# ACM 常用算法模板



# kuangbin

http://www.cnblogs.com/kuangbin/ Email:kuangbin2009@126.com Last build at July 8, 2018

# Contents

1	字符	串处理                      5
	1.1	KMP
	1.2	e-KMP
	1.3	Manacher
	1.4	AC 自动机
	1.5	后缀数组
	1.0	1.5.1 DA
		1.5.2 DC3
	1.6	后缀自动机
	1.0	1.6.1 基本函数
		1.6.2 例题
	1.7	字符串 hash
	1.1	子行中 Hash
2	数学	25
_	2.1	素数
		2.1.1 素数筛选(判断 <maxn td="" 的数是否素数)<=""></maxn>
		2.1.2 素数筛选(筛选出小于等于 MAXN 的素数)
		2.1.3 <b>大区间素数筛选</b> (POJ 2689)
	2.2	素数筛选和合数分解
	2.3	扩展欧几里得算法(求 $ax+by=gcd$ 的解以及逆元)
	2.3	が 展めた主句穿法 (水 ax+by-ged 的解め及送光) 27 求逆元 28
	2.4	2.4.1 扩展欧几里德法 28
		* *** *** == 1
	2.5	2.4.3       利用欧拉函数
	2.6	随机素数测试和大数分解 (POJ 1811)
	2.7	欧拉函数
		2.7.1 分解质因素求欧拉函数 32
		2.7.2 筛法欧拉函数
		2.7.3 求单个数的欧拉函数
		2.7.4 线性筛(同时得到欧拉函数和素数表) 32
	2.8	高斯消元(浮点数)
	2.9	FFT
	2.10	高斯消元法求方程组的解
		2.10.1 一类开关问题,对 2 取模的 01 方程组
		2.10.2 解同余方程组
		整数拆分 41
	2.12	求 $A^B$ 的约数之和对 MOD 取模
	2.13	莫比乌斯反演
		2.13.1 莫比乌斯函数
		2.13.2 <b>例题:</b> BZOJ2301
	2.14	Baby-Step Giant-Step
	2.15	自适应 simpson 积分
		<b>斐波那契数列取模循环节</b> 46
		原根
		 快速数论变换
	0	2.18.1 HDU4656 <b>卷积取模</b>
	2.19	其它公式
	2.10	9.10.1 Dolvo

3	数据	吉构 $56$
	3.1	划分树
	3.2	RMQ 57
		3.2.1 一维
		3.2.2 <b>二维</b>
	3.3	树链剖分 59
		3.3.1 点权
		3.3.2 边权
	3.4	伸展树(splay tree)
		3.4.1 <b>例题:</b> HDU1890
		3.4.2 <b>例题:</b> HDU3726
	3.5	动态树
		3.5.1 SPOJQTREE
		3.5.2 SPOJQTREE2
		3.5.3 SPOJQTREE4
		3.5.4 SPOJQTREE5
		3.5.5 SPOJQTREE6
		3.5.6 SPOJQTREE7
		3.5.7 HDU4010
	3.6	主席树
		3.6.1 <b>查询区间多少个不同的数</b> 95
		3.6.2 静态区间第 k 大 97
		3.6.3 <b>树上路径点权第</b> k 大
		3.6.4  动态第 k 大
	3.7	Treap
	3.8	KD 树
		3.8.1 HDU4347 K 近邻
		3.8.2 CF44G
		3.8.3 HDU4742
	3.9	替罪羊树 (ScapeGoat Tree)
		3.9.1 CF455D
	3.10	动态 KD 树
	3.11	树套树
		3.11.1 替罪羊树套 splay
	(A)	100
4	图论	130 目 5
	4.1	最短路
		4.1.1 Dijkstra 单源最短路
		4.1.2 Dijkstra <b>算法</b> + <b>堆优化</b>
		4.1.3 单源最短路 bellman_ford <b>算法</b>
	4.0	4.1.4 单源最短路 SPFA
	4.2	最小生成树
		4.2.1 Prim 算法
	4.0	4.2.2 Kruskal <b>算法</b>
	4.3	次小生成树
	4.4	有向图的强连通分量
		4.4.1 Tarjan
	1 5	4.4.2 Kosaraju
	4.5	
	4.6	割点与桥
		4.6.1 模板

		4.6.2 调用
	4.7	边双连通分支
	4.8	点双连通分支
	4.9	最小树形图
	-	二分图匹配
	1.10	4.10.1 邻接矩阵(匈牙利算法)
		4.10.2 邻接表(匈牙利算法)
		4.10.3 Hopcroft-Karp 算法
	111	- 10.5 Hoperon-Karp 昇辺
		二分图多重四配 · · · · · · · · · · · · · · · · · · ·
		一般图匹配带花树
		一般图最大加权匹配
		生成树计数
	4.16	最大流
		4.16.1 SAP 邻接矩阵形式
		4.16.2 SAP <b>邻接矩阵形式</b> 2
		4.16.3 ISAP 邻接表形式
		4.16.4 ISAP+bfs 初始化 + 栈优化
		4.16.5 dinic
		4.16.6 最大流判断多解
	4.17	最小费用最大流
		4.17.1 SPFA 版费用流
		4.17.2 zkw <b>费用流</b>
	4.18	2-SAT
		4.18.1 染色法(可以得到字典序最小的解)
		4.18.2 强连通缩点法(拓扑排序只能得到任意解)
	4.19	曼哈顿最小生成树
	4.20	LCA
		4.20.1 dfs+ST <b>在线算法</b>
		4.20.2 <b>离线</b> Tarjan <b>算法</b>
		4.20.3 LCA 倍增法
	4 21	欧拉路
	1.21	4.21.1 有向图
		4.21.2 无向图
		4.21.3 混合图
	4 22	树分治
	4.22	4.22.1 点分治 -HDU5016
		4.22.1 点分冶-HDU4918
		4.22.3 链分治 -HDU5039
		4.22.5 挺刀石 -пDU0059
5	搜索	205
0	5.1	Dancing Links
	0.1	5.1.1 精确覆盖
		5.1.2 可重复覆盖
	5.2	八数码
	J.∠	7. 数時
		0.4.1 IIDU1049 X門文系
6	动态	规划 212
3	6.1	最长上升子序列 O(nlogn)
	6.2	背包
	6.3	插头 DP
	(7.1)	

# ACM Template of kuangbin

			_
		6.3.1 HDU 4285	.3
7	计算	几何	8
	7.1	二维几何	8
	7.2	三维几何	8
	7.3	平面最近点对	2
	7.4	三维凸包	13
		7.4.1 HDU4273	
8	其他	24	_
	8.1	高精度	9
	8.2	完全高精度25	0
	8.3	strtok 和 sscanf 结合输入25	6
	8.4	解决爆栈,手动加栈	6
	8.5	STL	6
		8.5.1 优先队列 priority_queue	6
		8.5.2 set 和 multiset	
	8.6	输入输出外挂	8
	8.7		
		8.7.1 分块	
		8.7.2 Manhattan MST <b>的</b> dfs <b>顺序求解</b>	
	8.8	VIM 配置	<b>i</b> 4

## 1 字符串处理

#### 1.1 KMP

```
1
  /*
 2
    * next[] 的含义: x[i-next[i]...i-1]=x[0...next[i]-1]
 3
    * next[i] 为满足 x[i-z...i-1]=x[0...z-1] 的最大 z 值(就是 x 的自身匹配)
 4
 5
   void kmp_pre(char x[],int m,int next[]){
 6
       int i,j;
 7
       j=next[0]=-1;
 8
       i=0;
 9
       while(i<m){</pre>
           while(-1!=j && x[i]!=x[j])j=next[j];
10
11
            next[++i]=++i;
12
       }
13
   }
   /*
14
15
    * kmpNext[i] 的意思:next'[i]=next[next[...[next[i]]]](直到
       next'[i]<0 或者 x[next'[i]]!=x[i])
    * 这样的预处理可以快一些
16
17
    */
   void preKMP(char x[],int m,int kmpNext[]){
18
19
       int i,j;
20
       j=kmpNext[0]=-1;
21
       i=0;
22
       while(i<m){</pre>
           while (-1!=j \& x[i]!=x[j]) j=kmpNext[j];
23
            if(x[++i]==x[++j])kmpNext[i]=kmpNext[j];
24
            else kmpNext[i]=j;
25
26
       }
27
   }
28
   /*
29
    * 返回 x 在 y 中出现的次数,可以重叠
30
    */
31
   int next[10010];
   int KMP_Count(char x[],int m,char y[],int n){//x 是模式串, y 是主串
32
       int i,j;
33
34
       int ans=0;
35
       //preKMP(x,m,next);
36
       kmp_pre(x,m,next);
37
       i=j=0;
       while(i<n){</pre>
38
39
           while(-1!=j \&\& y[i]!=x[j])j=next[j];
40
            i++;j++;
41
            if(j>=m){
42
                ans++;
43
                j=next[j];
            }
44
45
       }
46
       return ans;
```

```
47
   }
   //经典题目: POJ 3167
48
49
   /*
50
    * POJ 3167 Cow Patterns
    * 模式串可以浮动的模式匹配问题
51
52
    * 给出模式串的相对大小,需要找出模式串匹配次数和位置
53
    * 比如说模式串: 1, 4, 4, 2, 3, 1 而主串: 5,6,2,10,10,7,3,2,9
54
    * 那么 2,10,10,7,3,2 就是匹配的
55
56
    * 统计比当前数小,和于当前数相等的,然后进行 kmp
57
    */
   const int MAXN=100010;
58
59
   const int MAXM=25010;
60
   int a[MAXN];
61
   |int b[MAXN];
62
   int n,m,s;
   int as[MAXN][30];
63
64
   int bs[MAXM][30];
65
   void init(){
66
       for(int i=0;i<n;i++){</pre>
            if(i==0){
67
68
                for(int j=1;j<=25;j++)as[i][j]=0;
69
            }
            else{
70
71
                for(int j=1;j<=25;j++)as[i][j]=as[i-1][j];</pre>
72
73
            as[i][a[i]]++;
74
75
       for(int i=0;i<m;i++){</pre>
76
            if(i==0){
77
                for(int j=1;j<=25;j++)bs[i][j]=0;
78
            }
79
            else{
80
                for(int j=1;j<=25;j++)bs[i][j]=bs[i-1][j];</pre>
81
82
            bs[i][b[i]]++;
       }
83
84
85
   int next[MAXM];
86
   void kmp_pre(){
87
       int i,j;
88
       j=next[0]=-1;
89
       i=0;
90
       while(i<m){</pre>
91
            int t11=0, t12=0, t21=0, t22=0;
92
            for(int k=1;k<b[i];k++){
93
                if(i-j>0)t11+=bs[i][k]-bs[i-j-1][k];
94
                else t11+=bs[i][k];
95
96
            if(i-j>0)t12=bs[i][b[i]]-bs[i-j-1][b[i]];
97
            else t12=bs[i][b[i]];
```

```
98
 99
              for(int k=1;k<b[j];k++){
                  t21+=bs[j][k];
100
101
              }
102
              t22=bs[j][b[j]];
103
              if(j==-1 || (t11==t21&&t12==t22)){
                  next[++i]=++j;
104
105
106
             else j=next[j];
         }
107
108
    vector<int>ans;
109
    void kmp(){
110
         ans.clear();
111
112
         int i,j;
113
         kmp_pre();
114
         i=j=0;
         while(i<n){</pre>
115
              int t11=0, t12=0, t21=0, t22=0;
116
              for(int k=1;k<a[i];k++){</pre>
117
                  if(i-j>0)t11+=as[i][k]-as[i-j-1][k];
118
119
                  else t11+=as[i][k];
120
              if(i-j>0)t12=as[i][a[i]]-as[i-j-1][a[i]];
121
122
             else t12=as[i][a[i]];
123
              for(int k=1;k<b[j];k++){</pre>
124
125
                  t21+=bs[j][k];
126
              }
127
              t22=bs[j][b[j]];
128
              if(j==-1 || (t11==t21&&t12==t22)){
129
                  i++;j++;
130
                  if(j>=m){}
                      ans.push_back(i-m+1);
131
132
                      j=next[j];
133
                  }
134
              }
135
             else j=next[j];
         }
136
137
138
    int main(){
139
         while(scanf("%d%d%d",&n,&m,&s)==3){
              for(int i=0;i<n;i++)scanf("%d",&a[i]);</pre>
140
              for(int i=0;i<m;i++)scanf("%d",&b[i]);</pre>
141
              init();
142
143
              kmp();
              printf("%d\n",ans.size());
144
              for(int i=0;i<ans.size();i++)</pre>
145
146
                  printf("%d\n",ans[i]);
147
148
         return 0;
```

```
149 |}
```

#### 1.2 e-KMP

```
1
   /*
 2
    * 扩展 KMP 算法
 3
   //next[i]:x[i...m-1] 与 x[0...m-1] 的最长公共前缀
   //extend[i]:y[i...n-1] 与 x[0...m-1] 的最长公共前缀
 6
   void pre_EKMP(char x[],int m,int next[]){
 7
       next[0] = m;
 8
       int j = 0;
       while( j+1 < m \&\& x[j] == x[j+1] )j++;
 9
       next[1] = j;
10
       int k = 1;
11
       for(int i = 2; i < m; i++){</pre>
12
13
            int p = next[k]+k-1;
14
            int L = next[i-k];
15
            if( i+L < p+1 )next[i] = L;
            else{
16
17
                j = max(0, p-i+1);
                while( i+j < m \&\& x[i+j] == x[j])j++;
18
19
                next[i] = j;
20
                k = i;
21
            }
22
       }
23
   void EKMP(char x[],int m,char y[],int n,int next[],int extend[]){
24
       pre_EKMP(x,m,next);
25
26
       int j = 0;
       while(j < n && j < m && x[j] == y[j])j++;
27
28
       extend[0] = j;
29
       int k = 0;
       for(int i = 1;i < n;i++){</pre>
30
            int p = extend[k]+k-1;
31
32
            int L = next[i-k];
            if(i+L < p+1)extend[i] = L;
33
34
            else{
35
                j = max(0, p-i+1);
                while( i+j < n \&\& j < m \&\& y[i+j] == x[j] )j++;
36
                extend[i] = j;
37
38
                k = i;
39
            }
40
       }
41 |}
   1.3
        Manacher
  /*
 2
    * 求最长回文子串
 3
    */
```

