

「明治十七年的上海爱丽丝」

HiedanoAkyuu、Oneman233、KR12138

2020 年 7 月 21 日

目录

1	字符串	1			
1.1	KMP	1			
1.2	EX-KMP	1			
1.3	Manacher	1			
1.4	串的最小表示	2			
1.5	Lyndon 分解	2			
1.6	后缀数组	2			
1.6.1	倍增 SA	2			
1.6.2	DC3	2			
1.7	回文自动机	2			
1.8	AC 自动机	3			
1.8.1	多模匹配	3			
1.8.2	自动机上 DP	4			
1.9	后缀自动机	5			
2	计算几何	5			
2.1	二维几何	5			
2.2	三维几何	7			
3	图论	7			
3.1	最短路	7			
3.1.1	Dijkstra	7			
3.1.2	SPFA	7			
3.1.3	Floyd	7			
3.1.4	负环	7			
3.1.5	差分约束	8			
3.2	最小生成树	9			
3.2.1	Prim	9			
3.2.2	Kruskal	9			
3.2.3	最小生成树计数	10			
3.2.4	次小生成树	10			
3.2.5	最小乘积生成树	11			
3.3	树的直径	12			
3.4	LCA	12			
3.4.1	Tarjan 离线	12			
3.4.2	倍增 LCA	12			
3.5	无向图与有向图联通性	13			
3.5.1	割点	13			
3.5.2	桥	13			
3.5.3	e-DCC	14			
3.5.4	v-DCC	14			
3.5.5	SCC	14			
3.5.6	2-SAT	15			
3.5.7	支配树	16			
3.6	二分图	16			
3.6.1	最大匹配-匈牙利	16			
3.6.2	带权匹配-KM	16			
3.7	网络流	17			
3.7.1	最大流-Dinic	17			
3.7.2	最小费用最大流-Dij+Dinic	17			
3.7.3	最小费用最大流-SPFA+Dinic	18			
3.7.4	上下界流	19			
3.8	欧拉路	19			
3.9	Prufer 序列	20			
4	数据结构	20			
4.1	树状数组	20			
4.2	线段树	20			
4.2.1	带优先级线段树	20			
4.2.2	李超线段树	20			
4.2.3	吉司机线段树	20			
4.2.4	线段树维护扫描线	21			
4.2.5	区间图	22			
4.3	RMQ	23			
			4.3.1	一维	23
			4.3.2	二维	23
			4.4	树链剖分	23
			4.4.1	点剖分	23
			4.4.2	边剖分	24
			4.5	平衡树	24
			4.5.1	Treap	24
			4.5.2	Splay	26
			4.5.3	红黑树-pbds	26
			4.6	动态树	26
			4.7	主席树	26
			4.8	树套树	27
			4.8.1	线段树套 Treap	27
			4.8.2	树状数组套线段树	29
			4.9	K-D Tree	30
			4.10	分治	30
			4.10.1	CDQ	30
			4.10.2	点分治	32
			4.10.3	dsu on tree	32
			4.10.4	整体二分	32
			4.11	分块	32
			4.11.1	普通分块	32
			4.11.2	莫队	33
			4.12	线性基	33
			4.13	珂朵莉树	34
			4.14	跳舞链	35
5	动态规划	35			
5.1	SOS	35			
5.2	动态 DP	35			
5.3	插头 DP	35			
6	数学	35			
6.1	三分	35			
6.2	矩阵类	36			
6.3	质数筛	36			
6.3.1	埃筛	36			
6.3.2	线筛	36			
6.4	质数判定	36			
6.4.1	Miller Rabin	36			
6.5	质因数分解	36			
6.5.1	Pollard-Rho	36			
6.6	逆元	37			
6.6.1	EX-GCD 求逆元	37			
6.6.2	线性筛逆元	37			
6.6.3	阶乘逆元	37			
6.7	欧拉函数	37			
6.7.1	欧拉线筛	37			
6.7.2	求单个数的欧拉函数	37			
6.7.3	欧拉降幂	37			
6.7.4	一般积性函数求法	38			
6.8	EX-GCD	38			
6.9	同余方程组	38			
6.9.1	CRT	38			
6.9.2	EXCRT	38			
6.10	N 次剩余	38			
6.10.1	模奇质数的 2 次剩余	38			
6.10.2	N 次剩余	39			
6.11	数论分块	39			
6.12	高斯消元	39			
6.12.1	普通消元	39			
6.12.2	异或方程组消元	39			
6.13	莫比乌斯反演	39			
6.13.1	莫比乌斯函数	39			
6.13.2	杜教筛	39			
6.13.3	洲阁筛	39			

6.13.4 min25 筛	39
6.14 BSGS	39
6.15 FFT	40
6.16 FFT	40
6.17 DFT 次数优化 FFT	40
6.18 FWT	41
6.19 NTT	41
6.20 NTT	41
6.21 NTT 分治求卷积	42
6.22 NTT 求多项式逆	43
6.23 数值计算	44
6.23.1 辛普森	44
6.23.2 自适应辛普森	44
6.24 康拓展开	44
6.25 卢卡斯定理	44
6.25.1 Lucas(循环或递归实现)	44
6.25.2 EXLucas(分块实现)	45
6.26 博弈论	46
6.26.1 SG 函数	46
7 其他	46
7.1 快读快写	46
7.2 高精度	46
7.3 进制转换	50
7.4 约瑟夫环	51
7.5 悬线法	51
7.6 蔡勒公式	51
7.7 三角公式	51
7.8 海伦公式	51
7.9 匹克定理	51
7.10 组合计数	51
7.10.1 计数原理	51
7.10.2 卡特兰数	51
7.10.3 Polya	51
7.10.4 二项式反演公式	51
7.10.5 斯特林反演公式	51
7.10.6 组合数恒等式	51

1 字符串

1.1 KMP

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAXN=1000005;
5 char s1[MAXN],s2[MAXN];
6 int nxt[MAXN];
7
8 /*
9     nxt[i] s2[i-x..i-1]=s2[0..x-1]且x最大
10     即s2[0..i]的真前缀与真后缀的最大匹配
11     "ABAAB\0"=>[-1 0 0 1 1 2]
12 */
13
14 void get_fail(char *s,int l)
15 {
16     int i=0,j;
17     j=nxt[0]=-1;
18     while(i<l)
19     {
20         while(~j&&s[j]!=s[i]) j=nxt[j];
21         nxt[++i]=++j;
22     }
23 }
24
25 void kmp(char *s1,char *s2,int l1,int l2)
26 {
27     int i=0,j=0;
28     get_fail(s2,l2);
29     while(i<l1)
30     {
31         while(~j&&s1[i]!=s2[j]) j=nxt[j];
32         i++,j++;
33         if(j>=l2); //匹配上了
34     }
35 }
36
37 int main()
38 {
39     scanf("%s%s",s1,s2);
40     int l1=strlen(s1),l2=strlen(s2);
41     kmp(s1,s2,l1,l2);
42     for(int i=0;i<=l2;i++)
43         printf("%d ",nxt[i]);
44     return 0;
45 }

```

1.2 EX-KMP

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /*
5     ex[i]: s1[i..l1-1]与s2的最大公共前缀长度
6     exnext[i]: s2[i..l2-1]与s2的最大公共前缀长度
7     get_exnext(s2) 求exnext[]
8     exkmp(s1,s2) 求ex[]
9 */
10
11 const int N=500005;
12 char s1[N],s2[N];

```

```

13 int ex[N],exnext[N];
14
15 void get_exnext(char s[N])
16 {
17     int n=strlen(s),p=1,j,i;
18     exnext[0]=n;
19     for(i=0;i<n-1&&s[i]==s[i+1];i++);
20     exnext[1]=i;
21     for(i=2;i<n;i++)
22         if(exnext[i-p]+i<p+exnext[p])
23             exnext[i]=exnext[i-p];
24     else
25     {
26         j=exnext[p]+p-i;
27         if(j<0) j=0;
28         while(i+j<n&&s[j]==s[i+j]) j++;
29         exnext[i]=j;
30         p=i;
31     }
32 }
33
34 void exkmp(char s1[N],char s2[N])
35 {
36     int l1=strlen(s1),l2=strlen(s2),p=0,i,j;
37     get_exnext(s2);
38     for(i=0;i<l1&&i<l2&&s1[i]==s2[i];i++);
39     ex[0]=i;
40     for(int i=1;i<l1;i++)
41     {
42         if(exnext[i-p]+i<p+ex[p])
43             ex[i]=exnext[i-p];
44         else
45         {
46             j=ex[p]+p-i;
47             if(j<0) j=0;
48             while(i+j<l1&&s1[i+j]==s2[j]) j++;
49             ex[i]=j;
50             p=i;
51         }
52     }
53 }

```

1.3 Manacher

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int N=1000005;
5 int cnt,len,ans,p[N*2];
6 char s[N],ss[N*2];
7
8 void init() //将每两个字符中插入一个字符
9 {
10     len=strlen(s),cnt=1;
11     ss[0]='!',ss[cnt]='#';
12     for(int i=0;i<len;i++)
13         ss[++cnt]=s[i],ss[++cnt]='#';
14     ss[cnt+1]=0; //这一步多组一定要有 血一样的教训...
15 }
16
17 void manacher()
18 {
19     int pos=0,mx=0;
20     for(int i=1;i<=cnt;i++)

```

```

21 {
22     if(i<mx) p[i]=min(p[pos*2-i],mx-i);
23     else p[i]=1;
24     while(ss[i+p[i]]==ss[i-p[i]]) p[i]++;
25     if(mx<i+p[i]) mx=i+p[i],pos=i;
26     ans=max(ans,p[i]-1);
27 }
28 }
29
30 int main()
31 {
32     scanf("%s",s);
33     init();
34     manacher();
35     printf("%d\n",ans);
36     return 0;
37 }

```

1.4 串的最小表示

1.5 Lyndon 分解

```

1  /*
2   S是Lyndon串 iff S<S[i:] holds for 0<i<|S|
3   s1,s2,...,sn是S的Lyndon分解有S=s1s2...sn且s1>=s2
4   >=...>=sn
5   输入串 s[0..n-1]
6   获得分解后Lyndon子串的右端点下标
7   例如 acadacdac -> [3,7,9]
8  */
9
10 char s[1024*1024+5];
11 vector<int> pos;
12
13 void LyndonFactorization(char s[])
14 {
15     int n=strlen(s),i=0,j,k;
16     while(i<n)
17     {
18         j=i,k=i+1;
19         while(k<n&&s[j]<=s[k])
20         {
21             if(s[j]==s[k]) j++; else j=i;
22             k++;
23         }
24         while(i<=j) i+=k-j,pos.push_back(i-1);
25     }
26 }

```

1.6 后缀数组

1.6.1 倍增 SA

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4  /*
5   str[0..len-1] 原串
6   sa[1..len] 排名第i的后缀的下标[1..len]
7   Rank[1..len] 从i开始的后缀的排名[1..len]
8   height[1..len] 排名第i的后缀与排名第i-1的后缀的lcp
9   i开始的后缀与j开始的后缀的lcp (Rank[i]<Rank[j])
10  min{height[Rank[i]+1..Rank[j]]}

```

```

11  */
12
13 const int MAXN=100005;
14 const int inf=0x3f3f3f3f;
15 int wa[MAXN],wb[MAXN],wv[MAXN],wz[MAXN],sa[MAXN],Rank
16 [MAXN],height[MAXN];
17 char str[MAXN];
18
19 inline bool cmp(int *r,int a,int b,int l){return r[a
20 ]==r[b]&&r[a+l]==r[b+l];}
21
22 void da(const char r[],int sa[],int n,int m)
23 {
24     int i,j,p,*x=wa,*y=wb,*t;
25     for(i=0;i<m;i++) wz[i]=0;
26     for(i=0;i<n;i++) wz[x[i]=r[i]]++;
27     for(i=1;i<m;i++) wz[i]+=wz[i-1];
28     for(i=n-1;i>=0;i--) sa[--wz[x[i]]]=i;
29     for(j=1,p=1;p<n;j*=2,m=p)
30     {
31         for(p=0,i=n-j;i<n;i++) y[p++]=i;
32         for(i=0;i<n;i++) if(sa[i]>=j) y[p++]=sa[i]-j;
33         for(i=0;i<n;i++) wv[i]=x[y[i]];
34         for(i=0;i<m;i++) wz[i]=0;
35         for(i=0;i<n;i++) wz[wv[i]]++;
36         for(i=1;i<m;i++) wz[i]+=wz[i-1];
37         for(i=n-1;i>=0;i--) sa[--wz[wv[i]]]=y[i];
38         for(t=x,x=y,y=t,p=1,x[sa[0]]=0,i=1;i<n;i++)
39             x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;
40     }
41 }
42
43 void calheight(const char *r,int *sa,int n)
44 {
45     int i,j,k=0;
46     for(i=1;i<n;i++) Rank[sa[i]]=i;
47     for(i=0;i<n;height[Rank[i++]]=k)
48     {
49         for(k?k--:0,j=sa[Rank[i]-1];r[i+k]==r[j+k];k++);
50         for(int i=n;i>=1;--i) sa[i]++,Rank[i]=Rank[i-1];
51     }
52 }
53
54 int main()
55 {
56     scanf("%s",str);
57     int len=strlen(str);
58     da(str,sa,len+1,130); //字符的值域
59     calheight(str,sa,len);
60     for(int i=1;i<=len;i++)
61         printf("sa[%d] %d\n",i,sa[i]);
62     for(int i=1;i<=len;i++)
63         printf("Rank[%d] %d\n",i,Rank[i]);
64     for(int i=1;i<=len;i++)
65         printf("height[%d] %d\n",i,height[i]);
66     return 0;
67 }

```

1.6.2 DC3

1.7 回文自动机

```

1 #include <bits/stdc++.h>
2 //jisuanke 41389
3
4  /*

```

```

5 fail[x]: x节点失配之后跳转到不等于自身的最长后缀回文子
   串
6 len[x]: 以x结尾的最长回文子串长度
7 diff[x]: 与”以x结尾的最长回文子串“本质不同的子串个
   数
8 same[x]: 与”以x结尾的最长回文子串“本质相同的子串个数
   (注意上面两个完全相反)
9 son[x][c]: 编号为x的节点表示的回文子串在两边添加字符c
   之后变成的回文子串编号
10 s[x]: 第x次添加的字符, s数组即原字符串
11 tot: 总节点个数, 节点编号由0到tot-1
12 last: 最后一个新建节点的编号
13 cur: 当前节点在PAM上的父亲编号
14
15 */
16
17 #define int long long
18 using namespace std;
19 const int N=1e6+5;
20
21 struct PAM
22 {
23     int tot,last,n,cur;
24     int fail[N],len[N],same[N],diff[N],son[N][26];
25     char s[N];
26     int get(int p,int x)
27     {
28         while(s[x-len[p]-1]!=s[x])
29             p=fail[p];
30         return p;
31     }
32     int newnode(int x)
33     {
34         len[tot]=x;
35         return tot++;
36     }
37     void build()
38     {
39         scanf("%s",s+1);
40         s[0]=-1,fail[0]=1,last=0;
41         newnode(0),newnode(-1);
42         for(n=1;s[n];++n)
43         {
44             s[n]='a';
45             cur=get(last,n);
46             if(!son[cur][s[n]])
47             {
48                 int now=newnode(len[cur]+2);
49                 fail[now]=son[get(fail[cur],n)][s[n]];
50                 diff[now]=diff[fail[now]]+1;
51                 son[cur][s[n]]=now;
52             }
53             same[last=son[cur][s[n]]]++;
54         }
55         for(int i=tot-1;i>=0;--i)
56             same[fail[i]]+=same[i];
57     }
58 }pam;
59
60 int v[26],ans=0;
61 void dfs(int x,int now)
62 {
63     if(pam.len[x]>0) ans+=pam.same[x]*now;
64     for(int i=0;i<26;++i)
65     {
66         if(pam.son[x][i]!=0)

```

```

67     {
68         if(!v[i])
69         {
70             v[i]=1;
71             dfs(pam.son[x][i],now+1);
72             v[i]=0;
73         }
74         else dfs(pam.son[x][i],now);
75     }
76 }
77
78 signed main()
79 {
80     pam.build();
81     dfs(0,0);//even string
82     dfs(1,0);//odd string
83     printf("%lld",ans);
84     return 0;
85 }
86

```

1.8 AC 自动机

1.8.1 多模匹配

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /*
5  trie静态开点+trie图优化
6 */
7
8 int sz,hd=1,nxt[1000005][26],fail[1000005],id
   [1000005],n;
9 char s[1000005];
10
11 void trie_clean()
12 {
13     sz=1;
14     memset(nxt,0,sizeof(nxt));
15     memset(fail,0,sizeof(fail));
16     memset(id,0,sizeof(id));
17 }
18
19 void trie_insert(int head,char s[],int len,int idx)
20 {
21     int p=head;
22     for(int i=0;i<len;i++)
23     {
24         int c=s[i]-'a';
25         if(!nxt[p][c]) nxt[p][c]=++sz;
26         p=nxt[p][c];
27     }
28     id[p]+=idx;
29 }
30
31 void acatm_build(int head)
32 {
33     int p,tp;
34     queue<int> q;
35     q.push(head);
36     fail[head]=0;
37     while(!q.empty())
38     {
39         p=q.front();

```

```

40     q.pop();
41     for(int i=0;i<26;i++)
42         if(nxt[p][i])
43         {
44             fail[nxt[p][i]]=p==head?head:nxt[fail[p]]
45             ][i];
46             q.push(nxt[p][i]);
47         }
48         else
49             nxt[p][i]=p==head?head:nxt[fail[p]][i];
50     }
51 }
52 int acatm_match(int head,char s[],int len)
53 {
54     int p=head,ret=0;
55     for(int i=0;i<len;i++)
56     {
57         int c=(int)s[i]-'a';
58         p=nxt[p][c];
59         for(int tp=p;tp;tp=fail[tp])
60             if(id[tp]) ret++;
61     }
62     return ret;
63 }

```

1.8.2 自动机上 DP

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /*
5  每个串有个权值
6  求一个长度为n的串使得每个串的权值乘以出现次数之和最大
7 */
8
9 int fail[2005],nxt[2005][26],cnt[2005],sz,hd,n,m,dp
10 [55][2005],from[55][2005];
11 char s[105][15];
12 string dps[55][2005];
13 void clear()
14 {
15     sz=hd=1;
16     memset(dp,0xc0,sizeof(dp));
17     memset(fail,0,sizeof(fail));
18     memset(nxt,0,sizeof(nxt));
19     memset(cnt,0,sizeof(cnt));
20 }
21 void trie_insert(int head,char s[],int len,int idx)
22 {
23     int p=head;
24     for(int i=0;i<len;i++)
25     {
26         int c=s[i]-'a';
27         if(!nxt[p][c]) nxt[p][c]=++sz;
28         p=nxt[p][c];
29     }
30     cnt[p]+=idx;
31 }
32 void acatm_build(int head)
33 {
34 }
35

```

```

36 queue<int> q;
37 q.push(head);
38 while(!q.empty())
39 {
40     int p=q.front();
41     q.pop();
42     for(int i=0;i<26;i++)
43         if(nxt[p][i])
44         {
45             fail[nxt[p][i]]=p==head?head:nxt[fail[p]]
46             ][i];
47             cnt[nxt[p][i]]+=cnt[fail[nxt[p][i]]];
48             q.push(nxt[p][i]);
49         }
50         else
51             nxt[p][i]=p==head?head:nxt[fail[p]][i];
52     }
53 }
54 bool scmp(string a,string b)
55 {
56     if(a.length()==b.length()) return a<b;
57     else return a.length()<b.length();
58 }
59 void solve()
60 {
61     clear();
62     scanf("%d%d",&n,&m);
63     for(int i=0;i<m;i++)
64         scanf("%s",s[i]);
65     for(int i=0;i<m;i++)
66     {
67         int x;
68         scanf("%d",&x);
69         trie_insert(hd,s[i],strlen(s[i]),x);
70     }
71     acatm_build(hd);
72
73     for(int i=0;i<=n;i++)
74         for(int j=0;j<=sz;j++)
75             dps[i][j]=string("");
76     int ans=0;
77     string anss;
78     queue<pair<int,int>> q;
79     dp[0][1]=0;
80     for(int i=0;i<n;i++)
81         for(int j=1;j<=sz;j++)
82             for(int k=0;k<26;k++)
83                 if(dp[i][j]+cnt[nxt[j][k]]>dp[i+1][nxt[j]
84                 ][k])
85                     ||dp[i][j]+cnt[nxt[j][k]]==dp[i+1][nxt[j]
86                     ][k]&&scmp(dps[i][j]+char('a'+k),
87                     dps[i+1][nxt[j][k]])
88                     {
89                         dps[i+1][nxt[j][k]]=dps[i][j]+char('
90                         a'+k);
91                         dp[i+1][nxt[j][k]]=dp[i][j]+cnt[nxt[
92                         j][k]];
93                     }
94     for(int i=0;i<=n;i++)
95         for(int j=1;j<=sz;j++)
96             if(dp[i][j]>ans||dp[i][j]==ans&&scmp(dps[i]
97             ][j],ans))
98                 {
99

```

```

94         ans=dp[i][j];
95         anss=dps[i][j];
96     }
97     for(int i=0;i<anss.length();i++)
98         printf("%c",anss[i]);
99     printf("\n");
100 }
101
102 int main()
103 {
104     int _;
105     scanf("%d",&_);
106     while(--) solve();
107     return 0;
108 }

```

1.9 后缀自动机

2 计算几何

2.1 二维几何

```

1  #include<bits/stdc++.h>
2  using namespace std;
3
4  #define db double
5  const db EPS=1e-9;
6  inline int sign(db a){return a<-EPS?-1:a>EPS;}
7  inline int cmp(db a,db b){return sign(a-b);}
8  struct P
9  {
10     db x,y;
11     P(){}
12     P(db x,db y):x(x),y(y){}
13     P operator+(P p){return {x+p.x,y+p.y};}
14     P operator-(P p){return {x-p.x,y-p.y};}
15     P operator*(db d){return {x*d,y*d};}
16     P operator/(db d){return {x/d,y/d};}
17     bool operator<(P p) const
18     {
19         int c=cmp(x,p.x);
20         if(c) return c==1;
21         return cmp(y,p.y)==1;
22     }
23     bool operator==(P o) const
24     {
25         return cmp(x,o.x)==0&&cmp(y,o.y)==0;
26     }
27     db distTo(P p){return (*this-p).abs();}
28     db alpha(){return atan2(y,x);}
29     void read(){scanf("%lf%lf",&x,&y);}
30     void write(){printf("%.10f,%.10f\n",x,y);}
31     db abs(){return sqrt(abs2());}
32     db abs2(){return x*x+y*y;}
33     P rot90(){return P(-y,x);}
34     P unit(){return *this/abs();}
35     int quad() const {return sign(y)==1||((sign(y)==0&&
36         sign(x)>=0);}
37     db dot(P p){return x*p.x+y*p.y;}
38     db det(P p){return x*p.y-y*p.x;}
39     P rot(db an){return {x*cos(an)-y*sin(an),x*sin(an)
40         +y*cos(an)};}

```

```

41 int compareAngle(P a,P b)
42 {
43     if(a.quad()!=b.quad()) return a.quad()<b.quad();
44     return sign(a.det(b))>0;
45 }
46
47 //For segment
48 #define cross(p1,p2,p3) ((p2.x-p1.x)*(p3.y-p1.y)-(p3.
49     x-p1.x)*(p2.y-p1.y))
50 #define crossOp(p1,p2,p3) sign(cross(p1,p2,p3))
51
52 bool chkLL(P p1,P p2,P q1,P q2) //0:parallel
53 {
54     db a1=cross(q1,q2,p1),a2=-cross(q1,q2,p2);
55     return sign(a1+a2)!=0;
56 }
57
58 P isLL(P p1,P p2,P q1,P q2) //crossover point if
59     chkLL()
60 {
61     db a1=cross(q1,q2,p1),a2=-cross(q1,q2,p2);
62     return (p1*a2+p2*a1)/(a1+a2);
63 }
64
65 bool intersect(db l1,db r1,db l2,db r2)
66 {
67     if(l1>r1) swap(l1,r1);if(l2>r2) swap(l2,r2);
68     return !(cmp(r1,l2)==-1||cmp(r2,l1)==-1);
69 }
70
71 bool isSS(P p1,P p2,P q1,P q2)
72 {
73     return intersect(p1.x,p2.x,q1.x,q2.x)&&intersect(
74         p1.y,p2.y,q1.y,q2.y)&&
75         crossOp(p1,p2,q1)*crossOp(p1,p2,q2)<=0&&crossOp(q1
76             ,q2,p1)*crossOp(q1,q2,p2)<=0;
77 }
78
79 bool isSS_strict(P p1,P p2,P q1,P q2)
80 {
81     return crossOp(p1,p2,q1)*crossOp(p1,p2,q2)<0
82         &&crossOp(q1,q2,p1)*crossOp(q1,q2,p2)<0;
83 }
84
85 bool isMiddle(db a,db m,db b)
86 {
87     return sign(a-m)==0||sign(b-m)==0||((a<m!=b<m);
88 }
89
90 bool isMiddle(P a,P m,P b)
91 {
92     return isMiddle(a.x,m.x,b.x)&&isMiddle(a.y,m.y,b.y
93         );
94 }
95
96 bool onSeg(P p1,P p2,P q)
97 {
98     return crossOp(p1,p2,q)==0&&isMiddle(p1,q,p2);
99 }
100
101 bool onSeg_strict(P p1,P p2,P q)
102 {
103     return crossOp(p1,p2,q)==0&&sign((q-p1).dot(p1-p2)
104         )*sign((q-p2).dot(p1-p2))<0;
105 }

