitset &x)const{
41      Bitset res;
42      for(int i=0;i<=lim;i++) res.v[i]=v[i]^x.v[i];
43      return res;
44    }
45    Bitset operator-(const Bitset &x)const{
46      Bitset res;ull lst=0;
47      for(int i=0;i<=lim;i++){
48       ull cur=(v[i]<x.v[i]+lst);
49       res.v[i]=v[i]-x.v[i]-lst;
50        lst=cur;
51     }
52      return res;
53    }
54  }ch[26],f,g;
55  auto getans = [&](int mid){
56      int l1=mid,l2=n-mid;
57      f.init();g.init();
58      for(int i=0;i<26;i++){
59          ch[i].init();
60      }
61      for(int i=mid+1;i<=n;i++)ch[s[i]-'a'].add(i);
62      for(int i=1;i<=mid;i++){
63          g=f|ch[s[i]-'a'];
64          f.shift1();
65          f.add(1);
66          f=g-f;
67          f=f^g;f=f&g;
68      }
69      return f.count();
70  };
```

wqs 二分

```
1  // 这里是 上凸 取 min
2  // 上凸取 max 二分的时候改变一下 mid 的变化方向
3  // 下凸取 min 改变 mid 算贡献的符号
4  //min max 指的是求的是最大值 还是最小值
5  int solve(int mid){
6      k=mid;
7      function<void(int,int)> dfs = [&](int x,int h){
8          info dp2[3];memset(dp2,0,sizeof(dp2));
9          for(auto [it,w]:ve[x])if(it!=h){
10              dfs(it,x);
11              gmax(dp2[0],dp[x][0]+dp[it][0]);
12              gmax(dp2[2],dp[x][0]+dp[it][1]);
13              gmax(dp2[2],dp[x][2]+dp[it][0]);
14              dp[x][0]=dp2[0];
15              dp[x][1]=dp2[1];
16              dp[x][2]=dp2[2];
17          }
18          dp[x][0]=dp2[2];
```

```
19          dp[x][1]=dp2[0]+info{a[x]+k,1};
20      };
21      dfs(1,0);
22      return dp[1][0].second;
23 }
24 signed main()
25 {
26      int l=-sum-1000000000000ll,r=sum+1000000000000ll;
27      int ans=1e18;
28      while(l<=r){
29          int mid=(l+r)/2ll;
30          if(solve(mid)>=m){
31              ans=dp[1][0].first-m*mid;
32              r=mid-1;
33          }
34          else l=mid+1;
35      }
36      if(ans!=(int)1e18)cout<<ans<<"\n";
37      else cout<<"Impossible\n";
38      return 0;
39 }
```

判断异或方程组是否有解

```
1  const int maxn=1e2+5;
2  //每个方程组一定是 xi1^xi2^xi3...=1/0 的形式
3  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
4  //b[n+1]存方程右边等于 0/1
5  int sum[maxn];
6  bool check(int x)
7  {
8      for(int i=1;i<=n;i++)if(b[x][i])return true;
9      return !b[x][n+1];
10 }
11 bool solve()
12 {
13     for(int i=1,p=1;i<=n;i++,p++)
14     {
15         if(!b[p][i])
16         {
17             int pos=0;
18             for(int j=p+1; j<=n; j++)
19                 if(b[j][i])
20                 {
21                     pos=j;
22                     break;
23                 }
24             if(pos)swap(b[p],b[pos]);
25         }
26         int flag=b[p][i];
27         for(int j=p+1; j<=n; j++) if(b[j][i]) b[j]^=b[p],flag=1;
28         if(!flag) p--;
29     }
30     for(int i=1; i<=n; i++) if(!check(i)) return false;
31     return true;
32 }
```

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) {
```

```
40      string s;
41      do{
42          s.push_back(char(num%10+'0'));
43          num/=10;
44      }while(num>0);
45      reverse(s.begin(),s.end());
46      out<<s;
47      return out;
48  }
```

树哈希

```cpp
1  #include <cctype>
2  #include <chrono>
3  #include <cstdio>
4  #include <random>
5  #include <set>
6  #include <vector>
7
8  typedef unsigned long long ull;
9
10 const ull mask = std::chrono::steady_clock::now().time_since_epoch().count();
11
12 ull h(ull x) {
13     return x * x * x * 1237123 + 19260817;
14 }
15 ull f(ull x) {
16     ull cur = h(x & ((1 << 31) - 1)) + h(x >> 31);
17     return cur;
18 }
19 ull shift(ull x) {
20   x ^= mask;
21   x ^= x << 13;
22   x ^= x >> 7;
23   x ^= x << 17;
24   x ^= mask;
25   return x;
26 }
27
28 const int N = 1e6 + 10;
29
30 int n;
31 ull hash[N];
32 std::vector<int> edge[N];
33 std::set<ull> trees;
34
35 void getHash(int x, int p) {
36   hash[x] = 1;
37   for (int i : edge[x]) {
38     if (i == p) {
39       continue;
40     }
41     getHash(i, x);
42     hash[x] += shift(hash[i]);
43   }
44   trees.insert(hash[x]);
45 }
```

```
46
47  int main() {
48    scanf("%d", &n);
49    for (int i = 1; i < n; i++) {
50      int u, v;
51      scanf("%d%d", &u, &v);
52      edge[u].push_back(v);
53      edge[v].push_back(u);
54    }
55    getHash(1, 0);
56    printf("%lu", trees.size());
57  }
58
```

求 s 所有前缀对于 t 的所有子串的最长公共子序列长度

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   struct PairLCS {
4       vector<vector<int>> ih, iv;
5       int n = 0, m = 0;
6       PairLCS(string s, string t) : n(s.size()), m(t.size()) {
7           ih = iv = vector<vector<int>>(n + 1, vector<int>(m + 1));
8           iota(ih[0].begin(),ih[0].end(), 0);
9           for (int i = 1; i <= n; i++) {
10              for (int j = 1; j <= m; j++) {
11                  if (s[i - 1] == t[j - 1]) {
12                      ih[i][j] = iv[i][j - 1];
13                      iv[i][j] = ih[i - 1][j];
14                  } else {
15                      ih[i][j] = std::max(ih[i - 1][j], iv[i][j - 1]);
16                      iv[i][j] = std::min(ih[i - 1][j], iv[i][j - 1]);
17                  }
18              }
19          }
20      }
21      int query(int a, int b, int c) const {
22          int res = 0;
23          for (int i = b + 1; i <= c; i++) res += ih[a][i] <= b;
24          return res;
25      }  // s[0,a) t[b,c)
26  };
27
28
29
30  int cas;
31
32  void solution() {
33    int q;
34    std::string s, t;
35    std::cin >> q >> s >> t;
36
37    // int n = s.size(), m = t.size();
38    PairLCS solver(s, t);
39
40    for (int _ = 0; _ < q; _++) {
41      int a, b, c;
```

11

```
42        std::cin >> a >> b >> c;
43        std::cout << solver.query(a, b, c) << '\n';
44    }
45 }
46
47
48 int main() {
49    ios::sync_with_stdio(false);
50    cin.tie(0);
51    int T = 1;
52    // std::cin >> T;
53    for (cas = 1; cas <= T; cas++) solution();
54
55    return 0;
56 }
```

线性基

```
1 struct LinearBasis
2 {
3      static const int maxbase = 35;
4      bool flag = false;
5      ll a[maxbase + 1];
6      int tot;
7      LinearBasis()
8      {
9          memset(a, 0, sizeof a);
10         tot=0;
11     }
12     LinearBasis(ll *x, int n)
13     {
14         LinearBasis();
15         build(x, n);
16     }
17     void build(ll *x, int n)
18     {
19         for(int i = 1; i <= n; ++i)
20             insert(x[i]);
21     }
22     void clear()
23     {
24         memset(a, 0, sizeof a);
25     }
26     bool insert(ll t)
27     {
28         //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
29         for(int i = maxbase; i >= 0; --i)
30         {
31             if(t & (1ll << i))
32             {
33                 if(!a[i])
34                 {
35                     a[i] = t;//这里表示插入成功
36                     break;
37                 }
38                 t ^= a[i];
```

```
39                    }
40                }
41                if(t == 0)flag = true;
42                return t;
43            }
44        bool query(ll t)
45        {
46                // 询问 t 是否可以被当前线性基表示，不插入
47                if(t > queryMax())return false;
48                if(t == 0)return true;
49                for(int i = maxbase; i >= 0; --i)
50                {
51                    if(t & (1ll << i))
52                    {
53                        if(!a[i])
54                        {
55                            return false;
56                        }
57                        t ^= a[i];
58                    }
59                }
60                return true;
61        }
62        void Insert(ll t)
63        {
64                //插入一个线性基，利用高斯消元法维护一个对角矩阵
65                for(int i = maxbase; i >= 0; --i)
66                {
67                    if(t >> i & 1)
68                    {
69                        if(a[i])t ^= a[i];
70                        else
71                        {
72                            a[i] = t;
73                            for(int j = i - 1; j >= 0; --j)if(a[j] && (a[i] >> j & 1))a[i]
    ^= a[j];
74                            for(int j = i + 1; j <= maxbase; ++j)if(a[j] >> i & 1)a[j] ^= a[i];
75                            break;
76                        }
77                    }
78                }
79        }
80        LinearBasis merge(const LinearBasis &l1, const LinearBasis &l2)
81        {
82                // 得到两个线性基的并
83                LinearBasis ret = l1;
84                for(int i = maxbase; i >= 0; --i)
85                    if(l2.a[i])
86                        ret.insert(l2.a[i]);
87                return ret;
88        }
89        LinearBasis intersection(const LinearBasis &l1, const LinearBasis &l2)
90        {
91                //得到两个线性基的交
92                LinearBasis all, ret, full;
93                ret.clear();
```

```
 94        for(int i = maxbase; i >= 0; --i)
 95        {
 96            all.a[i] = l1.a[i];
 97            full.a[i] = 1ll << i;
 98        }
 99        for(int i = maxbase; i >= 0; --i)
100        {
101            if(l2.a[i])
102            {
103                ll v = l2.a[i], k = 0;
104                bool flag = true;
105                for(int j = maxbase; j >= 0; --j)
106                {
107                    if(v & (1ll << j))
108                    {
109                        if(all.a[j])
110                        {
111                            v ^= all.a[j];
112                            k ^= full.a[j];
113                        }
114                        else
115                        {
116                            // l2's basis is not in l1's;
117                            flag = false;
118                            all.a[j] = v;
119                            full.a[j] = k;
120                            break;
121                        }
122                    }
123                }
124                if(flag)
125                {
126                    ll v = 0; // get intersection by k;
127                    for(int j = maxbase; j >= 0; --j)
128                    {
129                        if(k & (1ll << j))
130                        {
131                            v ^= l1.a[j];
132                        }
133                    }
134                    ret.insert(v);
135                }
136            }
137        }
138        return ret;
139    }
140    //询问最值
141    ll queryMax()
142    {
143        ll ret = 0;
144        for(int i = maxbase; i >= 0; --i)
145            if((ret ^ a[i]) > ret)
146                ret ^= a[i];
147        return ret;
148    }
149    ll queryMin()
```

```
150     {
151         for(int i = 0; i <= maxbase; ++i)
152             if(a[i])
153                 return a[i];
154         return 0;
155     }
156     ll Kth_Max(ll k)
157     {
158         ll res=0;
159         for(int i=62;i>=0;i--)
160             if (k&(1LL<<i)) res^=a[i];
161         return res;
162     }
163 };
164
```

维护多个二维向量能够表示的范围

```
1  int gcd(int x,int y){
2      if(y==0)return x;
3      else return gcd(y,x%y);
4  }
5  struct vec{
6      int a00,a01,a11;
7      void clear(){
8          a00=a01=a11=0;
9      }
10     void insert(int x,int y){
11         while(x!=0){
12             int t=a00/x;
13             a00-=x*t;
14             a01-=y*t;
15             swap(a00,x);
16             swap(a01,y);
17         }
18         a11=gcd(a11,abs(y));
19         if(a11!=0)a01%=a11;
20     }
21     bool query(int x,int y){
22         if(x!=0){
23             if(a00==0||x%a00!=0)return false;
24             int t=x/a00;
25             y-=a01*t;
26         }
27         if(y==0){
28             return true;
29         }
30         else return a11!=0&&y%a11==0;
31     }
32 };
```

字符串

AC 自动机

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const int maxn=1e6+10;
4  int n;
5  char c[maxn];
6  struct AC{
7      int trie[maxn][26],tot;
8      int e[maxn],fail[maxn],old[maxn];
9      void init(){
10         memset(trie,0,sizeof(trie));
11         memset(e,0,sizeof(e));
12         memset(fail,0,sizeof(fail));
13         memset(old,0,sizeof(old));
14         tot=0;
15     }
16     void insert(char *t){
17         int x=0;
18         for(int i=1;t[i];i++){
19             if(!trie[x][t[i]-'a']){
20                 trie[x][t[i]-'a']=++tot;
21             }
22             x=trie[x][t[i]-'a'];
23         }
24         e[x]++;
25     }
26     queue<int> qu;
27     void build(){
28         for(int i=0;i<26;i++){
29             if(trie[0][i])qu.push(trie[0][i]);
30         }
31         while(!qu.empty()){
32             int x=qu.front();
33             qu.pop();
34             for(int i=0;i<26;i++){
35                 if(trie[x][i]){
36                     fail[trie[x][i]]=trie[fail[x]][i];
37                     qu.push(trie[x][i]);
38                 }
39                 else trie[x][i]=trie[fail[x]][i];
40                 old[trie[x][i]]=e[fail[trie[x][i]]] ? fail[trie[x][i]] : old[fail[trie[x][i]]];
41             }
42         }
43     }
44     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
45         int x=0,res=0;
46         for(int i=1;t[i];i++){
47             x=trie[x][t[i]-'a'];
48             for(int j=x;j&&e[j]!=-1;j=old[j]){
49                 res+=e[j];
50                 e[j]=-1;
51             }
52         }
```

```
53          return res;
54      }
55  };
56  AC ac;
57  int main()
58  {
59      scanf("%d",&n);
60      for(int i=1;i<=n;i++){
61          scanf("%s",c+1);
62          ac.insert(c);
63      }
64      ac.build();
65      scanf("%s",c+1);
66
67      cout<<ac.query(c)<<endl;
68      return 0;
69  }
```

Dequehash

```
1   /*
2   严格 0base，不用管任何函数里面的东西，用就可以了，不要越界
3   pair<int,int> first 表示哈希 sum，second 表示当前位置的值
4   */
5   #define int long long
6   #define sz(a) (int)((a).size())
7   const int maxn=3e5+10;
8   const int mod=1e9+7,base=1331;
9   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;}
10  void add(int& a, int val, int p = mod) {if ((a = (a + val)) >= p) a -= p;}
11  void sub(int& a, int val, int p = mod) {if ((a = (a - val)) < 0) a += p;}
12  int mul(int a, int b, int p = mod) {return (int) a * b % p;}
13  int inv(int a, int p = mod) {return fpow(a, p - 2, p);}
14  int p[maxn],ip[maxn];
15  void init()
16  {
17      p[0] = 1; for(int i=1;i<maxn;i++) p[i] = mul(p[i - 1], base, mod);
18      for(int i=0;i<maxn;i++) ip[i] = inv(p[i], mod);
19  }
20  struct extendable_sequence {
21      deque<pair<int,int>> dq;
22      int pow_offset;
23
24      extendable_sequence() {
25          pow_offset = 0;
26          dq.push_back(make_pair(0, 0));
27      }
28
29      int size() {
30          return sz(dq) - 1;
31      }
32
33      pair<int,int>& operator [] (int i) {
34          return dq[i+1];
35      }
36
```

```
37    void add_back(vector<int> vals) {
38        int t = dq.back().first;
39        for(int i=0;i<sz(vals);i++) {
40            add(t, mul(vals[i], mul(p[sz(dq) - 1], ip[pow_offset], mod), mod), mod);
41            dq.push_back(make_pair(t, vals[i]));
42        }
43    }
44
45    void add_front(vector<int> vals) {
46        pow_offset += sz(vals);
47        int t = dq.front().first;
48        for(int i=sz(vals)-1;i>=0;i--) {
49            dq.front().second = vals[i];
50            sub(t, mul(vals[i], mul(p[i], ip[pow_offset], mod), mod), mod);
51            dq.push_front(make_pair(t, 0));
52        }
53    }
54
55    int calc(int l, int r) {
56        l++, r++;
57        if (l > r) return 0;
58        int res = dq[r].first;
59        sub(res, dq[l - 1].first, mod);
60        res = mul(res, ip[l - 1], mod);
61        res = mul(res, p[pow_offset], mod);
62        return res;
63    }
64 };
65 //返回(x+y)[l 到 r]的哈希值
66 int calc(extendable_sequence& x, extendable_sequence& y, int l, int r) {
67     int res = x.calc(l, min(r, sz(x) - 1));
68     add(res, mul(y.calc(max(0ll, l - sz(x)), r - sz(x)), p[sz(x)], mod), mod);
69     return res;
70 }
71 //返回(x+y)[i]单个元素的值
72 int calc(extendable_sequence& x, extendable_sequence& y, int i) {
73     if (i < sz(x)) {
74         return x[i].second;
75     }
76     if (i - sz(x) < sz(y)) {
77         return y[i - sz(x)].second;
78     }
79     return -1;
80 }
```

Exkmp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const int maxn = 1e6 + 1e3;
4  struct EXKMP
5  { // S 里找 T
6
7      char S[maxn], T[maxn];
8      int next[maxn], extend[maxn];
9      void Get_Next()
10     {
```

```cpp
        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;
}
```

19

Hash

```
1  const int N=1e6+10;
2  typedef long long ll;
3  const ll p1=31,p2=131;
4  const ll mod1=1e9+7,mod2=1e9+9;
5  typedef pair<ll,ll> hs;
6  const hs p = make_pair(p1,p2);
7  hs &operator+=(hs &a, hs b) {
8      a.first=(a.first+b.first)%mod1;
9      a.second=(a.second+b.second)%mod2;
10     return a;
11 }
12 hs operator+(hs a, hs b) { return a += b; }
13 hs &operator-=(hs &a, hs b) {
14     a.first=(a.first-b.first+mod1)%mod1;
15     a.second=(a.second-b.second+mod2)%mod2;
16     return a;
17 }
18 hs operator-(hs a, hs b) { return a -= b; }
19 hs &operator*=(hs &a, hs b) {
20     a.first=(a.first*b.first)%mod1;
21     a.second=(a.second*b.second)%mod2;
22     return a;
23 }
24 hs operator*(hs a, hs b) { return a *= b; }
25 struct Hash{
26     int n;
27     vector<hs>has1,has2,Pow;
28     void Hash_init(string &s){
29         n=(int)s.size();
30         Pow.resize(n+2);
31         has1.resize(n+2);
32         has2.resize(n+2);
33         Pow[0]=make_pair(1ll,1ll);
34         for(int i=1;i<=n;i++)Pow[i]=Pow[i-1]*p;
35         for(int i=1;i<=n;i++)has1[i]=has1[i-1]*p+hs{s[i-1]-'a'+1,s[i-1]-'a'+1};
36         for(int i=n;i>=1;i--)has2[i]=has2[i+1]*p+hs{s[i-1]-'a'+1,s[i-1]-'a'+1};
37     }
38     hs get1(int l,int r){
39         return has1[r]-has1[l-1]*Pow[r-l+1];
40     }
41     hs get2(int l,int r){
42         return has2[l]-has2[r+1]*Pow[r-l+1];
43     }
44 };
```

Kmp

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const int maxn=1e6+10;
4  struct KMP{//1base
5      int len1,n,nxt[maxn];//nxt 表示以 i 为结尾的前缀与后缀相同的长度
6      char s1[maxn],s[maxn];
7      void build()
8      {
9          n=strlen(s+1);
```

```
10          nxt[1]=0;
11          int x=2,now=1;//x 是 s2 当前搜索到的位置，now 是前缀位置
12          while(x<=n){
13              if(s[x]==s[now]){
14                  nxt[x]=now;
15                  now++;
16                  x++;
17              }
18              else{
19                  if(now>1){
20                      now=nxt[now-1]+1;
21                  }
22                  else{
23                      nxt[x]=0;
24                      now=1;
25                      x++;
26                  }
27              }
28          }
29      }
30      void find(){//s1 1base
31          int now=1,tar=1;
32          len1=strlen(s1+1);
33          while(tar<=len1){
34              if(s1[tar]==s[now]){
35                  tar++;
36                  now++;
37              }
38              else{
39                  if(now>1){
40                      now=nxt[now-1]+1;
41                  }
42                  else tar++;
43              }
44              if(now==n+1){
45                  printf("%d\n",tar-now+1);
46              }
47          }
48      }
49  };
50  KMP sol;
51  int main()
52  {
53      scanf("%s%s",sol.s1+1,sol.s+1);
54      sol.build();
55      sol.find();
56      for(int i=1;i<=sol.n;i++)cout<<sol.nxt[i]<<" ";
57      return 0;
58  }
```

Manacher

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const int maxn = 1.1e7 + 5;
4  struct ST{
5   char s[maxn * 2], str[maxn * 2];
```