const int MAXN=110010;

```
char Ma[MAXN*2];
 6
   int Mp[MAXN*2];
 7
   void Manacher(char s[],int len){
 8
       int l=0;
 9
       Ma[l++]='$';
       Ma[l++]='#';
10
       for(int i=0;i<len;i++){</pre>
11
12
            Ma[l++]=s[i];
13
            Ma[l++]='#';
       }
14
       Ma[l]=0;
15
       int mx=0,id=0;
16
17
       for(int i=0;i<l;i++){</pre>
            Mp[i]=mx>i?min(Mp[2*id-i],mx-i):1;
18
19
            while(Ma[i+Mp[i]]==Ma[i-Mp[i]])Mp[i]++;
20
            if(i+Mp[i]>mx){
21
                mx=i+Mp[i];
22
                id=i;
23
            }
       }
24
25
   }
26
   /*
27
    * abaaba
              0 1 2 3 4 5 6 7 8 9 10 11 12 13
28
    * i:
29
    * Ma[i]: $ # a # b # a # a # b
                                      #
                                             #
                                          а
    * Mp[i]: 1 1 2 1 4 1 2 7 2 1 4
                                          2
                                             1
30
                                      1
31
    */
32
   char s[MAXN];
33
   int main(){
34
       while(scanf("%s",s)==1){
            int len=strlen(s);
35
            Manacher(s,len);
36
37
            int ans=0;
38
            for(int i=0;i<2*len+2;i++)</pre>
39
                ans=\max(ans,Mp[i]-1);
40
            printf("%d\n",ans);
41
       }
42
       return 0;
43 |}
   1.4 AC 自动机
   //=========
 2
   // HDU 2222
 3
   // 求目标串中出现了几个模式串
   //==========
   struct Trie{
       int next[500010][26],fail[500010],end[500010];
 6
 7
       int root,L;
 8
       int newnode(){
 9
            for(int i = 0;i < 26;i++)
10
                next[L][i] = -1;
```

```
end[L++] = 0;
11
12
            return L-1;
13
14
        void init(){
15
            L = 0;
16
            root = newnode();
17
        void insert(char buf[]){
18
19
            int len = strlen(buf);
            int now = root;
20
            for(int i = 0;i < len;i++){</pre>
21
                if(next[now][buf[i]-'a'] == -1)
22
                     next[now][buf[i]-'a'] = newnode();
23
                now = next[now][buf[i]-'a'];
24
25
            }
26
            end[now]++;
27
        void build(){
28
29
            queue<int>0;
            fail[root] = root;
30
            for(int i = 0;i < 26;i++)
31
32
                if(next[root][i] == −1)
33
                     next[root][i] = root;
                else{
34
35
                     fail[next[root][i]] = root;
                     Q.push(next[root][i]);
36
37
                }
            while( !Q.empty() ){
38
39
                int now = Q.front();
40
                Q.pop();
                for(int i = 0;i < 26;i++)
41
42
                     if(next[now][i] == -1)
43
                         next[now][i] = next[fail[now]][i];
44
                     else
45
                     {
46
                         fail[next[now][i]]=next[fail[now]][i];
47
                         Q.push(next[now][i]);
                     }
48
49
            }
        }
50
        int query(char buf[]){
51
            int len = strlen(buf);
52
53
            int now = root;
54
            int res = 0;
            for(int i = 0;i < len;i++){</pre>
55
                now = next[now][buf[i]-'a'];
56
                int temp = now;
57
                while( temp != root ){
58
59
                     res += end[temp];
60
                     end[temp] = 0;
                     temp = fail[temp];
61
```

```
}
62
63
           }
64
           return res;
65
       void debug(){
66
           for(int i = 0;i < L;i++){</pre>
67
               printf("id_=_%3d,fail_=_%3d,end_=_%3d,chi_=_[",i,fail[i
68
                  ],end[i]);
69
               for(int j = 0; j < 26; j++)
                    printf("%2d",next[i][j]);
70
71
               printf("]\n");
72
           }
       }
73
74
   };
75
   char buf[1000010];
76
   Trie ac;
77
   int main(){
78
       int ⊺;
79
       int n;
       scanf("%d",&T);
80
       while( T— ){
81
82
           scanf("%d",&n);
           ac.init();
83
84
           for(int i = 0;i < n;i++){
               scanf("%s",buf);
85
86
               ac.insert(buf);
           }
87
           ac.build();
88
89
           scanf("%s",buf);
90
           printf("%d\n",ac.query(buf));
91
92
       return 0;
  |}
93
        后缀数组
   1.5
   1.5.1 DA
  /*
 1
 2
   *suffix array
   *倍增算法 O(n*logn)
   |*待排序数组长度为 n, 放在 0 n-1 中, 在最后面补一个 0
 5
   *da(str ,sa,rank,height, n , );//注意是 n;
   *例如:
 6
 7
   *n = 8;
   * num[]
             = { 1, 1, 2, 1, 1, 1, 1, 2, $ }; 注意 num 最后一位为 0, 其他
      大于 0
 9 | *rank[] = 4, 6, 8, 1, 2, 3, 5, 7, 0 ; rank[0 n-1] 为有效值, rank[n]
      必定为 0 无效值
10 |*sa[] = 8, 3, 4, 5, 0, 6, 1, 7, 2 ;sa[1 n] 为有效值, sa[0] 必定为 n 是
      无效值
11 |*height[]= 0, 0, 3, 2, 3, 1, 2, 0, 1 ;height[2 n] 为有效值
```

```
12 | *
13
   */
  const int MAXN=20010;
14
  |int t1[MAXN],t2[MAXN],c[MAXN];//求 SA 数组需要的中间变量,不需要赋值
   |//待排序的字符串放在 s 数组中,从 s[0] 到 s[n-1], 长度为 n, 且最大值小于 m,
16
17
   //除 s[n-1] 外的所有 s[i] 都大于 0, r[n-1]=0
   //函数结束以后结果放在 sa 数组中
18
   bool cmp(int *r,int a,int b,int l){
       return r[a] == r[b] && r[a+l] == r[b+l];
20
21
22
   void da(int str[],int sa[],int rank[],int height[],int n,int m){
23
       n++;
24
       int i, j, p, *x = t1, *y = t2;
25
       //第一轮基数排序,如果 s 的最大值很大,可改为快速排序
26
       for(i = 0;i < m;i++)c[i] = 0;
27
       for(i = 0;i < n;i++)c[x[i] = str[i]]++;
28
       for(i = 1;i < m;i++)c[i] += c[i-1];
29
       for(i = n-1; i >= 0; i--)sa[--c[x[i]]] = i;
       for(j = 1; j \le n; j \le 1)
30
31
           p = 0;
           //直接利用 sa 数组排序第二关键字
32
33
           for(i = n-j; i < n; i++)y[p++] = i;//后面的 j 个数第二关键字为
             空的最小
           for(i = 0; i < n; i++)if(sa[i] >= j)y[p++] = sa[i] - j;
34
35
           //这样数组 y 保存的就是按照第二关键字排序的结果
           //基数排序第一关键字
36
           for(i = 0; i < m; i++)c[i] = 0;</pre>
37
           for(i = 0; i < n; i++)c[x[y[i]]]++;
38
39
           for(i = 1; i < m;i++)c[i] += c[i-1];
40
           for(i = n-1; i >= 0;i--)sa[--c[x[y[i]]]] = y[i];
           //根据 sa 和 x 数组计算新的 x 数组
41
42
           swap(x,y);
43
           p = 1; x[sa[0]] = 0;
44
           for(i = 1;i < n;i++)
45
               x[sa[i]] = cmp(y,sa[i-1],sa[i],j)?p-1:p++;
46
           if(p >= n)break;
47
           m = p;//下次基数排序的最大值
48
49
       int k = 0;
50
51
       for(i = 0;i <= n;i++)rank[sa[i]] = i;</pre>
       for(i = 0;i < n;i++){
52
           if(k)k—;
53
54
           j = sa[rank[i]-1];
55
           while(str[i+k] == str[j+k])k++;
56
           height[rank[i]] = k;
       }
57
58
59
   int rank[MAXN],height[MAXN];
  |int RMQ[MAXN];
61
  |int mm[MAXN];
```

```
62
    |int best[20][MAXN];
 63
    void initRMQ(int n){
 64
         mm[0]=-1;
 65
         for(int i=1;i<=n;i++)</pre>
             mm[i]=((i&(i-1))==0)?mm[i-1]+1:mm[i-1];
 66
         for(int i=1;i<=n;i++)best[0][i]=i;</pre>
 67
         for(int i=1;i<=mm[n];i++)</pre>
 68
             for(int j=1;j+(1<<i)-1<=n;j++){
 69
 70
                  int a=best[i-1][j];
 71
                  int b=best[i-1][j+(1<<(i-1))];</pre>
 72
                  if(RMQ[a]<RMQ[b])best[i][j]=a;
 73
                  else best[i][j]=b;
 74
             }
 75
 76
    int askRMQ(int a,int b){
 77
         int t;
         t=mm[b-a+1];
 78
 79
         b=(1<< t)-1;
         a=best[t][a];b=best[t][b];
 80
 81
         return RMQ[a] < RMQ[b]?a:b;</pre>
 82
 83
    int lcp(int a,int b){
 84
         a=rank[a];b=rank[b];
 85
         if(a>b)swap(a,b);
 86
         return height[askRMQ(a+1,b)];
 87
    }
 88
    char str[MAXN];
    int r[MAXN];
 89
    int sa[MAXN];
 91
    int main()
 92
    {
 93
         while(scanf("%s",str) == 1){
 94
             int len = strlen(str);
 95
             int n = 2*len + 1;
             for(int i = 0;i < len;i++)r[i] = str[i];</pre>
 96
 97
             for(int i = 0; i < len; i++)r[len + 1 + i] = str[len - 1 - i]
                ];
             r[len] = 1;
 98
 99
             r[n] = 0;
100
             da(r,sa,rank,height,n,128);
              for(int i=1;i<=n;i++)RMQ[i]=height[i];</pre>
101
102
             initRMQ(n);
             int ans=0,st;
103
104
             int tmp;
             for(int i=0;i<len;i++){</pre>
105
                  tmp=lcp(i,n-i);//偶对称
106
107
                  if(2*tmp>ans){
108
                      ans=2*tmp;
109
                      st=i-tmp;
110
111
                  tmp=lcp(i,n-i-1);//奇数对称
```

```
112
                 if(2*tmp-1>ans){
113
                     ans=2*tmp-1;
114
                     st=i-tmp+1;
115
                 }
             }
116
117
             str[st+ans]=0;
             printf("%s\n",str+st);
118
119
        }
120
        return 0;
121 |}
    1.5.2 DC3
    da[] 和 str[] 数组要开大三倍,相关数组也是三倍
  1 /*
  2
    * 后缀数组
     * DC3 算法, 复杂度 O(n)
    * 所有的相关数组都要开三倍
  5
     */
    const int MAXN = 2010;
  7
    #define F(x) ((x)/3+((x)%3==1?0:tb))
    #define G(x) ((x) < tb?(x) *3+1:((x)-tb) *3+2)
    | int wa[MAXN*3],wb[MAXN*3],wv[MAXN*3],wss[MAXN*3];
    int c0(int *r,int a,int b){
 10
 11
        return r[a] == r[b] \&\& r[a+1] == r[b+1] \&\& r[a+2] == r[b+2];
 12
 13
    int c12(int k,int *r,int a,int b){
        if(k == 2)
 14
 15
             return r[a] < r[b] \mid | (r[a] == r[b] && c12(1,r,a+1,b+1));
        else return r[a] < r[b] \mid | (r[a] == r[b] && wv[a+1] < wv[b+1]
 16
           );
 17
 18
    void sort(int *r,int *a,int *b,int n,int m){
 19
        int i;
 20
        for(i = 0;i < n;i++)wv[i] = r[a[i]];</pre>
 21
        for(i = 0;i < m;i++)wss[i] = 0;</pre>
        for(i = 0;i < n;i++)wss[wv[i]]++;</pre>
 22
        for(i = 1;i < m;i++)wss[i] += wss[i-1];</pre>
 23
        for(i = n-1;i >= 0;i--)
 24
 25
             b[--wss[wv[i]]] = a[i];
 26
    void dc3(int *r,int *sa,int n,int m){
 27
 28
        int i, j, *rn = r + n;
        int *san = sa + n, ta = 0, tb = (n+1)/3, tbc = 0, p;
 29
 30
        r[n] = r[n+1] = 0;
 31
        for(i = 0;i < n;i++)if(i %3 != 0)wa[tbc++] = i;</pre>
 32
        sort(r + 2, wa, wb, tbc, m);
        sort(r + 1, wb, wa, tbc, m);
 33
 34
        sort(r, wa, wb, tbc, m);
        for(p = 1, rn[F(wb[0])] = 0, i = 1;i < tbc;i++)
 35
 36
             rn[F(wb[i])] = c0(r, wb[i-1], wb[i]) ? p - 1 : p++;
 37
        if(p < tbc)dc3(rn,san,tbc,p);</pre>
```

```
38
       else for(i = 0;i < tbc;i++)san[rn[i]] = i;
39
       for(i = 0;i < tbc;i++) if(san[i] < tb)wb[ta++] = san[i] * 3;</pre>
40
       if(n \% 3 == 1)wb[ta++] = n - 1;
41
       sort(r, wb, wa, ta, m);
42
       for(i = 0;i < tbc;i++)wv[wb[i] = G(san[i])] = i;</pre>
43
       for(i = 0, j = 0, p = 0;i < ta && j < tbc;p++)
            sa[p] = c12(wb[j] % 3, r, wa[i], wb[j]) ? wa[i++] : wb[j]
44
45
       for(;i < ta;p++)sa[p] = wa[i++];</pre>
46
       for(; j < tbc; p++)sa[p] = wb[j++];</pre>
47
48
   //str 和 sa 也要三倍
49
   void da(int str[],int sa[],int rank[],int height[],int n,int m){
       for(int i = n;i < n*3;i++)
50
51
            str[i] = 0;
52
       dc3(str, sa, n+1, m);
       int i,j,k = 0;
53
       for(i = 0;i <= n;i++)rank[sa[i]] = i;</pre>
54
       for(i = 0;i < n; i++){
55
56
            if(k) k---;
            j = sa[rank[i]-1];
57
58
            while(str[i+k] == str[j+k]) k++;
59
            height[rank[i]] = k;
       }
60
61 }
        后缀自动机
   1.6
   1.6.1 基本函数
  const int CHAR = 26;
```

```
2
   const int MAXN = 250010;
 3
   struct SAM_Node{
       SAM_Node *fa,*next[CHAR];
 4
 5
       int len;
 6
       long long cnt;
 7
       void clear(){
            fa = 0;
 8
 9
            memset(next,0,sizeof(next));
10
            cnt = 0;
11
   }pool[MAXN*2];
12
13
   SAM_Node *root,*tail;
   SAM_Node* newnode(int len){
14
15
       SAM_Node* cur = tail++;
16
       cur->clear();
17
       cur->len = len;
18
       return cur;
19
20
   void SAM_init(){
21
       tail = pool;
```

```
22
        root = newnode(0);
23
24
   SAM_Node* extend(SAM_Node* last,int x){
        SAM_Node *p = last, *np = newnode(p->len+1);
25
26
        while(p && !p->next[x])
27
             p\rightarrow next[x] = np, p = p\rightarrow fa;
        if(!p)np->fa = root;
28
29
        else {
30
             SAM_Node* q = p->next[x];
             if(q\rightarrow len == p\rightarrow len+1)np\rightarrow fa = q;
31
32
             else {
                  SAM_Node* nq = newnode(p->len+1);
33
                  memcpy(nq->next,q->next,sizeof(q->next));
34
                  nq->fa = q->fa;
35
                  q\rightarrow fa = np\rightarrow fa = nq;
36
37
                  while(p && p\rightarrownext[x] == q)
38
                      p\rightarrow next[x] = nq, p = p\rightarrow fa;
39
             }
40
        }
41
        return np;
42 |}
    1.6.2 例题
    CC TSUBSTR
    给了一个 Trie 树, 求 Trie 子树上的第 k 大的子串。
   /*
 1
 2
        http://www.codechef.com/problems/TSUBSTR/
 3
   Input:
   8 4
 4
 5
   abcbbaca
 6
   1 2
   2 3
 7
 8
   1 4
 9
   4 5
   4 6
10
   4 7
11
12 | 1 8
   abcdefghijklmnopqrstuvwxyz 5
   abcdefghijklmnopgrstuvwxyz 1
14
15 | bcadefghijklmnopqrstuvwxyz 5
16
   abcdefghijklmnopqrstuvwxyz 100
17
18 Output:
19
   12
20
   aba
21
22
   ba
23
   -1
24
   */
25 | const int CHAR = 26;
```

```
26 | const int MAXN = 250010;
27
   struct SAM_Node{
        SAM_Node *fa,*next[CHAR];
28
29
        int len;
        long long cnt;
30
        void clear(){
31
32
             fa = 0;
             memset(next,0,sizeof(next));
33
34
             cnt = 0;
35
        }
   }pool[MAXN*2];
36
   SAM_Node *root,*tail;
37
38
    SAM Node* newnode(int len){
39
        SAM_Node* cur = tail++;
40
        cur->clear();
41
        cur->len = len;
42
        return cur;
43
44
   void SAM_init(){
45
        tail = pool;
46
        root = newnode(0);
47
48
   SAM_Node* extend(SAM_Node* last, int x){
        SAM_Node *p = last, *np = newnode(p->len+1);
49
50
        while(p && !p->next[x])
             p\rightarrow next[x] = np, p = p\rightarrow fa;
51
52
        if(!p)np->fa = root;
        else {
53
54
             SAM_Node* q = p->next[x];
55
             if(q->len == p->len+1)np->fa = q;
56
             else {
                 SAM_Node* nq = newnode(p->len+1);
57
                 memcpy(nq->next,q->next,sizeof(q->next));
58
59
                 nq \rightarrow fa = q \rightarrow fa;
                 q\rightarrow fa = np\rightarrow fa = nq;
60
61
                 while(p && p\rightarrownext[x] == q)
62
                      p\rightarrow next[x] = nq, p = p\rightarrow fa;
63
             }
64
        }
65
        return np;
66
67
   char str[MAXN];
   struct Edge
68
69
   {
70
        int to,next;
   }edge[MAXN*2];
71
   int head[MAXN],tot;
   void addedge(int u,int v){
73
74
        edge[tot].to = v;
75
        edge[tot].next = head[u];
76
        head[u] = tot++;
```

```
77
   }
 78
 79
    SAM_Node *end[MAXN];
    |int topcnt[MAXN];// 拓扑排序使用
 80
    |SAM_Node *topsam[MAXN*2];
 81
 82
    char s2[40];
 83
    int order[40];
 84
 85
    int main()
 86
    {
 87
         int n,Q;
         while(scanf("%d%d",&n,&Q) == 2){
 88
 89
             scanf("%s", str+1);
 90
             memset(head, -1, sizeof(head)); tot = 0;
 91
             int u,v;
 92
             for(int i = 1;i < n;i++){
                 scanf("%d%d",&u,&v);
 93
 94
                 addedge(u,v); addedge(v,u);
 95
             addedge(0,1);
 96
 97
             SAM_init();
 98
             memset(end,0,sizeof(end));
 99
             end[0] = root;
             queue<int>q;
100
             q.push(0);
101
             while(!q.empty()){
102
                 u = q.front();
103
104
                 q.pop();
105
                 for(int i = head[u];i != -1;i = edge[i].next){
106
                      v = edge[i].to;
                      if(end[v] != 0)continue;
107
                      end[v] = extend(end[u],str[v]-'a');
108
                      q.push(v);
109
                 }
110
             }
111
112
             memset(topcnt,0,sizeof(topcnt));
113
             int num = tail - pool;
114
             for(int i = 0;i < num;i++)topcnt[pool[i].len]++;</pre>
             for(int i = 1;i <= n;i++)topcnt[i] += topcnt[i-1];</pre>
115
             for(int i = 0;i < num;i++)topsam[--topcnt[pool[i].len]] = &</pre>
116
                pool[i];
117
             for(int i = num-1;i >= 0;i--){
118
                 SAM_Node *p = topsam[i];
119
120
                 p\rightarrow cnt = 1;
                 for(int i = 0;i < 26;i++)
121
                      if(p->next[i])
122
123
                          p->cnt += p->next[i]->cnt;
124
125
             printf("%lld\n",root->cnt);
             long long k;
126
```

```
127
            while(Q---){
128
                 scanf("%s",s2);
                 for(int i = 0;i < 26;i++)order[i] = s2[i]-'a';</pre>
129
                 scanf("%lld",&k);
130
                 if(k > root->cnt){
131
                     printf("-1\n");
132
                     continue;
133
134
                 }
135
                 SAM_Node *p = root;
                 //这里的第 k 个子串是从空串算起的
136
                 while ( (--k) > 0 )  {
137
                     for(int i = 0; i < 26; i++)
138
                         if(p->next[order[i]]){
139
                             if(k <= p->next[order[i]]->cnt){
140
                                 printf("%c",'a'+order[i]);
141
142
                                  p = p->next[order[i]];
                                  break; //这个不要忘记
143
144
                             }
                             else k -= p->next[order[i]]->cnt;
145
                         }
146
147
148
                 printf("\n");
149
            }
150
        }
151
        return 0;
152 | }
    CF129 E
    给了 n 个字符串, 求每个字符串有多少个至少出现在 k 个字符串中的子串
    fail 树,两遍 dfs, 经典题。
   /* http://codeforces.com/contest/204/problem/E
  1
  2
    input
  3
    3 1
  4
    abc
  5
    а
  6
    ab
  7
    output
    6 1 3
  8
 9
    input
    7 4
 10
    rubik
 11
    furik
 12
 13
    abab
 14
    baba
    aaabbbababa
 15
 16
    abababababa
 17
    zero
    output
    1 0 9 9 21 30 0
 19
 20
     */
 21 | const int CHAR = 26;
```

```
22 | const int MAXN = 100010;
23
   //**************************
24
   struct SAM_Node{
25
        SAM_Node *fa,*next[CHAR];
26
        int len;
        void clear(){
27
28
            fa = 0;
            memset(next,0,sizeof(next));
29
30
   }pool[MAXN*2];
31
32
   SAM_Node *root,*tail;
   SAM_Node* newnode(int len){
33
34
        SAM Node* cur = tail++;
35
        cur->clear();
36
        cur->len = len;
37
        return cur;
38
39
   void SAM_init(){
40
        tail = pool;
41
        root = newnode(0);
42
43
   SAM_Node* extend(SAM_Node* last,int x){
44
        SAM_Node *p = last, *np = newnode(p->len+1);
45
        while(p && !p->next[x])
46
            p\rightarrow next[x] = np, p = p\rightarrow fa;
47
        if(!p)np->fa = root;
48
        else {
49
            SAM_Node* q = p->next[x];
50
            if(q->len == p->len+1)np->fa = q;
51
            else {
52
                 SAM_Node* nq = newnode(p->len+1);
                 memcpy(nq->next,q->next,sizeof(q->next));
53
54
                 nq->fa = q->fa;
55
                 q\rightarrow fa = np\rightarrow fa = nq;
                 while(p && p\rightarrownext[x] == q)
56
57
                     p\rightarrow next[x] = nq, p = p\rightarrow fa;
58
            }
59
        }
60
        return np;
61
   //**********Trie********
62
   struct Trie_Node{
63
64
        int next[CHAR];
65
        vector<int>belongs;
66
   }trie[MAXN];
67
   int trie_root,trie_tot;
   int trie_newnode(){
68
        memset(trie[trie_tot].next,-1,sizeof(trie[trie_tot].next));
69
70
        trie[trie_tot].belongs.clear();
71
        return trie_tot++;
72 |}
```

```
73 void Trie_init(){
74
        trie_tot = 0;
75
        trie_root = trie_newnode();
76
 77
    void insert(char buf[],int id){
78
        int now = trie_root;
 79
        int len = strlen(buf);
        for(int i = 0;i < len;i++){</pre>
 80
             if(trie[now].next[buf[i]-'a'] == -1)
 81
                 trie[now].next[buf[i]-'a'] = trie_newnode();
 82
 83
             now = trie[now].next[buf[i]-'a'];
             trie[now].belongs.push_back(id);
 84
        }
 85
 86
    //**** fail 树**********
 87
 88
    struct Edge{
 89
        int to,next;
 90
    }edge[MAXN*2];
    int head[MAXN*2],tot;
91
    void addedge(int u,int v){
92
93
        edge[tot].to = v; edge[tot].next = head[u]; head[u] = tot++;
 94
95
    int MtoT[MAXN*2];//SAM 结点映射到 Trie 结点
    int cnt[MAXN*2];
97
    int F[MAXN*2];
    int find(int x){
98
99
        if(F[x] == -1)return x;
100
        return F[x] = find(F[x]);
101
    }
102
    void bing(int u,int v)//注意方向性
103
        int t1 = find(u);
104
        int t2 = find(v);
105
106
        if(t1 != t2)F[t1] = t2;
107
108
    int L[MAXN];
109
    void Tarjan(int u){
110
        for(int i = head[u];i != -1;i = edge[i].next){
111
             Tarjan(edge[i].to);
112
            bing(edge[i].to,u);
113
        if(MtoT[u]){
114
             int tt = MtoT[u];
115
             int sz = trie[tt].belongs.size();
116
             for(int i = 0;i < sz;i++){
117
                 int v = trie[tt].belongs[i];
118
                 cnt[find(L[v])]--;
119
                 cnt[u]++;
120
121
                 L[v] = u;
122
             }
        }
123
```

```
124
    }
125
    void dfs1(int u){
        for(int i = head[u]; i != -1; i = edge[i].next){
126
127
             dfs1(edge[i].to);
128
             cnt[u] += cnt[edge[i].to];
129
        }
130
    long long ans[MAXN];
131
132
    void dfs2(int u){
133
        for(int i = head[u];i != −1;i = edge[i].next){
134
             int v = edge[i].to;
135
             cnt[v] += cnt[u];
             dfs2(v);
136
137
        if(MtoT[u]){
138
139
             int tt = MtoT[u];
             int sz = trie[tt].belongs.size();
140
             for(int i = 0;i < sz;i++){
141
                 int v = trie[tt].belongs[i];
142
                 ans[v] += cnt[u];
143
             }
144
145
        }
    }
146
147
148
    char str[MAXN];
    SAM_Node *end[MAXN];
149
150
    int main()
151
    {
152
        int n,k;
153
        while(scanf("%d%d",&n,&k) == 2){
154
             Trie_init();
155
             for(int i = 0;i < n;i++){
                 scanf("%s",str);
156
                 insert(str,i);
157
             }
158
159
             SAM_init();
160
             //根据 Trie 建立 SAM
             memset(end,0,sizeof(end));
161
             end[0] = root;
162
             memset(MtoT,0,sizeof(MtoT));
163
             MtoT[root—pool] = 0;
164
             queue<int>q;
165
             q.push(trie_root);
166
             while(!q.empty()){
167
                 int u = q.front();
168
                 q.pop();
169
                 for(int i = 0;i < 26;i++){
170
                     if(trie[u].next[i] == −1)continue;
171
                     int v = trie[u].next[i];
172
173
                     end[v] = extend(end[u],i);
                     MtoT[end[v]-pool] = v;
174
```

```
175
                      q.push(v);
176
                  }
              }
177
              //建立 fail 树
178
              int num = tail - pool;
179
             memset(head,-1,sizeof(head));
180
              tot = 0;
181
              for(SAM_Node *p = pool+1;p < tail;p++)</pre>
182
183
                  addedge(p->fa - pool,p - pool);
             memset(cnt,0,sizeof(cnt));
184
             memset(F,-1,sizeof(F));
185
             memset(L,0,sizeof(L));
186
             Tarjan(0);
187
              dfs1(0);
188
              for(int i = 0;i < num;i++){</pre>
189
                  if(cnt[i] >= k)cnt[i] = pool[i].len - pool[i].fa->len;
190
191
                  else cnt[i] = 0;
192
              }
             memset(ans,0,sizeof(ans));
193
194
              dfs2(0);
              for(int i = 0;i < n;i++){</pre>
195
196
                  printf("%I64d",ans[i]);
197
                  if(i < n−1)printf("<sub>□</sub>");
                  else printf("\n");
198
199
              }
         }
200
         return 0;
201
202 |}
```