```



```

100
101 P proj(P p1,P p2,P q)
102 {
103     P dir=p2-p1;
104     return p1+dir*(dir.dot(q-p1)/dir.abs2());
105 }
106
107 P reflect(P p1,P p2,P q)
108 {
109     return proj(p1,p2,q)*2-q;
110 }
111
112 db nearest(P p1,P p2,P q)
113 {
114     P h=proj(p1,p2,q);
115     if(isMiddle(p1,h,p2))
116         return q.distTo(h);
117     return min(p1.distTo(q),p2.distTo(q));
118 }
119
120 db disSS(P p1,P p2,P q1,P q2) //dist of 2 segments
121 {
122     if(isSS(p1,p2,q1,q2)) return 0;
123     return min(min(nearest(p1,p2,q1),nearest(p1,p2,q2)),
124                min(nearest(q1,q2,p1),nearest(q1,q2,p2)));
125 }
126
127 db rad(P p1,P p2)
128 {
129     return atan2l(p1.det(p2),p1.dot(p2));
130 }
131
132 db area(vector<P> ps)
133 {
134     db ret=0;
135     for(int i=0;i<ps.size();i++)
136         ret+=ps[i].det(ps[(i+1)%ps.size()]);
137     return ret/2;
138 }
139
140 int contain(vector<P> ps,P p) //2:inside,1:on_seg,0:
141     outside
142 {
143     int n=ps.size(),ret=0;
144     for(int i=0;i<n;i++)
145     {
146         P u=ps[i],v=ps[(i+1)%n];
147         if(onSeg(u,v,p)) return 1;
148         if(cmp(u.y,v.y)<=0) swap(u,v);
149         if(cmp(p.y,u.y)>0||cmp(p.y,v.y)<=0) continue;
150         ret^=crossOp(p,u,v)>0;
151     }
152     return ret*2;
153 }
154
155 int convexContain(vector<P> &ps,P p) //1:inside|
156     on_seg,0:outside
157 {
158     int n=ps.size(),l=1,r=n-1,mid;
159     while(l<r)
160     {
161         mid=(l+r+1)>>1;
162         if(sign(cross(ps[0],ps[mid],p))<0) r=mid-1;
163         else l=mid;
164     }
165 }
166
167 if(l==1&&sign(cross(ps[0],ps[1],p))<0) return 0;
168 if(l==n-1&&onSeg(ps[0],ps[n-1],p)) return 1;
169 if(l!=n-1&&sign(cross(ps[1],ps[l+1],p))>=0) return
170     1;
171 return 0;
172 }
173
174 vector<P> convexHull(vector<P> ps)
175 {
176     int n=ps.size();if(n<=1) return ps;
177     sort(ps.begin(),ps.end());
178     vector<P> qs(n*2);int k=0;
179     for(int i=0;i<n;qs[k++]=ps[i++])
180         while(k>1&&crossOp(qs[k-2],qs[k-1],ps[i])<=0)
181             --k;
182     for(int i=n-2,t=k;i>=0;qs[k++]=ps[i--])
183         while(k>t&&crossOp(qs[k-2],qs[k-1],ps[i])<=0)
184             --k;
185     qs.resize(k-1);
186     return qs;
187 }
188
189 db convexDiameter(vector<P> ps)
190 {
191     int n=ps.size();if(n<=1) return 0;
192     int is=0,js=0;
193     for(int k=1;k<n;k++) is=ps[k]<ps[is]?k:is,js=ps[js]
194         <ps[k]?js:k;
195     int i=is,j=js;
196     db ret=ps[i].distTo(ps[j]);
197     do{
198         if((ps[(i+1)%n]-ps[i]).det(ps[(j+1)%n]-ps[j])
199             >=0) (++j)%=n;
200         else (++i)%=n;
201         ret=max(ret,ps[i].distTo(ps[j]));
202     }while(i!=is||j!=js);
203     return ret;
204 }
205
206 struct L // p[0]->p[1]
207 {
208     P p[2];
209     L(P k1,P k2){p[0]=k1,p[1]=k2;}
210     P& operator [] (int k){return p[k];}
211     int include(P k){return sign((p[1]-p[0]).det(k-p
212         [0]))>0;}
213     P dir(){return p[1]-p[0];}
214     L push(db dis) // push dis (left hand)
215     {
216         P delta=(p[1]-p[0]).rot90().unit()*dis;
217         return {p[0]+delta,p[1]+delta};
218     }
219 };
220
221 bool parallel(L l0,L l1){return sign(l0.dir().det(l1.
222     dir()))==0;}
223
224 bool sameDir(L l0,L l1){return parallel(l0,l1)&&sign(
225     l0.dir().dot(l1.dir()))==1;}
226
227 bool operator < (L l0,L l1)
228 {
229     if(sameDir(l0,l1)) return l1.include(l0[0]);
230     return compareAngle(l0.dir(),l1.dir());
231 }

```

```

218 P isLL(L l0,L l1){return isLL(l0[0],l0[1],l1[0],l1
219 [1]);}
220
221 bool check(L u,L v,L w){return w.include(isLL(u,v));}
222
223 vector<P> halfPlaneIS(vector<L> &l)
224 {
225     sort(l.begin(),l.end());
226     deque<L> q;
227     for(int i=0;i<(int)l.size();i++)
228     {
229         if(i&&sameDir(l[i],l[i-1])) continue;
230         while(q.size()>1&&!check(q[q.size()-2],q[q.
231 size()-1],l[i])) q.pop_back();
232         while(q.size()>1&&!check(q[1],q[0],l[i])) q.
233 pop_front();
234         q.push_back(l[i]);
235     }
236     while(q.size()>2&&!check(q[q.size()-2],q[q.size()
237 -1],q[0])) q.pop_back();
238     while(q.size()>2&&!check(q[1],q[0],q[q.size()-1]))
239 q.pop_front();
240     vector<P> ret;
241     for(int i=0;i<(int)q.size();i++) ret.push_back(
242         isLL(q[i],q[(i+1)%q.size()]));
243     return ret;
244 }

```

2.2 三维几何

3 图论

3.1 最短路

3.1.1 Dijkstra

```

1 #include <bits/stdc++.h>
2 #define mkp(a,b) make_pair(a,b)
3 #define fst first
4 #define snd second
5 //luogu P4779
6 using namespace std;
7 typedef pair<int,int> pii;
8 const int inf=0x3f3f3f3f;
9 const int N=1000005;
10
11 struct edge
12 {
13     int y,v;
14     edge(int Y,int V):y(Y),v(V){}
15 };
16 vector<edge> e[N];
17 void add(int x,int y,int v)
18 {
19     e[x].push_back(edge(y,v));
20 }
21
22 int n,m,s;
23 int dis[N];
24 bool vis[N];
25
26 void dij(int s)
27 {

```

```

28     memset(dis,0x3f,sizeof(dis));
29     dis[s]=0;
30     priority_queue<pii,vector<pii>,greater<pii>> q;
31     q.push(mkp(0,s));
32     while(!q.empty())
33     {
34         int x=q.top().snd;
35         q.pop();
36         if(vis[x]) continue;
37         vis[x]=1;
38         for(auto y:e[x])
39         {
40             if(dis[x]+y.v<dis[y.y])
41             {
42                 dis[y.y]=dis[x]+y.v;
43                 q.push(mkp(dis[y.y],y.y));
44             }
45         }
46     }
47 }
48
49 int main()
50 {
51     scanf("%d%d%d",&n,&m,&s);
52     for(int i=1,x,y,z;i<=m;++i)
53     {
54         scanf("%d%d%d",&x,&y,&z);
55         add(x,y,z);
56     }
57     dij(s);
58     for(int i=1;i<=n;++i)
59         printf("%d ",dis[i]==inf?2147483647:dis[i]);
60     return 0;
61 }

```

3.1.2 SPFA

3.1.3 Floyd

3.1.4 负环

```

1 #include <bits/stdc++.h>
2 //luogu P3385
3 using namespace std;
4 const int N=2005;
5
6 int _,n,m,dis[N],cnt[N];
7 bool vis[N];
8 vector<int> e[N],v[N];
9 void add(int x,int y,int z)
10 {
11     e[x].push_back(y);
12     v[x].push_back(z);
13 }
14
15 bool spfa(int s)
16 {
17     queue<int> q;
18     memset(dis,0x3f,sizeof(dis));
19     memset(vis,0,sizeof(vis));
20     memset(cnt,0,sizeof(cnt));
21     dis[s]=0;
22     vis[s]=cnt[s]=1;
23     q.push(1);
24     while(!q.empty())

```

```

25 {
26     int f=q.front();
27     q.pop();
28     vis[f]=0;
29     for(int i=0;i<e[f].size();++i)
30     {
31         int y=e[f][i];
32         if(dis[y]>dis[f]+v[f][i])
33         {
34             dis[y]=dis[f]+v[f][i];
35             if(!vis[y])
36             {
37                 vis[y]=1;
38                 q.push(y);
39                 cnt[y]++;
40                 if(cnt[y]>n) return 1;
41             }
42         }
43     }
44 }
45 return 0;
46 }
47
48 int main()
49 {
50     scanf("%d",&_);
51     while(_--)
52     {
53         scanf("%d%d",&n,&m);
54         for(int i=1;i<=n;++i)
55             e[i].clear(),v[i].clear();
56         for(int i=1,x,y,z;i<=m;++i)
57         {
58             scanf("%d%d%d",&x,&y,&z);
59             if(z<0) add(x,y,z);
60             else add(x,y,z),add(y,x,z);
61         }
62         if(spfa(1)) puts("YES");
63         else puts("NO");
64     }
65     return 0;
66 }

```

3.1.5 差分约束

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4 //Gym 102411 E
5
6 /*
7 差分约束找最小解
8 有负环代表无解
9  $a_i - a_j \leq k$  表示为  $j$  到  $i$  的长度为  $k$  的边 (最大解)
10 最小解就交换边的方向
11 都跑最短路
12 */
13
14 int n;
15 ll l,r,k,a[100005];
16 vector<int> E[100005];
17 vector<ll> V[100005];
18
19 void add(int x,int y,ll z)

```

```

20 {
21     E[x].push_back(y);
22     V[x].push_back(z);
23 }
24
25 ll dis[100005];
26 bool vis[100005];
27 int cnt[100005];
28 bool spfa()
29 {
30     for(int i=1;i<=n;++i) dis[i]=0x3f3f3f3f3f3f3f;
31     dis[0]=0;
32     vis[0]=1;
33     cnt[0]=0;
34     queue<int> q;
35     q.push(0);
36     while(!q.empty())
37     {
38         int f=q.front();
39         q.pop();
40         vis[f]=0;
41         for(int i=0;i<E[f].size();++i)
42         {
43             int e=E[f][i];
44             int v=V[f][i];
45             if(dis[e]>dis[f]+v)
46             {
47                 dis[e]=dis[f]+v;
48                 cnt[e]=cnt[f]+1;
49                 if(cnt[e]>n) return 1;
50                 if(!vis[e])
51                 {
52                     vis[e]=1;
53                     q.push(e);
54                 }
55             }
56         }
57     }
58     return 0;
59 }
60
61 int main()
62 {
63     scanf("%d%d%d",&n,&l,&r,&k);
64     for(int i=1;i<=n;++i) scanf("%d",&a[i]);
65     for(int i=1;i<=n;++i)
66     {
67         if(a[i]>a[i+1])
68         {
69             add(i,i+1,k);
70             add(i+1,i,-1);
71         }
72         else if(a[i]<a[i+1])
73         {
74             add(i+1,i,k);
75             add(i,i+1,-1);
76         }
77         else if(a[i]==a[i+1])
78         {
79             add(i+1,i,0ll);
80             add(i,i+1,0ll);
81         }
82     }
83     for(int i=1;i<=n;++i) add(i,0,r),add(0,i,-1);
84     if(spfa()) return puts("-1"),0;

```

```

85     for(int i=1;i<=n;++i) printf("%lld ",-dis[i]);
86     return 0;
87 }

```

3.2 最小生成树

3.2.1 Prim

```

1  #include <bits/stdc++.h>
2  //luogu P3366
3  using namespace std;
4
5  /*
6   prim/kruskal一定要注意解决重边
7  */
8
9  const int N=5005;
10 const int inf=0x3f3f3f3f;
11
12 int n,m;
13 int mp[N][N];
14 int dis[N];
15
16 int prim(int s)
17 {
18     int sum=0;
19     int cnt=0;
20     for(int i=1;i<=n;++i)
21         dis[i]=mp[s][i];
22     cnt++;
23     while(1)
24     {
25         int mn=inf;
26         int now=-1;
27         for(int i=1;i<=n;++i)
28         {
29             if(dis[i]!=0&&dis[i]<mn)
30             {
31                 mn=dis[i];
32                 now=i;
33             }
34         }
35         if(now==-1) break;
36         sum+=dis[now];
37         dis[now]=0;
38         cnt++;
39         for(int i=1;i<=n;++i)
40         {
41             if(dis[i]!=0&&mp[now][i]<dis[i])
42                 dis[i]=mp[now][i];
43         }
44     }
45     if(cnt<n) return -1;
46     else return sum;
47 }
48
49 int main()
50 {
51     scanf("%d%d",&n,&m);
52     memset(mp,0x3f,sizeof(mp));
53     for(int i=1;i<=n;++i)
54         mp[i][i]=0;
55     for(int i=1,x,y,z;i<=m;++i)
56     {
57         scanf("%d%d%d",&x,&y,&z);

```

```

58         mp[x][y]=min(mp[x][y],z);
59         mp[y][x]=min(mp[y][x],z);
60     }
61     int ans=prim(1);
62     if(ans==-1) puts("orz");
63     else printf("%d",ans);
64     return 0;
65 }

```

3.2.2 Kruskal

```

1  #include <bits/stdc++.h>
2  //luogu P3366
3  using namespace std;
4
5  /*
6   prim/kruskal一定要注意解决重边
7  */
8
9  const int N=200005;
10
11 int n,m;
12 struct node
13 {
14     int x,y,z;
15 }o[N];
16
17 bool cmp(node a,node b)
18 {
19     return a.z<b.z;
20 }
21
22 int f[5005];
23 int _find(int x)
24 {
25     if(x!=f[x]) f[x]=_find(f[x]);
26     return f[x];
27 }
28 void _merge(int x,int y)
29 {
30     x=_find(x),y=_find(y);
31     if(x!=y) f[x]=y;
32 }
33
34 int kk()
35 {
36     for(int i=1;i<=n;++i)
37         f[i]=i;
38     sort(o+1,o+1+m,cmp);
39     int sum=0;
40     for(int i=1;i<=m;++i)
41     {
42         if(_find(o[i].x)!=_find(o[i].y))
43         {
44             sum+=o[i].z;
45             _merge(o[i].x,o[i].y);
46         }
47     }
48     int tmp=_find(1);
49     for(int i=2;i<=n;++i)
50         if(_find(i)!=tmp)
51             return -1;
52     return sum;
53 }

```

```

54
55 int main()
56 {
57     scanf("%d%d",&n,&m);
58     for(int i=1;i<=m;++i)
59         scanf("%d%d%d",&o[i].x,&o[i].y,&o[i].z);
60     int ans=kk();
61     if(ans==-1) puts("orz");
62     else printf("%d",ans);
63     return 0;
64 }

```

3.2.3 最小生成树计数

3.2.4 次小生成树

```

1 #include <bits/stdc++.h>
2 #define int long long
3 using namespace std;
4 const int N=5e5+5;
5 const int M=1e6+5;
6 int inf=1e18;
7 //luogu P1480
8
9 /*
10  求解严格小于的次小生成树
11 */
12
13 struct edge
14 {
15     int to,val;
16 };
17 vector<edge> e[N];
18
19 struct node
20 {
21     int x,y,v;
22 }eg[M];
23
24 void add(int a,int b,int c)
25 {
26     e[a].push_back({b,c});
27     e[b].push_back({a,c});
28 }
29
30 int fa[N][35],dep[N],mx[N][35],mn[N][35];
31 void dfs(int x,int f)
32 {
33     fa[x][0]=f;
34     for(auto i:e[x])
35     {
36         if(i.to==f) continue;
37         dep[i.to]=dep[x]+1;
38         mx[i.to][0]=i.val;
39         mn[i.to][0]=-inf;
40         dfs(i.to,x);
41     }
42 }
43
44 int n,m;
45 void cal()
46 {
47     for(int i=1;i<=30;++i)
48     {
49         for(int j=1;j<=n;++j)

```

```

50     {
51         fa[j][i]=fa[fa[j][i-1]][i-1];
52         mx[j][i]=max(mx[j][i-1],mx[fa[j][i-1]][i-1]);
53         mn[j][i]=max(mn[j][i-1],mn[fa[j][i-1]][i-1]);
54         if(mx[j][i-1]>mx[fa[j][i-1]][i-1])
55             mn[j][i]=max(mn[j][i],mx[fa[j][i-1]][i-1]);
56         else if(mx[j][i-1]<mx[fa[j][i-1]][i-1])
57             mn[j][i]=max(mn[j][i],mx[j][i-1]);
58     }
59 }
60 }
61
62 int lca(int x,int y)
63 {
64     if(dep[x]<dep[y])
65         swap(x,y);
66     for(int i=30;i>=0;--i)
67         if(dep[fa[x][i]]>=dep[y])
68             x=fa[x][i];
69     if(x==y) return x;
70     for(int i=30;i>=0;--i)
71         if(fa[x][i]^fa[y][i])
72             x=fa[x][i],
73             y=fa[y][i];
74     return fa[x][0];
75 }
76
77 int qmax(int u,int v,int val)
78 {
79     int ans=-inf;
80     for(int i=30;i>=0;--i)
81     {
82         if(dep[fa[u][i]]>=dep[v])
83         {
84             if(val!=mx[u][i])
85                 ans=max(ans,mx[u][i]);
86             else
87                 ans=max(ans,mn[u][i]);
88             u=fa[u][i];
89         }
90     }
91     return ans;
92 }
93
94 bool cmp(node a,node b)
95 {
96     return a.v<b.v;
97 }
98
99 int F[N];
100 int _find(int x)
101 {
102     if(x!=F[x]) F[x]=_find(F[x]);
103     return F[x];
104 }
105
106 bool flg[M];
107 signed main()
108 {
109     scanf("%lld%lld",&n,&m);
110     for(int i=1;i<=m;++i)
111     {

```

```

112     scanf("%lld%lld%lld",&eg[i].x,&eg[i].y,&eg[i].
113         v);
114     }
115     sort(eg+1,eg+1+m,cmp);
116     for(int i=1;i<=n;++i)
117     {
118         F[i]=i;
119     }
120     int cnt=0;
121     for(int i=1;i<=m;++i)
122     {
123         int fx=_find(eg[i].x);
124         int fy=_find(eg[i].y);
125         if(fx!=fy)
126         {
127             cnt+=eg[i].v;
128             F[fx]=fy;
129             add(eg[i].x,eg[i].y,eg[i].v);
130             flg[i]=1;
131         }
132     }
133     mn[1][0]=-inf;
134     dep[1]=1;
135     dfs(1,-1);
136     cal();
137     int ans=inf;
138     for(int i=1;i<=m;++i)
139     {
140         if(flgs[i]) continue;
141         int x=eg[i].x;
142         int y=eg[i].y;
143         int v=eg[i].v;
144         int l=lca(x,y);
145         int mxu=qmax(x,l,v);
146         int mxv=qmax(y,l,v);
147         ans=min(ans,cnt-max(mxu,mxv)+v);
148     }
149     printf("%lld",ans);
150     return 0;
}

```

3.2.5 最小乘积生成树

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4  //luogu P5540
5
6  /*
7   每条边有两个边权
8   使得一棵生成树sigma(ai)*sigma(bi)最小
9  */
10
11 const int N=205;
12 const int M=10005;
13 const int inf=0x3f3f3f3f;
14
15 struct UFS
16 {
17     int f[N];
18     void init(int n)
19     {
20         for(int i=1;i<=n;++i)
21             f[i]=i;

```

```

22     }
23     int find(int x)
24     {
25         if(x!=f[x]) f[x]=find(f[x]);
26         return f[x];
27     }
28     void merge(int x,int y)
29     {
30         x=find(x),y=find(y);
31         if(x!=y)
32             f[x]=y;
33     }
34     bool check(int x,int y)
35     {
36         return find(x)==find(y);
37     }
38 };
39
40 struct edge
41 {
42     int u,v,w,a,b;
43 };
44
45 bool cmp(edge a,edge b)
46 {
47     return a.w<b.w;
48 }
49
50 struct point
51 {
52     int x,y;
53 };
54 point operator - (point a,point b)
55 {
56     return (point){a.x-b.x,a.y-b.y};
57 }
58 int cross(point a,point b)
59 {
60     return a.x*b.y-a.y*b.x;
61 }
62
63 int n,m;
64 UFS s;
65 point ans=point{inf,inf};
66 edge e[M];
67
68 point kk()
69 {
70     point res=point{0,0};
71     int tot=0;
72     sort(e+1,e+1+m,cmp);
73     s.init(n);
74     for(int i=1;i<=m;++i)
75     {
76         int u=e[i].u,v=e[i].v,a=e[i].a,b=e[i].b;
77         if(s.check(u,v)) continue;
78         s.merge(u,v);
79         res.x+=a;
80         res.y+=b;
81         ++tot;
82         if(tot==n-1) break;
83     }
84     ll tmp=1ll*ans.x*ans.y;
85     ll now=1ll*res.x*res.y;
86     if(tmp>now||(tmp==now&&ans.x>res.x))

```

```

87     ans=res;
88     return res;
89 }
90
91 void solve(point a,point b)
92 {
93     for(int i=1;i<=m;++i)
94     {
95         e[i].w=e[i].b*(b.x-a.x)+e[i].a*(a.y-b.y);
96     }
97     point c=kk();
98     if(cross(b-a,c-a)>=0) return;
99     solve(a,c);
100    solve(c,b);
101 }
102
103 int main()
104 {
105     scanf("%d%d",&n,&m);
106     for(int i=1;i<=m;++i)
107     {
108         scanf("%d%d%d",&e[i].u,&e[i].v,&e[i].a,&e[i]
109             ].b);
110         e[i].u++;
111         e[i].v++;
112     }
113     for(int i=1;i<=m;++i)
114     {
115         e[i].w=e[i].a;
116     }
117     point a=kk();
118     for(int i=1;i<=m;++i)
119     {
120         e[i].w=e[i].b;
121     }
122     point b=kk();
123     solve(a,b);
124     printf("%d %d",ans.x,ans.y);
125     return 0;
126 }

```

3.3 树的直径

3.4 LCA

3.4.1 Tarjan 离线

```

1 #include <bits/stdc++.h>
2 //luogu P3379
3 using namespace std;
4
5 /*
6  tarjan求lca要注意时间复杂度可能会爆炸，模板题开了O2才
7  过，并且要小心数组越界
8 */
9
10 const int N=1000005;
11
12 int n,m,s,x,y;
13 vector<int> e[N],q[N],id[N];
14 int ans[N*2];
15 bool vis[N];
16 int f[N];
17 int _find(int x)
18 {

```

```

18     if(x!=f[x]) f[x]=_find(f[x]);
19     return f[x];
20 }
21 void _merge(int x,int y)
22 {
23     x=_find(x),y=_find(y);
24     if(x!=y) f[x]=y;
25 }
26
27 void tarjan(int u)
28 {
29     vis[u]=1;
30     for(auto v:e[u])
31     {
32         if(!vis[v])
33         {
34             tarjan(v);
35             _merge(v,u);
36         }
37     }
38     for(int i=0;i<q[u].size();++i)
39     {
40         int v=q[u][i];
41         int k=id[u][i];
42         if(vis[v]&&ans[k]==0)
43             ans[k]=_find(v);
44     }
45 }
46
47 int main()
48 {
49     scanf("%d%d",&n,&m,&s);
50     for(int i=1;i<=n;++i) f[i]=i;
51     for(int i=1;i<=n-1;++i)
52         scanf("%d%d",&x,&y),e[x].push_back(y),e[y].
53             push_back(x);
54     for(int i=1;i<=m;++i)
55         scanf("%d%d",&x,&y),
56             q[x].push_back(y),q[y].push_back(x),
57             id[x].push_back(i),id[y].push_back(i);
58     tarjan(s);
59     for(int i=1;i<=m;++i)
60         printf("%d\n",ans[i]);
61     return 0;
62 }

```

3.4.2 倍增 LCA

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /*
5  预处理 O(nlogn)
6  单次查询 O(logn)
7  多组记得清空fa
8 */
9
10 const int MAXN=500005;
11 int n,q,dep[MAXN],s,lg[MAXN],fa[MAXN][32];
12 vector<int> e[MAXN];
13
14 void dfs(int now,int pa)
15 {
16     dep[now]=dep[pa]+1;

```

```

17 fa[now][0]=pa;
18 for(int i=1;(1<<i)<=dep[now];i++)
19     fa[now][i]=fa[fa[now][i-1]][i-1];
20 for(auto to:e[now])
21     if(to!=pa) dfs(to,now);
22 }
23
24 int lca(int x,int y)
25 {
26     if(dep[x]<dep[y]) swap(x,y);
27     while(dep[x]>dep[y]) x=fa[x][lg[dep[x]-dep[y]]-1];
28     if(x==y) return x;
29     for(int i=lg[dep[x]]-1;i>=0;i--)
30         if(fa[x][i]!=fa[y][i])
31             x=fa[x][i],y=fa[y][i];
32     return fa[x][0];
33 }
34
35 int main()
36 {
37     for(int i=1;i<MAXN;i++)
38         lg[i]=lg[i-1]+(1<<lg[i-1]==i);
39     scanf("%d%d%d",&n,&q,&s);
40     for(int i=0,x,y;i<n-1;i++)
41     {
42         scanf("%d%d",&x,&y);
43         e[x].push_back(y),e[y].push_back(x);
44     }
45     dep[0]=0;
46     dfs(s,0);
47     for(int i=0,x,y;i<q;i++)
48     {
49         scanf("%d%d",&x,&y);
50         printf("%d\n",lca(x,y));
51     }
52     return 0;
53 }

```

3.5 无向图与有向图联通性

3.5.1 割点

```

1 #include <bits/stdc++.h>
2 #define int long long
3 //luogu P3469
4
5 /*
6     tarjan求割点的算法中，如果不保证连通性，应该使用被注释
7     掉的遍历方法
8     part数组储存了被这个割点分成的不同的几块各自的大小
9 */
10
11 using namespace std;
12 const int N=100005;
13
14 int n,m,x,y;
15 vector<int> e[N],part[N];
16 bool is[N];
17 int dfn[N],low[N],timer=0;
18 int sz[N];
19
20 void tarjan(int u,int f)
21 {
22     dfn[u]=low[u]=++timer;
23     sz[u]++;//

```

```

23 int son=0,tmp=0;
24 for(auto v:e[u])
25 {
26     if(dfn[v]==0)
27     {
28         tarjan(v,u);
29         sz[u]+=sz[v];//
30         low[u]=min(low[u],low[v]);
31         if(low[v]>=dfn[u]&&u!=f)
32         {
33             is[u]=1;
34             tmp+=sz[v];//
35             part[u].push_back(sz[v]);//
36         }
37         if(u==f) son++;
38     }
39     low[u]=min(low[u],dfn[v]);
40 }
41 if(son>=2&&u==f) is[u]=1;//point on the top
42 if(is[u]&&n-tmp-1!=0)
43     part[u].push_back(n-tmp-1);//
44 }
45
46 signed main()
47 {
48     scanf("%lld%lld",&n,&m);
49     for(int i=1;i<=m;i++)
50     {
51         scanf("%lld%lld",&x,&y);
52         e[x].push_back(y),e[y].push_back(x);
53     }
54     /*
55     for(int i=1;i<=n;i++)
56         if(!dfn[i]) tarjan(i,i);
57     */
58     tarjan(1,0);
59     for(int i=1;i<=n;i++)
60     {
61         if(!is[i]) printf("%lld\n",2*(n-1));
62         else{
63             int tmp=0;
64             for(auto j:part[i])
65                 tmp+=j*(j-1);
66             printf("%lld\n",n*(n-1)-tmp);
67         }
68     }
69     return 0;
70 }

```

3.5.2 桥

```

1 #include <bits/stdc++.h>
2 #define mkp make_pair
3 //uva796
4 using namespace std;
5 const int N=100005;
6 typedef pair<int,int> pii;
7 inline int read(){
8     char ch=getchar();int s=0,w=1;
9     while(ch<48||ch>57){if(ch=='-')w=-1;ch=getchar();}
10    while(ch>=48&&ch<=57){s=(s<<1)+(s<<3)+ch-48;ch=
11        getchar();}
12    return s*w;

```



```

13 inline void write(int x){
14     if(x<0)putchar('-'),x=-x;
15     if(x>9)write(x/10);
16     putchar(x%10+48);
17 }
18
19 int n;
20 int dfn[N],low[N],timer=0;
21 int fa[N];
22 vector<int> e[N];
23 vector<pii> ans;
24
25 void tarjan(int u,int f)
26 {
27     fa[u]=f;
28     dfn[u]=low[u]=++timer;
29     for(auto v:e[u])
30     {
31         if(!dfn[v])
32         {
33             tarjan(v,u);
34             low[u]=min(low[u],low[v]);
35             //if(dfn[u]<low[v]) is[u][v]=1;
36             //u is v's father
37         }
38         else if(v!=f) low[u]=min(low[u],dfn[v]);
39     }
40 }
41
42 void init()
43 {
44     timer=0;
45     for(int i=0;i<n;++i) dfn[i]=low[i]=fa[i]=0;
46     for(int i=0;i<n;++i) e[i].clear();
47     ans.clear();
48 }
49
50 void gao()
51 {
52     for(int i=0;i<n;++i)
53         if(!dfn[i]) tarjan(i,-1);
54     for(int i=0;i<n;++i)
55     {
56         int F=fa[i];
57         if(F!=-1&&dfn[F]<low[i])
58             ans.emplace_back(min(F,i),max(F,i));
59     }
60     sort(ans.begin(),ans.end());
61     printf("%d critical links\n",(int)ans.size());
62     for(auto i:ans)
63         printf("%d - %d\n",i.first,i.second);
64     puts("");
65 }
66
67 int main()
68 {
69     while(~scanf("%d",&n))
70     {
71         if(n==0)
72         {
73             puts("0 critical links");
74             puts("");
75             continue;
76         }
77         init();

```

