



# ACM 模板

Buns\_out & 无敌的爆哥

2023 年 6 月 2 日

## 目录

1	STL	5
1.1	手动开 O2	5
1.2	快读快写	5
1.3	set	6
1.4	string	6
1.5	STL 函数	6
1.6	rand()、测速	6
1.7	文件流	7
1.8	struct	7
1.9	vector	7
1.10	模拟退火	10
1.11	STL 函数	12
1.12	差分矩阵	13
1.13	前缀和矩阵	14
1.14	分治，最大字段和	14
2	字符串	17
2.1	kmp	17
2.2	exKMP	19
2.3	Manacher	20
2.4	字符串哈希	21
2.5	最小表示法	22
2.6	AC 自动机	23
2.7	后缀数组	26
2.8	后缀自动机	30
2.9	回文自动机	32
3	图论	35
3.1	差分约束	35
3.2	Floyd	35
3.3	2-SET	36
3.4	点双	38
3.5	边双	40
3.6	DINIC	41
3.7	MCMF	43
3.8	KM	45

3.9 最小生成树 Boruvka	47
3.10 基尔霍夫矩阵	49
3.11 全源最短路	51
3.12 全局最小割	54
4 数据结构	57
4.1 MultiplyLCA	57
4.2 ST	58
4.3 GrayCode	59
4.4 归并排序	60
4.5 树链剖分	61
4.6 扫描线线段树	66
4.7 莫队	69
5 数论	72
5.1 ExGcd	72
5.2 快速幂	72
5.3 整除分块	72
5.4 欧拉函数	73
5.5 等比数列求和	74
5.6 素数	74
5.7 排列组合其一	75
5.8 排列组合其二	76
5.9 排列组合其三	77
5.10 单调队列维护 DP	78
5.11 欧拉定理	79
5.12 拓展欧拉定理	79
5.13 排列组合隔板法	79
5.13.1 当你有 $n$ 个球分成 $m$ 份（每一份大于 $0$ ）时：	79
5.13.2 当你有 $n$ 个球分成 $m$ 份（每一份大于等于 $0$ ）时：	79
5.14 拓展中国剩余定理	80
5.15 向量旋转 $A$ 度时对应坐标为	80
5.16 威尔逊定理	80
5.17 泰勒展开	81
5.18 康托展开	82
5.19 逆康托展开	84
5.20 矩阵	85
5.20.1 矩阵求逆	85

5.20.2 高斯消元 . . . . .	88
5.20.3 求行列式值 . . . . .	90
5.21 拓展中国剩余定理 (CRT) . . . . .	92
5.22 拓展欧拉定理 . . . . .	94
5.23 卢卡斯求组合数 . . . . .	96
5.24 错排 . . . . .	98
5.25 高精度 . . . . .	100
5.26 高精度 GCD . . . . .	105
5.27 FFT(快速傅里叶变换) . . . . .	109
5.28 NTT (快速数论变换) . . . . .	111
5.29 多项式求逆 (ntt) . . . . .	113
5.30 bitset 处理多维偏序 . . . . .	115
6 计算几何 . . . . .	118
6.1 绕点旋转后坐标 . . . . .	118
6.2 正方形已知两点求另外两点 . . . . .	118
6.3 三角形 . . . . .	118
6.4 最小圆包括问题 . . . . .	121
6.5 俩个三维圆相交弦长 . . . . .	124
7 网络流建图 . . . . .	128
7.1 最小路径覆盖 . . . . .	128
7.1.1 最小不相交路径覆盖 . . . . .	128
7.1.2 最小可相交路径覆盖 . . . . .	128
7.1.3 最多不相交路径 . . . . .	128
7.2 最小割 . . . . .	128
7.2.1 最大权闭合子图 . . . . .	128
7.3 最小割 . . . . .	129
7.4 最大密度子图 . . . . .	129
7.5 二元关系最小割模型 . . . . .	130
7.6 二分图带权匹配 . . . . .	130
7.7 最大权不相交路径 . . . . .	130
7.8 不等式差分模型 (网络流解线性规划) . . . . .	130
7.9 有上下界的网络流 . . . . .	131
7.9.1 无源汇有上下界可行流 . . . . .	131
7.9.2 有源汇有上下界可行流 . . . . .	131
7.9.3 有源汇有上下界最大流和最小流 . . . . .	132
7.9.4 有源汇有上下界最小费用流 . . . . .	132

# 1 STL

## 1.1 手动开 O2

```
1 #pragma GCC optimize(2)
2 #pragma GCC optimize(3)
```

## 1.2 快读快写

```
1 inline void read(int &x){
2     int s = 0, w = 1; char ch = getchar();
3     while(ch < '0' || ch > '9'){ if(ch == '-') w = -1; ch =
        getchar(); }
4     while(ch >= '0' && ch <= '9') s = s * 10 + ch - '0', ch
        = getchar();
5     x = s*w;
6     return ;
7 }
8 inline void write(int x){
9     if(x<0){
10         putchar('-');
11         x=-x;
12     }
13     if(x>9)write(x/10);
14     putchar(x%10+'0');
15 }
```

```
1 unordered_map<int,int>mp;
2 unordered_set<int>s;
3 multiset<int>S;
4
5 int x=floor(0.1);//xia
6 int y=ceil(1.0);//shang
```

### 1.3 set

```
1 unordered_set<int>s;  
2 multiset<int>S;  
3 fl=lower_bound(a,a+13,x)-a;//在从小到大的序列中找出大于等于x的数的地址  
4 fl=upper_bound(a,a+13,x)-a;//在从小到大的序列中找出大于x的地址  
5  
6 fl=lower_bound(b,b+13,x,greater<int>())-b;//在从大到小的序列中找出小于等于x的数的地址  
7 fl=upper_bound(b,b+13,x,greater<int>())-b;//在从小到大的序列中找出小于x的数的地址  
8 s.erase(s.find(tmp));
```

### 1.4 string

```
1 string a=string(10,'9');  
2 int x=stoll(a);  
3 a=to_string(x);
```

### 1.5 STL 函数

```
1 next_permutation(p+1,p+1+9);  
2 prev_permutation(p+1,p+1+9);
```

### 1.6 rand()、测速

```
1 srand(time(0));  
2 clock_t start, finish;start = clock();  
3 finish = clock();  
4 solve();  
5 cout << "the_time_cost_is" <<double(finish - start)/  
    CLOCKS_PER_SEC;
```

## 1.7 文件流

```
1 freopen("C:\\Users\\tob\\Desktop\\in.txt","r",stdin);
2 freopen("C:\\Users\\tob\\Desktop\\out.txt","w",stdout);
3 fclose(stdin);
4 fclose(stdout);
```

## 1.8 struct

```
1 struct node
2 {
3     int sum,ls,rs,ts;
4     node():sum(0),ls(0),rs(0),ts(0){}
5     node(int a,int b,int c,int d):sum(a),ls(b),rs(c),ts(d)
6     {}
7 };
```

## 1.9 vector

```
1 int sum=accumulate(v.begin(),v.end(),0); // 求和
2 //去重
3 vector<int> vector_unique(vector<int>v ){
4     sort(v.begin(),v.end());
5     auto v_it = unique(v.begin(),v.end());
6     if(v_it != v.end())
7         v.erase(v_it,v.end());
8     return v;
9 }
10 //两个vector求并集
11 vector<int> vector_set_union(vector<int>v1 ,vector<int>v2)
12 {
13     vector<int>v;
14     sort(v1.begin(),v1.end());
15     sort(v2.begin(),v2.end());
```

```
15     set_union(v1.begin(),v1.end(),v2.begin(),v2.end(),
               back_inserter(v));
16     return v;
17 }
18 //两个vector求交集
19 vector<int> vector_set_intersection(vector<int>v1 ,vector<
    int>v2){
20     vector<int>v;
21     sort(v1.begin(),v1.end());
22     sort(v2.begin(),v2.end());
23     set_intersection(v1.begin(),v1.end(),v2.begin(),v2.end
        (),back_inserter(v));
24     return v;
25 }
26 //判断vector的某一元素是否存在
27 bool is_element_in_vector(vector<int>v,int element){
28     // vector<int>::iterator it;
29     auto it=find(v.begin(),v.end(),element);
30     if (it!=v.end()){
31         return true;
32     }
33     else{
34         return false;
35     }
36 }
37 void Erase()
38 {
39     vector<int>v;
40     v.push_back(1); v.push_back(2);v.push_back(3);
41     //直接引用
42     v.back()--=1;
43     //删除最后一个元素
44     v.erase(v.end()-1,v.end());
45     //删除第一个元素
46     v.erase(v.begin(),v.begin()+1);
47     for(auto i:v)
48         cout<<i<<" ";cout<<endl;
```



```
49     //sum
50     int sum=accumulate(v.begin(),v.end(),0);
51     // cout<<sum<<endl;
52     string str= accumulate(
53         next(v.begin()),
54         v.end(),
55         to_string(v[0]),
56         [](string a,int b){
57             return a+'-'+to_string(b);
58         }
59     );
60     cout<<str<<endl;
61     int MA = *max_element(v.begin(),v.end());//取值
62     int MA_fl = max_element(v.begin(),v.end())-v.begin();
        //取下标
63
64     int MI = *min_element(v.begin(),v.end());//取值
65     int MI_fk = min_element(v.begin(),v.end())-v.begin();
        //取下标
66 }
```

## 1.10 模拟退火

```
1  const int maxn=1e6+100;
2  struct node{
3      double x,y,z;
4  }a[maxn];
5  int n;
6  struct SaDown{
7      //退火系数
8      #define down 0.996
9      double ansx,ansy,ansz,answ;
10     //评估函数
11     double energy(double x,double y,double z){
12         double r=0;
13         for(int i=1;i<=n;i++){
14             r=max(r,sqrt((x-a[i].x)*(x-a[i].x)+(y-a[i].y)
15                     *(y-a[i].y)+(z-a[i].z)*(z-a[i].z)));
16         }
17         return r;
18     }
19     void sa()
20     {
21         double t=3000;
22         while(t>1e-15)
23         {
24             double ex=ansx+(rand()*2-RAND_MAX)*t;
25             double ey=ansy+(rand()*2-RAND_MAX)*t;
26             double ez=ansz+(rand()*2-RAND_MAX)*t;
27             double ew=energy(ex,ey,ez);
28             double de=ew-answ;
29             if(answ>ew)
30             {
31                 ansx=ex;
32                 ansy=ey;
33                 ansz=ez;
34                 answ=ew;
35             }
36         }
37     }
38 }
```

```
35         else if(exp(-de/t)*RAND_MAX>rand())
36         {
37             ansx=ex;
38             ansy=ey;
39             ansz=ez;
40         }
41         t*=down;
42     }
43 }
44 void doit(){
45     //退火次数
46     for(int i=1;i<=4;i++)
47         sa();
48 }
49 }sa;
50 void solve()
51 {
52     cin>>n;
53     for(int i=1;i<=n;i++){
54         scanf("%lf%lf%lf",&a[i].x,&a[i].y,&a[i].z);
55         sa.ansx+=a[i].x;
56         sa.ansy+=a[i].y;
57         sa.ansz+=a[i].z;
58     }
59     sa.ansx/=n;
60     sa.ansy/=n;
61     sa.ansz/=n;
62     sa.answ=sa.energy(sa.ansx,sa.ansy,sa.ansz);
63     sa.doit();
64     printf("%.3lf\n",sa.answ);
65 }
```

## 1.11 STL 函数

```
1 next_permutation(p+1,p+1+9);
2 prev_permutation(p+1,p+1+9);
3 //数字里面的 1 的个数
4 cnt=__builtin_popcount(x);
5 cnt=__builtin_popcountll(X);
6 //二进制表示形式中末尾0的个数
7 cnt=__builtin_ctz(x);
8 cnt=__builtin_ctzll(X);
9 //二进制表示形式中前导0的个数
10 cnt=__builtin_clz(x);
11 cnt=__builtin_clzll(X);
12 //二进制表示形式中1的个数的奇偶性（偶：0，奇：1）
13 cnt=__builtin_parity(x);
14 cnt=__builtin_parityll(X);
15 //二进制表示形式中最后一个1在第几位
16 cnt= __builtin_ffs(x);
17 cnt= __builtin_ffsll(X);
18 //快速开平方
19 cnt=__builtin_sqrt(x);
```

## 1.12 差分矩阵

```
1 void insert(int x1,int y1,int x2,int y2,int c)
2 {
3     s[x1][y1]+=c;
4     s[x2+1][y2+1]+=c;
5     s[x1][y2+1]-=c;
6     s[x2+1][y1]-=c;
7 }
8 void solve()
9 {
10     cin>>n>>m>>q;
11     for(int i=1;i<=n;i++)
12         for(int j=1;j<=m;j++)
13             cin>>p[i][j],insert(i,j,i,j,p[i][j]);
14     while(q-->0)
15     {
16         int a,b,c,d,k;
17         cin>>a>>b>>c>>d>>k;
18         insert(a,b,c,d,k);
19     }
20     for(int i=1;i<=n;i++)
21     {
22         for(int j=1;j<=m;j++){
23             p[i][j]=p[i-1][j]+p[i][j-1]-p[i-1][j-1]+s[i][j];
24             cout<<p[i][j]<<" ";
25         }
26         cout<<endl;
27     }
28 }
```

### 1.13 前缀和矩阵

```
1 void solve(int a)
2 {
3     cin>>n>>m>>q;
4     for(int i=1;i<=n;i++)
5         for(int j=1;j<=m;j++){
6             cin>>p[i][j];
7             s[i][j]=s[i-1][j]+s[i][j-1]-s[i-1][j-1]+p[i][j];
8         }
9     while(q--){
10         {
11             int x1,x2,y1,y2;
12             cin>>x1>>y1>>x2>>y2;
13             cout<<s[x2][y2]-s[x2][y1-1]-s[x1-1][y2]+s[x1-1][y1-1]<<endl;
14         }
15     }
```

### 1.14 分治，最大字段和

```
1 int p[maxn];
2 int n,m;
3 //分治
4 int MaxIn(int *a)
5 {
6     vector<int>dp(n+1,0);
7     int ma=0;
8     for(int i=1;i<=n;i++){
9         {
10             if(dp[i-1]>0)
11                 {
12                     dp[i]=dp[i-1]+a[i];
13                 }
14             else dp[i]=a[i];
```

```
15         ma=max(ma,dp[i]);
16     }
17     return ma;
18 }
19 int MaxIn(int *a,int l,int r)
20 {
21     if(l==r)
22         return a[l]>0?a[l]:0;
23     int mid=(l+r)>>1;
24     //左边区间的最大子段和
25     int l_max = MaxIn(a,l,mid);
26     //右边区间的最大子段和
27     int r_max = MaxIn(a,mid+1,r);
28
29     int sum = 0;
30     int left_max = 0;
31     for(int i=mid;i>=l;i--)
32     {
33         sum+=a[i];
34         if(sum>left_max)
35             left_max = sum;
36     }
37
38     sum = 0;
39     int right_max = 0;
40     for(int i=mid+1;i<=r;i++)
41     {
42         sum+=a[i];
43         if(sum>right_max)
44             right_max = sum;
45     }
46     int res = left_max + right_max;
47     if(res < l_max)
48         res = l_max;
49     if(res < r_max)
50         res = r_max;
51     return res;
```

```
52 }
53 void solve()
54 {
55     cin>>n;
56     for(int i=1;i<=n;i++)
57         cin>>p[i];
58     cout<<MaxIn(p)<<endl;
59 }
```



## 2 字符串

### 2.1 kmp

```
1 struct KMP{
2     int nxt[maxn];
3     int len;
4     void init(char *s)
5     {
6         nxt[1]=0;
7         len=strlen(s+1);
8         for(int i=2;i<=len;i++)
9         {
10             nxt[i]=nxt[i-1];
11             while(s[nxt[i]+1]!=s[i]&&next[i])
12                 nxt[i]=nxt[nxt[i]];
13             if(s[nxt[i]+1]==s[i])nxt[i]++;
14         }
15     }
16     void kmp(char *s,char *t)
17     {
18         int lens=strlen(s+1);
19         int lent=strlen(t+1);
20         int cnt=0;
21         for(int i=0,j=0;i<lens;i++)
22         {
23             while(j&&s[i+1]!=t[j+1])j=nxt[j];
24             if(s[i+1]==t[j+1])j++;
25             if(j==lent)
26             {
27                 cnt++;
28                 j=nxt[j];
29             }
30         }
31     }
32     /* 循环周期 形如 acaca 中 ac 是一个合法周期 */
33     vector<int> periodic()
```

```
34     {
35         vector<int>ret;
36         int now=len;
37         while(now)
38         {
39             now=nxt[now];
40             ret.push_back(len-now);
41         }
42         return ret;
43     }
44     /* 循环节 形如 acac 中ac、acac是循环节，aca不是*/
45     vector<int> periodic_loop()
46     {
47         vector<int>ret;
48         for(auto i:periodic())
49         {
50             if(len%i==0)
51                 ret.push_back(i);
52         }
53         return ret;
54     }
55     void debug(){
56         for (int i=0;i<=len;i++){
57             printf("[debug]_nxt[%d]=%d\n",i,nxt[i]);
58         }
59     }
60 }kmp;
```