## 1.7 字符串 hash

HDU4622 求区间不相同子串个数

```
const int HASH = 10007;
 2
   const int MAXN = 2010;
 3
   struct HASHMAP{
       int head[HASH],next[MAXN],size;
 4
 5
       unsigned long long state[MAXN];
       int f[MAXN];
 6
       void init(){
 7
            size = 0;
 8
 9
            memset(head,-1,sizeof(head));
10
       int insert(unsigned long long val,int _id){
11
12
            int h = val%HASH;
            for(int i = head[h]; i != -1;i = next[i])
13
                if(val == state[i]){
14
                    int tmp = f[i];
15
                    f[i] = _id;
16
                    return tmp;
17
18
19
            f[size] = _id;
```

```
20
            state[size] = val;
21
            next[size] = head[h];
22
            head[h] = size++;
23
            return 0;
24
        }
   }H;
25
   const int SEED = 13331;
26
   unsigned long long P[MAXN];
28
   unsigned long long S[MAXN];
29
   char str[MAXN];
   int ans[MAXN][MAXN];
30
31
   int main(){
        P[0] = 1;
32
33
        for(int i = 1;i < MAXN;i++)</pre>
            P[i] = P[i-1] * SEED;
34
35
        int T;
        scanf("%d",&T);
36
37
        while (T--)
            scanf("%s",str);
38
39
            int n = strlen(str);
            S[0] = 0;
40
41
            for(int i = 1;i <= n;i++)</pre>
                S[i] = S[i-1]*SEED + str[i-1];
42
43
            memset(ans,0,sizeof(ans));
            for(int L = 1; L <= n;L++){
44
45
                H.init();
                for(int i = 1; i + L - 1 \le n; i++){
46
47
                     int l = H.insert(S[i+L-1] - S[i-1]*P[L],i);
48
                     ans[i][i+L-1] ++;
49
                     ans[l][i+L-1]--;
50
                }
51
            }
            for(int i = n;i >= 0;i—)
52
53
                for(int j = i; j <= n; j++)
                     ans[i][j] += ans[i+1][j] + ans[i][j-1] - ans[i+1][j
54
                        -1;
            int m,u,v;
55
            scanf("%d",&m);
56
57
            while(m——){
58
                scanf("%d%d",&u,&v);
                printf("%d\n",ans[u][v]);
59
60
            }
61
        }
        return 0;
62
63 |}
```

## 2 数学

#### 2.1 素数

2.1.1 素数筛选(判断 < MAXN 的数是否素数)

```
1
    * 素数筛选, 判断小于 MAXN 的数是不是素数。
2
    * notprime 是一张表, 为 false 表示是素数, true 表示不是素数
3
4
    */
   const int MAXN=1000010;
5
   bool notprime[MAXN];//值为 false 表示素数,值为 true 表示非素数
7
   void init(){
8
       memset(notprime, false, sizeof(notprime));
9
       notprime[0]=notprime[1]=true;
       for(int i=2;i<MAXN;i++)</pre>
10
           if(!notprime[i]){
11
              if(i>MAXN/i)continue;//防止后面 i*i 溢出 (或者 i,j 用 long
12
              //直接从 i*i 开始就可以, 小于 i 倍的已经筛选过了, 注意是 j+=i
13
14
              for(int j=i*i;j<MAXN;j+=i)</pre>
15
                  notprime[j]=true;
16
           }
  }
17
   2.1.2 素数筛选(筛选出小于等于 MAXN 的素数)
  /*
1
2
   * 素数筛选,存在小于等于 MAXN 的素数
3
    * prime[0] 存的是素数的个数
4
   */
   const int MAXN=10000;
   int prime[MAXN+1];
7
   void getPrime(){
       memset(prime,0,sizeof(prime));
8
9
       for(int i=2;i<=MAXN;i++){</pre>
           if(!prime[i])prime[++prime[0]]=i;
10
11
           for(int j=1;j<=prime[0]&&prime[j]<=MAXN/i;j++){</pre>
              prime[prime[j]*i]=1;
12
13
              if(i%prime[j]==0) break;
14
           }
15
       }
16 | }
   2.1.3 大区间素数筛选(POJ 2689)
  /*
1
2
    * POJ 2689 Prime Distance
    * 给出一个区间 [L,U], 找出区间内容、相邻的距离最近的两个素数和
    * 距离最远的两个素数。
    * 1<=L<U<=2,147,483,647 区间长度不超过 1,000,000
5
    * 就是要筛选出 [L,U] 之间的素数
6
7
    */
```

```
const int MAXN=100010;
 9
   int prime[MAXN+1];
   void getPrime(){
10
        memset(prime,0,sizeof(prime));
11
12
        for(int i=2;i<=MAXN;i++){</pre>
13
            if(!prime[i])prime[++prime[0]]=i;
            for(int j=1;j<=prime[0]&&prime[j]<=MAXN/i;j++){</pre>
14
                 prime[prime[j]*i]=1;
15
16
                 if(i%prime[j]==0)break;
17
            }
        }
18
19
20
   bool notprime[1000010];
21
   int prime2[1000010];
   void getPrime2(int L,int R){
22
        memset(notprime, false, sizeof(notprime));
23
24
        if(L<2)L=2;
25
        for(int i=1;i<=prime[0]&&(long long)prime[i]*prime[i]<=R;i++){</pre>
26
            int s=L/prime[i]+(L%prime[i]>0);
            if(s==1)s=2;
27
28
            for(int j=s;(long long)j*prime[i]<=R;j++)</pre>
29
                 if((long long)j*prime[i]>=L)
30
                     notprime[j*prime[i]-L]=true;
        }
31
32
        prime2[0]=0;
33
        for(int i=0;i<=R-L;i++)</pre>
34
            if(!notprime[i])
35
                 prime2[++prime2[0]]=i+L;
36
   int main(){
37
38
        getPrime();
39
        int L,U;
40
        while(scanf("%d%d",&L,&U)==2){
            getPrime2(L,U);
41
42
            if(prime2[0]<2)printf("There are no adjacent primes.\n");</pre>
43
            else{
44
                 int x1=0,x2=100000000,y1=0,y2=0;
                 for(int i=1;i<prime2[0];i++){</pre>
45
46
                     if(prime2[i+1]-prime2[i]<x2-x1){
47
                         x1=prime2[i];
48
                         x2=prime2[i+1];
49
50
                     if(prime2[i+1]-prime2[i]>y2-y1){
51
                         y1=prime2[i];
52
                         y2=prime2[i+1];
                     }
53
54
55
                 printf("%d,%d⊔areuclosest,u%d,%d⊔areumostudistant.\n",
                    x1, x2, y1, y2);
56
            }
        }
57
```

```
58 |}
```