```

78         for(int i=0,x,y,z;i<n;++i)
79         {
80             scanf("%d %d",&x,&y);
81             for(int i=0;i<y;++i)
82                 z=read(),
83                 e[x].push_back(z),
84                 e[z].push_back(x);
85         }
86         gao();
87     }
88     return 0;
89 }

```

3.5.3 e-DCC

3.5.4 v-DCC

3.5.5 SCC

```

1 #include <bits/stdc++.h>
2 //luogu P2341
3 using namespace std;
4
5 /*
6     scc表示某标号的强连通分量中的点，co表示某个点属于哪个
7     强连通分量
8     gao函数是重建图，按照题意寻找有没有链
9 */
10
11 const int N=10005;
12
13 int n,m,x[N*5],y[N*5];
14 vector<int> e[N],scc[N];
15 int co[N],color=0;
16 stack<int> s;
17 bool vis[N];
18 int dfn[N],low[N],timer=0;
19
20 void tarjan(int u)
21 {
22     dfn[u]=low[u]=++timer;
23     s.push(u);
24     vis[u]=1;
25     for(auto v:e[u])
26     {
27         if(!dfn[v])
28         {
29             tarjan(v);
30             low[u]=min(low[u],low[v]);
31         }
32         else if(vis[v]) low[u]=min(low[u],dfn[v]);
33     }
34     if(low[u]==dfn[u])
35     {
36         ++color;
37         int t;
38         do
39         {
40             t=s.top();
41             s.pop();
42             co[t]=color;
43             vis[t]=0;
44             scc[color].push_back(t);
45         } while(u!=t);

```

```

46     }
47 }
48
49 int f[N];
50 int _find(int x)
51 {
52     if(x!=f[x]) f[x]=_find(f[x]);
53     return f[x];
54 }
55 void _merge(int x,int y)
56 {
57     x=_find(x),y=_find(y);
58     if(x!=y) f[x]=y;
59 }
60
61 int d[N];
62 void gao()
63 {
64     for(int i=1;i<=color;++i)
65         f[i]=i;
66     for(int i=1;i<=m;++i)
67     {
68         if(co[x[i]]!=co[y[i]])
69             _merge(co[x[i]],co[y[i]]),
70             d[co[x[i]]]++;
71     }
72     int F=_find(1);
73     for(int i=1;i<=color;++i)
74         if(_find(i)!=F) {puts("0");return;}
75     int ans=0,tmp=0;
76     for(int i=1;i<=color;++i)
77     {
78         if(d[i]==0)
79             ans+=scc[i].size(),tmp++;
80     }
81     if(tmp>1) ans=0;
82     printf("%d",ans);
83 }
84
85 int main()
86 {
87     scanf("%d%d",&n,&m);
88     for(int i=1;i<=m;++i)
89     {
90         scanf("%d%d",&x[i],&y[i]);
91         e[x[i]].push_back(y[i]);
92     }
93     for(int i=1;i<=n;++i)
94         if(!dfn[i]) tarjan(i);
95     gao();
96     return 0;
97 }

```

3.5.6 2-SAT

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 //luogu P4782
4
5 /*
6 2-SAT用于求解有n个布尔变量x1-xn和m个需要满足的条件
7 每个条件形式为xi=0(1)||xj=0(1), 是否有可行解
8 注意要开两倍空间建反向边
9 2-SAT输出方案的方法为: 先把缩点之后的新图进行拓扑排
   序, 然后判断每个点i, 如果i所在强连通分量的拓扑序在

```

```

10 i'所在的强连通分量的拓扑序之后, 那么第i场游戏使用
   该地图适合的第一种赛车, 否则使用第二种赛车。
11 但是由于Tarjan求强连通分量就是按拓扑排序的逆序给出的,
   所以直接使用强连通分量编号判断即可。
   即如果bel[]为每个点的所在强连通分量编号, 那么判断为:
   如果bel[i]<bel[i'], 那么使用该地图适合的第一种赛
   车, 否则使用第二种赛车。
12 */
13
14 const int N=2e6+5;
15
16 int n,m,a,va,b,vb;
17 int dfn[N],low[N],timer=0;
18 stack<int> s;
19 bool vis[N];
20 vector<int> e[N];
21 int co[N],color=0;
22
23 void add(int x,int y)
24 {
25     e[x].push_back(y);
26 }
27
28 void tarjan(int u)
29 {
30     dfn[u]=low[u]=++timer;
31     s.push(u);
32     vis[u]=1;
33     for(auto v:e[u])
34     {
35         if(!dfn[v])
36             tarjan(v),
37             low[u]=min(low[u],low[v]);
38         else if(vis[v])
39             low[u]=min(low[u],dfn[v]);
40     }
41     if(low[u]==dfn[u])
42     {
43         int v;
44         color++;
45         do
46         {
47             v=s.top();
48             s.pop();
49             vis[v]=0;
50             co[v]=color;
51         }
52         while(u!=v);
53     }
54 }
55
56 bool solve()
57 {
58     for(int i=1;i<=2*n;++i)
59         if(!dfn[i]) tarjan(i);
60     for(int i=1;i<=n;++i)
61         if(co[i]==co[i+n])
62             return 0;
63     return 1;
64 }
65
66 int main()
67 {
68     scanf("%d%d",&n,&m);
69     for(int i=1;i<=m;++i)

```

```

70 {
71     scanf("%d%d%d", &a, &va, &b, &vb);
72     int nota=va^1, notb=vb^1;
73     add(a+nota*n, b+vb*n); //not a and b
74     add(b+notb*n, a+va*n); //not b and a
75 }
76 if(solve())
77 {
78     puts("POSSIBLE");
79     for(int i=1; i<=n; ++i)
80         printf("%d ", co[i]>co[i+n]);
81 }
82 else puts("IMPOSSIBLE");
83 return 0;
84 }

```

3.5.7 支配树

3.6 二分图

3.6.1 最大匹配-匈牙利

```

1 #include <bits/stdc++.h>
2 //luogu P3386
3 using namespace std;
4
5 /*
6  hungary每一次遍历必须要清空vis数组
7 */
8
9 const int N=1005;
10
11 vector<int> e[N];
12 bool vis[N];
13 int match[N], rematch[N];
14
15 bool dfs(int u)
16 {
17     for(auto v:e[u])
18     {
19         if(!vis[v]){
20             vis[v]=1;
21             if(match[v]==0 || dfs(match[v]))
22             {
23                 match[v]=u;
24                 rematch[u]=v;
25                 return 1;
26             }
27         }
28     }
29     return 0;
30 }
31
32 int n, m, k;
33
34 int main()
35 {
36     scanf("%d%d%d", &n, &m, &k);
37     for(int i=1, x, y; i<=k; ++i)
38     {
39         scanf("%d%d", &x, &y);
40         if(x>n || y>m) continue;
41         e[x].push_back(y);
42     }
43     int ans=0;

```

```

44     for(int i=1; i<=n; ++i)
45     {
46         memset(vis, 0, sizeof(vis));
47         if(dfs(i)) ans++;
48     }
49     printf("%d", ans);
50     return 0;
51 }

```

3.6.2 带权匹配-KM

```

1 #include <bits/stdc++.h>
2 //hdu 2255
3 using namespace std;
4
5 /*
6  KM仅用于最大带权匹配一定是最大匹配的情况下
7 */
8
9 const int N=305;
10 const int inf=0x3f3f3f3f;
11
12 int n, mp[N][N];
13 int la[N], lb[N], delta;
14 bool va[N], vb[N];
15 int match[N];
16
17 bool dfs(int x)
18 {
19     va[x]=1;
20     for(int y=1; y<=n; ++y)
21     {
22         if(!vb[y]){
23             if(la[x]+lb[y]==mp[x][y])
24             {
25                 vb[y]=1;
26                 if(!match[y] || dfs(match[y]))
27                 {
28                     match[y]=x;
29                     return 1;
30                 }
31             }
32             else
33                 delta=min(delta, la[x]+lb[y]-mp[x][y]);
34         }
35     }
36     return 0;
37 }
38
39 int km()
40 {
41     for(int i=1; i<=n; ++i)
42     {
43         match[i]=0;
44         la[i]=-inf;
45         lb[i]=0;
46         for(int j=1; j<=n; ++j)
47         {
48             la[i]=max(la[i], mp[i][j]);
49         }
50     }
51     for(int i=1; i<=n; ++i)
52     {
53         while(1)

```

```

54     {
55         memset(va,0,sizeof(va));
56         memset(vb,0,sizeof(vb));
57         delta=inf;
58         if(dfs(i)) break;
59         for(int j=1;j<=n;++j)
60         {
61             if(va[j]) la[j]-=delta;
62             if(vb[j]) lb[j]+=delta;
63         }
64     }
65 }
66 int ans=0;
67 for(int i=1;i<=n;++i)
68     ans+=mp[match[i]][i];
69 return ans;
70 }
71
72 int main()
73 {
74     while(~scanf("%d",&n))
75     {
76         memset(mp,-0x3f,sizeof(mp));
77         for(int i=1;i<=n;++i)
78         {
79             for(int j=1;j<=n;++j)
80             {
81                 scanf("%d",&mp[i][j]);
82             }
83         }
84         printf("%d\n",km());
85     }
86     return 0;
87 }

```

3.7 网络流

3.7.1 最大流-Dinic

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4
5  /*
6   s,t 超级源、超级汇
7   cur[] 当前弧优化
8   时间复杂度 O(n^2*m)
9  */
10
11 const int MAXN=10005;
12 const ll inf=0x3f3f3f3f3f3f3f3f;
13 int n,m,s,t,tot,dis[MAXN],cur[MAXN];
14 struct edge
15 {
16     int to,cap,rev;
17     edge(){}
18     edge(int to,int cap,int rev):to(to),cap(cap),rev(rev){}
19 };
20 vector<edge> E[MAXN];
21
22 inline void add_edge(int x,int y,int f)
23 {
24     E[x].emplace_back(y,f,E[y].size());
25     E[y].emplace_back(x,0,E[x].size()-1);

```

```

26 }
27
28 int bfs()
29 {
30     for(int i=1;i<=n;i++) dis[i]=0x3f3f3f3f;
31     dis[s]=0;
32     queue<int> q;
33     q.push(s);
34     while(!q.empty())
35     {
36         int now=q.front();q.pop();
37         for(int i=0;i<E[now].size();i++)
38         {
39             edge &e=E[now][i];
40             if(dis[e.to]>dis[now]+1&&e.cap)
41             {
42                 dis[e.to]=dis[now]+1;
43                 if(e.to==t) return 1;
44                 q.push(e.to);
45             }
46         }
47     }
48     return 0;
49 }
50
51 ll dfs(int now,ll flow)
52 {
53     if(now==t) return flow;
54     ll rest=flow,k;
55     for(int i=cur[now];i<E[now].size();i++)
56     {
57         edge &e=E[now][i];
58         if(e.cap&&dis[e.to]==dis[now]+1)
59         {
60             cur[now]=i;
61             k=dfs(e.to,min(rest,(long long)e.cap));
62             e.cap-=k;
63             E[e.to][e.rev].cap+=k;
64             rest-=k;
65         }
66     }
67     return flow-rest;
68 }
69
70 ll dinic()
71 {
72     ll ret=0,delta;
73     while(bfs())
74     {
75         for(int i=1;i<=n;i++) cur[i]=0;
76         while(delta=dfs(s,inf)) ret+=delta;
77     }
78     return ret;
79 }

```

3.7.2 最小费用最大流-Dij+Dinic

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef pair<int,int> pii;
4
5  /*
6   出锅状态 勿用
7

```

```

8   第一遍跑的spfa,然后是加上势函数的dij,玄学
9   h[] 势函数
10  cur[] 当前弧优化
11  msmf 最大流时的最小费用
12  s,t 超级源、超级汇
13  时间复杂度  $O(n^2*m)$ 
14  */
15
16  const int MAXN=2005;
17  const int inf=0x3f3f3f3f;
18  int msmf,s,t,cur[MAXN],dis[MAXN],vis[MAXN],h[MAXN];
19  struct edge
20  {
21      int to,val,cap,rev;
22      edge(){}
23      edge(int to,int cap,int val,int rev):to(to),cap(
24          cap),val(val),rev(rev){}
25  };
26  vector<edge> E[MAXN];
27
28  inline void add_edge(int x,int y,int f,int cost)
29  {
30      E[x].emplace_back(y,f,cost,E[y].size());
31      E[y].emplace_back(x,0,-cost,E[x].size()-1);
32  }
33
34  int dij()
35  {
36      fill(dis,dis+t+1,inf);
37      priority_queue<pii,vector<pii>,greater<pii>> q;
38      q.emplace(0,s);dis[s]=0;
39      while(!q.empty())
40      {
41          pii p=q.top();q.pop();
42          int now=p.second;
43          if(dis[now]<p.first) continue;
44          for(int i=0;i<E[now].size();i++)
45          {
46              edge &e=E[now][i];
47              if(e.cap>0&&dis[e.to]>p.first+e.val+h[now]-
48                  h[e.to])
49              {
50                  dis[e.to]=p.first+e.val+h[now]-h[e.to];
51                  q.emplace(dis[e.to],e.to);
52              }
53          }
54      }
55      return dis[t]!=inf;
56  }
57
58  int dfs(int now,int flow)
59  {
60      if(now==t) return flow;
61      int rest=flow,k;
62      vis[now]=1;
63      for(int i=cur[now];i<E[now].size();i++)
64      {
65          edge &e=E[now][i];
66          if(e.cap&&dis[now]+e.val+h[now]-h[e.to]==dis[e
67              .to]&&!vis[e.to])
68          {
69              cur[now]=i;
70              k=dfs(e.to,min(e.cap,rest));
71              e.cap-=k;
72              E[e.to][e.rev].cap+=k;

```

```

70      rest-=k;
71      msmf+=k*e.val;
72  }
73  }
74  vis[now]=0;
75  return flow-rest;
76  }
77
78  int dinic()
79  {
80      int ret=0,delta;
81      while(dij())
82      {
83          for(int i=s;i<=t;i++) cur[i]=0;
84          while(delta=dfs(s,inf)) ret+=delta;
85          for(int i=s;i<=t;i++) h[i]+=(dis[i]==inf)?0:
86              dis[i];
87      }
88      return ret;

```

3.7.3 最小费用最大流-SPFA+Dinic

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4
5  /*
6   cur[] 当前弧优化
7   msmf 最大流时的最小费用
8   s,t 超级源、超级汇
9   多组记得清边和msmf
10  时间复杂度  $O(n^2*m)$ 
11  */
12
13  const ll inf=0x3f3f3f3f3f3f3f3fll;
14  ll msmf,dis[5005];
15  int s,t,n,m,cur[5005],vis[5005];
16  struct edge
17  {
18      int to,rev;
19      ll cap,cost;
20      edge(){}
21      edge(int to,ll cap,ll cost,int rev):to(to),cap(cap
22          ),cost(cost),rev(rev){}
23  };
24  vector<edge> E[5005];
25
26  inline void add_edge(int x,int y,ll f,ll c)
27  {
28      E[x].emplace_back(y,f,c,E[y].size());
29      E[y].emplace_back(x,0,-c,E[x].size()-1);
30  }
31
32  int spfa()
33  {
34      for(int i=0;i<=n;i++) vis[i]=0,dis[i]=inf; //从编
35      号最小的点到最大的点
36      dis[s]=0;
37      queue<int> q;
38      q.push(s);
39      while(!q.empty())
40      {
41          int p=q.front();q.pop();

```

```

40     vis[p]=0;
41     for(auto e:E[p])
42         if(e.cap&&dis[p]+e.cost<dis[e.to])
43         {
44             dis[e.to]=dis[p]+e.cost;
45             if(!vis[e.to])
46                 vis[e.to]=1,q.push(e.to);
47         }
48     }
49     return dis[t]!=inf;
50 }
51
52 ll dfs(int now,ll flow)
53 {
54     if(now==t) return flow;
55     ll rest=flow,k;
56     vis[now]=1;
57     for(int i=cur[now];i<E[now].size();i++)
58     {
59         edge &e=E[now][i];
60         if(e.cap&&dis[now]+e.cost==dis[e.to]&&!vis[e.to])
61         {
62             cur[now]=i;
63             k=dfs(e.to,min(rest,e.cap));
64             e.cap-=k;
65             E[e.to][e.rev].cap+=k;
66             msmf+=k*e.cost;
67             rest-=k;
68         }
69     }
70     vis[now]=0;
71     return flow-rest;
72 }
73
74 ll dinic()
75 {
76     msmf=0;
77     ll ret=0,delta;
78     while(spfa())
79     {
80         for(int i=0;i<n;i++) cur[i]=vis[i]=0; //从编号
81         //最小的点到最大的点
82         while(delta=dfs(s,inf)) ret+=delta;
83     }
84     return ret;
85 }
86
87 int main()
88 {
89     scanf("%d%d%d%d",&n,&m,&s,&t);
90     for(int i=0;i<m;i++)
91     {
92         int x,y;ll a,b;
93         scanf("%d%d%lld%lld",&x,&y,&a,&b);
94         add_edge(x,y,a,b);
95     }
96     ll mxflow=dinic();
97     printf("%lld %lld",mxflow,msmf);
98     return 0;
99 }

```

3.7.4 上下界流

3.8 欧拉路

```

1  #include <bits/stdc++.h>
2  //luogu P2731
3  using namespace std;
4  const int N=505;
5
6  /*
7   euler_path一定要找到正确的起点
8  */
9
10 int n;
11 int mp[N][N];
12 stack<int> st;
13 int deg[N];
14
15 void dfs(int x)
16 {
17     for(int i=1;i<=500;++i)
18     {
19         if(mp[x][i])
20         {
21             mp[x][i]--;
22             mp[i][x]--;
23             dfs(i);
24         }
25     }
26     st.push(x);
27 }
28
29 int main()
30 {
31     scanf("%d",&n);
32     for(int i=1,x,y;i<=n;++i)
33     {
34         scanf("%d%d",&x,&y);
35         mp[x][y]++;
36         mp[y][x]++;
37         deg[x]++;
38         deg[y]++;
39     }
40     int s=1;
41     for(int i=1;i<=500;++i)
42     {
43         if(deg[i]%2==1)
44         {
45             s=i;
46             break;
47         }
48     }
49     dfs(s);
50     while(!st.empty())
51     {
52         printf("%d\n",st.top());
53         st.pop();
54     }
55     return 0;
56 }

```



```

16 void pushdown(int rt)
17 {
18     if(!tag[rt]) return;
19     ll y=m1[rt];
20     if(y<m1[rt<<1])
21     {
22         tag[rt<<1]=1;
23         seg[rt<<1]--=(m1[rt<<1]-y)*cnt[rt<<1];
24         m1[rt<<1]=y;
25     }
26     if(y<m1[rt<<1|1])
27     {
28         tag[rt<<1|1]=1;
29         seg[rt<<1|1]--=(m1[rt<<1|1]-y)*cnt[rt<<1|1];
30         m1[rt<<1|1]=y;
31     }
32     tag[rt]=0;
33 }
34
35 void pushup(int rt)
36 {
37     seg[rt]=seg[rt<<1]+seg[rt<<1|1];
38     if(m1[rt<<1]==m1[rt<<1|1])
39     {
40         m1[rt]=m1[rt<<1];
41         cnt[rt]=cnt[rt<<1]+cnt[rt<<1|1];
42         m2[rt]=max(m2[rt<<1],m2[rt<<1|1]);
43     }
44     else if(m1[rt<<1]>m1[rt<<1|1])
45     {
46         m1[rt]=m1[rt<<1];
47         cnt[rt]=cnt[rt<<1];
48         m2[rt]=max(m2[rt<<1],m1[rt<<1|1]);
49     }
50     else
51     {
52         m1[rt]=m1[rt<<1|1];
53         cnt[rt]=cnt[rt<<1|1];
54         m2[rt]=max(m2[rt<<1|1],m1[rt<<1]);
55     }
56 }
57
58 void build(int rt,int l,int r)
59 {
60     tag[rt]=0;
61     if(l==r)
62     {
63         seg[rt]=m1[rt]=a[l];
64         cnt[rt]=1;
65         m2[rt]=INF;
66         return;
67     }
68     int m=l+r>>1;
69     if(l<=m) build(rt<<1,l,m);
70     if(m<r) build(rt<<1|1,m+1,r);
71     pushup(rt);
72 }
73
74 void modify(int rt,int l,int r,int L,int R,ll y)
75 {
76     if(y>=m1[rt]) return;
77     if(L<=l&&r<=R&&y>m2[rt])
78     {
79         tag[rt]=1;
80         seg[rt]--=(m1[rt]-y)*cnt[rt];

```

```

81         m1[rt]=y;
82         return;
83     }
84     pushdown(rt);
85     int m=l+r>>1;
86     if(L<=m) modify(rt<<1,l,m,L,R,y);
87     if(m<R) modify(rt<<1|1,m+1,r,L,R,y);
88     pushup(rt);
89 }
90
91 ll query(int rt,int l,int r,int L,int R)
92 {
93     if(L<=l&&r<=R) return seg[rt];
94     int m=l+r>>1;
95     pushdown(rt);
96     ll ret=0;
97     if(L<=m) ret+=query(rt<<1,l,m,L,R);
98     if(m<R) ret+=query(rt<<1|1,m+1,r,L,R);
99     pushup(rt);
100     return ret;
101 }

```

4.2.4 线段树维护扫描线

```

1  #include <bits/stdc++.h>
2  #define x1 xa
3  #define x2 xb
4  #define y1 ya
5  #define y1 yb
6  using namespace std;
7  typedef long long ll;
8  const int N=500005;
9
10 //luogu P5490
11
12 /*
13  上面的define是为了防止某些时候的CE
14  时刻记得节点[3,3]表示的是3号Y坐标与4号Y坐标构成的线段长度
15  */
16
17 int n;
18 ll x1,y1,x2,y2;
19 vector<ll> Y;
20
21 struct line
22 {
23     ll x,y1,y2;
24     int mark;
25 };
26 vector<line> v;
27 bool cmp(line a,line b){return a.x<b.x;}
28
29 struct seg
30 {
31     int l,r,cov;
32     ll len;
33 }tr[N<<2];
34
35 void build(int p,int l,int r)
36 {
37     tr[p].l=l,tr[p].r=r;
38     tr[p].len=tr[p].cov=0;
39     if(l==r) return;
40     int m=l+r>>1;

```



```

41     build(p<<1,l,m);
42     build(p<<1|1,m+1,r);
43 }
44
45 void up(int p)
46 {
47     int l=tr[p].l,r=tr[p].r;
48     if(tr[p].cov>0)
49         tr[p].len=Y[r]-Y[l-1];//
50     else
51         tr[p].len=tr[p<<1].len+tr[p<<1|1].len;
52 }
53
54 void modi(int L,int R,int V,int p=1)
55 {
56     int l=tr[p].l,r=tr[p].r;
57     if(L<=l&&r<=R)
58     {
59         tr[p].cov+=V;
60         up(p);
61         return;
62     }
63     int m=l+r>>1;
64     if(L<=m) modi(L,R,V,p<<1);
65     if(R>m) modi(L,R,V,p<<1|1);
66     up(p);
67 }
68
69 int main()
70 {
71     scanf("%d",&n);
72     for(int i=1;i<=n;++i)
73     {
74         scanf("%lld%lld%lld%lld",&x1,&y1,&x2,&y2);
75         Y.push_back(y1);
76         Y.push_back(y2);
77         v.push_back((line){x1,y1,y2,1});
78         v.push_back((line){x2,y1,y2,-1});
79     }
80     sort(v.begin(),v.end(),cmp);
81     sort(Y.begin(),Y.end());
82     for(auto &i:v)
83     {
84         i.y1=lower_bound(Y.begin(),Y.end(),i.y1)-Y.
            begin()+1;
85         i.y2=lower_bound(Y.begin(),Y.end(),i.y2)-Y.
            begin()+1;
86     }
87     build(1,1,Y.size());
88     ll ans=0;
89     for(int i=0;i<=v.size()-2;++i)
90     {
91         modi(v[i].y1,v[i].y2-1,v[i].mark);//
92         ans+=tr[1].len*(v[i+1].x-v[i].x);
93     }
94     printf("%lld",ans);
95     return 0;
96 }

```

4.2.5 区间图

```

1 #include <bits/stdc++.h>
2 //CF 786B
3

```

```

4 /*
5     建立两棵线段树分别表示出区间和入区间
6     第1到n标号的节点表示原来的n个点
7 */
8
9 using namespace std;
10 const int N=1e6+5;//original N=1e5
11 typedef long long ll;
12 const ll inf=0x3f3f3f3f3f3f3f3f;
13 typedef pair<ll,int> pli;
14
15 int n,q,s;
16 ll ans[N];
17 vector<int> e[N];
18 vector<ll> val[N];
19 bool vis[N];
20
21 void add(int x,int y,int z)
22 {
23     e[x].push_back(y);
24     val[x].push_back(z);
25 }
26
27 int cnt;
28 struct seg
29 {
30     int id[N];
31     void up(int p,bool flg)
32     {
33         int u=id[p];
34         int v=id[p<<1];
35         if(flg) swap(u,v);
36         add(u,v,0ll);
37         u=id[p];
38         v=id[p<<1|1];
39         if(flg) swap(u,v);
40         add(u,v,0ll);
41     }
42     void build(int flg,int p=1,int l=1,int r=n)
43     {
44         id[p]=++cnt;
45         if(l==r)
46         {
47             int u=id[p];
48             int v=l;
49             if(flg) swap(u,v);
50             add(u,v,0ll);
51             return;
52         }
53         int m=l+r>>1;
54         build(flg,p<<1,l,m);
55         build(flg,p<<1|1,m+1,r);
56         up(p,flg);
57     }
58     void segadd(int u,ll w,int L,int R,bool flg,int p
        =1,int l=1,int r=n)
59     {
60         if(L<=l&&r<=R)
61         {
62             int v=id[p];
63             if(flg) swap(u,v);
64             add(u,v,w);
65             return;
66         }
67         int m=l+r>>1;

```

```

68         if(L<=m) segadd(u,w,L,R,flg,p<<1,l,m);
69         if(R>m) segadd(u,w,L,R,flg,p<<1|m+1,r);
70     }
71 }in,out;
72
73 void dij()
74 {
75     priority_queue<pli,vector<pli>,greater<pli>> q;
76     q.emplace(0ll,s);
77     memset(ans,0x3f3f,sizeof(ans));
78     ans[s]=0;
79     while(!q.empty())
80     {
81         int t=q.top().second;q.pop();
82         if(vis[t]) continue;
83         vis[t]=1;
84         for(int i=0;i<e[t].size();++i)
85         {
86             int y=e[t][i];
87             ll v=val[t][i];
88             if(ans[t]+v<ans[y])
89             {
90                 ans[y]=ans[t]+v;
91                 q.emplace(ans[y],y);
92             }
93         }
94     }
95     for(int i=1;i<=n;++i)
96         printf("%lld ",(ans[i]==inf?-1ll:ans[i]));
97 }
98
99 int main()
100 {
101     scanf("%d%d%d",&n,&q,&s);
102     cnt=n;
103     //the first n points won't change
104     in.build(0);
105     out.build(1);
106     //in tree no flip,out tree must flip
107     for(int i=1;i<=q;++i)
108     {
109         int t,v,u,l,r;
110         ll w;
111         scanf("%d",&t);
112         if(t==1)
113         {
114             scanf("%d%d%lld",&v,&u,&w);
115             l=r=u;
116             t=2;
117         }
118         else scanf("%d%d%d%lld",&v,&l,&r,&w);
119         if(t==2) in.segadd(v,w,l,r,0);
120         else if(t==3) out.segadd(v,w,l,r,1);
121     }
122     dij();
123     return 0;
124 }

```

4.3 RMQ

4.3.1 一维

