



# Antiamuny Code Library

tarjen

# 目录

第零章 .....	1
赛前准备 .....	2
测样例脚本(python 版) .....	2
测样例脚本(bash 版) .....	2
对拍(cpp 版本) .....	2
对拍(bat 版本) .....	2
杂项 .....	3
rand .....	3
time .....	3
子集枚举 .....	3
高维前缀和 .....	3
三分 .....	3
bitset .....	4
bitset 手写 .....	4
lcslen(n2w) .....	5
wqs 二分 .....	6
判断异或方程组是否有解 .....	7
可以判断不同或相同的并查集 .....	8
大数 int128 .....	8
树哈希 .....	9
求 s 所有前缀对于 t 的所有子串的 lcs 长度 .....	10
线性基 .....	11
维护多个二维向量能够表示的范围 .....	14
字符串 .....	15
AC 自动机 .....	15
Dequehash .....	16
Exkmp .....	17
Hash .....	19
Kmp .....	19
Manacher .....	20
倍增 SA .....	21
后缀自动机 SAM .....	22
回文自动机 PAM .....	24
最小表示 .....	25
图论 .....	26
johnson 全源最短路 .....	26
kosaraju .....	26
K 短路 .....	27
lca(o1) .....	30
maxflow 只算值版本 .....	31
maxflow 网络流最大流 .....	32
tarjan 缩点 .....	33
二分图匹配 .....	33
二分图最优匹配 .....	34
二分图染色 .....	35
圆方树 .....	36
基环树 .....	37

带权并查集 dsu .....	39
带花树 .....	39
带负环最小费用最大流 .....	41
支配树 .....	43
最小环 .....	44
最小费用最大流 .....	44
欧拉回路 .....	45
数学 .....	47
组合数 .....	47
BSGS 指数方程余数问题(求 $a^x=b\%p$ ) .....	47
EXGCD .....	49
FFT .....	49
FWT .....	50
莫比乌斯反演 .....	52
NTT .....	52
任意模数 NTT .....	55
Pollard_Rho .....	57
扩展中国剩余定理 .....	58
拉格朗日插值 .....	59
拉格朗日插值没有模数 .....	60
杜教筛 .....	60
线性筛质数 .....	61
线性递推 .....	61
辛普森积分 .....	63
高斯消元(模意义) .....	63
高斯消元(浮点数) .....	64
计算几何 .....	65

## 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)

```
1 import os
2 import sys
3 import zipfile
4
5 c = sys.argv[1]
6 code
```

### 测样例脚本(bash 版)

### 对拍(cpp 版本)

```
1 int t=10000,j=0;
2 while(t)
3 {
4     cout<<"test:"<<+j<<"\n";
5     t--;
6     system("testin.exe > data.txt");
7     system("abiao Cheng.exe < data.txt > biaoda.txt");
8     system("nedtest.exe < data.txt > aatest.txt");
9     if(system("fc aatest.txt biaoda.txt")){
10         cout<<"error"<<"\n";
11         break;
12     }
13 }
14 if(t==0) cout<<"no error"<<endl;
15 //system("pause");
16 return 0;
```

### 对拍(bat 版本)

```
1 @echo off
2 setlocal enabledelayedexpansion
3
4 set T=0
5 :loop
6 if %T% gtr 100000 (
7     echo "Finished"
8     exit /b
9 )
10 set /a T+=1
11 echo T=!T!
12 testin.exe > data.txt
13 abiao Cheng.exe < data.txt > biaoda.txt
14 nedtest.exe < data.txt > aatest.txt
15
16 fc aatest.txt biaoda.txt
17 if errorlevel 1 (
18     echo "WA"
19     exit /b
20 )
21
22 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 shiftl(){
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(n2w)

```

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 shiftl(){
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  $x_1^{i_1}x_2^{i_2}x_3^{i_3}\dots=1/0$  的形式
3  bitset<maxn>b[maxn]; //1 表示这个方程组存在  $x_i$  比如  $x_3 \times x_5 = 1$  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

```
1 int f[maxn];
2 int getf(int x){
3     if(x<0)return -getf(-x);
4     if(x==f[x])return x;
5     else return f[x]=getf(f[x]);
6 }
7 bool merge(int x,int y){//如果是 x!=y 将 y 取反(x>0 y>0)
8     x=getf(x),y=getf(y);
9     if(x==y)return false;
10    if(x==y)return true;
11    if(x<0)f[-x]=-y;
12    else f[x]=y;
13    return true;
14 }
```