## 2.2 exKMP

对于个长度为  $n$  的字符串。定义函数表示  $z[i]$  表示  $s$  和  $s[i, n-1]$ （即以  $s[i]$  开头的后缀）的最长公共前缀（LCP）的长度。被称为的  $Z$  函数。

```

1 struct EXKMP{
2 //result: ext[i] = LCP(S[i,lens],T)
3 //require: nxt[i] = LCP(T[i,lent],T)
4 //nxt : s  Mode_String
5 //ext : s  Text_String t Mode_String
6 void exkmp(char *s,int lens,char *t,int lent,int *ext,int
    *nxt)
7 {
8     ext[0]=0;
9     for(int i=1,p0=0,p=0;i<=lens;i++)
10    {
11        ext[i]=i<=p?min(nxt[i-p0+1],p-i+1):0;
12        while(i+ext[i]<=lens&&ext[i]<lent&&s[i+ext[i]]==t[
            ext[i]+1])ext[i]++;
13        if(i+ext[i]-1>=p&&i!=1)p0=i,p=i+ext[i]-1;
            //最右端
14    }
15 }
16 }exKMP;
17 char s[maxn];
18 char t[maxn];
19 int nxt[maxn];
20 int ext[maxn];
21 void solve()
22 {
23     scanf("%s",t+1);
24     scanf("%s",s+1);
25     exKMP.exkmp(s,strlen(s+1),s,strlen(s+1),nxt,nxt);//
26     exKMP.exkmp(t,strlen(t+1),s,strlen(s+1),ext,nxt);
27 }
```

## 2.3 Manacher

```
1 struct Manacher{
2     char ch[maxn];
3     int lc[maxn];
4     int N;
5     void init(char *s){
6         int n=strlen(s+1); // puts(s+1);
7         ch[n*2+1]='#';
8         ch[0]='@';
9         ch[n*2+2]='\0';
10        for(int i=1;i<=n;i++)ch[i*2]=s[i],ch[i*2-1]='#';
11        N=n*2+1;
12    }
13    void manacher(){
14        lc[1]=1; int k=1,ma=1;
15        for (int i=2;i<=N;i++){
16            int p = k+lc[k]-1;
17            if (i<=p){lc[i]=min(lc[2*k-i],p-i+1);
18            }else{ lc[i]=1;}
19            while(ch[i+lc[i]]==ch[i-lc[i]])lc[i]++;
20            if(i+lc[i]>k+lc[k])k=i;
21            ma=max(ma,lc[i]-1);
22        }
23    }
24 }Manch;
25 char s[maxn];
26 void solve()
27 {
28     scanf("%s",s+1);
29     Manch.init(s);
30     Manch.manacher();
31 }
```

## 2.4 字符串哈希

```
1  const ull mod=21237044013013795711;
2  const ull prime=233317;
3  const ull base=131;
4  const ull more=19260817;
5  map<ull,bool>mp;
6  ull a[maxn];
7  int n;
8  ull get_hash(string &s)
9  {
10     ull ans=0;
11     for(int i=0;i<s.size();i++)
12         ans=(ans*base+(ull)s[i])%mod+prime;
13     return ans;
14 }
15 void solve()
16 {
17     cin>>n;
18     for(int i=1;i<=n;i++)
19     {
20         string str;
21         cin>>str;
22         a[i]=get_hash(str);
23     }
24     sort(a+1,a+1+n);
25     int ans=1;
26     for(int i=2;i<=n;i++)
27         if(a[i]!=a[i-1])
28             ans++;
29     cout<<ans<<endl;
30 }
```

## 2.5 最小表示法

```
1  int Get_Min(int *p,int n)
2  {
3      int i=0,j=1,k=0;
4      while(i<n&& j<n&&k<n)
5      {
6          if(p[(i+k)%n]==p[(j+k)%n])k++;
7          else
8          {
9              if(p[(i+k)%n]>p[(j+k)%n])i=i+k+1;
10             else j=j+k+1;
11             if(i==j)i++;
12             k=0;
13         }
14     }
15     return min(i,j);
16 }
17 string str;
18 int p[maxn];
19 int n;
20 void solve()
21 {
22     cin>>n;
23     for(int i=0;i<n;i++)
24         cin>>p[i];
25     int tmp=Get_Min(p,n);
26     // cout<<tmp<<endl;
27     for(int i=0;i<n;i++)
28         cout<<p[(i+tmp)%n]<<" ";
29 }
```

## 2.6 AC 自动机

```
1 string mp[maxn];
2 vector<string>v;
3 int n;
4 struct Trie{
5     int nxt[maxn][26],fail[maxn];
6     int val[maxn];
7     int cnt[maxn];
8     int id[maxn];
9     int q[maxn];
10    int root=0,indx=0;
11    int top=0,low=1;
12    void clear(){
13        for(int i=0;i<=indx;i++){
14            val[i]=cnt[i]=id[i]=q[i]=fail[i]=0;
15            for(int j=0;j<26;j++)
16                nxt[i][j]=0;
17        }
18        indx=top=0;low=1;//v.clear();
19    }
20    void insert(string& str,int x)
21    {
22        int rt=0;
23        for(int i=0;i<str.size();i++){
24            int tmp=str[i]-'a';
25            if(!nxt[rt][tmp])nxt[rt][tmp]=++indx;
26            rt=nxt[rt][tmp];
27        }
28        val[rt]++;id[x]=rt;
29    }
30    void build()
31    {
32        for(int i=0;i<26;i++)
33            if(nxt[0][i])
34                q[++top]=nxt[0][i];
35        while(low<=top){
```

```
36         int x=q[low++];
37         for(int i=0;i<26;i++){
38             int &rt=nxt[x][i];
39             if(!rt)rt=nxt[fail[x]][i];
40             else{
41                 fail[rt]=nxt[fail[x]][i];
42                 q[++top]=rt;
43             }
44         }
45     }
46 }
47 void query(string& s)
48 {
49     for(int i=0,j=0;i<s.size();i++){
50         j=nxt[j][s[i]-'a'];
51         cnt[j]++;
52     }
53     for(int i=indx;i;i--){
54         cnt[fail[q[i]]]+=cnt[q[i]];
55         //模式串i出现次数
56         // for(int i=1;i<=n;i++)
57         // cout<<cnt[id[i]]<<endl;
58         //模式串出现次数
59         // int ans=0;
60         // int now=0;
61         // for(int i=0;i<s.size();i++)
62         // {
63         //     now=nxt[now][s[i]-'a'];
64         //     for(int t=now;t&&val[t];t=fail[t])
65         //         ans+=val[t],val[t]=0;
66         // }
67         // cout<<ans<<endl;
68     }
69 }trie;
70 void solve()
71 {
72     while(cin>>n&&n)
```



```
73     {
74         trie.clear();
75         for(int i=1;i<=n;i++)
76         {
77             string str;
78             cin>>str;
79             mp[i]=str;
80             trie.insert(str,i);
81         }
82         trie.build();
83         string t;
84         cin>>t;
85         trie.query(t);
86         break;
87     }
88 }
```

## 2.7 后缀数组

```
1  #define maxn 1000050
2  char s[maxn];
3  int y[maxn],x[maxn],c[maxn],sa[maxn],rk[maxn],height[maxn],wt[30];
4  int mm[maxn];
5  int best[maxn][20];
6  int RMQ[maxn];
7  int n,m;
8  void putout(int x)
9  {
10     if(!x) {putchar(48);return;}
11     int l=0;
12     while(x) wt[++l]=x%10,x/=10;
13     while(l) putchar(wt[l--]+48);
14 }
15 void get_SA()
16 {
17     for (int i=1;i<=n;++i) ++c[x[i]=s[i]];
18     //c数组是桶
19     //x[i]是第i个元素的第一关键字
20     for (int i=2;i<=m;++i) c[i]+=c[i-1];
21     //做c的前缀和，我们就可以得出每个关键字最多是在第几名
22     for (int i=n;i>=1;--i) sa[c[x[i]]--]=i;
23     for (int k=1;k<=n;k<<=1)
24     {
25         int num=0;
26         for (int i=n-k+1;i<=n;++i) y[++num]=i;
27         //y[i]表示第二关键字排名为i的数，第一关键字的位置
28         //第n-k+1到第n位是没有第二关键字的 所以排名在最前面
29         for (int i=1;i<=n;++i) if (sa[i]>k) y[++num]=sa[i]-k;
30         //排名为i的数 在数组中是否在第k位以后
31         //如果满足(sa[i]>k) 那么它可以作为别人的第二关键字，就把它的第一关键字的位置添加进y就行了
```

```
32      //所以i枚举的是第二关键字的排名，第二关键字靠前的
      先入队
33      for (int i=1;i<=m;++i) c[i]=0;
34      //初始化c桶
35      for (int i=1;i<=n;++i) ++c[x[i]];
36      //因为上一次循环已经算出了这次的第一关键字 所以直
      接加就行了
37      for (int i=2;i<=m;++i) c[i]+=c[i-1]; //第一关键字排
      名为1~i的数有多少个
38      for (int i=n;i>=1;--i) sa[c[x[y[i]]]--]=y[i],y[i]
      ]=0;
39      //因为y的顺序是按照第二关键字的顺序来排的
40      //第二关键字靠后的，在同一个第一关键字桶中排名越靠
      后
41      //基数排序
42      swap(x,y);
43      //这里不用想太多，因为要生成新的x时要用到旧的，就
      把旧的复制下来，没别的意思
44      x[sa[1]]=1; num=1;
45      for (int i=2;i<=n;++i)
46          x[sa[i]]=(y[sa[i]]==y[sa[i-1]] && y[sa[i]+k]==
              y[sa[i-1]+k]) ? num : ++num;
47      //因为sa[i]已经排好序了，所以可以按排名枚举，生成
      下一次的第一个关键字
48      if (num==n) break;
49      m=num;
50      //这里就不用那个122了，因为都有新的编号了
51  }
52  for(int i=1;i<=n;i++)
53      cout<<sa[i]<<" "; cout<<endl;
54  }
55  void get_height()
56  {
57      int k=0;
58      for (int i=1;i<=n;++i) rk[sa[i]]=i;
59      for (int i=1;i<=n;++i)
60      {
```

```

61         if (rk[i]==1) continue;//第一名height为0
62         if (k) --k;//h[i]>=h[i-1]+1;
63         int j=sa[rk[i]-1];
64         while (j+k<=n && i+k<=n && s[i+k]==s[j+k]) ++k;
65         height[rk[i]]=k;//h[i]=height[rk[i]];
66     }
67     for(int i=1;i<=n;i++)RMQ[i]=height[i];
68     for(int i=1;i<=n;i++)
69         cout<<height[i]<<" ";cout<<endl;
70 }
71
72 void initST(int n){
73     mm[0]=-1;
74     for(int i=1;i<=n;i++)
75         mm[i]=((1<(i-1))&&0)?mm[i-1]+1:mm[i-1];
76     for(int i=1;i<=n;i++)best[i][0]=i;
77     for(int i=1;i<=mm[n];i++)
78     {
79         for(int j=1;j+(1<<i)-1<=n;j++){
80             int a=best[j][i-1];
81             int b=best[j+(1<<i-1)][j];
82             if(RMQ[a]<RMQ[b])best[j][i]=a;
83             else best[j][i]=b;
84         }
85     }
86 }
87 int queryST(int a,int b){
88     int t;
89     t=mm[b-a+1];
90     b--=(1<<t)-1;
91     a=best[a][t];
92     b=best[b][t];
93     return RMQ[a]<RMQ[b]?a:b;
94 }
95 int lcp(int a,int b){
96     a=rk[a];b=rk[b];
97     if(a>b)swap(a,b);

```

```
98     return height[queryST(a+1,b)];
99 }
100 void solve()
101 {
102     cin>>s+1;
103     n=strlen(s+1);m=122;
104     //n表示原字符串长度，m表示字符个数，ascii('z')=122
105     get_SA();
106     get_height();
107     initST(n);
108     cout<<lcp(1,3)<<endl;
109
110 }
```

## 2.8 后缀自动机

```
1  /*
2  请你求出 S 的所有出现次数不为1的子串的出现次数乘上该子串长
    度的最大值。
3  */
4  #include<bits/stdc++.h>
5  using namespace std;
6  using ll=long long;
7  const int N=3*1e6+10;
8  const int maxn=3e6+10;
9  char ch[N];
10 int nl;
11 ll res;
12 struct suffixautomation
13 {
14     int mp[maxn][30],fa[maxn],len[maxn],siz[maxn];
15     int ed,ct;
16     struct Edge{
17         int to,next;
18     }edge[maxn];
19     int head[maxn],cnt;
20     suffixautomation(){ed=ct=1;}
21     inline void add(int from,int to){
22         edge[++cnt].to=to;
23         edge[cnt].next=head[from];
24         head[from]=cnt;
25     }
26     inline void insert(int c)
27     {
28         int p=ed; siz[ed=++ct]=1; len[ed]=nl;//先初始化
            size和len
29         for(;p&&mp[p][c]==0;p=fa[p]){mp[p][c]=ed;}//然后顺
            着parent树的路径向上找
30         if(p==0){fa[ed]=1;return;}int q=mp[p][c];//case1
31         if(len[p]+1==len[q]){fa[ed]=q;return;}//case2
32         len[++ct]=len[p]+1;//case 3
```

```
33     for(int i=1;i<=26;i++)mp[ct][i]=mp[q][i];
34     fa[ct]=fa[q];fa[q]=ct;fa[ed]=ct;
35     for(int i=p;mp[i][c]==q;i=fa[i])mp[i][c]=ct;
36 }
37 inline void bt(){//暴力建树
38     for(int i=2;i<=ct;i++){
39         add(fa[i],i);
40     }
41 }
42 void dfs(int u)//dfs
43 {
44     for(int i=head[u];i;i=edge[i].next){
45         dfs(edge[i].to);
46         siz[u]+=siz[edge[i].to];
47     }
48     if(siz[u]!=1){
49         //个数乘长度
50         res=max(res,(ll)siz[u]*len[u]);
51     }
52 }
53 }sam;
54 int main()
55 {
56     scanf("%s",ch+1);//没啥好说的，建sam然后dfs
57     for(int i=1;ch[i]!='\0';i++){
58         sam.insert(ch[i]-'a'+1);
59     }
60     sam.bt();
61     sam.dfs(1);
62     printf("%lld",res);
63     return 0;//拜拜程序~
64 }
```

## 2.9 回文自动机

```
1  /*
2  注意由于要求以i结尾，所以串长两两不等，这里的回文子串个数
   即为本质不同回文子串个数
3  给定一个字符串 s。保证每个字符为小写字母。对于 s 的每个位
   置，请求出以该位置结尾的回文子串个数。
4  具体地，若第 i(i≥11)个位置的答案是k，第 i+1 个字符读入时的
   ASCII码为c，
5  则第i+1个字符实际的ASCII码为(c−97+k)mod26+97。所有字符在加
   密前后都为小写字母。
6  */
7  #include<bits/stdc++.h>
8  using namespace std;
9
10 const int N = 2e6 + 5;
11 struct PAM_Trie
12 {
13     int ch[26];
14     int fail, len, num;
15 };
16 struct PAM
17 {
18     PAM_Trie b[N];
19     int n, length, last, cnt, s[N];
20     char c[N];
21
22     PAM()
23     {
24         b[0].len = 0; b[1].len = -1;
25         b[0].fail = 1; b[1].fail = 0;
26         last = 0;
27         cnt = 1;
28     }
29     void read()
30     {
31         scanf("%s", c + 1);
```



```
32     length = strlen(c + 1);
33 }
34 int get_fail(int x)
35 {
36     while(s[n - b[x].len - 1] != s[n])
37     {
38         //printf("%d %d %d\n", x, n - b[x].len - 1, b[
39             x].fail);
39         x = b[x].fail;
40     }
41     return x;
42 }
43 void insert()
44 {
45     int p = get_fail(last);
46     if(!b[p].ch[s[n]])
47     {
48         b[++cnt].len = b[p].len + 2;
49         int tmp = get_fail(b[p].fail);
50         b[cnt].fail = b[tmp].ch[s[n]];
51         b[cnt].num = b[b[cnt].fail].num + 1;
52         b[p].ch[s[n]] = cnt;
53     }
54     last = b[p].ch[s[n]];
55 }
56 void solve()
57 {
58     int k = 0;
59     s[0] = 26;
60     for(n = 1; n <= length; n++)
61     {
62         c[n] = (c[n] - 97 + k) % 26 + 97;
63         s[n] = c[n] - 'a';
64         insert();
65         printf("%d ", b[last].num);
66         k = b[last].num;
67     }
```

```
68     }  
69 }P;  
70 signed main()  
71 {  
72     P.read();  
73     P.solve();  
74     return 0;  
75 }
```

