Antiamuny Code Library

tarjen

目录

第零章

- 安装 `typst`:
  - Linux, macOS, WSL

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

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

赛前准备

测样例脚本(python 版)

```python
import os
import sys
import zipfile

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

测样例脚本(bash 版)

对拍(cpp 版本)

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

对拍(bat 版本)

```bat
@echo off
setlocal enabledelayedexpansion

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

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

 goto loop
```

杂项

rand

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

time

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

}
```

子集枚举

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

    }
}
```

高维前缀和

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

三分

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

bitset

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

bitset 手写

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

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

```
  Bitset operator&(const Bitset &x)const{
    Bitset res;
    for(int i=0;i<=lim;i++) res.v[i]=v[i]&x.v[i];
    return res;
  }
  Bitset operator^(const Bitset &x)const{
    Bitset res;
    for(int i=0;i<=lim;i++) res.v[i]=v[i]^x.v[i];
    return res;
  }
  Bitset operator-(const Bitset &x)const{
    Bitset res;ull lst=0;
    for(int i=0;i<=lim;i++){
     ull cur=(v[i]<x.v[i]+lst);
     res.v[i]=v[i]-x.v[i]-lst;
      lst=cur;
   }
    return res;
  }
}
```

lcslen(n2w)

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

```
  Bitset operator&(const Bitset &x)const{
    Bitset res;
    for(int i=0;i<=lim;i++) res.v[i]=v[i]&x.v[i];
    return res;
  }
  Bitset operator^(const Bitset &x)const{
    Bitset res;
    for(int i=0;i<=lim;i++) res.v[i]=v[i]^x.v[i];
    return res;
  }
  Bitset operator-(const Bitset &x)const{
    Bitset res;ull lst=0;
    for(int i=0;i<=lim;i++){
     ull cur=(v[i]<x.v[i]+lst);
     res.v[i]=v[i]-x.v[i]-lst;
      lst=cur;
   }
    return res;
  }
}ch[26],f,g;
auto getans = [&](int mid){
    int l1=mid,l2=n-mid;
    f.init();g.init();
    for(int i=0;i<26;i++){
        ch[i].init();
    }
    for(int i=mid+1;i<=n;i++)ch[s[i]-'a'].add(i);
    for(int i=1;i<=mid;i++){
        g=f|ch[s[i]-'a'];
        f.shift1();
        f.add(1);
        f=g-f;
        f=f^g;f=f&g;
    }
    return f.count();
};
```

wqs 二分

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

```
        dp[x][1]=dp2[0]+info{a[x]+k,1};
    };
    dfs(1,0);
    return dp[1][0].second;
}
signed main()
{
    int l=-sum-10000000000000ll,r=sum+10000000000000ll;
    int ans=1e18;
    while(l<=r){
        int mid=(l+r)/2ll;
        if(solve(mid)>=m){
            ans=dp[1][0].first-m*mid;
            r=mid-1;
        }
        else l=mid+1;
    }
    if(ans!=(int)1e18)cout<<ans<<"\n";
    else cout<<"Impossible\n";
    return 0;
}
```

判断异或方程组是否有解

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

可以判断不同或相同的并查集

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

大数 int128

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

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

 ostream& operator << (ostream& out, __int128 num) {
```

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

树哈希

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

typedef unsigned long long ull;

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

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

const int N = 1e6 + 10;

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

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

```

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

```

求 **s** 所有前缀对于 **t** 的所有子串的 **lcs** 长度

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



int cas;

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

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

  for (int _ = 0; _ < q; _++) {
    int a, b, c;
```

```
    std::cin >> a >> b >> c;
    std::cout << solver.query(a, b, c) << '\n';
  }
}


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

    return 0;
}
```

线性基

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

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

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