大数 **int128**

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 inline __int128 read()
4 {
5     __int128 x=0,f=1;
6     char ch=getchar();
7     while(ch<'0' || ch>'9')
8     {
9         if(ch=='-')
10             f=-1;
11         ch=getchar();
12     }
13     while(ch>='0' && ch<='9')
14     {
15         x=x*10+ch-'0';
16         ch=getchar();
17     }
18     return x*f;
19 }
20 inline void write(__int128 x)
21 {
22     if(x<0)
23     {
24         putchar('-');
25         x=-x;
26     }
27     if(x>9)
28         write(x/10);
29     putchar(x%10+'0');
30 }
31
32 istream& operator >> (istream& in, __int128& num) {
33     string s;in>>s;
34     num=0;
35     for(auto it:s)num=num*10+it-'0';
36     return in;
37 }
38
39 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

```

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 **lcs** 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;

```

```

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]
74 ^= a[j];
75                 for(int j = i + 1; j <= maxbase; ++j)if(a[j] >> i & 1)a[j] ^=
76 a[i];
77                 break;
78             }
79         }
80     }
81 }
82 LinearBasis merge(const LinearBasis &l1, const LinearBasis &l2)
83 {
84     // 得到两个线性基的并
85     LinearBasis ret = l1;
86     for(int i = maxbase; i >= 0; --i)
87     {
88         if(l2.a[i])
89             ret.insert(l2.a[i]);
90     }
91     return ret;
92 }
93 LinearBasis intersection(const LinearBasis &l1, const LinearBasis &l2)
94 {
95     //得到两个线性基的交
96     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] = l1l << 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 & (l1l << 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 & (l1l << 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 >= 1; 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    {

```

```

11     int lenT = strlen(T + 1), p = 1, pos;
12     next[1] = lenT; // 对于 next[1] 要特殊考虑
13     while (p + 1 <= lenT && T[p] == T[p + 1])
14         ++p;
15     next[pos = 2] = p - 1; // next[2] 是为了初始化
16
17     for (int i = 3; i <= lenT; i++)
18     { // 注意此时 k + 1 = i
19         int len = next[i - pos + 1];
20         if (len + i < p + 1)
21             next[i] = len; // 对应上面第一种情况
22         else
23         {
24             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
25             while (i + j <= lenT && T[j + 1] == T[i + j])
26                 ++j; // 第二种需要暴力匹配
27             p = i + (next[pos = i] = j) - 1; // 记得更新 p, pos
28         }
29     }
30 }
31 void ExKMP()
32 {
33     int lenS = strlen(S + 1), lenT = strlen(T + 1), p = 1, pos;
34     Get_Next();
35     while (p <= lenT && S[p] == T[p])
36         ++p;
37     p = extend[pos = 1] = p - 1; // 初始化 extend[1]
38
39     for (int i = 2; i <= lenS; i++)
40     {
41         int len = next[i - pos + 1];
42         if (len + i < p + 1)
43             extend[i] = len;
44         else
45         {
46             int j = max(p - i + 1, 0);
47             while (i + j <= lenS && j <= lenT && T[j + 1] == S[i + j])
48                 ++j;
49             p = i + (extend[pos = i] = j) - 1;
50         }
51     } // 和上面基本一模一样啦
52 }
53 } sol;
54 int main()
55 {
56     scanf("%s", sol.S + 1);
57     scanf("%s", sol.T + 1);
58
59     sol.ExKMP();
60     int len = strlen(sol.S + 1);
61     for (int i = 1; i <= len; i++)
62         printf("%d%c", sol.extend[i], i == len ? '\n' : ' ');
63
64     return 0;
65 }

```

## 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{//lbase
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 lbase
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      void init(){
9          n = strlen(s + 1);
10         m = 300;

```



```

11  for(int i=0;i<=m;i++) c[i]=0;
12  for(int i=0;i<=2*n;i++) y[i]=0;
13  for(int i=1;i<=n;i++) c[rk[i]=s[i]]++;
14  for(int i=1;i<=m;i++) c[i]+=c[i-1];
15  for(int i=n;i>=1;i--) sa[c[rk[i]]--]=i;
16  for(int k=1;k<=n;k<=1){
17      int p=0;
18      for(int i=n-k+1;i<=n;i++) y[++p]=i;
19      for(int i=1;i<=n;i++){
20          if(sa[i]>k){
21              y[++p]=sa[i]-k;
22          }
23      }
24      for(int i=0;i<=m;i++) c[i]=0;
25      for(int i=1;i<=n;i++) c[rk[i]]++;
26      for(int i=1;i<=m;i++) c[i]+=c[i-1];
27      for(int i=n;i>=1;i--) sa[c[rk[y[i]]]--]=y[i];
28      for(int i=0;i<=n;i++) swap(rk[i],y[i]);
29      rk[sa[1]]=p=1;
30      for(int i=2;i<=n;i++){
31          rk[sa[i]]=(y[sa[i]] == y[sa[i-1]] && y[sa[i]+k] == y[sa[i-1]+k] ? p : ++p);
32      }
33      if(p>=n) break;
34      m=p;
35  }
36  for(int i=1,k=0;i<=n;i++){
37      if(k)k--;
38      int j=sa[rk[i]-1];
39      while(s[i+k] == s[j+k])k++;
40      ht[rk[i]] = k;
41  }
42  }
43  void writ()
44  {
45      printf("%s\n",s+1);
46      for(int i=1;i<=n;i++)cout<<sa[i]<<" ";cout<<"\n";
47      for(int i=1;i<=n;i++)cout<<ht[i]<<" ";cout<<"\n";
48      for(int i=1;i<=n;i++)cout<<rk[i]<<" ";cout<<"\n";
49  }
50
51  };
52  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 }
```

## 图论

### johnson 全源最短路