## 3 图论

### 3.1 差分约束

```
1 a-b<=c add(b,a,c)
2 求最小生成树
3
4 a-b>=c add(b,a,c)
5 求最大生成树
6 或add(a,b,-c)
7
8 a-b==c
9 add(b,a,c)
10 add(a,b,-c)
```

### 3.2 Floyd

```
1 for(int k=1;k<=n;k++)
2     for(int i=1;i<=n;i++)
3         for(int j=1;j<=n;j++)
4             dis[i][j]=min(dis[i][j],dis[i][k]+dis[k][j]);
5 }
```

## 3.3 2-SET

```
1 void solve()
2 {
3     cin>>n>>m;
4     /*
5     如果指定 x 一定为真
6     add(x,x+n);
7     如果x为假那么x一定为真
8     */
9     for(int i=1;i<=m;i++)
10    {
11        int x,a,y,b;
12        cin>>x>>a>>y>>b;
13        if(a==1&&b==1)//x为真或者y为真
14        {
15            v[x].push_back(y+n);//x为假那么y一定为真
16            v[y].push_back(x+n);//y为假那么x一定为真
17        }
18        else if(a==0&&b==1)//x为假或者y为真
19        {
20            v[x+n].push_back(y+n);//x为真那么y一定为真
21            v[y].push_back(x);//y为假那么x一定为假
22        }
23        else if(a==1&&b==0)//x为真或者y为假
24        {
25            v[x].push_back(y);//x为假那么y一定为假
26            v[y+n].push_back(x+n);//y为真那么x一定为真
27        }
28        else if(a==0&&b==0)//x为假或者y为假
29        {
30            v[x+n].push_back(y);
31            v[y+n].push_back(x);
32        }
33    }
34    for(int i=1;i<=n*2;i++)
35        if(!dfn[i])
```

```
36         tarjan(i);
37         //判断无解情况
38         for(int i=1;i<=n;i++)
39             if(col[i]==col[i+n]){
40                 cout<<"IMPOSSIBLE"<<endl;
41                 return ;
42             }
43         //输出拓扑序靠后(col较小)
44         cout<<"POSSIBLE"<<endl;
45         for(int i=1;i<=n;i++)
46             cout<<(col[i]>col[i+n])<<" ";
47         cout<<endl;
48     }
```

### 3.4 点双

```
1  struct Edge{
2      int to,next,w;
3  }edge[maxn];
4  int head[maxn],cnt;
5  int dfn[maxn],low[maxn],indx;
6  vector<int>bcc[maxn];
7  int col[maxn],tot;
8  bitset<maxn>vis;
9  stack<int>s;
10 int bri[maxn];
11 int cut[maxn];
12 int n,m,bcc_cnt;
13 void add(int from,int to)
14 {
15     edge[++cnt].to=to;
16     edge[cnt].next=head[from];
17     head[from]=cnt;
18 }
19 void tarjan(int x,int fa)
20 {
21     dfn[x]=low[x]=++indx;
22     vis[x]=1;
23     s.push(x);
24     int num=0;
25     for(int i=head[x];i;i=edge[i].next)
26     {
27         int y=edge[i].to;
28         if(!dfn[y])
29         {
30             num++;
31             tarjan(y,i);
32             low[x]=min(low[x],low[y]);
33             if(dfn[x]<low[y])
34                 bri[i]=bri[i^1]=1;
35             if(low[y]>=dfn[x])
```

```
36         {
37             cut[x]=1;
38             bcc_cnt++;
39             bcc[bcc_cnt].push_back(x);
40             int v;
41             do{
42                 v=s.top();s.pop();
43                 bcc[bcc_cnt].push_back(v);
44             }while(v!=y);
45         }
46     }
47     else if(vis[y]&&dfn[y]<dfn[x]&&i!=(fa^1))
48         low[x]=min(low[x],dfn[y]);
49 }
50 if(fa==0&&num==1)cut[x]=0;
51 }
```

## 3.5 边双

```
1 void tarjan(int x,int fa)
2 {
3     dfn[x]=low[x]=++indx;
4     vis[x]=1;
5     s.push(x);
6     for(int i=head[x];i;i=edge[i].next)
7     {
8         int y=edge[i].to;
9         if(!dfn[y])
10        {
11            tarjan(y,i);
12            low[x]=min(low[x],low[y]);
13            if(dfn[x]<low[y])
14                bri[i]=bri[i^1]=1;
15        }
16        else if(vis[y]&&dfn[y]<dfn[x]&&i!=(fa^1))
17            low[x]=min(low[x],dfn[y]);
18    }
19    if(dfn[x]==low[x])
20    {
21        tot++;
22        int y;
23        do{
24            y=s.top();s.pop();
25            vis[y]=0;
26            col[y]=tot;
27        }while(x!=y);
28    }
29 }
```



## 3.6 DINIC

```
1  struct Edge{
2      int to,next,w;
3  }edge[maxn];
4  int head[maxn],cnt=1;
5  int dep[maxn];
6  int n,m,s,t,tot;
7  void init(){
8      memset(head+1,0,sizeof(int)*tot);
9      cnt=1;
10     tot=0;
11 }
12 void add(int from,int to,int w)
13 {
14     edge[++cnt].w=w;
15     edge[cnt].to=to;
16     edge[cnt].next=head[from];
17     head[from]=cnt;
18 }
19 bool bfs(int s,int t)
20 {
21     memset(dep+1,0,sizeof(int)*tot);
22     queue<int>q;
23     dep[s]=1;
24     q.push(s);
25     while(!q.empty())
26     {
27         int x=q.front();q.pop();
28         if(x==t)return true;
29         for(int i=head[x];i;i=edge[i].next)
30         {
31             int y=edge[i].to;
32             int w=edge[i].w;
33             if(dep[y]==0&&w>0)
34                 q.push(y),dep[y]=dep[x]+1;
35         }
```

```
36     }
37     return false;
38 }
39 int dfs(int x,int flow,int t)
40 {
41     if(x==t)return flow;
42     int out=flow;
43     for(int i=head[x];i;i=edge[i].next)
44     {
45         int y=edge[i].to;
46         int w=edge[i].w;
47         if(w!=0&&dep[y]==dep[x]+1)
48         {
49             int tmp=dfs(y,min(out,w),t);
50             edge[i].w-=tmp;
51             edge[i^1].w+=tmp;
52             out-=tmp;
53             if(!out)break;
54         }
55     }
56     if(out==flow)dep[x]=0;
57     return flow-out;
58 }
59 int DINIC(int s,int t)
60 {
61     int ans=0;
62     while(bfs(s,t))
63         ans+=dfs(s,INF,t);
64     return ans;
65 }
```

## 3.7 MCMF

```
1 struct node{
2     int to,next,w,c;
3 }edge[maxn];
4 int head[maxn],cnt;
5 int dis[maxn],flow[maxn],last[maxn];
6 bitset<maxn>vis;
7 int n,m,s,t;
8 void init(){
9     memset(head+1,0,sizeof(int)*n);
10    cnt=1;
11 }
12 void add(int from,int to,int w,int c)
13 {
14     edge[++cnt].w=w;
15     edge[cnt].c=c;
16     edge[cnt].to=to;
17     edge[cnt].next=head[from];
18     head[from]=cnt;
19 }
20 bool spfa(int s,int t)
21 {
22     memset(dis+1,INF,sizeof(int)*n);
23     memset(last+1,-1,sizeof(int)*n);
24     vis.reset();
25     queue<int>q;
26     q.push(s);
27     dis[s]=0;
28     vis[s]=1;
29     flow[s]=INF;
30     while(!q.empty())
31     {
32         int x=q.front();q.pop();
33         vis[x]=0;
34         for(int i=head[x];i;i=edge[i].next)
35         {
```

```
36         int y=edge[i].to;
37         int w=edge[i].w;
38         int c=edge[i].c;
39         if(w>0&&dis[y]>dis[x]+c)
40         {
41             dis[y]=dis[x]+c;
42             last[y]=i;
43             flow[y]=min(flow[x],w);
44             if(!vis[y])
45                 q.push(y),vis[y]=1;
46         }
47     }
48 }
49 return last[t]!=-1;
50 }
51 void MCMF(int s,int t)
52 {
53     int maxf=0,maxc=0;
54     while(spfa(s,t))
55     {
56         maxf+=flow[t];
57         maxc+=flow[t]*dis[t];
58         for(int i=t;i!=s;i=edge[last[i]^1].to)
59         {
60             edge[last[i]].w-=flow[t];
61             edge[last[i]^1].w+=flow[t];
62         }
63     }
64     cout<<maxf<<" " <<maxc<<endl;
65 }
```



```
34             minv[j] -= delta;
35         }
36     }
37     j0 = j1;
38     } while (p[j0] != 0);
39     do {
40         long long j1 = way[j0];
41         p[j0] = p[j1];
42         j0 = j1;
43     } while (j0);
44 }
45 long long res = 0;
46 for (int i = 1; i <= m; i++) {
47     res += w[p[i]][i];
48 }
49 return res;
50 }
51 }
52 void solve()
53 {
54     while(cin>>n)
55     {
56         for(int i=1;i<=n;i++)
57         {
58             for(int j=1;j<=n;j++)
59             {
60                 cin>>w[i][j];
61             }
62         }
63         /*最小费用*/
64         cout<<KM::cal(n, n)<<endl;
65     }
66 }
```

## 3.9 最小生成树 Boruvka

```
1  #define mp make_pair
2  const int N=200005;
3  const int M=500005;
4  const LL inf=1e12;
5  int f[N],pd,n,m;
6  struct node{ int a,b; LL c; }p[M];
7  pair<LL,LL> E[N];
8  int find(int r){
9      return f[r]=(r==f[r])?r:find(f[r]);
10 }
11 LL Boruvka(){
12     LL res=0; pd=1; int num=0;
13     for(int i=1;i<=n;i++)
14         f[i]=i;
15     while(num<n-1)
16     {
17         int tmp=0;
18         for(int i=1;i<=n;i++)
19         {
20             int fa=find(i);
21             E[fa]={inf,inf};
22         }
23         for(int i=1;i<=m;i++)
24         {
25             int fa=find(p[i].a);
26             int fb=find(p[i].b);
27             if (fa==fb) continue;
28             tmp++;
29             E[fa]=min(E[fa],{p[i].c,i*1ll});
30             E[fb]=min(E[fb],{p[i].c,i*1ll});
31         }
32         if (tmp==0) break;
33         for(int i=1;i<=m;i++)
34         {
35             int fa=find(p[i].a); int fb=find(p[i].b);
```

```
36         if (fa==fb) continue;
37         if ((E[fa]==mp(p[i].c,i*1ll))||(E[fb]==mp(p[i]
           ].c,i*1ll)))
38             { f[fa]=fb; res+=p[i].c; num++; }
39     }
40 }
41 if (num<n-1) pd=0;
42 return res;
43 }
44 int main(){
45     scanf("%d%d",&n,&m);
46     for(int i=1;i<=m;i++)
47         scanf("%d%d%lld",&p[i].a,&p[i].b,&p[i].c);
48     LL ans=Boruvka();
49     if (!pd) printf("orz\n");
50     else printf("%lld\n",ans);
51     return 0;
52 }
```



### 3.10 基尔霍夫矩阵

```

1  /*
2  基尔霍夫局矩阵树
3  用于解决：
4      给定n个点m条边的无向图，求图的生成树个数
5  基本定义：
6  1.无向图G:给定n个点，m条边的无向图，设点集为V，边集为E，记
    为G(V,E)
7  2.度数矩阵D[G]:当i!=j时，D[i][j]=0,当i==j时,D[i][j]=点v的
    度数
8  3.邻接矩阵A[G]:当vi，vj有边链接时，A[i][j]=1,当vi，vj无边
    连接时，A[i][j]=0
9  4.基尔霍夫矩阵:K[G]:K[G]=D[G]-A[G],K[i][j]=D[i][j]-A[i][j]
10 定理：
11      对于已经得出的基尔霍夫矩阵，去掉随意一行一列得出的矩阵
    的行列式
12 其绝对值为生成树个数
13
14 一般来说求(1,1)~(n-1,n-1)这个行列式
15 */
16 ll det(int n)
17 {
18     ll res=1;
19     for(int i=2;i<=n;i++){
20         for(int j=i+1;j<=n;j++){
21             while(a[j][i]){
22                 ll t=a[i][i]/a[j][i];
23                 for(int k=i;k<=n;k++)
24                     a[i][k]=(a[i][k]-a[j][k]*t);
25                 for(int k=i;k<=n;k++)
26                     swap(a[i][k],a[j][k]);
27                 res=-res;
28             }
29         }
30         if(a[i][i]==0)res=-res;
31         res*=a[i][i];

```

```
32     }
33     if(res<0)res=-res;
34     return res;
35 }
36 void solve()
37 {
38     init();
39     cin>>n>>m;
40     while(m-->0)
41     {
42         int x,y;
43         cin>>x>>y;
44         a[x][x]++;
45         a[y][y]++;
46         a[x][y]--;
47         a[y][x]--;
48     }
49     cout<<det(n)<<endl;
50 }
```

## 3.11 全源最短路

```
1 struct node{
2     int x;ll dis;
3     bool operator<(const node&a)const{
4         return a.dis<dis;
5     }
6 };
7 struct Edge{int to,next;ll w;}edge[maxn];
8 int head[maxn],cnt;
9 ll dis[maxn];
10 ll h[maxn];
11 int n,m;
12 void add(int from,int to,ll w){
13     edge[++cnt].w=w;
14     edge[cnt].to=to;
15     edge[cnt].next=head[from];
16     head[from]=cnt;
17 }
18 bool spfa(int s)
19 {
20     for(int i=0;i<=n;i++)h[i]=INF;
21     vector<int>in(n+10,0);
22     bitset<maxn>vis;
23     queue<int>q;
24     vis[s]=1;
25     h[s]=0;
26     q.push(s);
27     while(!q.empty()){
28         int x=q.front();q.pop();
29         vis[x]=0;
30         in[x]++;
31         if(in[x]>n)return 0;
32         for(int i=head[x];i;i=edge[i].next){
33             int y=edge[i].to;
34             if(h[y]>h[x]+edge[i].w){
35                 h[y]=h[x]+edge[i].w;
```

```
36         if(!vis[y])q.push(y),vis[y]=1;
37     }
38 }
39 }
40 return 1;
41 }
42 void dij(int s)
43 {
44     for(int i=1;i<=n;i++)dis[i]=INF;
45     priority_queue<node>q;
46     bitset<maxn>vis;
47     q.push({s,0});
48     dis[s]=0;
49     while(!q.empty()){
50         auto [x,now]=q.top();q.pop();
51         if(vis[x])continue;
52         vis[x]=1;
53         for(int i=head[x];i;i=edge[i].next){
54             int y=edge[i].to;
55             if(dis[y]>dis[x]+edge[i].w){
56                 dis[y]=dis[x]+edge[i].w;
57                 q.push({y,dis[y]});
58             }
59         }
60     }
61 }
62 int main()
63 {
64     cin>>n>>m;
65     for(int i=1;i<=m;i++){
66         int x,y,z;
67         cin>>x>>y>>z;
68         add(x,y,z);
69     }
70     for(int i=1;i<=n;i++)
71         add(0,i,0);
72     //建立虚点，从0开始单源最短路
```

```
73 //0开始单源最短路即为势能
74 //同时spfa判断是否存在负环 (in[x]>n)
75 if(!spfa(0)){
76     cout<<-1<<endl;
77     return 0;
78 }
79 //每一条路加上势能差
80 for(int u=1;u<=n;u++){
81     for(int i=head[u];i;i=edge[i].next){
82         edge[i].w+=h[u]-h[edge[i].to];
83     }
84 }
85 for(int i=1;i<=n;i++){
86     dij(i);
87     ll ans=0;
88     for(int j=1;j<=n;j++){
89         if(dis[i]==INF)ans+=j*INF;
90         else ans+=j*(dis[j]+h[j]-h[i]);
91         //减去势能差
92     }
93     cout<<ans<<endl;
94 }
95 }
```