## 2.2 素数筛选和合数分解

```
//*************
 2
   //素数筛选和合数分解
   const int MAXN=10000;
  int prime[MAXN+1];
   void getPrime(){
 5
 6
       memset(prime,0,sizeof(prime));
 7
       for(int i=2;i<=MAXN;i++){</pre>
 8
           if(!prime[i])prime[++prime[0]]=i;
           for(int j=1;j<=prime[0]&&prime[j]<=MAXN/i;j++){</pre>
 9
               prime[prime[j]*i]=1;
10
               if(i%prime[j]==0) break;
11
12
           }
13
       }
14
15
   long long factor[100][2];
   int fatCnt;
16
   int getFactors(long long x){
17
       fatCnt=0;
18
       long long tmp=x;
19
20
       for(int i=1;prime[i]<=tmp/prime[i];i++){</pre>
21
           factor[fatCnt][1]=0;
22
           if(tmp%prime[i]==0){
               factor[fatCnt][0]=prime[i];
23
              while(tmp%prime[i]==0){
24
                   factor[fatCnt][1]++;
25
26
                  tmp/=prime[i];
27
28
               fatCnt++;
29
           }
30
       if(tmp!=1){
31
           factor[fatCnt][0]=tmp;
32
           factor[fatCnt++][1]=1;
33
34
35
       return fatCnt;
36
37
   //****************
        扩展欧几里得算法(求 ax+by=gcd 的解以及逆元)
   2.3
  //*********
 2
   //返回 d=gcd(a,b); 和对应于等式 ax+by=d 中的 x,y
 3
   long long extend_gcd(long long a,long long b,long long &x,long long
       &v){
       if(a==0&&b==0) return −1;//无最大公约数
 4
 5
       if(b==0){x=1;y=0;return a;}
 6
       long long d=extend_gcd(b,a%b,y,x);
 7
       y=a/b*x;
```

```
8
       return d;
9
   //********* 求逆元 **************
10
   //ax = 1 \pmod{n}
11
   long long mod_reverse(long long a,long long n){
12
13
       long long x,y;
       long long d=extend\_gcd(a,n,x,y);
14
       if(d==1) return (x%n+n)%n;
15
16
       else return -1;
17 | }
   2.4 求逆元
   2.4.1 扩展欧几里德法
   见上面的写法
   2.4.2 简洁写法
   注意: 这个只能求 a < m 的情况,而且必须保证 a 和 m 互质
1 \mid //x ax = 1( mod m) 的 x 值, 就是逆元 (0<a<m)
  long long inv(long long a,long long m){
3
       if(a == 1)return 1;
4
       return inv(m%a,m)*(m-m/a)%m;
5 |}
   2.4.3 利用欧拉函数
   mod 为素数, 而且 a 和 m 互质
1 | long long inv(long long a, long long mod)//为素数mod
2
   {
3
       return pow_m(a,mod-2,mod);
4 | }
        模线性方程组
   2.5
  long long extend_gcd(long long a,long long b,long long &x,long long
        &y){
2
       if(a == 0 && b == 0)return -1;
       if(b ==0 ){x = 1; y = 0; return a;}
3
4
       long long d = extend_gcd(b,a%b,y,x);
5
       y = a/b*x;
6
       return d;
7
   int m[10],a[10];//模数为 m, 余数为 a,X % m = a
9
   bool solve(int &m0,int &a0,int m,int a){
10
       long long y,x;
       int g = extend_gcd(m0,m,x,y);
11
12
       if (abs(a - a0)\%g) return false;
       x *= (a - a0)/g;
13
14
       x \% = m/g;
```

```
15
      a0 = (x*m0 + a0);
16
      m0 \times = m/g;
17
      a0 \% = m0;
      if( a0 < 0 )a0 += m0;
18
19
      return true;
20
  }
21
   /*
    * 无解返回 false, 有解返回 true;
22
    * 解的形式最后为 a0 + m0 * t (0<=a0<m0)
23
24
   */
   bool MLES(int &m0 ,int &a0,int n)//解为 X = a0 + m0 * k
25
26
27
      bool flag = true;
28
      m0 = 1;
29
      a0 = 0;
30
      for(int i = 0;i < n;i++)
31
          if( !solve(m0,a0,m[i],a[i]) )
32
      {
33
          flag = false;
34
          break;
35
36
      return flag;
37 |}
   2.6
       随机素数测试和大数分解 (POJ 1811)
  * Miller Rabin 算法进行素数测试
2
3
    * 速度快可以判断一个 < 2^63 的数是不是素数
4
5
   ******************
   const int S = 8; //随机算法判定次数一般 8\sim10 就够了
   // 计算 ret = (a*b)%c
7
                          a,b,c < 2^{63}
8
   long long mult_mod(long long a,long long b,long long c){
9
      a %= c;
10
      b %= c;
      long long ret = 0;
11
12
      long long tmp = a;
      while(b){
13
          if(b & 1){
14
15
              ret += tmp;
              if(ret > c)ret -= c;//直接取模慢很多
16
17
          }
18
          tmp <<= 1;
19
          if(tmp > c)tmp -= c;
20
          b >>= 1;
21
22
      return ret;
23
   }
   // 计算 ret = (a^n)%mod
25
   long long pow_mod(long long a,long long n,long long mod){
26
      long long ret = 1;
```

```
long long temp = a%mod;
27
28
      while(n){
          if(n & 1)ret = mult_mod(ret,temp,mod);
29
30
          temp = mult_mod(temp,temp,mod);
          n >>= 1;
31
32
33
      return ret;
34
  }
35
  |// 通过 a^(n—1)=1(mod n)来判断 n 是不是素数
  // n-1 = x * 2^t 中间使用二次判断
36
  // 是合数返回 true, 不一定是合数返回 false
37
  bool check(long long a,long long n,long long x,long long t) {
38
39
      long long ret = pow_mod(a,x,n);
40
      long long last = ret;
41
      for(int i = 1;i <= t;i++){
42
          ret = mult_mod(ret,ret,n);
          if(ret == 1 && last != 1 && last != n-1)return true;//合数
43
44
          last = ret;
45
46
      if(ret != 1)return true;
47
      else return false;
48
  }
49
  //*************
  // Miller_Rabin 算法
50
  // 是素数返回 true,(可能是伪素数)
  // 不是素数返回 false
52
  //**************
53
  bool Miller_Rabin(long long n){
54
55
      if( n < 2)return false;</pre>
      if( n == 2)return true;
56
57
      if((n&1) == 0)return false;//偶数
      long long x = n - 1;
58
      long long t = 0;
59
60
      while (x\&1)==0 x >>= 1; t++;
61
      62
63
64
      for(int i = 0;i < S;i++){
65
          long long a = rand()\%(n-1) + 1;
66
          if( check(a,n,x,t) )
67
             return false;
68
69
      return true;
70
  }
71
72
  //****************
  // pollard_rho 算法进行质因素分解
  //************
74
75
  long long factor[100];//质因素分解结果(刚返回时时无序的)
  int tol; //质因素的个数,编号 0\simtol-1
76
77
```

```
78
    long long gcd(long long a,long long b){
 79
        long long t;
 80
        while(b){
 81
            t = a;
            a = b;
 82
 83
            b = t\%b;
 84
        if(a >= 0)return a;
 85
 86
        else return -a;
 87
    }
 88
 89
     //找出一个因子
90
    long long pollard_rho(long long x,long long c){
91
        long long i = 1, k = 2;
 92
        srand(time(NULL));
93
        long long x0 = rand()\%(x-1) + 1;
94
        long long y = x0;
        while(1){
 95
96
            i ++;
97
            x0 = (mult_mod(x0, x0, x) + c)%x;
            long long d = gcd(y - x0,x);
 98
99
            if( d != 1 && d != x)return d;
100
            if(y == x0) return x;
101
            if(i == k){y = x0; k += k;}
102
        }
    }
103
     //对 n 进行素因子分解,存入 factor. k 设置为 107 左右即可
104
    void findfac(long long n,int k){
105
106
        if(n == 1)return;
107
        if(Miller_Rabin(n))
108
        {
109
            factor[tol++] = n;
110
            return;
111
        long long p = n;
112
113
        int c = k;
114
        while(p >= n)p = pollard_rho(p,c--);//值变化,防止死循环 k
115
        findfac(p,k);
        findfac(n/p,k);
116
117
118
     //POJ 1811
     //给出一个N(2 \le N \le 2^{54}),如果是素数,输出"Prime",否则输出最小的素因子
119
120
    int main(){
        int T;
121
122
        long long n;
        scanf("%d",&T);
123
124
        while (T--)
            scanf("%I64d",&n);
125
126
            if(Miller_Rabin(n))printf("Prime\n");
127
            else{
128
                tol = 0;
```

```
129
                 findfac(n,107);
130
                 long long ans = factor[0];
                 for(int i = 1;i < tol;i++)</pre>
131
                 ans = min(ans,factor[i]);
132
                 printf("%I64d\n",ans);
133
             }
134
135
         }
136
        return 0;
137 |}
         欧拉函数
    2.7
    2.7.1
         分解质因素求欧拉函数
   |getFactors(n);
    int ret = n;
  2
    for(int i = 0;i < fatCnt;i++){
         ret = ret/factor[i][0]*(factor[i][0]-1);
  5 |}
    2.7.2 筛法欧拉函数
   | int euler[3000001];
  1
  2
    void getEuler(){
  3
        memset(euler,0,sizeof(euler));
  4
        euler[1] = 1;
  5
        for(int i = 2;i <= 3000000;i++)
  6
             if(!euler[i])
  7
                for(int j = i; j <= 3000000; j += i){</pre>
  8
                    if(!euler[j])
  9
                        euler[j] = j;
 10
                    euler[j] = euler[j]/i*(i-1);
                }
 11
 12 |}
    2.7.3 求单个数的欧拉函数
    long long eular(long long n){
  1
        long long ans = n;
  2
  3
        for(int i = 2;i*i <= n;i++){</pre>
             if(n % i == 0){
  4
  5
                 ans -= ans/i;
  6
                 while(n % i == 0)
  7
                     n /= i;
  8
             }
  9
        if(n > 1)ans -= ans/n;
 10
 11
        return ans;
 12 | }
    2.7.4 线性筛(同时得到欧拉函数和素数表)
  1 const int MAXN = 10000000;
  2 | bool check[MAXN+10];
```

```
|int phi[MAXN+10];
 4
   int prime[MAXN+10];
 5
   int tot;//素数的个数
   void phi_and_prime_table(int N){
 7
       memset(check, false, sizeof(check));
 8
       phi[1] = 1;
 9
       tot = 0;
       for(int i = 2; i <= N; i++){
10
11
            if(!check[i]){
                prime[tot++] = i;
12
13
                phi[i] = i-1;
14
15
            for(int j = 0; j < tot; j++){
                if(i * prime[j] > N)break;
16
                check[i * prime[j]] = true;
17
18
                if( i % prime[j] == 0){
                    phi[i * prime[j]] = phi[i] * prime[j];
19
20
                    break;
21
                }
22
                else{
23
                    phi[i * prime[j]] = phi[i] * (prime[j] - 1);
24
                }
25
            }
       }
26
27 }
   2.8
        高斯消元 (浮点数)
 1 | #define eps 1e-9
   const int MAXN=220;
   |double a[MAXN][MAXN],x[MAXN];//方程的左边的矩阵和等式右边的值, 求解之后 x
      存的就是结果
   |int equ,var;//方程数和未知数个数
 5
   /*
 6
   * 返回 0 表示无解,1 表示有解
 7
   */
   int Gauss(){
 8
9
       int i,j,k,col,max_r;
10
       for(k=0,col=0;k<equ&&col<var;k++,col++){</pre>
11
            max_r=k;
            for(i=k+1;i<equ;i++)</pre>
12
              if(fabs(a[i][col])>fabs(a[max_r][col]))
13
14
                max_r=i;
15
            if(fabs(a[max_r][col])<eps)return 0;</pre>
16
            if(k!=max_r){
17
                for(j=col;j<var;j++)</pre>
18
                  swap(a[k][j],a[max_r][j]);
19
                swap(x[k],x[max_r]);
20
            }
21
            x[k]/=a[k][col];
22
            for(j=col+1;j<var;j++)a[k][j]/=a[k][col];</pre>
23
            a[k][col]=1;
```

```
for(i=0;i<equ;i++)</pre>
24
25
              if(i!=k){
26
                  x[i]=x[k]*a[i][col];
27
                  for(j=col+1; j < var; j++)a[i][j] = a[k][j] * a[i][col];</pre>
28
                  a[i][col]=0;
29
             }
30
31
       return 1;
32 |}
   2.9
        \mathbf{FFT}
  |//HDU 1402 求高精度乘法
   const double PI = acos(-1.0);
 3
   //复数结构体
   struct Complex{
 4
 5
       double x,y;//实部和虚部 x+yi
 6
       Complex(double _x = 0.0, double _y = 0.0)
            x = _x;
 7
 8
           y = _y;
 9
       Complex operator -(const Complex &b)const{
10
11
            return Complex(x-b.x,y-b.y);
12
13
       Complex operator +(const Complex &b)const{
14
            return Complex(x+b.x,y+b.y);
15
       Complex operator *(const Complex &b)const{
16
17
            return Complex(x*b.x-y*b.y,x*b.y+y*b.x);
18
       }
19
   };
20
   /*
    * 进行 FFT 和 IFFT 前的反转变换。
21
    * 位置 i 和(i 二进制反转后位置)互换
22
23
    * len 必须为 2 的幂
24
    */
25
   void change(Complex y[],int len){
26
       int i,j,k;
27
       for(i = 1, j = len/2; i < len-1; i++){
28
            if(i < j)swap(y[i],y[j]);
            //交换互为小标反转的元素, i<i 保证交换一次
29
            //i 做正常的 +1, j 左反转类型的 +1, 始终保持 i 和 j 是反转的
30
31
           k = len/2;
32
           while(j >= k){
33
                j -= k;
                k /= 2;
34
35
36
           if(j < k)j += k;
37
       }
   }
38
39
   /*
40
      做 FFT
```

```
41
    * len 必须为2^k形式
    * on==1 时是 DFT, on==-1 时是 IDFT
42
43
    */
   void fft(Complex y[],int len,int on){
44
45
        change(y,len);
        for(int h = 2; h <= len; h <<= 1){</pre>
46
47
            Complex wn(cos(-on*2*PI/h),sin(-on*2*PI/h));
48
            for(int j = 0; j < len; j+=h){
49
                 Complex w(1,0);
                 for(int k = j; k < j+h/2; k++){
50
                     Complex u = y[k];
51
                     Complex t = w*y[k+h/2];
52
53
                     y[k] = u+t;
54
                     y[k+h/2] = u-t;
55
                     w = w*wn;
56
                 }
            }
57
58
        }
        if(on == -1)
59
            for(int i = 0;i < len;i++)</pre>
60
61
                 y[i].x /= len;
62
   }
63
   const int MAXN = 200010;
   Complex x1[MAXN],x2[MAXN];
65
   char str1[MAXN/2],str2[MAXN/2];
   int sum[MAXN];
66
   int main(){
67
        while(scanf("%s%s",str1,str2)==2){
68
69
            int len1 = strlen(str1);
70
            int len2 = strlen(str2);
            int len = 1;
71
72
            while(len < len1*2 || len < len2*2)len<<=1;</pre>
            for(int i = 0;i < len1;i++)</pre>
73
74
                 x1[i] = Complex(str1[len1-1-i]-'0',0);
75
            for(int i = len1;i < len;i++)</pre>
76
                 x1[i] = Complex(0,0);
            for(int i = 0;i < len2;i++)</pre>
77
                 x2[i] = Complex(str2[len2-1-i]-'0',0);
78
79
            for(int i = len2;i < len;i++)</pre>
80
                 x2[i] = Complex(0,0);
81
            //求 DFT
            fft(x1, len, 1);
82
83
            fft(x2,len,1);
            for(int i = 0;i < len;i++)</pre>
84
                 x1[i] = x1[i]*x2[i];
85
            fft(x1, len, -1);
86
            for(int i = 0;i < len;i++)</pre>
87
                 sum[i] = (int)(x1[i].x+0.5);
88
89
            for(int i = 0;i < len;i++){</pre>
90
                 sum[i+1]+=sum[i]/10;
91
                 sum[i]%=10;
```

```
92
             }
 93
             len = len1 + len2 - 1;
             while(sum[len] <= 0 && len > 0)len—;
 94
 95
             for(int i = len;i >= 0;i---)
                 printf("%c",sum[i]+'0');
 96
 97
             printf("\n");
 98
         }
 99
         return 0;
100
    }
101
    //HDU 4609
102
    //给出 n 条线段长度,问任取 3 根,组成三角形的概率。
103
    //n<=10^5
                 用 FFT 求可以组成三角形的取法有几种
104
105
    const int MAXN = 400040;
106
    Complex x1[MAXN];
107
    int a[MAXN/4];
108
    long long num[MAXN];//100000*100000 会超 int
109
    long long sum[MAXN];
    int main(){
110
111
         int T;
         int n;
112
113
         scanf("%d",&T);
114
         while (T--)
             scanf("%d",&n);
115
             memset(num,0,sizeof(num));
116
             for(int i = 0;i < n;i++){
117
                 scanf("%d",&a[i]);
118
119
                 num[a[i]]++;
120
             }
121
             sort(a,a+n);
             int len1 = a[n-1]+1;
122
             int len = 1;
123
             while( len < 2*len1 )len <<= 1;</pre>
124
             for(int i = 0;i < len1;i++)</pre>
125
                 x1[i] = Complex(num[i],0);
126
             for(int i = len1;i < len;i++)</pre>
127
128
                 x1[i] = Complex(0,0);
129
             fft(x1, len, 1);
             for(int i = 0;i < len;i++)</pre>
130
131
                 x1[i] = x1[i]*x1[i];
             fft(x1, len, -1);
132
             for(int i = 0;i < len;i++)</pre>
133
                 num[i] = (long long)(x1[i].x+0.5);
134
             len = 2*a[n-1];
135
             //减掉取两个相同的组合
136
             for(int i = 0;i < n;i++)
137
138
                 num[a[i]+a[i]]--;
             for(int i = 1;i <= len;i++)num[i]/=2;</pre>
139
140
             sum[0] = 0;
141
             for(int i = 1;i <= len;i++)</pre>
142
                 sum[i] = sum[i-1]+num[i];
```

```
long long cnt = 0;
143
144
            for(int i = 0;i < n;i++){</pre>
               cnt += sum[len]-sum[a[i]];
145
                //减掉一个取大,一个取小的
146
               cnt -= (long long)(n-1-i)*i;
147
               //减掉一个取本身,另外一个取其它
148
               cnt -= (n-1);
149
               cnt -= (long long) (n-1-i)*(n-i-2)/2;
150
151
            }
            long long tot = (long long)n*(n-1)*(n-2)/6;
152
            printf("%.7lf\n",(double)cnt/tot);
153
154
        }
155
        return 0;
156 }
          高斯消元法求方程组的解
    2.10
    2.10.1 一类开关问题,对 2 取模的 01 方程组
    POJ 1681 需要枚举自由变元, 找解中 1 个数最少的
 1 / /对 2 取模的 01 方程组
   const int MAXN = 300;
   │//有 equ 个方程,var 个变元。增广矩阵行数为 equ,列数为 var+1,分别为 0 到
 4 | int equ, var;
 5 | int a[MAXN][MAXN]; //增广矩阵
   |int x[MAXN]; //解集
   |int free_x[MAXN];//用来存储自由变元(多解枚举自由变元可以使用)
    int free num;//自由变元的个数
 9
    //返回值为 -1 表示无解, 为 0 是唯一解, 否则返回自由变元个数
    int Gauss(){
 11
        int max_r,col,k;
 12
 13
        free_num = 0;
        for(k = 0, col = 0 ; k < equ && col < var ; k++, col++){
 14
 15
           max r = k;
            for(int i = k+1;i < equ;i++){
 16
                if(abs(a[i][col]) > abs(a[max r][col]))
 17
 18
                   max_r = i;
 19
            if(a[max r][col] == 0){
 20
 21
               k---;
                free_x[free_num++] = col;//这个是自由变元
 22
 23
               continue;
24
            if(max_r != k){
 25
 26
               for(int j = col; j < var+1; j++)</pre>
 27
                   swap(a[k][j],a[max_r][j]);
 28
 29
            for(int i = k+1;i < equ;i++){
                if(a[i][col] != 0){
 30
                   for(int j = col; j < var+1; j++)
 31
```

```
32
                         a[i][j] ^= a[k][j];
33
                }
34
            }
35
        for(int i = k;i < equ;i++)
36
37
            if(a[i][col] != 0)
                return -1;//无解
38
39
        if(k < var) return var—k;//自由变元个数
40
        //唯一解,回代
41
        for(int i = var-1; i >= 0;i--){
42
            x[i] = a[i][var];
43
            for(int j = i+1; j < var; j++)
44
                x[i] ^= (a[i][j] \&\& x[j]);
45
46
        return 0;
47
48
   int n;
49
   void init(){
50
        memset(a,0,sizeof(a));
51
        memset(x,0,sizeof(x));
52
        equ = n*n;
53
        var = n*n;
54
        for(int i = 0;i < n;i++)
            for(int j = 0; j < n; j++){
55
                int t = i*n+j;
56
                a[t][t] = 1;
57
58
                if(i > 0)a[(i-1)*n+j][t] = 1;
                if(i < n-1)a[(i+1)*n+j][t] = 1;
59
60
                if(j > 0)a[i*n+j-1][t] = 1;
                if(j < n-1)a[i*n+j+1][t] = 1;
61
62
            }
63
64
   void solve(){
65
        int t = Gauss();
66
        if(t == -1){
67
            printf("inf\n");
68
            return;
69
70
        else if(t == 0){
71
            int ans = 0;
72
            for(int i = 0;i < n*n;i++)
73
                ans += x[i];
74
            printf("%d\n",ans);
75
            return;
76
        }
        else
77
78
        {
79
            //枚举自由变元
            int ans = 0x3f3f3f3f;
80
81
            int tot = (1 << t);
82
            for(int i = 0;i < tot;i++){
```

```
83
                  int cnt = 0;
                  for(int j = 0;j < t;j++){</pre>
 84
 85
                      if(i&(1<<j)){
 86
                           x[free_x[j]] = 1;
 87
                           cnt++;
                      }
 88
 89
                      else x[free_x[j]] = 0;
 90
 91
                  for(int j = var-t-1; j >= 0; j--){
 92
                      int idx;
                      for(idx = j;idx < var;idx++)</pre>
 93
 94
                           if(a[j][idx])
 95
                               break;
                      x[idx] = a[j][var];
 96
                      for(int l = idx+1;l < var;l++)</pre>
 97
 98
                           if(a[j][l])
 99
                               x[idx] ^= x[l];
100
                      cnt += x[idx];
101
                  }
102
                  ans = min(ans,cnt);
103
104
             printf("%d\n",ans);
105
         }
106
107
    char str[30][30];
    int main(){
108
         int T;
109
         scanf("%d",&T);
110
111
         while (T--)
112
             scanf("%d",&n);
113
             init();
             for(int i = 0;i < n;i++){
114
                  scanf("%s",str[i]);
115
116
                  for(int j = 0; j < n; j++){}
                      if(str[i][j] == 'y')
117
118
                           a[i*n+j][n*n] = 0;
119
                      else a[i*n+j][n*n] = 1;
120
                  }
121
             }
122
             solve();
123
         }
124
         return 0;
125 |}
    2.10.2 解同余方程组
    POJ 2947 Widget Factory
  1 | / / 求解对 MOD 取模的方程组
    | const int MOD = 7;
    const int MAXN = 400;
    |int a[MAXN][MAXN];//增广矩阵
```

```
|int x[MAXN];//最后得到的解集
 6
   inline int gcd(int a,int b){
 7
        while(b != 0){
            int t = b;
 8
 9
            b = a\%b;
10
            a = t;
11
12
        return a;
13
14
   inline int lcm(int a,int b){
        return a/gcd(a,b)*b;
15
16
17
   long long inv(long long a,long long m){
        if(a == 1)return 1;
18
        return inv(m%a,m)*(m-m/a)%m;
19
20
   int Gauss(int equ,int var){
21
22
        int max_r,col,k;
23
        for(k = 0, col = 0; k < equ && col < var; k++,col++){
24
            max r = k;
25
            for(int i = k+1; i < equ;i++)
26
                if(abs(a[i][col]) > abs(a[max_r][col]))
27
                    max_r = i;
            if(a[max_r][col] == 0){
28
29
                k---;
                continue;
30
31
            if(max_r != k)
32
33
                for(int j = col; j < var+1; j++)</pre>
34
                    swap(a[k][j],a[max_r][j]);
            for(int i = k+1;i < equ;i++){</pre>
35
                if(a[i][col] != 0){
36
                     int LCM = lcm(abs(a[i][col]),abs(a[k][col]));
37
38
                    int ta = LCM/abs(a[i][col]);
39
                    int tb = LCM/abs(a[k][col]);
40
                    if(a[i][col]*a[k][col] < 0)tb = -tb;
41
                     for(int j = col; j < var+1; j++)
42
                         a[i][j] = ((a[i][j]*ta - a[k][j]*tb)%MOD + MOD)
                            %MOD;
43
                }
            }
44
45
46
        for(int i = k;i < equ;i++)
47
            if(a[i][col] != 0)
48
                return -1;//无解
49
        if(k < var) return var-k;//多解
        for(int i = var-1;i >= 0;i--){
50
            int temp = a[i][var];
51
            for(int j = i+1; j < var;j++){</pre>
52
53
                if(a[i][j] != 0){
54
                    temp -= a[i][j]*x[j];
```

```
55
                      temp = (temp%MOD + MOD)%MOD;
 56
                  }
 57
             }
 58
             x[i] = (temp*inv(a[i][i],MOD))%MOD;
 59
         }
 60
         return 0;
 61
 62
    int change(char s[]){
         if(strcmp(s,"MON") == 0) return 1;
 63
 64
         else if(strcmp(s,"TUE")==0) return 2;
         else if(strcmp(s,"WED")==0) return 3;
 65
         else if(strcmp(s,"THU")==0) return 4;
 66
         else if(strcmp(s,"FRI")==0) return 5;
 67
         else if(strcmp(s,"SAT")==0) return 6;
 68
 69
         else return 7;
 70
    }
 71
    int main(){
 72
         int n,m;
         while(scanf("%d%d",&n,&m) == 2){
 73
 74
             if(n == 0 && m == 0)break;
             memset(a,0,sizeof(a));
 75
 76
             char str1[10],str2[10];
 77
             int k;
             for(int i = 0;i < m;i++){</pre>
 78
 79
                  scanf("%d%s%s",&k,str1,str2);
                  a[i][n] = ((change(str2) - change(str1) + 1)%MOD + MOD)
 80
                     %MOD;
                  int t;
 81
 82
                  while(k--){
 83
                      scanf("%d",&t);
 84
                      t---;
 85
                      a[i][t] ++;
 86
                      a[i][t]%=MOD;
 87
                  }
 88
             }
 89
             int ans = Gauss(m,n);
 90
             if(ans == 0){
                  for(int i = 0;i < n;i++)</pre>
 91
 92
                      if(x[i] \leftarrow 2)
 93
                           x[i] += 7;
 94
                  for(int i = 0; i < n-1; i++)printf("%d<sub>\(\sigma\)</sub>", x[i]);
                  printf("%d\n",x[n-1]);
 95
 96
             }
 97
             else if(ans == -1)printf("Inconsistent_data.\n");
             else printf("Multiple_solutions.\n");
 98
 99
         }
100
         return 0;
101 |}
```