```
6    int Len[maxn * 2], len;
7     void getstr() {//重定义字符串
8        int k = 0;
9      len = strlen(s);
10     str[k++] = '@';//开头加个特殊字符防止越界
11       for (int i = 0; i < len; i++) {
12          str[k++] = '#';
13          str[k++] = s[i];
14       }
15       str[k++] = '#';
16       len = k;
17     str[k] = 0;//字符串尾设置为 0，防止越界
18   }
19    int manacher() {
20     int mx = 0, id;//mx 为最右边，id 为中心点
21     int maxx = 0;
22       for (int i = 1; i < len; i++) {
23         if (mx > i) Len[i] = min(mx - i, Len[2 * id - i]);//判断当前点超没超过 mx
24         else Len[i] = 1;//超过了就让他等于 1，之后再进行查找
25         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
26         if (Len[i] + i > mx) {//更新 mx
27           mx = Len[i] + i;
28           id = i;//更新中间点
29           maxx = max(maxx, Len[i]);//最长回文字串长度
30         }
31       }
32     return (maxx - 1);
33   }
34    void writ(){
35     printf("%s\n",str);
36       for(int i=0;i<len;i++){
37            cout<<Len[i]<<" ";
38       }
39     cout<<"\n";
40   }
41 };
42 ST s1,s2;
43 int main() {
44   scanf("%s", s1.s);
45  s1.getstr();
46  printf("%d\n",s1.manacher());
47   return 0;
48 }
```

倍增 SA

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 const int N = 1e6 + 10;//2*strlen
4 struct Suffix{
5   int ht[N],rk[N],sa[N],y[N],c[N];
6    int n,m;
7    char s[N];
8    int st[20][N];
9   void init(){
```

```
10          n=strlen(s+1);
11          m=300;
12      for(int i=0;i<=m;i++) c[i]=0;
13       for(int i=0;i<=2*n;i++) y[i]=0;
14       for(int i=1;i<=n;i++) c[rk[i]=s[i]]++;
15      for(int i=1;i<=m;i++) c[i]+=c[i-1];
16       for(int i=n;i>=1;i--) sa[c[rk[i]]--]=i;
17       for(int k=1;k<=n;k<<=1){
18        int p=0;
19        for(int i=n-k+1;i<=n;i++) y[++p]=i;
20         for(int i=1;i<=n;i++){
21          if(sa[i]>k){
22            y[++p]=sa[i]-k;
23          }
24         }
25        for(int i=0;i<=m;i++) c[i]=0;
26         for(int i=1;i<=n;i++) c[rk[i]]++;
27         for(int i=1;i<=m;i++) c[i]+=c[i-1];
28         for(int i=n;i>=1;i--) sa[c[rk[y[i]]]--]=y[i];
29         for(int i=0;i<=n;i++) swap(rk[i],y[i]);
30        rk[sa[1]]=p=1;
31        for(int i=2;i<=n;i++){
32         rk[sa[i]]=(y[sa[i]] == y[sa[i-1]] && y[sa[i]+k] == y[sa[i-1]+k] ? p : ++p);
33        }
34        if(p>=n) break;
35        m=p;
36     }
37      for(int i=1,k=0;i<=n;i++){
38       if(k)k--;
39        int j=sa[rk[i]-1];
40       while(s[i+k] == s[j+k] )k++;
41       ht[rk[i]] = k;
42     }
43          for(int i=1;i<=n;i++)st[0][i]=ht[i];
44          for(int j=1;j<20;j++){
45              for(int i=1;i+(1<<j)-1<=n;i++)st[j][i]=min(st[j-1][i],st[j-1]
   [i+(1<<(j-1))]);
46          }
47   }
48     int get(int l,int r){
49          int g=__lg(r-l+1);
50          return min(st[g][l],st[g][r-(1<<g)+1]);
51     }
52     int lcp(int x,int y){
53          x=rk[x],y=rk[y];
54          if(x==y)return n-x+1;
55          if(x>y)swap(x,y);
56          return get(x+1,y);
57     }
58     int query(int l1,int r1,int l2,int r2){
59          int len=lcp(l1,l2);
60          len=min({len,r1-l1+1,r2-l2+1});
61          if(len==min(r1-l1+1,r2-l2+1)){
62              if(r1-l1>r2-l2)return 1;
63              if(r1-l1<r2-l2)return -1;
64              if(r1-l1==r2-l2)return 0;
```

```
65            }
66            char p=s[l1+len],q=s[l2+len];
67            if(p>q)return 1;
68            if(p==q)return 0;
69            return -1;
70        }
71    void writ()
72    {
73        printf("%s\n",s+1);
74        for(int i=1;i<=n;i++)cout<<sa[i]<<" ";;cout<<"\n";
75        for(int i=1;i<=n;i++)cout<<ht[i]<<" ";;cout<<"\n";
76        for(int i=1;i<=n;i++)cout<<rk[i]<<" ";;cout<<"\n";
77    }
78
79 };
80 Suffix suf;
```

后缀自动机 SAM

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N=2e6;
4  struct SAM {
5      struct Node {
6          int tr[26];
7          int len, fa;
8          Node() { memset(tr, 0, sizeof(tr)); len = fa = 0; }
9      }ep[N << 1];
10     int last, tot, n;
11     char base;
12     vector<int> edg[N << 1];
13     int siz[N << 1];
14     void init(int _n) {
15         last = tot = 1;
16         base = 'a';
17         for (int i = 0; i <= 2 * _n; i++) {
18             ep[i] = Node();
19             edg[i].clear();
20             siz[i] = 0;
21         }
22     }
23     void insert(char x) {
24         int c = x - base;
25         int p = last;
26         int np = last = ++tot;
27         siz[np] = 1;
28         ep[np].len = ep[p].len + 1;
29         for (; p && !ep[p].tr[c]; p = ep[p].fa)
30             ep[p].tr[c] = np;
31         if (!p) ep[np].fa = 1;
32         else {
33             int q = ep[p].tr[c];
34             if (ep[q].len == ep[p].len + 1) ep[np].fa = q;
35             else {
36                 int nq = ++tot;
37                 ep[nq] = ep[q];
38                 ep[nq].len = ep[p].len + 1;
```

```
39                    ep[q].fa = ep[np].fa = nq;
40                    for (; p && ep[p].tr[c] == q; p = ep[p].fa)
41                        ep[p].tr[c] = nq;
42                }
43            }
44        }
45        void construct() {
46            for (int i = 2; i <= tot; i++) {
47                edg[ep[i].fa].push_back(i);
48            }
49        }
50        void dfs(int u) {
51            for (auto v : edg[u]) {
52                dfs(v);
53                siz[u] += siz[v];
54            }
55        }
56        void build(string& s) {
57            n = s.size();
58            init(n);
59            for (int i = 0; i < n; i++) {
60                insert(s[i]);
61            }
62            construct();
63            dfs(1);
64        }
65
66  } sam;
67
```

回文自动机 PAM

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   const int N=2e6+10;
4   struct PAM_Trie
5   {
6    int ch[26];
7    int fail,len,num;
8   };
9   struct PAM
10  {
11    PAM_Trie b[N];
12    int n,length,last,cnt;
13    char s[N];
14    PAM()
15    {
16      b[0].len = 0; b[1].len = -1;
17      b[0].fail = 1; b[1].fail = 0;
18     last = 0;
19     cnt = 1;
20    }
21    int get_fail(int x)
22    {
23     while(s[n-b[x].len-1]!=s[n])
24     {
25         x=b[x].fail;
```

```
26        }
27        return x;
28    }
29    void insert()
30    {
31      int p=get_fail(last);
32     if(!b[p].ch[s[n]-'a'])
33      {
34        b[++cnt].len=b[p].len+2;
35        int tmp=get_fail(b[p].fail);
36        b[cnt].fail=b[tmp].ch[s[n]-'a'];
37        b[cnt].num=b[b[cnt].fail].num+1;
38        b[p].ch[s[n]-'a']=cnt;
39      }
40      last=b[p].ch[s[n]-'a'];
41     cout<<last<<"\n";
42       //如果要统计出现次数 f[last]++;
43    }
44    void init()
45    {
46      length=strlen(s+1);
47      for(n=1;n<=length;n++)
48    {
49        insert();
50    }
51    }
52  }pa;
53  int main()
54  {
55   scanf("%s",pa.s+1);
56    pa.init();
57    return 0;
58  }
```

最小表示

```
1  int getMin(string s) {
2      int i = 0, j = 1, k = 0;
3      int len = s.length();
4      while(i<len && j<len && k<len) {
5          int tmp = s[(i + k) % len] - s[(j + k) % len];
6          if(tmp==0) k++;
7          else {
8              if(tmp>0) i += k + 1;
9              else j += k + 1;
10             if(i==j) j++;
11             k = 0;
12         }
13     }
14     return min(i, j);
15 }
```

Runs

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  struct Runs {
```

```
 5      int l, r, p;
 6  };
 7  namespace RunsGetter {
 8      const int N = 2e5 + 5;
 9      using namespace std;
10      struct StrHash{
11          ll p[N],h[N];int mod,buf;
12          StrHash (int _buf,int _mod){buf=_buf;mod=_mod;}
13          void Init(char *s,int n)
14          {
15              p[0]=1;
16              for (int i=1;i<=n;i++){
17                  p[i]=p[i-1]*buf%mod;
18                  h[i]=(h[i-1]*buf+s[i]-'a'+1)%mod;
19              }
20          }
21          bool tl(int u,int v,int len){
22              return (h[u]-h[v]+(h[v-len]-h[u-len])*p[len])%mod==0;
23          }
24          bool tr(int u,int v,int len){
25              return (h[u+len-1]-h[v+len-1]+(h[v-1]-h[u-1])*p[len])%mod==0;
26          }
27          int get(int l,int r){
28              return (h[r]+(mod-h[l-1])*p[r-l+1])%mod;
29          }
30      }h1(10,1000000007),h2(13,1000000009);
31      int n;char s[N];
32      int extl(int u,int v){
33          if (s[u]!=s[v])return 0;
34          int l=1,r=min(u,v),mid;
35          while(l<r){
36              mid=(l+r+1)>>1;
37              if (h1.tl(u,v,mid)&&h2.tl(u,v,mid))l=mid;
38              else r=mid-1;
39          }return l;
40      }
41      int extr(int u,int v)
42      {
43          if (s[u]!=s[v])return 0;
44          int l=1,r=n-max(u,v)+1,mid;
45          while(l<r){
46              mid=(l+r+1)>>1;
47              if (h1.tr(u,v,mid)&&h2.tr(u,v,mid))l=mid;
48              else r=mid-1;
49          }
50          return l;
51      }
52      Runs b[N*20];
53      bool cmpR(const Runs &A, const Runs &B)
54      {return A.l==B.l ? (A.r==B.r  ? A.p<B.p : A.r<B.r) : A.l<B.l;}
55      int tot;
56      int chk(int u,int v)
57      {
58          int tl=extl(u,v),tr=extr(u,v);
59          if (tl+tr>=v-u+1){
60              b[++tot]=(Runs){u-tl+1,v+tr-1,v-u};
```

```
61            return v+tr-1;
62        }
63        return 0;
64    }
65    vector<Runs> getRuns(int n0, char* t0)
66    {
67        n=n0;
68        for(int i=1;i<=n0;i++)s[i]=t0[i];
69        h1.Init(s,n);h2.Init(s,n);
70        for (int p=1;p<=n;p++)
71            for (int k=p,mx=0;k+p<=n;k+=p)
72            if (mx<k)
73                mx=max(mx,chk(k,k+p)-p);
74        sort(b+1,b+tot+1,cmpR);
75        int m=0;
76        for (int i=1;i<=tot;i++)
77            if (b[i].l!=b[i-1].l||b[i].r!=b[i-1].r)
78            b[++m]=b[i];
79        tot=m;
80        vector<Runs> e;
81        for(int i=1;i<=m;i++)e.push_back(b[i]);
82        return e;
83    }
84 }
```

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
```

```
6   int vis[maxn];    // vis[i]标记第一次 dfs1()点 i 是否访问过
7   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
8   int cnt;      //cnt 表示强连通分量的个数
9   void dfs1(int x){
10      vis[x] = 1;
11      for(auto it:ve[x])if(!vis[it])
12          dfs1(it);
13      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
14  }
15
16  void dfs2(int x){
17      if(col[x])    return;
18      col[x] = cnt;
19      for(auto it:ve[x])if(!col[it])
20          dfs2(it);
21  }
22  void Kosaraju(int n) {
23      cnt = 0;
24      sta.clear();
25      memset(vis,0,sizeof(vis));
26      memset(col,0,sizeof(col));
27      for(int i=1; i<=n; i++) //搜索所有点
28          dfs1(i);
29      for(int i=n-1; i>=0; i--){
30          if(!col[sta[i]]){
31              cnt++;
32              dfs2(sta[i]);
33          }
34      }
35  }
36
```

K 短路

```
1   //复杂度 nlogn +mlogm+klogk
2   #include <bits/stdc++.h>
3   #include <queue>
4
5   template <class T, class U>
6   inline bool smin(T &x, const U &y) {
7     return y < x ? x = y, 1 : 0;
8   }
9   template <class T, class U>
10  inline bool smax(T &x, const U &y) {
11    return x < y ? x = y, 1 : 0;
12  }
13
14  using LL = long long;
15  using PII = std::pair<int, int>;
16
17  constexpr int N(2.5e5 + 5);
18
19  using T = LL;
20  struct Edge {
21    int x, y; T z;
22  };
23  struct Heap {
```

```
24    struct Node {
25      int ls, rs, h, v;
26      T w;
27    } t[N * 40];
28    int cnt;
29    int newNode(int v, T w) {
30      t[++cnt] = {0, 0, 1, v, w};
31      return cnt;
32    }
33    int merge(int x, int y) {
34      if (!x) return y;
35      if (!y) return x;
36      if (t[x].w > t[y].w) std::swap(x, y);
37      t[++cnt] = t[x], x = cnt;
38      t[x].rs = merge(t[x].rs, y);
39      if (t[t[x].ls].h < t[t[x].rs].h) std::swap(t[x].ls, t[x].rs);
40      t[x].h = t[t[x].rs].h + 1;
41      return x;
42    }
43  } h;
44
45
46  std::vector<T> kShortestPath(int n, int k, int s, int t, const std::vector<Edge> &e) {
47    int m = e.size();
48    std::vector<int> deg(n + 1), g(m);
49    for (auto &[x, y, z] : e) deg[y]++;
50    for (int i = 1; i <= n; i++) deg[i] += deg[i - 1];
51    for (int i = 0; i < m; i++) g[--deg[e[i].y]] = i;
52
53    std::vector<T> d(n, -1);
54    std::vector<int> fa(n, -1), p;
55
56    using Q = std::pair<T, int>;
57    std::priority_queue<Q, std::vector<Q>, std::greater<Q>> q;
58
59    {
60      p.reserve(n);
61      d[t] = 0, q.push({0, t});
62
63      std::vector<bool> vis(n);
64      while (!q.empty()) {
65        int x = q.top().second;
66        q.pop();
67        if (vis[x]) continue;
68        vis[x] = true;
69        p.push_back(x);
70        for (int i = deg[x]; i < deg[x + 1]; i++) {
71          auto &[y, _, z] = e[g[i]];
72          if (d[y] == -1 || d[y] > d[x] + z) {
73            d[y] = d[x] + z, fa[y] = g[i];
74            q.push({d[y], y});
75          }
76        }
77      }
78    }
79
```

```
 80    if (d[s] == -1) std::vector<T>(k, -1);
 81    std::vector<int> heap(n);
 82    h.cnt = 0;
 83    for (int i = 0; i < m; i++) {
 84      auto &[x, y, z] = e[i];
 85      if (d[x] != -1 && d[y] != -1 && fa[x] != i) {
 86        heap[x] = h.merge(heap[x], h.newNode(y, d[y] + z - d[x]));
 87      }
 88    }
 89
 90    for (int x : p) {
 91      if (x != t) heap[x] = h.merge(heap[x], heap[e[fa[x]].y]);
 92    }
 93
 94    if (heap[s]) q.push({d[s] + h.t[heap[s]].w, heap[s]});
 95    std::vector<T> res = {d[s]};
 96
 97    for (int i = 1; i < k && !q.empty(); i++) {
 98      auto [w, o] = q.top();
 99      q.pop();
100
101      res.push_back(w);
102
103      int j = h.t[o].v;
104      if (heap[j]) q.push({w + h.t[heap[j]].w, heap[j]});
105      for (auto s : {h.t[o].ls, h.t[o].rs}) {
106        if (s) q.push({w + h.t[s].w - h.t[o].w, s});
107      }
108    }
109    res.resize(k, -1);
110    return res;
111  }
112
113  int a[N];
114  void solve() {
115    int n, k;
116    std::cin >> n >> k;
117
118    std::vector<Edge> e;
119    for (int i = 1; i <= n; i++) {
120      std::cin >> a[i];
121    }
122    e.push_back({0, 1, a[1]});
123    for (int i = 2; i <= n; i++) {
124      if (i - 3 > 0) e.push_back({i - 3, i, a[i]});
125      e.push_back({i - 2, i, a[i]});
126      e.push_back({i - 1, i, a[i]});
127    }
128    if (n - 1 >= 1) e.push_back({n - 1, n + 1, 0});
129    e.push_back({n, n + 1, 0});
130
131    auto res = kShortestPath(n + 2, k, 0, n + 1, e);
132
133    for (auto x : res) std::cout << x << "\n";
134  }
135
```

```
136  int main() {
137    // freopen("t.in", "r", stdin);
138
139    std::ios::sync_with_stdio(false);
140    std::cin.tie(nullptr);
141
142    int t = 1;
143
144    // std::cin >> t;
145
146    while (t--) {
147      solve();
148    }
149    return 0;
150  }
```

Maxflow 只算值版本

```
1   struct dinic{
2       struct E{
3           int to,cap,inv;
4       };
5       vector <E> g[N];
6       int dis[N],now[N];
7       void addedge(int u,int v,int w){
8           g[u].push_back({v,w,(int)g[v].size()});
9           g[v].push_back({u,0,(int)g[u].size()-1});
10      }
11      void bfs(int st){
12          queue<int>q;
13          memset(dis,0,sizeof dis);
14          q.push(st);dis[st]=1;
15          while(q.size()){
16              int u=q.front();q.pop();
17              for(auto &[v,w,inv]:g[u]){
18                  if(w&&!dis[v]){
19                      dis[v]=dis[u]+1;
20                      q.push(v);
21                  }
22              }
23          }
24      }
25      int dfs(int u,int t,int flow){
26          if(u==t)return flow;
27          for(int &i=now[u],sz=g[u].size(),d;i<sz;i++){
28              auto &[v,w,inv]=g[u][i];
29              if(w&&dis[v]>dis[u]){
30                  d=dfs(v,t,min(flow,w));
31                  if(d>0){
32                      w-=d;
33                      g[v][inv].cap+=d;
34                      return d;
35                  }
36              }
37          }
38          return 0;
39      }
```

```
40      int maxflow(int st,int ed){
41          for(int flow=0,res;;){
42              bfs(st);
43              if(!dis[ed])return flow;
44              memset(now,0,sizeof now);
45              while((res=dfs(st,ed,inf))>0){
46                  flow+=res;
47              }
48          }
49      }
50  };
```

Maxflow 网络流最大流

```
1   // 用 givest 定源点汇点
2   // addedge 一次加了正反两条边
3   // init 慎用
4   // S 必须是 0
5   // 输出方案注意是 head 开头
6   #include <bits/stdc++.h>
7   using namespace std;
8   const int N=2510,M=2510*10;
9   class Maxflow{
10  private:
11   int nedge=1,p[2*M],nex[2*M],head[N],c[2*M],cur[2*M];
12      int dist[2*N];
13      int S,T;
14   void Addedge(int a,int b,int v){
15     p[++nedge]=b;nex[nedge]=head[a];head[a]=nedge;
16     c[nedge]=v;
17   }
18      bool bfs(){
19       queue<int>q;
20      for(int i=S;i<=T;i++)dist[i]=-1;
21      dist[S]=0;q.push(S);
22      while(!q.empty()){
23        int now=q.front();q.pop();
24        for(int k=head[now];k;k=nex[k])if(dist[p[k]]==-1&&c[k]>0){
25          dist[p[k]]=dist[now]+1;
26          q.push(p[k]);
27        }
28      }
29      return dist[T]>-1;
30  }
31   int dfs(int x,int low){
32     if(x==T)return low;
33     if(low==0)return 0;
34     int used=0;
35     for(int &k=cur[x];k;k=nex[k])if(dist[p[k]]==dist[x]+1&&c[k]>0){
36       int a=dfs(p[k],min(c[k],low-used));
37       c[k]-=a;c[k^1]+=a;used+=a;
38       if(low==used)break;
39     }
40     if(used==0)dist[x]=-1;
41     return used;
42  }
43  public:
```

```
44    void init(int s,int t){
45      for(int i=S;i<=T;i++)head[i]=0;
46      S=s,T=t;
47     nedge=1;
48    }
49      void addedge(int a,int b,int v){
50          Addedge(a,b,v);
51          Addedge(b,a,0);
52    }
53     int dinic(){
54      int flow=0;
55      while(bfs()){
56        for(int i=S;i<=T;i++)cur[i]=head[i];
57       flow+=dfs(S,1e9);
58      }
59      return flow;
60    }
61   };
```

Tarjan 缩点

```
1  stack<int>s;
2  vector<int>ve[maxn];
3  int col[maxn],num,dfn[maxn],low[maxn],dfstime;
4  void tarjan(int u)
5  {
6      s.push(u);
7      dfn[u]=low[u]=++dfstime;
8      for(auto v:ve[u])
9      {
10         if(!dfn[v])
11         {
12             tarjan(v);
13             low[u]=min(low[u],low[v]);
14         }
15         else if(!col[v]) low[u]=min(low[u],dfn[v]);
16     }
17     if(dfn[u]==low[u])
18     {
19         col[u]=++num;
20         while(s.top()!=u)
21         {
22             col[s.top()]=num;
23             s.pop();
24         }
25         s.pop();
26     }
27 }
```

二分图匹配

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  int const N = 1510 * 4, M = 75010;
4  int e[M], ne[M], h[N], idx;
5  int n, m, k, match[N], backup[N], st[N];
6  void add(int a, int b) {
7      e[idx] = b, ne[idx] = h[a], h[a] = idx++;
```

```
 8  }
 9  int find(int x) {
10      for (int i = h[x]; ~i; i = ne[i]) {
11          int j = e[i];
12          if (!st[j]) {
13              st[j] = 1;
14              if (!match[j] || find(match[j])) {
15                  match[j] = x;
16                  return 1;
17              }
18          }
19      }
20      return 0;
21  }
22  int main() {
23      cin >> n >> m >> k;
24      memset(h, -1, sizeof h);
25      for (int i = 1, a, b; i <= k; ++i) {
26          scanf("%d%d", &a, &b);
27          add(a, b + n);
28      }
29      int maxMatch = 0;
30      for (int i = 1; i <= n; ++i) {
31          memset(st, 0, sizeof st);
32          if (find(i)) maxMatch++;
33      }
34      cout<<maxMatch<<endl;
35      return 0;
36  }
37
```

二分图最优匹配

```
 1  #include<bits/stdc++.h>
 2  using namespace std;
 3  const int maxn=110;
 4  int n, m;
 5  int a[maxn][maxn];
 6  int lx[maxn], ly[maxn], link[maxn];
 7  bool vx[maxn], vy[maxn];
 8  int dfs(int x)
 9  {
10      if(x==-1)return 0;
11      vx[x] = 1;
12      for (int i = 1; i <= n; i++)
13      {
14          if (!vy[i] && lx[x] + ly[i] == a[x][i])
15          {
16              vy[i] = 1;
17              if (link[i] == -1 || dfs(link[i]))
18              {
19                  link[i] = x;
20                  return 1;
21              }
22          }
23      }
24      return 0;
```

