

小小青蛙听风就是雨

HiedanoAkyuu、Oneman233、KR12138

2019 年 10 月 31 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 | 1 | | |
| 1.5 后缀数组 | 1 | | |
| 1.5.1 倍增 SA | 1 | | |
| 1.5.2 DC3 | 1 | | |
| 1.6 回文自动机 | 1 | | |
| 1.7 AC 自动机 | 1 | | |
| 1.7.1 多模匹配 | 1 | | |
| 1.7.2 自动机上 DP | 2 | | |
| 1.8 后缀自动机 | 2 | | |
| 2 计算几何 | 2 | | |
| 2.1 二维几何 | 2 | | |
| 2.2 三维几何 | 4 | | |
| 3 图论 | 4 | | |
| 3.1 最短路 | 4 | | |
| 3.1.1 Dijkstra | 4 | | |
| 3.1.2 SPFA | 4 | | |
| 3.1.3 Floyd | 4 | | |
| 3.1.4 负环 | 4 | | |
| 3.1.5 差分约束 | 4 | | |
| 3.2 最小生成树 | 4 | | |
| 3.2.1 Prim | 4 | | |
| 3.2.2 Kruskal | 4 | | |
| 3.2.3 最小生成树计数 | 4 | | |
| 3.2.4 次小生成树 | 4 | | |
| 3.2.5 最小乘积生成树 | 4 | | |
| 3.3 树的直径 | 4 | | |
| 3.4 LCA | 4 | | |
| 3.4.1 Tarjan 离线 | 4 | | |
| 3.4.2 倍增 LCA | 4 | | |
| 3.5 无向图与有向图联通性 | 4 | | |
| 3.5.1 割点 | 4 | | |
| 3.5.2 桥 | 4 | | |
| 3.5.3 e-DCC | 4 | | |
| 3.5.4 v-DCC | 4 | | |
| 3.5.5 SCC | 4 | | |
| 3.5.6 2-SAT | 4 | | |
| 3.5.7 支配树 | 4 | | |
| 3.6 二分图 | 4 | | |
| 3.6.1 最大匹配-匈牙利 | 4 | | |
| 3.6.2 带权匹配-KM | 4 | | |
| 3.7 网络流 | 4 | | |
| 3.7.1 最大流-Dinic | 4 | | |
| 3.7.2 最小费用最大流-Dij+Dinic | 4 | | |
| 3.7.3 上下界流 | 4 | | |
| 3.8 欧拉路 | 4 | | |
| 3.9 Prufer 序列 | 4 | | |
| 4 数据结构 | 4 | | |
| 4.1 树状数组 | 4 | | |
| 4.2 线段树 | 4 | | |
| 4.2.1 多操作线段树 | 4 | | |
| 4.2.2 吉司机线段树 | 4 | | |
| 4.2.3 扫描线 | 4 | | |
| 4.3 RMQ | 4 | | |
| 4.3.1 一维 | 4 | | |
| 4.3.2 二维 | 4 | | |
| 4.4 树链剖分 | 4 | | |
| 4.4.1 点剖分 | 4 | | |
| 4.4.2 边剖分 | 4 | | |
| 4.5 平衡树 | 4 | | |
| 4.5.1 Treap | 4 | | |
| 4.5.2 Splay | 4 | | |
| 4.6 动态树 | 4 | | |
| 4.7 主席树 | 4 | | |
| 4.8 树套树 | 4 | | |
| 4.8.1 线段树套 Treap | 4 | | |
| 4.8.2 线段树套树状数组 | 4 | | |
| 4.9 K-D Tree | 4 | | |
| 4.10 分治 | 4 | | |
| 4.10.1 CDQ | 4 | | |
| 4.10.2 点分治 | 4 | | |
| 4.10.3 dsu on tree | 4 | | |
| 4.10.4 整体二分 | 4 | | |
| 4.11 分块 | 4 | | |
| 4.11.1 普通分块 | 4 | | |
| 4.11.2 莫队 | 4 | | |
| 4.12 线性基 | 4 | | |
| 4.13 珂朵莉树 | 4 | | |
| 4.14 跳舞链 | 4 | | |
| 5 动态规划 | 4 | | |
| 5.1 SOS | 4 | | |
| 5.2 动态 DP | 4 | | |
| 5.3 插头 DP | 4 | | |
| 6 数学 | 4 | | |
| 6.1 矩阵类 | 4 | | |
| 6.2 质数筛 | 4 | | |
| 6.2.1 埃筛 | 4 | | |
| 6.2.2 线筛 | 4 | | |
| 6.3 质数判定 | 4 | | |
| 6.3.1 Miller Rabin | 4 | | |
| 6.4 质因数分解 | 4 | | |
| 6.4.1 Pollard-Rho | 4 | | |
| 6.5 逆元 | 4 | | |
| 6.5.1 EX-GCD 求逆元 | 4 | | |
| 6.5.2 线性筛逆元 | 4 | | |
| 6.5.3 阶乘逆元 | 4 | | |
| 6.6 欧拉函数 | 4 | | |
| 6.6.1 欧拉线筛 | 4 | | |
| 6.6.2 求单个数的欧拉函数 | 4 | | |
| 6.6.3 欧拉降幂 | 4 | | |
| 6.6.4 一般积性函数求法 | 4 | | |
| 6.7 EX-GCD | 4 | | |
| 6.8 CRT | 4 | | |
| 6.9 N 次剩余 | 4 | | |
| 6.10 数论分块 | 4 | | |
| 6.11 高斯消元 | 4 | | |
| 6.11.1 普通消元 | 4 | | |
| 6.11.2 异或方程组消元 | 4 | | |
| 6.12 莫比乌斯反演 | 4 | | |
| 6.12.1 莫比乌斯函数 | 4 | | |
| 6.12.2 杜教筛 | 4 | | |
| 6.12.3 洲阁筛 | 4 | | |
| 6.12.4 min25 筛 | 4 | | |
| 6.13 BSGS | 4 | | |
| 6.14 FFT | 4 | | |
| 6.15 FWT | 4 | | |
| 6.16 NTT | 4 | | |
| 6.17 数值计算 | 4 | | |
| 6.17.1 辛普森 | 4 | | |
| 6.17.2 自适应辛普森 | 4 | | |
| 6.18 康拓展开 | 4 | | |
| 6.19 卢卡斯定理 | 4 | | |