### 2.11 整数拆分

```
//HDU 4651
   //把数 n 拆成几个数(小于等于 n)相加的形式,问有多少种拆法。
   const int MOD = 1e9+7;
   int dp[100010];
 5
   void init(){
 6
       memset(dp,0,sizeof(dp));
 7
       dp[0] = 1;
 8
       for(int i = 1;i <= 100000;i++){
            for(int j = 1, r = 1; i - (3 * j * j - j) / 2 >= 0; j++, r
 9
              *= -1){}
10
                dp[i] += dp[i -(3 * j * j - j) / 2] * r;
                dp[i] %= MOD;
11
12
                dp[i] = (dp[i] + MOD) \% MOD;
                if( i - (3 * j * j + j) / 2 >= 0 ){
13
14
                    dp[i] += dp[i - (3 * j * j + j) / 2] * r;
15
                    dp[i] %= MOD;
16
                    dp[i] = (dp[i] + MOD) %MOD;
17
                }
18
            }
       }
19
20
21
   int main(){
22
       int T;
23
       int n;
24
       init();
       scanf("%d",&T);
25
26
       while(T——){
27
            scanf("%d",&n);
28
            printf("%d\n",dp[n]);
29
       }
30
       return 0;
31
   }
32
33
   //HDU 4658
   //数 n(<=10^5) 的划分,相同的数重复不能超过 k 个。
   const int MOD = 1e9+7;
36
   int dp[100010];
37
   void init(){
38
       memset(dp,0,sizeof(dp));
39
       dp[0] = 1;
40
       for(int i = 1;i <= 100000;i++){
            for(int j = 1, r = 1; i - (3 * j * j - j) / 2 >= 0; j++, r
41
              \star = -1)
                dp[i] += dp[i -(3 * j * j - j) / 2] * r;
42
43
                dp[i] %= MOD;
44
                dp[i] = (dp[i] + MOD) %MOD;
45
                if(i - (3 * j * j + j) / 2 >= 0){
                    dp[i] += dp[i - (3 * j * j + j) / 2] * r;
46
                    dp[i] %= MOD;
47
48
                    dp[i] = (dp[i] + MOD) % MOD;
49
                }
```

```
50
           }
51
       }
52
53
   int solve(int n,int k){
       int ans = dp[n];
54
55
       for(int j = 1, r = -1; n - k*(3 * j * j - j) / 2 >= 0; j++, r
          *= -1){}
56
           ans += dp[n - k*(3 * j * j - j) / 2] * r;
57
            ans %= MOD;
58
            ans = (ans+MOD)%MOD;
59
            if( n - k*(3 * j * j + j) / 2 >= 0 ){
                ans += dp[n - k*(3 * j * j + j) / 2] * r;
60
61
                ans %= MOD;
62
                ans = (ans+MOD)%MOD;
63
            }
64
       }
65
       return ans;
66
   int main(){
67
68
       init();
69
       int ⊺;
70
       int n,k;
71
       scanf("%d",&T);
       while (T--)
72
73
            scanf("%d%d",&n,&k);
74
           printf("%d\n",solve(n,k));
75
76
       return 0;
77 | }
         求 A^B 的约数之和对 MOD 取模
   |//参考 POJ 1845
 2
   //里面有一种求1+p+p^2+p···^3+p^n的方法。
   //需要素数筛选和合数分解的程序,需要先调用 getPrime();
   long long pow_m(long long a,long long n){
 5
       long long ret = 1;
 6
       long long tmp = a%MOD;
 7
       while(n){
 8
            if(n&1)ret = (ret*tmp)%MOD;
 9
            tmp = tmp*tmp%MOD;
10
           n >>= 1;
11
12
       return ret;
13
14
   //计算1+p+p^2+...+p^n
15
   long long sum(long long p,long long n){
16
       if(p == 0)return 0;
       if(n == 0)return 1;
17
18
       if(n & 1){
            return ((1+pow_m(p,n/2+1))%MOD*sum(p,n/2)%MOD)%MOD;
19
20
       }
```

```
21
       else return ((1+pow_m(p,n/2+1))%MOD*sum(p,n/2-1)+pow_m(p,n/2)%
          MOD)%MOD;
22
   //返回A^B的约数之和 % MOD
23
24
   long long solve(long long A,long long B){
25
       getFactors(A);
       long long ans = 1;
26
        for(int i = 0;i < fatCnt;i++){</pre>
27
28
            ans *= sum(factor[i][0],B*factor[i][1])%MOD;
29
            ans %= MOD;
30
       }
31
       return ans;
32 | }
          莫比乌斯反演
   2.13
   2.13.1 莫比乌斯函数
 1 | const int MAXN = 1000000;
 2 | bool check[MAXN+10];
   int prime[MAXN+10];
   |int mu[MAXN+10];
 5
   void Moblus(){
       memset(check, false, sizeof(check));
 6
 7
       mu[1] = 1;
 8
       int tot = 0;
 9
       for(int i = 2; i <= MAXN; i++){</pre>
            if(!check[i]){
10
11
                prime[tot++] = i;
                mu[i] = -1;
12
13
            for(int j = 0; j < tot; j++){</pre>
14
                if(i * prime[j] > MAXN) break;
15
16
                check[i * prime[j]] = true;
                if( i % prime[j] == 0){
17
                    mu[i * prime[j]] = 0;
18
19
                    break;
20
                }
21
                else{
                    mu[i * prime[j]] = -mu[i];
22
23
                }
24
            }
25
       }
26 }
          例题: BZOJ2301
   对于给出的 n 个询问,每次求有多少个数对 (x,y),满足 a \le x \le b, c \le y \le d,且
   gcd(x,y) = k, gcd(x,y) 函数为 x 和 y 的最大公约数。1 <= n <= 50000, 1 <= a <= b <=
   50000, 1 \le c \le d \le 50000, 1 \le k \le 50000
 1 | const int MAXN = 100000;
   |bool check[MAXN+10];
  int prime[MAXN+10];
```

```
int mu[MAXN+10];
 4
 5
   void Moblus(){
 6
        memset(check, false, sizeof(check));
 7
        mu[1] = 1;
 8
        int tot = 0;
        for(int i = 2; i <= MAXN; i++){</pre>
 9
            if(!check[i]){
10
                prime[tot++] = i;
11
12
                mu[i] = -1;
13
            for(int j = 0; j < tot; j ++){</pre>
14
                if( i * prime[j] > MAXN) break;
15
                check[i * prime[j]] = true;
16
                if( i % prime[j] == 0){
17
18
                     mu[i * prime[j]] = 0;
19
                     break;
                }
20
21
                else{
22
                     mu[i * prime[j]] = -mu[i];
23
                }
24
            }
25
        }
26
   int sum[MAXN+10];
27
28
   //找 [1,n],[1,m] 内互质的数的对数
29
   long long solve(int n,int m){
30
        long long ans = 0;
31
        if(n > m)swap(n,m);
32
        for(int i = 1, la = 0; i <= n; i = la+1){</pre>
            la = min(n/(n/i), m/(m/i));
33
            ans += (long long)(sum[la] - sum[i-1])*(n/i)*(m/i);
34
35
36
        return ans;
37
38
   int main(){
39
        Moblus();
40
        sum[0] = 0;
        for(int i = 1;i <= MAXN;i++)</pre>
41
            sum[i] = sum[i-1] + mu[i];
42
43
        int a,b,c,d,k;
44
        int T;
        scanf("%d",&T);
45
46
        while (T--)
47
            scanf("%d%d%d%d%d",&a,&b,&c,&d,&k);
48
            long long ans = solve(b/k,d/k) - solve((a-1)/k,d/k) - solve
               (b/k,(c-1)/k) + solve((a-1)/k,(c-1)/k);
49
            printf("%lld\n",ans);
50
        }
51
        return 0;
52 |}
```

### 2.14 Baby-Step Giant-Step

```
//(POJ 2417,3243)
   //baby_step giant_step
  |// a^x = b (mod n) n 是素数和不是素数都可以
   |// 求解上式 0<=x < n 的解
 5 #define MOD 76543
   void insert(int x,int y){
 8
       int k = x\%MOD;
9
       hs[top] = x, id[top] = y, next[top] = head[k], head[k] = top++;
10
11
   int find(int x){
       int k = x\%MOD;
12
13
       for(int i = head[k]; i != -1; i = next[i])
14
           if(hs[i] == x)
15
               return id[i];
16
       return -1;
17
18
   int BSGS(int a,int b,int n){
19
       memset(head,-1,sizeof(head));
20
       top = 1;
21
       if(b == 1)return 0;
22
       int m = sqrt(n*1.0), j;
23
       long long x = 1, p = 1;
24
       for(int i = 0; i < m; ++i, p = p*a%n)insert(p*b%n,i);
25
       for(long long i = m; ;i += m){
26
           if( (j = find(x = x*p%n)) != -1 )return i-j;
27
           if(i > n)break;
28
29
       return -1;
30 | }
         自适应 simpson 积分
   2.15
 1 | double simpson(double a, double b) {
 2
       double c = a + (b-a)/2;
 3
       return (F(a) + 4*F(c) + F(b))*(b-a)/6;
 4
   double asr(double a,double b,double eps,double A){
 6
       double c = a + (b-a)/2;
 7
       double L = simpson(a,c), R = simpson(c,b);
       if(fabs(L + R - A) \le 15*eps)return L + R + (L + R - A)/15.0;
 8
 9
       return asr(a,c,eps/2,L) + asr(c,b,eps/2,R);
10
11
  double asr(double a, double b, double eps){
12
       return asr(a,b,eps,simpson(a,b));
13 | }
         斐波那契数列取模循环节
```

必要时要上 unsigned long long HDU3977

```
long long gcd(long long a,long long b){
 2
       if(b == 0)return a;
 3
       return gcd(b,a%b);
 4
   long long lcm(long long a,long long b){
 5
       return a/gcd(a,b)*b;
 7
 8
   struct Matrix{
       long long mat[2][2];
10
   };
11
   Matrix mul_M(Matrix a, Matrix b, long long mod){
12
       Matrix ret;
13
       for(int i = 0;i < 2;i++)
            for(int j = 0; j < 2; j++){
14
15
                ret.mat[i][j] = 0;
16
                for(int k = 0; k < 2; k++){
17
                     ret.mat[i][j] += a.mat[i][k]*b.mat[k][j]%mod;
                    if(ret.mat[i][j] >= mod)ret.mat[i][j] -= mod;
18
                }
19
20
            }
21
       return ret;
22
23
   Matrix pow_M(Matrix a, long long n, long long mod) {
24
       Matrix ret;
25
       memset(ret.mat,0,sizeof(ret.mat));
26
       for(int i = 0;i < 2;i++)ret.mat[i][i] = 1;</pre>
27
       Matrix tmp = a;
28
       while(n){
29
            if(n&1)ret = mul_M(ret,tmp,mod);
30
            tmp = mul_M(tmp,tmp,mod);
31
            n >>= 1;
32
       }
33
       return ret;
34
35
   long long pow_m(long long a,long long n,long long mod)//a^b % mod{
36
       long long ret = 1;
37
       long long tmp = a%mod;
38
       while(n){
39
            if(n&1)ret = ret*tmp%mod;
40
            tmp = tmp*tmp%mod;
41
            n >>= 1;
42
43
       return ret;
44
45
   //素数筛选和合数分解
   const int MAXN = 1000000;
   int prime[MAXN+1];
47
48
   void getPrime(){
       memset(prime,0,sizeof(prime));
49
50
       for(int i = 2;i <= MAXN;i++){</pre>
            if(!prime[i])prime[++prime[0]] = i;
51
```

```
52
             for(int j = 1; j <= prime[0] && prime[j] <= MAXN/i; j++){</pre>
 53
                 prime[prime[j]*i] = 1;
 54
                 if(i%prime[j] == 0)break;
 55
             }
         }
 56
 57
 58
    long long factor[100][2];
 59
    int fatCnt;
 60
    int getFactors(long long x){
 61
         fatCnt = 0;
 62
         long long tmp = x;
         for(int i = 1;prime[i] <= tmp/prime[i];i++){</pre>
 63
 64
             factor[fatCnt][1] = 0;
 65
             if(tmp%prime[i] == 0){
 66
                 factor[fatCnt][0] = prime[i];
                 while(tmp%prime[i] == 0){
 67
 68
                      factor[fatCnt][1]++;
 69
                      tmp /= prime[i];
 70
                 fatCnt++;
 71
 72
             }
 73
         }
         if(tmp != 1){
 74
             factor[fatCnt][0] = tmp;
 75
 76
             factor[fatCnt++][1] = 1;
 77
 78
         return fatCnt;
 79
 80
    //勒让德符号
    int legendre(long long a,long long p){
 81
 82
         if(pow_m(a, (p-1) >> 1, p) == 1)return 1;
 83
         else return -1;
 84
    }
 85
    int f0 = 1;
 86
    int f1 = 1;
 87
    long long getFib(long long n,long long mod){
 88
         if(mod == 1)return 0;
 89
         Matrix A;
 90
         A.mat[0][0] = 0;
 91
         A.mat[1][0] = 1;
 92
         A.mat[0][1] = 1;
         A.mat[1][1] = 1;
 93
         Matrix B = pow_M(A,n,mod);
 94
 95
         long long ret = f0*B.mat[0][0] + f1*B.mat[1][0];
 96
         return ret%mod;
 97
 98
    long long fac[1000000];
 99
    long long G(long long p) {
100
         long long num;
101
         if(legendre(5,p) == 1)num = p-1;
102
         else num = 2*(p+1);
```

```
103
         //找出 num 的所有约数
104
         int cnt = 0;
         for(long long i = 1;i*i <= num;i++)</pre>
105
             if(num%i == 0){
106
                 fac[cnt++] = i;
107
                 if(i*i != num)
108
                      fac[cnt++] = num/i;
109
             }
110
111
         sort(fac,fac+cnt);
112
         long long ans;
         for(int i = 0;i < cnt;i++){</pre>
113
             if(getFib(fac[i],p) == f0 && getFib(fac[i]+1,p) == f1){
114
                 ans = fac[i];
115
                 break;
116
117
             }
118
         }
119
         return ans;
120
    long long find_loop(long long n){
121
         getFactors(n);
122
123
         long long ans = 1;
124
         for(int i = 0;i < fatCnt;i++){</pre>
125
             long long record = 1;
             if(factor[i][0] == 2)record = 3;
126
             else if(factor[i][0] == 3)record = 8;
127
             else if(factor[i][0] == 5)record = 20;
128
             else record = G(factor[i][0]);
129
             for(int j = 1;j < factor[i][1];j++)</pre>
130
131
                 record *= factor[i][0];
132
             ans = lcm(ans,record);
133
134
         return ans;
135
136
    void init(){
137
         getPrime();
138
139
    int main(){
140
         init();
         int T;
141
         int iCase = 0;
142
         int n;
143
         scanf("%d",&T);
144
         while (T--)
145
146
             iCase++;
             scanf("%d",&n);
147
             printf("Case_#%d:_%I64d\n",iCase,find_loop(n));
148
         }
149
150
         return 0;
151 |}
```

### 2.17 原根

定义:设 m>1,gcd(a,m)=1,使得  $a^d\equiv 1 \pmod{m}$  成立的最小的正整数 d 为 a 对模 m 的阶,记为  $\delta_m(a)$ . 如果  $\delta_m(a)=\varphi(m)$ ,则称 a 是模 m 的原根. 定理:若 m>1,(a,m)=1,正整数 d 满足  $a^d\equiv 1 \pmod{m}$ ,则  $\delta_m(a)$  整除 d. 定理:模 m 有原根的充要条件是  $m=2,4,p^n,2p^n$ ,其中 p 是奇质数,n 是任意正整数. 定理:如果模 m 有原根,那么它一定有  $\varphi(\varphi(m))$  个原根. 定理:如果 p 是素数,那么素数 p 一定有原根,并且模 p 的原根的个数为  $\varphi(p-1)$ . 求模素数 p 原根的方法:对 p-1 素因子分解,即  $p-1=p_1^{a_1}p_2^{a_2}...p_k^{a_k}$  的标准分解式,若恒有

$$g^{\frac{p-1}{p_i}} \neq 1 (mod \ p)$$

成立,则g就是p的原根。(对于合数求原根,只需要把p-1换成 $\varphi(p)$ )即可。

#### 求素数的最小原根程序

```
//*************
 2
   //素数筛选和合数分解
   const int MAXN=100000;
   int prime[MAXN+1];
   void getPrime(){
 5
       memset(prime,0,sizeof(prime));
 6
 7
       for(int i=2;i<=MAXN;i++){</pre>
 8
            if(!prime[i])prime[++prime[0]]=i;
 9
            for(int j=1;j<=prime[0]&&prime[j]<=MAXN/i;j++){</pre>
10
                prime[prime[j]*i]=1;
11
                if(i%prime[j]==0) break;
12
            }
       }
13
14
   long long factor[100][2];
15
   int fatCnt;
16
   int getFactors(long long x){
17
       fatCnt=0;
18
       long long tmp=x;
19
       for(int i=1;prime[i] <= tmp/prime[i]; i++){</pre>
20
            factor[fatCnt][1]=0;
21
            if(tmp%prime[i]==0){
22
                factor[fatCnt][0]=prime[i];
23
                while(tmp%prime[i]==0){
24
                    factor[fatCnt][1]++;
25
26
                    tmp/=prime[i];
27
                fatCnt++;
28
29
            }
30
       if(tmp!=1){
31
32
            factor[fatCnt][0]=tmp;
33
            factor[fatCnt++][1]=1;
34
35
       return fatCnt;
36
   }
```

```
37
   //***********
38
   long long pow_m(long long a,long long n,long long mod){
39
       long long ret = 1;
40
       long long tmp = a%mod;
41
       while(n){
42
            if(n&1)ret = ret*tmp%mod;
43
            tmp = tmp*tmp%mod;
44
           n >>= 1;
45
       }
46
       return ret;
47
48
   //求素数 P 的最小的原根
49
   void solve(int P){
       if(P == 2){
50
51
           printf("1\n");
52
            return;
       }
53
54
       getFactors(P-1);
       for(int g = 2; g < P;g++){
55
            bool flag = true;
56
            for(int i = 0;i < fatCnt;i++){</pre>
57
58
                int t = (P-1)/factor[i][0];
                if(pow_m(g,t,P) == 1){
59
                    flag = false;
60
61
                    break;
                }
62
63
            }
            if(flag){
64
65
                printf("%d\n",g);
66
                return;
67
            }
68
       }
69
70
   int main(){
71
       getPrime();
72
       int T;
73
       int P;
       scanf("%d",&T);
74
75
       while (T--)
76
            scanf("%d",&P);
77
            solve(P);
78
79
       return 0;
80 |}
```

# 2.18 快速数论变换

#### 2.18.1 HDU4656 卷积取模

```
HDU4656 x_k = b * c^{(2k)} + d, F(x) = a_0 x_0 + a_1 x_1 + a_2 x_2 + ... + a_{n-1} x_{n-1} Given n, b, c, d, a_0, ..., a_{n-1},
```

calculate  $F(x_0), ..., F(x_{n-1})$ .