```
25  }
26  bool deal()
27  {
28      memset(ly, 0, sizeof(ly));
29      memset(lx, 0xf7, sizeof(lx));
30      memset(link, -1, sizeof(link));
31      for (int i = 1; i <= n; i++)
32      {
33          for (int j = 1; j <= n; j++)
34              lx[i] = max(lx[i], a[i][j]);
35      }
36      for(int i = 1; i <= n; i++)
37      {
38          while(1)
39          {
40              memset(vx, 0, sizeof(vx));
41              memset(vy, 0, sizeof(vy));
42              if (dfs(i)) break;
43              int delta = 0x7f7f7f7f;
44              for (int j= 1; j <= n; j++)
45              {
46                  if (vx[j] == 1)
47                      for(int k = 1; k <= n; k++)
48                          if (vy[k] == 0) delta = min(delta, lx[j]+ ly[k]- a[j][k]);
49              }
50              if (delta == 0x7f7f7f7f)  return 0;
51              for (int j= 1; j <= n; j++)
52                  if (vx[j] == 1) lx[j] -= delta;
53              for(int k = 1; k <= n; k++)
54                  if (vy[k] == 1) ly[k] += delta;
55          }
56      }
57      return 1;
58  }
59  int main()
60  {
61      if  (deal() == 1) {
62          int ans = 0;
63          for(int i = 1; i <= n; i++)
64          {
65              ans += a[link[i]][i];
66          }
67          cout << ans <<'\n';//取最小就把所有边权取负再跑
68      }
69      return 0;
70  }
```

二分图染色

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const int maxn=1010;
4  int n,m;
5  vector<int>ve[maxn];
6  int col[maxn][maxn],ans[maxn*2];
7  void dfs(int x,int y,int c1,int c2)
8  {
```

```
9        if(col[y][c1]){
10           dfs(y,col[y][c1],c2,c1);
11           col[x][c1]=y;
12           col[y][c1]=x;
13       }
14       else {
15           col[x][c1]=y;
16           col[y][c1]=x;
17           col[y][c2]=0;
18       }
19   }
20   map<pair<int,int>,int>ma;
21   int main()
22   {
23       ios::sync_with_stdio(false);
24       cin.tie(0);
25       cin>>n>>m;
26       int anss=0;
27       for(int i=1;i<=m;i++){
28           int x,y;
29           cin>>x>>y;
30           ve[x].push_back(y);
31           ma[{x,y}]=i;
32           int c1=1,c2=1;
33           while(col[x][c1])c1++;
34           while(col[y][c2])c2++;
35           anss=max({c1,c2,anss});
36           if(c1>c2){
37               swap(x,y);swap(c1,c2);
38           }
39           if(c1==c2){
40               col[x][c1]=y;
41               col[y][c1]=x;
42           }
43           else{
44               dfs(x,y,c1,c2);
45           }
46       }
47       cout<<anss<<"\n";
48       for(int i=1;i<=n;i++){
49           for(int j=1;j<=anss;j++)if(col[i][j])ans[ma[{i,col[i][j]}]]=j;
50       }
51       for(int i=1;i<=m;i++)cout<<ans[i]<<"\n";
52       return 0;
53   }
```

圆方树

```
1        vector<vector<int>> e1(n);
2        int cnt = n;
3
4        int now = 0;
5        vector<int> dfn(n, -1), low(n);
6        vector<int> stk;
7        function<void(int)> tarjan = [&](int u) {
8            stk.push_back(u);
9            dfn[u] = low[u] = now++;
```

```
10          for (auto v : ve[u]) {
11              if (dfn[v] == -1) {
12                  tarjan(v);
13                  low[u] = min(low[u], low[v]);
14                  if (low[v] == dfn[u]) {
15                      e1.push_back({});
16                      int x;
17                      do {
18                          x = stk.back();
19                          stk.pop_back();
20                          e1[cnt].push_back(x);
21                      } while (x != v);
22                      e1[u].push_back(cnt);
23                      ++cnt;
24                  }
25              } else {
26                  low[u] = min(low[u], dfn[v]);
27              }
28          }
29      };
30      tarjan(0);
```

基环树

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const int maxn=2e5+10;
4  /*
5  1 init
6  2 addedge
7  3 Get
8  */
9  struct Graph{
10     vector<int>ve[maxn];
11     int base[maxn],id[maxn];
12     bool Incircle[maxn];
13     vector<int> Circle;
14     int len=0;
15     int dep[maxn],f[21][maxn];
16     int n;
17     void init(int _n){
18         n=_n;
19         for(int i=1;i<=n;i++){
20             for(int j=0;j<21;j++)f[j][i]=0;
21             ve[i].clear();
22             Incircle[i]=false;
23             id[i]=-1;
24             base[i]=i;
25             Circle.clear();
26             len=0;
27             dep[i]=0;
28         }
29     }
30     void addedge(int x,int y){
31         ve[x].push_back(y);
32         ve[y].push_back(x);
33     }
```

```
34    void dfs(int x,int fa)
35    {
36        base[x]=base[fa];
37        dep[x]=dep[fa]+1;
38        for(int i=0;i<=19;i++)
39            f[i+1][x]=f[i][f[i][x]];
40        for(auto it:ve[x])
41        {
42            if(it==fa) continue;
43            f[0][it]=x;
44            dfs(it,x);
45        }
46    }
47    void Get(){
48        vector<int> sta;
49        vector<bool> vis(n+1,false);
50        function<bool(int,int)> dfs2 = [&](int x,int h){
51            vis[x]=true;
52            sta.push_back(x);
53            for(auto it:ve[x])if(it!=h){
54                if(vis[it]){
55                    Circle.push_back(it);
56                    while(!sta.empty()&&sta.back()!=it){
57                        Circle.push_back(sta.back());
58                        sta.pop_back();
59                    }
60                    return true;
61                }
62                else{
63                    if(dfs2(it,x))return true;
64                }
65            }
66            sta.pop_back();
67            return false;
68        };
69        dfs2(1,0);
70        len=(int)Circle.size();
71        for(auto it:Circle)Incircle[it]=true;
72        for(auto it:Circle){
73            for(auto it2:ve[it])if(!Incircle[it2]){
74                f[0][it2]=it;
75                dfs(it2,it);
76            }
77        }
78        for(int i=0;i<len;i++)id[Circle[i]]=i;
79    }
80    int lca(int x,int y)
81    {
82        if(dep[x]<dep[y]) swap(x,y);
83        for(int i=20;i>=0;i--)
84        {
85            if(dep[f[i][x]]>=dep[y]) x=f[i][x];
86            if(x==y) return x;
87        }
88        for(int i=20;i>=0;i--)
89            if(f[i][x]!=f[i][y])
```

```
90                    x=f[i][x],y=f[i][y];
91            return f[0][x];
92        }
93    int dis(int x,int y){
94        if(base[x]==base[y]){
95            int l=lca(x,y);
96            return dep[x]+dep[y]-2*dep[l];
97        }
98        else{
99            int g=(id[base[x]]-id[base[y]]+len)%len;
100           return dep[x]+dep[y]+min(g,len-g);
101       }
102   }
103 };
104 Graph g;
```

带权并查集 dsu

```
1  const int maxn=1e5+10;
2  int f[maxn],dis[maxn];
3  int getf(int x){
4      if(x==f[x])return x;
5      int z=getf(f[x]);
6      dis[x]+=dis[f[x]];
7      return f[x]=z;
8  }
9  void unit(int i,int j,int len){
10     int x=getf(i),y=getf(j);
11     f[x]=y;
12     //在赋值之前因为 x 是头节点所以 dis 一定等于 0
13     dis[x]=dis[j]-dis[i]+len;
14 }
```

带花树

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  struct blossom {//0base
4      int n, vis_t;
5      vector<vector<int>> E;
6      vector<int> match, label, org, vis, parent;
7      queue<int> Q;
8      blossom(int _n) {
9          n = _n;
10         E = vector<vector<int>>(n, vector<int>());
11         match.assign(n, -1);
12         label.resize(n);
13         org.resize(n);
14         iota(org.begin(), org.end(), 0);
15         parent.assign(n, -1);
16         vis.assign(n, 0);
17         vis_t = 0;
18     }
19     void addEdge(int u, int v) {
20         E[u].emplace_back(v);
21         E[v].emplace_back(u);
22     }
23     auto lca(int v, int u) {
```

```
24          vis_t++;
25          while (true) {
26              if (v != -1) {
27                  if (vis[v] == vis_t) {
28                      return v;
29                  }
30                  vis[v] = vis_t;
31                  if (match[v] == -1) {
32                      v = -1;
33                  } else {
34                      v = org[parent[match[v]]];
35                  }
36              }
37              swap(v, u);
38          }
39      }
40      void agument(int v) {
41          while (v != -1) {
42              auto pv = parent[v];
43              auto nxt = match[pv];
44              match[v] = pv;
45              match[pv] = v;
46              v = nxt;
47          }
48      }
49      void flower(int v, int u, int a) {
50          while (org[v] != a) {
51              parent[v] = u;
52              u = match[v];
53              if (label[u] == 1) {
54                  label[u] = 0;
55                  Q.emplace(u);
56              }
57              org[v] = org[u] = a;
58              v = parent[u];
59          }
60      }
61      auto bfs(int root) {
62          fill(label.begin(), label.end(), -1);
63          iota(org.begin(), org.end(), 0);
64          while (!Q.empty()) {
65              Q.pop();
66          }
67          Q.emplace(root);
68          label[root] = 0;
69          while (!Q.empty()) {
70              auto u = Q.front();
71              Q.pop();
72              for (auto v : E[u]) {
73                  if (label[v] == -1) {
74                      label[v] = 1;
75                      parent[v] = u;
76                      if (match[v] == -1) {
77                          agument(v);
78                          return true;
79                      }
```

```
80              label[match[v]] = 0;
81              Q.push(match[v]);
82              continue;
83          } else if (label[v] == 0 && org[v] != org[u]) {
84              auto a = lca(org[u], org[v]);
85              flower(v, u, a);
86              flower(u, v, a);
87          }
88      }
89   }
90   return false;
91   }
92   void solve() {
93       for (int i = 0; i < n; ++i) {
94           if (match[i] == -1) {
95               bfs(i);
96           }
97       }
98   }
99 };
100 int main()
101 {
102     blossom G(n);
103     for (int i = 0; i < n; ++i) {
104         for (int j = i + 1; j < n; ++j) {
105             auto [xi, yi] = stone[i];
106             auto [xj, yj] = stone[j];
107             if (abs(xi - xj) + abs(yi - yj) <= L) {
108                 G.addEdge(i, j);
109             }
110         }
111     }
112     G.solve();
113     int num = 0;
114     for (int i = 0; i < n; ++i) {
115         if (G.match[i] != -1) {
116             num++;
117         }
118     }
119 }
```

带负环最小费用最大流

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const int N = 200 + 5, M = 2e4 + N;
4  struct flow {
5    int cnt = 1, hd[N], nxt[M << 1], to[M << 1], limit[M << 1], cst[M << 1];
6    void add(int u, int v, int w, int c) {
7      nxt[++cnt] = hd[u], hd[u] = cnt, to[cnt] = v, limit[cnt] = w, cst[cnt] = c;
8      nxt[++cnt] = hd[v], hd[v] = cnt, to[cnt] = u, limit[cnt] = 0, cst[cnt] = -c;
9    }
10   int fl[N], fr[N], dis[N], in[N];
11   pair<int, int> mincost(int s, int t) {
12     int flow = 0, cost = 0;
13     while (1) {
14       queue<int> q;
```

```cpp
      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;
int main() {
  cin >> n >> m >> s >> t;
```

```
71    for (int i = 1; i <= m; i++) {
72      int u, v, w, c;
73      cin >> u >> v >> w >> c, f.add(u, v, w, c);
74    }
75    pair<int, int> res = f.mincost(n, s, t, 0, n + 1);
76    cout << res.first << " " << res.second << endl;
77    return 0;
78 }
79
```

支配树

```
1  /*
2  1base
3  注意 up 是数组需要外界导入
4  使用的时候直接 dtree:: 即可
5  */
6  namespace dtree{
7    const int MAXN = 200020;
8   vector<int> E[MAXN], RE[MAXN], rdom[MAXN];
9
10   int S[MAXN], RS[MAXN], cs;
11  int par[MAXN], val[MAXN], sdom[MAXN], rp[MAXN], dom[MAXN];
12
13   void clear(int n) {
14     cs = 0;
15     for(int i=0;i<=n;i++) {
16       par[i] = val[i] = sdom[i] = rp[i] = dom[i] = S[i] = RS[i] = 0;
17      E[i].clear(); RE[i].clear(); rdom[i].clear();
18     }
19   }
20   void add_edge(int x, int y) { E[x].push_back(y); }
21  void Union(int x, int y) { par[x] = y; }
22  int Find(int x, int c = 0) {
23    if(par[x] == x) return c ? -1 : x;
24    int p = Find(par[x], 1);
25    if(p == -1) return c ? par[x] : val[x];
26     if(sdom[val[x]] > sdom[val[par[x]]]) val[x] = val[par[x]];
27    par[x] = p;
28     return c ? p : val[x];
29  }
30   void dfs(int x) {
31     RS[ S[x] = ++cs ] = x;
32    par[cs] = sdom[cs] = val[cs] = cs;
33    for(int e : E[x]) {
34       if(S[e] == 0) dfs(e), rp[S[e]] = S[x];
35      RE[S[e]].push_back(S[x]);
36     }
37   }
38   int solve(int s, int *up) {//s 是起点
39     dfs(s);
40     for(int i=cs;i;i--) {
41       for(int e : RE[i]) sdom[i] = min(sdom[i], sdom[Find(e)]);
42       if(i > 1) rdom[sdom[i]].push_back(i);
43       for(int e : rdom[i]) {
44         int p = Find(e);
45         if(sdom[p] == i) dom[e] = i;
```

45

```
46          else dom[e] = p;
47        }
48        if(i > 1) Union(i, rp[i]);
49      }
50      for(int i=2;i<=cs;i++) if(sdom[i] != dom[i]) dom[i] = dom[dom[i]];
51      for(int i=2;i<=cs;i++) up[RS[i]] = RS[dom[i]];
52      return cs;
53    }
54  }
```

最小环

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   const int maxn=1e2+10;
4   const int inf=2e7+10;
5   int a[maxn][maxn],b[maxn][maxn];
6   int main()
7   {
8       int n;cin>>n;
9       int m;cin>>m;
10      for(int i=1;i<=n;i++){
11          for(int j=1;j<=n;j++)a[i][j]=b[i][j]=inf;
12          a[i][i]=b[i][i]=0;
13      }
14      while(m--){
15          int x,y;cin>>x>>y;
16          int w;cin>>w;
17          a[x][y]=min(a[x][y],w);
18          a[y][x]=min(a[y][x],w);
19          b[x][y]=min(b[x][y],w);
20          b[y][x]=min(b[y][x],w);
21      }
22      int ans=inf;
23      for(int i=1;i<=n;i++){
24          for(int j=1;j<i;j++){
25              for(int k=j+1;k<i;k++){
26                  ans=min(ans,a[i][j]+a[i][k]+b[j][k]);
27              }
28          }
29          for(int j=1;j<=n;j++){
30              for(int k=1;k<=n;k++)b[j][k]=min(b[j][i]+b[i][k],b[j][k]);
31          }
32      }
33      if(ans==inf)cout<<"No solution.";
34      else cout<<ans;
35  }
```

最小费用最大流

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   const int N=1e4,M=1e6;
4   struct SSP {
5       int cnt = 1, hd[N], nxt[M << 1], to[M << 1], limit[M << 1], cst[M << 1];
6       void init(){
7           memset(hd,0,sizeof(hd));
8           cnt=1;
```

```
 9          }
10          // w limit c cost
11          void add(int u, int v, int w, int c) {
12              nxt[++cnt] = hd[u], hd[u] = cnt, to[cnt] = v, limit[cnt] = w, cst[cnt] = c;
13              nxt[++cnt] = hd[v], hd[v] = cnt, to[cnt] = u, limit[cnt] = 0, cst[cnt] = -c;
14          }
15
16          int fr[N], fl[N], in[N], dis[N];
17
18          pair<int, int> min_cost(int s, int t) {
19              int flow = 0, cost = 0;
20              while (true) { // SPFA
21                  queue<int> q;
22                  memset(dis, 0x3f, sizeof(dis));
23                  memset(in, 0, sizeof(in));
24                  fl[s] = 1e9, dis[s] = 0, q.push(s);
25                  while (!q.empty()) {
26                      int cur = q.front();
27                      q.pop(), in[cur] = 0;
28                      for (int i = hd[cur]; i; i = nxt[i]) {
29                          int it = to[i], d = dis[cur] + cst[i];
30                          if (limit[i] && d < dis[it]) {
31                              fl[it] = min(limit[i], fl[cur]), fr[it] = i, dis[it] = d;
32                              if (!in[it]) in[it] = 1, q.push(it);
33                          }
34                      }
35                  }
36                  if (dis[t] > 1e9) return {flow, cost};//改成>0 就是可行流
37                  flow += fl[t], cost += dis[t] * fl[t];
38                  for (int u = t; u != s; u = to[fr[u] ^ 1]) limit[fr[u]] -= fl[t],
     limit[fr[u] ^ 1] += fl[t];
39              }
40          }
41 } Sol;
```

欧拉回路

```
 1 #include<bits/stdc++.h>
 2 using namespace std;
 3 const int M=2333,N=666;
 4 struct edge
 5 {
 6  int nxt,to;
 7 }e[M<<1];
 8 int head[N],tot=1;
 9 int cut[M<<1];
10 void add(int u,int v)
11 {
12   e[++tot]=(edge){head[u],v},head[u]=tot;
13   e[++tot]=(edge){head[v],u},head[v]=tot;
14 }
15 vector<int> st;
16 void dfs(int u)//欧拉回路
17 {
18     for(int i=head[u];i!=0;i=e[i].nxt)
19     {
20         if(cut[i]) continue;
```

```
21          int v=e[i].to;
22          cut[i]=cut[i^1]=1;
23          dfs(v);
24      }
25      st.push(u);
26  }
27  int main()
28  {
29
30
31      return 0;
32  }
33
```

三四元环计数

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   const int mod=1e9+7;
4   void add(int &x,int y){
5       if((x+=y)>=mod)x-=mod;
6   }
7   void solve(){
8       int ans3=0,ans4=0;
9       int n,m;cin>>n>>m;
10      vector<pair<int,int>> edges(m);
11      vector<int>d(n+1),flg(n+1),id(n+1),rk(n+1),c(n+1);
12      vector<vector<int>> ve(n+1),f(n+1),g(n+1);
13      for(int i=0;i<m;i++){
14          int u,v;cin>>u>>v;
15          edges[i]=make_pair(u,v);
16          ve[u].push_back(v);
17          ve[v].push_back(u);
18      }
19      for(int i=1;i<=n;i++)
20          d[id[i]=i]=(int)ve[i].size();
21      for(auto [u,v]:edges){
22          if(d[u]>d[v]||(d[u]==d[v]&&u>v))g[u].push_back(v);
23          else g[v].push_back(u);
24      }
25      sort(id.begin()+1,id.end(),[&](int x,int y){
26          return d[x]<d[y];
27      });
28      for(int i=1;i<=n;i++)rk[id[i]]=i;
29      for(int u=1;u<=n;u++)for(auto v:ve[u])
30          if(rk[v]>rk[u])f[u].push_back(v);
31      for(int u=1,C=0;u<=n;u++){
32          for(auto v:ve[u])for(auto w:f[v])if(rk[w]>rk[u])add(ans4,c[w]),++c[w];
33          for(auto v:ve[u])for(auto w:f[v])if(rk[w]>rk[u])c[w]=0;
34          ++C;
35          for(auto v:g[u])flg[v]=C;
36          for(auto v:g[u])
37              for(auto w:g[v])add(ans3,(flg[w]==C));
38      }
39      cout<<ans3<<" "<<ans4<<endl;
40  }
41  /*
```

```
42  5 7
43  1 2
44  2 3
45  4 2
46  5 1
47  1 4
48  3 5
49  3 1
50
51  ans3=3 ans4=2
52  */
```

数据结构

SegmentSet

```
1  /*
2  every pair of pair<int,int> will not intersect
3  if i is true then it will be in the set
4  */
5  struct SegmentSet{
6      set<pair<int,int>>s;
7      void write(){
8          for(auto [x,y]:s)cout<<x<<"/"<<y<<" ";;cout<<"\n";
9      }
10     //make l...r to be true
11     void insert(int l,int r)
12     {
13         int L=l,R=r;
14         auto it=s.lower_bound(make_pair(L,(int)-2e9));
15         while(it!=s.end()&&it->first<=R+1){
16             R=max(it->second,R);
17             it=s.erase(it);
18         }
19         if(it!=s.begin()){
20             it--;
21             if(it->second+1>=L){
22                 L=min(L,it->first);
23                 R=max(R,it->second);
24                 s.erase(it);
25             }
26         }
27         s.insert(make_pair(L,R));
28     }
29     // if l...r all true return false
30     bool query_no_full(int l,int r){
31         auto it=s.lower_bound(make_pair(l,(int)-2e9));
32         if(it!=s.end()){
33             if(l==it->first&&r<=it->second)return false;
34         }
35         if(it!=s.begin()){
36             it--;
37             if(it->second>=r)return false;
38         }
39         return true;
40     }
41     //make l...r to be false
42     void del(int l,int r){
43         auto it=s.lower_bound(make_pair(l,(int)-2e9));
44         while(it!=s.end()&&it->first<=r){
45             if(it->second<=r){
46                 it=s.erase(it);continue;
47             }
48             int R=it->second;
49             s.erase(it);
50             s.insert(make_pair(r+1,R));
51             break;
52         }
53         it=s.lower_bound(make_pair(l,(int)-2e9));
```

```
54          if(it!=s.begin()){
55              it--;
56              int L=it->first,R=it->second;
57              if(R>=l){
58                  s.erase(it);
59                  if(L<=l-1)s.insert(make_pair(L,l-1));
60                  if(R>=r+1)s.insert(make_pair(r+1,R));
61              }
62          }
63      }
64      // if l...r all false return false
65      bool query_at_least_one(int l,int r){
66          auto it=s.lower_bound(make_pair(l,(int)-2e9));
67          if(it!=s.end()){
68              if(it->first<=r)return true;
69          }
70          if(it!=s.begin()){
71              it--;
72              if(it->second>=l)return true;
73          }
74          return false;
75      }
76  };
```