```

1 //dp[i][j]表示从a[i]开始,包括a[i]在内的2的j次方个数字中
  的最值
2 for(int i=1;i<=n;++i)

```

```

3     dp[i][0]=a[i];
4     for(int j=1;j<=30;++j){
5         for(int i=1;i+(1LL<<(j-1))<=n;++i){
6             dp[i][j]=max(dp[i][j-1],dp[i+(1LL<<(j-1))][j-1]); //min
7         }
8     }
9
10    int ask(int l,int r){
11        int k=(int)log2(r-l+1);
12        return max(dp[l][k],dp[r-(1LL<<k)+1][k]); //min
13    }

```

4.3.2 二维

4.4 树链剖分

4.4.1 点剖分

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 typedef long long ll;
4
5 /*
6     轻重链剖分 单次复杂度 O(log^2(n))
7     a[i] 表示dfs标号为i的点的值,而非点i的值
8     1 x y z 表示将树从x到y结点最短路径上所有节点值都加上z
9     2 x y 表示求树从x到y结点最短路径上所有节点值之和
10    3 x z 表示将以x为根节点的子树内所有节点值都加上z
11    4 x 表示求以x为根节点的子树内所有节点值之和
12 */
13
14 const int MAXN=100005;
15 ll mod,lazy[MAXN<<2],seg[MAXN<<2],a[MAXN],tmp[MAXN];
16 int n,q,r,cnt,tot,dep[MAXN],top[MAXN],id[MAXN],son[
17     MAXN],num[MAXN],fa[MAXN];
18 vector<int> e[MAXN];
19
20 void dfs1(int now,int f)
21 {
22     dep[now]=dep[f]+1;
23     fa[now]=f;
24     num[now]=1;
25     son[now]=0;
26     for(auto to:e[now])
27     {
28         if(to==f) continue;
29         dfs1(to,now);
30         num[now]+=num[to];
31         if(num[to]>num[son[now]]) son[now]=to;
32     }
33 }
34
35 void dfs2(int now,int f)
36 {
37     id[now]=++cnt;
38     top[now]=f;
39     if(son[now]) dfs2(son[now],f);
40     for(auto to:e[now])
41     {
42         if(to!=fa[now]&&to!=son[now])
43             dfs2(to,to);
44     }
45 }
46
47 inline void pushdown(int rt,ll lnum,ll rnum)
48 {

```

```

46     if(!lazy[rt]) return;
47     seg[rt<<1]=(seg[rt<<1]+lazy[rt]*lnum%mod)%mod;
48     seg[rt<<1|1]=(seg[rt<<1|1]+lazy[rt]*rnum%mod)%mod;
49     lazy[rt<<1]=(lazy[rt<<1]+lazy[rt])%mod;
50     lazy[rt<<1|1]=(lazy[rt<<1|1]+lazy[rt])%mod;
51     lazy[rt]=0;
52 }
53
54 inline void pushup(int rt)
55 {
56     seg[rt]=(seg[rt<<1]+seg[rt<<1|1])%mod;
57 }
58
59 void build(int rt,int l,int r)
60 {
61     lazy[rt]=0;
62     if(l==r)
63     {
64         seg[rt]=a[l]%mod;
65         return;
66     }
67     int m=l+r>>1;
68     if(l<=m) build(rt<<1,l,m);
69     if(m<r) build(rt<<1|1,m+1,r);
70     pushup(rt);
71 }
72
73 void modify(int rt,int l,int r,int L,int R,ll x)
74 {
75     if(L<=l&&r<=R)
76     {
77         lazy[rt]=(lazy[rt]+x)%mod;
78         seg[rt]=(seg[rt]+x*(r-l+1)%mod)%mod;
79         return;
80     }
81     int m=l+r>>1;
82     pushdown(rt,m-l+1,r-m);
83     if(L<=m) modify(rt<<1,l,m,L,R,x);
84     if(m<R) modify(rt<<1|1,m+1,r,L,R,x);
85     pushup(rt);
86 }
87
88 ll query(int rt,int l,int r,int L,int R)
89 {
90     if(L<=l&&r<=R) return seg[rt];
91     int m=l+r>>1;
92     ll ret=0;
93     pushdown(rt,m-l+1,r-m);
94     if(L<=m) ret=(ret+query(rt<<1,l,m,L,R))%mod;
95     if(m<R) ret=(ret+query(rt<<1|1,m+1,r,L,R))%mod;
96     pushup(rt);
97     return ret;
98 }
99
100 int main()
101 {
102     scanf("%d%d%d%lld",&n,&q,&r,&mod);
103     for(int i=1;i<=n;i++) scanf("%lld",&tmp[i]);
104     for(int i=1,x,y;i<=n;i++)
105     {
106         scanf("%d%d",&x,&y);
107         e[x].push_back(y),e[y].push_back(x);
108     }
109     num[0]=0,dep[r]=0;
110     dfs1(r,r);

```

```

111     dfs2(r,r);
112     for(int i=1;i<=n;i++) a[id[i]]=tmp[i];
113     build(1,1,n);
114
115     while(q--)
116     {
117         int op,x,y,ll z;
118         scanf("%d%d",&op,&x);
119         if(op==4)
120         {
121             printf("%lld\n",query(1,1,n,id[x],id[x]+num
122                 [x]-1));
123             continue;
124         }
125         if(op==1)
126         {
127             scanf("%d%lld",&y,&z);z%=mod;
128             while(top[x]!=top[y])
129             {
130                 if(dep[top[x]]<dep[top[y]]) swap(x,y);
131                 modify(1,1,n,id[top[x]],id[x],z);
132                 x=fa[top[x]];
133             }
134             if(dep[x]>dep[y]) swap(x,y);
135             modify(1,1,n,id[x],id[y],z);
136         }
137         else if(op==2)
138         {
139             scanf("%d",&y);
140             ll ans=0;
141             while(top[x]!=top[y])
142             {
143                 if(dep[top[x]]<dep[top[y]]) swap(x,y);
144                 ans=(ans+query(1,1,n,id[top[x]],id[x]))%
145                     mod;
146                 x=fa[top[x]];
147             }
148             if(dep[x]>dep[y]) swap(x,y);
149             ans=(ans+query(1,1,n,id[x],id[y]))%mod;
150             printf("%lld\n",ans);
151         }
152         else
153         {
154             scanf("%lld",&z);z%=mod;
155             modify(1,1,n,id[x],id[x]+num[x]-1,z);
156         }
157     }
    return 0;
}

```

4.4.2 边剖分

4.5 平衡树

4.5.1 Treap

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 const int MAXN=1e5+5;
5 const int inf=0x7fffffff;
6 int n,op,x;
7
8 /*
9     树内初始化时有无穷大和无穷小两个结点

```

```

10  _delete(root,x) 删除一个x
11  _insert(root,x) 插入一个x
12  getRank(root,x) 返回x的排名+1(包含了无穷小)
13  getVal(root,x+1) 返回排名为x的数
14  getPrev(x) x的前驱
15  getNext(x) x的后继
16  */
17
18 namespace Treap
19 {
20     int tot,root;
21     struct node
22     {
23         int cnt,val,dat,siz,lc,rc;
24     }bst[MAXN];
25
26     inline void pushup(int rt)
27     {
28         bst[rt].siz=bst[rt].cnt;
29         if(bst[rt].lc) bst[rt].siz+=bst[bst[rt].lc].siz;
30         if(bst[rt].rc) bst[rt].siz+=bst[bst[rt].rc].siz;
31     }
32
33     inline void zig(int &rt)
34     {
35         int p=bst[rt].lc;
36         bst[rt].lc=bst[p].rc;
37         bst[p].rc=rt;
38         rt=p;
39         pushup(bst[rt].rc);pushup(rt);
40     }
41
42     inline void zag(int &rt)
43     {
44         int p=bst[rt].rc;
45         bst[rt].rc=bst[p].lc;
46         bst[p].lc=rt;
47         rt=p;
48         pushup(bst[rt].lc);pushup(rt);
49     }
50
51     int new_node(int val)
52     {
53         bst[++tot].val=val;
54         bst[tot].dat=rand();
55         bst[tot].siz=bst[tot].cnt=1;
56         bst[tot].lc=bst[tot].rc=0;
57         return tot;
58     }
59
60     void build()
61     {
62         new_node(-inf);new_node(inf);
63         root=1,bst[1].rc=2;
64         pushup(1);
65     }
66
67     void _insert(int &rt,int val)
68     {
69         if(rt==0)
70         {
71             rt=new_node(val);
72             return;

```

```

73     }
74     if(bst[rt].val==val)
75     {
76         bst[rt].cnt++;
77         pushup(rt);
78         return;
79     }
80     if(val<bst[rt].val)
81     {
82         _insert(bst[rt].lc,val);
83         if(bst[rt].dat<bst[bst[rt].lc].dat) zig(rt)
84         ;
85     }
86     else
87     {
88         _insert(bst[rt].rc,val);
89         if(bst[rt].dat<bst[bst[rt].rc].dat) zag(rt)
90         ;
91     }
92     pushup(rt);
93 }
94
95 void _delete(int &rt,int val)
96 {
97     if(rt==0) return;
98     if(bst[rt].val==val)
99     {
100         if(bst[rt].cnt>1)
101         {
102             bst[rt].cnt--;
103             pushup(rt);
104             return;
105         }
106         if(bst[rt].rc||bst[rt].lc)
107         {
108             if(bst[rt].rc==0||bst[bst[rt].rc].dat<
109                 bst[bst[rt].lc].dat)
110                 zig(rt),_delete(bst[rt].rc,val);
111             else
112                 zag(rt),_delete(bst[rt].lc,val);
113             pushup(rt);
114         }
115         else rt=0;
116         return;
117     }
118     if(val<bst[rt].val) _delete(bst[rt].lc,val);
119     else _delete(bst[rt].rc,val);
120     pushup(rt);
121 }
122
123 int getPrev(int val)
124 {
125     int ret=1,rt=root;
126     while(rt)
127     {
128         if(bst[rt].val==val)
129         {
130             if(bst[rt].lc)
131             {
132                 rt=bst[rt].lc;
133                 while(bst[rt].rc) rt=bst[rt].rc;
134                 ret=rt;
135             }
136             break;

```

```

135     }
136     if(bst[rt].val<val&&bst[rt].val>bst[ret].
        val) ret=rt;
137     if(val<bst[rt].val) rt=bst[rt].lc;
138     else rt=bst[rt].rc;
139 }
140 return bst[ret].val;
141 }
142
143 int getNext(int val)
144 {
145     int ret=2,rt=root;
146     while(rt)
147     {
148         if(bst[rt].val==val)
149         {
150             if(bst[rt].rc)
151             {
152                 rt=bst[rt].rc;
153                 while(bst[rt].lc) rt=bst[rt].lc;
154                 ret=rt;
155             }
156             break;
157         }
158         if(bst[rt].val>val&&bst[rt].val<bst[ret].
            val) ret=rt;
159         if(val<bst[rt].val) rt=bst[rt].lc;
160         else rt=bst[rt].rc;
161     }
162     return bst[ret].val;
163 }
164
165 int getRank(int rt,int val)
166 {
167     if(rt==0) return 0;
168     if(val==bst[rt].val) return bst[rt].lc.
        siz+1;
169     if(val<bst[rt].val) return getRank(bst[rt].lc,
        val);
170     else return bst[rt].lc.siz+bst[rt].cnt+
        getRank(bst[rt].rc,val);
171 }
172
173 int getVal(int rt,int k)
174 {
175     if(rt==0) return inf;
176     if(bst[rt].lc.siz>=k) return getVal(bst[rt].lc,k);
177     if(bst[rt].lc.siz+bst[rt].cnt>=k) return
        bst[rt].val;
178     return getVal(bst[rt].rc,k-bst[rt].lc.siz
        -bst[rt].cnt);
179 }
180 }
181
182 int main()
183 {
184     using namespace Treap;
185     srand(time(0));
186     build();
187     scanf("%d",&n);
188     while(n--)
189     {
190         scanf("%d",&op,&x);
191         if(op==1) _insert(root,x);

```

```

192     else if(op==2) _delete(root,x);
193     else if(op==3) printf("%d\n",getRank(root,x)
        -1);
194     else if(op==4) printf("%d\n",getVal(root,x+1))
        ;
195     else if(op==5) printf("%d\n",getPrev(x));
196     else if(op==6) printf("%d\n",getNext(x));
197 }
198 return 0;
199 }

```

4.5.2 Splay

4.5.3 红黑树-pbds

```

1 #include <ext/pb_ds/tree_policy.hpp>
2 #include <ext/pb_ds/assoc_container.hpp>
3 using namespace __gnu_pbds;
4 tree<pii,null_type,less<pii>,rb_tree_tag,
    tree_order_statistics_node_update> t;
5
6 t.insert({x,i+1}); //插入x, 用独特的正整数i+1标注 (因为
    erase太辣鸡)
7 t.erase(t.lower_bound({x,0})); //删除x (删除单个元素)
8 t.order_of_key({x,0})+1; //x的排名 (小于x的元素个数+1)
9 t.find_by_order(x-1)->first; //排名为x的元素 (第x小的
    数)
10 prev(t.lower_bound({x,0}))->first; //x的前驱 (小于x且
    最大)
11 t.lower_bound({x+1,0})->first; //x的后继 (大于x且最小)

```

4.6 动态树

4.7 主席树

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /*
5     离散化+区间k小
6 */
7
8 const int MAXN=200005;
9 int n,m,a[MAXN],tmp[MAXN],org[MAXN],root[MAXN],tot=0;
10 struct tree
11 {
12     int cnt,lc,rc;
13 }seg[30*MAXN];
14
15 int build(int l,int r)
16 {
17     int p=tot++;
18     if(l==r)
19     {
20         seg[p].cnt=0;
21         return p;
22     }
23     int m=l+r>>1;
24     seg[p].lc=build(l,m);
25     seg[p].rc=build(m+1,r);
26     seg[p].cnt=seg[seg[p].lc].cnt+seg[seg[p].rc].cnt;
27     return p;
28 }
29

```

```

30 int modify(int rt,int l,int r,int x)
31 {
32     int p=tot++;
33     seg[p]=seg[rt];
34     if(l==r)
35     {
36         seg[p].cnt++;
37         return p;
38     }
39     int m=l+r>>1;
40     if(x<=m) seg[p].lc=modify(seg[rt].lc,l,m,x);
41     else seg[p].rc=modify(seg[rt].rc,m+1,r,x);
42     seg[p].cnt=seg[seg[p].lc].cnt+seg[seg[p].rc].cnt;
43     return p;
44 }
45
46 int query(int p,int q,int l,int r,int k)
47 {
48     if(l==r) return l;
49     int m=l+r>>1;
50     int lc=seg[seg[q].lc].cnt-seg[seg[p].lc].cnt;
51     if(lc>=k) return query(seg[p].lc,seg[q].lc,l,m,k);
52     else return query(seg[p].rc,seg[q].rc,m+1,r,k-lc);
53 }
54
55 int main()
56 {
57     scanf("%d%d",&n,&m);
58     for(int i=1;i<=n;i++)
59         scanf("%d",&a[i]),tmp[i]=a[i];
60     sort(tmp+1,tmp+n+1);
61     root[0]=build(1,n);
62     for(int i=1;i<=n;i++)
63     {
64         int k=lower_bound(tmp+1,tmp+n+1,a[i])-tmp;
65         org[k]=a[i];
66         a[i]=k;
67         root[i]=modify(root[i-1],1,n,a[i]);
68     }
69     while(m--)
70     {
71         int x,y,k;
72         scanf("%d%d%d",&x,&y,&k);
73         printf("%d\n",org[query(root[x-1],root[y],1,n,k)]);
74     }
75     return 0;
76 }

```

4.8 树套树

4.8.1 线段树套 Treap

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /*
5     空间 O(nlogn)
6     单点修改,区间rank,前驱后继(不存在则为±2147483647) 单
7     次 O(log^2(n))
8     区间排名为k的值 单次 O(log^3(n))
9 */

```

```

10 const int inf=2147483647;
11 const int MAXN=50005;
12 int root[MAXN<<2],n,m,a[MAXN];
13 struct Treap
14 {
15     int tot;
16     struct node
17     {
18         int lc,rc,dat,val,cnt,siz;
19     }bst[MAXN*4*20];
20
21     int newnode(int v)
22     {
23         bst[++tot].val=v;
24         bst[tot].dat=rand();
25         bst[tot].siz=bst[tot].cnt=1;
26         bst[tot].lc=bst[tot].rc=0;
27         return tot;
28     }
29
30     void zig(int &rt)
31     {
32         int p=bst[rt].lc;
33         bst[rt].lc=bst[p].rc;
34         bst[p].rc=rt;
35         rt=p;
36         pushup(bst[rt].rc);
37         pushup(rt);
38     }
39
40     void zag(int &rt)
41     {
42         int p=bst[rt].rc;
43         bst[rt].rc=bst[p].lc;
44         bst[p].lc=rt;
45         rt=p;
46         pushup(bst[rt].lc);
47         pushup(rt);
48     }
49
50     void pushup(int rt)
51     {
52         bst[rt].siz=bst[rt].cnt;
53         if(bst[rt].lc) bst[rt].siz+=bst[bst[rt].lc].siz;
54         if(bst[rt].rc) bst[rt].siz+=bst[bst[rt].rc].siz;
55     }
56
57     int build()
58     {
59         int rt=newnode(-inf);
60         bst[rt].rc=newnode(inf);
61         pushup(rt);
62         return rt;
63     }
64
65     void _delete(int &rt,int x)
66     {
67         if(bst[rt].val==x)
68         {
69             if(bst[rt].cnt>1)
70             {
71                 bst[rt].cnt--;
72                 pushup(rt);

```