7 其他5

7.1 快读快写5

7.2 约瑟夫环5

7.3 悬线法5

7.4 蔡勒公式5

7.5 三角公式5

7.6 海伦公式5

7.7 匹克定理5

7.8 组合计数5

7.8.1 计数原理5

7.8.2 卡特兰数5

7.8.3 Polya5

7.8.4 二项式反演公式5

7.8.5 斯特林反演公式5

7.8.6 组合数恒等式5

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 }
15
16 void manacher()
17 {
18     int pos=0,mx=0;
19     for(int i=1;i<=cnt;i++)
20     {

```

```

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

```

1.4 串的最小表示

1.5 后缀数组

1.5.1 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
    [MAXN],height[MAXN];
16 char str[MAXN];
17
18 inline bool cmp(int *r,int a,int b,int l){return r[a
    ]==r[b]&&r[a+l]==r[b+l];}
19
20 void da(const char r[],int sa[],int n,int m)
21 {
22     int i,j,p,*x=wa,*y=wb,*t;
23     for(i=0;i<m;i++) wz[i]=0;
24     for(i=0;i<n;i++) wz[x[i]=r[i]]++;
25     for(i=1;i<m;i++) wz[i]+=wz[i-1];
26     for(i=n-1;i>=0;i--) sa[--wz[x[i]]]=i;
27     for(j=1,p=1;p<n;j*=2,m=p)
28     {
29         for(p=0,i=n-j;i<n;i++) y[p++]=i;
30         for(i=0;i<n;i++) if(sa[i]>=j) y[p++]=sa[i]-j;
31         for(i=0;i<n;i++) wv[i]=x[y[i]];
32         for(i=0;i<m;i++) wz[i]=0;
33         for(i=0;i<n;i++) wz[wv[i]]++;
34         for(i=1;i<m;i++) wz[i]+=wz[i-1];
35         for(i=n-1;i>=0;i--) sa[--wz[wv[i]]]=y[i];
36         for(t=x,x=y,y=t,p=1,x[sa[0]]=0,i=1;i<n;i++)
37             x[sa[i]]=cmp(y,sa[i-1],sa[i],j)?p-1:p++;
38     }
39 }
40

```

```

41 void calheight(const char *r,int *sa,int n)
42 {
43     int i,j,k=0;
44     for(i=1;i<=n;i++) Rank[sa[i]]=i;
45     for(i=0;i<n;height[Rank[i++]]=k)
46         for(k?k--:0,j=sa[Rank[i]-1];r[i+k]==r[j+k];k++);
47     for(int i=n;i>=1;--i) sa[i]++,Rank[i]=Rank[i-1];
48 }
49
50 int main()
51 {
52     scanf("%s",str);
53     int len=strlen(str);
54     da(str,sa,len+1,130); //字符的值域
55     calheight(str,sa,len);
56     for(int i=1;i<=len;i++)
57         printf("sa[%d] %d\n",i,sa[i]);
58     for(int i=1;i<=len;i++)
59         printf("Rank[%d] %d\n",i,Rank[i]);
60     for(int i=1;i<=len;i++)
61         printf("height[%d] %d\n",i,height[i]);
62     return 0;
63 }

```

1.5.2 DC3

1.6 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
6   串
7   len[x]: 以x结尾的最长回文子串长度
8   diff[x]: 与"以x结尾的最长回文子串"本质不同的子串个
9   数
10  same[x]: 与"以x结尾的最长回文子串"本质相同的子串个数
11  (注意上面两个完全相反)
12  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
13  之后变成的回文子串编号
14  s[x]: 第x次添加的字符, s数组即原字符串
15  tot: 总节点个数, 节点编号由0到tot-1
16  last: 最后一个新建节点的编号
17  cur: 当前节点在PAM上的父亲编号
18 */
19
20 #define int long long
21 using namespace std;
22 const int N=1e6+5;
23
24 struct PAM
25 {
26     int tot,last,n,cur;
27     int fail[N],len[N],same[N],diff[N],son[N][26];
28     char s[N];
29     int get(int p,int x)
30     {
31         while(s[x-len[p]-1]!=s[x])
32             p=fail[p];
33         return p;
34     }
35     int newnode(int x)
36     {
37         len[tot]=x;
38     }
39 }
40

```

```

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[diff[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
79 signed main()
80 {
81     pam.build();
82     dfs(0,0);//even string
83     dfs(1,0);//odd string
84     printf("%lld",ans);
85     return 0;
86 }

```

1.7 AC 自动机

1.7.1 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
53 int acatm_match(int head,char s[],int len)
54 {
55     int p=head,ret=0;
56     for(int i=0;i<len;i++)
57     {
58         int c=(int)s[i]-'a';
59         p=nxt[p][c];
60         for(int tp=p;tp;tp=fail[tp])
61             if(id[tp]) ret++;
62     }
63     return ret;
64 }

```

1.7.2 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
    [55][2005],from[55][2005];
10 char s[105][15];
11 string dps[55][2005];
12
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
22 void trie_insert(int head,char s[],int len,int idx)
23 {
24     int p=head;
25     for(int i=0;i<len;i++)
26     {
27         int c=s[i]-'a';
28         if(!nxt[p][c]) nxt[p][c]=++sz;
29         p=nxt[p][c];
30     }
31     cnt[p]+=idx;
32 }
33
34 void acatm_build(int head)
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]][i];
46                 cnt[nxt[p][i]]+=cnt[fail[nxt[p][i]]];
47                 q.push(nxt[p][i]);
48             }
49         else
50             nxt[p][i]=p==head?head:nxt[fail[p]][i];
51     }
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
60 void solve()
61 {
62     clear();
63     scanf("%d%d",&n,&m);
64     for(int i=0;i<m;i++)
65         scanf("%s",s[i]);
66     for(int i=0;i<m;i++)

```