SegmentTree

```
1  //?????????? ???????>=??sum??????
2  #include <bits/stdc++.h>
3  using namespace std;
4  const int maxn=1e6+10;
5  struct Node{
6      int l,r,res,tag;
7  };
8  struct SegmentTree{
9      Node a[maxn*4];
10     void tag_init(int i){
11         a[i].tag=0;
12     }
13     void tag_union(int fa,int i){
14         a[i].tag+=a[fa].tag;
15     }
16     void tag_cal(int i){
17         a[i].res+=a[i].tag*(a[i].r-a[i].l+1);
18     }
19     void pushdown(int i){
20         tag_cal(i);
21         if(a[i].l!=a[i].r){
22             tag_union(i,i*2);
23             tag_union(i,i*2+1);
24         }
25         tag_init(i);
26     }
27     void pushup(int i){
28         if(a[i].l==a[i].r)return;
29         pushdown(i*2);
30         pushdown(i*2+1);
31         a[i].res=a[i*2].res+a[i*2+1].res;
```

```
32          }
33          void build(int i,int l,int r){
34              a[i].l=l,a[i].r=r;tag_init(i);
35              if(l>=r)return;
36              int mid=(l+r)/2;
37              build(i*2,l,mid);
38              build(i*2+1,mid+1,r);
39          }
40          void update(int i,int l,int r,int w){
41              pushdown(i);
42              if(a[i].r<l||a[i].l>r||l>r)return;
43              if(a[i].l>=l&&a[i].r<=r){
44                  a[i].tag=w;
45                  return;
46              }
47              update(i*2,l,r,w);
48              update(i*2+1,l,r,w);
49              pushup(i);
50          }
51          int query(int i,int l,int r){
52              pushdown(i);
53              if(a[i].r<l||a[i].l>r||l>r)return 0;
54              if(a[i].l>=l&&a[i].r<=r){
55                  return a[i].res;
56              }
57              return query(i*2,l,r)+query(i*2+1,l,r);
58          }
59          int min_right(int qL, int& nowsum,int querysum, int i) {//???????>=sum???
60              pushdown(i);
61              if (a[i].r < qL)return -1;
62              if (qL <= a[i].l) {
63                  int ss = nowsum+a[i].res;
64                  if (ss<querysum) {
65                      nowsum = ss;
66                      return -1;
67                  }
68                  if (a[i].l == a[i].r)return a[i].l;
69              }
70              int pos = min_right(qL, nowsum,querysum,i*2);
71              if (pos != -1)return pos;
72              return min_right(qL, nowsum,querysum,2*i+1);
73          }
74          int max_left(int qR,int &nowsum,int querysum,int i){//???????>=sum???
75              pushdown(i);
76              if(a[i].l > qR)return -1;
77              if(qR>=a[i].r){
78                  int ss=nowsum+a[i].res;
79                  if(ss<querysum){
80                      nowsum=ss;
81                      return -1;
82                  }
83                  if(a[i].l==a[i].r)return a[i].r;
84              }
85              int pos=max_left(qR,nowsum,querysum,i*2+1);
86              if(pos!=-1)return pos;
87              return max_left(qR,nowsum,querysum,i*2);
```

```
88          }
89   };
90   SegmentTree tri;
91   int main()
92   {
93        ios::sync_with_stdio(false);
94        cin.tie(0);
95        int n,q;cin>>n>>q;
96        tri.build(1,1,n);
97        for(int i=1;i<=n;i++){
98            int x;cin>>x;tri.update(1,i,i,x);
99        }
100       while(q--){
101           int ops;cin>>ops;
102           if(ops==1){
103               int l,r,x;cin>>l>>r>>x;
104               tri.update(1,l,r,x);
105           }
106           if(ops==2){
107               int l,r;cin>>l>>r;
108               cout<<tri.query(1,l,r)<<"\n";
109           }
110           if(ops==3){
111               int x,sum;cin>>x>>sum;
112               int nowsum=0;
113               cout<<tri.min_right(x,nowsum,sum,1)<<"\n";
114           }
115           if(ops==4){
116               int x,sum;cin>>x>>sum;
117               int nowsum=0;
118               cout<<tri.max_left(x,nowsum,sum,1)<<"\n";
119           }
120       }
121  }
122
```

三维偏序 cdq

```
1    #include <bits/stdc++.h>
2    using namespace std;
3    const int maxn=4e5+10;
4    struct Treearray{
5        int c[maxn],n;
6        void set_n(int _n){
7            n=_n;
8            for(int i=1;i<=n;i++)c[i]=0;
9        }
10       int lowbit(int x){
11           return x&(-x);
12       }
13       void update(int i,int k){     //在 i 位置加上 k
14           while(i <= n){
15               c[i] += k;
16               i += lowbit(i);
17           }
18       }
19       int getsum(int i){          //求 A[1 - i]的和
```

```
20        int res = 0;
21        while(i > 0){
22            res += c[i];
23            i -= lowbit(i);
24        }
25        return res;
26    }
27 };
28 Treearray tri;
29 int a[maxn],b[maxn],c[maxn];
30 int n,k;
31 int ans[maxn],id[maxn],cnt[maxn],t[maxn];
32 void subdiv(int l,int r){
33    if(l==r)return;
34    int mid=(l+r)/2;
35    subdiv(l,mid);
36    subdiv(mid+1,r);
37    sort(id+l,id+mid+1,[&](int x,int y){
38        if(b[x]==b[y])return c[x]<c[y];
39        return b[x]<b[y];
40    });
41    sort(id+mid+1,id+r+1,[&](int x,int y){
42        if(b[x]==b[y])return c[x]<c[y];
43        return b[x]<b[y];
44    });
45
46    // for(int i=l;i<=r;i++)cout<<id[i]<<" \n"[i==r];
47    assert(tri.getsum(tri.n)==0);
48    for(int i=l,j=mid+1;i<=mid||j<=r;){
49        if(j!=r+1&&(i==mid+1||b[id[i]]>b[id[j]])){
50            // cout<<"id="<<id[j]<<" +"<<tri.getsum(c[id[j]])<<"\n";
51            ans[id[j]]+=tri.getsum(c[id[j]]);
52            j++;
53        }
54        else{
55            tri.update(c[id[i]],t[id[i]]);
56            i++;
57        }
58    }
59    for(int i=l;i<=mid;i++)tri.update(c[id[i]],-t[id[i]]);
60 }
61 int main()
62 {
63    ios::sync_with_stdio(false);
64    cin.tie(0);
65    cin>>n>>k;
66    tri.set_n(200000);
67    vector<tuple<int,int,int>> v(n);
68    for(int i=0;i<n;i++){
69        int x,y,z;cin>>x>>y>>z;v[i]=make_tuple(x,y,z);
70    }
71    sort(v.begin(),v.end());
72    int g=0;
73    for(int i=0;i<n;i++){
74        int j=i;
75        while(j+1<n&&v[i]==v[j+1])j++;
```

```
76          g++;
77          a[g]=get<0>(v[i]);
78          b[g]=get<1>(v[i]);
79          c[g]=get<2>(v[i]);
80          t[g]=j-i+1;
81          i=j;
82          // cout<<"g="<<g<<" "<<a[g]<<" "<<b[g]<<" "<<c[g]<<"\n";
83       }
84       for(int i=1;i<=g;i++)id[i]=i;
85       subdiv(1,g);
86       for(int i=1;i<=g;i++){
87           cnt[ans[i]+t[i]-1]+=t[i];
88           // cout<<"i="<<i<<" ans="<<ans[i]<<"\n";
89       }
90       for(int i=0;i<n;i++)cout<<cnt[i]<<"\n";
91       return 0;
92   }
93
```

主席树

```
1   //注意 Sum 和 cnt 的区别
2   #include <bits/stdc++.h>
3   using namespace std;
4   const int maxn = 1e5; // 数据范围
5   int n, m;
6   struct Persistent_SegmentTree
7   {
8       int sum[(maxn << 5) + 10], root[maxn + 10], lch[(maxn << 5) + 10],
9           rch[(maxn << 5) + 10], cnt[(maxn<<5)+10];
10      int tot = 0;
11      void init()
12      {
13          tot = 0;
14      }
15      int a[maxn + 10];
16      void update(int &rot, int pr, int L, int R, int k)
17      { // 插入操作
18          rot = ++tot;
19          lch[rot] = lch[pr];
20          rch[rot] = rch[pr];
21          sum[rot] = sum[pr] + k;
22          cnt[rot] = cnt[pr] + 1;
23          if (L == R)
24              return;
25          int mid = (L + R) >> 1;
26          if (k <= mid)
27              update(lch[rot], lch[pr],  L, mid,k);
28          else
29              update(rch[rot], rch[pr],  mid + 1, R,k);
30      }
31      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
32      {
33          if (l <= L && R <= r)
34              return cnt[t] - cnt[s];
35          int res = 0;
```

```
36          int mid = (L + R) >> 1;
37          if (l <= mid)
38              res += getcnt(lch[s], lch[t], L, mid, l, r);
39          if (r > mid)
40              res += getcnt(rch[s], rch[t], mid + 1, R, l, r);
41          return res;
42      }
43
44      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
45      {
46          if (l <= L && R <= r)
47              return sum[t] - sum[s];
48          int res = 0;
49          int mid = (L + R) >> 1;
50          if (l <= mid)
51              res += getsum(lch[s], lch[t], L, mid, l, r);
52          if (r > mid)
53              res += getsum(rch[s], rch[t], mid + 1, R, l, r);
54          return res;
55      }
56      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
57          int ss = nowsum+cnt[t]-cnt[s];
58          if (ss<k) {
59              nowsum = ss;
60              ans+=sum[t]-sum[s];
61              return -1;
62          }
63          if (l == r){
64              ans+=(k-nowsum)*l;
65              return l;
66          }
67          int mid=(l+r)/2;
68          int pos = get_Kth_min_Sum(lch[s],lch[t],l,mid,k,nowsum,ans);
69          if (pos != -1)return pos;
70          return get_Kth_min_Sum(rch[s],rch[t],mid+1,r,k,nowsum,ans);
71      }
72      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
73          int ss = nowsum+cnt[t]-cnt[s];
74          if (ss<k) {
75              nowsum = ss;
76              ans+=sum[t]-sum[s];
77              return -1;
78          }
79          if (l == r){
80              ans+=(k-nowsum)*l;
81              return l;
82          }
83          int mid=(l+r)/2;
84          int pos = get_Kth_max_Sum(rch[s],rch[t],mid+1,r,k,nowsum,ans);
85          if (pos != -1)return pos;
86          return get_Kth_max_Sum(lch[s],lch[t],l,mid,k,nowsum,ans);
87      }
88      int get_upper(int s,int t,int l,int r,int x){//第一个大于等于 x 的数
```

```cpp
 89              if (r < x)return -1;
 90              if (x <= l) {
 91                  int ss = cnt[t]-cnt[s];
 92                  if (ss==0) {
 93                      return -1;
 94                  }
 95                  if (l == r)return l;
 96              }
 97              int mid=(l+r)/2;
 98              int pos = get_upper(lch[s],lch[t],l,mid,x);
 99              if (pos != -1)return pos;
100              return get_upper(rch[s],rch[t],mid+1,r,x);
101          }
102          int get_lower(int s,int t,int l,int r,int x){//第一个小于等于 x 的数
103              if(l > x)return -1;
104              if(x>=r){
105                  int ss=cnt[t]-cnt[s];
106                  if(ss==0){
107                      return -1;
108                  }
109                  if(l==r)return r;
110              }
111              int mid=(l+r)/2;
112              int pos=get_lower(rch[s],rch[t],mid+1,r,x);
113              if(pos!=-1)return pos;
114              return get_lower(lch[s],lch[t],l,mid,x);
115          }
116      };
117      Persistent_SegmentTree tri;
118      int main()
119      {
120          int n,q;
121          cin >> n >> q;
122
123          for (int i = 1; i <= n; i++)
124          {
125              int x;
126              cin >> x;
127              tri.update(tri.root[i], tri.root[i - 1], 1, n, x);
128              assert(1<=x&&x<=n);
129          }
130          while (q--)
131          {
132              int ops;cin>>ops;
133              int l,r,k,L,R;
134              if(ops==1){
135                  cin>>l>>r>>L>>R;
136                  cout<<tri.getsum(tri.root[l-1],tri.root[r],1,n,L,R)<<"\n";
137              }
138              if(ops==2){// get kth max no output -1
139                  cin>>l>>r>>k;
140                  int nowsum=0,ans=0;
141                  cout<<tri.get_Kth_max_Sum(tri.root[l-1],tri.root[r],1,n,k,nowsum,ans)<<"\n";
142              }
143              if(ops==3){// get kth max sum siz<k return allsum
144                  cin>>l>>r>>k;
```

```
145            int nowsum=0,ans=0;
146            tri.get_Kth_max_Sum(tri.root[l-1],tri.root[r],1,n,k,nowsum,ans);
147            cout<<ans<<"\n";
148        }
149        if(ops==4){// get kth min no output -1
150            cin>>l>>r>>k;
151            int nowsum=0,ans=0;
152            cout<<tri.get_Kth_min_Sum(tri.root[l-1],tri.root[r],1,n,k,nowsum,ans)<<"\n";
153        }
154        if(ops==5){// get kth min sum siz<k return allsum
155            cin>>l>>r>>k;
156            int nowsum=0,ans=0;
157            tri.get_Kth_min_Sum(tri.root[l-1],tri.root[r],1,n,k,nowsum,ans);
158            cout<<ans<<"\n";
159        }
160        if(ops==6){// get the min element >= k no return -1
161            cin>>l>>r>>k;
162            cout<<tri.get_upper(tri.root[l-1],tri.root[r],1,n,k)<<"\n";
163        }
164        if(ops==7){// get the max element <= k no return -1
165            cin>>l>>r>>k;
166            cout<<tri.get_lower(tri.root[l-1],tri.root[r],1,n,k)<<"\n";
167        }
168    }
169 }
```

二维树状数组

```
1  const int maxn=1010;
2  struct treearray{
3      int mkp1[maxn][maxn],mkp2[maxn][maxn],mkp3[maxn][maxn],mkp4[maxn][maxn];
4      inline int lowbit(int x)
5      {
6          return x&(-x);
7      }
8      inline void Update(int x,int y,int k)
9      {
10         for(int i=x;i<=n;i+=lowbit(i))
11         {
12             for(int j=y;j<=m;j+=lowbit(j))
13             {
14                 mkp1[i][j]+=k;
15                 mkp2[i][j]+=k*x;
16                 mkp3[i][j]+=k*y;
17                 mkp4[i][j]+=k*x*y;
18             }
19         }
20     }
21     inline void update(int a,int b,int x,int y,int k)
22     {
23         Update(a,b,k);
24         Update(a,y+1,-k);
25         Update(x+1,b,-k);
26         Update(x+1,y+1,k);
27     }
28     inline int Query(int x,int y)
29     {
```

```
30          int ans=0;
31          for(int i=x;i>=1;i-=lowbit(i))
32          {
33              for(int j=y;j>=1;j-=lowbit(j))
34              {
35                  ans+=(x+1)*(y+1)*mkp1[i][j]
36                  -(y+1)*mkp2[i][j]-(x+1)*mkp3[i][j]+mkp4[i][j];
37              }
38          }
39          return ans;
40      }
41      inline int query(int a,int b,int x,int y)
42      {
43          return Query(x,y)+Query(a-1,b-1)-Query(x,b-1)-Query(a-1,y);
44      }
45  };
46  treearray tri;
47  int main()
48  {
49
50  }
51
```

动态开点线段树

```
1   // root 表示整棵线段树的根结点；cnt 表示当前结点个数
2   const int maxn=1e5+10;
3   int n, cnt, root;
4   int sum[maxn*20], ls[maxn*20], rs[maxn*20];
5
6   // 用法：update(root, 1, n, x, f); 其中 x 为待修改节点的编号
7   void update(int& p, int L, int R, int x, int f) {  // 引用传参
8       if (!p) p = ++cnt;  // 当结点为空时，创建一个新的结点
9       if (L == R) {
10          sum[p] += f;
11          return;
12      }
13      int m = L + ((R - L) >> 1);
14      if (x <= m)
15          update(ls[p], L, m, x, f);
16      else
17          update(rs[p], m + 1, R, x, f);
18      sum[p] = sum[ls[p]] + sum[rs[p]];  // pushup
19  }
20  // 用法：query(root, 1, n, l, r);
21  int query(int p, int L, int R, int l, int r) {
22      if (!p) return 0;  // 如果结点为空，返回 0
23      if (L >= l && R <= r) return sum[p];
24      int m = L + ((R - L) >> 1), ans = 0;
25      if (l <= m) ans += query(ls[p], L, m, l, r);
26      if (r > m) ans += query(rs[p], m + 1, R, l, r);
27      return ans;
28  }
29  int merge(int a, int b, int l, int r) {
30      if (!a) return b;
31      if (!b) return a;
32      if (l == r) {
```

```
33          sum[a]+=sum[b];
34          return a;
35      }
36      int mid = (l + r) >> 1;
37      ls[a] = merge(ls[a], ls[b], l, mid);
38      rs[a] = merge(rs[a], rs[b], mid + 1, r);
39      sum[a]=sum[ls[a]]+sum[rs[a]];
40      return a;
41  }
42  void split(int &p, int &q, int L, int R, int l, int r) {//p 原树 q 新树
43      if (R < l || r < L) return;
44      if (!p) return;
45      if (l <= L && R <= r) {
46          q = p;
47          p = 0;
48          return;
49      }
50      if (!q) q = ++cnt;
51      int m = L + R >> 1;
52      if (l <= m) split(ls[p], ls[q], L, m, l, r);
53      if (m < r) split(rs[p], rs[q], m + 1, R, l, r);
54      sum[p]=sum[ls[p]]+sum[rs[p]];
55      sum[q]=sum[ls[q]]+sum[rs[q]];
56  }
57  int query1(int& Sum,int Up, int rt, int l, int r) {//min_x of f(x)+x>Up
58      // cout<<"Sum="<<Sum<<" Up="<<Up<<" rt="<<rt<<" l="<<l<<" r="<<r<<"\n";
59      if(!rt){
60          if(r+Sum<=Up)return -1;
61          else return r-(r+Sum-Up)+1;
62      }
63      if (true) {
64          int ss = Sum+sum[rt];
65          if (ss+r<=Up) {
66              Sum = ss;
67              return -1;
68          }
69          if (l == r)return l;
70      }
71      int mid = (l + r) / 2;
72      int pos = query1(Sum,Up,ls[rt], l, mid);
73      if (pos != -1)return pos;
74      return query1(Sum,Up,rs[rt], mid + 1, r);
75  }
```

可持久化 01trie

```
1  const int maxn=1e5+10;
2  struct Persistent_Trie
3  {
4      int ch[maxn][2],tot,sum[maxn];
5      void clear()
6      {
7          tot=0;
8          return;
9      }
10     void cpy(int from,int to)
11     {
```

```
12          ch[to][0]=ch[from][0];
13          ch[to][1]=ch[from][1];
14          sum[to]=sum[from];
15          return;
16      }
17      void insert(int &root,int old,int num,int nowbit)
18      {
19          root=++tot;
20          cpy(old,root);
21          sum[root]++;
22          if(nowbit==-1)return;
23          if(num&(1<<nowbit))
24          {
25              insert(ch[root][1],ch[root][1],num,nowbit-1);
26          }
27          else
28          {
29              insert(ch[root][0],ch[root][0],num,nowbit-1);
30          }
31      }
32      int query(int s,int t,int x)
33      {
34          int ans=0;
35          for(int i=30;i>=0;--i)
36          {
37              if(sum[ch[t][!(x&(1<<i))]]-sum[ch[s][!(x&(1<<i))]])
38              {
39                  t=ch[t][!(x&(1<<i))];
40                  s=ch[s][!(x&(1<<i))];
41                  ans|=(1<<i);
42              }
43              else
44              {
45                  t=ch[t][!!(x&(1<<i))];
46                  s=ch[s][!!(x&(1<<i))];
47              }
48          }
49          return ans;
50      }
51  };
```

最近公共祖先 LCA

```
1  const int maxn=1e5+10;
2  vector<int>ve[maxn];
3  int dep[maxn],f[21][maxn];
4  void dfs(int x,int fa)
5  {
6    dep[x]=dep[fa]+1;
7    for(int i=0;i<=19;i++)
8     f[i+1][x]=f[i][f[i][x]];
9      for(auto it:ve[x])
10   {
11     if(it==fa) continue;
12   f[0][it]=x;
13     dfs(it,x);
14  }
```

```
15 }
16 int lca(int x,int y)
17 {
18  if(dep[x]<dep[y]) swap(x,y);
19  for(int i=20;i>=0;i--)
20  {
21     if(dep[f[i][x]]>=dep[y]) x=f[i][x];
22     if(x==y) return x;
23  }
24   for(int i=20;i>=0;i--)
25    if(f[i][x]!=f[i][y])
26      x=f[i][x],y=f[i][y];
27  return f[0][x];
28 }
```

Lca(o1)

```
1  #define int long long
2  const int maxn=1e5+10;//注意开两倍大小的空间 在dp上
3  vector<pair<int,int>>ve[maxn];
4  int dep[maxn];
5  pair<int,int>dp[21][maxn*3];
6  int red[maxn],d[maxn];
7  int Dep[maxn];
8  int dfn[maxn];
9  void dfs(int x,int fa,int l,int dis)
10 {
11     if(red[x])dis=0;
12     d[x]=dis;
13  dep[x]=dep[fa]+1;
14   Dep[x]=Dep[fa]+l;
15     for(auto [it,len]:ve[x])
16  {
17     if(it==fa) continue;
18    dfs(it,x,len,dis+len);
19  }
20 }
21 vector<int> sp;
22 void dfs2(int u, int fa)
23 {
24
25     dfn[u] = sp.size();
26     sp.push_back(u);
27     for (auto& e : ve[u])if(e.first!=fa)
28     {
29         int& v = e.first;
30         dfs2(v, u);
31         sp.push_back(u);
32     }
33 }
34 void initrmq()
35 {
36     int n = sp.size();
37     for (int i = 0; i < n; i++) dp[0][i] = {dfn[sp[i]], sp[i]};
38     for (int i = 1; (1 << i) <= n; i++)
39         for (int j = 0; j + (1 << i) - 1 < n; j++)
40             dp[i][j] = min(dp[i - 1][j], dp[i - 1][j + (1 << (i - 1))]);
```

```
41  }
42  int lca(int u, int v)
43  {
44      int l = dfn[u], r = dfn[v];
45      if (l > r) swap(l, r);
46      int k = __lg(r-l+1);
47      return min(dp[k][l], dp[k][r - (1 << k) + 1]).second;
48  }
```