```

73         return;
74     }
75     if(bst[rt].lc||bst[rt].rc)
76     {
77         if(bst[rt].rc==0||bst[bst[rt].rc].dat<
            bst[bst[rt].lc].dat)
78             zig(rt),_delete(bst[rt].rc,x);
79         else
80             zag(rt),_delete(bst[rt].lc,x);
81         pushup(rt);
82     }
83     else rt=0;
84     return;
85 }
86 if(x<bst[rt].val) _delete(bst[rt].lc,x);
87 else _delete(bst[rt].rc,x);
88 pushup(rt);
89 }
90
91 void _insert(int &rt,int x)
92 {
93     if(rt==0)
94     {
95         rt=newnode(x);
96         return;
97     }
98     if(bst[rt].val==x) bst[rt].cnt++;
99     else if(x<bst[rt].val)
100     {
101         _insert(bst[rt].lc,x);
102         if(bst[bst[rt].lc].dat>bst[rt].dat) zig(rt)
103             ;
104     }
105     else
106     {
107         _insert(bst[rt].rc,x);
108         if(bst[bst[rt].rc].dat>bst[rt].dat) zag(rt)
109             ;
110     }
111     pushup(rt);
112 }
113
114 int get_rank(int rt,int x)
115 {
116     if(!rt) return 1;
117     if(bst[rt].val==x) return bst[bst[rt].lc].siz
118         +1;
119     if(x<bst[rt].val) return get_rank(bst[rt].lc,x
120         );
121     else return get_rank(bst[rt].rc,x)+bst[bst[rt]
122         ].lc].siz+bst[rt].cnt;
123 }
124
125 int get_num(int rt,int x)
126 {
127     if(!rt) return 0;
128     if(bst[rt].val==x) return bst[bst[rt].lc].siz+
129         bst[rt].cnt;
130     if(x<bst[rt].val) return get_num(bst[rt].lc,x)
131         ;
132     else return get_num(bst[rt].rc,x)+bst[bst[rt].
133         lc].siz+bst[rt].cnt;
134 }
135
136 int get_prev(int rt,int x)
137 {
138     if(!rt) return -1;
139     if(bst[rt].val==x)
140     {
141         if(bst[rt].lc) return bst[rt].lc;
142         else return -1;
143     }
144     if(x<bst[rt].val) return get_prev(bst[rt].lc,x);
145     else return get_prev(bst[rt].rc,x);
146 }
147
148 int get_next(int rt,int x)
149 {
150     if(!rt) return -1;
151     if(bst[rt].val==x)
152     {
153         if(bst[rt].rc) return bst[rt].rc;
154         else return -1;
155     }
156     if(x>bst[rt].val) return get_next(bst[rt].rc,x);
157     else return get_next(bst[rt].lc,x);
158 }
159
160 void treap_init()
161 {
162     root=0;
163     n=0;
164 }
165
166 void treap_build(int l,int r)
167 {
168     if(l>=r) return;
169     int m=(l+r)>>1;
170     treap_build(l,m);
171     treap_build(m+1,r);
172     root=treap_build(l,r);
173 }
174
175 void treap_insert(int x)
176 {
177     _insert(root,x);
178     pushup(root);
179 }
180
181 void treap_delete(int x)
182 {
183     _delete(root,x);
184     pushup(root);
185 }
186
187 int treap_get_rank(int x)
188 {
189     return get_rank(root,x);
190 }
191
192 int treap_get_num(int x)
193 {
194     return get_num(root,x);
195 }
196
197 int treap_get_prev(int x)
198 {
199     return get_prev(root,x);
200 }
201
202 int treap_get_next(int x)
203 {
204     return get_next(root,x);
205 }
206
207 void treap_modify(int l,int r,int x,int v,int y)
208 {
209     if(l==r) return;
210     int m=(l+r)>>1;
211     treap_modify(l,m,x,v,y);
212     treap_modify(m+1,r,x,v,y);
213     root=treap_modify(l,r,x,v,y);
214 }
215
216 void treap_build(int l,int r)
217 {
218     if(l>=r) return;
219     int m=(l+r)>>1;
220     treap_build(l,m);
221     treap_build(m+1,r);
222     root=treap_build(l,r);
223 }
224
225 void treap_init()
226 {
227     root=0;
228     n=0;
229 }
230
231 void treap_build(int l,int r)
232 {
233     if(l>=r) return;
234     int m=(l+r)>>1;
235     treap_build(l,m);
236     treap_build(m+1,r);
237     root=treap_build(l,r);
238 }
239
240 void treap_insert(int x)
241 {
242     _insert(root,x);
243     pushup(root);
244 }
245
246 void treap_delete(int x)
247 {
248     _delete(root,x);
249     pushup(root);
250 }
251
252 int treap_get_rank(int x)
253 {
254     return get_rank(root,x);
255 }
256
257 int treap_get_num(int x)
258 {
259     return get_num(root,x);
260 }
261
262 int treap_get_prev(int x)
263 {
264     return get_prev(root,x);
265 }
266
267 int treap_get_next(int x)
268 {
269     return get_next(root,x);
270 }
271
272 void treap_modify(int l,int r,int x,int v,int y)
273 {
274     if(l==r) return;
275     int m=(l+r)>>1;
276     treap_modify(l,m,x,v,y);
277     treap_modify(m+1,r,x,v,y);
278     root=treap_modify(l,r,x,v,y);
279 }
280
281 void treap_build(int l,int r)
282 {
283     if(l>=r) return;
284     int m=(l+r)>>1;
285     treap_build(l,m);
286     treap_build(m+1,r);
287     root=treap_build(l,r);
288 }
289
290 void treap_init()
291 {
292     root=0;
293     n=0;
294 }
295
296 void treap_build(int l,int r)
297 {
298     if(l>=r) return;
299     int m=(l+r)>>1;
300     treap_build(l,m);
301     treap_build(m+1,r);
302     root=treap_build(l,r);
303 }
304
305 void treap_insert(int x)
306 {
307     _insert(root,x);
308     pushup(root);
309 }
310
311 void treap_delete(int x)
312 {
313     _delete(root,x);
314     pushup(root);
315 }
316
317 int treap_get_rank(int x)
318 {
319     return get_rank(root,x);
320 }
321
322 int treap_get_num(int x)
323 {
324     return get_num(root,x);
325 }
326
327 int treap_get_prev(int x)
328 {
329     return get_prev(root,x);
330 }
331
332 int treap_get_next(int x)
333 {
334     return get_next(root,x);
335 }
336
337 void treap_modify(int l,int r,int x,int v,int y)
338 {
339     if(l==r) return;
340     int m=(l+r)>>1;
341     treap_modify(l,m,x,v,y);
342     treap_modify(m+1,r,x,v,y);
343     root=treap_modify(l,r,x,v,y);
344 }
345
346 void treap_build(int l,int r)
347 {
348     if(l>=r) return;
349     int m=(l+r)>>1;
350     treap_build(l,m);
351     treap_build(m+1,r);
352     root=treap_build(l,r);
353 }
354
355 void treap_init()
356 {
357     root=0;
358     n=0;
359 }
360
361 void treap_build(int l,int r)
362 {
363     if(l>=r) return;
364     int m=(l+r)>>1;
365     treap_build(l,m);
366     treap_build(m+1,r);
367     root=treap_build(l,r);
368 }
369
370 void treap_insert(int x)
371 {
372     _insert(root,x);
373     pushup(root);
374 }
375
376 void treap_delete(int x)
377 {
378     _delete(root,x);
379     pushup(root);
380 }
381
382 int treap_get_rank(int x)
383 {
384     return get_rank(root,x);
385 }
386
387 int treap_get_num(int x)
388 {
389     return get_num(root,x);
390 }
391
392 int treap_get_prev(int x)
393 {
394     return get_prev(root,x);
395 }
396
397 int treap_get_next(int x)
398 {
399     return get_next(root,x);
400 }
401
402 void treap_modify(int l,int r,int x,int v,int y)
403 {
404     if(l==r) return;
405     int m=(l+r)>>1;
406     treap_modify(l,m,x,v,y);
407     treap_modify(m+1,r,x,v,y);
408     root=treap_modify(l,r,x,v,y);
409 }
410
411 void treap_build(int l,int r)
412 {
413     if(l>=r) return;
414     int m=(l+r)>>1;
415     treap_build(l,m);
416     treap_build(m+1,r);
417     root=treap_build(l,r);
418 }
419
420 void treap_init()
421 {
422     root=0;
423     n=0;
424 }
425
426 void treap_build(int l,int r)
427 {
428     if(l>=r) return;
429     int m=(l+r)>>1;
430     treap_build(l,m);
431     treap_build(m+1,r);
432     root=treap_build(l,r);
433 }
434
435 void treap_insert(int x)
436 {
437     _insert(root,x);
438     pushup(root);
439 }
440
441 void treap_delete(int x)
442 {
443     _delete(root,x);
444     pushup(root);
445 }
446
447 int treap_get_rank(int x)
448 {
449     return get_rank(root,x);
450 }
451
452 int treap_get_num(int x)
453 {
454     return get_num(root,x);
455 }
456
457 int treap_get_prev(int x)
458 {
459     return get_prev(root,x);
460 }
461
462 int treap_get_next(int x)
463 {
464     return get_next(root,x);
465 }
466
467 void treap_modify(int l,int r,int x,int v,int y)
468 {
469     if(l==r) return;
470     int m=(l+r)>>1;
471     treap_modify(l,m,x,v,y);
472     treap_modify(m+1,r,x,v,y);
473     root=treap_modify(l,r,x,v,y);
474 }
475
476 void treap_build(int l,int r)
477 {
478     if(l>=r) return;
479     int m=(l+r)>>1;
480     treap_build(l,m);
481     treap_build(m+1,r);
482     root=treap_build(l,r);
483 }
484
485 void treap_init()
486 {
487     root=0;
488     n=0;
489 }
490
491 void treap_build(int l,int r)
492 {
493     if(l>=r) return;
494     int m=(l+r)>>1;
495     treap_build(l,m);
496     treap_build(m+1,r);
497     root=treap_build(l,r);
498 }
499
500 void treap_insert(int x)
501 {
502     _insert(root,x);
503     pushup(root);
504 }
505
506 void treap_delete(int x)
507 {
508     _delete(root,x);
509     pushup(root);
510 }
511
512 int treap_get_rank(int x)
513 {
514     return get_rank(root,x);
515 }
516
517 int treap_get_num(int x)
518 {
519     return get_num(root,x);
520 }
521
522 int treap_get_prev(int x)
523 {
524     return get_prev(root,x);
525 }
526
527 int treap_get_next(int x)
528 {
529     return get_next(root,x);
530 }
531
532 void treap_modify(int l,int r,int x,int v,int y)
533 {
534     if(l==r) return;
535     int m=(l+r)>>1;
536     treap_modify(l,m,x,v,y);
537     treap_modify(m+1,r,x,v,y);
538     root=treap_modify(l,r,x,v,y);
539 }
540
541 void treap_build(int l,int r)
542 {
543     if(l>=r) return;
544     int m=(l+r)>>1;
545     treap_build(l,m);
546     treap_build(m+1,r);
547     root=treap_build(l,r);
548 }
549
550 void treap_init()
551 {
552     root=0;
553     n=0;
554 }
555
556 void treap_build(int l,int r)
557 {
558     if(l>=r) return;
559     int m=(l+r)>>1;
560     treap_build(l,m);
561     treap_build(m+1,r);
562     root=treap_build(l,r);
563 }
564
565 void treap_insert(int x)
566 {
567     _insert(root,x);
568     pushup(root);
569 }
570
571 void treap_delete(int x)
572 {
573     _delete(root,x);
574     pushup(root);
575 }
576
577 int treap_get_rank(int x)
578 {
579     return get_rank(root,x);
580 }
581
582 int treap_get_num(int x)
583 {
584     return get_num(root,x);
585 }
586
587 int treap_get_prev(int x)
588 {
589     return get_prev(root,x);
590 }
591
592 int treap_get_next(int x)
593 {
594     return get_next(root,x);
595 }
596
597 void treap_modify(int l,int r,int x,int v,int y)
598 {
599     if(l==r) return;
600     int m=(l+r)>>1;
601     treap_modify(l,m,x,v,y);
602     treap_modify(m+1,r,x,v,y);
603     root=treap_modify(l,r,x,v,y);
604 }
605
606 void treap_build(int l,int r)
607 {
608     if(l>=r) return;
609     int m=(l+r)>>1;
610     treap_build(l,m);
611     treap_build(m+1,r);
612     root=treap_build(l,r);
613 }
614
615 void treap_init()
616 {
617     root=0;
618     n=0;
619 }
620
621 void treap_build(int l,int r)
622 {
623     if(l>=r) return;
624     int m=(l+r)>>1;
625     treap_build(l,m);
626     treap_build(m+1,r);
627     root=treap_build(l,r);
628 }
629
630 void treap_insert(int x)
631 {
632     _insert(root,x);
633     pushup(root);
634 }
635
636 void treap_delete(int x)
637 {
638     _delete(root,x);
639     pushup(root);
640 }
641
642 int treap_get_rank(int x)
643 {
644     return get_rank(root,x);
645 }
646
647 int treap_get_num(int x)
648 {
649     return get_num(root,x);
650 }
651
652 int treap_get_prev(int x)
653 {
654     return get_prev(root,x);
655 }
656
657 int treap_get_next(int x)
658 {
659     return get_next(root,x);
660 }
661
662 void treap_modify(int l,int r,int x,int v,int y)
663 {
664     if(l==r) return;
665     int m=(l+r)>>1;
666     treap_modify(l,m,x,v,y);
667     treap_modify(m+1,r,x,v,y);
668     root=treap_modify(l,r,x,v,y);
669 }
670
671 void treap_build(int l,int r)
672 {
673     if(l>=r) return;
674     int m=(l+r)>>1;
675     treap_build(l,m);
676     treap_build(m+1,r);
677     root=treap_build(l,r);
678 }
679
680 void treap_init()
681 {
682     root=0;
683     n=0;
684 }
685
686 void treap_build(int l,int r)
687 {
688     if(l>=r) return;
689     int m=(l+r)>>1;
690     treap_build(l,m);
691     treap_build(m+1,r);
692     root=treap_build(l,r);
693 }
694
695 void treap_insert(int x)
696 {
697     _insert(root,x);
698     pushup(root);
699 }
700
701 void treap_delete(int x)
702 {
703     _delete(root,x);
704     pushup(root);
705 }
706
707 int treap_get_rank(int x)
708 {
709     return get_rank(root,x);
710 }
711
712 int treap_get_num(int x)
713 {
714     return get_num(root,x);
715 }
716
717 int treap_get_prev(int x)
718 {
719     return get_prev(root,x);
720 }
721
722 int treap_get_next(int x)
723 {
724     return get_next(root,x);
725 }
726
727 void treap_modify(int l,int r,int x,int v,int y)
728 {
729     if(l==r) return;
730     int m=(l+r)>>1;
731     treap_modify(l,m,x,v,y);
732     treap_modify(m+1,r,x,v,y);
733     root=treap_modify(l,r,x,v,y);
734 }
735
736 void treap_build(int l,int r)
737 {
738     if(l>=r) return;
739     int m=(l+r)>>1;
740     treap_build(l,m);
741     treap_build(m+1,r);
742     root=treap_build(l,r);
743 }
744
745 void treap_init()
746 {
747     root=0;
748     n=0;
749 }
750
751 void treap_build(int l,int r)
752 {
753     if(l>=r) return;
754     int m=(l+r)>>1;
755     treap_build(l,m);
756     treap_build(m+1,r);
757     root=treap_build(l,r);
758 }
759
760 void treap_insert(int x)
761 {
762     _insert(root,x);
763     pushup(root);
764 }
765
766 void treap_delete(int x)
767 {
768     _delete(root,x);
769     pushup(root);
770 }
771
772 int treap_get_rank(int x)
773 {
774     return get_rank(root,x);
775 }
776
777 int treap_get_num(int x)
778 {
779     return get_num(root,x);
780 }
781
782 int treap_get_prev(int x)
783 {
784     return get_prev(root,x);
785 }
786
787 int treap_get_next(int x)
788 {
789     return get_next(root,x);
790 }
791
792 void treap_modify(int l,int r,int x,int v,int y)
793 {
794     if(l==r) return;
795     int m=(l+r)>>1;
796     treap_modify(l,m,x,v,y);
797     treap_modify(m+1,r,x,v,y);
798     root=treap_modify(l,r,x,v,y);
799 }
800
801 void treap_build(int l,int r)
802 {
803     if(l>=r) return;
804     int m=(l+r)>>1;
805     treap_build(l,m);
806     treap_build(m+1,r);
807     root=treap_build(l,r);
808 }
809
810 void treap_init()
811 {
812     root=0;
813     n=0;
814 }
815
816 void treap_build(int l,int r)
817 {
818     if(l>=r) return;
819     int m=(l+r)>>1;
820     treap_build(l,m);
821     treap_build(m+1,r);
822     root=treap_build(l,r);
823 }
824
825 void treap_insert(int x)
826 {
827     _insert(root,x);
828     pushup(root);
829 }
830
831 void treap_delete(int x)
832 {
833     _delete(root,x);
834     pushup(root);
835 }
836
837 int treap_get_rank(int x)
838 {
839     return get_rank(root,x);
840 }
841
842 int treap_get_num(int x)
843 {
844     return get_num(root,x);
845 }
846
847 int treap_get_prev(int x)
848 {
849     return get_prev(root,x);
850 }
851
852 int treap_get_next(int x)
853 {
854     return get_next(root,x);
855 }
856
857 void treap_modify(int l,int r,int x,int v,int y)
858 {
859     if(l==r) return;
860     int m=(l+r)>>1;
861     treap_modify(l,m,x,v,y);
862     treap_modify(m+1,r,x,v,y);
863     root=treap_modify(l,r,x,v,y);
864 }
865
866 void treap_build(int l,int r)
867 {
868     if(l>=r) return;
869     int m=(l+r)>>1;
870     treap_build(l,m);
871     treap_build(m+1,r);
872     root=treap_build(l,r);
873 }
874
875 void treap_init()
876 {
877     root=0;
878     n=0;
879 }
880
881 void treap_build(int l,int r)
882 {
883     if(l>=r) return;
884     int m=(l+r)>>1;
885     treap_build(l,m);
886     treap_build(m+1,r);
887     root=treap_build(l,r);
888 }
889
890 void treap_insert(int x)
891 {
892     _insert(root,x);
893     pushup(root);
894 }
895
896 void treap_delete(int x)
897 {
898     _delete(root,x);
899     pushup(root);
900 }
901
902 int treap_get_rank(int x)
903 {
904     return get_rank(root,x);
905 }
906
907 int treap_get_num(int x)
908 {
909     return get_num(root,x);
910 }
911
912 int treap_get_prev(int x)
913 {
914     return get_prev(root,x);
915 }
916
917 int treap_get_next(int x)
918 {
919     return get_next(root,x);
920 }
921
922 void treap_modify(int l,int r,int x,int v,int y)
923 {
924     if(l==r) return;
925     int m=(l+r)>>1;
926     treap_modify(l,m,x,v,y);
927     treap_modify(m+1,r,x,v,y);
928     root=treap_modify(l,r,x,v,y);
929 }
930
931 void treap_build(int l,int r)
932 {
933     if(l>=r) return;
934     int m=(l+r)>>1;
935     treap_build(l,m);
936     treap_build(m+1,r);
937     root=treap_build(l,r);
938 }
939
940 void treap_init()
941 {
942     root=0;
943     n=0;
944 }
945
946 void treap_build(int l,int r)
947 {
948     if(l>=r) return;
949     int m=(l+r)>>1;
950     treap_build(l,m);
951     treap_build(m+1,r);
952     root=treap_build(l,r);
953 }
954
955 void treap_insert(int x)
956 {
957     _insert(root,x);
958     pushup(root);
959 }
960
961 void treap_delete(int x)
962 {
963     _delete(root,x);
964     pushup(root);
965 }
966
967 int treap_get_rank(int x)
968 {
969     return get_rank(root,x);
970 }
971
972 int treap_get_num(int x)
973 {
974     return get_num(root,x);
975 }
976
977 int treap_get_prev(int x)
978 {
979     return get_prev(root,x);
980 }
981
982 int treap_get_next(int x)
983 {
984     return get_next(root,x);
985 }
986
987 void treap_modify(int l,int r,int x,int v,int y)
988 {
989     if(l==r) return;
990     int m=(l+r)>>1;
991     treap_modify(l,m,x,v,y);
992     treap_modify(m+1,r,x,v,y);
993     root=treap_modify(l,r,x,v,y);
994 }
995
996 void treap_build(int l,int r)
997 {
998     if(l>=r) return;
999     int m=(l+r)>>1;
1000     treap_build(l,m);
1001     treap_build(m+1,r);
1002     root=treap_build(l,r);
1003 }
1004
1005 void treap_init()
1006 {
1007     root=0;
1008     n=0;
1009 }
1010
1011 void treap_build(int l,int r)
1012 {
1013     if(l>=r) return;
1014     int m=(l+r)>>1;
1015     treap_build(l,m);
1016     treap_build(m+1,r);
1017     root=treap_build(l,r);
1018 }
1019
1020 void treap_insert(int x)
1021 {
1022     _insert(root,x);
1023     pushup(root);
1024 }
1025
1026 void treap_delete(int x)
1027 {
1028     _delete(root,x);
1029     pushup(root);
1030 }
1031
1032 int treap_get_rank(int x)
1033 {
1034     return get_rank(root,x);
1035 }
1036
1037 int treap_get_num(int x)
1038 {
1039     return get_num(root,x);
1040 }
1041
1042 int treap_get_prev(int x)
1043 {
1044     return get_prev(root,x);
1045 }
1046
1047 int treap_get_next(int x)
1048 {
1049     return get_next(root,x);
1050 }
1051
1052 void treap_modify(int l,int r,int x,int v,int y)
1053 {
1054     if(l==r) return;
1055     int m=(l+r)>>1;
1056     treap_modify(l,m,x,v,y);
1057     treap_modify(m+1,r,x,v,y);
1058     root=treap_modify(l,r,x,v,y);
1059 }
1060
1061 void treap_build(int l,int r)
1062 {
1063     if(l>=r) return;
1064     int m=(l+r)>>1;
1065     treap_build(l,m);
1066     treap_build(m+1,r);
1067     root=treap_build(l,r);
1068 }
1069
1070 void treap_init()
1071 {
1072     root=0;
1073     n=0;
1074 }
1075
1076 void treap_build(int l,int r)
1077 {
1078     if(l>=r) return;
1079     int m=(l+r)>>1;
1080     treap_build(l,m);
1081     treap_build(m+1,r);
1082     root=treap_build(l,r);
1083 }
1084
1085 void treap_insert(int x)
1086 {
1087     _insert(root,x);
1088     pushup(root);
1089 }
1090
1091 void treap_delete(int x)
1092 {
1093     _delete(root,x);
1094     pushup(root);
1095 }
1096
1097 int treap_get_rank(int x)
1098 {
1099     return get_rank(root,x);
1100 }
1101
1102 int treap_get_num(int x)
1103 {
1104     return get_num(root,x);
1105 }
1106
1107 int treap_get_prev(int x)
1108 {
1109     return get_prev(root,x);
1110 }
1111
1112 int treap_get_next(int x)
1113 {
1114     return get_next(root,x);
1115 }
1116
1117 void treap_modify(int l,int r,int x,int v,int y)
1118 {
1119     if(l==r) return;
1120     int m=(l+r)>>1;
1121     treap_modify(l,m,x,v,y);
1122     treap_modify(m+1,r,x,v,y);
1123     root=treap_modify(l,r,x,v,y);
1124 }
1125
1126 void treap_build(int l,int r)
1127 {
1128     if(l>=r) return;
1129     int m=(l+r)>>1;
1130     treap_build(l,m);
1131     treap_build(m+1,r);
1132     root=treap_build(l,r);
1133 }
1134
1135 void treap_init()
1136 {
1137     root=0;
1138     n=0;
1139 }
1140
1141 void treap_build(int l,int r)
1142 {
1143     if(l>=r) return;
1144     int m=(l+r)>>1;
1145     treap_build(l,m);
1146     treap_build(m+1,r);
1147     root=treap_build(l,r);
1148 }
1149
1150 void treap_insert(int x)
1151 {
1152     _insert(root,x);
1153     pushup(root);
1154 }
1155
1156 void treap_delete(int x)
1157 {
1158     _delete(root,x);
1159     pushup(root);
1160 }
1161
1162 int treap_get_rank(int x)
1163 {
1164     return get_rank(root,x);
1165 }
1166
1167 int treap_get_num(int x)
1168 {
1169     return get_num(root,x);
1170 }
1171
1172 int treap_get_prev(int x)
1173 {
1174     return get_prev(root,x);
1175 }
1176
1177 int treap_get_next(int x)
1178 {
1179     return get_next(root,x);
1180 }
1181
1182 void treap_modify(int l,int r,int x,int v,int y)
1183 {
1184     if(l==r) return;
1185     int m=(l+r)>>1;
1186     treap_modify(l,m,x,v,y);
1187     treap_modify(m+1,r,x,v,y);
1188     root=treap_modify(l,r,x,v,y);
1189 }
1190
1191 void treap_build(int l,int r)
1192 {
1193     if(l>=r) return;
1194     int m=(l+r)>>1;
1195     treap_build(l,m);
1196     treap_build(m+1,r);
1197     root=treap_build(l,r);
1198 }
1199
1200 void treap_init()
1201 {
1202     root=0;
1203     n=0;
1204 }
1205
1206 void treap_build(int l,int r)
1207 {
1208     if(l>=r) return;
1209     int m=(l+r)>>1;
1210     treap_build(l,m);
1211     treap_build(m+1,r);
1212     root=treap_build(l,r);
1213 }
1214
1215 void treap_insert(int x)
1216 {
1217     _insert(root,x);
1218     pushup(root);
1219 }
1220
1221 void treap_delete(int x)
1222 {
1223     _delete(root,x);
1224     pushup(root);
1225 }
1226
1227 int treap_get_rank(int x)
1228 {
1229     return get_rank(root,x);
1230 }
1231
1232 int treap_get_num(int x)
1233 {
1234     return get_num(root,x);
1235 }
1236
1237 int treap_get_prev(int x)
1238 {
1239     return get_prev(root,x);
1240 }
1241
1242 int treap_get_next(int x)
1243 {
1244     return get_next(root,x);
1245 }
1246
1247 void treap_modify(int l,int r,int x,int v,int y)
1248 {
1249     if(l==r) return;
1250     int m=(l+r)>>1;
1251     treap_modify(l,m,x,v,y);
1252     treap_modify(m+1,r,x,v,y);
1253     root=treap_modify(l,r,x,v,y);
1254 }
1255
1256 void treap_build(int l,int r)
1257 {
1258     if(l>=r) return;
1259     int m=(l+r)>>1;
1260     treap_build(l,m);
1261     treap_build(m+1,r);
1262     root=treap_build(l,r);
1263 }
1264
1265 void treap_init()
1266 {
1267     root=0;
1268     n=0;
1269 }
1270
1271 void treap_build(int l,int r)
1272 {
1273     if(l>=r) return;
1274     int m=(l+r)>>1;
1275     treap_build(l,m);
1276     treap_build(m+1,r);
1277     root=treap_build(l,r);
1278 }
1279
1280 void treap_insert(int x)
1281 {
1282     _insert(root,x);
1283     pushup(root);
1284 }
1285
1286 void treap_delete(int x)
1287 {
1288     _delete(root,x);
1289     pushup(root);
1290 }
1291
1292 int treap_get_rank(int x)
1293 {
1294     return get_rank(root,x);
1295 }
1296
1297 int treap_get_num(int x)
1298 {
1299     return get_num(root,x);
1300 }
1301
1302 int treap_get_prev(int x)
1303 {
1304     return get_prev(root,x);
1305 }
1306
1307 int treap_get_next(int x)
1308 {
1309     return get_next(root,x);
1310 }
1311
1312 void treap_modify(int l,int r,int x,int v,int y)
1313 {
1314     if(l==r) return;
1315     int m=(l+r)>>1;
1316     treap_modify(l,m,x,v,y);
1317     treap_modify(m+1,r,x,v,y);
1318     root=treap_modify(l,r,x,v,y);
1319 }
1320
1321 void treap_build(int l,int r)
1322 {
1323     if(l>=r) return;
1324     int m=(l+r)>>1;
1325     treap_build(l,m);
1326     treap_build(m+1,r);
1327     root=treap_build(l,r);
1328 }
1329
1330 void treap_init()
1331 {
1332     root=0;
1333     n=0;
1334 }
1335
1336 void treap_build(int l,int r)
1337 {
1338     if(l>=r) return;
1339     int m=(l+r)>>1;
1340     treap_build(l,m);
1341     treap_build(m+1,r);
1342     root=treap_build(l,r);
1343 }
1344
1345 void treap_insert(int x)
1346 {
1347     _insert(root,x);
1348     pushup(root);
1349 }
1350
1351 void treap_delete(int x)
1352 {
1353     _delete(root,x);
1354     pushup(root);
1355 }
1356
1357 int treap_get_rank(int x)
1358 {
1359     return get_rank(root,x);
1360 }
1361
1362 int treap_get_num(int x)
1363 {
1364     return get_num(root,x);
1365 }
1366
1367 int treap_get_prev(int x)
1368 {
1369     return get_prev(root,x);
1370 }
1371
1372 int treap_get_next(int x)
1373 {
1374     return get_next(root,x);
1375 }
1376
1377 void treap_modify(int l,int r,int x,int v,int y)
1378 {
1379     if(l==r) return;
1380     int m=(l+r)>>1;
1381     treap_modify(l,m,x,v,y);
1382     treap_modify(m+1,r,x,v,y);
1383     root=treap_modify(l,r,x,v,y);
1384 }
1385
1386 void treap_build(int l,int r)
1387 {
1388     if(l>=r) return;
1389     int m=(l+r)>>1;
1390     treap_build(l,m);
1391     treap_build(m+1,r);
```



```

192 int query(int rt,int l,int r,int op,int L,int R,int x
    )
193 {
194     if(L<=l&&r<=R)
195     {
196         if(op==1) return treap.get_rank(root[rt],x)-2;
197         if(op==2) return treap.get_num(root[rt],x)-1;
198         if(op==4) return treap.get_prev(root[rt],x);
199         if(op==5) return treap.get_nxt(root[rt],x);
200     }
201     int m=l+r>>1,ret;
202     if(op==1||op==2)
203     {
204         ret=0;
205         if(L<=m) ret+=query(rt<<1,l,m,op,L,R,x);
206         if(m<R) ret+=query(rt<<1|m+1,r,op,L,R,x);
207     }
208     if(op==4)
209     {
210         ret=-inf;
211         if(L<=m) ret=max(ret,query(rt<<1,l,m,op,L,R,x)
            );
212         if(m<R) ret=max(ret,query(rt<<1|m+1,r,op,L,R
            ,x));
213     }
214     if(op==5)
215     {
216         ret=inf;
217         if(L<=m) ret=min(ret,query(rt<<1,l,m,op,L,R,x)
            );
218         if(m<R) ret=min(ret,query(rt<<1|m+1,r,op,L,R
            ,x));
219     }
220     return ret;
221 }
222
223 int main()
224 {
225     srand(time(0));
226     scanf("%d",&n,&m);
227     build(1,1,n);
228     for(int i=1;i<=n;i++)
229     {
230         scanf("%d",a+i);
231         modify(1,1,n,i,a[i],1);
232     }
233     while(m--)
234     {
235         int op,l,r,k,pos;
236         scanf("%d",&op);
237         if(op==1)
238         {
239             scanf("%d%d",&l,&r,&k);
240             printf("%d\n",query(1,1,n,op,l,r,k)+1);
241         }
242         else if(op==2)
243         {
244             scanf("%d%d",&l,&r,&k);
245             int L=-inf,R=inf,mid;
246             while(L<R)
247             {
248                 mid=(L+R+1)>>1;
249                 if(query(1,1,n,1,l,r,mid)+1>k) R=mid-1;
250                 else L=mid;

```

```

251         printf("%d\n",L);
252     }
253     else if(op==3)
254     {
255         scanf("%d",&pos,&k);
256         modify(1,1,n,pos,a[pos],-1);
257         a[pos]=k;
258         modify(1,1,n,pos,k,1);
259     }
260     else
261     {
262         scanf("%d%d",&l,&r,&k);
263         printf("%d\n",query(1,1,n,op,l,r,k));
264     }
265 }
266 return 0;
267 }

```

4.8.2 树状数组套线段树

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /*
5     带单点修区间k小
6     用的时候注意下空间 时空 O(nlog^2(n))
7     外层 add(pos,x,y) 空间上为pos的点且值域上为x的点加上y
8         query(l,r,k) 询问区间[l,r]里k小
9     内层 modify 值域线段树动态开点
10        query 值域线段树区间k小
11    VAL 值域大小
12 */
13
14 const int MAXN=200005;
15 int n,a[MAXN],X[MAXN],Y[MAXN],c1,c2,VAL;
16 struct SEG
17 {
18     int root[MAXN],lc[MAXN*500],rc[MAXN*500],cnt[MAXN
        *500],tot;
19     void modify(int &rt,int l,int r,int x,int y)
20     {
21         if(rt==0) rt=++tot;
22         cnt[rt]+=y;
23         if(l==r) return;
24         int m=l+r>>1;
25         if(x<=m) modify(lc[rt],l,m,x,y);
26         else modify(rc[rt],m+1,r,x,y);
27     }
28     int query(int l,int r,int k)
29     {
30         if(l==r) return l;
31         int sum=0,m=l+r>>1;
32         for(int i=0;i<c1;i++) sum+=cnt[lc[X[i]]];
33         for(int i=0;i<c2;i++) sum+=cnt[lc[Y[i]]];
34         if(sum>=k)
35         {
36             for(int i=0;i<c1;i++) X[i]=lc[X[i]];
37             for(int i=0;i<c2;i++) Y[i]=lc[Y[i]];
38             return query(l,m,k);
39         }
40         else
41         {
42             for(int i=0;i<c1;i++) X[i]=rc[X[i]];
43             for(int i=0;i<c2;i++) Y[i]=rc[Y[i]];

```



```

44         return query(m+1,r,k-sum);
45     }
46 }
47 }seg;
48
49 void add(int pos,int x,int y)
50 {
51     for(;pos<=n;pos+=pos&-pos) seg.modify(seg.root[pos]
52         ,1,VAL,x,y);
53 }
54
55 int query(int l,int r,int k)
56 {
57     c1=c2=0;
58     for(int i=l-1;i;i-=i&-i) X[c1++]=seg.root[i];
59     for(int i=r;i;i-=i&-i) Y[c2++]=seg.root[i];
60     return seg.query(1,VAL,k);
61 }

```

4.9 K-D Tree

```

1  #include<bits/stdc++.h>
2  #define fi first
3  #define se second
4  using namespace std;
5  //HDOJ 4347
6
7  /*
8   给出 n 个 K 维的点，又给出 t 个查询，每个查询也给出一
9   个 K 维的点，要求查询 n 个点中距离这个点最近的 M
10   个点。
11 */
12
13 const int maxn=5e4+10;
14 const int maxk=5;
15 int k,idx;
16 struct Point
17 {
18     int x[maxk];
19     bool operator<(const Point &o)const
20     {
21         return x[idx]<o.x[idx];
22     }
23     void print()
24     {
25         for(int i=0;i<k;i++) printf("%d%c",x[i],(i==k
26             -1)?'\n':' ');
27     }
28 }poi[maxn];
29 typedef pair<double,Point> P;
30 priority_queue<P> Q;
31 void clear(priority_queue<P> &Q)
32 {
33     if(Q.empty()) return;
34     priority_queue<P> tp;
35     swap(Q,tp);
36 }
37
38 struct kdTree
39 {
40     #define sqr(x) ((x)*(x))
41     #define ls (rt<<1)
42     #define rs (rt<<1|1)
43     Point o[maxn<<2];
44     int son[maxn<<2];

```

```

41
42 void build(int rt,int l,int r,int dep)
43 {
44     if(l>r) return;
45     son[rt]=r-l, son[ls]=son[rs]=-1;
46     idx=dep%k;
47     int mid=(l+r)>>1;
48     nth_element(poi+l,poi+mid,poi+r+1);
49     o[rt]=poi[mid];
50     build(ls,l,mid-1,dep+1);
51     build(rs,mid+1,r,dep+1);
52 }
53
54 void query(int rt,Point p,int m,int dep)
55 {
56     if(son[rt]==-1) return;
57     P nd(0,o[rt]);
58     for(int i=0;i<k;i++) nd.fi+=sqr(nd.se.x[i]-p.x
59         [i]);
60     int dim=dep%k, x=ls, y=rs; bool fg=0;
61     if(p.x[dim]>=o[rt].x[dim]) swap(x,y);
62     if(~son[x]) query(x,p,m,dep+1);
63     if(Q.size()<m) Q.push(nd), fg=1;
64     else
65     {
66         if(nd.fi<Q.top().fi) Q.pop(), Q.push(nd);
67         if(sqr(p.x[dim]-o[rt].x[dim])<Q.top().fi)
68             fg=1;
69     }
70     if(~son[y] && fg) query(y,p,m,dep+1);
71 }kdt;
72
73 int n;
74 int main()
75 {
76     while(scanf("%d",&n,&k)!=EOF)
77     {
78         for(int i=0;i<n;i++) for(int j=0;j<k;j++)
79             scanf("%d",&(poi[i].x[j]));
80         kdt.build(1,0,n-1,0);
81         int t; scanf("%d",&t);
82         stack<Point> ans;
83         while(t--)
84         {
85             Point qry;
86             for(int i=0;i<k;i++) scanf("%d",&qry.x[i]);
87             int m; scanf("%d",&m);
88             clear(Q);
89             kdt.query(1,qry,m,0);
90             printf("the closest %d points are:\n",m);
91             while(Q.size()) ans.push(Q.top().se), Q.pop
92                 ();
93             while(ans.size()) ans.top().print(), ans.
94                 pop();
95         }
96     }
97 }