```

67     {
68         int x;
69         scanf("%d",&x);
70         trie_insert(hd,s[i],strlen(s[i]),x);
71     }
72     acatm_build(hd);
73
74     for(int i=0;i<=n;i++)
75         for(int j=0;j<=sz;j++)
76             dps[i][j]=string("");
77     int ans=0;
78     string anss;
79     queue<pair<int,int> > q;
80     dp[0][1]=0;
81     for(int i=0;i<=n;i++)
82         for(int j=1;j<=sz;j++)
83             for(int k=0;k<26;k++)
84                 if(dp[i][j]+cnt[nxt[j][k]]>dp[i+1][nxt[j][k]]
85                    ||dp[i][j]+cnt[nxt[j][k]]==dp[i+1][nxt[j][k]]
86                    &&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('a'+k);
90                     dp[i+1][nxt[j][k]]=dp[i][j]+cnt[nxt[j][k]];
91                 }
92     for(int i=0;i<=n;i++)
93         for(int j=1;j<=sz;j++)
94             if(dp[i][j]>ans||dp[i][j]==ans&&scmp(dps[i][j],anss))
95             {
96                 ans=dp[i][j];
97                 anss=dps[i][j];
98             }
99     for(int i=0;i<anss.length();i++)
100         printf("%c",anss[i]);
101     printf("\n");
102 }
103
104 int main()
105 {
106     int _;
107     scanf("%d",&_);
108     while(_--) solve();
109     return 0;
110 }

```

1.8 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){return {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&&
    sign(x)>=0);}
36 db dot(P p){return x*p.x+y*p.y;}
37 db det(P p){return x*p.y-y*p.x;}
38 P rot(db an){return {x*cos(an)-y*sin(an),x*sin(an)
    +y*cos(an)};}
39 };
40
41 //For segment
42 #define cross(p1,p2,p3) ((p2.x-p1.x)*(p3.y-p1.y)-(p3.
    x-p1.x)*(p2.y-p1.y))
43 #define crossOp(p1,p2,p3) sign(cross(p1,p2,p3))
44
45 bool chkLL(P p1,P p2,P q1,P q2) //0:parallel
46 {
47     db a1=cross(q1,q2,p1),a2=-cross(q1,q2,p2);
48     return sign(a1+a2)!=0;
49 }
50
51 P isLL(P p1,P p2,P q1,P q2) //crossover point if
    chkLL()
52 {
53     db a1=cross(q1,q2,p1),a2=-cross(q1,q2,p2);
54     return (p1*a2+p2*a1)/(a1+a2);
55 }
56
57 bool intersect(db l1,db r1,db l2,db r2)
58 {
59     if(l1>r1) swap(l1,r1);if(l2>r2) swap(l2,r2);
60     return !(cmp(r1,l2)==-1||cmp(r2,l1)==-1);
61 }
62
63 bool isSS(P p1,P p2,P q1,P q2)
64 {
65     return intersect(p1.x,p2.x,q1.x,q2.x)&&intersect(
        p1.y,p2.y,q1.y,q2.y)&&
66     crossOp(p1,p2,q1)*crossOp(p1,p2,q2)<=0&&crossOp(q1
        ,q2,p1)*crossOp(q1,q2,p2)<=0;
67 }
68

```

```

69 bool isSS_strict(P p1,P p2,P q1,P q2)
70 {
71     return crossOp(p1,p2,q1)*crossOp(p1,p2,q2)<0
    &&crossOp(q1,q2,p1)*crossOp(q1,q2,p2)<0;
72 }
73
74 bool isMiddle(db a,db m,db b)
75 {
76     return sign(a-m)==0||sign(b-m)==0||(a<m!=b<m);
77 }
78
79 bool isMiddle(P a,P m,P b)
80 {
81     return isMiddle(a.x,m.x,b.x)&&isMiddle(a.y,m.y,b.y
    );
82 }
83
84 bool onSeg(P p1,P p2,P q)
85 {
86     return crossOp(p1,p2,q)==0&&isMiddle(p1,q,p2);
87 }
88
89 bool onSeg_strict(P p1,P p2,P q)
90 {
91     return crossOp(p1,p2,q)==0&&sign((q-p1).dot(p1-p2)
    )*sign((q-p2).dot(p1-p2))<0;
92 }
93
94 P proj(P p1,P p2,P q)
95 {
96     P dir=p2-p1;
97     return p1+dir*(dir.dot(q-p1)/dir.abs2());
98 }
99
100 P reflect(P p1,P p2,P q)
101 {
102     return proj(p1,p2,q)*2-q;
103 }
104
105 db nearest(P p1,P p2,P q)
106 {
107     P h=proj(p1,p2,q);
108     if(isMiddle(p1,h,p2))
109         return q.distTo(h);
110     return min(p1.distTo(q),p2.distTo(q));
111 }
112
113 db disSS(P p1,P p2,P q1,P q2) //dist of 2 segments
114 {
115     if(isSS(p1,p2,q1,q2)) return 0;
116     return min(min(nearest(p1,p2,q1),nearest(p1,p2,q2)
    ),min(nearest(q1,q2,p1),nearest(q1,q2,p2)));
117 }
118
119 db rad(P p1,P p2)
120 {
121     return atan2l(p1.det(p2),p1.dot(p2));
122 }
123
124 db area(vector<P> ps)
125 {
126     db ret=0;
127     for(int i=0;i<ps.size();i++)
128         ret+=ps[i].det(ps[(i+1)%ps.size()]);
129     return ret/2;
130 }