点分治

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   const int maxn = 100010;
4   const int inf =2e9;
5   int n, siz[maxn], dist[maxn], vis[maxn],maxx[maxn];
6   vector<pair<int, int>> ve[maxn];
7
8   void add_edge(int x, int y, int z)
9   {
10      ve[x].emplace_back(y, z);
11  }
12
13  void calcsiz(int x, int fa, int sum, int &rt)
14  {
15      siz[x] = 1;
16      maxx[x] = 0;
17      for (auto &[to, len] : ve[x])
18          if (to != fa && !vis[to])
19          {
20              calcsiz(to, x, sum, rt);
21              maxx[x] = max(maxx[x], siz[to]);
22              siz[x] += siz[to];
23          }
24      maxx[x] = max(maxx[x], sum - siz[x]);
25      if (maxx[x] < maxx[rt])
26          rt = x;
27  }
28
29
30  void calcdist(int x, int fa)
31  {
32      for (auto &[to, len] : ve[x])
33          if (to != fa && !vis[to])
34          {
35              dist[to] = dist[x] + len, calcdist(to, x);
36          }
37  }
38  void dfs(int x, int fa)
39  {
40      vis[x] = true;
41      for (auto &[to, len] : ve[x])
42          if (to != fa && !vis[to])
43          {
44              dist[to] = len;
45              calcdist(to, x);
46          }
```

```
47
48
49    for (auto &[to, len] : ve[x])
50        if (to != fa && !vis[to])
51        {
52            int sum = siz[to];
53            int rt = 0;
54            calcsiz(to, x, sum, rt);
55            calcsiz(rt, -1, sum, rt);
56            dfs(rt, x);
57        }
58 }
59
60 int main()
61 {
62    ios::sync_with_stdio(false);
63    cin.tie(0);
64    cin >> n ;
65    for (int i = 1; i < n; i++)
66    {
67        int a, b,c;
68        cin >> a >> b>>c, add_edge(a, b, c), add_edge(b, a, c);
69    }
70    int rt = 0;
71    maxx[rt] = inf;
72    int sum = n;
73    calcsiz(1, -1, sum, rt);
74    calcsiz(rt, -1, sum, rt);
75    dfs(rt, -1);
76    return 0;
77 }
78
```

珂朵莉树

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  struct SegmentMap{
4      map<int,int> ma;
5      int sum=0;
6      int n;
7      SegmentMap(int _n=0){
8          n=_n;
9          ma[1]=0;
10         ma[n+1]=-1;
11     }
12     void del(int l,int r,int x){//减去当前 (l,r,x) 的贡献
13         sum-=(r-l+1)*x;
14     }
15     void add(int l,int r,int x){//加上当前 (l,r,x) 的贡献
16         sum+=(r-l+1)*x;
17     }
18     void split(int x){
19         auto t=prev(ma.upper_bound(x))->second;
20         ma[x]=t;
21     }
22     void update(int l,int r,auto&& T){
```

```
23              split(l);split(r+1);
24              auto it=prev(ma.upper_bound(l));
25              int pr=-1;
26              while(it->first<=r){
27                  int nowl=it->first,nowr=next(it)->first-1;
28                  int x=it->second;
29                  del(nowl,nowr,x);
30                  add(nowl,nowr,T(x));
31                  if(l!=nowl&&pr==T(x))it=ma.erase(it);
32                  else {
33                      it->second=T(x);
34                      it=next(it);
35                  }
36                  pr=T(x);
37              }
38              if(it->first!=n+1&&it->second==pr)ma.erase(it);
39              if(l!=1){
40                  it=ma.lower_bound(l);
41                  if(prev(it)->second==it->second)ma.erase(it);
42              }
43          }
44  };
45  int main()
46  {
47      ios::sync_with_stdio(false);
48      cin.tie(0);
49      int n,q;cin>>n>>q;
50      SegmentMap sol(n);
51      while(q--){
52          int op;cin>>op;
53          if(op==1){
54              int l,r;cin>>l>>r;
55              sol.update(l,r,[](int x){
56                  return (int)sqrt(x);
57              });
58          }
59          else{
60              int l,r,x;cin>>l>>r>>x;
61              sol.update(l,r,[x](int y){
62                  return x;
63              });
64          }
65          cout<<sol.sum<<"\n";
66      }
67  }
```

笛卡尔树

```
1  const int maxn=2e5+10;
2  vector<int> ve[maxn];
3  int a[maxn];
4  int n;
5  int build(){
6      int top=0;
7      vector<int> Stack(n+1,0);
8      for(int i=0;i<=n;i++){
9          ve[i].clear();
```

```
10          ve[i].resize(2,-1);
11      }
12  Stack[++top]=1;
13   for(int i=2;i<=n;i++){
14    while(top&&a[Stack[top]]>=a[i])top--;
15     if(!top)ve[i][0]=Stack[top+1];
16    else ve[i][0]=ve[Stack[top]][1],ve[Stack[top]][1]=i;
17    Stack[++top]=i;
18   }
19      return Stack[1];
20  }
```

线段树区间加区间历史最小值

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   #define int long long
4   const int maxn=1e6+10;
5   const int inf =2e9;
6   struct Node{
7       int l,r,res,tag,tag2,res2;
8   };
9   struct SegmentTree{
10      Node a[maxn*4];
11      void tag_init(int i){
12          a[i].tag=a[i].tag2=0;
13      }
14      void tag_union(int fa,int i){
15          if(a[fa].tag2<0)a[i].tag2=min(a[i].tag2,a[i].tag+a[fa].tag2);
16          a[i].tag+=a[fa].tag;
17      }
18      void tag_cal(int i){
19          if(a[i].tag2<0)a[i].res2=min(a[i].res2,a[i].res+a[i].tag2);
20          a[i].res+=a[i].tag;
21      }
22      void pushdown(int i){
23          tag_cal(i);
24          if(a[i].l!=a[i].r){
25              tag_union(i,i*2);
26              tag_union(i,i*2+1);
27          }
28          tag_init(i);
29      }
30      void pushup(int i){
31          if(a[i].l==a[i].r)return;
32          pushdown(i*2);
33          pushdown(i*2+1);
34          a[i].res=min(a[i*2].res,a[i*2+1].res);
35          a[i].res2=min(a[i*2].res2,a[i*2+1].res2);
36      }
37      void build(int i,int l,int r){
38          a[i].l=l,a[i].r=r;tag_init(i);
39          if(l>=r)return;
40          int mid=(l+r)/2;
41          build(i*2,l,mid);
42          build(i*2+1,mid+1,r);
43      }
```

```
44      void update(int i,int l,int r,int w){
45          if(a[i].r<l||a[i].l>r||l>r)return;
46          pushdown(i);
47          if(a[i].l>=l&&a[i].r<=r){
48              a[i].tag+=w;
49              a[i].tag2=min(a[i].tag2,a[i].tag);
50              return;
51          }
52          update(i*2,l,r,w);
53          update(i*2+1,l,r,w);
54          pushup(i);
55      }
56      int query(int i,int l,int r){
57          pushdown(i);
58          if(a[i].r<l||a[i].l>r||l>r)return inf;
59          if(a[i].l>=l&&a[i].r<=r){
60              return a[i].res2;
61          }
62          return min(query(i*2,l,r),query(i*2+1,l,r));
63      }
64  };
65  SegmentTree tri;
66  signed main()
67  {
68      int n,m;
69      cin>>n>>m;
70      tri.build(1,1,n);
71      for(int i=1;i<=m;i++){
72          int ops;cin>>ops;
73          if(ops==1){
74              int l,r,x;cin>>l>>r>>x;
75              tri.update(1,l,r,x);
76          }
77          else{
78              int l,r;cin>>l>>r;
79              cout<<tri.query(1,l,r)<<"\n";
80          }
81      }
82      return 0;
83  }
```

莫队

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   const int N=1e6+5;
4   int n,m,a[N],block,ans[N],now,cnt[N];
5   struct node
6   {
7       int l,r,i;
8       friend bool operator<(const node a,const node b)
9       {
10          if((a.l/block)==(b.l/block)) return ((a.l/block%2)?a.r>b.r:a.r<b.r);
11          else return a.l<b.l;
12      }
13  }q[N];
14  void add(int x)
```

```
15  {
16  }
17  void del(int x)
18  {
19  }
20  int main()
21  {
22      block=max(1,n/sqrt(m));
23      sort(q+1,q+1+m);
24      int l=1,r=0;
25      for(int i=1;i<=m;i++)
26      {
27          while(l<q[i].l) del(l++);
28          while(l>q[i].l) add(--l);
29          while(r<q[i].r) add(++r);
30          while(r>q[i].r) del(r--);
31          ans[q[i].i]=now;
32      }
33      for(int i=1;i<=m;i++)
34          printf("%d\n",ans[i]);
35      return 0;
36  }
37
```

虚树

```
1   //带边权的虚树
2   //ve2 是虚树,ve 是原树
3   //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
4   //用的时候就 addintoxs(x)丢进去，每次 init 就全部删除
5   //add 是加原边，ADD 不用管它
6   //solve 里面是必备的几步，不能去除
7   #include <bits/stdc++.h>
8   using namespace std;
9   #define int long long
10  const int maxn=1e5+10;
11  const int inf =1e18;
12  int n,m;
13  struct XS{
14      struct kkk{
15          int to,len;
16      };
17      int dep[maxn],f[maxn][21],id[maxn];
18      int mlen[maxn][21],ans[maxn][21];
19      int state[maxn];
20      int sum[maxn],all;
21      vector<kkk> ve[maxn],ve2[maxn];
22      int now;//统计 dfs 序
23      void dfs(int u,int fa)
24      {
25          id[u]=now++;
26          dep[u]=dep[fa]+1;
27          for(int i=0;i<=19;i++){
28              f[u][i+1]=f[f[u][i]][i];
29              mlen[u][i+1]=mlen[f[u][i]][i]+mlen[u][i];
30          }
31          for(auto v:ve[u])
```

```
32          {
33              if(v.to==fa) continue;
34              f[v.to][0]=u;
35              mlen[v.to][0]=v.len;
36              dfs(v.to,u);
37          }
38      }
39      int lca(int x,int y)
40      {
41          if(dep[x]<dep[y]) swap(x,y);
42          for(int i=20;i>=0;i--)
43          {
44              if(dep[f[x][i]]>=dep[y]) x=f[x][i];
45              if(x==y) return x;
46          }
47          for(int i=20;i>=0;i--)
48              if(f[x][i]!=f[y][i])
49                  x=f[x][i],y=f[y][i];
50          return f[x][0];
51      }
52      int getlen(int x,int y)
53      {
54          if(x==y)return 0;
55          if(dep[x]<dep[y]) swap(x,y);
56          int res=0;
57          for(int i=20;i>=0;i--){
58              if(dep[f[x][i]]>=dep[y]){
59                  res=res+mlen[x][i];
60                  x=f[x][i];
61              }
62              if(x==y)return res;
63          }
64          return inf;
65      }
66      void add(int x,int y,int len){//加边
67          ve[x].push_back({y,len});
68          ve[y].push_back({x,len});
69      }
70      void ADD(int x,int y,int len){//虚树加边
71          ve2[x].push_back({y,len});
72          ve2[y].push_back({x,len});
73      }
74      vector<int> vis;//b vis 这两个用来初始化
75      bool b[maxn];
76      vector<int> v1;//用来表示加入虚树的点
77      int sta[maxn];
78      void init()//删除虚树
79      {
80          for(auto it:vis){
81              ve2[it].clear();
82              b[it]=false;
83              sum[it]=0;
84          }
85          vis.clear();
86          v1.clear();
87      }
```

```
88      void makexs()
89      {
90          sort(v1.begin(),v1.end(),[&](int x,int y){
91              return id[x]<id[y];
92          });
93          sta[0]=0;
94          int top=1;
95          sta[top]=1;
96          vis.push_back(1);
97          for(auto it:v1){
98              int l=lca(it,sta[top]);
99              if(l!=sta[top]){
100                 while(id[l]<id[sta[top-1]]){
101                     int w=getlen(sta[top],sta[top-1]);
102                     ADD(sta[top-1],sta[top],w);
103                     vis.push_back(sta[top--]);
104                 }
105                 if(id[l]>id[sta[top-1]]){
106                     int w=getlen(sta[top],l);
107                     ADD(l,sta[top],w);
108                     vis.push_back(sta[top]);
109                     sta[top]=l;
110                 }
111                 else {
112                     int w=getlen(sta[top],sta[top-1]);
113                     ADD(sta[top],sta[top-1],w);
114                     vis.push_back(sta[top--]);
115                 }
116             }
117             sta[++top]=it;
118         }
119         for(int i=1;i<top;i++){
120             int w=getlen(sta[i],sta[i+1]);
121             ADD(sta[i],sta[i+1],w);
122             vis.push_back(sta[i+1]);
123         }
124     }
125     void addintoxs(int x){
126         if(x==1)return;
127         v1.push_back(x);
128         b[x]=true;
129     }
130     void solve(){
131         now=0;
132         dfs(1,0);
133         for(int i=1;i<=n;i++){
134             makexs();
135             init();
136         }
137     }
138 };
139 XS xs;
```

LCT 维护联通性

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

```
 3  const int maxn = 10010;
 4
 5  struct LCT {
 6      int ch[maxn][2], fa[maxn], tag[maxn];
 7
 8      void clear(int x) { ch[x][0] = ch[x][1] = fa[x] = tag[x] = 0; }
 9
10      int getch(int x) { return ch[fa[x]][1] == x; }
11
12      int isroot(int x) { return ch[fa[x]][0] != x && ch[fa[x]][1] != x; }
13
14      void pushdown(int x) {
15          if (tag[x]) {
16          if (ch[x][0]) swap(ch[ch[x][0]][0], ch[ch[x][0]][1]), tag[ch[x][0]] ^= 1;
17          if (ch[x][1]) swap(ch[ch[x][1]][0], ch[ch[x][1]][1]), tag[ch[x][1]] ^= 1;
18          tag[x] = 0;
19          }
20      }
21
22      void update(int x) {
23          if (!isroot(x)) update(fa[x]);
24          pushdown(x);
25      }
26
27      void rotate(int x) {
28          int y = fa[x], z = fa[y], chx = getch(x), chy = getch(y);
29          fa[x] = z;
30          if (!isroot(y)) ch[z][chy] = x;
31          ch[y][chx] = ch[x][chx ^ 1];
32          fa[ch[x][chx ^ 1]] = y;
33          ch[x][chx ^ 1] = y;
34          fa[y] = x;
35      }
36
37      void splay(int x) {
38          update(x);
39          for (int f = fa[x]; f = fa[x], !isroot(x); rotate(x))
40          if (!isroot(f)) rotate(getch(x) == getch(f) ? f : x);
41      }
42
43      void access(int x) {
44          for (int f = 0; x; f = x, x = fa[x]) splay(x), ch[x][1] = f;
45      }
46
47      void makeroot(int x) {
48          access(x);
49          splay(x);
50          swap(ch[x][0], ch[x][1]);
51          tag[x] ^= 1;
52      }
53
54      int find(int x) {
55          access(x);
56          splay(x);
57          while (ch[x][0]) x = ch[x][0];
58          splay(x);
```

```
59          return x;
60      }
61 /*--------------------------------------------------------*/
62      bool query(int x,int y){//查询是否为同一颗树
63          return find(x)==find(y);
64      }
65      void addedge(int x,int y){
66          if (find(x) != find(y)) makeroot(x), fa[x] = y;
67      }
68      void deledge(int x,int y){
69          makeroot(x);
70          access(y);
71          splay(y);
72          if (ch[y][0] == x && !ch[x][1]) ch[y][0] = fa[x] = 0;
73      }
74 } st;
75
76 int n, q, x, y;
77 char op[maxn];
78
79 int main() {
80      scanf("%d%d", &n, &q);
81      while (q--) {
82          scanf("%s%d%d", op, &x, &y);
83          if (op[0] == 'Q') {
84              if (st.query(x,y))
85              printf("Yes\n");
86          else
87              printf("No\n");
88          }
89          if (op[0] == 'C')st.addedge(x,y);
90          if (op[0] == 'D')st.deledge(x,y);
91      }
92      return 0;
93 }
```

树链剖分

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  const int maxn=1e5+10;
5  int mod;
6  void add(int &x,int y){if((x+=y)>=mod)x-=mod;}
7  struct Node{
8      int l,r,res,tag;
9  };
10 struct SegmentTree{
11     Node a[maxn*4];
12     void tag_init(int i){
13         a[i].tag=0;
14     }
15     void tag_union(int fa,int i){
16         add(a[i].tag,a[fa].tag);
17     }
18     void tag_cal(int i){
19         add(a[i].res,(ll)a[i].tag*(a[i].r-a[i].l+1)%mod);
```

```
20        }
21        void pushdown(int i){
22            tag_cal(i);
23            if(a[i].l!=a[i].r){
24                tag_union(i,i*2);
25                tag_union(i,i*2+1);
26            }
27            tag_init(i);
28        }
29        void pushup(int i){
30            if(a[i].l==a[i].r)return;
31            pushdown(i*2);
32            pushdown(i*2+1);
33            a[i].res=(a[i*2].res+a[i*2+1].res)%mod;
34        }
35        void build(int i,int l,int r){
36            a[i].l=l,a[i].r=r;tag_init(i);a[i].res=0;
37            if(l>=r)return;
38            int mid=(l+r)/2;
39            build(i*2,l,mid);
40            build(i*2+1,mid+1,r);
41        }
42        void update(int i,int l,int r,int w){
43            pushdown(i);
44            if(a[i].r<l||a[i].l>r||l>r)return;
45            if(a[i].l>=l&&a[i].r<=r){
46                a[i].tag=w;
47                return;
48            }
49            update(i*2,l,r,w);
50            update(i*2+1,l,r,w);
51            pushup(i);
52        }
53        int query(int i,int l,int r){
54            pushdown(i);
55            if(a[i].r<l||a[i].l>r||l>r)return 0;
56            if(a[i].l>=l&&a[i].r<=r){
57                return a[i].res;
58            }
59            return (query(i*2,l,r)+query(i*2+1,l,r))%mod;
60        }
61 };
62 SegmentTree tri;
63 vector<int> ve[maxn];
64 int L[maxn],R[maxn],tot,root=1;
65 int siz[maxn],top[maxn],dep[maxn],fa[maxn],dfn[maxn];
66 void dfs_sz(int x,int h){
67     if(x!=root)ve[x].erase(find(ve[x].begin(),ve[x].end(),h));
68     siz[x] = 1;
69     dep[x]=dep[h]+1;
70     fa[x]=h;
71     for(auto &it:ve[x]) {
72         dfs_sz(it,x);
73         siz[x] += siz[it];
74         if(siz[it] > siz[ve[x][0]]) {
75             swap(it, ve[x][0]);
```

```
76              }
77          }
78    }
79    void dfs_hld(int x) {
80        L[x] =++tot;
81        dfn[tot]=x;
82        for(auto it: ve[x]) {
83            top[it] = (it == ve[x][0] ? top[x] : it);
84            dfs_hld(it);
85        }
86        R[x] =tot;
87    }
88    void chain_add(int x, int y, int w) {// chain add w
89        while(top[x] != top[y]) {
90        if(dep[top[x]] < dep[top[y]]) swap(x, y);
91        tri.update(1, L[top[x]], L[x], w);
92            x = fa[top[x]];
93     }
94      if(L[x] > L[y]) swap(x, y);
95      tri.update(1, L[x], L[y], w);
96    }
97    int chain_sum(int x, int y) {// query the length of chain
98      int sum=0;
99        while(top[x] != top[y]) {
100      if(dep[top[x]] < dep[top[y]]) swap(x, y);
101       add(sum,tri.query(1, L[top[x]], L[x]));
102            x = fa[top[x]];
103      }
104      if(L[x] > L[y]) swap(x, y);
105      add(sum,tri.query(1, L[x], L[y]));
106        return sum;
107    }
108    int main()
109    {
110        ios::sync_with_stdio(false);
111        cin.tie(0);
112        int n,q;cin>>n>>q>>root>>mod;
113        vector<int> a(n+1);
114        for(int i=1;i<=n;i++)cin>>a[i],a[i]%=mod;
115        for(int i=1;i<n;i++){
116            int x,y;cin>>x>>y;
117            ve[x].push_back(y);
118            ve[y].push_back(x);
119        }
120        top[root]=root;
121        tri.build(1,1,n);dfs_sz(root,0);dfs_hld(root);
122        for(int i=1;i<=n;i++){
123            tri.update(1,L[i],L[i],a[i]);
124        }
125        while(q--){
126            int op;cin>>op;
127            if(op==1){
128                int x,y,w;cin>>x>>y>>w;
129                chain_add(x,y,w);
130            }
131            if(op==2){
```

```
132              int x,y;cin>>x>>y;
133              cout<<chain_sum(x,y)<<"\n";
134          }
135          if(op==3){//update the subtree of root x +=y
136              int x,y;cin>>x>>y;
137              tri.update(1,L[x],R[x],y);
138          }
139          if(op==4){
140              int x;cin>>x;
141              cout<<tri.query(1,L[x],R[x])<<"\n";
142          }
143      }
144
145  }
```

数学

组合数

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  #define int long long
4  const int maxn=100010;
5  const int mod=1e9+7;
6  int ksm(int x,int k){
7      int res=1;
8      while(k){
9          if(k&1)res=res*x%mod;
10         x=x*x%mod;
11         k/=2;
12     }
13     return res;
14 }
15 int ny(int x){
16     return ksm(x,mod-2);
17 }
18 void add(int &x,int y){
19     if((x+=y)>=mod)x-=mod;
20 }
21 void del(int &x,int y){
22     if((x-=y)<0)x+=mod;
23 }
24 int inv[maxn],fac[maxn];
25 int C(int n,int m){return n==0?1:fac[n]*inv[n-m]%mod*inv[m]%mod;}
26 int A(int n,int m){return n==0?1:fac[n]*inv[n-m]%mod;}
27 void init(){
28     inv[0]=fac[0]=1;
29     inv[1]=1;
30     for(int i=1;i<maxn;i++){
31         fac[i]=fac[i-1]*i%mod;
32     }
33     inv[1]=1;
34     for(int i=2;i<maxn;i++){
35         inv[i]=(int)(mod-mod/i)*inv[mod%i]%mod;
36     }
37     inv[0]=1;
38     for(int i=1;i<maxn;i++){
39         inv[i]=inv[i-1]*inv[i]%mod;
40     }
41 }
42 signed main()
43 {
44     init();
45     if(mod==(int)(1e9+7))assert(C(2000,1000)==72475738);
46     if(mod==9982444353)assert(C(2000,1000)==472799582);
47 }
```