## 3.12 全局最小割

```

1  int G[MAX_N][MAX_N];
2  int v[MAX_N]; //v[i]代表节点i合并到的顶点
3  int w[MAX_N]; //定义 $w(A,x) = \sum w(v[i],x)$ ,  $v[i] \in A$ 
4  bool visited[MAX_N]; //用来标记是否该点加入了A集合
5  int squ[MAX_N]; //记录移除的节点次序
6  int index; //记录最小割的位置, 以便分开整个图的节点
7
8  int stoer_wagner(int n)
9  {
10     int min_cut = INF, r=0;
11     for (int i = 0; i < n; ++i){
12         v[i] = i; //初始还未合并, 都代表节点本身
13     }
14     while (n > 1){
15         int pre = 0; //pre用来表示之前加入A集合的点 (在t之前一个加进去的点)
16         memset(visited, 0, sizeof(visited));
17         memset(w, 0, sizeof(w));
18         for (int i = 1; i < n; ++i){
19             //求出 某一轮最大生成树的最后两个节点, 并且去除最后的t, 将与t连接的边归并
20             int k = -1;
21             for (int j = 1; j < n; ++j){ //选取V-A中的w(A, x)最大的点x加入集合
22                 if (!visited[v[j]]){
23                     w[v[j]] += G[v[pre]][v[j]];
24                     if (k == -1 || w[v[k]] < w[v[j]])
25                         k = j;
26                 }
27             }
28             visited[v[k]] = true; //标记该点x已经加入A集合
29             if (i == n - 1) //若|A|=|V| (所有点都加入了A), 结束
30                 {
31                     const int s = v[pre], t = v[k];

```

```

32          //令倒数第二个加入A的点 (v[pre]) 为s, 最后
           一个加入A的点 (v[k]) 为t
33          cout<<t<<"---"<<s<<endl;
34          squ[r++]=t;
35          if(w[t]<min_cut){
36              min_cut=w[t];
37              index=r;
38          }
39          //min_cut = min(min_cut, w[t]);
40          // 则s-t最小割为w(A,t), 用其更新min_cut
41          for (int j = 0; j < n; ++j) //Contract(s,
           t)
42          {
43              G[s][v[j]] += G[v[j]][t];
44              G[v[j]][s] += G[v[j]][t];
45          }
46          v[k] = v[--n]; //删除最后一个点 (即删除t,
           也即将t合并到s)
47      }
48      pre = k;
49  }
50  }
51  return min_cut;
52  }
53  int main(int argc, char *argv[]){
54      int n, m;
55      while (scanf("%d%d", &n, &m) != EOF)
56      {
57          memset(G, 0, sizeof(G));
58          while (m--)
59          {
60              int u, v, w;
61              scanf("%d%d%d", &u, &v, &w);
62              G[u][v] += w;
63              G[v][u] += w;
64          }
65          int z=n;

```

```
66         //printf("%d\n", stoer_wagner(n));
67         cout<<"\r\n归并的步骤为："<<endl;
68         int res=stoer_wagner(n);
69         cout<<"\r\n最小割的总权值为：□"<<res<<"\r\n图划分
           为部分A： ";
70         //cout<<"图划分为部分A： ";
71         for(int i=0;i<z;i++)
72         {
73             if(i==index)
74                 cout<<"部分B： ";
75             cout<<squ[i]<<"□□";
76         }
77     }
78     return 0;
79 }
```



## 4 数据结构

### 4.1 MultiplyLCA

```
1 struct Multiply_LCA{
2     vector<int>v[maxn];
3     int dep[maxn];
4     int f[maxn][25];
5
6     void dfs(int x,int fa)
7     {
8         dep[x]=dep[fa]+1;
9         f[x][0]=fa;
10        for(int i=1;i<=20;i++)
11            f[x][i]=f[f[x][i-1]][i-1];
12        for(auto y:v[x])
13            if(y!=fa)
14                RMQ(y,x);
15    }
16    int LCA(int x,int y)
17    {
18        if(dep[x]<dep[y])swap(x,y);
19        for(int i=20;i>=0;i--)
20            if(dep[f[x][i]]>=dep[y])
21                x=f[x][i];
22        if(x==y)return x;
23        for(int i=20;i>=0;i--)
24            if(dep[f[x][i]]!=dep[f[y][i]])
25                x=f[x][i],y=f[y][i];
26        return f[x][0];
27    }
28 };
```

## 4.2 ST

```
1 struct ST{
2     int f[maxn][30];
3     void init(){
4         for(int i=1;i<=n;i++)//距离为0初始化
5             f[i][0]=p[i];
6         int t=log(n)/log(2)+1;//j定位
7         for(int j=1;j<t;j++){
8             for(int i=1;i<=n-(1<<j)+1;i++){
9                 f[i][j]=max(f[i][j-1],f[i+(1<<(j-1))][j-1]);
10                //f[i][j-1]为f[i][j]的长为 $2^{j-1}$ 的前半段,
11                //另一段为f[i+(1<<(j-1))][j-1]
12            }
13        }
14        int query(int l,int r){
15            int k=log2(r-l+1);
16            return max(f[l][k],f[r-(1<<k)+1][k]);
17        }
18    }st;
```

### 4.3 GrayCode

```
1 // n位格雷码
2 vector<int> grayCode(int n) {
3     int count = 1 << n;
4     vector<int> res(count,0);
5     for(int i = 1 ; i < count; i ++ )
6     {
7         int bin = i, cur = bin >> (n - 1);
8         for(int k = n - 1; k > 0; k --)
9             cur = (cur << 1) + (((bin >> k) & 1) ^ ((bin
10                 >>(k - 1)) & 1));
11         res[i] = cur;
12     }
13     return res;
14 }
15 //递归实现n位格雷码
16 vector<string> gray_code(int n){
17     if(n==1)return {"0","1"};
18     else{
19         vector<string>v,v1;
20         v1=gray_code(n-1);
21         for(int i=0;i<v1.size();i++)
22             v.push_back("0"+v1[i]);
23         for(int i=(v1.size()-1);i>-1;i--)
24             v.push_back("1"+v1[i]);
25         return v;
26     }
27 }
28 void solve()
29 {
30     int n;
31     cin>>n;
32     vector<string>v=gray_code(n);
33     for(int i=0;i<v.size();i++)
34         cout<<v[i]<<endl;
35 }
```

## 4.4 归并排序

```
1 void doit()
2 {
3     int n;
4     vector<int>p(n+1,0);
5     auto mg_sort = [&](auto mg_sort,int l,int r)->ll{
6         if(l>=r)return 0;
7         int mid=(l+r)>>1;
8         ll t = mg_sort(mg_sort,l,mid)+mg_sort(mg_sort,mid
9             +1,r);
10        vector<int>temp;
11        int i=l,j=mid+1;
12        while(i<=mid&&j<=r)
13        {
14            if(p[i]<=p[j])temp.push_back(p[i++]);
15            else t+=mid-i+1,temp.push_back(p[j++]);
16        }
17        while(i<=mid)temp.push_back(p[i++]);
18        while(j<=r)temp.push_back(p[j++]);
19        for(int i=l,j=0;i<=r;i++,j++)
20            p[i]=temp[j];
21        return t;
22    };
23    mg_sort(mg_sort,1,n);
24 }
```

## 4.5 树链剖分

```

1  struct Seg{int l,r,val,lazy;}t[maxn<<2];
2  struct Edge{int to,next,w;}edge[maxn];
3  int head[maxn],cnt;
4  int siz[maxn],f[maxn],dep[maxn],son[maxn];
5  int id[maxn],di[maxn],top[maxn],tot;
6  int w[maxn];
7  int n,root,q,P;
8  inline void add(int from,int to,int w=0){
9      edge[++cnt]={to,head[from],w};
10     head[from]=cnt;
11 }
12 inline void pushup(int rt){
13     t[rt].val=t[rt<<1].val+t[rt<<1|1].val;
14 }
15 inline void pushdown(int rt){
16     if(t[rt].lazy){
17         (t[rt<<1].val+=(t[rt].lazy*(t[rt<<1].r-t[rt<<1].l
18             +1))%P)%=P;
19         (t[rt<<1|1].val+=(t[rt].lazy*(t[rt<<1|1].r-t[rt
20             <<1|1].l+1))%P)%=P;
21         (t[rt<<1].lazy+=t[rt].lazy)%=P;
22         (t[rt<<1|1].lazy+=t[rt].lazy)%=P;
23         t[rt].lazy=0;
24     }
25 }
26 inline void build(int rt,int l,int r)
27 {
28     t[rt]={l,r,0,0};
29     if(l==r){
30         (t[rt].val=w[di[l]])%P;
31         return ;
32     }
33     int mid=(l+r)>>1;
34     build(rt<<1,l,mid);
35     build(rt<<1|1,mid+1,r);

```

```
34     pushup(rt);
35 }
36 inline void update(int rt,int l,int r,int k)
37 {
38     if(l<=t[rt].l&&t[rt].r<=r){
39         (t[rt].val+=(k*(t[rt].r-t[rt].l+1))%P)%=P;
40         (t[rt].lazy+=k)%=P;
41         return ;
42     }
43     pushdown(rt);
44     int mid=(t[rt].l+t[rt].r)>>1;
45     if(l<=mid)update(rt<<1,l,r,k);
46     if(r>mid)update(rt<<1|1,l,r,k);
47     pushup(rt);
48 }
49 inline int query(int rt,int l,int r){
50     if(l<=t[rt].l&&t[rt].r<=r)
51         return t[rt].val;
52     pushdown(rt);
53     int mid=(t[rt].l+t[rt].r)>>1;
54     int ans=0;
55     if(l<=mid)(ans+=query(rt<<1,l,r))%=P;
56     if(r>mid)(ans+=query(rt<<1|1,l,r))%=P;
57     return ans;
58 }
59 inline void upRange(int x,int y,int k)
60 {
61     while(top[x]!=top[y]){
62         if(dep[top[x]]<dep[top[y]])swap(x,y);
63         update(1,id[top[x]],id[x],k);
64         x=f[top[x]];
65     }
66     if(dep[x]>dep[y])swap(x,y);
67     update(1,id[x],id[y],k);
68 }
69 inline void upSon(int x,int k){
70     update(1,id[x],id[x]+siz[x]-1,k);
```

```
71 }
72 inline int qRange(int x,int y)
73 {
74     int ans=0;
75     while(top[x]!=top[y]){
76         if(dep[top[x]]<dep[top[y]])swap(x,y);
77         (ans+=query(1,id[top[x]],id[x]))%=P;
78         x=f[top[x]];
79     }
80     if(dep[x]>dep[y])swap(x,y);
81     (ans+=query(1,id[x],id[y]))%=P;
82     return ans;
83 }
84 inline int qSon(int x){
85     return query(1,id[x],id[x]+siz[x]-1);
86 }
87 inline int LCA(int x,int y)
88 {
89     while(top[x]!=top[y]){
90         if(dep[top[x]]<dep[top[y]])swap(x,y);
91         x=f[top[x]];
92     }
93     if(dep[x]>dep[y])swap(x,y);
94     return x;
95 }
96 inline void dfs1(int x)
97 {
98     siz[x]=1;
99     for(int i=head[x];i;i=edge[i].next){
100         int y=edge[i].to;
101         if(y==f[x])continue;
102         f[y]=x;dep[y]=dep[x]+1;
103         dfs1(y);
104         siz[x]+=siz[y];
105         if(siz[y]>siz[son[x]])
106             son[x]=y;
107     }
```

```
108 }
109 inline void dfs2(int x,int topf)
110 {
111     id[x]=++tot;
112     di[tot]=x;
113     top[x]=topf;
114     if(son[x])dfs2(son[x],topf);
115     for(int i=head[x];i;i=edge[i].next)
116     {
117         int y=edge[i].to;
118         if(y==f[x]||y==son[x])continue;
119         dfs2(y,y);
120     }
121 }
122 void solve()
123 {
124     cin>>n>>q>>root>>P;
125     for(int i=1;i<=n;i++)
126         cin>>w[i];
127     for(int i=1;i<n;i++)
128     {
129         int x,y;
130         cin>>x>>y;
131         add(x,y);
132         add(y,x);
133     }
134     f[root]=0;
135     dfs1(root);
136     dfs2(root,root);
137     build(1,1,n);
138     while(q--)
139     {
140         int op,x,y,k;
141         cin>>op;
142         if(op==1)
143         {
144             cin>>x>>y>>k;
```



```
145         upRange(x,y,k);
146     }
147     else if(op==2)
148     {
149         cin>>x>>y;
150         cout<<qRange(x,y)<<endl;
151     }
152     else if(op==3)
153     {
154         cin>>x>>k;
155         upSon(x,k);
156     }
157     else
158     {
159         cin>>x;
160         cout<<qSon(x)<<endl;
161     }
162 }
163 }
```

## 4.6 扫描线线段树

```
1 struct Seg{
2     int l,r,len,cv,_l,_r,cnt;
3 }t[maxn];
4 pp(4) p[maxn];
5 int w[maxn];
6 int n,m,cnt;
7 void build(int rt,int l,int r)
8 {
9     t[rt]={l,r,0,0,0,0,0};
10    if(l==r)return ;
11    int mid=(l+r)>>1;
12    build(rt<<1,l,mid);
13    build(rt<<1|1,mid+1,r);
14 }
15 void pushup(int rt)
16 {
17     if(t[rt].cv)
18     {
19         t[rt]._l=t[rt]._r=1;
20         t[rt].len=w[t[rt].r+1]-w[t[rt].l];
21         t[rt].cnt=1;
22     }
23     else if(t[rt].l==t[rt].r)
24     {
25         t[rt]._l=t[rt]._r=0;
26         t[rt].len=0;
27         t[rt].cnt=0;
28     }
29     else
30     {
31         t[rt]._l=t[rt<<1]._l;
32         t[rt]._r=t[rt<<1|1]._r;
33         t[rt].len=t[rt<<1].len+t[rt<<1|1].len;
34         t[rt].cnt=t[rt<<1].cnt+t[rt<<1|1].cnt-(t[rt<<1]._r
            &t[rt<<1|1]._l);
```

```
35     }
36 }
37 void update(int rt,int l,int r,int c)
38 {
39     if(l<=t[rt].l&& t[rt].r<=r)
40     {
41         t[rt].cv+=c;
42         pushup(rt);
43         return ;
44     }
45     int mid=(t[rt].l+t[rt].r)>>1;
46     if(l<=mid)update(rt<<1,l,r,c);
47     if(r>mid)update(rt<<1|1,l,r,c);
48     pushup(rt);
49 }
50 void solve()
51 {
52     cin>>n;
53     for(int i=1;i<=n;i++)
54     {
55         int x,y,_x,_y;
56         cin>>x>>y>>_x>>_y;
57         w[++m]=y;
58         w[++m]=_y;
59         p[i*2-1]={x,y,_y,1};
60         p[i*2]={_x,y,_y,-1};
61     }
62     sort(w+1,w+1+m);
63     m=unique(w+1,w+1+m)-w-1;
64     sort(p+1,p+1+n+n,[&](pp(4)a,pp(4)b){
65         if(a[0]==b[0])return a[3]>b[3];
66         return a[0]<b[0];
67     });
68     for(int i=1;i<=n<<1;i++)
69     {
70         p[i][1]=lower_bound(w+1,w+1+m,p[i][1])-w;
71         p[i][2]=lower_bound(w+1,w+1+m,p[i][2])-w;
```

```
72     }
73     build(1,1,m);
74     int pre=0;
75     int ans=0;
76     for(int i=1;i<=n<<1;i++)
77     {
78         auto [x,y,_y,op]=p[i];
79         update(1,y,_y-1,op);
80         ans+=abs(t[1].len-pre);
81         ans+=(p[i+1][0]-p[i][0])*2*t[1].cnt;
82         pre=t[1].len;
83     }
84     cout<<ans<<endl;
85 }
```

## 4.7 莫队

```
1  struct Query{
2      int l,r,id,block;
3      bool operator < (const Query& q)const
4      {
5          if(block==q.block)
6              return r<q.r;
7          else
8              return block<q.block;
9      }
10 }p[maxn];
11 int cnt_a[maxn],cnt_b[maxn];
12 int a[maxn],b[maxn];
13 bool ans[maxn];
14 int n,m,ok=0,cnt=0;
15 void Debug()
16 {
17     printf("[Debug]:");
18     int ma=0;
19     for(int i=1;i<=n;i++)
20         ma=max({ma,a[i],b[i]});
21     for(int i=1;i<=ma;i++)
22         cout<<cnt_a[i];cout<<endl;
23     printf("[Debug]:");
24     for(int i=1;i<=ma;i++)
25         cout<<cnt_b[i];
26     cout<<endl;
27     cout<<cnt<<endl;
28 }
29 inline void upd_a(int x,int fl)
30 {
31     int fl_1=0,fl_2=0;
32     if((cnt_a[x]>0&&cnt_b[x]>0)|| (cnt_a[x]==0&&cnt_b[x]
33         ]==0))fl_1=1;
34     if((cnt_a[x]+fl>0&&cnt_b[x]>0)|| (cnt_a[x]+fl==0&&cnt_b
35         [x]==0))fl_2=1;
```

```
34     if(fl_1>fl_2)cnt++;
35     else if(fl_1<fl_2)cnt--;
36     cnt_a[x]+=fl;
37     if(cnt)ok=1;
38     else ok=0;
39 }
40 inline void upd_b(int x,int fl)
41 {
42     int fl_1=0,fl_2=0;
43     if((cnt_b[x]>0&&cnt_a[x]>0)|| (cnt_b[x]==0&&cnt_a[x]
        ]==0))fl_1=1;
44     if((cnt_b[x]+fl>0&&cnt_a[x]>0)|| (cnt_b[x]+fl==0&&cnt_a
        [x]==0))fl_2=1;
45     if(fl_1>fl_2)cnt++;
46     else if(fl_1<fl_2)cnt--;
47     cnt_b[x]+=fl;
48     if(cnt)ok=1;
49     else ok=0;
50 }
51 void solve()
52 {
53     cin>>n>>m;
54     for(int i=1;i<=n;i++)
55         cin>>a[i];
56     for(int i=1;i<=n;i++)
57         cin>>b[i];
58     int tmp=sqrt(n);
59     for(int i=1;i<=m;i++)
60     {
61         cin>>p[i].l>>p[i].r;
62         p[i].id=i;
63         p[i].block=i/tmp;
64     }
65     sort(p+1,p+1+m);
66     for(int i=1;i<=max(p[1].l,p[1].r);i++)
67     {
68         if(i<=p[1].l)
```

```
69         upd_a(a[i],1);
70         if(i<=p[1].r)
71             upd_b(b[i],1);
72     }
73     ans[p[1].id]=ok;
74     int l=p[1].l;
75     int r=p[1].r;
76     // Debug();
77     for(int i=2;i<=m;i++)
78     {
79         while(l<p[i].l)//l++;
80             l++,upd_a(a[l],1);
81         while(l>p[i].l)//l--;
82             upd_a(a[l],-1),l--;
83         while(r<p[i].r)//r++;
84             r++,upd_b(b[r],1);
85         while(r>p[i].r)//r--;
86             upd_b(b[r],-1),r--;
87         ans[p[i].id]=ok;
88         // Debug();
89     }
90     for(int i=1;i<=m;i++)
91     {
92         if(ans[i])cout<<"NO"<<endl;
93         else cout<<"YES"<<endl;
94     }
95 }
```