```



```

131 }
132
133 int contain(vector<P> ps,P p) //2:inside,1:on_seg,0:
    outside
134 {
135     int n=ps.size(),ret=0;
136     for(int i=0;i<n;i++)
137     {
138         P u=ps[i],v=ps[(i+1)%n];
139         if(onSeg(u,v,p)) return 1;
140         if(cmp(u.y,v.y)<=0) swap(u,v);
141         if(cmp(p.y,u.y)>0||cmp(p.y,v.y)<=0) continue;
142         ret^=crossOp(p,u,v)>0;
143     }
144     return ret*2;
145 }
146
147 vector<P> convexHull(vector<P> ps)
148 {
149     int n=ps.size();if(n<=1) return ps;
150     sort(ps.begin(),ps.end());
151     vector<P> qs(n*2);int k=0;
152     for(int i=0;i<n;qs[k++]=ps[i++])
153         while(k>1&&crossOp(qs[k-2],qs[k-1],ps[i])<=0)
154             --k;
155     for(int i=n-2,t=k;i>=0;qs[k++]=ps[i--])
156         while(k>t&&crossOp(qs[k-2],qs[k-1],ps[i])<=0)
157             --k;
158     qs.resize(k-1);
159     return qs;
160 }
161
162 db convexDiameter(vector<P> ps)
163 {
164     int n=ps.size();if(n<=1) return 0;
165     int is=0,js=0;
166     for(int k=1;k<n;k++) is=ps[k]<ps[is]?k:is,js=ps[js
167         ]<ps[k]?js:k;
168     int i=is,j=js;
169     db ret=ps[i].distTo(ps[j]);
170     do{
171         if((ps[(i+1)%n]-ps[i]).det(ps[(j+1)%n]-ps[j])
172             >=0) (++j)%=n;
173         else (++i)%=n;
174         ret=max(ret,ps[i].distTo(ps[j]));
175     }while(i!=is||j!=js);
176     return ret;
177 }

```

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

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

3.2.5 最小乘积生成树

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 {
19     if(x!=f[x]) f[x]=_find(f[x]);
20     return f[x];
21 }
22 void _merge(int x,int y)
23 {
24     x=_find(x),y=_find(y);
25     if(x!=y) f[x]=y;
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%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].
            push_back(x);
53     for(int i=1;i<=m;++i)
54         scanf("%d%d",&x,&y),
55         q[x].push_back(y),q[y].push_back(x),
56         id[x].push_back(i),id[y].push_back(i);
57     tarjan(s);
58     for(int i=1;i<=m;++i)
59         printf("%d\n",ans[i]);
60     return 0;
61 }

```

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  */
8
9  const int MAXN=500005;
10 int n,q,dep[MAXN],s,lg[MAXN],fa[MAXN][32];
11 vector<int> e[MAXN];
12
13 void dfs(int now,int pa)
14 {
15     dep[now]=dep[pa]+1;
16     fa[now][0]=pa;
17     for(int i=1;(1<<i)<=dep[now];i++)
18         fa[now][i]=fa[fa[now][i-1]][i-1];
19     for(auto to:e[now])
20         if(to!=pa) dfs(to,now);
21 }
22
23 int lca(int x,int y)
24 {
25     if(dep[x]<dep[y]) swap(x,y);
26     while(dep[x]>dep[y]) x=fa[x][lg[dep[x]-dep[y]]-1];
27     if(x==y) return x;
28     for(int i=lg[dep[x]]-1;i>=0;i--)
29         if(fa[x][i]!=fa[y][i])
30             x=fa[x][i],y=fa[y][i];
31     return fa[x][0];
32 }
33
34 int main()
35 {
36     for(int i=1;i<MAXN;i++)
37         lg[i]=lg[i-1]+(1<<lg[i-1]==i);
38     scanf("%d%d%d",&n,&q,&s);
39     for(int i=0,x,y;i<n-1;i++)
40     {
41         scanf("%d%d",&x,&y);
42         e[x].push_back(y),e[y].push_back(x);
43     }

```

```

44     dep[0]=0;
45     dfs(s,0);
46     for(int i=0,x,y;i<q;i++)
47     {
48         scanf("%d%d",&x,&y);
49         printf("%d\n",lca(x,y));
50     }
51     return 0;
52 }

```

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]++;//
24     int son=0,tmp=0;
25     for(auto v:e[u])
26     {
27         if(dfn[v]==0)
28         {
29             tarjan(v,u);
30             sz[u]+=sz[v];//
31             low[u]=min(low[u],low[v]);
32             if(low[v]>=dfn[u]&&u!=f)
33             {
34                 is[u]=1;
35                 tmp+=sz[v];//
36                 part[u].push_back(sz[v]);//
37             }
38             if(u==f) son++;
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=1000005;
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 }
14 inline void write(int x){
15     if(x<0)putchar('-'),x=-x;
16     if(x>9)write(x/10);
17     putchar(x%10+48);
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  */

```

```

9
10 const int N=10005;
11
12 int n,m,x[N*5],y[N*5];
13 vector<int> e[N],scc[N];
14 int co[N],color=0;
15 stack<int> s;
16 bool vis[N];
17 int dfn[N],low[N],timer=0;
18
19 void tarjan(int u)
20 {
21     dfn[u]=low[u]=++timer;
22     s.push(u);
23     vis[u]=1;
24     for(auto v:e[u])
25     {
26         if(!dfn[v])
27         {
28             tarjan(v);
29             low[u]=min(low[u],low[v]);
30         }
31         else if(vis[v]) low[u]=min(low[u],dfn[v]);
32     }
33     if(low[u]==dfn[u])
34     {
35         ++color;
36         int t;
37         do
38         {
39             t=s.top();
40             s.pop();
41             co[t]=color;
42             vis[t]=0;
43             scc[color].push_back(t);
44         }
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 */
10
11 const int N=2e6+5;
12
13 int n,m,a,va,b,vb;
14 int dfn[N],low[N],timer=0;
15 stack<int> s;
16 bool vis[N];
17 vector<int> e[N];
18 int co[N],color=0;
19
20 void add(int x,int y)
21 {
22     e[x].push_back(y);
23 }
24
25 void tarjan(int u)
26 {
27     dfn[u]=low[u]=++timer;
28     s.push(u);
29     vis[u]=1;
30     for(auto v:e[u])
31     {
32         if(!dfn[v])
33             tarjan(v),
34             low[u]=min(low[u],low[v]);
35         else if(vis[v])
36             low[u]=min(low[u],dfn[v]);
37     }

```