```

4.10 分治

4.10.1 CDQ

```

1  #include <bits/stdc++.h>
2  using namespace std;
3

```

```

4  /*
5     严格大于的三维偏序
6     无法处理重复的数字，但是注意"大于"的神秘排序方法
7  */
8
9  const int N=500005;
10
11  int n,c[N];
12  struct node{
13     int a,b,c,cnt;
14  }o[N],t[N];
15
16  bool cmp(node a,node b){
17     if(a.a!=b.a) return a.a>b.a;
18     else return a.c<b.c;
19  }
20
21  inline int lowbit(int x){return x&-x;}
22  int tr[N];
23  void add(int x,int y){
24     for(;x<=n;x+=lowbit(x)) tr[x]+=y;
25  }
26  int sum(int x){
27     int res=0;
28     for(;x>=1;x-=lowbit(x)) res+=tr[x];
29     return res;
30  }
31
32  void cdq(int l,int r){
33     if(l==r) return;
34     int m=(l+r)/2;
35     cdq(l,m);
36     cdq(m+1,r);
37     int p=l,q=m+1,tot=1;
38     while(p<=m&&q<=r){
39         if(o[p].b>o[q].b) add(o[p].c,1),t[tot++]=o[p++];
40         else o[q].cnt+=sum(n)-sum(o[q].c),t[tot++]=o[q++];
41     }
42     while(p<=m) add(o[p].c,1),t[tot++]=o[p++];
43     while(q<=r) o[q].cnt+=sum(n)-sum(o[q].c),t[tot++]=o[q++];
44     for(int i=l;i<=m;++i) add(o[i].c,-1);
45     for(int i=l;i<=r;++i) o[i]=t[i];
46  }
47
48  int main()
49  {
50     scanf("%d",&n);
51     for(int i=1;i<=n;++i) scanf("%d",&o[i].a);
52     for(int i=1;i<=n;++i) scanf("%d",&o[i].b);
53     for(int i=1;i<=n;++i) scanf("%d",&o[i].c),c[i]=o[i].c;
54     sort(c+1,c+1+n);
55     for(int i=1;i<=n;++i) o[i].c=lower_bound(c+1,c+1+n,o[i].c)-c;
56     sort(o+1,o+1+n,cmp);
57     cdq(1,n);
58     int ans=0;
59     for(int i=1;i<=n;++i) if(o[i].cnt>0) ans++;
60     printf("%d",ans);
61     return 0;
62  }

```

```

1  #include <bits/stdc++.h>
2  using namespace std;
3
4  /*
5     解决有等于的三维偏序
6     严格小于等于的个数，可以解决重复问题，有离散化
7  */
8
9  const int maxn=500005;
10
11  int n,k;
12  int cnt[maxn]; //save the ans
13  struct ss{
14     int a,b,c,w,ans;
15  }tmps[maxn],s[maxn]; //struct
16  bool cmp1(ss x,ss y){ //sort1
17     if(x.a==y.a){
18         if(x.b!=y.b) return x.b<y.b;
19         else return x.c<y.c;
20     }
21     else return x.a<y.a;
22  }
23  bool cmp2(ss x,ss y){ //sort2
24     if(x.b!=y.b) return x.b<y.b;
25     else return x.c<y.c;
26  }
27
28  struct tree_array{//tree_array
29     int tr[maxn+5],n;
30     int lowbit(int x){return x&-x;}
31     int ask(int x){int ans=0;for(;x>=1;x-=lowbit(x))ans+=tr[x];return ans;}
32     void add(int x,int y){for(;x<=n;x+=lowbit(x))tr[x]+=y;}
33  }t;
34
35  void cdq(int l,int r){
36     if(l==r) return;
37     int m=l+r>>1;
38     cdq(l,m);
39     cdq(m+1,r);
40     sort(s+l,s+m+1,cmp2);
41     sort(s+m+1,s+r+1,cmp2); //sort2
42     int i=l,j=m+1;
43     for(;j<=r;++j){
44         while(i<=m&&s[i].b<=s[j].b){ //the second dimension
45             t.add(s[i].c,s[i].w); //use the tree_array to save the ans
46             ++i;
47         }
48         s[j].ans+=t.ask(s[j].c); //contribution
49     }
50     for(int j=l;j<=i;++j)
51         t.add(s[j].c,-s[j].w); //init the first half
52  }
53
54  int main(){
55     scanf("%d%d",&n,&k);
56     for(int i=1;i<=n;++i)
57         scanf("%d%d%d",&tmps[i].a,&tmps[i].b,&tmps[i].c);
58     sort(tmps+1,tmps+1+n,cmp1); //sort1
59     int now=0,nn=0;
60     for(int i=1;i<=n;++i){

```

```

61     now++;
62     if(tmps[i].a!=tmps[i+1].a||tmps[i].b!=tmps[i
        +1].b
63     ||tmps[i].c!=tmps[i+1].c){
64         s[++nn]=tmps[i];
65         s[nn].w=now;
66         now=0;
67     }//compress the same
68 }
69 t.n=maxn;//tree_array on the range
70 cdq(1,nn);
71 for(int i=1;i<=nn;++i)
72     cnt[s[i].ans+s[i].w-1]+=s[i].w;//
73 for(int i=0;i<n;++i)
74     printf("%d\n",cnt[i]);
75 return 0;
76 }

```

4.10.2 点分治

4.10.3 dsu on tree

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4
5  /*
6   统计每颗子树内的出现次数最多的数(们)的和
7   复杂度 O(nlogn)
8  */
9
10 int n,c[100005],cnt[100005],mx,son[100005],siz
    [100005],hson;
11 ll ans[100005],sum;
12 vector<int> e[100005];
13
14 void dfs1(int now,int fa)
15 {
16     son[now]=0,siz[now]=1;
17     for(auto to:e[now])
18     {
19         if(to==fa) continue;
20         dfs1(to,now);
21         siz[now]+=siz[to];
22         if(siz[to]>siz[son[now]]) son[now]=to;
23     }
24 }
25
26 void cal(int now,int fa,int y)
27 {
28     cnt[c[now]]+=y;
29     if(cnt[c[now]]==mx) sum+=c[now];
30     else if(cnt[c[now]]>mx) mx=cnt[c[now]],sum=c[now];
31     for(auto to:e[now])
32         if(to!=fa&&to!=hson) cal(to,now,y);
33 }
34
35 void dfs2(int now,int fa,int keep)
36 {
37     for(auto to:e[now])
38     {
39         if(to==fa||to==son[now]) continue;
40         dfs2(to,now,0);
41     }
42     if(son[now]) dfs2(son[now],now,1);

```

```

43     hson=son[now];
44     cal(now,fa,1);
45     hson=0;
46     ans[now]=sum;
47     if(!keep) cal(now,fa,-1),sum=0,mx=0;
48 }
49
50 int main()
51 {
52     scanf("%d",&n);
53     for(int i=1;i<=n;i++) scanf("%d",c+i);
54     for(int i=1,x,y;i<=n;i++)
55     {
56         scanf("%d%d",&x,&y);
57         e[x].push_back(y),e[y].push_back(x);
58     }
59     dfs1(1,1);
60     dfs2(1,1,1);
61     for(int i=1;i<=n;i++) printf("%lld ",ans[i]);
62     return 0;
63 }

```

4.10.4 整体二分

4.11 分块

4.11.1 普通分块

```

1  #include <bits/stdc++.h>
2  //luogu P3203
3  using namespace std;
4  const int N=500005;
5
6  int n,m,tot;
7  int a[N],cnt[N],pos[N];
8  int id[N],from[N],to[N];
9  int o,x,y;
10
11 void modify(int i)
12 {
13     if(i+a[i]>n)
14     {
15         pos[i]=i;
16         cnt[i]=0;
17         return;
18     }
19     if(id[i]==id[i+a[i]])
20     {
21         pos[i]=pos[i+a[i]];
22         cnt[i]=cnt[i+a[i]]+1;
23     }
24     else
25     {
26         pos[i]=i+a[i];
27         cnt[i]=1;
28     }
29 }
30
31 void ask(int x)
32 {
33     int p=x,res=0;
34     while(p!=pos[p])
35         res+=cnt[p],
36         p=pos[p];
37     printf("%d\n",res+1);

```

```

38 }
39
40 int main()
41 {
42     scanf("%d",&n);
43     tot=(int)sqrt(n);
44     for(int i=1;i<=tot;++i)
45     {
46         from[i]=(i-1)*tot+1;
47         to[i]=i*tot;
48     }
49     if(to[tot]<n)
50     {
51         tot++;
52         from[tot]=to[tot-1];
53         to[tot]=n;
54     }
55     for(int i=1;i<=tot;++i)
56     {
57         for(int j=from[i];j<=to[i];++j)
58             id[j]=i;
59     }
60     for(int i=1;i<=n;++i)
61         scanf("%d",&a[i]);
62     for(int i=n;i>=1;--i)
63         modify(i);
64     scanf("%d",&m);
65     while(m-->0)
66     {
67         scanf("%d",&o);
68         if(o==2)
69         {
70             scanf("%d%d",&x,&y);
71             x++;
72             a[x]=y;
73             for(int i=x;i>=from[id[x]];--i)
74                 modify(i);
75         }
76         else if(o==1)
77         {
78             scanf("%d",&x);
79             x++;
80             ask(x);
81         }
82     }
83     return 0;
84 }

```

```

15     if(a.id==b.id)
16     {
17         if(a.id%2==0) return a.r<b.r;
18         else return a.r>b.r;
19     }
20     else return a.id<b.id;
21 }
22
23 int now=0;
24 void del(int x)
25 {
26     now-=cnt[a[x]]*cnt[a[x]];
27     cnt[a[x]]--;
28     now+=cnt[a[x]]*cnt[a[x]];
29 }
30 void add(int x)
31 {
32     now-=cnt[a[x]]*cnt[a[x]];
33     cnt[a[x]]++;
34     now+=cnt[a[x]]*cnt[a[x]];
35 }
36
37 int main()
38 {
39     scanf("%d%d%d",&n,&m,&k);
40     int sz=sqrt(m);
41     for(int i=1;i<=n;++i)
42         scanf("%d",&a[i]);
43     for(int i=1;i<=m;++i)
44         scanf("%d%d",&q[i].l,&q[i].r),q[i].ans=i;
45     for(int i=1;i<=m;++i)
46         q[i].id=(q[i].l+sz-1)/sz;
47     sort(q+1,q+1+m,cmp);
48     int L,R;
49     L=R=q[1].l;
50     R--;
51     for(int i=1;i<=m;++i)
52     {
53         while(L<q[i].l) del(L++);
54         while(L>q[i].l) add(--L);
55         while(R>q[i].r) del(R--);
56         while(R<q[i].r) add(++R);
57         ans[q[i].ans]=now;
58     }
59     for(int i=1;i<=m;++i)
60         printf("%d\n",ans[i]);
61     return 0;
62 }

```

4.11.2 莫队

```

1 #include <bits/stdc++.h>
2 //luogu P3203
3 using namespace std;
4 const int N=500005;
5
6 int n,m,k,a[N];
7 struct node
8 {
9     int l,r,id,ans;
10 }q[N];
11 int cnt[N],ans[N];
12
13 bool cmp(node a,node b)
14 {

```

4.12 线性基

```

1 /*
2  *
3  * bool have(int x) 返回线性基中的数字能否表示数字x
4  * void ins(int x) 插入数字x
5  * int mn() 返回能表示的最小值
6  * int mx() 返回能表示的最大值
7  * void bug() 输出p数组
8  * int kth(int k) 返回能表示的所有值当中排名第k的
9  */
10
11 struct LB{
12     int p[100],N,tmp[100];
13     bool flag;

```

```

14  LB(){memset(p,0,sizeof(p));memset(tmp,0,sizeof(tmp)
    );flag=0;N=62;}
15  void ins(int x){
16      for(int i=N;i>=0;--i){
17          if(x&(1LL<<i)){
18              if(!p[i]) {p[i]=x;return;sz++;}
19              else x^=p[i];
20          }
21      }
22      flag=1;
23  }
24  int mx(){
25      int ans=0;
26      for(int i=N;i>=0;--i){
27          if((ans^p[i])>ans) ans^=p[i];
28      }
29      return ans;
30  }
31  int mn(){
32      if(flag) return 0;
33      for(int i=0;i<=N;++i)
34          if(p[i]) return p[i];
35  }
36  bool have(int x){
37      for(int i=N;i>=0;--i){
38          if(x&(1LL<<i)){
39              if(!p[i]) return 0;
40              else x^=p[i];
41          }
42      }
43      return 1;
44  }
45  int kth(int k){
46      int res=0,cnt=0;
47      k-=flag;
48      if(!k) return 0;
49      for(int i=0;i<=N;++i){
50          for(int j=i-1;j>=0;--j){
51              if(p[j]&(1LL<<j)) p[i]^=p[j];
52          }
53          if(p[i]) tmp[cnt++]=p[i];
54      }
55      if(k>=(1LL<<cnt)) return -1;//can't find
56      for(int i=0;i<cnt;++i)
57          if(k&(1LL<<i)) res^=tmp[i];
58      return res;
59  }
60  void bug(){
61      for(int i=0;i<=N;++i) cout<<p[i]<<' ';
62      cout<<endl;
63  }
64  }lb;

```

4.13 珂朵莉树

```

1  #include <bits/stdc++.h>
2  #define int long long //be careful
3  //CF896C
4  using namespace std;
5
6  /*
7   珂朵莉树的左右split顺序很重要，并且set集合一开始不要为
   空，否则会RE
8  */

```

```

9
10 const int N=1000005;
11
12 int qpow(int a,int b,int mod)
13 {
14     int res=1,tmp=a%mod;
15     while(b)
16     {
17         if(b&1) res=res*tmp%mod;
18         tmp=tmp*tmp%mod;
19         b>>=1;
20     }
21     return res;
22 }
23
24 struct node
25 {
26     int l,r;
27     mutable int v;
28     node(int L,int R=-1,int V=0):l(L),r(R),v(V){}
29     bool operator < (const node& o)const{return l<o.l
        ;}
30 };
31 set<node> s;
32 typedef set<node>::iterator it;
33
34 it split(int pos)
35 {
36     it i=s.lower_bound(node(pos));
37     if(i!=s.end()&&i->l==pos) return i;
38     --i;
39     int L=i->l,R=i->r,V=i->v;
40     s.erase(i);
41     s.insert(node(L,pos-1,V));
42     return s.insert(node(pos,R,V)).first;
43 }
44
45 void assign(int l,int r,int val)
46 {
47     it ir=split(r+1),il=split(l);
48     s.erase(il,ir);
49     s.insert(node(l,r,val));
50 }
51
52 void add(int l,int r,int val)
53 {
54     it ir=split(r+1),il=split(l);
55     for(;il!=ir;il++)
56         il->v+=val;
57 }
58
59 int rk(int l,int r,int k)
60 {
61     vector<pair<int,int>> v;
62     it ir=split(r+1),il=split(l);
63     for(;il!=ir;il++)
64         v.emplace_back(il->v,il->r-il->l+1);
65     sort(v.begin(),v.end());
66     for(int i=0;i<v.size();++i)
67     {
68         k-=v[i].second;
69         if(k<=0) return v[i].first;
70     }
71     return -1; //can't find
72 }

```

```

73
74 int sum(int l,int r,int ex,int mod)
75 {
76     it ir=split(r+1),il=split(l);
77     int res=0;
78     for(;il!=ir;il++)
79         res=(res+qpow(il->v,ex,mod)*(il->r-il->l+1)%
80             mod)%mod;
81     return res;
82 }
83 inline int read(){
84     char ch=getchar();int s=0,w=1;
85     while(ch<48||ch>57){if(ch=='-')w=-1;ch=getchar();}
86     while(ch>=48&&ch<=57){s=(s<<1)+(s<<3)+ch-48;ch=
87         getchar();}
88     return s*w;
89 }
90 inline void write(int x){
91     if(x<0)putchar('-'),x=-x;
92     if(x>9)write(x/10);
93     putchar(x%10+48);
94 }
95 //Fast I/O
96 int n,m,seed,vmax,a[N];
97 int rnd()
98 {
99     int ret=seed;
100     seed=(seed*7+13)%1000000007;
101     return ret;
102 }
103
104 signed main()
105 {
106     n=read(),m=read(),seed=read(),vmax=read();
107     for(int i=1;i<=n;++i)
108     {
109         a[i]=(rnd()%vmax)+1;
110         s.insert(node(i,i,a[i]));
111     }
112     for(int i=1;i<=m;++i)
113     {
114         int op=(rnd()%4)+1;
115         int l=(rnd()%n)+1;
116         int r=(rnd()%n)+1;
117         if(l>r) swap(l,r);
118         int x,y;
119         if(op==3) x=(rnd()%(r-l+1))+1;
120         else x=(rnd()%vmax)+1;
121         if(op==4) y=(rnd()%vmax)+1;
122         switch(op)
123         {
124             case 1:
125                 add(l,r,x);break;
126             case 2:
127                 assign(l,r,x);break;
128             case 3:
129                 write(rk(l,r,x)),puts("");break;
130             case 4:
131                 write(sum(l,r,x,y)),puts("");break;
132         }
133     }
134     return 0;
135 }

```

```

136 }

```

4.14 跳舞链

5 动态规划

5.1 SOS

```

1
2 /*
3     (1<<N)-1-a[i]是a[i]的补集
4 */
5
6 for(int i=0;i<(1<<N);i++) dp[i]=a[i];
7 for(int i=0;i<N;i++)
8     for(int mask=0;mask<(1<<N);mask++)
9         if(mask&(1<<i))
10             dp[mask]+=dp[mask^(1<<i)];

```

5.2 动态 DP

5.3 插头 DP

6 数学

6.1 三分

```

1 //答案都取r 浮点数可以取(l+r)/2
2 //浮点数极小值
3 while(l+eps<r)
4 {
5     double lm=(l+r)/2,rm=(lm+r)/2;
6     if(judge(lm)>judge(rm)) l=lm;
7     else r=rm;
8 }
9
10 //整数极小值
11 while(l+1<r)
12 {
13     int lm=(l+r)>>1,rm=(lm+r)>>1;
14     if(judge(lm)>judge(rm)) l=lm;
15     else r=rm;
16 }
17
18 //浮点数极大值
19 while(l+eps<r)
20 {
21     double lm=(l+r)/2,rm=(lm+r)/2;
22     if(judge(lm)>judge(rm)) r=rm;
23     else l=lm;
24 }
25
26 //整数极大值
27 while(l+1<r)
28 {
29     int lm=(l+r)>>1,rm=(lm+r)>>1;
30     if(judge(lm)>judge(rm)) r=rm;
31     else l=lm;
32 }

```

6.2 矩阵类

6.3 质数筛

6.3.1 埃筛

6.3.2 线筛

6.4 质数判定

6.4.1 Miller Rabin

$$0 < x < p, p \mid x^2 \equiv 1 \pmod{p} \Leftrightarrow x = 1 \vee x = p - 1$$

$$0 < a < p, p \mid a^{p-1} \equiv 1 \pmod{p}$$

判错概率不超过 $4^{-S}(S \rightarrow \infty)$, 时间复杂度 $O(S * \log n)$

```

1 //HDU2138多组输入判断n个数中几个是质数
2 #include <cstdio>
3 #include <cstdlib>
4 #include <ctime>
5 #include <random>
6 using namespace std;
7 typedef long long ll;
8
9 inline ll qpow(ll a, ll b, ll P){ //a^b%P
10     ll ans=1;
11     for(; b>=1; a=a*a%P)
12         if(b&1) ans=ans*a%P;
13     return ans;}
14
15 inline bool MR(int tc, ll n){ //用随机测试数据tc测n, 返回n是质数?1:0
16     ll u=n-1, t=0;
17     while(!(u&1)) ++t, u>>=1; //把n-1表示为u*2^t, 循环到u为奇数为止
18     ll x=qpow(tc,u,n);
19     if(x==1) return 1;
20     for(int i=1; i<=t; ++i, x=x*x%n)
21         if(x!=n-1&&x!=1&&x*x%n==1) return 0;
22     return x==1;
23 }
24
25 mt19937_64 rnd(time(0)); //c++11的std的魔法, 下文可用rnd()生成[0,2^64)
26
27 inline bool isPrime(ll n, int S=10){ //对n测试S次, 返回n是质数?1:0
28     if(n==2) return 1;
29     if(n<2 || !(n&1)) return 0;
30     // srand(time(0));
31     while(S--){
32         // if(!MR((ll)rand()*rand()*rand()%(n-1)+1,n))
33             return 0;
34         if(!MR(rnd()%(n-1)+1,n)) return 0;
35     }
36     return 1;
37 }
38
39 inline void solve(int N){
40     int cnt=0;
41     while(N--){
42         ll n; scanf("%lld",&n);
43         if(isPrime(n)) ++cnt;
44     }
45     printf("%d\n",cnt);
46 }
47
48 int main(int argc, char** argv){

```

```

47     int _; while(~scanf("%d",&_))
48         solve(_);
49     return 0;
50 }

```

6.5 质因数分解

6.5.1 Pollard-Rho

Pollard 设计的求大数因子方法, 因为随机找的数会形成环形, 因此常称为 ρ 算法大致原理是随机找几个数求差, 让差与大数 N 求 \gcd , 据说复杂度约 $O(N^{1/4})$ 随机找数的方法见 ρ 函数。若要求质因子, 一般需要用一下 MillerRabin 判素数

```

1 //洛谷P4718求最大质因子
2 #include <cstdio>
3 #include <cstdlib>
4 #include <ctime>
5 #include <random>
6 #include <algorithm>
7 using namespace std;
8 typedef long long ll;
9 typedef unsigned long long ull;
10 typedef long double lb;
11
12 ll m, ans; //对每个m, 将其最大质因子存进ans
13
14 mt19937_64 rnd(time(0)); //c++11的std的魔法, 下文可用rnd()生成[0,2^64)
15
16 ll read(){
17     ll k=0; int f=1;
18     char c=getchar();
19     while(!isdigit(c)){ if(c=='-') f=-1; c=getchar(); }
20     while(isdigit(c)) k= k*10 + c-48, c=getchar();
21     return k*f;
22 }
23
24 inline ll Abs(ll x){ return x<0 ? -x : x;} //取绝对值
25
26 inline ll qmul(ull x, ull y, ll p){ //O(1)x*y%p
27     return (x*y - (ull)((lb)x/p*y)*p + p)%p;
28 }
29
30 inline ll qpow(ll x, ll y, ll p){ //x^y%p
31     ll res=1;
32     for(; y>=1; x=qmul(x,x,p))
33         if(y&1) res=qmul(res,x,p);
34     return res;
35 }
36
37 inline bool MR(int tc, ll p){ //miller rabin判质数, 用tc检测p是不是质数
38     if(qpow(tc,p-1,p)!=1) return 0; //费马小定理
39     ll y=p-1, z;
40     while(!(y&1)){ //二次探测
41         y>>=1;
42         z=qpow(tc,y,p);
43         if(z!=1 && z!=p-1) return 0;
44         if(z==p-1) return 1;
45     }
46     return 1;
47 }
48

```



```

49 inline bool isPrime(ll x){ //用5个小质数做MR测试, 返回n
    是质数?1:0
50     if(x<2) return 0;
51     if(x==2 || x==3 || x==5 || x==7 || x==43) return
        1;
52     return MR(2,x) && MR(3,x) && MR(5,x) && MR(7,x) &&
        MR(43,x);
53 }
54
55 inline ll rho(ll p){ //玄学求出p的非平凡因子
    ll x,y,s,c; //s用来存 (y-x) 的乘积
56     for(;;){ //求出一个因子来
57         y=x+rnd()%p;
58         c=rnd()%p;
59         s=1;
60         int i=0, I=1;
61         while(++i){ //开始玄学倍增生成
62             x=(qmul(x,x,p)+c)%p; //以平方再+c的方式生成下
                一个x
63             s=qmul(s,Abs(y-x),p); //将每一次的 (y-x) 都累
                乘起来
64             if(x==y || !s) break; //换下一组, 当s=0时, 继
                续下去是没意义的
65             if(!(i%127) || i==I){ //每127次求一次gcd, 以
                及按j倍增的求gcd
66                 ll g=__gcd(s,p);
67                 if(g>1) return g;
68                 if(i==I) y=x, I<=1; //维护倍增正确性, 并
                判环
69             }
70         }
71     }
72 }
73 }
74
75 inline void PR(ll p){ //找p的最大质因子, 存进全局变量
    ans
76     if(p<=ans) return ; //最优性剪枝
77     if(isPrime(p)) return ans=p, void();
78     ll pi=rho(p); //玄学求出任一因子p_i
79     while(p%pi==0) p/=pi;
80     return PR(pi), PR(p); //向下分治
81 }
82
83 int main(){
84     int n=read();
85     for(int i=0; i<n; ++i){
86         ans=1;
87         PR( m=read() );
88         if(ans==m) puts("Prime");
89         else printf("%lld\n",ans);
90     }
91     return 0;
92 }

```

6.6 逆元

6.6.1 EX-GCD 求逆元

a 与 P 互质是 a 在模 P 时有乘法逆元的充要条件用扩展欧几里得算法解出的 $ax + by = 1$ 的解 x 即为 a 在模 P 时的乘法逆元顺带一提, 有乘法逆元时, 根据欧拉定理, 逆元一定为 $a^{\phi(P)-1} \bmod P$

```

1 //洛谷P1082
2 #include <bits/stdc++.h>
3 typedef long long ll;
4

```

```

5 ll a,P,x,y;
6
7 inline ll exgcd(ll a, ll b, ll &x, ll &y){
8     if(!b) return x=1, y=0, a;
9     ll g=exgcd(b, a%b, x, y);
10    ll z=x; x=y; y=z-a/b*y;
11    return g;
12 }
13
14 inline ll exinv(int a,int P){ //用exgcd求a模P的逆元,
    无解时返回-1
15    ll x,y;
16    if(exgcd(a,P,x,y)!=1) return -1;
17    else return (x%P+P)%P;
18 }
19
20 int main(){
21     scanf("%lld%lld",&a,&P);
22     printf("%lld",exinv(a,P));
23     return 0;
24 }

```

6.6.2 线性筛逆元

6.6.3 阶乘逆元

6.7 欧拉函数

6.7.1 欧拉线筛

6.7.2 求单个数的欧拉函数

6.7.3 欧拉降幂

费马小定理: p 为质数时,

$$a^{p-1} \equiv 1 \pmod{p}$$

欧拉定理: a, p 互质时,

$$a^{\phi(p)} \equiv 1 \pmod{p}$$

扩展欧拉定理: a, p 不互质时,

$$a^b \equiv \begin{cases} a^{b \bmod \phi(p) + \phi(p)} & b \geq \phi(p) \\ a^b & 0 \leq b < \phi(p) \end{cases} \pmod{p}$$

```

1 //SP10050:用扩展欧拉定理求乘方塔a^a^a... (b个a)的后九位
    数
2 #include <bits/stdc++.h>
3 using namespace std;
4 typedef long long ll;
5
6 struct ST{
7     int v;
8     bool ge; //大于等于模数与否
9     ST(int v=0,bool g=0): v(v), ge(g) {}
10 }
11
12 ST qpow(ll a,ll b,int p){ //快速幂过程中取模过p iff 返
    回值.ge==1
13     ll ans=1ll;
14     bool ge=0;
15     while(b){
16         if(b&1)
17             ans*=a,
18             ge |= ans>=p,
19             ans%=p;
20         b>>=1;

```