```
    ll queryMin()
    {
        for(int i = 0; i <= maxbase; ++i)
            if(a[i])
                return a[i];
        return 0;
    }
    ll Kth_Max(ll k)
    {
        ll res=0;
        for(int i=62;i>=0;i--)
            if (k&(1LL<<i)) res^=a[i];
        return res;
    }
};

```

维护多个二维向量能够表示的范围

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

字符串

AC 自动机

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

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

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

Dequehash

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

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

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

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

```

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

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

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

Exkmp

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

    char S[maxn], T[maxn];
    int next[maxn], extend[maxn];
    void Get_Next()
    {
```

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

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

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

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

    return 0;
}
```

Hash

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

Kmp

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

```
        nxt[1]=0;
        int x=2,now=1;//x 是 s2 当前搜索到的位置，now 是前缀位置
        while(x<=n){
            if(s[x]==s[now]){
                nxt[x]=now;
                now++;
                x++;
            }
            else{
                if(now>1){
                    now=nxt[now-1]+1;
                }
                else{
                    nxt[x]=0;
                    now=1;
                    x++;
                }
            }
        }
    }
    void find(){//s1 1base
        int now=1,tar=1;
        len1=strlen(s1+1);
        while(tar<=len1){
            if(s1[tar]==s[now]){
                tar++;
                now++;
            }
            else{
                if(now>1){
                    now=nxt[now-1]+1;
                }
                else tar++;
            }
            if(now==n+1){
                printf("%d\n",tar-now+1);
            }
        }
    }
};
KMP sol;
int main()
{
    scanf("%s%s",sol.s1+1,sol.s+1);
    sol.build();
    sol.find();
    for(int i=1;i<=sol.n;i++)cout<<sol.nxt[i]<<" ";
    return 0;
}
```

Manacher

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

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

倍增 SA

```
#include<bits/stdc++.h>
using namespace std;
const int N = 1e6 + 10;//2*strlen
struct Suffix{
  int ht[N],rk[N],sa[N],y[N],c[N];
    int n,m;
    char s[N];
 void init(){
       n=strlen(s+1);
        m=300;
```

```
    for(int i=0;i<=m;i++) c[i]=0;
     for(int i=0;i<=2*n;i++) y[i]=0;
     for(int i=1;i<=n;i++) c[rk[i]=s[i]]++;
    for(int i=1;i<=m;i++) c[i]+=c[i-1];
     for(int i=n;i>=1;i--) sa[c[rk[i]]--]=i;
     for(int k=1;k<=n;k<<=1){
      int p=0;
      for(int i=n-k+1;i<=n;i++) y[++p]=i;
       for(int i=1;i<=n;i++){
        if(sa[i]>k){
          y[++p]=sa[i]-k;
         }
       }
      for(int i=0;i<=m;i++) c[i]=0;
       for(int i=1;i<=n;i++) c[rk[i]]++;
       for(int i=1;i<=m;i++) c[i]+=c[i-1];
       for(int i=n;i>=1;i--) sa[c[rk[y[i]]]--]=y[i];
       for(int i=0;i<=n;i++) swap(rk[i],y[i]);
       rk[sa[1]]=p=1;
      for(int i=2;i<=n;i++){
        rk[sa[i]]=(y[sa[i]] == y[sa[i-1]] && y[sa[i]+k] == y[sa[i-1]+k] ? p : ++p);
       }
       if(p>=n) break;
       m=p;
    }
     for(int i=1,k=0;i<=n;i++){
      if(k)k--;
       int j=sa[rk[i]-1];
      while(s[i+k] == s[j+k] )k++;
      ht[rk[i]] = k;
    }
   }
     void writ()
     {
         printf("%s\n",s+1);
         for(int i=1;i<=n;i++)cout<<sa[i]<<" ";;cout<<"\n";
         for(int i=1;i<=n;i++)cout<<ht[i]<<" ";;cout<<"\n";
         for(int i=1;i<=n;i++)cout<<rk[i]<<" ";;cout<<"\n";
     }

 };
 Suffix suf;
```