```

38     if(low[u]==dfn[u])
39     {
40         int v;
41         color++;
42         do
43         {
44             v=s.top();
45             s.pop();
46             vis[v]=0;
47             co[v]=color;
48         }
49         while(u!=v);
50     }
51 }
52
53 bool solve()
54 {
55     for(int i=1;i<=2*n;++i)
56         if(!dfn[i]) tarjan(i);
57     for(int i=1;i<=n;++i)
58         if(co[i]==co[i+n])
59             return 0;
60     return 1;
61 }
62
63 int main()
64 {
65     scanf("%d%d",&n,&m);
66     for(int i=1;i<=m;++i)
67     {
68         scanf("%d%d%d",&a,&va,&b,&vb);
69         int nota=va^1,notb=vb^1;
70         add(a+nota*n,b+vb*n);//not a and b
71         add(b+notb*n,a+va*n);//not b and a
72     }
73     if(solve())
74     {
75         puts("POSSIBLE");
76         for(int i=1;i<=n;++i)
77             printf("%d ",co[i]>co[i+n]);
78     }
79     else puts("IMPOSSIBLE");
80     return 0;
81 }

```

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(
19         rev){}
20 }
21 vector<edge> E[MAXN];
22
23 inline void add_edge(int x,int y,int f)
24 {
25     E[x].emplace_back(y,f,E[y].size());
26     E[y].emplace_back(x,0,E[x].size()-1);
27 }
28
29 int bfs()
30 {
31     for(int i=1;i<=n;i++) dis[i]=0x3f3f3f3f;
32     dis[s]=0;
33     queue<int> q;
34     q.push(s);
35     while(!q.empty())
36     {
37         int now=q.front();q.pop();
38         for(int i=0;i<E[now].size();i++)
39         {
40             edge &e=E[now][i];
41             if(dis[e.to]>dis[now]+1&&e.cap)
42             {
43                 dis[e.to]=dis[now]+1;
44                 if(e.to==t) return 1;
45                 q.push(e.to);
46             }
47         }
48     }
49     return 0;
50 }
51
52 ll dfs(int now,ll flow)
53 {
54     if(now==t) return flow;
55     ll rest=flow,k;
56     for(int i=cur[now];i<E[now].size();i++)
57     {
58         edge &e=E[now][i];
59         if(e.cap&&dis[e.to]==dis[now]+1)
60         {
61             cur[now]=i;
62             k=dfs(e.to,min(rest,(long long)e.cap));
63         }
64     }
65     return flow-k;
66 }

```



```

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;

```

```

43         for(int i=0;i<E[now].size();i++)
44         {
45             edge &e=E[now][i];
46             if(e.cap>0&&dis[e.to]>p.first+e.val+h[now]-
47                 h[e.to])
48             {
49                 dis[e.to]=p.first+e.val+h[now]-h[e.to];
50                 q.emplace(dis[e.to],e.to);
51             }
52         }
53     }
54     return dis[t]!=inf;
55 }
56
57 int dfs(int now,int flow)
58 {
59     if(now==t) return flow;
60     int rest=flow,k;
61     vis[now]=1;
62     for(int i=cur[now];i<E[now].size();i++)
63     {
64         edge &e=E[now][i];
65         if(e.cap&&dis[now]+e.val+h[now]-h[e.to]==dis[
66             e.to]&&!vis[e.to])
67         {
68             cur[now]=i;
69             k=dfs(e.to,min(e.cap,rest));
70             e.cap-=k;
71             E[e.to][e.rev].cap+=k;
72             rest-=k;
73             msmf+=k*e.val;
74         }
75     }
76     vis[now]=0;
77     return flow-rest;
78 }
79
80 int dinic()
81 {
82     int ret=0,delta;
83     while(dij())
84     {
85         for(int i=s;i<=t;i++) cur[i]=0;
86         while(delta=dfs(s,inf)) ret+=delta;
87         for(int i=s;i<=t;i++) h[i]+=(dis[i]==inf)?0:
88             dis[i];
89     }
90     return ret;
91 }

```

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=0x3f3f3f3f3f3f3f3f;
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),cost(cost),rev(rev){}
22 };
23 vector<edge> E[5005];
24
25 inline void add_edge(int x,int y,ll f,ll c)
26 {
27     E[x].emplace_back(y,f,c,E[y].size());
28     E[y].emplace_back(x,0,-c,E[x].size()-1);
29 }
30
31 int spfa()
32 {
33     for(int i=0;i<=n;i++) vis[i]=0,dis[i]=inf; //从编号最小的点到最大的点
34     dis[s]=0;
35     queue<int> q;
36     q.push(s);
37     while(!q.empty())
38     {
39         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; //从编号最小的点到最大的点
81         while(delta=dfs(s,inf)) ret+=delta;
82     }
83     return ret;
84 }
85
86 int main()
87 {
88     scanf("%d%d%d%d",&n,&m,&s,&t);
89     for(int i=0;i<m;i++)
90     {
91         int x,y,ll a,b;
92         scanf("%d%d%lld%lld",&x,&y,&a,&b);
93         add_edge(x,y,a,b);
94     }
95     ll mxflow=dinic();
96     printf("%lld %lld",mxflow,msmf);
97     return 0;
98 }

```

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 }

```

3.9 Prufer 序列

4 数据结构

4.1 树状数组

4.2 线段树

4.2.1 带优先级线段树

4.2.2 吉司机线段树

```

1  #include <bits/stdc++.h>
2  using namespace std;
3  typedef long long ll;
4
5  /*
6   modify 将区间大于x的数变成x
7   query 询问区间和
8   单次复杂度 O(log^2(n))
9  */
10
11 const ll INF=0xc0c0c0c0c0c0c0c0ll;
12 const int MAXN=200005;
13 ll seg[MAXN<<2],m1[MAXN<<2],m2[MAXN<<2],cnt[MAXN<<2],
14   tag[MAXN<<2],a[MAXN];
15 int n,q;
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 }