## 5 数论

### 5.1 ExGcd

```
1 int exgcd(int a,int b,int& x,int& y)
2 {
3     if(b==0)return x=1,y=0,a;
4     int d=exgcd(b,a%b,y,x);
5     y-=x*(a/b);
6     return d;
7 }
```

### 5.2 快速幂

```
1 int mull(int a,int b){int qw=0;while(b){if(b&1)qw=(qw+a)%
    mod;a=(a+a)%mod;b>>=1;}return qw;}
2
3 int ksm(int b,int p){int r=1;b%=mod;while(p){if(p&1)r=(r*b
    )%mod;p>>=1;b=(b*b)%mod;}return r;}
4
5 int ksm(int b,int p,int mod){int r=1;b%=mod;while(p){if(p
    &1)r=(r*b)%mod;p>>=1;b=(b*b)%mod;}return r;}
```

### 5.3 整除分块

```
1 ll division_block(ll n){
2     ll res = 0;
3     for(ll l = 1, r; l <= n; l = r + 1){
4         r = n / (n / l);
5         // cout<<r<<endl;
6         res += n / l * (r - l + 1);
7     }
8     return res;
9 }
```



## 5.4 欧拉函数

```
1 int prime[maxn],phi[maxn];
2 bool vis[maxn];
3 int n;
4 void doit()
5 {
6     for(int i=2;i<=maxn;i++){
7         if(!vis[i])prime[++prime[0]]=i,phi[i]=i-1;
8         for(int j=1;j<=prime[0]&&prime[j]*i<=maxn;j++){
9             vis[i*prime[j]]=1;
10            if(i%prime[j]==0){
11                phi[i*prime[j]]=phi[i]*prime[j];
12                break;
13            }
14            phi[i*prime[j]]=phi[i]*(prime[j]-1);
15        }
16    }
17 }
18 int PHI(int x)
19 {
20     //0(sqrt(x)
21     int ans=x;
22     for(int i=2;i*i<=x;i++)
23         if(x%i==0)
24         {
25             while(x%i==0)x/=i;
26             ans=ans*(i-1)/i;
27         }
28     if(x>1)ans=ans*(x-1)/x;
29     return ans;
30 }
```

## 5.5 等比数列求和

```
1 int ksm(int b,int p,int mod){int r=1;b%=mod;while(p){if(p
    &1)r=(r*b)%mod;p>>=1;b=(b*b)%mod;}return r;}
2 int sumq(int q,int n,int p)
3 {
4     if(n==1)return 1;
5     if(n%2==0)return ((1+ksm(q,n/2,p))*sumq(q,n/2,p)%p)%p;
6     else return (1+q*sumq(q,n-1,p))%p;
7 }
8 //  $q^0 + q^1 + q^2 + \dots + q^{n-1}$ 
9 void solve()
10 {
11     int q,n,p;
12     cin >> q >> n >> p;
13     cout << sumq(q,n,p) % p << endl;
14 }
```

## 5.6 素数

```
1 bool su(int x)
2 {
3     if(x==1)return 0;
4     if(x==2||x==3)return 1;
5     if(x%6!=5&& x%6!=1)return 0;
6     int tmp=sqrt(x);
7     for(int i=5;i<=tmp;i+=6)
8         if(x%i==0||x%(i+2)==0)
9             return 0;
10     return 1;
11 }
```

## 5.7 排列组合其一

```
1 int gcd(int a,int b){
2     if(b==0)return a;
3     else return gcd(b,a%b);
4 }
5 //解线性同余方程，扩展欧几里德定理
6 int x,y;
7 void Extended_gcd(int a,int b){
8     if(b==0){x=1;y=0;}
9     else{Extended_gcd(b,a%b);long t=x;x=y;y=t-(a/b)*y;}
10 }
11 //计算不大的C(n,m)
12 int C(int a,int b){
13     if(b>a)return 0;b=(ff[a-b]%mod*ff[b])%mod;a=ff[a];
14     int c=gcd(a,b);a/=c;b/=c;Extended_gcd(b,M);x=(x+mod)%
        mod;
15     x=(x*a)%mod;return x;
16 }
17 //Lucas定理
18 int Combination(int n, int m)
19 {
20     int ans=1;int a,b;
21     while(m||n){a=n%mod;b=m%mod;n/=mod;m/=mod;
22         ans=(ans*C(a,b))%mod;}
23     return ans;
24 }
25 int A(int n,int m){return (Combination(n,m)*ff[m])%mod;}
26
27 signed main()
28 {
29     int i,m,n;
30     ff[0]=1;
31     for(int i=1; i<=M; i++) //预计算n!
32         ff[i]=(ff[i-1]*i)%mod;
33     while(~scanf("%lld%lld",&n, &m))
34     {
```

```
35     printf("%lld\n",Combination(n,m));
36 }
37 return 0;
38 }
```

## 5.8 排列组合其二

```
1 ll mod=998244353;
2 ll qmi(ll a, ll k)//快速幂模板
3 {
4     int res = 1;
5     while (k)
6     {
7         if (k & 1) res = (ll)res * a % mod;
8         a = (ll)a * a % mod;
9         k >>= 1;
10    }
11    return res;
12 }
13 ll C(ll a, ll b)//通过定理求组合数C(a, b)
14 {
15     ll res = 1;
16     for (int i = 1, j = a; i <= b; i ++, j -- )
17     {
18         res = (ll)res * j % mod;
19         res = (ll)res * qmi(i, mod - 2) % mod;
20     }
21     return res;
22 }
23 ll lucas(ll a, ll b)
24 {
25     if (a < mod && b < mod) return C(a, b);
26     return (ll)C(a % mod, b % mod)*lucas(a / mod, b / mod)
27         % mod;
28 }
```

## 5.9 排列组合其三

```
1  int ksm(int b,int p){int r=1;b%=mod;while(p){if(p&1)r=(r*b)%mod;p>>=1;b=(b*b)%mod;}return r;}
2
3  int C(int n,int m){
4      return F[n]*INF[n-m]*INF[m];
5  }
6  F[0]=INF[0]=1;
7  for(int i=1;i<M;i++)
8  {
9      F[i]=F[i-1]*i%mod;
10     INF[i]=ksm(F[i],mod-2);
11 }
```

## 5.10 单调队列维护 DP

```
1  int pre[N];
2  int dp[N];
3  int q[N];
4  int n,m;
5  void solve()
6  {
7      cin>>n>>m;
8      for(int i=1;i<=n;i++)
9      {
10         int v,w,s;
11         cin>>v>>w>>s;
12         // 复制上一次的i-1 的DP
13         memcpy(pre,dp,sizeof(dp));
14         // 枚举余数为j
15         for(int j=0;j<v;j++)
16         {
17             int head=0;
18             int tail=-1;
19             for(int k=j;k<=m;k+=v)
20             {
21                 //k-q[head]>s*v 长度长了，最多s个
22                 if(head<=tail&&q[head]<k-s*v)
23                     head++;
24                 while(head<=tail&&pre[q[tail]]-(q[tail]-j)
25                     /v*w<=pre[k]-(k-j)/v*w)
26                     tail--;
27                 if(head<=tail)
28                     dp[k]=max(dp[k],pre[q[head]]+(k-q[head]
29                         )/v*w);
30                 q[++tail]=k;
31             }
32         }
33     }
```

### 5.11 欧拉定理

当取模为  $m$  时

$$a^{\varphi(m)} \equiv 1 \pmod{m} \quad \gcd(a, m) = 1$$

$$a^b \equiv a^{b \bmod \varphi(m)} \pmod{m}$$

### 5.12 拓展欧拉定理

$$a^b = \begin{cases} a^b, & b < \varphi(m) \\ a^{b \bmod (\varphi(m) + \varphi(m))}, & b \geq \varphi(m) \end{cases}$$

### 5.13 排列组合隔板法

5.13.1 当你有  $n$  个球分成  $m$  份（每一份大于 0）时：

$$C_{n-1}^{m-1}$$

5.13.2 当你有  $n$  个球分成  $m$  份（每一份大于等于 0）时：

$$C_{n+m-1}^{m-1}$$

## 5.14 拓展中国剩余定理

$$x \equiv r_1(\text{mod } m_1)$$

$$x \equiv r_2(\text{mod } m_2)$$

$$\begin{cases} x - m_1q = r_1 \\ x + m_2p = r_2 \end{cases}$$

$$m_1q + m_2p = r_2 - r_1$$

$$q = q_0 + \frac{m_2}{\gcd(m_1, m_2)} \cdot k$$

$$p = p_0 - \frac{m_1}{\gcd(m_1, m_2)} \cdot k$$

$$x = m_1q + r_1 = q_0m_1 + \frac{m_1m_2}{\gcd(m_1, m_2)} + r_1$$

$$x = \text{lcm}(m_1, m_2) \cdot k + q_0m_1 + r_1$$

## 5.15 向量旋转 A 度时对应坐标为

$$[x \cos A - y \sin A, \quad x \sin A + y \cos A]$$

## 5.16 威尔逊定理

$$(p - 1)! \equiv -1 \text{ mod } p(\text{prime})$$



## 5.17 泰勒展开

$$\begin{aligned}
 f(x) &= f(x_0) + \frac{f'(x_0)}{1!} \cdot (x - x_0) + \frac{f''(x_0)}{2!} \cdot (x - x_0)^2 \cdots \frac{f^{(n)}(x_0)}{n!} \cdot (x - x_0)^n \\
 \frac{1}{1-x} &= 1 + x + x^2 + x^3 + \cdots \\
 \frac{1}{1-x^2} &= 1 + x^2 + x^4 + \cdots \\
 \frac{1}{(1-x)^n} &= \sum_{i=0}^{\infty} C_{n+i-1}^i x^i \\
 \frac{1}{(1+x)^n} &= \sum_{i=0}^{\infty} (-1)^i C_{n+i-1}^i x^i \\
 e^x &= 1 + \frac{x^1}{1!} + \cdots \equiv \sum_{n=0}^{\infty} \frac{x^n}{n!} \\
 \frac{e^x + e^{-x}}{2} &= 1 + \frac{x^2}{2!} + \frac{x^4}{4!} + \cdots \\
 \frac{e^x - e^{-x}}{2} &= x + \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots
 \end{aligned}$$

## 5.18 康托展开

可以  $O(n\log n)$  复杂度求一个排列对于排列中的排名

```
1  const int N=1e6+10,mod=998244353;
2  long long fac[N];
3  int tr[N];
4  int n;
5  int a[N];
6  int lowbit(int x)
7  {
8      return x&-x;
9  }
10 void add(int x)
11 {
12     for(int i=x;i<=n;i+=lowbit(i)) tr[i]+=1;
13 }
14 int qr(int x)
15 {
16     int res=0;
17     for(int i=x;i;i-=lowbit(i)) res+=tr[i];
18     return res;
19 }
20
21 int main()
22 {
23     ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
24     cin>>n;
25     fac[0]=1;
26     for(int i=1;i<=n;i++) {
27         cin>>a[i];
28         fac[i]=fac[i-1]*i%mod;
29     }
30     long long ans=0;
31     for(int i=1;i<=n;i++)
32     {
33         int t=qr(a[i]);
34         ans=(ans+(a[i]-1-t)*fac[n-i]%mod)%mod;
```

```
35         add(a[i]);  
36     }  
37     cout<<(ans+1)%mod;  
38     return 0;  
39 }
```

### 5.19 逆康托展开

已经知道排名得到原排列，逆向推出原序列

```
1 vector<int> incantor(int x,int n) {
2     x--; // 得到以0开始的排名
3     vector<int> res(n); // 保存数列答案
4     int cnt;
5     bool st[10]; // 标记数组
6     memset(st,0,sizeof st);
7     for(int i = 0;i < n; ++i) {
8         cnt = x/fact[n - i - 1]; // 比a[i]小且没有出现过的
            数的个数
9         x %= fact[n - i - 1]; // 更新 x
10        for(int j = 1;j <= n; ++j) { // 找到a[i], 从1开始向
            后找
11            if(st[j]) continue; // 如果被标记过, 就跳过
12            if(!cnt) { // 如果cnt == 0说明当前数是a[i]
13                st[j] = 1; // 标记
14                res[i] = j; // 第i位是j
15                break;
16            }
17            cnt --; // 如果当前不是0, 就继续往后找
18        }
19    }
20    return res; // 返回答案
21 }
```

## 5.20 矩阵

### 5.20.1 矩阵求逆

代取模

```
1  const int N=1010,mod=1e9+7;
2
3  long long ksm(long long a,long long b)
4  {
5      a%=mod;
6      long long res=1;
7      while(b)
8      {
9          if(b&1) res=res*a%mod;
10         a=a*a%mod;
11         b>>=1;
12     }
13     return res;
14 }
15
16 int n;
17
18 long long a[N][N];
19
20 bool Ju()
21 {
22     for(int i=1;i<=n;i++)
23     {
24         int r=i;
25         for(int j=i;j<=n;j++)
26         {
27             if(a[j][i]>0) r=j;
28         }
29         if(r!=i) swap(a[r],a[i]);
30         if(a[i][i]==0) return false;
31         long long x=ksm(a[i][i],mod-2);
32         for(int j=n*2;j>=i;j--) a[i][j]=a[i][j]*x%mod;
```

```
33     for(int j=i+1;j<=n;j++)
34     {
35         for(int k=n*2;k>=i;k--)
36         {
37             a[j][k]-=a[j][i]*a[i][k]%mod;
38             a[j][k]=(a[j][k]%mod+mod)%mod;
39         }
40     }
41 }
42 for(int i=n;i>=2;i--)
43 {
44     for(int j=1;j<i;j++)
45     {
46         for(int k=n+1;k<=n+n;k++)
47         {
48             a[j][k]-=a[j][i]*a[i][k]%mod;
49             a[j][k]=(a[j][k]%mod+mod)%mod;
50         }
51     }
52 }
53 return true;
54 }
55
56 int main()
57 {
58     ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
59     cin>>n;
60     for(int i=1;i<=n;i++)
61     {
62         for(int j=1;j<=n;j++)
63         {
64             cin>>a[i][j];
65         }
66         a[i][i+n]=1;
67     }
68     if(Ju())
69     {
```

```
70     for(int i=1;i<=n;i++)
71     {
72         for(int j=n+1;j<=2*n;j++)
73             cout<<a[i][j]<<" ";
74         cout<<"\n";
75     }
76 }
77 else cout<<"No Solution";
78 return 0;
79 }
```