后缀自动机 SAM

```
#include <bits/stdc++.h>
using namespace std;
const int N=2e6;
struct SAM {
    struct Node {
        int tr[26];
        int len, fa;
        Node() { memset(tr, 0, sizeof(tr)); len = fa = 0; }
    }ep[N << 1];
     int last, tot, n;
     char base;
     vector<int> edg[N << 1];
```

```
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

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

```
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

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

kosaraju

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

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

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

```

K 短路

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

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

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

constexpr int N(2.5e5 + 5);

using T = LL;
struct Edge {
  int x, y; T z;
};
struct Heap {
```

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


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

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

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

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

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

```cpp

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

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

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

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

     res.push_back(w);

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

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

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

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

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

```

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

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

  int t = 1;

  // std::cin >> t;

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

lca(o1)

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

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

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

maxflow 只算值版本

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

```
            while((res=dfs(st,ed,inf))>0){
                flow+=res;
            }
        }
    }
};
```

maxflow 网络流最大流

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

```
    void addedge(int a,int b,int v){
        Addedge(a,b,v);
        Addedge(b,a,0);
 }
  int dinic(){
   int flow=0;
    while(bfs()){
      for(int i=S;i<=T;i++)cur[i]=head[i];
     flow+=dfs(S,1e9);
    }
    return flow;
 }
};
```

tarjan 缩点

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

二分图匹配

```
#include <bits/stdc++.h>
using namespace std;
int const N = 1510 * 4, M = 75010;
int e[M], ne[M], h[N], idx;
int n, m, k, match[N], backup[N], st[N];
void add(int a, int b) {
    e[idx] = b, ne[idx] = h[a], h[a] = idx++;
}
int find(int x) {
    for (int i = h[x]; ~i; i = ne[i]) {
        int j = e[i];
        if (!st[j]) {
```

```
            st[j] = 1;
            if (!match[j] || find(match[j])) {
                match[j] = x;
                return 1;
            }
        }
    }
    return 0;
}
int main() {
    cin >> n >> m >> k;
    memset(h, -1, sizeof h);
    for (int i = 1, a, b; i <= k; ++i) {
        scanf("%d%d", &a, &b);
        add(a, b + n);
    }
    int maxMatch = 0;
    for (int i = 1; i <= n; ++i) {
        memset(st, 0, sizeof st);
        if (find(i)) maxMatch++;
    }
    cout<<maxMatch<<endl;
    return 0;
}

```

二分图最优匹配

```
#include<bits/stdc++.h>
using namespace std;
const int maxn=110;
int n, m;
int a[maxn][maxn];
int lx[maxn], ly[maxn], link[maxn];
bool vx[maxn], vy[maxn];
int dfs(int x)
{
    if(x==-1)return 0;
    vx[x] = 1;
    for (int i = 1; i <= n; i++)
    {
        if (!vy[i] && lx[x] + ly[i] == a[x][i])
        {
            vy[i] = 1;
            if (link[i] == -1 || dfs(link[i]))
            {
                link[i] = x;
                return 1;
            }
        }
    }
    return 0;
}
bool deal()
{
    memset(ly, 0, sizeof(ly));
    memset(lx, 0xf7, sizeof(lx));
```

```
    memset(link, -1, sizeof(link));
    for (int i = 1; i <= n; i++)
    {
        for (int j = 1; j <= n; j++)
            lx[i] = max(lx[i], a[i][j]);
    }
    for(int i = 1; i <= n; i++)
    {
        while(1)
        {
            memset(vx, 0, sizeof(vx));
            memset(vy, 0, sizeof(vy));
            if (dfs(i)) break;
            int delta = 0x7f7f7f7f;
            for (int j= 1; j <= n; j++)
            {
                if (vx[j] == 1)
                    for(int k = 1; k <= n; k++)
                        if (vy[k] == 0) delta = min(delta, lx[j]+ ly[k]- a[j]
[k]);
            }
            if (delta == 0x7f7f7f7f)  return 0;
            for (int j= 1; j <= n; j++)
                if (vx[j] == 1) lx[j] -= delta;
            for(int k = 1; k <= n; k++)
                if (vy[k] == 1) ly[k] += delta;
        }
    }
    return 1;
}
int main()
{
       if  (deal() == 1) {
           int ans = 0;
           for(int i = 1; i <= n; i++)
           {
               ans += a[link[i]][i];
           }
           cout << ans <<'\n';//取最小就把所有边权取负再跑
       }
    return 0;
}
```