```

4.2.3 线段树维护扫描线

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 两维

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[
    MAXN],num[MAXN],fa[MAXN];
17 vector<int> e[MAXN];
18
19 void dfs1(int now,int f)
20 {
21     dep[now]=dep[f]+1;
22     fa[now]=f;
23     num[now]=1;
24     son[now]=0;
25     for(auto to:e[now])
26     {
27         if(to==f) continue;
28         dfs1(to,now);

```

```

29         num[now]+=num[to];
30         if(num[to]>num[son[now]]) son[now]=to;
31     }
32 }
33
34 void dfs2(int now,int f)
35 {
36     id[now]=++cnt;
37     top[now]=f;
38     if(son[now]) dfs2(son[now],f);
39     for(auto to:e[now])
40         if(to!=fa[now]&&to!=son[now])
41             dfs2(to,to);
42 }
43
44 inline void pushdown(int rt,ll lnum,ll rnum)
45 {
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     }
158     return 0;

```

157 }

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].
30             siz;
31         if(bst[rt].rc) bst[rt].siz+=bst[bst[rt].rc].
32             siz;
33     }
34
35     inline void zig(int &rt)
36     {
37         int p=bst[rt].lc;
38         bst[rt].lc=bst[p].rc;
39         bst[p].rc=rt;
40         rt=p;
41         pushup(bst[rt].rc);pushup(rt);
42     }
43
44     inline void zag(int &rt)
45     {
46         int p=bst[rt].rc;
47         bst[rt].rc=bst[p].lc;
48         bst[p].lc=rt;
49         rt=p;
50         pushup(bst[rt].lc);pushup(rt);
51     }
52
53     int new_node(int val)
54     {
55         bst[++tot].val=val;
56         bst[tot].dat=rand();
57         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);

```

```

118     pushup(rt);
119 }
120
121 int getPrev(int val)
122 {
123     int ret=1,rt=root;
124     while(rt)
125     {
126         if(bst[rt].val==val)
127         {
128             if(bst[rt].lc)
129             {
130                 rt=bst[rt].lc;
131                 while(bst[rt].rc) rt=bst[rt].rc;
132                 ret=rt;
133             }
134             break;
135         }
136         if(bst[rt].val<val&&bst[rt].val>bst[ret].
137             val) ret=rt;
138         if(val<bst[rt].val) rt=bst[rt].lc;
139         else rt=bst[rt].rc;
140     }
141     return bst[ret].val;
142 }
143
144 int getNext(int val)
145 {
146     int ret=2,rt=root;
147     while(rt)
148     {
149         if(bst[rt].val==val)
150         {
151             if(bst[rt].rc)
152             {
153                 rt=bst[rt].rc;
154                 while(bst[rt].lc) rt=bst[rt].lc;
155                 ret=rt;
156             }
157             break;
158         }
159         if(bst[rt].val>val&&bst[rt].val<bst[ret].
160             val) ret=rt;
161         if(val<bst[rt].val) rt=bst[rt].lc;
162         else rt=bst[rt].rc;
163     }
164     return bst[ret].val;
165 }
166
167 int getRank(int rt,int val)
168 {
169     if(rt==0) return 0;
170     if(val==bst[rt].val) return bst[bst[rt].lc].
171         siz+1;
172     if(val<bst[rt].val) return getRank(bst[rt].lc,
173         val);
174     else return bst[bst[rt].lc].siz+bst[rt].cnt+
175         getRank(bst[rt].rc,val);
176 }
177
178 int getVal(int rt,int k)
179 {
180     if(rt==0) return inf;
181     if(bst[bst[rt].lc].siz>=k) return getVal(bst[
182         rt].lc,k);

```

```

177     if(bst[bst[rt].lc].siz+bst[rt].cnt>=k) return
178         bst[rt].val;
179     return getVal(bst[rt].rc,k-bst[bst[rt].lc].siz
180         -bst[rt].cnt);
181 }
182 int main()
183 {
184     using namespace Treap;
185     srand(time(0));
186     build();
187     scanf("%d",&n);
188     while(n-->0)
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)
194             -1);
195         else if(op==4) printf("%d\n",getVal(root,x+1))
196             ;
197         else if(op==5) printf("%d\n",getPrev(x));
198         else if(op==6) printf("%d\n",getNext(x));
199     }
200     return 0;
201 }

```

4.5.2 Splay

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",&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-->0)
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,
74             k)]);
75     }
76     return 0;
77 }

```

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
11 const int inf=2147483647;
12 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<
78                     bst[bst[rt].lc].dat)
79                     zig(rt),_delete(bst[rt].rc,x);
80                 else
81                     zag(rt),_delete(bst[rt].lc,x);
82                 pushup(rt);
83             }
84             else rt=0;
85             return;
86         }
87         if(x<bst[rt].val) _delete(bst[rt].lc,x);
88         else _delete(bst[rt].rc,x);
89         pushup(rt);
90     }
91
92     void _insert(int &rt,int x)
93     {
94         if(rt==0)
95         {
96             rt=newnode(x);
97             return;
98         }
99         if(bst[rt].val==x) bst[rt].cnt++;
100         else if(x<bst[rt].val)
101         {
102             _insert(bst[rt].lc,x);
103             if(bst[bst[rt].lc].dat>bst[rt].dat) zig(rt);
104         }
105         else
106         {
107             _insert(bst[rt].rc,x);
108             if(bst[bst[rt].rc].dat>bst[rt].dat) zag(rt);
109         }
110         pushup(rt);
111     }
112
113     int get_rank(int rt,int x)
114     {
115         if(!rt) return 1;
116         if(bst[rt].val==x) return bst[bst[rt].lc].siz+1;
117         if(x<bst[rt].val) return get_rank(bst[rt].lc,x);
118         else return get_rank(bst[rt].rc,x)+bst[bst[rt].lc].siz+bst[rt].cnt;
119     }
120
121     int get_num(int rt,int x)
122     {
123         if(!rt) return 0;
124         if(bst[rt].val==x) return bst[bst[rt].lc].siz+bst[rt].cnt;
125         if(x<bst[rt].val) return get_num(bst[rt].lc,x);
126         else return get_num(bst[rt].rc,x)+bst[bst[rt].lc].siz+bst[rt].cnt;
127     }
128
129     int get_prev(int rt,int x)
130     {
131         int ret=-inf;