```
1 struct graph {
2     vector<vector<pair<int, ll>>> e;
3     graph(int n) : e(n + 1) {}
4     void adde(int u, int v, ll w) { e[u].push_back({v, w}); }
5     vector<ll> h;
6     // initialize h(u), return false if there exists a negative cycle
7     bool init() {
8         int n = e.size();
9         h.assign(n, 0);
10        queue<int> que;
11        for (int u = 1; u < n; u++) que.push(u);
12        vector<int> vis(n, 0), cnt(n, n + 1);
13        while (que.size()) {
14            auto u = que.front();
15            que.pop();
16            vis[u] = false;
17            if (!cnt[u]--) return false; // exists a negative cycle
18            for (auto &[v, w] : e[u])
19                if (h[v] > h[u] + w) {
20                    h[v] = h[u] + w;
21                    if (!vis[v]) que.push(v), vis[v] = 1;
22                }
23        }
24        return true;
25    }
26    // single source shortest path from given sink based on h(u)
27    vector<ll> query(int s) {
28        int n = e.size();
29        vector<ll> dis(n, inf);
30        priority_queue<pair<ll, int>, vector<pair<ll, int>>,
31            greater<pair<ll, int>>>
32            que;
33        que.push({dis[s] = 0, s});
34        while (que.size()) {
35            auto [du, u] = que.top();
36            que.pop();
37            if (dis[u] < du) continue;
38            for (auto [v, w] : e[u]) {
39                auto dv = du + w + h[u] - h[v];
40                if (dis[v] > dv) que.push({dis[v] = dv, v});
41            }
42        }
43        for (int i = 0; i < n; i++) dis[i] += h[i] - h[s];
44        return dis;
45    }
46 };
```

### kosaraju

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 const int maxn = 100010;
4 vector<int> ve[maxn], ve2[maxn];
5 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 }

```

### 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     }

```

## 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;
15            memset(dis, 0x3f, sizeof(dis));
16            q.push(s), fl[s] = 1e9, dis[s] = 0;
17            while (!q.empty()) {
18                int t = q.front();

```

```

19     q.pop(), in[t] = 0;
20     for (int i = hd[t]; i; i = nxt[i]) {
21         int it = to[i], d = dis[t] + cst[i];
22         if (limit[i] && d < dis[it]) {
23             dis[it] = d, fl[it] = min(fl[t], limit[i]), fr[it] = i;
24             if (!in[it]) in[it] = 1, q.push(it);
25         }
26     }
27 }
28 if (dis[t] > 1e9) return make_pair(flow, cost);
29 flow += fl[t], cost += dis[t] * fl[t];
30 for (int u = t; u != s; u = to[fr[u] ^ 1])
31     limit[fr[u]] -= fl[t], limit[fr[u] ^ 1] += fl[t];
32 }
33 }
34 };
35 struct bounded_flow {
36     int e, u[M], v[M], lo[M], hi[M], cst[M];
37     void add(int _u, int _v, int w, int c) {
38         if (c < 0) {
39             u[++e] = _u, v[e] = _v, lo[e] = w, hi[e] = w, cst[e] = c;
40             u[++e] = _v, v[e] = _u, lo[e] = 0, hi[e] = w, cst[e] = -c;
41         } else
42             u[++e] = _u, v[e] = _v, lo[e] = 0, hi[e] = w, cst[e] = c;
43     }
44     flow g;
45     pair<int, int> mincost(int n, int s, int t, int ss, int tt) {
46         static int w[N];
47         memset(w, 0, sizeof(w));
48         int flow = 0, cost = 0, tot = 0;
49         for (int i = 1; i <= e; i++) {
50             w[u[i]] -= lo[i], w[v[i]] += lo[i];
51             cost += lo[i] * cst[i];
52             g.add(u[i], v[i], hi[i] - lo[i], cst[i]);
53         }
54         for (int i = 1; i <= n; i++)
55             if (w[i] > 0)
56                 g.add(ss, i, w[i], 0), tot += w[i];
57             else if (w[i] < 0)
58                 g.add(i, tt, -w[i], 0);
59         g.add(t, s, 1e9, 0);
60         pair<int, int> res = g.mincost(ss, tt);
61         cost += res.second;
62         flow += g.limit[g.hd[s]];
63         g.hd[s] = g.nxt[g.hd[s]], g.hd[t] = g.nxt[g.hd[t]];
64         res = g.mincost(s, t);
65         return make_pair(flow + res.first, cost + res.second);
66     }
67 } f;
68 int n, m, s, t;
69 int main() {
70     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  lbase
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;
46                 else dom[e] = p;
47             }
48             if(i > 1) Union(i, rp[i]);
49         }
50     }
51 }