二分图染色

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

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

圆方树

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

    int now = 0;
    vector<int> dfn(n, -1), low(n);
    vector<int> stk;
    function<void(int)> tarjan = [&](int u) {
        stk.push_back(u);
        dfn[u] = low[u] = now++;
        for (auto v : ve[u]) {
            if (dfn[v] == -1) {
                tarjan(v);
                low[u] = min(low[u], low[v]);
```

```
                if (low[v] == dfn[u]) {
                    e1.push_back({});
                    int x;
                    do {
                        x = stk.back();
                        stk.pop_back();
                        e1[cnt].push_back(x);
                    } while (x != v);
                    e1[u].push_back(cnt);
                    ++cnt;
                }
            } else {
                low[u] = min(low[u], dfn[v]);
            }
        }
    };
    tarjan(0);
```

基环树

```
#include <bits/stdc++.h>
using namespace std;
const int maxn=2e5+10;
/*
1 init
2 addedge
3 Get
*/
struct Graph{
    vector<int>ve[maxn];
    int base[maxn],id[maxn];
    bool Incircle[maxn];
    vector<int> Circle;
    int len=0;
    int dep[maxn],f[21][maxn];
    int n;
    void init(int _n){
        n=_n;
        for(int i=1;i<=n;i++){
            for(int j=0;j<21;j++)f[j][i]=0;
            ve[i].clear();
            Incircle[i]=false;
            id[i]=-1;
            base[i]=i;
            Circle.clear();
            len=0;
            dep[i]=0;
        }
    }
    void addedge(int x,int y){
        ve[x].push_back(y);
        ve[y].push_back(x);
    }
    void dfs(int x,int fa)
    {
        base[x]=base[fa];
        dep[x]=dep[fa]+1;
```

```
        for(int i=0;i<=19;i++)
            f[i+1][x]=f[i][f[i][x]];
        for(auto it:ve[x])
        {
            if(it==fa) continue;
            f[0][it]=x;
            dfs(it,x);
        }
    }
    void Get(){
        vector<int> sta;
        vector<bool> vis(n+1,false);
        function<bool(int,int)> dfs2 = [&](int x,int h){
            vis[x]=true;
            sta.push_back(x);
            for(auto it:ve[x])if(it!=h){
                if(vis[it]){
                    Circle.push_back(it);
                    while(!sta.empty()&&sta.back()!=it){
                        Circle.push_back(sta.back());
                        sta.pop_back();
                    }
                    return true;
                }
                else{
                    if(dfs2(it,x))return true;
                }
            }
            sta.pop_back();
            return false;
        };
        dfs2(1,0);
        len=(int)Circle.size();
        for(auto it:Circle)Incircle[it]=true;
        for(auto it:Circle){
            for(auto it2:ve[it])if(!Incircle[it2]){
                f[0][it2]=it;
                dfs(it2,it);
            }
        }
        for(int i=0;i<len;i++)id[Circle[i]]=i;
    }
    int lca(int x,int y)
    {
        if(dep[x]<dep[y]) swap(x,y);
        for(int i=20;i>=0;i--)
        {
            if(dep[f[i][x]]>=dep[y]) x=f[i][x];
            if(x==y) return x;
        }
        for(int i=20;i>=0;i--)
            if(f[i][x]!=f[i][y])
                x=f[i][x],y=f[i][y];
        return f[0][x];
    }
    int dis(int x,int y){
```