```



```

131 while(rt)
132 {
133     if(bst[rt].val==x)
134     {
135         if(bst[rt].lc)
136         {
137             rt=bst[rt].lc;
138             while(bst[rt].rc) rt=bst[rt].rc;
139             ret=bst[rt].val;
140         }
141         break;
142     }
143     if(bst[rt].val<x&&bst[rt].val>ret) ret=bst[rt].val;
144     if(x<bst[rt].val) rt=bst[rt].lc;
145     else rt=bst[rt].rc;
146 }
147 return ret;
148 }
149
150 int get_nxt(int rt,int x)
151 {
152     int ret=inf;
153     while(rt)
154     {
155         if(bst[rt].val==x)
156         {
157             if(bst[rt].rc)
158             {
159                 rt=bst[rt].rc;
160                 while(bst[rt].lc) rt=bst[rt].lc;
161                 ret=bst[rt].val;
162             }
163             break;
164         }
165         if(bst[rt].val>x&&bst[rt].val<ret) ret=bst[rt].val;
166         if(x<bst[rt].val) rt=bst[rt].lc;
167         else rt=bst[rt].rc;
168     }
169     return ret;
170 }
171 }treap;
172
173 void build(int rt,int l,int r)
174 {
175     root[rt]=treap.build();
176     if(l==r) return;
177     int m=l+r>>1;
178     build(rt<<1,l,m);
179     build(rt<<1|1,m+1,r);
180 }
181
182 void modify(int rt,int l,int r,int x,int v,int y)
183 {
184     if(y==-1) treap._delete(root[rt],v);
185     else treap._insert(root[rt],v);
186     if(l==r) return;
187     int m=l+r>>1;
188     if(x<=m) modify(rt<<1,l,m,x,v,y);
189     else modify(rt<<1|1,m+1,r,x,v,y);
190 }
191
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|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|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|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             }
252             printf("%d\n",L);
253         }
254     }
255 }

```

```

253     else if(op==3)
254     {
255         scanf("%d%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%d",&l,&r,&k);
263         printf("%d\n",query(1,1,n,op,l,r,k));
264     }
265 }
266 return 0;
267 }

```

```

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],1,VAL,x,y);
52 }
53
54 int query(int l,int r,int k)
55 {
56     c1=c2=0;
57     for(int i=l-1;i-=i&-i) X[c1++]=seg.root[i];
58     for(int i=r;i-=i&-i) Y[c2++]=seg.root[i];
59     return seg.query(1,VAL,k);
60 }

```

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 }

```

4.9 K-D Tree

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-=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 }

```

```

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=1;i<=m;++i) add(o[i].c,-1);
45     for(int i=1;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   解决有等值的三维偏序
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<=n;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(1,m);
39     cdq(m+1,r);
40     sort(s+1,s+m+1,cmp2);
41     sort(s+m+1,s+r+1,cmp2); //sort2
42     int i=1,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=1;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<=nn;++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--)
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 }

```

```

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 {
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++);

```

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[i]&(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
8   空，否则会RE
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 };
32 set<node> s;
33 typedef set<node>::iterator it;
34
35 it split(int pos)
36 {
37     it i=s.lower_bound(node(pos));
38     if(i!=s.end()&&i->l==pos) return i;
39     --i;
40     int L=i->l,R=i->r,V=i->v;
41     s.erase(i);
42     s.insert(node(L,pos-1,V));
43     return s.insert(node(pos,R,V)).first;
44 }
45
46 void assign(int l,int r,int val)
47 {

```

```

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
84 inline int read(){
85     char ch=getchar();int s=0,w=1;
86     while(ch<48||ch>57){if(ch=='-')w=-1;ch=getchar();}
87     while(ch>=48&&ch<=57){s=(s<<1)+(s<<3)+ch-48;ch=
88     getchar();}
89     return s*w;
90 }
91
92 inline void write(int x){
93     if(x<0)putchar('-'),x=-x;
94     if(x>9)write(x/10);
95     putchar(x%10+48);
96 }
97
98 //Fast I/O
99
100 int n,m,seed,vmax,a[N];
101 int rnd()
102 {
103     int ret=seed;
104     seed=(seed*7+13)%1000000007;
105     return ret;
106 }
107
108 signed main()
109 {
110     n=read(),m=read(),seed=read(),vmax=read();
111     for(int i=1;i<=n;++i)
112     {
113         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 }
135 return 0;
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

//答案都取 r 浮点数可以取 (l+r)/2 //浮点数极小值 while(l+eps<r) double lm=(l+r)/2,rm=(lm+r)/2; if(judge(lm)>judge(rm)) l=lm; else r=rm; //整数极小值 while(l+1<r) int lm=(l+r)>>1,rm=(lm+r)>>1; if(judge(lm)>judge(rm)) l=lm; else r=rm; //浮点数极大值 while(l+eps<r) double lm=(l+r)/2,rm=(lm+r)/2; if(judge(lm)>judge(rm)) r=rm; else l=lm; //整数极大值 while(l+1<r) int lm=(l+r)>>1,rm=(lm+r)>>1; if(judge(lm)>judge(rm)) r=rm; else l=lm;

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

6.5 质因数分解

6.5.1 Pollard-Rho

6.6 逆元

6.6.1 EX-GCD 求逆元

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

6.7.4 一般积性函数求法

6.8 EX-GCD

6.9 CRT

6.10 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

6.15 FFT

6.16 FWT

6.17 NTT

6.18 数值计算

6.18.1 辛普森

6.18.2 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 }

```

6.19 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 }

```

6.20 卢卡斯定理

$$C_N^m \equiv C_{N \bmod p}^m \cdot C_{N/p}^{m/p} \pmod{p}$$

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

可理解为将 N 和 m 表示为 p 进制数，对每一位的 N_i 和 m_i 分别求组合数，再累乘

```

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 }