$$\begin{split} F_{x_k} &= \sum_{i=0}^{n-1} a_i (bc^{2k} + d)^i \\ &= \sum_{i=0}^{n-1} a_i \sum_{j=0}^{i} C_i^j (bc^{2k})^j d^{i-j} \\ &= \sum_{j=0}^{n-1} (bc^{2k})^j j!^{-1} \sum_{i=j}^{n-1} a_i d^{i-j} i! (i-j)!^{-1} \\ &= \sum_{j=0}^{n-1} (bc^{2k})^j j!^{-1} \sum_{i=0}^{n-1-j} a_{n-1-i} (n-1-i)! d^{n-1-i-j} (n-1-i-j)!^{-1} \\ &= \sum_{j=0}^{n-1} (bc^{2k})^j j!^{-1} p_j \\ &= \sum_{j=0}^{n-1} b^j j!^{-1} p_j c^{2jk} \\ &= c^{k^2} \sum_{j=0}^{n-1} b^j j!^{-1} p_j c^{j^2} c^{-(k-j)^2} \\ &= c^{k^2} q_k \end{split}$$

其中  $p_i$  和  $q_k$  都是卷积,可以使用 NTT 进行快速计算。

```
//***********
  |//快速数论变换 (NTT)
  //求 A 和 B 的卷积,结果对 P 取模
  //做长度为 N1 的变换,选取两个质数 P1 和 P2
  |//P1-1 和 P2-1 必须是 N1 的倍数
   //E1 和 E2 分别是 P1,P2 的原根
  //F1 是 E1 模 P1 的逆元,F2 是 E2 模 P2 的逆元
   //I1 是 N1 对模 P1 的逆元,I2 是 N1 对模 P2 的逆元
9
   //
  //然后使用中国剩余定理,保证了结果是小于 MM=P1*P2 的
   //M1 = (P2 对 P1 的逆元)*P2
   //M2 = (P1 对 P2 的逆元)*P1
12
13
  |const int P = 1000003;//结果对 P 取模
14
  const int N1 = 262144;// 2^{18}
  |const int N2 = N1+1;//数组大小
17 const int P1 = 998244353; //P1 = 2^{23} * 7 * 17 + 1
  const int P2 = 995622913; //P2 = 2^{19} * 3 * 3 * 211 + 1
18
  const int E1 = 996173970;
  const int E2 = 88560779;
  const int F1 = 121392023; //E1*F1 = 1 (mod P1)
  const int F2 = 840835547; //E2*F2 = 1 (mod P2)
  const int I1 = 998240545; //I1*N1 = 1(mod P1)
  const int I2 = 995619115;//I2*N1 = 1(mod P2)
25 | const long long M1 = 397550359381069386LL;
```

```
26 const long long M2 = 596324591238590904LL;
27
   const long long MM = 993874950619660289LL;//MM = P1*P2
   //计算 x*y 对 z 取模
28
29
   long long mul(long long x,long long y,long long z) {
30
       return (x*y - (long long)(x/(long double)z*y+1e-3)*z+z)%z;
31
32
   int trf(int x1,int x2){
33
       return (mul(M1, x1, MM)+mul(M2, x2, MM))%MM%P;
34
   }
   int A[N2],B[N2],C[N2];
35
36
   int A1[N2],B1[N2],C1[N2];
37
   void fft(int *A,int PM,int PW){
38
       for (int m = N1,h;h = m/2, m \ge 2;PW = (long long)PW*PW%PM,m=h)
            for(int i = 0,w=1;i < h;i++, w = (long long)w*PW%PM)</pre>
39
40
                for(int j = i; j < N1; j += m){
41
                    int k = j+h, x = (A[j]-A[k]+PM)%PM;
                     (A\lceil j\rceil +=A\lceil k\rceil)\%=PM;
42
43
                    A[k] = (long long)w*x%PM;
44
45
       for(int i = 0, j = 1; j < N1-1; j++){
            for(int k = N1/2; k > (i^=k); k /= 2);
46
47
            if(j < i)swap(A[i],A[j]);
48
       }
49
50
   //计算 A 和 B 的卷积, 结果保存在 C 中, 结果对 P 取模
51
   void mul(){
52
       memset(C,0,sizeof(C));
53
       memcpy(A1,A,sizeof(A));
54
       memcpy(B1,B,sizeof(B));
55
       fft(A1,P1,E1); fft(B1,P1,E1);
56
       for(int i = 0;i < N1;i++)C1[i] = (long long)A1[i]*B1[i]%P1;</pre>
57
       fft(C1,P1,F1);
       for(int i = 0;i < N1;i++)C1[i] = (long long)C1[i]*I1%P1;</pre>
58
59
       fft(A,P2,E2); fft(B,P2,E2);
       for(int i = 0;i < N1;i++)C[i] = (long long)A[i]*B[i]%P2;</pre>
60
61
       fft(C,P2,F2);
62
       for(int i = 0;i < N1;i++)C[i] = (long long)C[i]*I2%P2;</pre>
63
       for(int i = 0;i < N1;i++)C[i] = trf(C1[i],C[i]);</pre>
64
   |int INV[P];//逆元
65
   const int MAXN = 100010;
   int F[MAXN];//阶乘
67
68
   int a[MAXN];
69
   int pd[MAXN];
70
   int pb[MAXN];
71
   |int pc2[MAXN];
   int p[MAXN];
72
73
   int main()
74
   {
75
       //预处理逆元
76
       INV[1] = 1;
```

```
77
         for(int i = 2;i < P;i++)
 78
             INV[i] = (long long)P/i*(P-INV[P%i])%P;
 79
         F[0] = 1;
         for(int i = 1;i < MAXN;i++)</pre>
 80
             F[i] = (long long)F[i-1]*i%P;
 81
 82
         int n,b,c,d;
 83
         while(scanf("%d%d%d%d",&n,&b,&c,&d) == 4){
             for(int i = 0;i < n;i++)scanf("%d",&a[i]);</pre>
 84
 85
             pd[0] = 1;
             for(int i = 1;i < n;i++)
 86
                 pd[i] = (long long)pd[i-1]*d%P;
 87
             memset(A,0,sizeof(A));
 88
             memset(B,0,sizeof(B));
 89
             for(int i = 0;i < n;i++)</pre>
 90
                 A[i] = (long long)a[n-1-i]*F[n-1-i]%P;
 91
             for(int i = 0;i < n;i++)</pre>
 92
 93
                  B[i] = (long long)pd[i]*INV[F[i]]%P;
 94
             mul();
             for(int i = 0;i < n;i++)p[i] = C[i];</pre>
 95
 96
             reverse(p,p+n);
 97
             memset(A,0,sizeof(A));
 98
             pb[0] = 1;
 99
             for(int i = 1;i < n;i++)
                 pb[i] = (long long)pb[i-1]*b%P;
100
101
             pc2[0] = 1;
             int c2 = (long long)c*c%P;
102
             for(int i = 1, s = c;i < n;i++){
103
                 pc2[i] = (long long)pc2[i-1]*s%P;
104
105
                 s = (long long)s*c2%P;
106
             for(int i = 0;i < n;i++)</pre>
107
                 A[i] = (long long)pb[i]*INV[F[i]]%P*p[i]%P*pc2[i]%P;
108
             memset(B,0,sizeof(B));
109
110
             B[0] = 1;
             for(int i = 1;i < n;i++)</pre>
111
112
                  B[i] = B[N1-i] = INV[pc2[i]];
113
             mul();
             for(int i = 0;i < n;i++)C[i] = (long long)C[i]*pc2[i]%P;</pre>
114
115
             for(int i = 0;i < n;i++)
116
                 printf("%d\n",C[i]);
117
         }
118
         return 0;
119 |}
```

# 2.19 其它公式

#### 2.19.1 Polya

设 G 是 p 个对象的一个置换群,用 k 种颜色去染这 p 个对象,若一种染色方案在群 G 的作用下变为另一种方案,则这两个方案当作是同一种方案,这样的不同染色方案数为:

```
L = \frac{1}{|G|} \times \Sigma(k^{C(f)}), f \in G
```

# C(f) 为循环节,|G| 表示群的置换方法数

对于有n个位置的手镯,有n种旋转置换和n种翻转置换

# 对于旋转置换:

$$C(f_i) = gcd(n,i)$$
,  $i$  表示一次转过  $i$  颗宝石,  $i = 0$  时  $c = n$ ;

### 对于翻转置换:

如果 n 为偶数: 则有  $\frac{n}{2}$  个置换  $C(f)=\frac{n}{2}$ ,有  $\frac{n}{2}$  个置换  $C(f)=\frac{n}{2}+1$ 