```
        if(base[x]==base[y]){
            int l=lca(x,y);
            return dep[x]+dep[y]-2*dep[l];
        }
        else{
            int g=(id[base[x]]-id[base[y]]+len)%len;
             return dep[x]+dep[y]+min(g,len-g);
         }
     }
 };
 Graph g;
```

带权并查集 **dsu**

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

带花树

```
#include <bits/stdc++.h>
using namespace std;
struct blossom {//0base
    int n, vis_t;
    vector<vector<int>> E;
    vector<int> match, label, org, vis, parent;
    queue<int> Q;
    blossom(int _n) {
        n = _n;
         E = vector<vector<int>>(n, vector<int>());
         match.assign(n, -1);
         label.resize(n);
         org.resize(n);
         iota(org.begin(), org.end(), 0);
         parent.assign(n, -1);
         vis.assign(n, 0);
         vis_t = 0;
     }
     void addEdge(int u, int v) {
         E[u].emplace_back(v);
         E[v].emplace_back(u);
     }
     auto lca(int v, int u) {
         vis_t++;
         while (true) {
             if (v != -1) {
                 if (vis[v] == vis_t) {
```

```
                    return v;
                }
                vis[v] = vis_t;
                if (match[v] == -1) {
                    v = -1;
                } else {
                    v = org[parent[match[v]]];
                }
            }
            swap(v, u);
        }
    }
    void agument(int v) {
        while (v != -1) {
            auto pv = parent[v];
            auto nxt = match[pv];
            match[v] = pv;
            match[pv] = v;
            v = nxt;
        }
    }
    void flower(int v, int u, int a) {
        while (org[v] != a) {
            parent[v] = u;
            u = match[v];
            if (label[u] == 1) {
                label[u] = 0;
                Q.emplace(u);
            }
            org[v] = org[u] = a;
            v = parent[u];
        }
    }
    auto bfs(int root) {
        fill(label.begin(), label.end(), -1);
        iota(org.begin(), org.end(), 0);
        while (!Q.empty()) {
            Q.pop();
        }
        Q.emplace(root);
        label[root] = 0;
        while (!Q.empty()) {
            auto u = Q.front();
            Q.pop();
            for (auto v : E[u]) {
                if (label[v] == -1) {
                    label[v] = 1;
                    parent[v] = u;
                    if (match[v] == -1) {
                        agument(v);
                        return true;
                    }
                    label[match[v]] = 0;
                    Q.push(match[v]);
                    continue;
                } else if (label[v] == 0 && org[v] != org[u]) {
```

```
                    auto a = lca(org[u], org[v]);
                    flower(v, u, a);
                    flower(u, v, a);
                }
            }
        }
        return false;
    }
    void solve() {
        for (int i = 0; i < n; ++i) {
            if (match[i] == -1) {
                bfs(i);
            }
        }
    }
};
 int main()
 {
     blossom G(n);
     for (int i = 0; i < n; ++i) {
         for (int j = i + 1; j < n; ++j) {
             auto [xi, yi] = stone[i];
             auto [xj, yj] = stone[j];
             if (abs(xi - xj) + abs(yi - yj) <= L) {
                 G.addEdge(i, j);
             }
         }
     }
     G.solve();
     int num = 0;
     for (int i = 0; i < n; ++i) {
         if (G.match[i] != -1) {
             num++;
         }
     }
 }
```

带负环最小费用最大流