## 5.20.2 高斯消元

求行列式变量值

```
1  const int N=110;
2  int n;
3  double a[N][N];
4  void pr()
5  {
6      for(int i=1;i<=n;i++)
7      {
8          for(int j=1;j<=n+1;j++)
9              cout<<a[i][j]<<' ';
10             cout<<'\\n';
11         }
12     }
13     bool Gauss()
14     {
15         for(int i=1;i<=n;i++)
16         {
17             int r=i;
18             for(int j=i;j<=n;j++)
19             {
20                 if(abs(a[j][i])>=1e-8) r=j;
21             }
22             if(r!=i) swap(a[r],a[i]);
23             if(abs(a[i][i])<1e-8) return false;
24             for(int j=n+1;j>=i;j--)
25             {
26                 a[i][j]/=a[i][i];
27             }
28             for(int j=n;j>i;j--)
29             {
30                 for(int k=n+1;k>=i;k--)
31                 {
32                     a[j][k]-=a[j][i]*a[i][k];
33                 }
34             }
```



```
35     }
36 }
37 for(int i=n;i>=2;i--)
38 {
39     for(int j=i-1;j>=1;j--)
40     {
41         a[j][n+1]-=a[j][i]*a[i][n+1];
42     }
43 }
44 return true;
45 }
46
47 int main()
48 {
49     cin>>n;
50     for(int i=1;i<=n;i++)
51     {
52         for(int j=1;j<=n+1;j++)
53             cin>>a[i][j];
54     }
55     if(Gauss()){
56         for(int i=1;i<=n;i++)
57             printf("%.2lf\n",a[i][n+1]);
58     }
59     else{
60         cout<<"No Solution";
61     }
62     return 0;
63 }
```

## 5.20.3 求行列式值

原理是通过高斯消元将行列式下三角清 0，那么答案就是主对角线的累乘，这里由于模值不一定是质数，不能保障每一个元素都在  $P$  取模情况下有逆元，所以采用了特别妙的方法，通过主元行与其他行进来辗转相除来进行清 0 过程。原理是行列式中一行减去另外一行的乘积，行列式答案不变。

```
1 int mod=998244353;
2 const int N=610;
3 int n;
4 long long mp[N][N];
5 void pr()
6 {
7     for(int i=0;i<n;i++)
8     {
9         for(int j=0;j<n;j++) cout<<mp[i][j]<<" ";cout<<"\n";
10    }
11 }
12 void gauss()
13 {
14     long long res=1;
15     for(int i=0;i<n;i++)
16     {
17         for(int j=i+1;j<n;j++)
18         {
19             while(mp[i][i])
20             {
21                 long long div=mp[j][i]/mp[i][i];
22                 for(int k=i;k<n&&div;k++) {
23                     mp[j][k]=(mp[j][k]-div*mp[i][k])%mod+
24                         mod)%mod;
25                 }
26                 for(int k=i;k<n;k++) swap(mp[i][k],mp[j][k]);res=-res;
27             }
28             for(int k=i;k<n;k++) swap(mp[i][k],mp[j][k]);
29             res=-res;
30         }
31     }
32 }
```

```
28     }
29     }
30     res=(res+mod)%mod;
31     for(int i=0;i<n;i++) res=res*mp[i][i]%mod;
32     cout<<res;
33 }
34 int main()
35 {
36     ios::sync_with_stdio(false);
37     cin.tie(0);cout.tie(0);
38     cin>>n>>mod;
39     for(int i=0;i<n;i++)
40     {
41         for(int j=0;j<n;j++)
42         {
43             cin>>mp[i][j];
44         }
45     }
46     gauss();
47     return 0;
48 }
```

## 5.21 拓展中国剩余定理 (CRT)

CRT 求的问题是有一个数余上不同的模值  $M$  之后, 求余数  $R$  值

通过公式分析可以推出最终的模值为  $M=\text{LCM}(m_1,m_2,m_3\dots)$ ,  $R=\text{LCM}*k+q.m_1+r_1$

```
1 long long gcd(long long a,long long b)
2 {
3     return b?gcd(b,a%b):a;
4 }
5
6 __int128 exgcd(__int128 a,__int128 b,__int128 &x,__int128
    &y)
7 {
8     if(b==0)
9     {
10         x=1;
11         y=0;
12         return a;
13     }
14     __int128 d,x1,y1;
15     d=exgcd(b,a%b,x1,y1);
16     x=y1;
17     y=x1-a/b*y1;
18     return d;
19 }
20 const int N=15;
21 int n;
22 void solve()
23 {
24     long long r,m;
25     __int128 m1,r1;
26     cin>>n;
27     cin>>m>>r;
28     m1=m,r1=r;
29     for(int i=1;i<n;i++)
30     {
31         __int128 m2,r2,x,y;
32         cin>>m>>r;
```

```
33     m2=m,r2=r;
34     long long d=exgcd(m1,m2,x,y);
35     if((r2-r1)%d){
36         cout<<-1;return;
37     }
38     x=x*(r2-r1)/d;
39     x=(x%(m2/d)+(m2/d))%(m2/d);
40     r1=m1*x+r1;
41     m1=m1*m2/d;
42     r1=(r1%m1+m1)%m1;
43 }
44 r=(r1%m1+m1)%m1;
45 cout<<r;
46
47 }
```

## 5.22 拓展欧拉定理

高次运算

```
1 long long ksm(long long a,long long b,long long mod)
2 {
3     long long res=1;
4     a%=mod;
5     while(b)
6     {
7         if(b&1) res=res*a%mod;
8         a=a*a%mod;
9         b>>=1;
10    }
11    return res;
12 }
13
14 int main()
15 {
16     ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
17     long long a,m,b;
18     string s;
19     cin>>a>>m>>s;
20     long long ans=m,n=m;
21     for(int i=2;1ll*i*i<=n;i++)
22     {
23         if(n%i==0)
24         {
25             ans/=i;
26             ans*=(i-1);
27             while(n%i==0) n/=i;
28         }
29     }
30     if(n!=1) ans/=n,ans*=(n-1);
31     b=0;
32     int f=0;
33     for(int i=0;i<s.size();i++)
34     {
```

```
35         b=(b*10ll+s[i]-'0');
36         if(b>ans)
37         {
38             f=1;b%=ans;
39         }
40     }
41     if(f) b+=ans;
42     cout<<ksm(a,b,m);
43     return 0;
44 }
```

## 5.23 卢卡斯求组合数

```
1  int mod=1e5+3;
2  const int N=1e5+3;
3  long long fac[N],inv[N];
4  long long ksm(long long a,long long b)
5  {
6      long long res=1;
7      a%=mod;
8      while(b)
9      {
10         if(b&1) res=res*a%mod;
11         a=a*a%mod;
12         b>>=1;
13     }
14     return res;
15 }
16 void init()
17 {
18     fac[0]=inv[0]=1;
19     for(int i=1;i<N;i++)
20     {
21         fac[i]=fac[i-1]*i%mod;
22         inv[i]=inv[i-1]*ksm(i,mod-2)%mod;
23     }
24 }
25 long long C(long long n,long long m)
26 {
27     return fac[n]*inv[m]%mod*inv[n-m]%mod;
28 }
29 long long P(long long n,long long m)
30 {
31     return fac[n]*inv[n-m]%mod;
32 }
33 int n,m;
34 int gcd(int a,int b)
35 {
```



```
36     return b?gcd(b,a%b):a;
37 }
38 long long lucas(long long n,long long m)
39 {
40     if(m==0) return 1;
41     return lucas(n/mod,m/mod)*C(n%mod,m%mod)%mod;
42 }
43 void solve()
44 {
45     int n,m;
46     cin>>n>>m>>mod;
47     init();
48     cout<<(lucas(n+m,m))%mod<<'\\n';
49
50 }
```

## 5.24 错排

$n$  个不相同盒子和  $n$  个不相同球，对应球不能放到下标相同盒子中

```
1  const int N=2e3+10;
2  /*
3   就错排 假设n行为n个格子 n列为n个物品
4   然后有唯一的障碍就是第i个物体不能放在自己第i个格子
5   题目就可以转换成这个问题
6   */
7  vector<int> f[N];
8  vector<int> add(vector<int>a,vector<int>b)
9  {
10     vector<int>c;
11     int tp=0;
12     for(int i=0;i<max(a.size(),b.size());i++)
13     {
14         if(i<a.size()&&i<b.size()) tp+=a[i]+b[i];
15         else if(i<a.size()) tp+=a[i];
16         else tp+=b[i];
17         c.push_back(tp%10);
18         tp/=10;
19     }
20     while(tp) c.push_back(tp%10),tp/=10;
21     return c;
22 }
23 vector<int>mul(vector<int>a,long long x)
24 {
25     long long tp=0;
26     vector<int>c;
27     for(int i=0;i<a.size();i++)
28     {
29         tp+=a[i]*x;
30         c.push_back(tp%10);
31         tp/=10;
32     }
33     while(tp) c.push_back(tp%10),tp/=10;
34     return c;
```

```
35 }
36 void pr(vector<int>a)
37 {
38     for(int i=(int)a.size()-1;i>=0;i--) cout<<a[i];
39 }
40 int main()
41 {
42     ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
43     int n;
44     cin>>n;
45     f[1].push_back(0);
46     f[2].push_back(1);
47     for(int i=3;i<=n;i++)
48     {
49         f[i]=mul(add(f[i-1],f[i-2]),(i-1ll));
50     }
51     pr(f[n]);
52     return 0;
```

## 5.25 高精度

```
1  int tw=1e6,w=6;
2
3  vector<int>add(vector<int>a,vector<int>b)
4  {
5      if((int)a.size()<(int)b.size()) swap(a,b);
6      long long tp=0;
7      for(int i=0;i<(int)a.size();i++)
8      {
9          if(i<(int)b.size()){
10             tp+=b[i];
11         }
12         tp+=a[i];
13         a[i]=tp%tw;
14         tp/=tw;
15     }
16     while(tp) a.push_back(tp%tw),tp/=tw;
17     return a;
18 }
19 vector<int>add(vector<int>a,int b)
20 {
21     int tp=b;
22     for(int i=0;i<(int)a.size();i++)
23     {
24         tp+=a[i];
25         a[i]=tp%tw;
26         tp/=tw;
27         if(tp==0) break;
28     }
29     while(tp) a.push_back(tp%tw),tp/=tw;
30     return a;
31 }
32 //sub只针对a>=b
33 vector<int>sub(vector<int>a,vector<int>b)
34 {
35     for(int i=0;i<(int)a.size();i++)
```

```
36     {
37         if(i<(int)b.size()){
38             a[i]-=b[i];
39         }
40         if(a[i]<0) a[i]+=tw,a[i+1]--;
41     }
42     while((int)a.size()>1&&a.back()==0) a.pop_back();
43     return a;
44 }
45 vector<int>sub(vector<int>a,int b)
46 {
47     a[0]-=b;
48     for(int i=0;i<(int)a.size();i++)
49     {
50         if(a[i]<0) a[i]+=tw,a[i+1]--;
51         else break;
52     }
53     while((int)a.size()>1&&a.back()==0) a.pop_back();
54     return a;
55 }
56
57 vector<int>mul(vector<int>a,vector<int>b)
58 {
59     vector<int>c((int)a.size()+(int)b.size()+3);
60     for(int i=0;i<(int)a.size();i++)
61     {
62         for(int j=0;j<(int)b.size();j++)
63         {
64             long long tp=1ll*a[i]*b[j];
65             c[i+j+1]+=tp/tw;
66             c[i+j]+=tp%tw;
67             c[i+j+1]+=c[i+j]/tw;
68             c[i+j]%=tw;
69         }
70     }
71     while((int)c.size()>1&&c.back()==0) c.pop_back();
72     return c;
```

```
73 }
74 vector<int>mul(vector<int>a,int b)
75 {
76     long long tp=0;
77     for(int i=0;i<(int)a.size();i++)
78     {
79         tp+=1ll*b*a[i];
80         a[i]=tp%tw;
81         tp/=tw;
82     }
83     while(tp) a.push_back(tp%tw),tp/=tw;
84     return a;
85 }
86
87 vector<int>div(vector<int>a,int b)
88 {
89     long long tp=0;
90     for(int i=(int)a.size();i-1>=0;i--)
91     {
92         tp=tp*tw+a[i-1];
93         a[i-1]=tp/b;
94         tp%=b;
95     }
96     while((int)a.size()>1&& a.back()==0) a.pop_back();
97     return a;
98 }
99
100 void read(vector<int>&aa,string &a)
101 {
102     for(int i=(int)a.size();i-1>=0;i-=w)
103     {
104         int res=0;
105         for(int j=max(0,i-w);j<=i-1;j++)
106         {
107             res=res*10+(a[j]-'0');
108         }
109         aa.push_back(res);
```

```
110     }
111     // reverse(aa.begin(),aa.end());
112 }
113
114 /*
115 1 a>b
116 -1 b<a
117 0 a=b
118 */
119 int equal(vector<int>a,vector<int>b)
120 {
121     if((int)a.size()>(int)b.size()) return 1;
122     else if((int)a.size()<(int)b.size()) return -1;
123     else{
124         for(int i=(int)a.size();i-1>=0;i--)
125         {
126             if(a[i-1]>b[i-1]) return 1;
127             else if(a[i-1]<b[i-1]) return -1;
128         }
129         return 0;
130     }
131 }
132
133 void pr(vector<int>a)
134 {
135     if((int)a.size()==1&&a[0]==0)
136     {
137         cout<<0;
138     }
139     else{
140         int f=0;
141         for(int i=(int)a.size();i-1>=0;i--)
142         {
143             int x=a[i-1];
144             int tww=tw;
145             for(int j=0;j<w;j++)
146             {
```

```
147         tww/=10;
148         int y=x/tww;
149         if(y) f=1;
150         if(f) cout<<y;
151         x%=tww;
152     }
153 }
154 }
155 cout<<'\\n';
156 }
157
158 void solve()
159 {
160     string a,b;
161     cin>>a>>b;
162     vector<int>aa,bb;
163     read(aa,a);read(bb,b);
164 }
```



## 5.26 高精度 GCD

```
1  const int mod=998244353,N=1e5+10;
2  long long n;
3  long long gcd(long long a,long long b)
4  {
5      return b?gcd(b,a%b):a;
6  }
7  //typedef pair<long long,int> pli;
8  int a[N];
9  int mp[N];
10 int tw=1000000;
11 int eva(vector<int>a,vector<int>b)
12 {
13     if(a.size()>b.size()) return 1;
14     else if(a.size()<b.size()) return -1;
15     else{
16         for(int i=a.size();i-1>=0;i--)
17         {
18             if(a[i-1]>b[i-1]) return 1;
19             else if(a[i-1]<b[i-1]) return -1;
20         }
21     }
22     return 0;
23 }
24
25 vector<int>mul(vector<int>a,int w)
26 {
27     long long tp=0;
28     vector<int>c;
29     for(int i=0;i<a.size();i++)
30     {
31         tp+=a[i]*w;
32         c.push_back(tp%tw);
33         tp/=tw;
34     }
35     while(tp) c.push_back(tp%tw),tp/=tw;
```

```
36     return c;
37 }
38
39 vector<int>div(vector<int>a,int w)
40 {
41     long long tp=0;
42     vector<int>c;
43     for(int i=a.size()-1;i>=0;i--)
44     {
45         tp=tp*tw+a[i];
46         c.push_back(tp/w);tp%=w;
47     }
48     reverse(c.begin(), c.end());
49     while(c.back()==0&& c.size()!=1) c.pop_back();
50     return c;
51 }
52
53 vector<int>sub(vector<int>a,vector<int>b)
54 {
55     vector<int>c;
56     for(int i=0;i<(int)a.size();i++)
57     {
58         int tp=0;
59         if(i<(int)a.size()&&i<(int)b.size())
60         {
61             tp+=a[i]-b[i];
62         }
63         else tp+=a[i];
64         if(tp<0) tp+=tw,a[i+1]--;
65         c.push_back(tp);
66     }
67     return c;
68 }
69
70
71 void solve()
72 {
```

```
73     string s,s1;
74     cin>>s>>s1;
75     vector<int>a,b;
76     for(int i=s.size();i-1>=0;i-=6)
77     {
78         long long re=0;
79         for(int j=max(0,i-6);j<i;j++)
80         {
81             re=re*10ll+s[j]-'0';
82         }
83         a.push_back(re);
84     }
85     for(int i=s1.size();i-1>=0;i-=6)
86     {
87         long long re=0;
88         for(int j=max(0,i-6);j<i;j++)
89         {
90             re=re*10ll+s1[j]-'0';
91         }
92         b.push_back(re);
93     }
94     long long tp=0;
95     while(1)
96     {
97         int t=eva(a,b);
98         if(t==-1) swap(a,b);
99         if(t==0||b.size()==1&&b[0]==0) break;
100        if(!(a[0]&1)&&!(b[0]&1))
101        {
102            tp++;
103            a=div(a,2);b=div(b,2);
104        }
105        else if(!(a[0]&1)){
106            a=div(a,2);
107        }
108        else if(!(b[0]&1)){
109            b=div(b,2);
```

```
110     }
111     else {
112         a=sub(a,b);
113     }
114 }
115 while(tp--)
116 {
117     a=mul(a,2);
118 }
119 vector<int>ans;
120 for(int i=(int)a.size();i-1>=0;i--) {
121     int w=tw;
122     w/=10;
123     ans.push_back(a[i-1]/w);
124     a[i-1]%=w;
125     w/=10;
126     ans.push_back(a[i-1]/w);
127     a[i-1]%=w;
128     w/=10;
129     ans.push_back(a[i-1]/w);
130     a[i-1]%=w;
131     w/=10;
132     ans.push_back(a[i-1]/w);
133     a[i-1]%=w;
134     w/=10;
135     ans.push_back(a[i-1]/w);
136     a[i-1]%=w;
137     w/=10;
138     ans.push_back(a[i-1]/w);
139     a[i-1]%=w;
140 }
141 reverse(ans.begin(), ans.end());
142 while(ans.back()==0&&ans.size()!=1) ans.pop_back();
143 for(int i=(int)ans.size();i-1>=0;i--) cout<<ans[i-1];
144 }
```