如果 n 为奇数:  $C(f) = \frac{n}{2} + 1$ 

# 3 数据结构

### 3.1 划分树

```
1
 2
    * 划分树(查询区间第 k 大)
 3
    */
   const int MAXN = 100010;
   |int tree[20][MAXN];//表示每层每个位置的值
   int sorted[MAXN];//已经排序好的数
   |int toleft[20][MAXN];//toleft[p][i] 表示第 i 层从 1 到 i 有数分入左边
 7
 8
 9
   void build(int l,int r,int dep){
       if(l == r)return;
10
       int mid = (l+r)>>1;
11
12
       int same = mid — l + 1;//表示等于中间值而且被分入左边的个数
       for(int i = l; i <= r; i++) //注意是 l, 不是 one
13
14
           if(tree[dep][i] < sorted[mid])</pre>
15
               same——;
16
       int lpos = l;
17
       int rpos = mid+1;
       for(int i = l;i <= r;i++){
18
           if(tree[dep][i] < sorted[mid])</pre>
19
20
               tree[dep+1][lpos++] = tree[dep][i];
           else if(tree[dep][i] == sorted[mid] && same > 0){
21
22
               tree[dep+1][lpos++] = tree[dep][i];
23
               same—-;
           }
24
25
           else
26
               tree[dep+1][rpos++] = tree[dep][i];
27
           toleft[dep][i] = toleft[dep][l-1] + lpos - l;
28
29
       build(l,mid,dep+1);
30
       build(mid+1,r,dep+1);
31
   }
32
   //查询区间第 k 大的数,[L,R] 是大区间,[l,r] 是要查询的小区间
33
   int query(int L,int R,int l,int r,int dep,int k){
34
       if(l == r)return tree[dep][l];
35
36
       int mid = (L+R)>>1;
       int cnt = toleft[dep][r] - toleft[dep][l-1];
37
       if(cnt >= k){
38
39
           int newl = L + toleft[dep][l-1] - toleft[dep][L-1];
40
           int newr = newl + cnt -1;
41
           return query(L,mid,newl,newr,dep+1,k);
42
       }
       else{
43
44
           int newr = r + toleft[dep][R] - toleft[dep][r];
45
           int newl = newr - (r-l-cnt);
46
           return query(mid+1,R,newl,newr,dep+1,k-cnt);
47
       }
```

```
48
   }
49
   int main(){
50
       int n,m;
51
       while(scanf("%d%d",&n,&m)==2){
           memset(tree,0,sizeof(tree));
52
           for(int i = 1;i <= n;i++){</pre>
53
               scanf("%d",&tree[0][i]);
54
55
               sorted[i] = tree[0][i];
56
           }
57
           sort(sorted+1,sorted+n+1);
58
           build(1,n,0);
           int s,t,k;
59
           while(m——){
60
61
               scanf("%d%d%d",&s,&t,&k);
62
               printf("%d\n",query(1,n,s,t,0,k));
63
           }
       }
64
65
       return 0;
66 |}
   3.2
       RMQ
   3.2.1 一维
   求最大值,数组下标从1开始。
   求最小值,或者最大最小值下标,或者数组从①开始对应修改即可。
 1 | const int MAXN = 50010;
 2
   int dp[MAXN][20];
   int mm[MAXN];
   //初始化 RMQ, b 数组下标从 1 开始,从 0 开始简单修改
 5
   void initRMQ(int n,int b[]){
 6
       mm\lceil 0\rceil = -1;
 7
       for(int i = 1; i <= n;i++){
           mm[i] = ((i&(i-1)) == 0)?mm[i-1]+1:mm[i-1];
 8
 9
           dp[i][0] = b[i];
10
       for(int j = 1; j <= mm[n];j++)</pre>
11
           for(int i = 1; i + (1 << j) -1 <= n; i++)
12
13
               dp[i][j] = max(dp[i][j-1],dp[i+(1<<(j-1))][j-1]);
14
   }
15
   //查询最大值
   |int rmq(int x,int y){
16
17
       int k = mm[y-x+1];
18
       return max(dp[x][k],dp[y-(1<< k)+1][k]);
19 |}
   3.2.2 二维
    * 二维 RMQ, 预处理复杂度 n*m*log*(n)*log(m)
 2
 3
    * 数组下标从 1 开始
```

```
4
    */
   int val[310][310];
 5
   int dp[310][310][9][9];//最大值
   int mm[310];//二进制位数减一,使用前初始化
 7
 8
   void initRMQ(int n,int m){
 9
       for(int i = 1; i <= n; i++)
            for(int j = 1; j <= m; j++)</pre>
10
                dp[i][j][0][0] = val[i][j];
11
12
        for(int ii = 0; ii <= mm[n]; ii++)</pre>
            for(int jj = 0; jj <= mm[m]; jj++)</pre>
13
14
                if(ii+jj)
15
                    for(int i = 1; i + (1 << ii) - 1 <= n; i++)
                         for(int j = 1; j + (1 << jj) - 1 <= m; j++){}
16
                             if(ii)dp[i][j][ii][jj] = max(dp[i][j][ii
17
                                -1][jj],dp[i+(1<<(ii-1))][j][ii-1][jj]);
18
                             else dp[i][j][ii][jj] = max(dp[i][j][ii][jj
                                -1],dp[i][j+(1<<(jj-1))][ii][jj-1]);
19
                         }
20
   //查询矩形内的最大值 (x1<=x2,y1<=y2)
21
22
   int rmq(int x1,int y1,int x2,int y2){
       int k1 = mm[x2-x1+1];
23
24
       int k2 = mm[y2-y1+1];
25
       x2 = x2 - (1 << k1) + 1;
26
       y2 = y2 - (1 << k2) + 1;
27
       return max(max(dp[x1][y1][k1][k2],dp[x1][y2][k1][k2]),max(dp[x2
           ][y1][k1][k2],dp[x2][y2][k1][k2]));
28
29
   int main(){
30
        //在外面对 mm 数组进行初始化
31
       mm[0] = -1;
32
       for(int i = 1;i <= 305;i++)
33
            mm[i] = ((i&(i-1))==0)?mm[i-1]+1:mm[i-1];
34
       int n,m;
35
       int Q;
36
       int r1,c1,r2,c2;
37
       while(scanf("%d%d",&n,&m) == 2){
            for(int i = 1;i <= n;i++)</pre>
38
39
                for(int j = 1; j <= m; j++)
                     scanf("%d",&val[i][j]);
40
41
            initRMQ(n,m);
            scanf("%d",&Q);
42
43
            while (Q--)
44
                scanf("%d%d%d%d",&r1,&c1,&r2,&c2);
45
                if(r1 > r2)swap(r1,r2);
                if(c1 > c2)swap(c1,c2);
46
47
                int tmp = rmq(r1,c1,r2,c2);
48
                printf("%d<sub>□</sub>",tmp);
                if(tmp == val[r1][c1] || tmp == val[r1][c2] || tmp ==
49
                   val[r2][c1] || tmp == val[r2][c2])
50
                    printf("yes\n");
```

```
51
               else printf("no\n");
52
           }
53
       }
54
       return 0;
55 |}
       树链剖分
   3.3
   3.3.1 点权
   基于点权,查询单点值,修改路径的上的点权(HDU 3966 树链剖分 + 树状数组)
 1 | const int MAXN = 50010;
 2
   struct Edge{
 3
       int to,next;
  }edge[MAXN*2];
 4
  |int head[MAXN],tot;
   int top[MAXN];//top[v] 表示 v 所在的重链的顶端节点
   |int fa[MAXN];//父亲节点
   |int deep[MAXN];//深度
   |int num[MAXN];//num[v] 表示以 v 为根的子树的节点数
  |int p[MAXN];//p[v] 表示 v 对应的位置
   |int fp[MAXN];//fp 和 p 数组相反
   |int son[MAXN];//重儿子
13
   int pos;
14
   void init(){
15
       tot = 0;
16
       memset(head,-1,sizeof(head));
       pos = 1; / / 使用树状数组, 编号从头 1 开始
17
       memset(son, -1, sizeof(son));
18
19
20
   void addedge(int u,int v){
21
       edge[tot].to = v; edge[tot].next = head[u]; head[u] = tot++;
22
23
   void dfs1(int u,int pre,int d){
       deep[u] = d;
24
25
       fa[u] = pre;
       num[u] = 1;
26
27
       for(int i = head[u]; i != -1; i = edge[i].next){
           int v = edge[i].to;
28
           if(v != pre){
29
               dfs1(v,u,d+1);
30
               num[u] += num[v];
31
32
               if(son[u] == -1 \mid \mid num[v] > num[son[u]])
                   son[u] = v;
33
34
           }
35
       }
36
37
   void getpos(int u,int sp){
       top[u] = sp;
38
39
       p[u] = pos++;
40
       fp[p[u]] = u;
```

```
41
       if(son[u] == -1) return;
42
       getpos(son[u],sp);
43
       for(int i = head[u];i != -1;i = edge[i].next){
            int v = edge[i].to;
44
45
            if( v != son[u] && v != fa[u])
46
                getpos(v,v);
47
       }
48
   }
49
50
   //树状数组
   int lowbit(int x){
51
       return x&(-x);
52
53
   }
54
   int c[MAXN];
55
   int n;
56
   int sum(int i){
       int s = 0;
57
       while(i > 0)
58
59
       {
60
            s += c[i];
            i -= lowbit(i);
61
62
       }
63
       return s;
64
65
   void add(int i,int val){
66
       while(i <= n){
67
            c[i] += val;
68
            i += lowbit(i);
69
       }
70
71
   //u->v 的路径上点的值改变 val
72
   void Change(int u,int v,int val){
73
       int f1 = top[u], f2 = top[v];
       int tmp = 0;
74
       while(f1 != f2){
75
76
            if(deep[f1] < deep[f2]){
77
                swap(f1,f2);
78
                swap(u,v);
79
            }
            add(p[f1],val);
80
81
            add(p[u]+1,-val);
82
            u = fa[f1];
83
            f1 = top[u];
84
       }
85
       if(deep[u] > deep[v]) swap(u,v);
       add(p[u],val);
86
87
       add(p[v]+1,-val);
88
89
   int a[MAXN];
90
   int main(){
       int M,P;
91
```

```
92
        while(scanf("%d%d%d",&n,&M,&P) == 3){
93
            int u,v;
            int C1,C2,K;
94
95
            char op[10];
96
            init();
            for(int i = 1;i <= n;i++){</pre>
 97
                 scanf("%d",&a[i]);
 98
99
            }
100
            while (M——) {
                scanf("%d%d",&u,&v);
101
102
                addedge(u,v);
                addedge(v,u);
103
            }
104
            dfs1(1,0,0);
105
106
            getpos(1,1);
107
            memset(c,0,sizeof(c));
            for(int i = 1;i <= n;i++){</pre>
108
                add(p[i],a[i]);
109
                add(p[i]+1,-a[i]);
110
111
            }
            while(P--){
112
113
                scanf("%s",op);
114
                if(op[0] == 'Q'){
                     scanf("%d",&u);
115
                     printf("%d\n",sum(p[u]));
116
                }
117
                else{
118
119
                     scanf("%d%d%d",&C1,&C2,&K);
120
                     if(op[0] == 'D')
121
                         K = -K;
122
                     Change (C1,C2,K);
123
                }
            }
124
125
        }
126
        return 0;
127 |}
    3.3.2 边权
    基于边权,修改单条边权,查询路径边权最大值(SPOJ QTREE 树链剖分 + 线段树)
  1 | const int MAXN = 10010;
  2
    struct Edge{
  3
        int to,next;
    }edge[MAXN*2];
    int head[MAXN],tot;
    |int top[MAXN];//top[v] 表示 v 所在的重链的顶端节点
  7
    |int fa[MAXN]; //父亲节点
    |int deep[MAXN];//深度
    |int num[MAXN];//num[v] 表示以 v 为根的子树的节点数
   |int p[MAXN];//p[v] 表示 v 与其父亲节点的连边在线段树中的位置
```

```
11 | int fp[MAXN]; / /和 p 数组相反
12
   |int son[MAXN];//重儿子
13
   int pos;
14
   void init(){
15
       tot = 0;
       memset(head,-1,sizeof(head));
16
17
       pos = 0;
18
       memset(son,-1,sizeof(son));
19
   }
20
   void addedge(int u,int v){
21
       edge[tot].to = v;edge[tot].next = head[u];head[u] = tot++;
22
   }
23
   //第一遍 dfs 求出 fa,deep,num,son
   void dfs1(int u,int pre,int d){
24
       deep[u] = d;
25
26
       fa[u] = pre;
27
       num[u] = 1;
28
       for(int i = head[u]; i != -1; i = edge[i].next){
29
            int v = edge[i].to;
            if(v != pre){
30
                dfs1(v,u,d+1);
31
32
                num[u] += num[v];
33
                if(son[u] == -1 \mid \mid num[v] > num[son[u]])
34
                    son[u] = v;
35
            }
       }
36
37
38
   //第二遍 dfs 求出 top 和 p
   void getpos(int u,int sp){
40
       top[u] = sp;
       p[u] = pos++;
41
42
       fp[p[u]] = u;
       if(son[u] == -1) return;
43
44
       getpos(son[u],sp);
45
       for(int i = head[u]; i != -1; i = edge[i].next){
46
            int v = edge[i].to;
47
            if(v != son[u] && v != fa[u])
48
                getpos(v,v);
49
       }
   }
50
51
   //线段树
52
53
   struct Node{
54
       int l,r;
55
       int Max;
   }segTree[MAXN*3];
56
   void build(int i,int l,int r){
57
58
       segTree[i].l = l;
59
       segTree[i].r = r;
60
       segTree[i].Max = 0;
61
       if(l == r)return;
```

```
62
        int mid = (l+r)/2;
 63
        build(i<<1,l,mid);</pre>
 64
        build((i<<1)|1,mid+1,r);
 65
    void push_up(int i){
 66
 67
        segTree[i].Max = max(segTree[i<<1].Max,segTree[(i<<1)|1].Max);</pre>
 68
 69
    // 更新线段树的第 k 个值为 val
 70
    void update(int i,int k,int val){
 71
        if(segTree[i].l == k && segTree[i].r == k){
             segTree[i].Max = val;
 72
 73
             return;
 74
        }
 75
        int mid = (segTree[i].l + segTree[i].r)/2;
        if(k <= mid)update(i<<1,k,val);</pre>
 76
 77
        else update((i<<1)|1,k,val);
 78
        push_up(i);
 79
    //查询线段树中「l,r] 的最大值
 80
 81
    int query(int i,int l,int r){
        if(segTree[i].l == l && segTree[i].r == r)
 82
 83
             return segTree[i].Max;
        int mid = (segTree[i].l + segTree[i].r)/2;
 84
        if(r <= mid)return query(i<<1,l,r);</pre>
 85
 86
        else if(l > mid)return query((i<<1)|1,l,r);</pre>
 87
        else return max(query(i<<1,l,mid),query((i<<1)|1,mid+1,r));</pre>
 88
    }
 89
    //查询 u->v 边的最大值
    int find(int u,int v){
 91
        int f1 = top[u], f2 = top[v];
 92
        int tmp = 0;
        while(f1 != f2){
 93
             if(deep[f1] < deep[f2]){
 94
 95
                 swap(f1,f2);
 96
                 swap(u,v);
 97
             }
 98
             tmp = max(tmp,query(1,p[f1],p[u]));
 99
             u = fa[f1]; f1 = top[u];
100
101
        if(u == v)return tmp;
102
        if(deep[u] > deep[v]) swap(u,v);
103
        return max(tmp,query(1,p[son[u]],p[v]));
104
105
    int e[MAXN][3];
106
    int main(){
        int T;
107
108
        int n;
        scanf("%d",&T);
109
        while (T--)
110
111
             init();
             scanf("%d",&n);
112
```

```
for(int i = 0;i < n-1;i++){</pre>
113
114
                 scanf("%d%d%d",&e[i][0],&e[i][1],&e[i][2]);
                 addedge(e[i][0],e[i][1]);
115
                 addedge(e[i][1],e[i][0]);
116
             }
117
             dfs1(1,0,0);
118
119
             getpos(1,1);
             build(1,0,pos-1);
120
121
             for(int i = 0; i < n-1; i++){
                 if(deep[e[i][0]] > deep[e[i][1]])
122
123
                     swap(e[i][0],e[i][1]);
124
                 update(1,p[e[i][1]],e[i][2]);
             }
125
             char op[10];
126
127
             int u, v;
128
            while(scanf("%s",op) == 1){
                 if(op[0] == 'D')break;
129
                 scanf("%d%d",&u,&v);
130
                 if(op[0] == 'Q')
131
                     printf("%d\n",find(u,v));//查询 u->v 路径上边权的最大值
132
                 else update(1,p[e[u-1][1]],v);//修改第 u 条边的长度为 v
133
             }
134
135
        }
136
        return 0;
137 }
         伸展树(splay tree)
    3.4
    3.4.1 例题: HDU1890
  1 | const int MAXN = 100010;
 2
    struct Node;
  3
    Node* null;
  4
    struct Node{
  5
        Node *ch[2],*fa;
  6
        int size;
  7
        int rev;
  8
        Node(){
  9
             ch[0] = ch[1] = fa = null; rev = 0;
 10
        inline void push_up(){
 11
 12
             if(this == null)return;
 13
             size = ch[0]->size + ch[1]->size + 1;
 14
 15
        inline void setc(Node* p,int d){
             ch[d] = p;
 16
             p->fa = this;
 17
 18
 19
        inline bool d(){
 20
             return fa->ch[1] == this;
 21
        }
```

```
22
        void clear(){
23
             size = 1;
             ch[0] = ch[1] = fa = null;
24
25
             rev = 0;
26
        }
        void Update_Rev(){
27
28
             if(this == null)return;
29
             swap(ch[0],ch[1]);
30
             rev ^= 1;
31
        inline void push_down(){
32
             if(this == null)return;
33
             if(rev){
34
                  ch[0]->Update_Rev();
35
36
                  ch[1]->Update_Rev();
37
                  rev = 0;
38
             }
39
        }
40
        inline bool isroot(){
             return fa == null || this != fa->ch[0] && this != fa->ch
41
                 \lceil 1 \rceil;
42
        }
43
   };
   inline void rotate(Node* x)
44
45
    {
46
        Node *f = x \rightarrow fa, *ff = x \rightarrow fa \rightarrow fa;
47
        f->push_down();
        x->push_down();
48
49
        int c = x->d(), cc = f->d();
50
        f\rightarrow setc(x\rightarrow ch[!c],c);
51
        x \rightarrow setc(f,!c);
        if(ff->ch[cc] == f)ff->setc(x,cc);
52
53
        else x\rightarrow fa = ff;
54
        f->push_up();
55
56
   inline void splay(Node* &root,Node* x,Node* goal)
57
        while(x->fa != goal){
58
59
             if(x->fa->fa == goal)rotate(x);
60
             else {
61
                  x->fa->fa->push_down();
                  x->fa->push_down();
62
                  x->push_down();
63
                  bool f = x->fa->d();
64
                  x\rightarrow d() == f ? rotate(x\rightarrow fa):rotate(x);
65
66
                  rotate(x);
             }
67
68
69
        x->push_up();
70
        if(goal == null)root = x;
71 |}
```

```
Node* get_kth(Node* r,int k)
 73
     {
 74
         Node* x = r;
 75
         x->push_down();
         while (x->ch[0]->size+1 != k) {
 76
 77
              if(k < x->ch[0]->size+1)x = x->ch[0];
 78
              else{
 79
                   k = x - ch[0] - size + 1;
 80
                   x = x -> ch[1];
 81
 82
              x->push_down();
         }
 83
 84
         return x;
 85
 86
     Node* get_next(Node* p){
 87
         p->push_down();
 88
         p = p - > ch[1];
 89
         p->push_down();
 90
         while(p->ch[0] != null){
 91
              p = p \rightarrow ch[0];
 92
              p->push_down();
 93
         }
 94
         return p;
 95
 96
    Node pool[MAXN],*tail;
    Node *node[MAXN];
 97
    Node *root;
 99
     void build(Node* &x,int l,int r,Node* fa)
100
101
         if(l > r)return;
         int mid = (l+r)/2;
102
         x = tail++;
103
         x->clear();
104
         x->fa = fa;
105
         node[mid] = x;
106
         build(x \rightarrow ch[0], l, mid-1, x);
107
108
         build(x \rightarrow ch[1], mid+1, r, x);
109
         x->push_up();
110
111
    void init(int n)
112
     {
         tail = pool;
113
114
         null = tail++;
         null \rightarrow fa = null \rightarrow ch[0] = null \rightarrow ch[1] = null;
115
         null->size = 0; null->rev = 0;
116
         Node *p = tail++;
117
118
         p->clear();
119
         root = p;
120
         p = tail++;
121
         p->clear();
122
         root->setc(p,1);
```

```
123
         build(root->ch[1]->ch[0],1,n,root->ch[1]);
124
         root->ch[1]->push_up();
         root->push_up();
125
126
    int a[MAXN];
127
    int b[MAXN];
128
    bool cmp(int i,int j)
129
130
    {
131
         if(a[i] != a[j])return a[i] < a[j];</pre>
132
         else return i < j;</pre>
133
134
    int main()
135
    {
136
         int n;
         while(scanf("%d",&n) == 1 && n){
137
138
             for(int i = 1;i <= n;i++){
                 scanf("%d",&a[i]);
139
140
                 b[i] = i;
141
             }
             init(n);
142
             sort(b+1,b+n+1,cmp);
143
144
             for(int i = 1;i <= n;i++){
145
                 splay(root, node[b[i]], null);
                 int sz = root->ch[0]->size;
146
                 printf("%d",root->ch[0]->size);
147
                 if(i == n)printf("\n");
148
                 else printf("□");
149
                 splay(root,get_kth(root,i),null);
150
151
                 splay(root,get_kth(root,sz+2),root);
152
                 root->ch[1]->ch[0]->Update_Rev();
153
             }
154
         }
155
         return 0;
156
    }
    3.4.2 例题: HDU3726
  1 | const int MAXN = 20010;
  2
    struct Node;
    Node* null;
  3
    struct Node
  5
    {
  6
         Node *ch[2],*fa;//指向儿子和父亲结点
  7
         int size,key;
  8
         Node(){
  9
             ch[0] = ch[1] = fa = null;
 10
         inline void setc(Node* p,int d){
 11
             ch[d] = p;
 12
 13
             p->fa = this;
 14
 15
         inline bool d(){
```

```
return fa->ch[1] == this;
16
17
        }
        void push_up(){
18
19
             size = ch[0]->size + ch[1]->size + 1;
20
        void clear(){
21
22
             size = 1;
             ch[0] = ch[1] = fa = null;
23
24
        inline bool isroot()
25
26
        {
             return fa == null || this != fa->ch[0] && this != fa->ch
27
                [1];
28
        }
29
   };
30
   inline void rotate(Node* x)
31
32
        Node *f = x \rightarrow fa, *ff = x \rightarrow fa \rightarrow fa;
33
        int c = x->d(), cc = f->d();
        f\rightarrow setc(x\rightarrow ch[!c],c);
34
35
        x->setc(f,!c);
36
        if(ff->ch[cc] == f)ff->setc(x,cc);
37
        else x->fa = ff;
38
        f->push_up();
39
40
   inline void splay(Node* &root,Node* x,Node* goal)
41
   {
42
        while(x->fa != goal){
43
             if(x->fa->fa == goal)rotate(x);
44
             else {
45
                  bool f = x \rightarrow fa \rightarrow d();
46
                  x\rightarrow d() == f ? rotate(x\rightarrow fa) : rotate(x);
47
                  rotate(x);
48
             }
49
        }
50
        x->push_up();
51
        if(goal == null)root = x;
52
   //找到 r 子树里面的第 k 个
53
54
   Node* get_kth(Node* r,int k)
55
    {
56
        Node* x = r;
57
        while(x->ch[0]->size+1 != k){
58
             if(k < x->ch[0]->size+1)x = x->ch[0];
59
             else {
60
                  k = x - ch[0] - size + 1;
61
                  x = x - ch[1];
62
             }
63
        }
64
        return x;
65 |}
```

```
|//在 root 的树中删掉 x
 67
    void erase(Node* &root,Node* x)
 68
 69
         splay(root,x,null);
 70
         Node* t = root;
 71
         if(t->ch[1] != null){
 72
             root = t->ch[1];
 73
             splay(root,get_kth(root,1),null);
 74
             root->setc(t->ch[0],0);
         }
 75
 76
         else{
 77
             root = root->ch[0];
 78
 79
         root—>fa = null;
 80
         if(root != null)root->push_up();
 81
 82
    void insert(Node* &root,Node* x)
 83
    {
 84
         if(root == null){
 85
             root = x;
 86
             return;
 87
         }
 88
         Node* now = root;
 89
         Node* pre = root->fa;
 90
         while(now != null){
 91
             pre = now;
 92
             now = now - > ch[x - > key > = now - > key];
 93
         }
 94
         x->clear();
 95
         pre->setc(x,x->key >= pre->key);
 96
         splay(root,x,null);
 97
 98
    void merge(Node* &A,Node* B)
 99
100
         if(A->size <= B->size)swap(A,B);
101
         queue<Node*>Q;
102
         Q.push(B);
         while(!Q.empty()){
103
             Node* fr = Q.front();
104
105
             Q.pop();
             if(fr->ch[0] != null)Q.push(fr->ch[0]);
106
             if(fr->ch[1] != null)Q.push(fr->ch[1]);
107
             fr->clear();
108
109
             insert(A,fr);
         }
110
111
112
    Node pool[MAXN],*tail;
113
114
    struct Edge
115
    {
116
         int u, v;
```

```
117 |}edge[60010];
118
    int a[MAXN];
119
    bool del[60010];
120
    struct QUERY
121
122
         char op[10];
123
         int u,v;
124
    |}query[500010];
125
    int y[500010];
126
127
    Node* node[MAXN];
128 | Node* root[MAXN];
129
    int F[MAXN];
    int find(int x)
130
131
    {
132
         if(F[x] == -1)return x;
         return F[x] = find(F[x]);
133
134
135
    void debug(Node *root)
136
    {
137
         if(root == null)return;
138
         debug(root->ch[0]);
139
         printf("size: \( \) \%d\\\ \n \', \root \( \) \size, \root \( \) \;
140
         debug(root->ch[1]);
141
    }
142
    int main()
143
144
    {
145
         int n,m;
146
         int iCase = 0;
         while(scanf("%d%d",&n,&m) == 2)
147
         {
148
             if(n == 0 \&\& m == 0)break;
149
             iCase++;
150
             memset(F,-1,sizeof(F));
151
             tail = pool;
152
             null = tail++;
153
             null->size = 0; null->ch[0] = null->ch[1] = null->fa = null
154
155
             null->key = 0;
             for(int i = 1;i <= n;i++)scanf("%d",&a[i]);</pre>
156
             for(int i = 0;i < m;i++)
157
158
             {
                 scanf("%d%d",&edge[i].u,&edge[i].v);
159
                 del[i] = false;
160
161
             int 0 = 0;
162
             while(1)
163
164
             {
165
                 scanf("%s",&query[Q].op);
                 if(query[Q].op[0] == 'E')break;
166
```

```
if(query[Q].op[0] == 'D'){
167
168
                     scanf("%d",&query[Q].u);
169
                     query[Q].u—;
170
                     del[query[Q].u] = true;
171
                 else if(query[Q].op[0] == 'Q'){
172
                     scanf("%d%d",&query[Q].u,&query[Q].v);
173
174
175
                 else{
                      scanf("%d%d",&query[Q].u,&query[Q].v);
176
177
                     y[Q] = a[query[Q].u];
                     a[query[Q].u] = query[Q].v;
178
179
                 }
180
                 Q++;
181
             }
182
             for(int i = 1;i <= n;i++){
                 node[i] = tail++;
183
184
                 node[i]->clear();
                 node[i]->key = a[i];
185
                 root[i] = node[i];
186
187
188
             for(int i = 0;i < m;i++)
                 if(!del[i]){
189
                     int u = edge[i].u;
190
                     int v = edge[i].v;
191
                     int t1 = find(u);
192
                     int t2 = find(v);
193
                     if(t1 == t2)continue;
194
195
                     F[t2] = t1;
196
                     merge(root[t1],root[t2]);
197
                 }
             vector<int>ans;
198
             for(int i = Q-1; i >= 0; i---){
199
                 if(query[i].op[0] == 'D'){
200
                     int u = edge[query[i].u].u;
201
202
                     int v = edge[query[i].u].v;
203
                     int t1 = find(u);
                     int t2 = find(v);
204
                     if(t1 == t2)continue;
205
206
                     F[t2] = t1;
                     merge(root[t1],root[t2]);
207
208
209
                 else if(query[i].op[0] == 'Q'){
                     int u = query[i].u;
210
211
                     int k = query[i].v;
                     u = find(u);
212
                     if(k <= 0 || k > root[u]->size){
213
214
                          ans.push_back(0);
                     }
215
216
                     else{
217
                          k = root[u] -> size - k + 1;
```

```
218
                         Node* p = get_kth(root[u],k);
219
                         ans.push_back(p->key);
                     }
220
221
                 }
                 else{
222
223
                     int u = query[i].u;
                     int t1 = find(u);
224
225
                     Node* p = node[u];
226
                     erase(root[t1],p);
                     p->clear();
227
228
                     p\rightarrow key = y[i];
229
                     a[u] = y[i];
                     insert(root[t1],p);
230
231
                 }
232
             }
233
            double ret = 0;
234
             int sz = ans.size();
235
             for(int i = 0;i < sz;i++)ret += ans[i];</pre>
             if(sz)ret /= sz;
236
            printf("Case_wd:_w.6lf\n",iCase,ret);
237
238
239
        return 0;
240 }
    3.5
         动态树
         SPOJQTREE
    3.5.1
    给定一棵 n 个结点的树, 树的边上有权。有两种操作:
    1. 修改一条边上的权值。
    2. 查询两个结点 x 和 y 之间的最短路径中经过的最大的边的权值。
    其中n <= 10^4
  1 |// http://www.spoj.com/problems/QTREE/
    const int MAXN = 10010;
    struct Node *null;
    struct Node{
  4
  5
        Node *fa,*ch[2];
        int Max,key;
  6
  7
        inline void push_up(){
             if(this == null)return;
  8
  9
             Max = max(key, max(ch[0]->Max, ch[1]->Max));
 10
        inline void setc(Node *p,int d){
 11
            ch[d] = p;
 12
             p->fa = this;
 13
 14
        inline bool d(){
 15
 16
             return
                    fa->ch[1] == this;
 17
        inline bool isroot() {
 18
```

```
19
             return fa == null || fa->ch[0] != this && fa->ch[1] != this
20
        inline void rot(){
21
22
             Node *f = fa, *ff = fa->fa;
             int c = d(), cc = fa->d();
23
             f->setc(ch[!c],c);
24
             this—>setc(f,!c);
25
             if(ff->ch[cc] == f)ff->setc(this,cc);
26
             else this->fa = ff;
27
28
             f->push_up();
29
        }
30
        inline Node* splay(){
             while(!isroot()){
31
32
                 if(!fa->isroot())
33
                      d() == fa -> d() ? fa -> rot() : rot();
34
                 rot();
35
             }
36
             push_up();
             return this;
37
38
39
        inline Node* access(){
             for (Node *p = this, *q = null; p != null; q = p, p = p -> fa) {
40
                 p\rightarrow splay()\rightarrow setc(q,1);
41
42
                 p->push_up();
43
44
             return splay();
45
        }
46
   };
47
   Node pool[MAXN],*tail;
   Node *node[MAXN];
48
49
   void init(int n){
        tail = pool;
50
51
        null = tail++;
        null \rightarrow fa = null \rightarrow ch[0] = null \rightarrow ch[1] = null;
52
53
        null->Max = null->key = 0;
54
        for(int i = 1;i <= n;i++){
55
             node[i] = tail++;
             node[i] \rightarrow fa = node[i] \rightarrow ch[0] = node[i] \rightarrow ch[1] = null;
56
57
             node[i]->Max = node[i]->key = 0;
58
        }
59
   struct Edge{
60
61
        int to,next;
62
        int w,id;
   }edge[MAXN*2];
63
   int head[MAXN],tot;
64
    inline int addedge(int u,int v,int w,int id){
65
66
        edge[tot].to = v;
67
        edge[tot].w = w;
        edge[tot].id = id;
68
```

```
69
         edge[tot].next = head[u];
 70
         head[u] = tot++;
 71
 72
    Node *ee[MAXN];
 73
    bool vis[MAXN];
 74
    void bfs(int n){
 75
         for(int i = 1;i <= n;i++)vis[i] = false;</pre>
 76
         queue<int>q;
 77
         q.push(1);
 78
         vis[1] = true;
 79
         while(!q.empty()){
             int u = q.front();
 80
             q.pop();
 81
             for(int i = head[u]; i != -1; i = edge[i].next){
 82
                  int v = edge[i].to;
 83
 84
                  if(vis[v])continue;
                  vis[v] = true;
 85
 86
                  q.push(v);
                  ee[edge[i].id] = node[v];
 87
                  node[v]->key = edge[i].w;
 88
 89
                  node[v]->push_up();
 90
                  node[v]->fa = node[u];
 91
             }
         }
 92
 93
 94
    inline int ask(Node *x,Node *y){
 95
         x->access();
         for(x = null; y != null; x = y, y = y\rightarrowfa){
 96
 97
             y->splay();
 98
             if(y->fa == null)return max(y->ch[1]->Max,x->Max);
 99
             y \rightarrow setc(x,1);
             y->push_up();
100
         }
101
102
103
    int main()
104
     {
105
         int ⊺;
         scanf("%d",&T);
106
         int n;
107
         while (T---)
108
             scanf("%d",&n);
109
             for(int i = 1; i <= n; i++)head[i] = -1;
110
111
             tot = 0;
             init(n);
112
113
             int u, v, w;
             for(int i = 1;i < n;i++){
114
                  scanf("%d%d%d",&u,&v,&w);
115
                  addedge(u,v,w,i);
116
117
                  addedge(v,u,w,i);
118
             }
             bfs(n);
119
```

```
120
             char op[20];
121
             int x,y;
            while(scanf("%s",op) == 1){
122
                 if(strcmp(op,"DONE") == 0)break;
123
                 scanf("%d%d",&x,&y);
124
                 if(op[0] == 'Q'){
125
                     printf("%d\n",ask(node[x],node[y]));
126
127
                 }
128
                 else {
                     ee[x]->splay()->key = y;
129
130
                     ee[x]->push_up();
                 }
131
             }
132
133
        }
134
        return 0;
135 }
    3.5.2 SPOJQTREE2
    给定一棵 n 个结点的树, 树的边上有权。有两种操作:
    1. 查询两个结点 x 和 y 之间的最短路径长度。
    2. 查询从 x 到 y 的最短路径的第 K 条边的长度。
    其中 n <= 10^4
   // http://www.spoj.com/problems/QTREE2/
    const int MAXN = 10010;
  2
    struct Node *null;
    struct Node{
  5
        Node *fa,*ch[2];
  6
        int sum, val;
  7
        int size;
  8
        int id;
  9
        void clear(){
             fa = ch[0] = ch[1] = null;
 10
 11
            sum = val = 0;
 12
             size = 1;
 13
 14
        inline void push_up(){
             if(this == null)return;
 15
             sum = val + ch[0] -> sum + ch[1] -> sum;
 16
 17
            size = ch[0]->size + ch[1]->size + 1;
 18
        }
 19
        inline void setc(Node *p,int d){
            ch[d] = p;
 20
 21
             p->fa = this;
22
        inline bool d(){
 23
 24
             return fa->ch[1] == this;
 25
 26
        inline bool isroot(){
```

```
27
             return fa == null || fa->ch[0] != this && fa->ch[1] != this
28
        inline void rot(){
29
             Node *f = fa, *ff = fa->fa;
30
             int c = d(), cc = fa \rightarrow d();
31
             f->setc(ch[!c],c);
32
             this—>setc(f,!c);
33
             if(ff->ch[cc] == f)ff->setc(this,cc);
34
             else this->fa = ff;
35
36
             f->push_up();
37
        }
38
        inline Node* splay(){
            while(!isroot()){
39
40
                 if(!fa->isroot())
41
                      d() == fa -> d() ? fa -> rot() : rot();
42
                 rot();
43
             }
44
             push_up();
45
             return this;
46
47
        inline Node* access(){
             for (Node *p = this, *q = null; p != null; q = p, p = p -> fa) {
48
                 p\rightarrow splay()\rightarrow setc(q,1);
49
50
                 p->push_up();
51
             }
52
             return splay();
53
        }
54
   };
55
   Node pool[MAXN],*tail;
   Node *node[MAXN];
56
57
   void init(int n){
        tail = pool;
58
59
        null = tail++;
        null \rightarrow fa = null \rightarrow ch[0] = null \rightarrow ch[1] = null;
60
61
        null->size = null->sum = null->val = 0;
62
        for(int i = 1;i <= n;i++){
             node[i] = tail++;
63
            node[i]->id = i;
64
65
             node[i]->clear();
66
        }
67
   struct Edge{
68
69
        int to,next;
70
        int w;
   }edge[MAXN*2];
71
   int head[MAXN],tot;
72
   inline void addedge(int u,int v,int w){
73
74
        edge[tot].to = v;
75
        edge[tot].w = w;
        edge[tot].next = head[u];
76
```

```
77
         head[u] = tot++;
 78
 79
    void dfs(int u,int pre){
 80
         for(int i = head[u];i != -1;i = edge[i].next){
              int v = edge[i].to;
 81
              if(v == pre)continue;
 82
              dfs(v,u);
 83
 84
              node[v]->val = edge[i].w;
 85
              node[v]->push_up();
              node[v] \rightarrow fa = node[u];
 86
         }
 87
 88
    }
 89
    //查询 x->v 的距离
 90
    inline int query_sum(Node *x,Node *y){
 91
         x->access();
 92
         for(x = null; y != null; x = y, y = y\rightarrowfa){
 93
             y->splay();
 94
              if(y->fa == null)
 95
                  return y->ch[1]->sum + x->sum;
 96
             y \rightarrow setc(x,1);
 97
             y->push_up();
 98
         }
 99
    }
    //在 splay 中得到第 k 个点
100
101
    Node* get_kth(Node* r,int k){
102
         Node *x = r;
         while (x->ch[0]->size+1 != k) {
103
              if(k < x->ch[0]->size+1)x = x->ch[0];
104
105
              else {
106
                  k = x->ch[0]->size+1;
                  x = x \rightarrow ch[1];
107
              }
108
         }
109
110
         return x;
111
112
    //查询 x->y 路径上的第 k 个点
113
    inline int query_kth(Node *x,Node *y,int k){
114
         x->access();
         for(x = null; y != null; x = y, y = y\rightarrowfa){
115
116
              y->splay();
              if(y->fa == null){
117
                  if(y->ch[1]->size+1 == k)return y->id;
118
                  else if(y\rightarrow ch[1]\rightarrow size+1 > k)
119
                       return get_kth(y->ch[1],y->ch[1]->size+1-k)->id;
120
                  else return get_kth(x,k-(y->ch[1]->size+1))->id;
121
              }
122
123
             y \rightarrow setc(x,1);
124
             y->push_up();
125
         }
126
    int main()
127
```

```
128 | {
129
         int T,n;
         scanf("%d",&T);
130
         while (T--)
131
             scanf("%d",&n);
132
             for(int i = 1;i <= n;i++)head[i] = -1;</pre>
133
134
             tot = 0;
135
             init(n);
136
             int u,v,w;
137
             for(int i = 1;i < n;i++){
138
                 scanf("%d%d%d",&u,&v,&w);
                 addedge(u,v,w);
139
                 addedge(v,u,w);
140
141
142
             dfs(1,1);
143
             char op[20];
             while(scanf("%s",op) == 1){
144
                 if(strcmp(op,"DONE") == 0)break;
145
                 if(op[0] == 'D'){
146
                      scanf("%d%d",&u,&v);
147
                      printf("%d\n",query_sum(node[u],node[v]));
148
149
                 }
                 else {
150
                      int k; scanf("%d%d%d",&u,&v,&k);
151
                      printf("%d\n",query_kth(node[u],node[v],k));
152
                 }
153
             }
154
155
156
         return 0;
157 |}
```