```

21     if(!b) break; //防止被没有乘到的a更新ge
22     a*=a;
23     ge |= a>=p, //注意ans*取余后的a可能更新不了ge, 在
        这也要更新
24     a%=p;
25 }
26 return ST(ans,ge);
27 }
28
29 map<int,int>eu;
30 ll euler(ll n){ //欧拉函数值
31     if(eu[n]) return eu[n];
32     ll n0=n, ans=n, ed=sqrt(n);
33     for(int i=2; i<=ed; ++i)
34         if(n%i==0){
35             ans-=ans/i;
36             while(n%i==0) n/=i;
37         }
38     if(n>1) ans-=ans/n;
39     return eu[n0]=ans;
40 }
41
42 ST tower(ll a,ll b,int p){ //计算b层a取余p的值
43     if(p==1) return ST(0,1); //特判取模1的特殊情况
44     if(a==1) return ST(1,0); //特判不取模1但底为1的特殊
        情况
45     if(b==1) return a<p? ST(a,0): ST(a%p,1); //递归终
        点
46     int phip=euler(p);
47     ST ans=tower(a,b-1,phip); //递归计算取余phip后的指
        数
48     if(ans.ge) ans.v+=phip; //扩展欧拉定理
49     return qpow(a, ans.v, p);
50 }
51
52 void solve(){
53     ll a,b; scanf("%lld%lld",&a,&b);
54     if(b==0) return printf("%d\n",1), void(0);
55     else if(a==0) return printf("%d\n", b%2? 0: 1),
        void(0);
56     ST ans=tower(a,b,1000000000);
57     if(ans.ge) printf("...%09d\n",ans.v);
58     else printf("%d\n",ans.v);
59 }
60
61 int main(int argc, char** argv) {
62     int _; scanf("%d",&_); while(--)
63         solve();
64     return 0;
65 }

```

6.7.4 一般积性函数求法

6.8 EX-GCD

6.9 同余方程组

6.9.1 CRT

6.9.2 EXCRT

求解 $x \bmod m_i = a_i$ 方程组, 其中 m_i 不一定为质数和 CRT 本身没啥关系, 是用数学归纳法求解齐次同余方程组的

```

1 //洛谷P4777
2 #include <bits/stdc++.h>
3 typedef long long ll;
4 const int MN = 3e5 + 5;
5
6 ll a[MN],m[MN];
7
8 inline ll exgcd(ll a, ll b, ll &x, ll &y){
9     if(!b) return x=1, y=0, a;
10    ll g=exgcd(b, a%b, x, y);
11    ll z=x; x=y; y=z-a/b*y;
12    return g;}
13
14 ll smul(ll a,ll b,ll p){ //记得传参时先给ab取余一发p
15    ll ans=0;
16    for(;b>>=1){
17        if(b&1) ans= (ans+a)%p;
18        a= (a<<1)%p;}
19    return ans;}
20
21 //ll qmul(ll a,ll b,ll p){ //玄学高精度乘法, 备用, 可能
    可以代替上一个
22 // a%=p, b%=p;
23 // ll t=(long double)a*b/p;
24 // ll ans=a*b-t*p;
25 //return ans<0? ans+p: ans;}
26
27 ll excrt(int n){ //解[0,n)
28    ll X, Y, M=m[0], ans=a[0];
29    for(int i=1; i<n; ++i){
30        ll A=M, B=m[i];
31        ll c=(a[i]-ans%B+B)%B; //新同余方程的右部
32        ll g=exgcd(A,B,X,Y);
33        if(c%g!=0) return -1;
34        X=smul(X,c/g,B/g);
35        ans+=X*M;
36        M*=B/g;
37        ans=(ans%M+M)%M;}
38    return (ans%M+M)%M;}
39
40 int main(){
41     int n; scanf("%d",&n);
42     for(int i=0; i<n; ++i) scanf("%lld%lld",m+i,a+i);
43     printf("%lld",excrt(n));
44     return 0;
45 }

```

6.10 N 次剩余

6.10.1 模奇质数的 2 次剩余

```

1 /*
2 求解  $x^2 \equiv P \pmod n$ , 当P为奇质数时, 可用类似虚数平方的方法求
    解,
3 类似实数域一元二次方程, 答案可能为两不同解, 两相同解或无实
    数解, 详见solve()
4 */
5 //洛谷P5491求二次剩余
6 #include <bits/stdc++.h>
7 using namespace std;
8 typedef long long ll;
9
10 int P; //模数, 为了避免传参而放在全局变量
11

```

```

12 inline ll qpow(ll a,int b){ //a^b%P
13     ll ans=1;
14     for(; b>=1,a=a*a%P)
15         if(b&1) ans=ans*a%P;
16     return ans;}
17
18 ll t,tt; //tt为t的平方。注意!!! 复数类中每次乘法都要用到tt!!!
19 struct CP{ //求解二次剩余专用的魔改复数类
20     ll x,y;
21     CP(ll x=0,ll y=0):x(x),y(y){}
22     CP operator*=(const CP&r){ //乘以r, 模数P为全局变量
23         return *this = CP((x*r.x+y*r.y%P*tt)%P,(y*
24             r.x%P+x*r.y%P)%P);
25     }
26     CP qpow(int n){ //n次幂, 模数P为全局变量
27         CP rt = CP(1,0);
28         for(; n>=1){
29             if(n&1) rt *= *this;
30             *this *= *this;
31         }
32         return *this = rt;
33     };
34 };
35 int cipolla(int n){ //求x*x%P=n的一个解, P是奇质数, 无
36     解时return-1
37     if(n==0) return 0;
38     if(qpow(n,(P-1)>>1)==P-1) return -1; //无解
39     srand(time(0)); //初始化随机数种子
40     for(;;){ //随机找到一个满足break条件的t即可
41         t=rand()%P;
42         tt=(t*t%P-n+P)%P;
43         if(qpow(tt,(P-1)>>1)==P-1) break;
44     }
45     CP rt=CP(t,1).qpow((P+1)>>1);
46     return rt.x;
47 }
48 inline void solve(){
49     int n; scanf("%d",&n,&P); //注意P是全局变量, 之后
50     就不传参了
51     int ans=cipolla(n);
52     int ans2=(P-ans)%P; //此处取模纯粹是为了避免ans为0时
53     ans2为P
54     if(ans==-1) return puts("Hola!"), void();
55     if(ans2<ans) swap(ans,ans2);
56     if(ans!=ans2) printf("%d %d\n",ans,ans2);
57     else printf("%d\n",ans);
58 }
59 int main(int argc, char** argv){
60     int _; scanf("%d",&_); while(--)
61         solve();
62     return 0;
63 }

```

6.10.2 N 次剩余

6.11 数论分块

6.12 高斯消元

6.12.1 普通消元

6.12.2 异或方程组消元

6.13 莫比乌斯反演

6.13.1 莫比乌斯函数

6.13.2 杜教筛

6.13.3 洲阁筛

6.13.4 min25 筛

6.14 BSGS

求解 $a^x \bmod p = b$: 当模数 p 是质数时可直接使用 *baby_step_giant_step*, 设 p 的正平方根为 s , BSGS 的原理是将 a 的 p 次幂分块成 s 个,

预处理好大约 $s+1$ 个 $b * a^i (i \in [0, s])$, 再用遍历各分块, 查找是否存在 a^{s*i} 等于预处理的 $b * a^i$; 当模数 p 不是质数时, 在 EXBSGS 先约分 b, p 到 a, p 互质, 再调用 BSGS。

```

1 //洛谷P4195
2 #include <bits/stdc++.h>
3 using namespace std;
4 typedef long long ll;
5
6 inline ll exgcd(ll a, ll b, ll &x, ll &y){
7     if(!b) return x=1, y=0, a;
8     ll g=exgcd(b, a%b, x, y);
9     ll z=x; x=y; y=z-a/b*y;
10    return g;}
11
12 inline ll exinv(int a,int P){ //用exgcd求a模P的逆元,
13    无解时返回-1
14    ll x,y;
15    if(exgcd(a,P,x,y)!=1) return -1;
16    else return (x%P+P)%P;
17 }
18
19 inline ll qpow(ll a,int b,int P){ //a^b%P
20     ll ans=1;
21     for(;b>=1,a=a*a%P)
22         if(b&1) ans=ans*a%P;
23     return ans;}
24
25 int bsgs(int a, int b, int p){ //a^x=b%p的最小非负x,
26    无解时返回-1
27    unordered_map<int,int>hsh;
28    a%=p, b%=p;
29    int s=sqrt(p)+1, bai=b;
30    for(int i=0; i<s; ++i) hsh[bai]=i, bai=ll(bai)*a%p;
31
32    int as=qpow(a,s,p);
33    if(as==0) return b==0? 1: -1;
34    int asi=1;
35    for(int i=0; i<=s; ++i){
36        int t=hsh.find(asi)==hsh.end()? -1 : hsh[asi];
37        if(t>=0&&s*i>=t) return s*i-t;
38        asi=ll(asi)*as%p;
39    }
40    return -1;

```

```

38 }
39
40 int exbsgs(int a, int b, int p){ //a^x=b%p 的最小非负x
    , 无解时返回 -1
    a%=p, b%=p;
    if(b==1) return 0;
    int k=1, cnt=0, d;
    while((d=__gcd(a,p))!=1){
        if(b%d) return -1;
        p/=d, b/=d, k=1ll(a)/d*k%p, ++cnt;
        if(b==k) return cnt;
    }
    int ans=bsgs(a,1ll(b)*exinv(k,p)%p,p);
    if(ans>=0) ans+=cnt;
    return ans;
}

53
54 inline int solve(){ //模板题中以全0为输入结尾
55     int a,b,p;
56     cin>>a>>p>>b;
57     if(!a && !b && !p) return 0;
58     int x=exbsgs(a,b,p);
59     if(x==-1) cout<<"No Solution\n";
60     else cout<<x<<"\n";
61     return 1;
62 }
63
64 int main(int argc, char** argv){
65     ios::sync_with_stdio(0);
66     while(solve());
67     return 0;
68 }

```

6.15 FFT

6.16 FFT

```

1  /*
2  大致原理: O(NlogN)地将一个N次多项式系数表达式转化成关于(1
    的N次复数根)的N个点值
3  对两个多项式的点值做乘法, 得到新的点值表达式, 再O(NlogN)
    地转回系数表达式
4  利用R数组和“蝴蝶定理”可以将递归FFT过程转为循环过程, 优化
    复杂度的常数
5  写得很乱, 欢迎有空的同学来更新个好看点的
6  */
7  //洛谷P3803裸的多项式乘积
8  #include <bits/stdc++.h>
9  using namespace std;
10 typedef long long ll;
11 typedef double db;
12
13 const int MN = 3e6 + 5;
14 const db pi = acos(-1);
15
16 struct CP{
17     db x,y; //实部, 虚部
18     CP (db x=0, db y=0): x(x), y(y) {}
19     CP operator+(CP &t){ return CP(x+t.x, y+t.y); }
20     CP operator-(CP &t){ return CP(x-t.x, y-t.y); }

```

```

21     CP operator*(CP &t){ return CP(x*t.x - y*t.y, x*t.
        y + y*t.x); }
22 }a[MN],b[MN];
23
24 int read(){
25     int k=0, f=1;
26     char c=getchar();
27     while(!isdigit(c)){ if(c=='-') f=-1; c=getchar();
        }
28     while(isdigit(c)) k= k*10 + c-48, c=getchar();
29     return k*f;
30 }
31 void write(int x){
32     if(x<0){ putchar('-'); x=~(x-1); }
33     int s[20],top=0;
34     while(x){ s[++top]=x%10; x/=10; }
35     if(!top) s[++top]=0;
36     while(top) putchar(s[top--]+'0');
37 }
38
39 int N,M,n,ln; //左多项式次数, 右多项式次数, fft次数及其
    位数
40 int R[MN]; //01串逆转后对应的下标
41 inline void calR(){ //通过NM初始化以上数据
42     n=1;
43     while(n<=N+M) n<<=1, ++ln;
44     for(int i=1; i<n; ++i) R[i] = (R[i>>1] >>1) | ((i
        &1)<< ln-1);
45 }
46 inline void init(){
47     N=read(); M=read();
48     for(int i=0; i<=N; ++i) a[i].x=read();
49     for(int i=0; i<=M; ++i) b[i].x=read();
50     calR();
51 }
52 void fft(CP c[], int f=1){ //f取-1时是逆变换
53     for(int i=0; i<n; ++i) if(i<R[i]) swap(c[i], c[R[i]
        ]));
54     for(int j=1; j<n; j<<=1){
55         CP wn(cos(pi/j), f*sin(pi/j));
56         for(int k=0; k<n; k+=(j<<1)){
57             CP t(1, 0);
58             for(int l=0; l<j; ++l){
59                 CP cl=c[k+l], cr=t*c[j+k+l];
60                 c[k+l]=cl+cr;
61                 c[j+k+l]=cl-cr;
62                 t=t*wn;
63             }
64         }
65     }
66 }
67
68 int main(int argc, char** argv) {
69     // ios::sync_with_stdio(0);
70     init();
71     fft(a), fft(b);
72     for(int i=0; i<=n; ++i) a[i]=a[i]*b[i];
73     fft(a,-1);
74     for(int i=0; i<=N+M; ++i) write(int(a[i].x/n +
        0.5)), putchar(' ');
75     return 0;
76 }

```

6.17 DFT 次数优化 FFT

```

1  /*
2  大致原理:  $O(N\log N)$ 地将一个 $N$ 次多项式系数表达式转化成关于 $(1$ 
      的 $N$ 次复数根)的 $N$ 个点值
3  对两个多项式的点值做乘法, 得到新的点值表达式, 再 $O(N\log N)$ 
      地转回系数表达式
4  利用 $R$ 数组和“蝴蝶定理”可以将递归FFT过程转为循环过程, 优化
      复杂度的常数
5  写得很乱, 欢迎有空的同学来更新个好看点的
6  */
7  //洛谷P3803裸的多项式乘积, 利用了 $(a+bi)^2 = (a^2-b^2) +$ 
       $2abi$ 减少变换次数
8  #include <bits/stdc++.h>
9  using namespace std;
10 typedef long long ll;
11 typedef double db;
12
13 const int MN = 3e6 + 5;
14 const db pi = acos(-1);
15
16 struct CP{
17     db x,y; //实部, 虚部
18     CP (db x=0, db y=0): x(x), y(y) {}
19     CP operator+(CP &t){ return CP(x+t.x, y+t.y); }
20     CP operator-(CP &t){ return CP(x-t.x, y-t.y); }
21     CP operator*(CP &t){ return CP(x*t.x - y*t.y, x*t.
        y + y*t.x); }
22 }a[MN];
23
24 int read(){
25     int k=0, f=1;
26     char c=getchar();
27     while(!isdigit(c)){ if(c=='-') f=-1; c=getchar(); }
28     while(isdigit(c)) k= k*10 + c-48, c=getchar();
29     return k*f;
30 }
31 void write(int x){
32     if(x<0){ putchar('-'); x=~(x-1); }
33     int s[20],top=0;
34     while(x){ s[++top]=x%10; x/=10; }
35     if(!top) s[++top]=0;
36     while(top) putchar(s[top--]+'0');
37 }
38
39 int N,M,n,ln; //左多项式次数, 右多项式次数, fft次数及其
      位数
40 int R[MN]; //01串逆转后对应的下标
41 inline void calR(){ //通过NM初始化以上数据
42     n=1;
43     while(n<=N+M) n<<=1, ++ln;
44     for(int i=1; i<n; ++i) R[i] = (R[i>>1] >>1) | ((i
        &1)<< ln-1);
45 }
46 inline void init(){
47     N=read(); M=read();
48     for(int i=0; i<N; ++i) a[i].x=read();
49     for(int i=0; i<M; ++i) a[i].y=read();
50     calR();
51 }
52 void fft(CP c[], int f=1){ //f取-1时是逆变换
53     for(int i=0; i<n; ++i) if(i<R[i]) swap(c[i], c[R[i]
        ]]);
54     for(int j=1; j<n; j<=<1){
55         CP wn(cos(pi/j), f*sin(pi/j));
56         for(int k=0; k<n; k+=(j<<1)){

```

```

57             CP t(1, 0);
58             for(int l=0; l<j; ++l){
59                 CP cl=c[k+l], cr=t*c[j+k+l];
60                 c[k+l]=cl+cr;
61                 c[j+k+l]=cl-cr;
62                 t=t*wn;
63             }
64         }
65     }
66 }
67
68 int main(int argc, char** argv) {
69     // ios::sync_with_stdio(0);
70     init();
71     fft(a); //, fft(b);
72     for(int i=0; i<n; ++i) a[i]=a[i]*a[i];
73     fft(a, -1);
74     for(int i=0; i<=N+M; ++i) write(int(a[i].y/n/2 +
        0.5)), putchar(' ');
75     return 0;
76 }

```

6.18 FWT

6.19 NTT

6.20 NTT

```

1  /*
2  大致原理: 令 $P=998244353$ ,  $g=3$ ,  $g$ 的 $n$ 次幂模 $P$ 可得到 $P$ 种值, 称
       $g$ 为模 $P$ 的原根
3   $O(N\log N)$ 地将一个 $N$ 次多项式系数表达式转化成关于模 $P$ 的原根的 $g$ 
      的次幂的 $N$ 个点值
4  对两个多项式的点值做乘法, 得到新的点值表达式, 再 $O(N\log N)$ 
      地转回系数表达式
5  利用 $R$ 数组和“蝴蝶定理”可以将递归NTT过程转为循环过程, 优化
      复杂度的常数
6  写得很乱, 欢迎有空的同学来更新个好看点的
7  顺带一提求逆和分治里用的是细节不同的另外两种实现的NTT, 不
      过也都很乱
8  */
9  //洛谷P3803裸的多项式乘积, 在大数据下的表现可能反而不如
      FFT
10 #include <bits/stdc++.h>
11 using namespace std;
12 typedef long long ll;
13
14 const int MN = 3e6 + 5;
15 const int P = 998244353;
16
17 int qpow(ll a,int b){
18     ll ans=1;
19     for(;b;b>>=1){
20         if(b&1)
21             ans=ans*a%P;
22         a=a*a%P;}
23     return ans;}
24
25 int read(){
26     int k=0, f=1;

```

```

27 char c=getchar();
28 while(!isdigit(c)){ if(c=='-') f=-1; c=getchar();
    }
29 while(isdigit(c)) k= k*10 + c-48 , c=getchar();
30 return k*f;
31 }
32 void write(int x){
33     if(x<0){ putchar('-'); x=~(x-1); }
34     int s[20],top=0;
35     while(x){ s[++top]=x%10; x/=10; }
36     if(!top) s[++top]=0;
37     while(top) putchar(s[top--]+'0');
38 }
39
40 int a[MN],b[MN],c[MN];
41 int N,M,n,ln; //左多项式次数, 右多项式次数, fft次数及其
    位数
42 int R[MN]; //01串逆转后对应的下标
43 inline void calR(){ //通过NM初始化以上数据
44     n=1;
45     while(n<=N+M) n<=1, ++ln;
46     for(int i=1; i<n; ++i) R[i] = (R[i>>1] >>1) | ((i
        &1)<< ln-1);
47 }
48 inline void init(){
49     N=read(); M=read();
50     for(int i=0; i<=N; ++i) a[i]=read();
51     for(int i=0; i<=M; ++i) b[i]=read();
52     calR();
53 }
54 void ntt(int c[], int f=1){ //f取-1时是逆变换
55     for(int i=0; i<n; ++i) if(i<R[i]) swap(c[i], c[R[i]
        ]]);
56     for(int j=1; j<n; j<=1){
57         int j2=j<<1;
58         int rt=qpow(3,(P-1)/j2);
59         if(f==1) rt=qpow(rt, P-2); //逆变换时除以rt, 即
            乘以rt逆元
60         for(int k=0; k<n; k+=j2){
61             int t=1;
62             for(int l=0; l<j; ++l){
63                 int tl=c[k+l], tr=ll(t)*c[j+k+l]%P;
64                 c[k+l]=(tl+tr)%P;
65                 c[j+k+l]=(tl-tr)%P+P)%P;
66                 t=ll(t)*rt%P;
67             }
68         }
69     }
70 }
71
72 int main(int argc, char** argv) {
73     // ios::sync_with_stdio(0);
74     init();
75     ntt(a), ntt(b);
76     for(int i=0; i<=n; ++i) c[i]=ll(a[i])*b[i]%P;
77     ntt(c,-1);
78     ll mi=qpow(n,P-2);
79     for(int i=0; i<=N+M; ++i) write((c[i]*mi)%P),
        putchar(' ');
80     return 0;
81 }

```

6.21 NTT 分治求卷积

```

1 //大致原理是cdq分治套NTT求f_i = f和g的前i项卷积
2 //洛谷P4721求f_i = \sigma_{j=1}^i f_{i-j} * g_j
3 #include <bits/stdc++.h>
4 using namespace std;
5 typedef long long ll;
6
7 const int P = 998244353;
8 const int G = 3;
9 const int MN = 2e5 + 5;
10
11 inline int read(){
12     int k=0, f=1;
13     char c=getchar();
14     while(!isdigit(c)){ if(c=='-') f=-1; c=getchar();
15     }
16     while(isdigit(c)) k=k*10+c-48, c=getchar();
17     return k*f;
18 }
19
20 int qpow(ll a, int b){
21     ll ans=1ll;
22     for(;b>=1){
23         if(b&1)
24             ans=ans*a%P;
25         a=a*a%P;
26     }
27     return ans;
28 }
29
30 inline int inv(ll value){ return qpow(value, P-2); }
31
32 namespace NTT{
33     int R[MN]; //01串逆转后对应的下标
34     void calR(int len){ //计算len次多项式的R数组
35         int loglen = R[0] = 0;
36         while(1<<loglen < len) ++loglen;
37         for(int i=1; i<len; ++i) R[i] = (R[i>>1] >>1)
38             | ((i&1)<< loglen-1);
39     }
40
41     void ntt(int *a, int n, int f=1){ //f取-1时是逆变换
42         for(int i=0; i<n; ++i) if(R[i] < i) swap(a[i],
43             a[R[i]]);
44         int baseW = qpow(G, (P-1) / n);
45         if(f==1) baseW = inv(baseW);
46         for(int len = 2; len <= n; len <= 1){
47             int mid = len >>1;
48             int wn = qpow(baseW, n / len);
49             for(int *pos = a; pos != a+n; pos += len){
50                 int w = 1;
51                 for(int i=0; i<mid; ++i) {
52                     int x = pos[i], y = (ll)pos[mid + i]
53                         * w % P;
54                     pos[i] = ((ll)x + y) % P;
55                     pos[mid + i] = ((ll)x - y + P) % P;
56                     w = (ll)w * wn % P;
57                 }
58             }
59         }
60     }
61
62     void mult(int *a, int *b, int len){ //封装好的len
63         次多项式乘法, 答案存进a
64         calR(len);
65         ntt(a,len);
66         ntt(b,len);
67         for(int i=0; i<len; ++i) a[i] = (ll)a[i] * b[i]
68             % P;
69         ntt(a,len,-1);
70         int x = inv(len);

```



```

60     for(int i=0; i<len; ++i) a[i] = (ll)a[i] * x %
        P;
61     }
62 }
63 using namespace NTT;
64
65 int f[MN],g[MN],A[MN],B[MN]; //输出输入及其副本
66 int n; //多项式长度
67
68 void cdq(int l, int r){ //分治[l,r]
69     if(l==r) return;
70     int mid = l+r >> 1, len = 1;
71     while(len < r-l) len <<= 1;
72     cdq(l,mid);
73     for(int i=0; i<len; ++i) A[i] = 0, B[i] = 0;
74     for(int i=0; i<=mid-1; ++i) A[i] = f[l+i]; //1开始的
        当前区间的左半f
75     for(int i=0; i<r-1; ++i) B[i] = g[i+1]; //1开始的
        当前区间长度的g
76     mult(A,B,len); //卷积求出左半端的贡献
77     for(int i=mid+1; i<=r; ++i) f[i] = (f[i] + A[i-1]
        -1) % P;
78     cdq(mid+1,r);
79 }
80
81 int main(){
82     n = read();
83     for(int i=1; i<n; ++i) g[i] = read();
84     f[0] = 1; //题目给的初值
85     cdq(0,n-1);
86     for(int i=0; i<n; ++i) printf("%d ",f[i]);
87     return 0;
88 }

```

6.22 NTT 求多项式逆

```

1 //洛谷P4238倍增求模 $x^n$ 的多项式，即相乘后忽略 $n$ 次及更高次只
    剩1
2 #include <bits/stdc++.h>
3 using namespace std;
4 typedef long long ll;
5
6 const int P = 998244353;
7 const int G = 3;
8 const int MN = 4e5 + 5;
9
10 inline int read(){
11     int k=0, f=1;
12     char c=getchar();
13     while(!isdigit(c)){ if(c=='-') f=-1; c=getchar();
14     }
15     while(isdigit(c)) k=k*10+c-48, c=getchar();
16     return k*f;
17
18 int qpow(ll a, int b){
19     ll ans=1ll;
20     for(;b;b>>=1){
21         if(b&1)
22             ans=ans*a%P;
23         a=a*a%P;
24     }
25     return ans;
26
27 inline int inv(ll value){ return qpow(value, P-2); }

```

```

27 //namespace NTT{
28     int R[MN]; //01串逆转后对应的下标
29     void calcR(int len){ //计算len次多项式的R数组
30         int loglen = R[0] = 0;
31         while(1<<loglen < len) ++loglen;
32         for(int i=1; i<len; ++i) R[i] = (R[i>>1] >>1)
33             | ((i&1)<< loglen-1);
34     }
35     void ntt(int *a, int n, int f=1){ //f取-1时是逆变换
36         for(int i=0; i<n; ++i) if(R[i] < i) swap(a[i],
37             a[R[i]]);
38         int baseW = qpow(G, (P-1) / n);
39         for(int len = 2; len <= n; len <<= 1){
40             int mid = len >>1;
41             int wn = qpow(baseW, n / len);
42             for(int *pos = a; pos != a+n; pos += len){
43                 int w = 1;
44                 for(int i=0; i<mid; ++i){
45                     int x = pos[i], y = (ll)pos[mid + i]
46                         * w % P;
47                     pos[i] = ((ll)x + y) % P;
48                     pos[mid + i] = ((ll)x - y + P) % P;
49                     w = (ll)w * wn % P;
50                 }
51             }
52             if(f==1){
53                 int miv = inv( ll(n) );
54                 reverse(a+1,a+n);
55                 for(int i=0; i<n; ++i) a[i] = ll(a[i]) *
56                     miv % P;
57             }
58         }
59     }
60     void mult(int *a, int *b, int len){ //len次多项式
        乘法，答案存进a
61     calcR(len);
62     ntt(a,len);
63     ntt(b,len);
64     for(int i=0; i<len; ++i) a[i] = (ll)a[i] * b[i]
65         % P;
66     ntt(a,len,-1);
67     int x = inv(len);
68     for(int i=0; i<len; ++i) a[i] = (ll)a[i] * x %
69         P;
70 }
71 //}
72 //using namespace NTT;
73
74 int f[MN],g[MN],A[MN],B[MN]; //输出输入及其副本
75 int n; //多项式长度
76
77 void inv(int *a, int *b, int len){ //封装好的len次多项
    式求逆，答案存进b
78 if(len==1) return b[0] = inv(ll(a[0])), void();
79 inv(a, b, len+1 >>1);
80 int LEN = 1; //扩展到2的整数次幂的len
81 while(LEN < (len<<1)) LEN <<= 1;
82 calcR(LEN);
83 for(int i=0; i<len; ++i) A[i] = a[i];
84 for(int i=len; i<LEN; ++i) A[i] = 0;
85 ntt(A,LEN), ntt(b,LEN);
86 for(int i=0; i<LEN; ++i) b[i] = ll(2 - ll(A[i]) *
87     b[i] % P + P) % P * b[i] % P;
88 ntt(b,LEN,-1);
89 for(int i=len; i<LEN; ++i) b[i] = 0;

```