## 5.27 FFT(快速傅里叶变换)

系数表示法—> 点值表示法—> 系数表示法

采用单位根, 运用负数表达, 系数需要小数表达可以选择 `fft`

```

1  //(F(x)=f(x)*g(x) 求F(x)每一项系数 )
2  #include<bits/stdc++.h>
3  using namespace std;
4  const int N=3e6+10;
5  inline int read()
6  {
7      char c=getchar();int x=0,f=1;
8      while(c<'0' || c>'9'){if(c=='-')f=-1;c=getchar();}
9      while(c>='0'&& c<='9'){x=x*10+c-'0';c=getchar();}
10     return x*f;
11 }
12 const double Pi=acos(-1.0);
13 struct complex_
14 {
15     double x,y;
16     complex_ (double xx=0,double yy=0){x=xx,y=yy;}
17 }a[N],b[N];
18 complex_ operator + (complex_ a,complex_ b){ return
    complex_(a.x+b.x , a.y+b.y);}
19 complex_ operator - (complex_ a,complex_ b){ return
    complex_(a.x-b.x , a.y-b.y);}
20 complex_ operator * (complex_ a,complex_ b){ return
    complex_(a.x*b.x-a.y*b.y , a.x*b.y+a.y*b.x);}//不懂的看
    复数的运算那部分
21 int r[N];
22 void fft(complex_ *A,int limit,int type)
23 {
24     for(int i=0;i<limit;i++)
25         if(i<r[i]) swap(A[i],A[r[i]]);//求出要迭代的序列
26     for(int mid=1;mid<limit;mid<=<1)//待合并区间的中点
27     {
28         complex_ Wn( cos(Pi/mid) , type*sin(Pi/mid) ); //
            单位根

```

```

29     for(int R=mid<<1,j=0;j<limit;j+=R)//R是区间的右端
        点，j表示前已经到哪个位置了
30     {
31         complex_ w(1,0);//幂
32         for(int k=0;k<mid;k++,w=w*Wn)//枚举左半部分
33         {
34             complex_ x=A[j+k],y=w*A[j+mid+k];//蝴蝶效
                应
35             A[j+k]=x+y;
36             A[j+mid+k]=x-y;
37         }
38     }
39 }
40 }
41 int main()
42 {
43     int n,m;
44     n=read(),m=read();
45     for(int i=0;i<=n;i++) a[i].x=read();
46     for(int i=0;i<=m;i++) b[i].x=read();
47     int limit=1,l=0;
48     while(limit<=n+m) limit<<=1,l++;
49     for(int i=0;i<limit;i++)
50         r[i]= ( r[i>>1]>>1 )| ( (i&1)<<(l-1) ) ;
51     // 在原序列中 i 与 i/2 的关系是：i可以看做是i/2的二
        进制上的每一位左移一位得来
52     // 那么在反转后的数组中就需要右移一位，同时特殊处理一
        下复数
53     fft(a,limit,1);
54     fft(b,limit,1);
55     for(int i=0;i<limit;i++) a[i]=a[i]*b[i];
56     fft(a,limit,-1);
57     for(int i=0;i<=n+m;i++)
58         printf("%d_",(int)(a[i].x/limit+0.5));
59     return 0;
60 }

```

## 5.28 NTT（快速数论变换）

采用原根来表达，都为整数运算，可以相对 `fft` 减少精度的影响。

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  const int N = 3e6 + 10, mod = 998244353, G = 3, Gi =
    332748118;
4  inline int read() {
5      char c = getchar(); int x = 0, f = 1;
6      while(c < '0' || c > '9') {if(c == '-') f = -1; c =
        getchar();}
7      while(c >= '0' && c <= '9') x = x * 10 + c - '0', c =
        getchar();
8      return x * f;
9  }
10 int n,m,r[N];
11 long long a[N], b[N];
12 inline long long ksm(long long a, long long b) {
13     long long res = 1;
14     a%=mod;
15     while(b) {
16         if(b & 1) res = (res * a) % mod;
17         a = (a * a) % mod;
18         b >>= 1;
19     }
20     return res;
21 }
22 inline void NTT(long long *A,int limit,int type) {
23     for(int i = 0; i < limit; i++)
24         if(i < r[i]) swap(A[i], A[r[i]]);
25     for(int mid = 1; mid < limit; mid <= 1) {
26         long long Wn = ksm( type == 1 ? G : Gi , (mod - 1)
            / (mid << 1));
27         for(int j = 0; j < limit; j += (mid << 1)) {
28             long long w = 1;
29             for(int k = 0; k < mid; k++, w = (w * Wn) %
                mod) {

```

```
30         int x = A[j + k], y = w * A[j + k + mid]
           % mod;
31         A[j + k] = (x + y) % mod,
32         A[j + k + mid] = (x - y + mod) % mod;
33     }
34 }
35 }
36 }
37 int main() {
38     n = read(); m = read();
39     for(int i = 0; i <= n; i++) a[i] = (read() + mod) %
        mod;
40     for(int i = 0; i <= m; i++) b[i] = (read() + mod) %
        mod;
41     int limit=1,l=0;
42     while(limit <= m + n) limit <= 1, l++;
43     for(int i = 0; i < limit; i++) r[i] = (r[i >> 1] >> 1)
        | ((i & 1) << (l - 1));
44     NTT(a,limit,1);NTT(b,limit,1);
45     for(int i = 0; i < limit; i++) a[i] = (a[i] * b[i]) %
        mod;
46     NTT(a,limit,-1);
47     long long inv = ksm(limit, mod - 2);
48     for(int i = 0; i <= n + m; i++)
49         printf("%d_", (a[i] * inv) % mod);
50     return 0;
51 }
```



## 5.29 多项式求逆 (ntt)

```
1  #include<bits/stdc++.h>
2  using namespace std;
3  int read() {
4      int q=0;char ch=' ';
5      while(ch<'0' || ch>'9') ch=getchar();
6      while(ch>='0' && ch<='9') q=q*10+ch-'0',ch=getchar();
7      return q;
8  }
9  const int mod=998244353,G=3,N=2100000;
10
11 long long ksm(long long a,long long b)
12 {
13     long long res=1;
14     a%=mod;
15     while(b)
16     {
17         if(b&1) res=1ll*res*a%mod;
18         a=1ll*a*a%mod;
19         b>>=1;
20     }
21     return res;
22 }
23 long long a[N],b[N],c[N];
24 int r[N];
25 void NTT1(long long *a,int n,int x) {
26     for(int i=0;i<n;++i) if(i<r[i]) swap(a[i],a[r[i]]);
27     for(int i=1;i<n;i<=1) {
28         int gn=ksm(G,(mod-1)/(i<1));
29         for(int j=0;j<n;j+=(i<1)) {
30             int t1,t2,g=1;
31             for(int k=0;k<i;++k,g=1LL*g*gn%mod) {
32                 t1=a[j+k],t2=1LL*g*a[j+k+i]%mod;
33                 a[j+k]=(t1+t2)%mod,a[j+k+i]=(t1-t2+mod)%mod;
34             }
35         }
36     }
37 }
```

```
35     }
36 }
37 if(x==1) return;
38 long long ny=ksm(n,mod-2); reverse(a+1,a+n);
39 for(int i=0;i<n;++i) a[i]=1LL*a[i]*ny%mod;
40 }
41 void work(int n,long long *a,long long *b) {
42     if(n==1) {b[0]=ksm(a[0],mod-2);return;}
43     work((n+1)>>1,a,b);
44     int l=0,limit=1;
45     while(limit<2*n) limit<=1,++l;
46     for(int i=1;i<limit;++i) r[i]=(r[i>>1]>>1)|((i&1)<<(l
        -1));
47     for(int i=0;i<n;++i) c[i]=a[i];
48     for(int i=n;i<limit;++i) c[i]=0;
49     NTT1(c,limit,1);
50     NTT1(b,limit,1);
51     for(int i=0;i<limit;++i)
52         b[i]=1LL*(2-1LL*c[i]*b[i]%mod+mod)%mod*b[i]%mod;
53     NTT1(b,limit,-1);
54     for(int i=n;i<limit;++i) b[i]=0;
55 }
56 int main()
57 {
58     int n;
59     n=read();
60     for(int i=0;i<n;++i) a[i]=read();
61     work(n,a,b);
62     for(int i=0;i<n;++i) printf("%lld_",b[i]);
63     return 0;
64 }
```

## 5.30 bitset 处理多维偏序

```
1  #include<bits/stdc++.h>
2  using namespace std;
3
4  const int N=5010,M=510;
5  int n,m;
6  int mp[M][N];
7  long long p[N],dp[N];
8  bitset<N>tp[N];
9  vector<int>tr[N];
10 int in[N];
11 typedef pair<int,int> pii;
12 bool cmp(pii x,pii y)
13 {
14     if(x.first==y.first)
15     {
16         return x.second<y.second;
17     }
18     return x.first<y.first;
19 }
20 int main()
21 {
22     ios::sync_with_stdio(false);cin.tie(0);cout.tie(0);
23     cin>>n>>m;
24     for(int i=1;i<=m;i++)
25     {
26         cin>>p[i];
27         dp[i]=p[i];
28     }
29     for(int i=1;i<=n;i++)
30     {
31         for(int j=1;j<=m;j++)
32         {
33             cin>>mp[i][j];
34         }
35     }
```

```
36     for(int i=1;i<=m;i++)
37     {
38         for(int j=1;j<=m;j++){
39             tp[i][j]=1;
40         }
41     }
42     for(int i=1;i<=n;i++)
43     {
44         vector<pii>v;
45         for(int j=1;j<=m;j++)
46         {
47             v.push_back({mp[i][j],j});
48         }
49         sort(v.begin(),v.end(),cmp);
50         bitset<N>tpp;
51         int l=0;
52         for(int j=0;j<(int)v.size();j++)
53         {
54             if(v[l].first!=v[j].first)
55             {
56                 while(l<j){
57                     tpp[v[l].second]=1;
58                     l++;
59                 }
60             }
61             tp[v[j].second]&=tpp;
62         }
63     }
64     long long ans=0;
65     for(int i=1;i<=m;i++)
66     {
67         for(int j=1;j<=m;j++)
68         {
69             if(tp[i][j]){
70                 tr[j].push_back(i);
71                 in[i]++;
72             }
```

```
73     }
74 }
75 queue<int>q;
76 for(int i=1;i<=m;i++)
77 {
78     if(in[i]==0) q.push(i);
79 }
80 while(q.size())
81 {
82     int v=q.front();
83     q.pop();
84     ans=max(ans,dp[v]);
85     for(int u:tr[v])
86     {
87         dp[u]=max(dp[u],dp[v]+p[u]);
88         if(--in[u]==0) q.push(u);
89     }
90 }
91 cout<<ans;
92 return 0;
93 }
```

## 6 计算几何

### 6.1 绕点旋转后坐标

```
1 pii f(double x1,double y1,double x0,double y0,double d)
2 {
3     d=d/180*acos(-1);//如果是360一周
4     double x=(x1-x0)*cos(d)-(y1-y0)*sin(d)+x0;
5     double y=(x1-x0)*sin(d)+(y1-y0)*cos(d)+y0;
6     return {x,y};
7 }
```

### 6.2 正方形已知两点求另外两点

```
1 void solve()
2 {
3     cin>>a>>b>>c>>d;
4     double x=(a-b+c+d)*0.5;
5     double y=(a+b-c+d)*0.5;
6     if(ceil(x) == floor(x)&&ceil(y) == floor(y))cout<<(int
7         )x<<"_"<<(int)y<<endl;
8     else cout<<"No_Answer!"<<endl;
9 }
```

### 6.3 三角形

```
1 class triangle{
2     double S(pii x,pii y,pii z)
3     {
4         double s=0.5*abs((y.first-x.first)*(z.second-x.
5             second)-(z.first-x.first)*(y.second-x.second));
6         return s;
7     }
8     double S(double x,double y,double z)
9     {
```

```
9      //海伦公式
10     double p = 0.5*(x+y+z);
11     double s;
12     s=sqrt(p*(p-x)*(p-y)*(p-z));
13     // 或者
14     s=0.25*sqrt((x+y+z)*(x+y-z)*(x+z-y)*(y+z-x));
15     return s;
16 }
17 //能否构成三角形
18 bool ok(pii x,pii y,pii z)
19 {
20     if(x.first!=y.first&& y.first!=z.first&&z.first!=x.
        first)
21     {
22         double k1 = 1.0 * (y.second - x.second) / (y.
            first - x.first);
23         double k2 = 1.0 * (z.second - y.second) / (z.
            first - y.first);
24         double k3 = 1.0 * (z.second - x.second) / (z.
            first - x.first);
25         if(k1 == k2 && k2 == k3)
26             return 0;
27         else
28             return 1;
29     }
30     else if(x.first == y.first && y.first == z.first)
31         return 0;
32     else
33     {
34         if(x.second == y.second && y.second == z.
            second)
35             return 0;
36         else
37             return 1;
38     }
39 }
40 //能否构成三角形
```

```
41     bool ok(int x1,int y1,int x2,int y2,int x3,int y3)
42     {
43         if (x1 != x2 && x2 != x3 && x3 != x1)
44         {
45             double k1 = 1.0 * (y2 - y1) / (x2 - x1);
46             double k2 = 1.0 * (y3 - y2) / (x3 - x2);
47             double k3 = 1.0 * (y3 - y1) / (x3 - x1);
48             if (k1 == k2 && k2 == k3)
49                 return 0;
50             else
51                 return 1;
52         }
53         else if (x1 == x2 && x2 == x3)
54             return 0;
55         else
56         {
57             if (y1 == y2 && y2 == y3)
58                 return 0;
59             else
60                 return 1;
61         }
62     }
63 }
64 };
```



## 6.4 最小圆包括问题

给出空间中的点，求空间中某个点到这些点最大距离的最小值（三分）

```
1  const int N=110;
2  int n;
3  struct po{
4      double x,y,z;
5  }p[N];
6
7  double db(double x,double y)
8  {
9      return (x-y)*(x-y);
10 }
11
12 double getd(double x,double y,double z,int i)
13 {
14     return sqrt(db(x,p[i].x)+db(y,p[i].y)+db(z,p[i].z));
15 }
16
17 double getr(double x,double y,double z)
18 {
19     double ans=0;
20     for(int i=1;i<=n;i++)
21     {
22         ans=max(ans,getd(x,y,z,i));
23     }
24     return ans;
25 }
26
27 double getz(double x,double y)
28 {
29     double l=-1e5,r=1e5;
30     double ans=1e20;
31     while(r-l>1e-8)
32     {
33         double ml,mr;
34         ml=(l+r)/2.0;
```

```
35         mr=(ml+r)/2.0;
36         double ans1=getr(x,y,ml);
37         double ans2=getr(x,y,mr);
38         ans=min(ans,min(ans1,ans2));
39         if(ans1<ans2)
40         {
41             r=mr;
42         }
43         else l=ml;
44     }
45     return ans;
46 }
47
48 double gety(double x)
49 {
50     double l=-1e5,r=1e5;
51     double ans=1e20;
52     while(r-l>1e-8)
53     {
54         double ml,mr;
55         ml=(l+r)/2.0;
56         mr=(ml+r)/2.0;
57         double ans1=getz(x,ml);
58         double ans2=getz(x,mr);
59         ans=min(ans,min(ans1,ans2));
60         if(ans1<ans2)
61         {
62             r=mr;
63         }
64         else l=ml;
65     }
66     return ans;
67 }
68
69 double getx()
70 {
71     double l=-1e5,r=1e5;
```

```
72     double ans=1e20;
73     while(r-l>1e-8)
74     {
75         double ml,mr;
76         ml=(l+r)/2.0;
77         mr=(ml+r)/2.0;
78         double ans1=gety(ml);
79         double ans2=gety(mr);
80         ans=min(ans,min(ans1,ans2));
81         if(ans1<ans2)
82         {
83             r=mr;
84         }
85         else l=ml;
86     }
87     return ans;
88 }
89
90 int main()
91 {
92     scanf("%d",&n);
93     for(int i=1;i<=n;i++)
94     {
95         scanf("%lf%lf%lf",&p[i].x,&p[i].y,&p[i].z);
96     }
97     printf("%.20lf",getx());
98     return 0;
99 }
```

## 6.5 俩个三维圆相交弦长