BSGS 指数方程余数问题(求 a^x=b%p)

```
1  #include <cstdio>
2  #include <cstring>
3  #include <cmath>
4  #include <algorithm>
```

```cpp
#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);
```

```
61        }
62        return 0;
63  }
```

EXGCD

```
1  int exgcd(int a, int b, int &x, int &y){//求ax+by=gcd(a,b)  !(a==0&&b==0)
2      if(b==0){
3          x=1;
4          y=0;
5          return a;
6      }
7      int d=exgcd(b,a%b,x,y);
8      int t=x;
9      x=y;
10     y=t-(a/b)*y;
11     return d;
12 }
```

FFT

```
1  //当vector用就可以了
2  #include <bits/stdc++.h>
3  #define fp(i, a, b) for (int i = (a), i##_ = (b) + 1; i < i##_; ++i)
4  #define fd(i, a, b) for (int i = (a), i##_ = (b) - 1; i > i##_; --i)
5  using namespace std;
6  using ll = int64_t;
7  using db = double;
8  /*--------------------------------------------------------------------------*/
9  struct cp {
10     db x, y;
11     cp(db real = 0, db imag = 0) : x(real), y(imag){};
12     cp operator+(cp b) const { return {x + b.x, y + b.y}; }
13     cp operator-(cp b) const { return {x - b.x, y - b.y}; }
14     cp operator*(cp b) const { return {x * b.x - y * b.y, x * b.y + y * b.x}; }
15 };
16 using vcp = vector<cp>;
17 using Poly = vector<int>;
18 namespace FFT {
19     const db pi = acos(-1);
20     vcp Omega(int L) {
21         vcp w(L); w[1] = 1;
22         for (int i = 2; i < L; i <<= 1) {
23             auto w0 = w.begin() + i / 2, w1 = w.begin() + i;
24             cp wn(cos(pi / i), sin(pi / i));
25             for (int j = 0; j < i; j += 2)
26                 w1[j] = w0[j >> 1], w1[j + 1] = w1[j] * wn;
27         }
28         return w;
29     }
30     auto W = Omega(1 << 21); // NOLINT
31     void DIF(cp *a, int n) {
32         cp x, y;
33         for (int k = n >> 1; k; k >>= 1)
34             for (int i = 0; i < n; i += k << 1)
35                 for (int j = 0; j < k; ++j)
36                     x = a[i + j], y = a[i + j + k],
37                     a[i + j + k] = (a[i + j] - y) *  W[k + j], a[i + j] = x + y;
```

```
38          }
39      void IDIT(cp *a, int n) {
40          cp x, y;
41          for (int k = 1; k < n; k <<= 1)
42              for (int i = 0; i < n; i += k << 1)
43                  for (int j = 0; j < k; ++j)
44                      x = a[i + j], y = a[i + j + k] * W[k + j],
45                      a[i + j + k] = x - y, a[i + j] = x + y;
46          const db Inv = 1. / n;
47          fp(i, 0, n - 1) a[i].x *= Inv, a[i].y *= Inv;
48          reverse(a + 1, a + n);
49      }
50  }
51
52  namespace Polynomial {
53      // basic operator
54      void DFT(vcp &a) { FFT::DIF(a.data(), a.size()); }
55      void IDFT(vcp &a) { FFT::IDIT(a.data(), a.size()); }
56      int norm(int n) { return 1 << (__lg(n - 1) + 1); }
57
58      // Poly mul
59      vcp &dot(vcp &a, vcp &b) { fp(i, 0, a.size() - 1) a[i] = a[i] * b[i]; return a; }
60      Poly operator+(Poly a, Poly b) {
61          int maxlen = max(a.size(), b.size());
62          Poly ans(maxlen + 1);
63          a.resize(maxlen + 1), b.resize(maxlen + 1);
64          for (int i = 0; i < maxlen;i++)
65              ans[i] = a[i] + b[i];
66          return ans;
67      }
68      Poly operator*(ll k, Poly a) {
69          Poly ans;
70          for(auto i:a)
71              ans.push_back(k * i);
72          return ans;
73      }
74      Poly operator*(Poly a, Poly b) {
75          int n = a.size() + b.size() - 1;
76          vcp c(norm(n));
77          fp(i, 0, a.size() - 1) c[i].x = a[i];
78          fp(i, 0, b.size() - 1) c[i].y = b[i];
79          DFT(c), dot(c, c), IDFT(c), a.resize(n);
80          fp(i, 0, n - 1) a[i] = int(c[i].y * .5 + .5);
81          return a;
82      }
83  }
84  /*-------------------------------------------------------------------------*/
85  using namespace Polynomial;
```

FWT

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const int mod = 998244353;
4  void add(int &x, int y) {
5    if ((x += y) >= mod) x -= mod;
6  }
```

```
 7  void del(int &x, int y) {
 8    if ((x -= y) < 0) x += mod;
 9  }
10  void fwtor(int a[], int m, int opt)  //(1,-1)
11  {
12    for (int len = 2; len <= m; len <<= 1)
13      for (int p = len >> 1, i = 0; i < m; i += len)
14        for (int j = i; j < i + p; j++)
15          if (opt > 0)
16            add(a[j + p], a[j]);
17          else
18            del(a[j + p], a[j]);
19  }
20  void fwtand(int a[], int m, int opt)  //(1,-1)
21  {
22    for (int len = 2; len <= m; len <<= 1)
23      for (int p = len >> 1, i = 0; i < m; i += len)
24        for (int j = i; j < i + p; j++)
25          if (opt > 0)
26            add(a[j], a[j + p]);
27          else
28            del(a[j], a[j + p]);
29  }
30  void fwtxor(int a[], int m, int opt)  //(1,1/2)
31  {
32    for (int len = 2; len <= m; len <<= 1)
33      for (int p = len >> 1, i = 0; i < m; i += len)
34        for (int j = i; j < i + p; j++) {
35          add(a[j], a[j + p]);
36          a[j + p] = (a[j] - 2ll * a[j + p] % mod + mod) % mod;
37          a[j] = 1ll * a[j] * opt % mod;
38          a[j + p] = 1ll * a[j + p] * opt % mod;
39        }
40  }
41  int a[1 << 17], b[1 << 17], c[1 << 17];
42  void mul(int a[], int b[], int c[], int m) {
43    for (int i = 0; i < m; i++) c[i] = 1ll * a[i] * b[i] % mod;
44  }
45  void print(int a[], int m) {
46    for (int i = 0; i < m; i++) cout << a[i] << " \n"[i == m - 1];
47  }
48  int main() {
49    int n;
50    cin >> n;
51    int m = 1 << n;
52    for (int i = 0; i < m; i++) cin >> a[i];
53    for (int i = 0; i < m; i++) cin >> b[i];
54
55    fwtor(a, m, 1), fwtor(b, m, 1), mul(a, b, c, m);
56    fwtor(a, m, -1), fwtor(b, m, -1), fwtor(c, m, -1), print(c, m);
57
58    fwtand(a, m, 1), fwtand(b, m, 1), mul(a, b, c, m);
59    fwtand(a, m, -1), fwtand(b, m, -1), fwtand(c, m, -1), print(c, m);
60
61    fwtxor(a, m, 1), fwtxor(b, m, 1), mul(a, b, c, m);
```

```
62      fwtxor(c, m, (mod + 1) / 2), print(c, m);
63  }
```

莫比乌斯反演

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   const int maxn=1e5+10;
4   int pr[maxn],tot,mul[maxn],phi[maxn];
5   bool vis[maxn];
6   void init(int n)
7   {
8       mul[1]=phi[1]=1;
9       for(int i=2;i<=n;i++)
10      {
11          if(!vis[i])
12          {
13              mul[i]=-1;
14              pr[++tot]=i;
15              phi[i]=i-1;
16          }
17          for(int j=1;j<=tot && (long long)pr[j]*i<=n;j++)
18          {
19              int num=pr[j]*i;
20              vis[num]=1;
21              mul[num]=-mul[i];
22              phi[num]=phi[i]*phi[pr[j]];
23              if(i%pr[j]==0)
24              {
25                  phi[num]=pr[j]*phi[i];
26                  mul[num]=0;
27                  break;
28              }
29          }
30      }
31  }
```

NTT

```
1   #include <bits/stdc++.h>
2
3   #define fp(i, a, b) for (int i = (a), i##_ = (b) + 1; i < i##_; ++i)
4   #define fd(i, a, b) for (int i = (a), i##_ = (b) - 1; i > i##_; --i)
5   #define file(s) freopen(s".in","r",stdin),freopen(s".out","w",stdout)
6   using namespace std;
7   const int maxn = 2e5 + 5, P = 998244353;
8   using arr = int[maxn];
9   using ll = int64_t;
10  /*-------------------------------------------------------------------------*/
11  class Cipolla {
12      int P, I2{};
13      using pll = pair<ll, ll>;
14  #define X first
15  #define Y second
16      ll mul(ll a, ll b) const { return a * b % P; }
17      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}; }
```

```
18      template<class T> T POW(T a, int b, T x) { for (; b; b >>= 1, a = mul(a, a)) if
   (b & 1) x = mul(x, a); return x; }
19  public:
20      Cipolla(int p = 0) : P(p) {}
21      pair<int, int> sqrt(int n) {
22          int a = rand(), x;
23          if (!(n %= P))return {0, 0};
24          if (POW(n, (P - 1) >> 1, (int)1) == P - 1) return {-1, -1};
25          while (POW(I2 = ((ll) a * a - n + P) % P, (P - 1) >> 1, (int)1) == 1) a
   = rand();
26          x = (int) POW(pll{a, 1}, (P + 1) >> 1, {1, 0}).X;
27          if (2 * x > P) x = P - x;
28          return {x, P - x};
29      }
30  #undef X
31  #undef Y
32  };
33  /*-------------------------------------------------------------------------------*/
34  #define ADD(a, b) (((a) += (b)) >= P ? (a) -=P : 0) // (a += b) %= P
35  #define SUB(a, b) (((a) -= (b)) < 0 ? (a) += P: 0)  // ((a -= b) += P) %= P
36  #define MUL(a, b) ((ll) (a) * (b) % P)
37  //vector<int> getInv(int L) {
38  //    vector<int> inv(L); inv[1] = 1;
39  //    fp(i, 1, L - 1) inv[i] = MUL((P - P / i), inv[P % i]);
40  //    return inv;
41  //}
42  //auto inv = getInv(maxn); // NOLINT
43  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; }
44  //int INV(int a) { return a < maxn ? inv[a] : POW(a); }
45  namespace NTT {
46      const int g = 3;
47      vector<int> Omega(int L) {
48          int wn = POW(g, P / L);
49          vector<int> w(L); w[L >> 1] = 1;
50          fp(i, L / 2 + 1, L - 1) w[i] = MUL(w[i - 1], wn);
51          fd(i, L / 2 - 1, 1) w[i] = w[i << 1];
52          return w;
53      }
54      auto W = Omega(1 << 21); // NOLINT
55      void DIF(int *a, int n) {
56          for (int k = n >> 1; k; k >>= 1)
57              for (int i = 0, y; i < n; i += k << 1)
58                  fp(j, 0, k - 1)
59                      y = a[i + j + k], a[i + j + k] = MUL(a[i + j] - y + P, W[k + j]),
   ADD(a[i + j], y);
60      }
61      void IDIT(int *a, int n) {
62          for (int k = 1; k < n; k <<= 1)
63              for (int i = 0, x, y; i < n; i += k << 1)
64                  fp(j, 0, k - 1)
65                      x = a[i + j], y = MUL(a[i + j + k], W[k + j]),
66                      a[i + j + k] = x - y < 0 ? x - y + P : x - y, ADD(a[i + j], y);
67          int Inv = P - (P - 1) / n;
68          fp(i, 0, n - 1) a[i] = MUL(a[i], Inv);
69          reverse(a + 1, a + n);
```

```
70          }
71      }
72      namespace Polynomial {
73          using Poly = std::vector<int>;
74
75          // mul/div int
76          Poly &operator*=(Poly &a, int b) { for (auto &x : a) x = MUL(x, b); return a; }
77          Poly operator*(Poly a, int b) { return a *= b; }
78          Poly operator*(int a, Poly b) { return b * a; }
79          Poly &operator/=(Poly &a, int b) { return a *= POW(b); }
80          Poly operator/(Poly a, int b) { return a /= b; }
81
82          // Poly add/sub
83          Poly &operator+=(Poly &a, Poly b) {
84              a.resize(max(a.size(), b.size()));
85              fp(i, 0, b.size() - 1) ADD(a[i], b[i]);
86              return a;
87          }
88          Poly operator+(Poly a, Poly b) { return a += b; }
89          Poly &operator-=(Poly &a, Poly b) {
90              a.resize(max(a.size(), b.size()));
91              fp(i, 0, b.size() - 1) SUB(a[i], b[i]);
92              return a;
93          }
94          Poly operator-(Poly a, Poly b) { return a -= b; }
95
96          // Poly mul
97          void DFT(Poly &a) { NTT::DIF(a.data(), a.size()); }
98          void IDFT(Poly &a) { NTT::IDIT(a.data(), a.size()); }
99          int norm(int n) { return 1 << (32 - __builtin_clz(n - 1)); }
100         void norm(Poly &a) { if (!a.empty()) a.resize(norm(a.size()), 0); }
101         Poly &dot(Poly &a, Poly &b) {
102             fp(i, 0, a.size() - 1) a[i] = MUL(a[i], b[i]);
103             return a;
104         }
105         Poly operator*(Poly a, Poly b) {
106             int n = a.size() + b.size() - 1, L = norm(n);
107             if (a.size() <= 8 || b.size() <= 8) {
108                 Poly c(n);
109                 fp(i, 0, a.size() - 1) fp(j, 0, b.size() - 1)
110                     c[i + j] = (c[i + j] + (ll) a[i] * b[j]) % P;
111                 return c;
112             }
113             a.resize(L), b.resize(L);
114             DFT(a), DFT(b), dot(a, b), IDFT(a);
115             return a.resize(n), a;
116         }
117
118         // Poly inv
119         Poly Inv2k(Poly a) { // a.size() = 2^k
120             int n = a.size(), m = n >> 1;
121             if (n == 1) return {POW(a[0])};
122             Poly b = Inv2k(Poly(a.begin(), a.begin() + m)), c = b;
123             b.resize(n), DFT(a), DFT(b), dot(a, b), IDFT(a);
124             fp(i, 0, n - 1) a[i] = i < m ? 0 : P - a[i];
125             DFT(a), dot(a, b), IDFT(a);
```

```
126             return move(c.begin(), c.end(), a.begin()), a;
127         }
128         Poly Inv(Poly a) {
129             int n = a.size();
130             norm(a), a = Inv2k(a);
131             return a.resize(n), a;
132         }
133
134         // Poly div/mod
135         Poly operator/(Poly a,Poly b){
136             int k = a.size() - b.size() + 1;
137             if (k < 0) return {0};
138             reverse(a.begin(), a.end());
139             reverse(b.begin(), b.end());
140             b.resize(k), a = a * Inv(b);
141             a.resize(k), reverse(a.begin(), a.end());
142             return a;
143         }
144         pair<Poly, Poly> operator%(Poly a, const Poly& b) {
145             Poly c = a / b;
146             a -= b * c, a.resize(b.size() - 1);
147             return {c, a};
148         }
149
150         // Poly sqrt
151         Poly Sqrt(Poly a) {
152             int n = a.size(), k = norm(n);
153             Poly b = {(new Cipolla(P))->sqrt(a[0]).first}, c;
154             a.resize(k * 2, 0);
155             for (int L = 2; L <= k; L <<= 1) {
156                 b.resize(2 * L, 0), c = Poly(a.begin(), a.begin() + L) * Inv(b);
157                 fp(i, 0, 2 * L - 1) b[i] = MUL(b[i] + c[i], (P + 1) / 2);
158             }
159             return b.resize(n), b;
160         }
161
162         // Poly calculus
163         void Derivative(Poly &a) {
164             fp(i, 1, a.size() - 1) a[i - 1] = MUL(i, a[i]);
165             a.pop_back();
166         }
167 }
```

任意模数 NTT

```
1  const long long mod =1e18;
2  namespace polynomial {
3      typedef complex<long double> cplx;
4      const long double pi = acos((long double)-1.0);
5      const int len = 15, mask = (1 << len) - 1;
6      struct UnitRoot {
7          static vector<cplx> w;
8          static vector<cplx> get_root(int n) {
9              n = 1 << 32 - __builtin_clz(n);
10             if (n > w.size()) {
11                 w.resize(n);
12                 for (int i = 0; i < n; i++)
```

```
13                        w[i] = cplx(cos(2 * i * pi / n), sin(2 * i * pi / n));
14                }
15                int m = w.size() / n;
16                vector<cplx> res(n);
17                for (int i = 0, j = 0; i < n; i++, j += m) res[i] = w[j];
18                return res;
19            }
20        };
21        vector<cplx> UnitRoot::w;
22
23        void fft(vector<cplx> &p, const vector<cplx> &w) {
24            int n = w.size();
25            for (int i = 1, j = 0; i < n - 1; ++i) {
26                int s = n;
27                do {
28                    s >>= 1;
29                    j ^= s;
30                } while (~j & s);
31                if (i < j) {
32                    swap(p[i], p[j]);
33                }
34            }
35            for (int d = 0; (1 << d) < n; ++d) {
36                int m = 1 << d, m2 = m * 2, rm = n >> (d + 1);
37                for (int i = 0; i < n; i += m2) {
38                    for (int j = 0; j < m; ++j) {
39                        auto &p1 = p[i + j + m], &p2 = p[i + j];
40                        auto t = w[rm * j] * p1;
41                        p1 = p2 - t;
42                        p2 = p2 + t;
43                    }
44                }
45            }
46        }
47        vector<long long> conv(const vector<long long> &a,const vector<long long> &b) {
48            vector<cplx> w = UnitRoot::get_root(a.size() + b.size() - 1);
49            int n = w.size();
50            vector<cplx> A(n), B(n), C(n), D(n);
51            for (int i = 0; i < a.size(); ++i) A[i] = cplx(a[i] >> len, a[i] & mask);
52            for (int i = 0; i < b.size(); ++i) B[i] = cplx(b[i] >> len, b[i] & mask);
53            fft(A, w), fft(B, w);
54            for (int i = 0; i < n; ++i) {
55                int j = (n - i) % n;
56                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);
57                C[j] = da * dd + da * dc * cplx(0, 1);D[j] = db * dd + db * dc * cplx(0, 1);
58            }
59            fft(C, w), fft(D, w);
60            vector<long long> res(a.size() + b.size() - 1);
61            for (int i = 0; i < res.size(); ++i) {
62                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;
63                res[i] = ((dd << (len * 2)) + ((db + dc) << len) + da) % mod;
64            }
```

```
65        return res;
66    }
67 };
68 using namespace polynomial;
```

Pollard_Rho

```
1  typedef long long ll;
2  map<ll, bool>P;
3  mt19937_64 rnd(time(0));
4  namespace Pollard_Rho
5  {
6  #define ldb long double
7  ll mul(ll x, ll y, ll mod)
8  {
9    return ((x * y - (ll)((ldb)x / mod * y) * mod) + mod) % mod;
10  }
11  ll gcd(ll a, ll b)
12  {
13    return (b == 0 ? a : gcd(b, a % b));
14  }
15  ll ksm(ll a, ll b, ll mod)
16  {
17    ll ans = 1; a %= mod;
18    while (b) {if (b & 1)ans = mul(ans, a, mod); b >>= 1; a = mul(a, a, mod);}
19   return ans;
20  }
21  int pr[15] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
22  bool Miller_Rabin(ll n)
23  {
24   if (n == 2 || n == 3)return 1;
25   if (n % 2 == 0 || n == 1)return 0;
26   ll d = n - 1;
27    int s = 0;
28   while (d % 2 == 0)s ++, d >>= 1;
29   for (int i = 0; i <= 11; i ++)
30   {
31     if (pr[i] >= n)break;
32     ll a = pr[i];
33     ll x = ksm(a, d, n);
34    ll y = 0;
35     for (int j = 0; j <= s - 1; j ++)
36     {
37       y = mul(x, x, n);
38       if (y == 1 && x != 1 && x != (n - 1))return 0;
39      x = y;
40     }
41     if (y != 1)return 0;
42  }
43   return 1;
44  }
45  ll Pollard_Rho(ll n)
46  {
47   ll now, pre, g;
48    while (true)
49   {
50      now = pre = rnd() % (n - 1) + 1;
```

```
51      g = 1;
52      ll c = rnd() % (n - 1) + 1;
53       for (int i = 1, fst = 1 ;; i ++)
54       {
55          now = (mul(now, now, n) + c) % n;
56          g = mul(g, abs(now - pre), n);
57        if (now == pre || !g)break;
58         if (!(i & 127) || i == fst)
59         {
60            g = gcd(g, n);
61          if (g > 1)return g;
62           if (i == fst)pre = now, fst <<= 1;
63        }
64       }
65    }
66  }
67  void Find(ll n)
68  {
69    if (n == 1)return ;
70    if (Miller_Rabin(n))
71   {
72      P[n] = 1;
73       return ;
74   }
75    ll p = Pollard_Rho(n);
76    int c = 0;
77    while (!(n % p))
78    {
79       n /= p, c ++;
80    }
81    Find(p);
82   Find(n);
83  }
84  }
85  void solve(int x,set<int> &s){
86      Pollard_Rho :: Find(x);
87      for (auto [x, _] : P)s.insert(x);
88      P.clear();
89  }
```

扩展中国剩余定理

```
1  #define int long long
2  int mul(int a,int b,int mod){//O(1)取模快速乘，不会爆 long long
3    return (a*b-(int)((long double)a/mod*b)*mod+mod)%mod;
4  }
5  int exgcd(int a, int b, int& x, int& y){
6   if(!b){
7      x = 1, y = 0;
8      return a;
9    }
10   int d = exgcd(b,a%b,y,x);
11   y -= a/b*x;
12   return d;
13  }
14  int solve(int n,vector<int>&mo,vector<int>&res){
15   int a1,m1;
```

```
16      a1=res[0],m1=mo[0];
17    bool ok = 1;
18    for(int i=1;i<n;i++){
19       int a2,m2,k1,k2;
20          m2=mo[i],a2=res[i];
21      int d = exgcd(m1,m2,k1,k2);
22       if((a2-a1)%d) ok = 0;
23       else{
24          k1=mul(k1,(a2-a1)/d,m2/d);//这个地方必须要用取模快速乘
25          a1=a1+k1*m1;
26        m1=abs(m1/d*m2);
27      }
28     }
29    if(ok)return (a1%m1+m1)%m1;
30       else return -1;
31  }
```