```

```

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] = -
14     c;
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],
39         limit[fr[u] ^ 1] += fl[t];
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
```

## 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==998244353)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>
```

```

5 #include <unordered_map>
6
7 using namespace std;
8
9 typedef long long LL;
10
11 const int INF = 0x3f3f3f3f;
12
13 int a, b, p;
14 unordered_map<int, int> hs;
15
16 int exgcd(int a, int b, int &x, int &y) {
17     if (!b) {
18         x = 1, y = 0;
19         return a;
20     }
21     int d = exgcd(b, a % b, y, x);
22     y -= a / b * x;
23     return d;
24 }
25
26 int BSGS(int a, int b, int p) {
27     if (1 % p == b % p) return 0;
28     int k = sqrt(p) + 1;
29     hs.clear();
30     for (int y = 0, r = b % p; y < k; y++) {
31         hs[r] = y;
32         r = (LL)r * a % p;
33     }
34     int ak = 1;
35     for (int i = 1; i <= k; i++) ak = (LL)ak * a % p;
36     for (int x = 1, l = ak; x <= k; x++) {
37         if (hs.count(l)) return k * x - hs[l];
38         l = (LL)l * ak % p;
39     }
40     return -INF;
41 }
42
43 int exBSGS(int a, int b, int p) {
44     b = (b % p + p) % p;
45     if (1 % p == b % p) return 0;
46     int x, y;
47     int d = exgcd(a, p, x, y);
48     if (d > 1) {
49         if (b % d) return -INF;
50         exgcd(a / d, p / d, x, y);
51         return exBSGS(a, (LL)b / d * x % (p / d), p / d) + 1;
52     }
53     return BSGS(a, b, p);
54 }
55
56 int main() {
57     while (~scanf("%d%d%d", &a, &p, &b), a || b || p) {
58         int res = exBSGS(a, b, p);
59         if (res < 0) puts("No Solution");
60         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 >= 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 >= 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 {
49     vector<cplx> w = UnitRoot::get_root(a.size() + b.size() - 1);
50     int n = w.size();
51     vector<cplx> A(n), B(n), C(n), D(n);
52     for (int i = 0; i < a.size(); ++i) A[i] = cplx(a[i] >> len, a[i] & mask);
53     for (int i = 0; i < b.size(); ++i) B[i] = cplx(b[i] >> len, b[i] & mask);
54     fft(A, w), fft(B, w);
55     for (int i = 0; i < n; ++i) {
56         int j = (n - i) % n;
57         cplx da = (A[i] - conj(A[j])) * cplx(0, -0.5), db = (A[i] + conj(A[j]))
58         * cplx(0.5, 0), dc = (B[i] - conj(B[j])) * cplx(0, -0.5), dd = (B[i] + conj(B[j])) *
59         cplx(0.5, 0);
60         C[j] = da * dd + da * dc * cplx(0, 1); D[j] = db * dd + db * dc * cplx(0,
61         1);
62     }
63     fft(C, w), fft(D, w);
64     vector<long long> res(a.size() + b.size() - 1);
65     for (int i = 0; i < res.size(); ++i) {
66         long long da = (long long)(C[i].imag() / n + 0.5) % mod, db = (long long)
67         (C[i].real() / n + 0.5) % mod, dc = (long long)(D[i].imag() / n + 0.5) % mod, dd = (long
68         long)(D[i].real() / n + 0.5) % mod;
69         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>=1;
12 {if(b&1)res=res*a%mod;a=a*a%mod;}return res;}
13 ll n;
14 namespace linear_seq {
15     const int N=10010;
16     ll res[N],base[N],_c[N],_md[N];
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 ((lll<<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