```
1  #define double long double
2  double x[10],y[10],z[10],r1,r2;
3
4  struct pp{
5      double A,B,C;
6  }xc[10];
7  double ex=1e-8;
8  pp getxc(double x1,double yy1,double z1,double x2,double
    y2,double z2)
9  {
10     pp a;
11     a={yy1*z2-z1*y2,z1*x2-x1*z2,x1*y2-yy1*x2};
12     return a;
13
14 }
15
16 double getd(double x,double y,double z,double A,double B,
    double C,double b)
17 {
18     return abs(x*A+y*B+z*C+b)/sqrt(A*A+B*B+C*C);
19 }
20 double get(double x)
21 {
22     return x*x;
23 }
24 double getd(double x,double y,double z,double xx,double yy
    ,double zz)
25 {
26     return sqrt(get(x-xx)+get(y-yy)+get(z-zz));
27 }
28 double qr(pair<double,double>x,pair<double,double>y)
29 {
30     double res=min(x.second,y.second)-max(x.first,y.first)
        ;
31     return res>0?res:0;
```

```
32 }
33 void solve()
34 {
35     scanf("%Lf%Lf%Lf%Lf%Lf%Lf", &r2, &x[3], &y[3], &z[3], &x[4], &y[4], &z[4]);
36     xc[0] = getxc(x[2], y[2], z[2], x[4], y[4], z[4]);
37
38     double xx, yy, zz;
39     if(xc[0].A)
40     {
41         xx = 0;
42         yy = (-x[1]*x[2]*z[4] - y[1]*y[2]*z[4] - z[1]*z[2]*z[4] +
              x[3]*x[4]*z[2] + y[3]*y[4]*z[2] + z[3]*z[4]*z[2]) / (
              y[4]*z[2] - z[4]*y[2]);
43         zz = (-x[1]*x[2]*y[4] - y[1]*y[2]*y[4] - z[1]*z[2]*y[4] +
              x[3]*x[4]*y[2] + y[3]*y[4]*y[2] + z[3]*z[4]*y[2]) / (
              y[2]*z[4] - z[2]*y[4]);
44     }
45     else if(xc[0].B)
46     {
47         xx = (-x[1]*x[2]*z[4] - y[1]*y[2]*z[4] - z[1]*z[2]*z[4] +
              x[3]*x[4]*z[2] + y[3]*y[4]*z[2] + z[3]*z[4]*z[2]) / (
              z[2]*x[4] - x[2]*z[4]);
48         yy = 0;
49         zz = (-x[1]*x[2]*x[4] - y[1]*y[2]*x[4] - z[1]*z[2]*x[4] +
              x[3]*x[4]*x[2] + y[3]*y[4]*x[2] + z[3]*z[4]*x[2]) / (
              z[4]*x[2] - x[4]*z[2]);
50     }
51     else if(xc[0].C)
52     {
53         xx = (-x[1]*x[2]*y[4] - y[1]*y[2]*y[4] - z[1]*z[2]*y[4] +
              x[3]*x[4]*y[2] + y[3]*y[4]*y[2] + z[3]*z[4]*y[2]) / (
              x[4]*y[2] - y[4]*x[2]);
54         yy = (-x[1]*x[2]*x[4] - y[1]*y[2]*x[4] - z[1]*z[2]*x[4] +
              x[3]*x[4]*x[2] + y[3]*y[4]*x[2] + z[3]*x[4]*z[2]) / (
              x[2]*y[4] - y[2]*x[4]);
55         zz = 0;
```

```

56     }
57     xc[2]=getxc(x[2],y[2],z[2],xc[0].A,xc[0].B,xc[0].C);
58     xc[3]=getxc(x[4],y[4],z[4],xc[0].A,xc[0].B,xc[0].C);
59     double b1=-xx*xc[2].A-yy*xc[2].B-zz*xc[2].C;
60     double b2=-xx*xc[3].A-yy*xc[3].B-zz*xc[3].C;
61     double d1=getd(x[1],y[1],z[1],xc[2].A,xc[2].B,xc[2].C,
        b1);
62     double d2=getd(x[3],y[3],z[3],xc[3].A,xc[3].B,xc[3].C,
        b2);
63     double xx1,yy1,zz1,xx2,yy2,zz2;
64     double tt=-(xc[2].A*(x[1]-xx)+xc[2].B*(y[1]-yy)+xc[2].
        C*(z[1]-zz))/(xc[2].A*xc[2].A+xc[2].B*xc[2].B+xc
        [2].C*xc[2].C);
65     double tt1=-(xc[3].A*(x[3]-xx)+xc[3].B*(y[3]-yy)+xc
        [3].C*(z[3]-zz))/(xc[3].A*xc[3].A+xc[3].B*xc[3].B+
        xc[3].C*xc[3].C);
66     xx1=tt*xc[2].A+x[1];yy1=tt*xc[2].B+y[1];zz1=tt*xc[2].C
        +z[1];
67     xx2=tt1*xc[3].A+x[3];yy2=tt1*xc[3].B+y[3];zz2=tt1*xc
        [3].C+z[3];
68     if(r1-d1<=0||r2-d2<=0)
69     {
70         puts("0");
71     }
72     else{
73         double dd1=sqrt(r1*r1-d1*d1),dd2=sqrt(r2*r2-d2*d2)
            ;
74         double ddd=getd(xx1,yy1,zz1,xx2,yy2,zz2);
75         double res=qr({-dd1,dd1},{ddd-dd2,ddd+dd2});
76         printf("%.20Lf\n",res);
77     }
78 }
79
80 int main()
81 {
82     while(~scanf("%Lf%Lf%Lf%Lf%Lf%Lf",&r1,&x[1],&y[1],&
        z[1],&x[2],&y[2],&z[2]))

```

```
83         solve();  
84         return 0;  
85     }
```

## 7 网络流建图

### 7.1 最小路径覆盖

**最小路径覆盖：**在一个有向无环图中，找出最少的路径，使得这些路径经过了所有的点。最小路径覆盖分为**最小不相交路径覆盖**和**最小可相交路径覆盖**，区别是这些路径是否可以相交

#### 7.1.1 最小不相交路径覆盖

**建图方法：**把原图的每个点  $u$  拆成两个点  $u_1, u_2$ ，如果有一条有向边  $(a, b)$ ，则连边  $(a_2, b_1)$ ，容易发现这是一个二分图，那么用下面的定理就可以求出答案

**定理：**最小路径覆盖 = 原图节点数 - 新图最大匹配数

**证明：**

一开始每个点都是一条路径，每次找一条匹配边，代表合并两条路径

由于路径不相交（即每个点的入度和出度至少有一个为 1），所以二分图上的边也不相交（如果相交则说明某个点的入度或出度大于 1），这正好是匹配的定义

每条匹配边代表答案  $-1$ ，所以最小路径覆盖 = 原图节点数 - 新图最大匹配数

#### 7.1.2 最小可相交路径覆盖

对原图传递闭包，即若原图中  $(u, v)$  连通，则增加边  $(u, v)$ 。这可以用 *Floyd* 算法  $O(n^3)$  实现。然后对新图做最小不相交路径覆盖即可。因为在原图中相交的路径在传递闭包后可以拆分成另一条边，这样就不相交了

#### 7.1.3 最多不相交路径

这种问题变化比较多，但都能表示成以下形式：已知一些路径，每个节点只能属于一条路径，求能选择多少条路径使它们不相交。

主要的方法是拆点，将一个点拆成两个，然后连边，容量表示该点最多经过次数

### 7.2 最小割

#### 7.2.1 最大权闭合子图

**定义：**有一个有向图，每一个点都有一个权值，选择一个权值和最大的子图，使得每个点的后继都在子图里面，这个子图就叫最大权闭合子图。



**建图方法：**从源点  $s$  向每个正权点连一条容量为权值的边，每个负权点向汇点  $t$  连一条容量为权值的绝对值的边，有向图原来的边容量全部为无限大。

**定理：**最大权闭合子图 = 所有正权点之和 - 最小割

**关键性质：**如果  $s$  与  $i$  有边，表示  $i$  在子图中。如果  $i$  与  $t$  有边，表示  $i$  不在于子图中。即：割掉  $s$  与  $i$  表示不选  $i$ ，割掉  $i$  与  $t$  表示选  $i$ 。

**性质 1：**原图之间的边一定不会被割掉

边权为无穷大，当然不会被选进最小割

**性质 2：**只有  $s$  到  $t$  不联通时，才得到最大权闭合子图

反证法：若  $s$  到  $t$  连通，则一定存在节点  $i, j$  使  $s$  到  $i$  有边， $i$  到  $j$  有边（引理 1）， $j$  到  $t$  有边。而根据性质 1： $i$  在子图中， $j$  不在子图中，这与最大权闭合子图的定义矛盾，证毕由引理 2 可得，图的一个割就是一个闭合子图

由于一个割的边权和 = 不选的正权点 + 选的负权点绝对值 = 不选的正权点 - 选的负权点  
一个割的边权和 = 不选的正权点 + 选的负权点绝对值 = 不选的正权点 - 选的负权点。

闭合子图 = 正权点 + 负权点 = 所有正权和 - 不选的正权点 + 选的负权点 = 所有正权和 - 割的边权和  
闭合子图 = 正权点 + 负权点 = 所有正权和 - 不选的正权点 + 选的负权点 = 所有正权和 - 割的边权和

显然割的边权和最小的时候得到最大权闭合子图，证毕

### 7.3 最小割

**定理：**二分图最大独立集 =  $n$  - 二分图最大匹配

其实二分图独立集是特殊的一种最大权闭合子图。我们根据上文“收益”的思想，把选某个点的收益看为 1，左部节点为正权点，右部节点为负权点。按照最大权闭合子图的方式建图，答案为 正权和 - 最小割 =  $n$  - 最小割 =  $n$  - 最大流。我们发现把最大权闭合子图中  $INF$  的边换成 1 也不影响答案，因为图中其他边的容量都为 1。这样图就变成了二分图匹配中的图，最大流 = 二分图最大匹配

### 7.4 最大密度子图

**定义：**图的密度是图上的边的数量除以点数。求密度最大的子图。

**建图：**看到平均数想到 01 分数规划。二分答案  $mid$ ，那么问题转化为判定是否存在

在一个子图，使得边数  $-mid \cdot \text{点数} >$

0

边数  $-mid \cdot \text{点数} > 0$ 。那么可以把每条边的权看成 1，每个点的权看成  $-mid$ ，限制是选择一条边就必须选择边连接的两个点。于是把边看成左部点，点看成右部点，跑最大权闭合子图，若答案  $> 0$ ，则合法。

## 7.5 二元关系最小割模型

**定义：**有若干个变量，每个变量有 2 种取值，有若干个限制，每个限制形如“若变量  $x = a, y = b$ ，就要付出  $c$  的代价”。最大化所有变量的值之和减去最小代价。

**建图：**每个变量建一个点， $S$  到  $x$  连边表示  $x$  的一种取值的代价， $x$  到  $T$  连边表示  $x$  的另一种取值的代价。对于一个限制，在两个点之间连边。边权需要列方程解出。

## 7.6 二分图带权匹配

**定义：**每条边有边权，求匹配边权值之和最大的匹配

**建图：**在边上加上权，跑费用流即可

## 7.7 最大权不相交路径

**定义：**每条路径有一个权值（一般是边权和），在不相交路径数最多的情况下，最大化费用

**建图：**同最多不相交路径，在连接两个拆点的边上加上费用跑费用流即可

## 7.8 不等式差分模型（网络流解线性规划）

**定义：**对于一些不太好直接想到建图的问题，我们可以数学建模，列出方程然后用线性规划求解。这样的好处是思维量较小，只要做代数变换就可以建图，而不用考虑建图的实际意义。我们需要把式子做差，使得每个未知数仅在两个等式中出现。

根据网络流中每个点流量平衡的思想，我们可以把  $-x_i$  看成从点  $i$  流出  $x_i$  的流量， $+x_i$  看成流入  $x_i$  的流量。等式为 0 就代表流量平衡。

**建图：**每个等式为图中一个顶点，添加源点  $S$  和汇点  $T$ 。

如果一个等式右边为非负整数  $c$ ，从源点  $S$  向该等式对应的顶点连接一条容量为  $c$ ，权值为 0 的有向边；如果一个等式右边为负整数  $c$ ，从该等式对应的顶点向汇点  $T$  连接一条容量为  $c$ ，权值为 0 的有向边。

如果一个变量  $x_i$  在第  $j$  个等式中出现为  $x_i$ ，在第  $k$  个等式中出现为  $-x_i$ ，且在目标

函数里的系数为  $c_i$ ，从顶点  $j$  向顶点  $k$  连接一条容量为  $+\infty$ ，费用为  $c_i$  的有向边。

- 如果一个变量  $y_i$  在第  $j$  个等式中出现为  $y_i$ ，在第  $k$  个等式中出现为  $-y_i$ ，且在目标函数里没有出现，从顶点  $j$  向顶点  $k$  连接一条容量为  $+\infty$ ，权值为 0 的有向边。

## 7.9 有上下界的网络流

### 7.9.1 无源汇有上下界可行流

**定义：** 无源汇网络指的是没有源点和汇点，每个点都有入边和出边且满足流量守恒的网络。在这个网络上求一个流量方案，使得每条边的流量必须在  $[l_i, r_i]$  之间，且每个点流量守恒。

有上下界费用流的核心是”补偿”。我们先假设每条边的流量均为  $l_i$ ，那么一定会有一些点流量不守恒。现在我们需要构造一个附加网络，使得把附加网络 and 原网络叠加（即对应边流量相加）之后的图满足流量守恒。

因为 Dinic 只能求有源汇最大流，所以是不能直接求出附加网络的流量的。那么我们可以在附加网络上添加一些不在原网络上的边和点，来实现我们的限制。

记  $d_i = \text{点 } i \text{ 的入流} - \text{点 } i \text{ 的出流}$ ，然后建附加网络：

1. 新建源点  $ss$  和汇点  $tt$
2. 对于原图中的每条边  $e_i = (u, v)$ ，连边  $(u, v, r_i - l_i)$ ，也就是说附加网络包括原网络的边。
3. 新建边来满足流量守恒

若  $d_i = 0$  则该点流量平衡，不用处理

若  $d_i > 0$  则入流  $>$  出流，那么附加网络中  $i$  的出边需要增加流量，我们连边  $(ss, i, d_i)$ ，这样求最大流的时候出边的流量会增加  $d + i$ ，叠加后满足流量守恒

若  $d_i < 0$  则入流  $<$  出流，那么附加网络中  $i$  的入边需要增加流量，同理连边  $(i, tt, -d_i)$ ，这样求最大流的时候入边的流量会增加  $-d_i$ ，叠加后满足流量守恒

那么当且仅当步骤 3 中新建边满流时有解，总可行流为  $\max flow(ss, tt) + \sum l_i$ 。每条边在原图中流量 = 容量下界 + 附加流中它的流量

### 7.9.2 有源汇有上下界可行流

**定义：** 在有源汇网络上求一个流量方案，使得每条边的流量必须在  $[l_i, r_i]$  之间，且除源汇外每个点流量守恒。

设原网络的源和汇分别为  $s, t$  我们在原网络上加一条边  $(t, s, +\infty)$ ，相当于把到汇点的所有流量都流回源点，这样每个点流量都守恒。然后套无源汇的方法即可。注意总流量 =  $t$  到  $s$  的无穷边在原图中的流量

### 7.9.3 有源汇有上下界最大流和最小流

**定义：**在有源汇网络上求一个流量方案，使得每条边的流量必须在  $[l_i, r_i]$  之间，且除源汇外每个点流量守恒。在这个条件下使得总流量最大/最小。

先按上面的方法求出一个有源汇有上下界可行流。然后在附加网络上再跑一次  $s$  到  $t$  的最大流（注意不是  $ss, tt!$ ）。最大流 = 可行流 + 第二次跑的  $s$  到  $t$  最大流。

再跑一次最大流是因为附加网络上属于原图的边还有流量没被“榨干”。容易发现只要附加网络上不属于原图的边满流，属于原图的边怎么跑流量都是守恒的。因为第一次跑最大流已经保证所有点守恒，第二次跑最大流不会经过不属于原图的边，因此等价于对原图跑一次普通的最大流，除源汇外流量守恒。两次合起来总流量一定守恒，这就保证了正确性。

同理求最小流就跑一次  $t$  到  $s$  的最大流。最小流 = 可行流 - 第二次跑的  $t$  到  $s$  最大流。这是因为 Dinic 过程中反向边的流量增加等价于正向边的流量减少。

### 7.9.4 有源汇有上下界最小费用流

**定义：**在有源汇网络上求一个流量方案，使得每条边的流量必须在  $[l_i, r_i]$  之间，且除源汇外每个点流量守恒。每条边单位流量的费用为  $c_i$ 。在这个条件下使得总费用最小，费用定义同一般费用流。（不要求总流量最大）

这是有上下界费用流常被误解的一点，即最小费用流求的是费用最小的可行流，而不是最大流。

因此按有源汇可行流的方法建图，把原图中的边带上费用。总费用 =  $\text{mincost}(ss, tt) + \sum l_i c_i$