```
#include <bits/stdc++.h>
using namespace std;
const int N = 200 + 5, M = 2e4 + N;
struct flow {
  int cnt = 1, hd[N], nxt[M << 1], to[M << 1], limit[M << 1], cst[M << 1];
  void add(int u, int v, int w, int c) {
    nxt[++cnt] = hd[u], hd[u] = cnt, to[cnt] = v, limit[cnt] = w, cst[cnt] = c;
    nxt[++cnt] = hd[v], hd[v] = cnt, to[cnt] = u, limit[cnt] = 0, cst[cnt] = -c;
  }
  int fl[N], fr[N], dis[N], in[N];
  pair<int, int> mincost(int s, int t) {
    int flow = 0, cost = 0;
    while (1) {
      queue<int> q;
      memset(dis, 0x3f, sizeof(dis));
      q.push(s), fl[s] = 1e9, dis[s] = 0;
      while (!q.empty()) {
        int t = q.front();
```

```
        q.pop(), in[t] = 0;
        for (int i = hd[t]; i; i = nxt[i]) {
          int it = to[i], d = dis[t] + cst[i];
          if (limit[i] && d < dis[it]) {
            dis[it] = d, fl[it] = min(fl[t], limit[i]), fr[it] = i;
            if (!in[it]) in[it] = 1, q.push(it);
          }
        }
      }
      if (dis[t] > 1e9) return make_pair(flow, cost);
      flow += fl[t], cost += dis[t] * fl[t];
      for (int u = t; u != s; u = to[fr[u] ^ 1])
        limit[fr[u]] -= fl[t], limit[fr[u] ^ 1] += fl[t];
    }
  }
};
struct bounded_flow {
  int e, u[M], v[M], lo[M], hi[M], cst[M];
  void add(int _u, int _v, int w, int c) {
    if (c < 0) {
      u[++e] = _u, v[e] = _v, lo[e] = w, hi[e] = w, cst[e] = c;
      u[++e] = _v, v[e] = _u, lo[e] = 0, hi[e] = w, cst[e] = -c;
    } else
      u[++e] = _u, v[e] = _v, lo[e] = 0, hi[e] = w, cst[e] = c;
  }
  flow g;
  pair<int, int> mincost(int n, int s, int t, int ss, int tt) {
    static int w[N];
    memset(w, 0, sizeof(w));
    int flow = 0, cost = 0, tot = 0;
    for (int i = 1; i <= e; i++) {
      w[u[i]] -= lo[i], w[v[i]] += lo[i];
      cost += lo[i] * cst[i];
      g.add(u[i], v[i], hi[i] - lo[i], cst[i]);
    }
    for (int i = 1; i <= n; i++)
      if (w[i] > 0)
        g.add(ss, i, w[i], 0), tot += w[i];
      else if (w[i] < 0)
        g.add(i, tt, -w[i], 0);
    g.add(t, s, 1e9, 0);
    pair<int, int> res = g.mincost(ss, tt);
    cost += res.second;
    flow += g.limit[g.hd[s]];
    g.hd[s] = g.nxt[g.hd[s]], g.hd[t] = g.nxt[g.hd[t]];
    res = g.mincost(s, t);
    return make_pair(flow + res.first, cost + res.second);
  }
} f;
int n, m, s, t;
int main() {
  cin >> n >> m >> s >> t;
  for (int i = 1; i <= m; i++) {
    int u, v, w, c;
    cin >> u >> v >> w >> c, f.add(u, v, w, c);
  }
```

```
  pair<int, int> res = f.mincost(n, s, t, 0, n + 1);
  cout << res.first << " " << res.second << endl;
  return 0;
}

```

支配树

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

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

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

```
     for(int i=2;i<=cs;i++) if(sdom[i] != dom[i]) dom[i] = dom[dom[i]];
    for(int i=2;i<=cs;i++) up[RS[i]] = RS[dom[i]];
    return cs;
  }
 }
```

最小环

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

最小费用最大流

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

```
        nxt[++cnt] = hd[v], hd[v] = cnt, to[cnt] = u, limit[cnt] = 0, cst[cnt] = -c;
    }

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

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

欧拉回路

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

```
24      }
25      st.push(u);
26  }
27  int main()
28  {
29
30
31    return 0;
32  }
33
```

数学

组合数

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

BSGS 指数方程余数问题(求 a^x=b%p)