```

83 }
84
85 int main(){
86     n = read();
87     for(int i=0; i<n; ++i) g[i] = read();
88     inv(g,f,n);
89     for(int i=0; i<n; ++i) printf("%d ",f[i]);
90     return 0;
91 }

```

6.23 数值计算

6.23.1 辛普森

6.23.2 自适应辛普森

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  const double eps=1e-12;
4
5  /*
6   调用 asr(l,r,simpson(l,r))
7  */
8
9  inline double f(double x)
10 {
11     return x; //被积函数
12 }
13
14 double simpson(double l,double r)
15 {
16     double mid=(l+r)/2;
17     return (f(l)+4*f(mid)+f(r))*(r-l)/6;
18 }
19
20 double asr(double l,double r,double ans)
21 {
22     double mid=(l+r)/2;
23     double l1=simpson(l,mid),r1=simpson(mid,r);
24     if(fabs(l1+r1-ans)<eps) return l1+r1;
25     return asr(l,mid,l1)+asr(mid,r,r1);
26 }
27
28 int main()
29 {
30
31     return 0;
32 }

```

6.24 康拓展开

```

1  //contor展开
2  int bit[maxn];
3  void add(int x,int y)
4  {
5      for(;x<=n;x+=lowbit(x)) bit[x]+=y;
6  }
7
8  int sum(int x)
9  {
10     int res=0;
11     for(;x>0;x-=lowbit(x)) res+=bit[x];
12     return res;
13 }

```

```

14
15 int contor(vector& p)
16 {
17     int ans=0;
18     rre(i,p.size()-1,0)
19     {
20         add(p[i],1);
21         int cnt=sum(p[i]-1);
22         ans=(ans+cnt*fac[p.size()-i-1]%MOD)%MOD;
23     }
24     return ans+1;
25 }
26
27 //逆contor展开
28 int k,s[50005];
29 int tr[200005];
30
31 void up(int p) {tr[p]=tr[l1(p)]+tr[rr(p)];}
32
33 void build(int p=1,int l=1,int r=k)
34 {
35     if(l==r)
36     {
37         tr[p]=1;
38         return;
39     }
40     int m=mm(l,r);
41     build(l1(p),l,m);
42     build(rr(p),m+1,r);
43     up(p);
44 }
45
46 int ask(int cnt,int p=1,int l=1,int r=k)
47 {
48     if(l==r)
49     {
50         tr[p]=0;
51         return l;
52     }
53     int m=mm(l,r);
54     int ans=-1;
55     if(cnt<=tr[l1(p)]) ans=ask(cnt,l1(p),l,m);
56     else ans=ask(cnt-tr[l1(p)],rr(p),m+1,r);
57     up(p);
58     return ans;
59 }
60
61 vei recontor()
62 {
63     vei v(k+1);
64     re(i,1,k) v[i]=ask(s[i]+1);
65     return v;
66 }

```

6.25 卢卡斯定理

6.25.1 Lucas(循环或递归实现)

$$C_N^m \equiv C_{N \bmod p}^{m \bmod p} * C_{N/p}^{m/p} (\bmod p)$$

$$\binom{N}{m} \equiv \binom{N \bmod p}{m \bmod p} \cdot \binom{N/p}{m/p} (\bmod p)$$

可理解为将 N 和 m 表示为 p 进制数 (形如 $\sum N_i p^i$)，对每一位的 N_i 和 m_i 分别求组合数，再累乘，注意此处的 p 必须为质数

```

1 //洛谷P3807
2 #include <bits/stdc++.h>
3 using namespace std;
4 typedef long long ll;
5 const int MN = 5e6 + 5;
6
7 inline ll qpow(ll a, ll b, int P){ //a^b % P
8     ll ans=1;
9     for(; b>=1; a=a*a%P)
10         if(b&1) ans=ans*a%P;
11     return ans;
12
13 ll fct[MN], fi[MN]; //阶乘及其逆元
14 inline void init(int k, int P){ //打表模P的[1,k]阶乘及其逆元
15     fct[0]=1;
16     for(int i=1; i<=k; ++i) fct[i]=fct[i-1]*i%P;
17     if(k<P){
18         fi[k]=qpow(fct[k], P-2, P);
19         for(int i=k; i>=1; --i) fi[i-1]=fi[i]*i%P;
20     }else{ //k阶乘为0, 会把所有逆元都变成0, 应从P-1开始
21         fi[P-1]=qpow(fct[P-1], P-2, P);
22         for(int i=P-1; i>=1; --i) fi[i-1]=fi[i]*i%P;
23     }
24 }
25
26 inline int C(int N, int m, int P){ //C_N^m % P
27     if(m>N) return 0;
28     return fct[N]*fi[m]%P*fi[N-m]%P;
29 }
30
31 //ll lucas(int N, int m, int P){ //递归求C_N^m % P
32 // if(!m) return 1;
33 // return C(N%P, m%P, P)*lucas(N/P, m/P, P)%P;
34 //}
35
36 int lucas(int N, int m, int P){ //循环求C_N^m % P
37     ll rt=1;
38     while(N&& m)
39         (rt*=C(N%P, m%P, P))%=P,
40         N/=P, m/=P;
41     return rt;
42 }
43
44 void solve(){
45     int n, m, p; scanf("%d%d%d", &n, &m, &p);
46     init(n+m, p);
47     printf("%lld\n", lucas(n+m, m, p));
48 }
49
50 int main(int argc, char** argv){
51     int _; scanf("%d", &_); while(--)
52         solve();
53     return 0;
54 }

```

6.25.2 EXLucas(分块实现)

模数 P 不是质数时不能使用卢卡斯定理计算组合数！
 可对 P 做质因子分解，对各质因子分别求解组合数，得到同余方程组，再用 CRT 求解。

具体方法是将 P 分解成 Σp^k 的形式，再计算模 p^k 意义的阶乘。
 由于 $n > p$ 时暴力算 $n! \bmod p$ 为 0，因此计算阶乘时需先不断递归分块给阶乘除以质因子 p ，算完阶乘后再把除掉的 p 乘回去，才能计算出模 p^k 意义的组合数。

```

1 //洛谷P4720
2 #include <bits/stdc++.h>
3 using namespace std;
4 typedef long long ll;
5
6 inline ll qpow(ll a, ll b, int P){ //a^b % P, 此题中b可能爆int
7     ll ans=1;
8     for(; b>=1; a=a*a%P)
9         if(b&1) ans=ans*a%P;
10    return ans;
11
12 inline ll exgcd(ll a, ll b, ll &x, ll &y){
13     if(!b) return x=1, y=0, a;
14     ll g=exgcd(b, a%b, x, y);
15     ll z=x; x=y; y=z-a/b*y;
16     return g;
17
18 inline ll exinv(int a, int P){ //用exgcd求a模P的逆元
19     ll x, y;
20     if(exgcd(a, P, x, y)!=1) return -1;
21     else return (x%P+P)%P;
22 }
23
24 inline int g(ll n, int p){ //n!中质因子p的次数
25     if(n<p) return 0;
26     return n/p+g(n/p, p);
27 }
28
29 int f(ll n, int p, int pk){ //n!/(p^x) % pk, 其中x=g(n, p)
30     if(n==0) return 1;
31     ll s=1, s2=1; //<=pk的分块乘积, >pk的块外乘积
32     for(ll i=1; i<=pk; ++i)
33         if(i%p) s=s*i%pk;
34     s=qpow(s, n/pk, pk);
35     for(ll i=n/pk*pk; i<=n; ++i)
36         if(i%p) s2=i%pk*s2%pk;
37     return f(n/p, p, pk)*s%pk*s2%pk;
38 }
39
40 inline ll c(ll N, ll m, int p, int pk){ //C_m^N % (p^k)
41     ll rt=f(N, p, pk);
42     (rt*=qpow(f(m, p, pk), pk/p*(p-1)-1, pk))%=pk;
43     // (rt*=exinv(f(m, p, pk), pk))%=pk;
44     (rt*=qpow(f(N-m, p, pk), pk/p*(p-1)-1, pk))%=pk;
45     // (rt*=exinv(f(N-m, p, pk), pk))%=pk;
46     (rt*=qpow(p, g(N, p)-g(m, p)-g(N-m, p), pk))%=pk;
47     return rt;
48 }
49
50 inline ll crt(ll ai, int p, int pk, int P){ //x%(p^k)=ai, pk乘积=P
51     return ai*(P/pk)%P*exinv(P/pk, pk)%P;
52 // return ai*(P/pk)%P*qpow(P/pk, pk/p*(p-1)-1, pk)%P;
53 }
54
55 int exlucas(ll N, ll m, int P){ //C_m^N % P
56     ll rt=0, P2=P;
57     int ed=sqrt(P)+1;
58     for(int p=2; p<=ed; ++p){

```



```

59     int pk=1;
60     while(P2%p==0) pk*=p, P2/=p;
61     if(pk>1) (rt+=crt(c(N,m,p,pk),p,pk,P))%=P;
62 }
63 if(P2>1) (rt+=crt(c(N,m,P2,P2),P2,P2,P))%=P;
64 return rt;
65 }
66
67 int main(int argc, char** argv){
68     ios::sync_with_stdio(0);
69     ll N,m,P; cin>>N>>m>>P;
70     cout<<exlucas(N,m,P);
71     return 0;
72 }

```

6.26 博弈论

6.26.1 SG 函数

```

1  #include <bits/stdc++.h>
2  //Gym 101128 G
3  using namespace std;
4  const int N=1005;
5
6  int p,k;
7  vector<int> v(N);
8  int sg[N];
9  bool vis[N];
10
11 void gao(int n)
12 {
13     memset(sg,0,sizeof(sg));
14     for(int i=1;i<=n;++i)//由小到大枚举
15     {
16         memset(vis,0,sizeof(vis));
17         for(int j=0;j<=min(i,k);++j)
18         {
19             if(i==j) continue;
20             int tmp=i-j-v[i-j];//枚举子状态
21             if(tmp>=0)
22                 vis[sg[tmp]]=1;
23         }
24         for(int j=0;;j++)
25             if(!vis[j])
26             {
27                 sg[i]=j;
28                 break;
29             }
30     }
31 }
32
33 int main()
34 {
35     int ans=0;
36     scanf("%d%d",&p,&k);
37     for(int i=1,n;i<=p;++i)
38     {
39         scanf("%d",&n);
40         for(int j=1;j<=n;++j)
41             scanf("%d",&v[j]);
42         gao(n);
43         ans^=sg[n];//多堆求每一堆的异或值即可
44     }
45     if(ans==0) puts("Bob will win.");//先手必败
46     else puts("Alice can win.");

```

```

47     return 0;
48 }

```

7 其他

7.1 快读快写

```

1  inline int read()
2  {
3      char ch=getchar();int s=0,w=1;
4      while(ch<48||ch>57){if(ch=='-')w=-1;ch=getchar();}
5      while(ch>=48&&ch<=57){s=(s<<1)+(s<<3)+ch-48;ch=
6          getchar();}
7      return s*w;
8  }
9
10 inline void write(int x)
11 {
12     if(x<0)putchar('-'),x=-x;
13     if(x>9)write(x/10);
14     putchar(x%10+48);
15 }

```

7.2 高精度

```

1  #include <bits/stdc++.h>
2  #define MAXN 9999
3  #define MAXSIZE 1000
4  #define DLEN 4
5  using namespace std;
6
7  class BigNum
8  {
9  private:
10     int a[MAXSIZE];
11     int len;
12 public:
13     BigNum(){ len = 1;memset(a,0,sizeof(a)); }
14     void XD();
15     BigNum(const int);
16     BigNum(const long long int);
17     BigNum(const char*);
18     BigNum(const string &);
19     BigNum(const BigNum &);
20     BigNum &operator = (const BigNum &);
21     BigNum &operator = (const int &);
22     BigNum &operator = (const long long int &);
23
24     friend istream& operator >> (istream&, BigNum&);
25     friend ostream& operator << (ostream&, BigNum&);
26
27     template<typename T> BigNum operator << (const T
28         &) const;
29     template<typename T> BigNum operator >> (const T
30         &) const;
31
32     BigNum operator + (const BigNum &) const;
33     BigNum operator - (const BigNum &) const;
34     BigNum operator * (const BigNum &) const;
35     bool operator > (const BigNum& b) const;
36     bool operator < (const BigNum& b) const;
37     bool operator == (const BigNum& b) const;

```

```

36  template<typename T> BigNum operator / (const T &
    const;
37  template<typename T> BigNum operator ^ (const T &
    const;
38  template<typename T> T operator % (const T &)
    const;
39
40  template<typename T> BigNum operator + (const T& b
    ) const {BigNum t = b; t = *this + t; return
    t;};
41  template<typename T> BigNum operator - (const T& b
    ) const {BigNum t = b; t = *this - t; return
    t;};
42  template<typename T> BigNum operator * (const T& b
    ) const {BigNum t = b; t = (*this) * t;
    return t;};
43  template<typename T> bool operator < (const T& b)
    const {BigNum t = b; return ((*this) < t);};
44  template<typename T> bool operator > (const T& b)
    const {BigNum t = b; return ((*this) > t);};
45  template<typename T> bool operator == (const T& b)
    const {BigNum t = b; return ((*this) == t);};
46
47  bool operator <= (const BigNum& b) const {return
    (*this) < b || (*this) == b;};
48  bool operator >= (const BigNum& b) const {return
    (*this) > b || (*this) == b;};
49  bool operator != (const BigNum& b) const {return
    !((*this) == b);};
50
51  template<typename T> bool operator >= (const T& b)
    const {BigNum t = b; return !((*this) < t);};
52  template<typename T> bool operator <= (const T& b)
    const {BigNum t = b; return !((*this) > t);};
53  template<typename T> bool operator != (const T& b)
    const {BigNum t = b; return !((*this) == t)
    ;};
54
55  BigNum& operator += (const BigNum& b) {*this = *
    this + b; return *this;};
56  BigNum& operator -= (const BigNum& b) {*this = *
    this - b; return *this;};
57  BigNum& operator *= (const BigNum& b) {*this = *
    this * b; return *this;};
58  template<typename T> BigNum& operator /= (const T&
    b) {*this = *this/b; return *this;};
59  template<typename T> BigNum& operator %= (const T&
    b) {*this = *this%b; return *this;};
60  template<typename T> BigNum& operator += (const T&
    b) {*this = *this+b; return *this;};
61  template<typename T> BigNum& operator -= (const T&
    b) {*this = *this-b; return *this;};
62  template<typename T> BigNum& operator *= (const T&
    b) {*this = *this*b; return *this;};
63  template<typename T> BigNum& operator ^= (const T&
    b) {*this = *this^b; return *this;};
64
65  BigNum operator ++ (int) {BigNum t = *this; *this
    += 1; return t;};
66  BigNum operator -- (int) {BigNum t = *this; *this
    -= 1; return t;};
67  BigNum& operator -- () {*this -= 1; return *this;};
68  BigNum& operator ++ () {*this += 1; return *this;};
69
70  template<typename T> BigNum& operator <= (const T
    & b) {*this = *this <= b; return *this;};
71  template<typename T> BigNum& operator >= (const T
    & b) {*this = *this >= b; return *this;};
72
73  template<typename T> BigNum friend operator + (
    const T& a, const BigNum& b) {BigNum t = a; t
    = t + a; return t;};
74  template<typename T> BigNum friend operator - (
    const T& a, const BigNum& b) {BigNum t = a; t
    = t - b; return t;};
75  template<typename T> BigNum friend operator * (
    const T& a, const BigNum& b) {BigNum t = a; t
    = t * b; return t;};
76  template<typename T> friend bool operator < (const
    T& a, const BigNum& b) {return b > a;};
77  template<typename T> friend bool operator > (const
    T& a, const BigNum& b) {return b < a;};
78  template<typename T> friend bool operator <= (
    const T& a, const BigNum& b) {return b >= a;};
79  template<typename T> friend bool operator >= (
    const T& a, const BigNum& b) {return b <= a;};
80  template<typename T> friend bool operator == (
    const T& a, const BigNum& b) {return b == a;};
81  template<typename T> friend bool operator != (
    const T& a, const BigNum& b) {return b != a;};
82
83  void print();
84  int Size();
85  int the_first();
86  int the_last();
87  int to_int();
88  long long int to_long();
89  string to_String();
90  };
91
92  BigNum::BigNum(const int b)
93  {
94      int c,d = b;
95      len = 0;
96      memset(a,0,sizeof(a));
97      while(d > MAXN){
98          c = d - (d / (MAXN+1)) * (MAXN+1);
99          d = d / (MAXN+1);
100         a[len++] = c;
101     }
102     a[len++] = d;
103 }
104 BigNum::BigNum(const long long int b)
105 {
106     long long int c,d = b;
107     len = 0;
108     memset(a,0,sizeof(a));
109     while(d > MAXN){
110         c = d - (d / (MAXN+1)) * (MAXN+1);
111         d = d / (MAXN+1);
112         a[len++] = c;
113     }
114     a[len++] = d;
115 }
116 BigNum::BigNum(const string& s)
117 {
118     int t,k,index,l,i;
119     memset(a,0,sizeof(a));
120     l = s.size();
121     len = l/DLEN;

```

```

122     if(1%DLEN)
123         len++;
124     index = 0;
125     for(i = l-1; i >= 0 ;i -= DLEN){
126         t = 0;
127         k = i-DLEN+1;
128         if(k < 0) k = 0;
129         for(int j = k; j <= i; j++)
130             t = t*10 + s[j] - '0';
131         a[index++] = t;
132     }
133 }
134 BigNum::BigNum(const char* s)
135 {
136     int t,k,index,l,i;
137     memset(a,0,sizeof(a));
138     l = strlen(s);
139     len = l/DLEN;
140     if(1%DLEN)
141         len++;
142     index = 0;
143     for(i = l-1; i >= 0; i -= DLEN){
144         t = 0;
145         k = i - DLEN + 1;
146         if(k < 0) k = 0;
147         for(int j = k; j <= i; j++)
148             t = t*10 + s[j] - '0';
149         a[index++] = t;
150     }
151 }
152 BigNum::BigNum(const BigNum & b) : len(b.len)
153 {
154     memset(a,0,sizeof(a));
155     for(int i = 0 ; i < len ; i++)
156         a[i] = b.a[i];
157 }
158 BigNum & BigNum::operator = (const BigNum& n)
159 {
160     len = n.len;
161     memset(a,0,sizeof(a));
162     for(int i = 0 ; i < len ; i++)
163         a[i] = n.a[i];
164     return *this;
165 }
166 BigNum & BigNum::operator = (const int& num)
167 {
168     BigNum t(num);
169     *this = t;
170     return *this;
171 }
172 BigNum & BigNum::operator = (const long long int& num
173 )
174 {
175     BigNum t(num);
176     *this = t;
177     return *this;
178 }
179 istream& operator >> (istream & in, BigNum & b)
180 {
181     char ch[MAXSIZE*4];
182     int i = -1;
183     in>>ch;
184     int l = strlen(ch);
185     int cnt = 0, sum = 0;
186     for(i = l-1; i >= 0; ){

```

```

186         sum = 0;
187         int t = 1;
188         for(int j = 0; j < 4 && i >= 0; j++,i--,t *=
189             10)
190             sum += (ch[i] - '0') * t;
191         b.a[cnt] = sum;
192         cnt++;
193     }
194     b.len = cnt;
195     return in;
196 }
197 ostream& operator << (ostream& out, BigNum& b)
198 {
199     int i;
200     cout << b.a[b.len - 1];
201     for(i = b.len - 2 ; i >= 0 ; i--){
202         cout.width(DLEN);
203         cout.fill('0');
204         cout << b.a[i];
205     }
206     return out;
207 }
208
209 template<typename T> BigNum BigNum::operator << (
210     const T& b) const
211 {
212     T temp = 1;
213     for(int i = 0; i < b; i++)
214         temp *= 2;
215     BigNum t = (*this) * temp;
216     return t;
217 }
218 template<typename T> BigNum BigNum::operator >> (
219     const T& b) const
220 {
221     T temp = 1;
222     for(int i = 0; i < b; i++)
223         temp *= 2;
224     BigNum t = (*this) / temp;
225     return t;
226 }
227
228 BigNum BigNum::operator + (const BigNum& b) const
229 {
230     BigNum t(*this);
231     int i,big;
232     big = b.len > len ? b.len : len;
233     for(i = 0 ; i < big ; i++){
234         t.a[i] += b.a[i];
235         if(t.a[i] > MAXN){
236             t.a[i + 1]++;
237             t.a[i] -= MAXN + 1;
238         }
239     }
240     if(t.a[big] != 0)
241         t.len = big + 1;
242     else
243         t.len = big;
244     return t;
245 }
246
247 BigNum BigNum::operator - (const BigNum& b) const
248 {
249     int i,j,big;
250     bool flag;

```

```

248     BigNum t1,t2;
249     if(*this>b){
250         t1 = *this;
251         t2 = b;
252         flag = 0;
253     }
254     else{
255         t1 = b;
256         t2 = *this;
257         flag = 1;
258     }
259     big = t1.len;
260     for(i = 0 ; i < big ; i++){
261         if(t1.a[i] < t2.a[i]){
262             j = i + 1;
263             while(t1.a[j] == 0)
264                 j++;
265             t1.a[j--]--;
266             while(j > i)
267                 t1.a[j--] += MAXN;
268             t1.a[i] += MAXN + 1 - t2.a[i];
269         }
270         else
271             t1.a[i] -= t2.a[i];
272     }
273     t1.len = big;
274     while(t1.a[t1.len - 1] == 0 && t1.len > 1){
275         t1.len--;
276         big--;
277     }
278     if(flag)
279         t1.a[big-1] = 0-t1.a[big-1];
280     return t1;
281 }
282
283 BigNum BigNum::operator * (const BigNum& b) const
284 {
285     BigNum ret;
286     int i,j,up;
287     int temp,temp1;
288     for(i = 0 ; i < len ; i++){
289         up = 0;
290         for(j = 0 ; j < b.len ; j++){
291             temp = a[i] * b.a[j] + ret.a[i + j] + up;
292             if(temp > MAXN){
293                 temp1 = temp - temp / (MAXN + 1) * (MAXN
294                     + 1);
295                 up = temp / (MAXN + 1);
296                 ret.a[i + j] = temp1;
297             }
298             else{
299                 up = 0;
300                 ret.a[i + j] = temp;
301             }
302             if(up != 0) ret.a[i + j] = up;
303         }
304         ret.len = i + j;
305         while(ret.a[ret.len - 1] == 0 && ret.len > 1)
306             ret.len--;
307         return ret;
308     }
309     template<typename T> BigNum BigNum::operator / (const
310         T& b) const
311 {

```

```

311     BigNum ret;
312     T i,down = 0;
313     for(i = len - 1 ; i >= 0 ; i--){
314         ret.a[i] = (a[i] + down * (MAXN + 1)) / b;
315         down = a[i] + down * (MAXN + 1) - ret.a[i] * b
316             ;
317     }
318     ret.len = len;
319     while(ret.a[ret.len - 1] == 0 && ret.len > 1)
320         ret.len--;
321     return ret;
322 }
323 template<typename T> T BigNum::operator % (const T& b
324     ) const
325 {
326     T i,d=0;
327     for (i = len-1; i>=0; i--){
328         d = ((d * (MAXN+1))% b + a[i])% b;
329     }
330     return d;
331 }
332 template<typename T> BigNum BigNum::operator^(const T
333     & n) const
334 {
335     BigNum t,ret(1);
336     int i;
337     if(n < 0) return 0;
338     if(n == 0)
339         return 1;
340     if(n == 1)
341         return *this;
342     int m = n;
343     while(m > 1){
344         t = * this;
345         for(i = 1; (i<<1) <= m;i <= 1)
346             t = t*t;
347         m-=i;
348         ret=ret*t;
349         if(m == 1) ret = ret * (*this);
350     }
351     return ret;
352 }
353 bool BigNum::operator > (const BigNum& b) const
354 {
355     int tot;
356     if(len > b.len)
357         return true;
358     else if(len == b.len){
359         tot = len - 1;
360         while(a[tot] == b.a[tot] && tot >= 0)
361             tot--;
362         if(tot >= 0 && a[tot] > b.a[tot])
363             return true;
364         else
365             return false;
366     }
367     else
368         return false;
369 }
370
371 bool BigNum::operator < (const BigNum& b) const
372 {

```

```

373 int tot;
374 if(len > b.len)
375     return false;
376 else if(len == b.len){
377     tot = len - 1;
378     while(a[tot] == b.a[tot] && tot >= 0)
379         tot--;
380     if(tot >= 0 && a[tot] > b.a[tot])
381         return false;
382     else
383         return true;
384 }
385 else
386     return true;
387 }
388
389 bool BigNum::operator == (const BigNum& b) const
390 {
391     int tot = len-1;
392     if(len != b.len)
393         return false;
394     while(a[tot] == b.a[tot] && tot >= 0)
395         tot--;
396     if(tot < 0)
397         return true;
398     return false;
399 }
400
401 void BigNum::print()
402 {
403     int i;
404     cout << a[len - 1];
405     for(i = len-2; i >= 0; i--){
406         cout.width(DLEN);
407         cout.fill('0');
408         cout << a[i];
409     }
410     cout << endl;
411 }
412 int BigNum::Size()
413 {
414     int t = a[len-1], cnt = 0;
415     while(t){ t /= 10; cnt++; }
416     cnt += (len-1)*4;
417     return cnt;
418 }
419 int BigNum::the_first()
420 {
421     int t = a[len-1];
422     while(t > 10){ t /= 10; }
423     return t;
424 }
425 int BigNum::the_last()
426 {
427     int t = a[0];
428     return t%10;
429 }
430 int BigNum::to_int()
431 {
432     int i, num;
433     num = a[len-1];
434     for(i = len-2; i >= 0; i--){
435         num = num*(MAXN+1) + a[i];
436     }
437     return num;

```

```

438 long long int BigNum::to_long()
439 {
440     int i;
441     long long int num;
442     num = a[len-1];
443     for(i = len-2; i >= 0; i--){
444         num = num*(MAXN+1) + a[i];
445     }
446     return num;
447 }
448 int main()
449 {
450     BigNum a,b;
451     cin>>a>>b;
452     (a+b).print();
453     return 0;
454 }

```

7.3 进制转换

```

1 #include <bits/stdc++.h>
2 using namespace std;
3
4 /*
5 2019 ICPC Asia Yinchuan Regional I
6 x进制转换成y进制 x,y<=52
7 */
8
9 int x,y,n;
10 char s[1000005];
11 vector<char> t;
12
13 inline int get(char x)
14 {
15     if(x>='0'&&x<='9') return x-'0';
16     if(x>='A'&&x<='Z') return x-'A'+10;
17     return x-'a'+36;
18 }
19
20 inline char gao(int x)
21 {
22     if(x<10) return x+'0';
23     if(x<36) return x-10+'A';
24     return x-36+'a';
25 }
26
27 int main()
28 {
29     scanf("%d%d%s",&x,&y,&s);
30     int i=0;n=strlen(s);
31     while(i<n)
32     {
33         int r=0,cur;
34         for(int j=i;j<n;j++)
35         {
36             cur=r*x+get(s[j]);
37             s[j]=gao(cur/y);
38             r=cur%y;
39         }
40         t.push_back(gao(r));
41         while(s[i]=='0') i++;
42     }
43     reverse(t.begin(),t.end());
44     for(auto c:t) printf("%c",c);

```

```
45     return 0;  
46 }
```

7.4 约瑟夫环

7.5 悬线法

7.6 蔡勒公式

7.7 三角公式

7.8 海伦公式

7.9 匹克定理

7.10 组合计数

7.10.1 计数原理

7.10.2 卡特兰数

7.10.3 Polya

7.10.4 二项式反演公式

7.10.5 斯特林反演公式

7.10.6 组合数恒等式