```

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;
38     template<typename T> BigNum operator / (const T &)
39         const;
40     template<typename T> BigNum operator ^ (const T &)
41         const;
42     template<typename T> T operator % (const T &)
43         const;
44
45     template<typename T> BigNum operator + (const T& b
46         ) const {BigNum t = b; t = *this + t; return
47         t;}
48     template<typename T> BigNum operator - (const T& b
49         ) const {BigNum t = b; t = *this - t; return
50         t;}
51     template<typename T> BigNum operator * (const T& b
52         ) const {BigNum t = b; t = (*this) * t;
53         return t;}
54     template<typename T> bool operator < (const T& b)
55         const {BigNum t = b; return ((*this) < t);}
56     template<typename T> bool operator > (const T& b)
57         const {BigNum t = b; return ((*this) > t);}
58     template<typename T> bool operator == (const T& b)
59         const {BigNum t = b; return ((*this) == t);}
60
61     bool operator <= (const BigNum& b) const {return
62         (*this) < b || (*this) == b;}
63     bool operator >= (const BigNum& b) const {return
64         (*this) > b || (*this) == b;}
65     bool operator != (const BigNum& b) const {return
66         !((*this) == b);}
67
68     template<typename T> bool operator >= (const T& b)
69         const {BigNum t = b; return !((*this) < t);}
70     template<typename T> bool operator <= (const T& b)
71         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  BigNum::BigNum(const int b)
92  {
93      int c,d = b;
94      len = 0;
95      memset(a,0,sizeof(a));
96      while(d > MAXN){
97          c = d - (d / (MAXN+1)) * (MAXN+1);
98          d = d / (MAXN+1);
99          a[len++] = c;
100      }
101      a[len++] = d;
102  }
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(l%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  }
135  BigNum::BigNum(const char* s)
136  {
137      int t,k,index,l,i;
138      memset(a,0,sizeof(a));
139      l = strlen(s);
140      len = l/DLEN;
141      if(l%DLEN)
142          len++;
143      index = 0;
144      for(i = l-1; i >= 0; i -= DLEN){
145          t = 0;
146          k = i - DLEN + 1;
147          if(k < 0) k = 0;
148          for(int j = k; j <= i; j++){
149              t = t*10 + s[j] - '0';
150              a[index++] = t;
151          }
152      }
153  }
154  BigNum::BigNum(const BigNum & b) : len(b.len)
155  {
156      memset(a,0,sizeof(a));
157      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; ){
187         sum = 0;
188         int t = 1;
189         for(int j = 0; j < 4 && i >= 0; j++,i--,t *=
190             10)
191             sum += (ch[i]-'0')*t;
192         b.a[cnt] = sum;
193         cnt++;
194     }
195     b.len = cnt++;
196     return in;
197 }
198 ostream& operator << (ostream& out, BigNum& b)
199 {
200     int i;
201     cout << b.a[b.len - 1];
202     for(i = b.len - 2 ; i >= 0 ; i--){
203         cout.width(DLEN);
204         cout.fill('0');
205         cout << b.a[i];
206     }
207     return out;
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 >> (

```

```

218     const T& b) const
219 {
220     T temp = 1;
221     for(int i = 0; i < b; i++)
222         temp *= 2;
223     BigNum t = (*this) / temp;
224     return t;
225 }
226 BigNum BigNum::operator + (const BigNum& b) const
227 {
228     BigNum t(*this);
229     int i,big;
230     big = b.len > len ? b.len : len;
231     for(i = 0 ; i < big ; i++){
232         t.a[i] += b.a[i];
233         if(t.a[i] > MAXN){
234             t.a[i + 1]++;
235             t.a[i] -=MAXN+1;
236         }
237     }
238     if(t.a[big] != 0)
239         t.len = big + 1;
240     else
241         t.len = big;
242     return t;
243 }
244 BigNum BigNum::operator - (const BigNum& b) const
245 {
246     int i,j,big;
247     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     {
312         BigNum ret;
313         T i,down = 0;
314         for(i = len - 1 ; i >= 0 ; i--){
315             ret.a[i] = (a[i] + down * (MAXN + 1)) / b;
316             down = a[i] + down * (MAXN + 1) - ret.a[i] * b
317                 ;
318         }
319         ret.len = len;
320         while(ret.a[ret.len - 1] == 0 && ret.len > 1)
321             ret.len--;
322         return ret;
323     }
324     template<typename T> T BigNum::operator % (const T& b
325         ) const
326     {
327         T i,d=0;
328         for (i = len-1; i>=0; i--){
329             d = ((d * (MAXN+1))% b + a[i])% b;
330         }
331         return d;
332     }
333     template<typename T> BigNum BigNum::operator^(const T
334         & n) const
335     {
336         BigNum t,ret(1);
337         int i;
338         if(n < 0) return 0;
339         if(n == 0)
340             return 1;
341         if(n == 1)
342             return *this;
343         int m = n;

```

```

342     while(m > 1){
343         t =* this;
344         for(i = 1; (i<<1) <= m;i <= 1)
345             t = t*t;
346         m-=i;
347         ret=ret*t;
348         if(m == 1) ret = ret * (*this);
349     }
350     return ret;
351 }
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 }
439 long long int BigNum::to_long()
440 {
441     int i;
442     long long int num;
443     num = a[len-1];
444     for(i = len-2; i >= 0; i--){
445         num = num*(MAXN+1) + a[i];
446     }
447     return num;
448 }
449 int main()
450 {
451     BigNum a,b;
452     cin>>a>>b;
453     (a+b).print();
454     return 0;
455 }

```

7.3 约瑟夫环

7.4 悬线法

7.5 蔡勒公式

7.6 三角公式

7.7 海伦公式

7.8 匹克定理

7.9 组合计数

7.9.1 计数原理

7.9.2 卡特兰数

7.9.3 Polya

7.9.4 二项式反演公式

7.9.5 斯特林反演公式

7.9.6 组合数恒等式