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

```cpp
#include <unordered_map>

using namespace std;

typedef long long LL;

 const int INF = 0x3f3f3f3f;

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

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

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

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

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

```
    }
    return 0;
}
```

EXGCD

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

FFT

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

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

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

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

FWT

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

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

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

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

   fwtxor(a, m, 1), fwtxor(b, m, 1), mul(a, b, c, m);
```

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

莫比乌斯反演

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

NTT

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

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

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

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

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

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

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

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

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

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

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

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

任意模数 NTT

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

```
            }
            int m = w.size() / n;
            vector<cplx> res(n);
            for (int i = 0, j = 0; i < n; i++, j += m) res[i] = w[j];
            return res;
        }
    };
    vector<cplx> UnitRoot::w;

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

```
        }
        return res;
    }
};
using namespace polynomial;
```

Pollard_Rho

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

```
    now = pre = rnd() % (n - 1) + 1;
   g = 1;
   ll c = rnd() % (n - 1) + 1;
    for (int i = 1, fst = 1 ;; i ++)
   {
      now = (mul(now, now, n) + c) % n;
      g = mul(g, abs(now - pre), n);
     if (now == pre || !g)break;
      if (!(i & 127) || i == fst)
      {
        g = gcd(g, n);
       if (g > 1)return g;
        if (i == fst)pre = now, fst <<= 1;
      }
     }
  }
}
void Find(ll n)
{
  if (n == 1)return ;
  if (Miller_Rabin(n))
 {
    P[n] = 1;
    return ;
 }
  ll p = Pollard_Rho(n);
 int c = 0;
 while (!(n % p))
 {
    n /= p, c ++;
  }
  Find(p);
 Find(n);
}
}
void solve(int x,set<int> &s){
    Pollard_Rho :: Find(x);
    for (auto [x, _] : P)s.insert(x);
    P.clear();
}
```

扩展中国剩余定理

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

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

拉格朗日插值

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

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

拉格朗日插值没有模数

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

杜教筛

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

}
int Sumg(int n){// g 的 n 项前缀和

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

```
}
```

线性筛质数

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

线性递推

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

     vector<int> Md;
     void mul(ll *a,ll *b,int k) {
         rep(i,0,k+k) _c[i]=0;
         rep(i,0,k) if (a[i]) rep(j,0,k) _c[i+j]=(_c[i+j]+a[i]*b[j])%mod;
         for (int i=k+k-1;i>=k;i--) if (_c[i])
             rep(j,0,siz(Md)) _c[i-k+Md[j]]=(_c[i-k+Md[j]]-_c[i]*_md[Md[j]])%mod;
         rep(i,0,k) a[i]=_c[i];
     }
     int solve(ll n,VI a,VI b) {
         ll ans=0,pnt=0;
         int k=siz(a);
         assert(siz(a)==siz(b));
         rep(i,0,k) _md[k-1-i]=-a[i];_md[k]=1;
         Md.clear();
         rep(i,0,k) if (_md[i]!=0) Md.push_back(i);
         rep(i,0,k) res[i]=base[i]=0;
         res[0]=1;
```

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

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

```
}
```

辛普森积分

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

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

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

高斯消元(模意义)

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

```
            for(j=col;j<var;j++)
                swap(a[k][j],a[max_r][j]);
            swap(x[k],x[max_r]);
        }
        x[k]=x[k]*ny(a[k][col])%mod;
        for(j=col+1;j<var;j++)a[k][j]=a[k][j]*ny(a[k][col])%mod;
        a[k][col]=1;
        for(i=0;i<equ;i++)
            if(i!=k){
                del(x[i],x[k]*a[i][col]%mod);
                for(j=col+1;j<var;j++)del(a[i][j],a[k][j]*a[i][col]%mod);
                a[i][col]=0;
            }
    }
    return 1;
}
```

高斯消元(浮点数)

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

计算几何