拉格朗日插值

```
1  #define int long long
2  const int N=1e6+10,mod=998244353;
3  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;}
4  int fac[N+5],facinv[N+5],inv[N+5];
5  struct LR{
6    int Inv(int n){return ksm(n,mod-2);}
7   void init(){   //预处理阶乘和阶乘逆元,逆元.
8      fac[0]=inv[0]=inv[1]=1;
9          for(int i=1;i<=N;i++)
10             fac[i]=fac[i-1]*i%mod;
11      facinv[N]=Inv(fac[N]);
12    for(int i=N-1;~i;i--)
13             facinv[i]=facinv[i+1]*(i+1)%mod;
14    for(int i=2;i<N+5;i++)
15      inv[i]=(mod-mod/i)*inv[mod%i]%mod;
16   }
17   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)
18          int n=x.size();
19      int s=0;
20          for(int i=0;i<n;i++)if(x[i]==k)return y[i];
21    for(int i=0;i<n;i++){
22       int p=y[i]%mod,q=1;
23       for(int j=0;j<n;j++){
24          if(i==j) continue;
25         p=p*((k-x[j])%mod+mod)%mod;
26         q=q*((x[i]-x[j])%mod+mod)%mod;
27       }
28       s=(s+p*Inv(q)%mod)%mod;
29     }return (s%mod+mod)%mod;
30   }
31   int inpo(vector<int>&f,int x){   //给定 连续 i 属于[0,n] f(i) 拉插 f(x)
32          int n=f.size()-1;
33          if(x>=0&&x<=n)return f[x];
34    int p,s=0;
35          vector<int>pre(n+1),suf(n+1);
36          pre[0]=x-0;
37          for(int i=1;i<=n;i++)pre[i]=pre[i-1]*(x-i)%mod;
```

```
38          suf[n]=x-n;
39          for(int i=n-1;i>=0;i--)suf[i]=suf[i+1]*(x-i)%mod;
40      for(int i=0;i<=n;i++){
41        p=facinv[n-i]%mod*facinv[i]%mod;
42              if(i>0)p=p*pre[i-1]%mod;
43              if(i<n)p=p*suf[i+1]%mod;
44        if((n-i)&1)  s=(s-p*f[i]%mod+mod)%mod;
45        else s=(s+p*f[i]%mod)%mod;
46      }
47      return (s%mod+mod)%mod;
48    }
49  }sol;
```

拉格朗日插值没有模数

```
1  struct LR{
2   int inter(std::vector <int> vec, int x){
3      int n = vec.size() - 1;
4      int ans = 0;
5      for (int i = 0; i <= n; ++ i){
6          int div = 1;
7          for (int j = 0; j <= n; ++ j){
8              if (i != j) div *= (i - j);
9          }
10         bool flag = div < 0;
11         div = std::abs(div);
12         int prod = vec[i];
13         for (int j = 0; j <= n; ++ j){
14             if (i == j) continue;
15             int gcd = std::abs(std::__gcd(x - j, div));
16             prod *= (x - j) / gcd;
17             div /= gcd;
18         }
19         ans += flag ? -prod : prod;
20      }
21      return ans;
22  }
```

杜教筛

```
1  const int maxn=3e6+10;
2  int sumf[maxn];
3  int Sum(int n){// 这是 f * g 的 n 项前缀和
4
5  }
6  int Sumg(int n){// g 的 n 项前缀和
7
8  }
9  map<int,int> f;
10 int F (int n) {
11  if (n <= 3000'000) return sumf[n]; // 预处理出 n 较小时的前缀和
12  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
13  int ans = Sum(n);
14  for (int l = 2, r; l <= n; l = r + 1) // 整除分块
15      r = n / (n / l), ans -= (Sumg(r) - Sumg(l-1)) * F (n / l);
16      // [l,r] 的 F (n / l) 是一样的，对 g(x) 求个和即可
17  return f[n] = ans / Sumg(1); // 别忘了除上 g(1)
```

```
18   }
19
```

线性筛质数

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   const int maxn=1e6+10;
4   bool v[maxn];
5   int n,pr;
6   vector<int> p;
7   void init()
8   {
9       v[1]=true;
10      for(int i=2;i<maxn;i++)
11      {
12       if(!v[i])p.push_back(i);
13       for(int j=0;j<p.size()&&i*p[j]<maxn;++j){v[i*p[j]]=true;if(i%p[j]==0)break;}
14      }
15  }
16
```

线性递推

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   #define rep(i,a,n) for (int i=a;i<n;i++)
4   #define per(i,a,n) for (int i=n-1;i>=a;i--)
5   #define all(x) (x).begin(),(x).end()
6   #define siz(x) ((int)(x).size())
7   typedef vector<int> VI;
8   typedef long long ll;
9   typedef pair<int,int> PII;
10  const ll mod=1000000007;
11  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;}
12  ll n;
13  namespace linear_seq {
14      const int N=10010;
15      ll res[N],base[N],_c[N],_md[N];
16
17      vector<int> Md;
18      void mul(ll *a,ll *b,int k) {
19          rep(i,0,k+k) _c[i]=0;
20          rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
21          for (int i=k+k-1;i>=k;i--) if (_c[i])
22              rep(j,0,siz(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
23          rep(i,0,k) a[i]=_c[i];
24      }
25      int solve(ll n,VI a,VI b) {
26          ll ans=0,pnt=0;
27          int k=siz(a);
28          assert(siz(a)==siz(b));
29          rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
30          Md.clear();
31          rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
32          rep(i,0,k) res[i]=base[i]=0;
33          res[0]=1;
```

```
34          while ((1ll<<pnt)<=n) pnt++;
35          for (int p=pnt;p>=0;p--) {
36              mul(res,res,k);
37              if ((n>>p)&1) {
38                  for (int i=k-1;i>=0;i--) res[i+1]=res[i];res[0]=0;
39                  rep(j,0,siz(Md)) res[Md[j]]=(res[Md[j]]-res[k]*_md[Md[j]])%mod;
40              }
41          }
42          rep(i,0,k) ans=(ans+res[i]*b[i])%mod;
43          if (ans<0) ans+=mod;
44          return ans;
45      }
46      VI BM(VI s) {
47          VI C(1,1),B(1,1);
48          int L=0,m=1,b=1;
49          rep(n,0,siz(s)) {
50              ll d=0;
51              rep(i,0,L+1) d=(d+(ll)C[i]*s[n-i])%mod;
52              if (d==0) ++m;
53              else if (2*L<=n) {
54                  VI T=C;
55                  ll c=mod-d*powmod(b,mod-2)%mod;
56                  while (siz(C)<siz(B)+m) C.push_back(0);
57                  rep(i,0,siz(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
58                  L=n+1-L; B=T; b=d; m=1;
59              } else {
60                  ll c=mod-d*powmod(b,mod-2)%mod;
61                  while (siz(C)<siz(B)+m) C.push_back(0);
62                  rep(i,0,siz(B)) C[i+m]=(C[i+m]+c*B[i])%mod;
63                  ++m;
64              }
65          }
66          return C;
67      }
68      int gao(VI a,ll n) {
69          VI c=BM(a);
70          c.erase(c.begin());
71          rep(i,0,siz(c)) c[i]=(mod-c[i])%mod;
72          return solve(n,c,VI(a.begin(),a.begin()+siz(c)));
73      }
74  };
75
76  int main() {
77      vector<int>v;
78      v.push_back(2);
79      v.push_back(24);
80      v.push_back(96);
81      v.push_back(416);
82      v.push_back(1536);
83      v.push_back(5504);
84      v.push_back(18944);
85      v.push_back(64000);
86      v.push_back(212992);
87      v.push_back(702464);
88    scanf("%lld", &n);
89      printf("%lld\n",1LL * linear_seq::gao(v,n-1) % mod);
```

```
90  }
91
```

辛普森积分

```
1  double simpson(double l, double r) {
2      double mid = (l + r) / 2;
3      return (r - l) * (f(l) + 4 * f(mid) + f(r)) / 6;   // 辛普森公式
4  }
5
6  double asr(double l, double r, double eps, double ans, int step) {//step 是递归的下限
7      double mid = (l + r) / 2;
8      double fl = simpson(l, mid), fr = simpson(mid, r);
9      if (abs(fl + fr - ans) <= 15 * eps && step < 0)
10         return fl + fr + (fl + fr - ans) / 15;   // 足够相似的话就直接返回
11     return asr(l, mid, eps / 2, fl, step - 1) +
12            asr(mid, r, eps / 2, fr, step - 1);   // 否则分割成两段递归求解
13 }
14
15 double calc(double l, double r, double eps) {
16     return asr(l, r, eps, simpson(l, r), 12);
17 }
18
```

高斯消元(模意义)

```
1  #define int long long
2  const int eps=0;
3  const int maxn=220;
4  const int mod=1e6+3;
5  int ksm(int x,int k){
6      int res=1;
7      while(k){
8          if(k&1)res=res*x%mod;
9          x=x*x%mod;
10         k/=2;
11     }
12     return res;
13 }
14 int ny(int x){
15     return ksm(x,mod-2);
16 }
17 void add(int &x,int y){
18     if((x+=y)>=mod)x-=mod;
19 }
20 void del(int &x,int y){
21     if((x-=y)<0)x+=mod;
22 }
23 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
24 int Gauss(int equ,int var){//equ 方程数 var 未知数个数   return 1表示有解
25     int i,j,k,col,max_r;
26     for(k=0,col=0;k<equ&&col<var;k++,col++){
27         max_r=k;
28         for(i=k+1;i<equ;i++)
29             if((a[i][col])>(a[max_r][col]))
30                 max_r=i;
31         if((a[max_r][col])==0)return 0;
32         if(k!=max_r){
```

```
33          for(j=col;j<var;j++)
34              swap(a[k][j],a[max_r][j]);
35          swap(x[k],x[max_r]);
36      }
37      x[k]=x[k]*ny(a[k][col])%mod;
38      for(j=col+1;j<var;j++)a[k][j]=a[k][j]*ny(a[k][col])%mod;
39      a[k][col]=1;
40      for(i=0;i<equ;i++)
41          if(i!=k){
42              del(x[i],x[k]*a[i][col]%mod);
43              for(j=col+1;j<var;j++)del(a[i][j],a[k][j]*a[i][col]%mod);
44              a[i][col]=0;
45          }
46      }
47      return 1;
48  }
```

高斯消元(浮点数)

```
1  #define lf double
2  //0 base
3  const lf eps=1e-9;
4  const int maxn=220;
5  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
6  int Gauss(int equ,int var){//equ 方程数 var 未知数个数  return 1表示有解
7      int i,j,k,col,max_r;
8      for(k=0,col=0;k<equ&&col<var;k++,col++){
9          max_r=k;
10         for(i=k+1;i<equ;i++)
11             if(fabs(a[i][col])>fabs(a[max_r][col]))
12                 max_r=i;
13         if(fabs(a[max_r][col])<eps)return 0;
14         if(k!=max_r){
15             for(j=col;j<var;j++)
16                 swap(a[k][j],a[max_r][j]);
17             swap(x[k],x[max_r]);
18         }
19         x[k]/=a[k][col];
20         for(j=col+1;j<var;j++)a[k][j]/=a[k][col];
21         a[k][col]=1;
22         for(i=0;i<equ;i++)
23             if(i!=k){
24                 x[i]-=x[k]*a[i][col];
25                 for(j=col+1;j<var;j++)a[i][j]-=a[k][j]*a[i][col];
26                 a[i][col]=0;
27             }
28     }
29     return 1;
30  }
```

计算几何

开头

```
1  #include <bits/stdc++.h>
2  using namespace std;
3
4  using point_t=long double;  //全局数据类型，可修改为 long long 等
5
6  constexpr point_t eps=1e-8;
7  constexpr long double PI=3.1415926535897932384l;
8
```

点与向量

```
1  // 点与向量
2  template<typename T> struct point
3  {
4      T x,y;
5
6      bool operator==(const point &a) const {return (abs(x-a.x)<=eps && abs(y-a.y)<=eps);}
7      bool operator<(const point &a) const {if (abs(x-a.x)<=eps) return y<a.y-eps;
   return x<a.x-eps;}
8      bool operator>(const point &a) const {return !(*this<a || *this==a);}
9      point operator+(const point &a) const {return {x+a.x,y+a.y};}
10     point operator-(const point &a) const {return {x-a.x,y-a.y};}
11     point operator-() const {return {-x,-y};}
12     point operator*(const T k) const {return {k*x,k*y};}
13     point operator/(const T k) const {return {x/k,y/k};}
14     T operator*(const point &a) const {return x*a.x+y*a.y;}  // 点积
15     T operator^(const point &a) const {return x*a.y-y*a.x;}  // 叉积，注意优先级
16     int toleft(const point &a) const {const auto t=(*this)^a; return (t>eps)-(t<-
   eps);}  // to-left 测试
17     T len2() const {return (*this)*(*this);}  // 向量长度的平方
18     T dis2(const point &a) const {return (a-(*this)).len2();}  // 两点距离的平方
19
20     // 涉及浮点数
21     long double len() const {return sqrtl(len2());}  // 向量长度
22     long double dis(const point &a) const {return sqrtl(dis2(a));}  // 两点距离
23     long double ang(const point &a) const {return acosl(max(-1.0l,min(1.0l,((*this)*a)/
   (len()*a.len()))));}  // 向量夹角
24     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）
25     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）
26  };
27
28  using Point=point<point_t>;
```

极角排序

```
1  // 极角排序
2  struct argcmp
3  {
4      bool operator()(const Point &a,const Point &b) const
5      {
6          const auto quad=[](const Point &a)
```

```
 7            {
 8                if (a.y<-eps) return 1;
 9                if (a.y>eps) return 4;
10                if (a.x<-eps) return 5;
11                if (a.x>eps) return 3;
12                return 2;
13            };
14            const int qa=quad(a),qb=quad(b);
15            if (qa!=qb) return qa<qb;
16            const auto t=a^b;
17            // if (abs(t)<=eps) return a*a<b*b-eps;   // 不同长度的向量需要分开
18            return t>eps;
19        }
20 };
```

直线

```
 1  // 直线
 2  template<typename T> struct line
 3  {
 4      point<T> p,v;  // p 为直线上一点，v 为方向向量
 5
 6      bool operator==(const line &a) const {return v.toleft(a.v)==0 && v.toleft(p-a.p)==0;}
 7      int toleft(const point<T> &a) const {return v.toleft(a-p);}  // to-left 测试
 8      bool operator<(const line &a) const  // 半平面交算法定义的排序
 9      {
10          if (abs(v^a.v)<=eps && v*a.v>=-eps) return toleft(a.p)==-1;
11          return argcmp()(v,a.v);
12      }
13
14      // 涉及浮点数
15      point<T> inter(const line &a) const {return p+v*((a.v^(p-a.p))/(v^a.v));}  // 直线交点
16      long double dis(const point<T> &a) const {return abs(v^(a-p))/v.len();}  // 点到直线距离
17      point<T> proj(const point<T> &a) const {return p+v*((v*(a-p))/(v*v));}  // 点在直线上的投影
18  };
19
20  using Line=line<point_t>;
```

线段

```
 1  //线段
 2  template<typename T> struct segment
 3  {
 4      point<T> a,b;
 5
 6      bool operator<(const segment &s) const {return make_pair(a,b)<make_pair(s.a,s.b);}
 7
 8      // 判定性函数建议在整数域使用
 9
10      // 判断点是否在线段上
11      // -1 点在线段端点 | 0 点不在线段上 | 1 点严格在线段上
12      int is_on(const point<T> &p) const
13      {
```

```
14        if (p==a || p==b) return -1;
15        return (p-a).toleft(p-b)==0 && (p-a)*(p-b)<-eps;
16    }
17
18    // 判断线段直线是否相交
19    // -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
20    int is_inter(const line<T> &l) const
21    {
22        if (l.toleft(a)==0 || l.toleft(b)==0) return -1;
23        return l.toleft(a)!=l.toleft(b);
24    }
25
26    // 判断两线段是否相交
27    // -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
28    int is_inter(const segment<T> &s) const
29    {
30        if (is_on(s.a) || is_on(s.b) || s.is_on(a) || s.is_on(b)) return -1;
31        const line<T> l{a,b-a},ls{s.a,s.b-s.a};
32        return l.toleft(s.a)*l.toleft(s.b)==-1 && ls.toleft(a)*ls.toleft(b)==-1;
33    }
34
35    // 点到线段距离
36    long double dis(const point<T> &p) const
37    {
38        if ((p-a)*(b-a)<-eps || (p-b)*(a-b)<-eps) return min(p.dis(a),p.dis(b));
39        const line<T> l{a,b-a};
40        return l.dis(p);
41    }
42
43    // 两线段间距离
44    long double dis(const segment<T> &s) const
45    {
46        if (is_inter(s)) return 0;
47        return min({dis(s.a),dis(s.b),s.dis(a),s.dis(b)});
48    }
49 };
50
51 using Segment=segment<point_t>;
```

多边形

```
1  // 多边形
2  template<typename T> struct polygon
3  {
4      vector<point<T>> p;   // 以逆时针顺序存储
5
6      size_t nxt(const size_t i) const {return i==p.size()-1?0:i+1;}
7      size_t pre(const size_t i) const {return i==0?p.size()-1:i-1;}
8
9      // 回转数
10     // 返回值第一项表示点是否在多边形边上
11     // 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
12     pair<bool,int> winding(const point<T> &a) const
13     {
14         int cnt=0;
15         for (size_t i=0;i<p.size();i++)
```

```
16              {
17                  const point<T> u=p[i],v=p[nxt(i)];
18                  if (abs((a-u)^(a-v))<=eps && (a-u)*(a-v)<=eps) return {true,0};
19                  if (abs(u.y-v.y)<=eps) continue;
20                  const Line uv={u,v-u};
21                  if (u.y<v.y-eps && uv.toleft(a)<=0) continue;
22                  if (u.y>v.y+eps && uv.toleft(a)>=0) continue;
23                  if (u.y<a.y-eps && v.y>=a.y-eps) cnt++;
24                  if (u.y>=a.y-eps && v.y<a.y-eps) cnt--;
25              }
26          return {false,cnt};
27      }
28
29      // 多边形面积的两倍
30      // 可用于判断点的存储顺序是顺时针或逆时针
31      T area() const
32      {
33          T sum=0;
34          for (size_t i=0;i<p.size();i++) sum+=p[i]^p[nxt(i)];
35          return sum;
36      }
37
38      // 多边形的周长
39      long double circ() const
40      {
41          long double sum=0;
42          for (size_t i=0;i<p.size();i++) sum+=p[i].dis(p[nxt(i)]);
43          return sum;
44      }
45  };
46
47  using Polygon=polygon<point_t>;
```

凸多边形

```
1  //凸多边形
2  template<typename T> struct convex: polygon<T>
3  {
4      // 闵可夫斯基和
5      convex operator+(const convex &c) const
6      {
7          const auto &p=this->p;
8          vector<Segment> e1(p.size()),e2(c.p.size()),edge(p.size()+c.p.size());
9          vector<point<T>> res; res.reserve(p.size()+c.p.size());
10         const auto cmp=[](const Segment &u,const Segment &v) {return argcmp()(u.b-
    u.a,v.b-v.a);};
11         for (size_t i=0;i<p.size();i++) e1[i]={p[i],p[this->nxt(i)]};
12         for (size_t i=0;i<c.p.size();i++) e2[i]={c.p[i],c.p[c.nxt(i)]};
13         rotate(e1.begin(),min_element(e1.begin(),e1.end(),cmp),e1.end());
14         rotate(e2.begin(),min_element(e2.begin(),e2.end(),cmp),e2.end());
15         merge(e1.begin(),e1.end(),e2.begin(),e2.end(),edge.begin(),cmp);
16         const auto check=[](const vector<point<T>> &res,const point<T> &u)
17         {
18             const auto back1=res.back(),back2=*prev(res.end(),2);
19             return (back1-back2).toleft(u-back1)==0 && (back1-back2)*(u-back1)>=-eps;
20         };
21         auto u=e1[0].a+e2[0].a;
```

```
22          for (const auto &v:edge)
23          {
24              while (res.size()>1 && check(res,u)) res.pop_back();
25              res.push_back(u);
26              u=u+v.b-v.a;
27          }
28          if (res.size()>1 && check(res,res[0])) res.pop_back();
29          return {res};
30      }
31
32      // 旋转卡壳
33      // func 为更新答案的函数，可以根据题目调整位置
34      template<typename F> void rotcaliper(const F &func) const
35      {
36          const auto &p=this->p;
37          const auto area=[](const point<T> &u,const point<T> &v,const point<T> &w)
    {return (w-u)^(w-v);};
38          for (size_t i=0,j=1;i<p.size();i++)
39          {
40              const auto nxti=this->nxt(i);
41              func(p[i],p[nxti],p[j]);
42              while (area(p[this->nxt(j)],p[i],p[nxti])>=area(p[j],p[i],p[nxti]))
43              {
44                  j=this->nxt(j);
45                  func(p[i],p[nxti],p[j]);
46              }
47          }
48      }
49
50      // 凸多边形的直径的平方
51      T diameter2() const
52      {
53          const auto &p=this->p;
54          if (p.size()==1) return 0;
55          if (p.size()==2) return p[0].dis2(p[1]);
56          T ans=0;
57          auto func=[&](const point<T> &u,const point<T> &v,const point<T> &w)
    {ans=max({ans,w.dis2(u),w.dis2(v)});};
58          rotcaliper(func);
59          return ans;
60      }
61
62      // 判断点是否在凸多边形内
63      // 复杂度 O(logn)
64      // -1 点在多边形边上 | 0 点在多边形外 | 1 点在多边形内
65      int is_in(const point<T> &a) const
66      {
67          const auto &p=this->p;
68          if (p.size()==1) return a==p[0]?-1:0;
69          if (p.size()==2) return segment<T>{p[0],p[1]}.is_on(a)?-1:0;
70          if (a==p[0]) return -1;
71          if ((p[1]-p[0]).toleft(a-p[0])==-1 || (p.back()-p[0]).toleft(a-p[0])==1)
    return 0;
72          const auto cmp=[&](const Point &u,const Point &v){return (u-p[0]).toleft(v-
    p[0])==1;};
73          const size_t i=lower_bound(p.begin()+1,p.end(),a,cmp)-p.begin();
```

```
74          if (i==1) return segment<T>{p[0],p[i]}.is_on(a)?-1:0;
75          if (i==p.size()-1 && segment<T>{p[0],p[i]}.is_on(a)) return -1;
76          if (segment<T>{p[i-1],p[i]}.is_on(a)) return -1;
77          return (p[i]-p[i-1]).toleft(a-p[i-1])>0;
78      }
79
80      // 凸多边形关于某一方向的极点
81      // 复杂度 O(logn)
82      // 参考资料: https://codeforces.com/blog/entry/48868
83      template<typename F> size_t extreme(const F &dir) const
84      {
85          const auto &p=this->p;
86          const auto check=[&](const size_t i){return dir(p[i]).toleft(p[this->nxt(i)]-
    p[i])>=0;};
87          const auto dir0=dir(p[0]); const auto check0=check(0);
88          if (!check0 && check(p.size()-1)) return 0;
89          const auto cmp=[&](const Point &v)
90          {
91              const size_t vi=&v-p.data();
92              if (vi==0) return 1;
93              const auto checkv=check(vi);
94              const auto t=dir0.toleft(v-p[0]);
95              if (vi==1 && checkv==check0 && t==0) return 1;
96              return checkv^(checkv==check0 && t<=0);
97          };
98          return partition_point(p.begin(),p.end(),cmp)-p.begin();
99      }
100
101      // 过凸多边形外一点求凸多边形的切线，返回切点下标
102      // 复杂度 O(logn)
103      // 必须保证点在多边形外
104      pair<size_t,size_t> tangent(const point<T> &a) const
105      {
106          const size_t i=extreme([&](const point<T> &u){return u-a;});
107          const size_t j=extreme([&](const point<T> &u){return a-u;});
108          return {i,j};
109      }
110
111      // 求平行于给定直线的凸多边形的切线，返回切点下标
112      // 复杂度 O(logn)
113      pair<size_t,size_t> tangent(const line<T> &a) const
114      {
115          const size_t i=extreme([&](...){return a.v;});
116          const size_t j=extreme([&](...){return -a.v;});
117          return {i,j};
118      }
119 };
120
121 using Convex=convex<point_t>;
```

圆

```
1 // 圆
2 struct Circle
3 {
4     Point c;
5     long double r;
```

```cpp
6
7      bool operator==(const Circle &a) const {return c==a.c && abs(r-a.r)<=eps;}
8      long double circ() const {return 2*PI*r;}  // 周长
9      long double area() const {return PI*r*r;}  // 面积
10
11     // 点与圆的关系
12     // -1 圆上 | 0 圆外 | 1 圆内
13     int is_in(const Point &p) const {const long double d=p.dis(c); return abs(d-
   r)<=eps?-1:d<r-eps;}
14
15     // 直线与圆关系
16     // 0 相离 | 1 相切 | 2 相交
17     int relation(const Line &l) const
18     {
19         const long double d=l.dis(c);
20         if (d>r+eps) return 0;
21         if (abs(d-r)<=eps) return 1;
22         return 2;
23     }
24
25     // 圆与圆关系
26     // -1 相同 | 0 相离 | 1 外切 | 2 相交 | 3 内切 | 4 内含
27     int relation(const Circle &a) const
28     {
29         if (*this==a) return -1;
30         const long double d=c.dis(a.c);
31         if (d>r+a.r+eps) return 0;
32         if (abs(d-r-a.r)<=eps) return 1;
33         if (abs(d-abs(r-a.r))<=eps) return 3;
34         if (d<abs(r-a.r)-eps) return 4;
35         return 2;
36     }
37
38     // 直线与圆的交点
39     vector<Point> inter(const Line &l) const
40     {
41         const long double d=l.dis(c);
42         const Point p=l.proj(c);
43         const int t=relation(l);
44         if (t==0) return vector<Point>();
45         if (t==1) return vector<Point>{p};
46         const long double k=sqrt(r*r-d*d);
47         return vector<Point>{p-(l.v/l.v.len())*k,p+(l.v/l.v.len())*k};
48     }
49
50     // 圆与圆交点
51     vector<Point> inter(const Circle &a) const
52     {
53         const long double d=c.dis(a.c);
54         const int t=relation(a);
55         if (t==-1 || t==0 || t==4) return vector<Point>();
56         Point e=a.c-c; e=e/e.len()*r;
57         if (t==1 || t==3)
58         {
59             if (r*r+d*d-a.r*a.r>=-eps) return vector<Point>{c+e};
60             return vector<Point>{c-e};
```

100

```
61              }
62              const long double costh=(r*r+d*d-a.r*a.r)/(2*r*d),sinth=sqrt(1-costh*costh);
63              return vector<Point>{c+e.rot(costh,-sinth),c+e.rot(costh,sinth)};
64          }
65
66          // 圆与圆交面积
67          long double inter_area(const Circle &a) const
68          {
69              const long double d=c.dis(a.c);
70              const int t=relation(a);
71              if (t==-1) return area();
72              if (t<2) return 0;
73              if (t>2) return min(area(),a.area());
74              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);
75              const long double sinth1=sqrt(1-costh1*costh1),sinth2=sqrt(1-costh2*costh2);
76              const long double th1=acos(costh1),th2=acos(costh2);
77              return r*r*(th1-costh1*sinth1)+a.r*a.r*(th2-costh2*sinth2);
78          }
79
80          // 过圆外一点圆的切线
81          vector<Line> tangent(const Point &a) const
82          {
83              const int t=is_in(a);
84              if (t==1) return vector<Line>();
85              if (t==-1)
86              {
87                  const Point v={-(a-c).y,(a-c).x};
88                  return vector<Line>{{a,v}};
89              }
90              Point e=a-c; e=e/e.len()*r;
91              const long double costh=r/c.dis(a),sinth=sqrt(1-costh*costh);
92              const Point t1=c+e.rot(costh,-sinth),t2=c+e.rot(costh,sinth);
93              return vector<Line>{{a,t1-a},{a,t2-a}};
94          }
95
96          // 两圆的公切线
97          vector<Line> tangent(const Circle &a) const
98          {
99              const int t=relation(a);
100             vector<Line> lines;
101             if (t==-1 || t==4) return lines;
102             if (t==1 || t==3)
103             {
104                 const Point p=inter(a)[0],v={-(a.c-c).y,(a.c-c).x};
105                 lines.push_back({p,v});
106             }
107             const long double d=c.dis(a.c);
108             const Point e=(a.c-c)/(a.c-c).len();
109             if (t<=2)
110             {
111                 const long double costh=(r-a.r)/d,sinth=sqrt(1-costh*costh);
112                 const Point d1=e.rot(costh,-sinth),d2=e.rot(costh,sinth);
113                 const Point u1=c+d1*r,u2=c+d2*r,v1=a.c+d1*a.r,v2=a.c+d2*a.r;
114                 lines.push_back({u1,v1-u1}); lines.push_back({u2,v2-u2});
115             }
```

```
116        if (t==0)
117        {
118            const long double costh=(r+a.r)/d,sinth=sqrt(1-costh*costh);
119            const Point d1=e.rot(costh,-sinth),d2=e.rot(costh,sinth);
120            const Point u1=c+d1*r,u2=c+d2*r,v1=a.c-d1*a.r,v2=a.c-d2*a.r;
121            lines.push_back({u1,v1-u1}); lines.push_back({u2,v2-u2});
122        }
123        return lines;
124    }
125
126    //  圆的反演
127    tuple<int,Circle,Line> inverse(const Line &l) const
128    {
129        const Circle null_c={{0.0,0.0},0.0};
130        const Line null_l={{0.0,0.0},{0.0,0.0}};
131        if (l.toleft(c)==0) return {2,null_c,l};
132        const Point v=l.toleft(c)==1?Point{l.v.y,-l.v.x}:Point{-l.v.y,l.v.x};
133        const long double d=r*r/l.dis(c);
134        const Point p=c+v/v.len()*d;
135        return {1,{(c+p)/2,d/2},null_l};
136    }
137
138    tuple<int,Circle,Line> inverse(const Circle &a) const
139    {
140        const Circle null_c={{0.0,0.0},0.0};
141        const Line null_l={{0.0,0.0},{0.0,0.0}};
142        const Point v=a.c-c;
143        if (a.is_in(c)==-1)
144        {
145            const long double d=r*r/(a.r+a.r);
146            const Point p=c+v/v.len()*d;
147            return {2,null_c,{p,{-v.y,v.x}}};
148        }
149        if (c==a.c) return {1,{c,r*r/a.r},null_l};
150        const long double d1=r*r/(c.dis(a.c)-a.r),d2=r*r/(c.dis(a.c)+a.r);
151        const Point p=c+v/v.len()*d1,q=c+v/v.len()*d2;
152        return {1,{(p+q)/2,p.dis(q)/2},null_l};
153    }
154 };
155
156 // 圆与多边形面积交
157 long double area_inter(const Circle &circ,const Polygon &poly)
158 {
159     const auto cal=[](const Circle &circ,const Point &a,const Point &b)
160     {
161         if ((a-circ.c).toleft(b-circ.c)==0) return 0.0l;
162         const auto ina=circ.is_in(a),inb=circ.is_in(b);
163         const Line ab={a,b-a};
164         if (ina && inb) return ((a-circ.c)^(b-circ.c))/2;
165         if (ina && !inb)
166         {
167             const auto t=circ.inter(ab);
168             const Point p=t.size()==1?t[0]:t[1];
169             const long double ans=((a-circ.c)^(p-circ.c))/2;
170             const long double th=(p-circ.c).ang(b-circ.c);
171             const long double d=circ.r*circ.r*th/2;
```

```
172              if ((a-circ.c).toleft(b-circ.c)==1) return ans+d;
173              return ans-d;
174          }
175          if (!ina && inb)
176          {
177              const Point p=circ.inter(ab)[0];
178              const long double ans=((p-circ.c)^(b-circ.c))/2;
179              const long double th=(a-circ.c).ang(p-circ.c);
180              const long double d=circ.r*circ.r*th/2;
181              if ((a-circ.c).toleft(b-circ.c)==1) return ans+d;
182              return ans-d;
183          }
184          const auto p=circ.inter(ab);
185          if (p.size()==2 && Segment{a,b}.dis(circ.c)<=circ.r+eps)
186          {
187              const long double ans=((p[0]-circ.c)^(p[1]-circ.c))/2;
188            const long double th1=(a-circ.c).ang(p[0]-circ.c),th2=(b-circ.c).ang(p[1]-
    circ.c);
189              const long double d1=circ.r*circ.r*th1/2,d2=circ.r*circ.r*th2/2;
190              if ((a-circ.c).toleft(b-circ.c)==1) return ans+d1+d2;
191              return ans-d1-d2;
192          }
193          const long double th=(a-circ.c).ang(b-circ.c);
194          if ((a-circ.c).toleft(b-circ.c)==1) return circ.r*circ.r*th/2;
195          return -circ.r*circ.r*th/2;
196      };
197
198      long double ans=0;
199      for (size_t i=0;i<poly.p.size();i++)
200      {
201          const Point a=poly.p[i],b=poly.p[poly.nxt(i)];
202          ans+=cal(circ,a,b);
203      }
204      return ans;
205  }
```

判断多条线段是否有交点

```
1  // 判断多条线段是否有交点
2  // 扫描线, 复杂度 O(nlogn)
3  bool segs_inter(const vector<Segment> &segs)
4  {
5      if (segs.empty()) return false;
6      using seq_t=tuple<point_t,int,Segment>;
7      const auto seqcmp=[](const seq_t &u, const seq_t &v)
8      {
9          const auto [u0,u1,u2]=u;
10         const auto [v0,v1,v2]=v;
11         if (abs(u0-v0)<=eps) return make_pair(u1,u2)<make_pair(v1,v2);
12         return u0<v0-eps;
13     };
14     vector<seq_t> seq;
15     for (auto seg:segs)
16     {
17         if (seg.a.x>seg.b.x+eps) swap(seg.a,seg.b);
18         seq.push_back({seg.a.x,0,seg});
19         seq.push_back({seg.b.x,1,seg});
```

```
20        }
21        sort(seq.begin(),seq.end(),seqcmp);
22        point_t x_now;
23        auto cmp=[&](const Segment &u, const Segment &v)
24        {
25            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;
26          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;
27        };
28        multiset<Segment,decltype(cmp)> s{cmp};
29        for (const auto [x,o,seg]:seq)
30        {
31            x_now=x;
32            const auto it=s.lower_bound(seg);
33            if (o==0)
34            {
35                if (it!=s.end() && seg.is_inter(*it)) return true;
36                if (it!=s.begin() && seg.is_inter(*prev(it))) return true;
37                s.insert(seg);
38            }
39            else
40            {
41                if (next(it)!=s.end() && it!=s.begin() && (*prev(it)).is_inter(*next(it)))
    return true;
42                s.erase(it);
43            }
44        }
45        return false;
46    }
```

半平面交

```
1  // 半平面交
2  // 排序增量法，复杂度 O(nlogn)
3  // 输入与返回值都是用直线表示的半平面集合
4  vector<Line> halfinter(vector<Line> l, const point_t lim=1e9)
5  {
6      const auto check=[](const Line &a,const Line &b,const Line &c){return
   a.toleft(b.inter(c))<0;};
7      // 无精度误差的方法，但注意取值范围会扩大到三次方
8      /*const auto check=[](const Line &a,const Line &b,const Line &c)
9      {
10         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);
11         return p.toleft(q)<0;
12     };*/
13     l.push_back({{-lim,0},{0,-1}}); l.push_back({{0,-lim},{1,0}});
14     l.push_back({{lim,0},{0,1}}); l.push_back({{0,lim},{-1,0}});
15     sort(l.begin(),l.end());
16     deque<Line> q;
17     for (size_t i=0;i<l.size();i++)
18     {
19         if (i>0 && l[i-1].v.toleft(l[i].v)==0 && l[i-1].v*l[i].v>eps) continue;
20         while (q.size()>1 && check(l[i],q.back(),q[q.size()-2])) q.pop_back();
21         while (q.size()>1 && check(l[i],q[0],q[1])) q.pop_front();
22         if (!q.empty() && q.back().v.toleft(l[i].v)<=0) return vector<Line>();
23         q.push_back(l[i]);
24     }
```

```
25         while (q.size()>1 && check(q[0],q.back(),q[q.size()-2])) q.pop_back();
26         while (q.size()>1 && check(q.back(),q[0],q[1])) q.pop_front();
27         return vector<Line>(q.begin(),q.end());
28     }
```

圆面积并

```
1  // 圆面积并
2  // 轮廓积分, 复杂度 O(n^2logn)
3  // ans[i] 表示被至少覆盖了 i+1 次的区域的面积
4  vector<long double> area_union(const vector<Circle> &circs)
5  {
6      const size_t siz=circs.size();
7      using arc_t=tuple<Point,long double,long double,long double>;
8      vector<vector<arc_t>> arcs(siz);
9      const auto eq=[](const arc_t &u,const arc_t &v)
10     {
11         const auto [u1,u2,u3,u4]=u;
12         const auto [v1,v2,v3,v4]=v;
13         return u1==v1 && abs(u2-v2)<=eps && abs(u3-v3)<=eps && abs(u4-v4)<=eps;
14     };
15
16     auto cut_circ=[&](const Circle &ci,const size_t i)
17     {
18         vector<pair<long double,int>> evt;
19         evt.push_back({-PI,0}); evt.push_back({PI,0});
20         int init=0;
21         for (size_t j=0;j<circs.size();j++)
22         {
23             if (i==j) continue;
24             const Circle &cj=circs[j];
25             if (ci.r<cj.r-eps && ci.relation(cj)>=3) init++;
26             const auto inters=ci.inter(cj);
27             if (inters.size()==1) evt.push_back({atan2l((inters[0]-ci.c).y,(inters[0]-ci.c).x),0});
28             if (inters.size()==2)
29             {
30                 const Point dl=inters[0]-ci.c,dr=inters[1]-ci.c;
31                 long double argl=atan2l(dl.y,dl.x),argr=atan2l(dr.y,dr.x);
32                 if (abs(argl+PI)<=eps) argl=PI;
33                 if (abs(argr+PI)<=eps) argr=PI;
34                 if (argl>argr+eps)
35                 {
36                     evt.push_back({argl,1}); evt.push_back({PI,-1});
37                     evt.push_back({-PI,1}); evt.push_back({argr,-1});
38                 }
39                 else
40                 {
41                     evt.push_back({argl,1});
42                     evt.push_back({argr,-1});
43                 }
44             }
45         }
46         sort(evt.begin(),evt.end());
47         int sum=init;
48         for (size_t i=0;i<evt.size();i++)
49         {
```

```
50              sum+=evt[i].second;
51                  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});
52              if (abs(evt[i+1].first-PI)<=eps) break;
53          }
54      };
55
56      const auto oint=[](const arc_t &arc)
57      {
58          const auto [cc,cr,l,r]=arc;
59          if (abs(r-l-PI-PI)<=eps) return 2.0l*PI*cr*cr;
60          return cr*cr*(r-l)+cc.x*cr*(sin(r)-sin(l))-cc.y*cr*(cos(r)-cos(l));
61      };
62
63      for (size_t i=0;i<circs.size();i++)
64      {
65          const auto &ci=circs[i];
66          cut_circ(ci,i);
67      }
68      vector<long double> ans(siz);
69      for (size_t i=0;i<siz;i++)
70      {
71          long double sum=0;
72          sort(arcs[i].begin(),arcs[i].end());
73          int cnt=0;
74          for (size_t j=0;j<arcs[i].size();j++)
75          {
76              if (j>0 && eq(arcs[i][j],arcs[i][j-1])) arcs[i+(++cnt)].push_back(arcs[i]
   [j]);
77              else cnt=0,sum+=oint(arcs[i][j]);
78          }
79          ans[i]=sum/2;
80      }
81      return ans;
82  }
```

多边形面积并

```
1  // 多边形面积并
2  // 轮廓积分，复杂度 O(n^2logn)，n 为边数
3  // ans[i] 表示被至少覆盖了 i+1 次的区域的面积
4  vector<long double> area_union(const vector<Polygon> &polys)
5  {
6      const size_t siz=polys.size();
7      vector<vector<pair<Point,Point>>> segs(siz);
8      const auto check=[](const Point &u,const Segment &e){return !((u<e.a && u<e.b) ||
   (u>e.a && u>e.b));};
9
10     auto cut_edge=[&](const Segment &e,const size_t i)
11     {
12         const Line le{e.a,e.b-e.a};
13         vector<pair<Point,int>> evt;
14         evt.push_back({e.a,0}); evt.push_back({e.b,0});
15         for (size_t j=0;j<polys.size();j++)
16         {
17             if (i==j) continue;
18             const auto &pj=polys[j];
```

```cpp
            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

```
1   //  点集形成的最小最大三角形
2   //  极角序扫描线，复杂度 O(n^2logn)
3   //  最大三角形问题可以使用凸包与旋转卡壳做到 O(n^2)
4   pair<point_t,point_t> minmax_triangle(const vector<Point> &vec)
5   {
6       if (vec.size()<=2) return {0,0};
7       vector<pair<int,int>> evt;
8       evt.reserve(vec.size()*vec.size());
9       point_t maxans=0,minans=numeric_limits<point_t>::max();
10      for (size_t i=0;i<vec.size();i++)
11      {
12          for (size_t j=0;j<vec.size();j++)
13          {
14              if (i==j) continue;
15              if (vec[i]==vec[j]) minans=0;
16              else evt.push_back({i,j});
17          }
18      }
19      sort(evt.begin(),evt.end(),[&](const pair<int,int> &u,const pair<int,int> &v)
20      {
21          const Point du=vec[u.second]-vec[u.first],dv=vec[v.second]-vec[v.first];
22          return argcmp()({du.y,-du.x},{dv.y,-dv.x});
23      });
24      vector<size_t> vx(vec.size()),pos(vec.size());
25      for (size_t i=0;i<vec.size();i++) vx[i]=i;
26      sort(vx.begin(),vx.end(),[&](int x,int y){return vec[x]<vec[y];});
27      for (size_t i=0;i<vx.size();i++) pos[vx[i]]=i;
28      for (auto [u,v]:evt)
29      {
30          const size_t i=pos[u],j=pos[v];
31          const size_t l=min(i,j),r=max(i,j);
32          const Point vecu=vec[u],vecv=vec[v];
33          if (l>0) minans=min(minans,abs((vec[vx[l-1]]-vecu)^(vec[vx[l-1]]-vecv)));
34          if (r<vx.size()-1) minans=min(minans,abs((vec[vx[r+1]]-vecu)^(vec[vx[r+1]]-
    vecv)));
35          maxans=max({maxans,abs((vec[vx[0]]-vecu)^(vec[vx[0]]-
    vecv)),abs((vec[vx.back()]-vecu)^(vec[vx.back()]-vecv))});
36          if (i<j) swap(vx[i],vx[j]),pos[u]=j,pos[v]=i;
37      }
38      return {minans,maxans};
39  }
```

点集的凸包

```
1   // 点集的凸包
2   // Andrew 算法，复杂度 O(nlogn)
3   Convex convexhull(vector<Point> p)
4   {
5       vector<Point> st;
6       if (p.empty()) return Convex{st};
7       sort(p.begin(),p.end());
8       const auto check=[](const vector<Point> &st,const Point &u)
9       {
10          const auto back1=st.back(),back2=*prev(st.end(),2);
11          return (back1-back2).toleft(u-back1)<=0;
12      };
13      for (const Point &u:p)
```

```
14      {
15          while (st.size()>1 && check(st,u)) st.pop_back();
16          st.push_back(u);
17      }
18      size_t k=st.size();
19      p.pop_back(); reverse(p.begin(),p.end());
20      for (const Point &u:p)
21      {
22          while (st.size()>k && check(st,u)) st.pop_back();
23          st.push_back(u);
24      }
25      st.pop_back();
26      return Convex{st};
27  }
```

给定三点求圆心

```
1  void cal(Point& a, Point& b, Point& c){ //圆上三点定圆心
2      double a1=b.x-a.x, b1=b.y-a.y, c1=(a1*a1+b1*b1)/2;
3      double a2=c.x-a.x, b2=c.y-a.y, c2=(a2*a2+b2*b2)/2;
4      double d =a1*b2-a2*b1;
5      X =a.x+(c1*b2-c2*b1)/d;
6      Y =a.y+(a1*c2-a2*c1)/d;
7      r=dis2(a);
8  }
9
```