## 3.5.3 SPOJQTREE4

给定一棵 n 个结点的树,树的边上有权,每个结点有黑白两色,初始时所有的结点都是白的。有两种操作:

1. 对一个结点执行反色操作(白变黑, 黑变白)

2. 查询树中距离最远的两个白点的距离。

其中 $n <= 10^5$ , 查询数目不超过 $10^5$ .

```
1 |//http://www.spoj.com/problems/QTREE4/
  const int MAXN = 100010;
  const int INF = 0x3f3f3f3f;
   struct Node *null;
5
   struct Node{
       Node *fa,*ch[2];
6
7
       multiset<int>st0,st1;//st0 是链,st1 是路径
       int dd,d0;//d0 是该点对应边的长度, dd 是重链长度
8
9
       int w0;//白点值为 0,黑点值为 -INF
       int ls,rs,ms;
10
11
       inline void clear(){
12
           fa = ch[0] = ch[1] = null;
```

```
st0.clear(); st1.clear();
13
14
            st0.insert(-INF);
            st0.insert(-INF);
15
            st1.insert(-INF);
16
            w0 = 0; dd = d0 = 0;
17
            ls = rs = ms = -INF;
18
19
       inline void push_up(){
20
21
            if(this == null)return;
            dd = d0 + ch[0] -> dd + ch[1] -> dd;
22
            int m0 = max(w0,*st0.rbegin()), ml = max(m0,ch[0]->rs+d0),
23
               mr = max(m0, ch[1] -> ls);
            ls = max(ch[0]->ls,ch[0]->dd + d0 + mr);
24
            rs = max(ch[1] -> rs, ch[1] -> dd + ml);
25
            multiset<int>::reverse_iterator it = st0.rbegin();
26
27
            ++it:
            int t0 = max((*st0.rbegin()) + (*it) , *st1.rbegin());
28
29
            if(w0 == 0)
30
                t0 = max(t0, max(0, *st0.rbegin()));
            ms = max(max(ml+ch[1]->ls,mr+d0+ch[0]->rs),max(ch[0]->
31
               ms,ch[1]->ms)),t0);
32
33
       inline void setc(Node *p,int d){
            ch[d] = p;
34
35
            p->fa = this;
36
       inline bool d(){
37
            return fa->ch[1] == this;
38
39
40
       inline bool isroot(){
            return fa == null || fa->ch[0] != this && fa->ch[1] != this
41
42
43
       inline void rot(){
44
            Node *f = fa, *ff = fa -> fa;
45
            int c = d(), cc = fa->d();
46
            f->setc(ch[!c],c);
            this—>setc(f,!c);
47
48
            if(ff->ch[cc] == f)ff->setc(this,cc);
49
            else this—>fa = ff;
50
            f->push_up();
51
       inline Node* splay(){
52
            while(!isroot()){
53
                if(!fa->isroot())
54
                    d()==fa->d() ? fa->rot() : rot();
55
56
                rot();
57
            }
58
            push_up();
59
            return this;
60
       }
```

```
61
         inline Node* access(){
 62
             for (Node *p = this, *q = null; p != null; q = p, p = p -> fa) {
 63
                  p->splay();
                  if(p->ch[1] != null){
 64
                      p->st0.insert(p->ch[1]->ls);
 65
                      p->st1.insert(p->ch[1]->ms);
 66
 67
                  if(q != null){
 68
                      p->st0.erase(p->st0.find(q->ls));
 69
                      p->st1.erase(p->st1.find(q->ms));
 70
 71
 72
                  p\rightarrow setc(q,1);
 73
                  p->push_up();
 74
 75
             return splay();
 76
         }
 77
    };
    Node pool[MAXN],*tail;
 78
    Node *node[MAXN];
 79
    inline void init(int n){
 80
         tail = pool;
 81
 82
         null = tail++;
         null->fa = null->ch[0] = null->ch[1] = null;
 83
         null->st0.clear(); null->st1.clear();
 84
         null->ls = null->rs = null->ms = -INF;
 85
         null \rightarrow w0 = -INF;
 86
         null \rightarrow d0 = null \rightarrow dd = 0;
 87
         for(int i = 1;i <= n;i++){</pre>
 88
 89
             node[i] = tail++;
 90
             node[i]->clear();
         }
 91
 92
    struct Edge{
 93
         int to,next,w;
 94
 95
    }edge[MAXN*2];
    int head[MAXN],tot;
 97
    inline void addedge(int u,int v,int w){
         edge[tot].to = v;
 98
 99
         edge[tot].w = w;
         edge[tot].next = head[u];
100
         head[u] = tot++;
101
102
    inline void dfs(int u,int pre){
103
         for(int i = head[u]; i != -1; i = edge[i].next){
104
             int v = edge[i].to;
105
             if(v == pre)continue;
106
             node[v]->fa = node[u];
107
             node[v]->d0 = edge[i].w;
108
109
             dfs(v,u);
110
             node[u]->st0.insert(node[v]->ls);
             node[u]->st1.insert(node[v]->ms);
111
```

```
112
         }
113
         node[u]->push_up();
114
    template <class T>
115
    inline bool scan_d(T &ret) {
116
117
        char c; int sgn;
        if(c=getchar(),c==EOF) return 0;
118
        while(c!='-'&&(c<'0'||c>'9')) c=getchar();
119
120
        sgn=(c=='-')?-1:1;
121
        ret=(c=='-')?0:(c-'0');
       while(c=getchar(),c>='0'&&c<='9') ret=ret*10+(c-'0');</pre>
122
123
        ret*=sgn;
124
        return 1;
125
    int main()
126
127
    {
128
         int n;
129
         while(scanf("%d",&n) == 1){
             for(int i = 1;i <= n;i++)head[i] = -1;</pre>
130
131
             tot = 0;
             init(n);
132
133
             int u, v, w;
             for(int i = 1;i < n;i++){</pre>
134
135
                  scan_d(u); scan_d(v);scan_d(w);
136
                  addedge(u,v,w);
137
                  addedge(v,u,w);
             }
138
             dfs(1,1);
139
140
             int ans = node[1]->ms;
141
             int Q;
             char op[10];
142
             scanf("%d",&Q);
143
144
             while(Q--){
                  scanf("%s",op);
145
                  if(op[0] == 'C'){
146
147
                      scan_d(u);
148
                      node[u]->access();
149
                      node[u]->splay();
150
                      if(node[u]->w0 == 0)node[u]->w0 = -INF;
                      else node[u]->w0 = 0;
151
                      node[u]->push up();
152
                      ans = node[u]->ms;
153
                  }
154
155
                  else{
156
                      if(ans < 0)puts("They have disappeared.");</pre>
                      else printf("%d\n",ans);
157
                  }
158
             }
159
160
         }
161
         return 0;
162 |}
```

## 3.5.4 SPOJQTREE5

给定一棵 n 个结点的树,边权均为 1。每个结点有黑白两色,初始时所有结点都是黑的。两种查询操作:

- 1. 对一个结点执行反色操作(白变黑,黑变白)
- 2. 查询距离某个特定结点 i 最远的白点的距离。

其中 $<=10^5$ ,查询数目不超过 $10^5$ .

```
1 |//http://www.spoj.com/problems/QTREE5/
   const int MAXN = 100010;
   const int INF = 0x3f3f3f3f;
   struct Node *null;
 5
   struct Node{
 6
        Node *fa,*ch[2];
        multiset<int>st;
 7
 8
        int dd,d0;
        int w0;
 9
        int ls,rs;
10
        inline void clear(){
11
12
            fa = ch[0] = ch[1] = null;
            st.clear();
13
14
            st.insert(INF);
            w0 = INF; dd = d0 = 0;
15
            ls = rs = INF;
16
17
        inline void push_up(){
18
            if(this == null)return;
19
            dd = d0 + ch[0] -> dd + ch[1] -> dd;
20
            int m0 = min(w0,*st.begin()), ml = min(m0,ch[0]->rs+d0), mr
21
                = min(m0,ch[1]->ls);
22
            ls = min(ch[0] -> ls, ch[0] -> dd + d0 + mr);
            rs = min(ch[1]->rs,ch[1]->dd + ml);
23
24
25
        inline void setc(Node *p,int d){
26
            ch[d] = p;
27
            p->fa = this;
28
29
        inline bool d(){
            return fa->ch[1] == this;
30
31
32
        inline bool isroot(){
            return fa == null || fa->ch[0] != this && fa->ch[1] != this
33
34
        inline void rot(){
35
            Node *f = fa, *ff = fa->fa;
36
            int c = d(), cc = fa \rightarrow d();
37
38
            f->setc(ch[!c],c);
39
            this->setc(f,!c);
            if(ff->ch[cc] == f)ff->setc(this,cc);
40
            else this->fa = ff;
41
```

```
42
             f->push_up();
43
44
        inline Node* splay(){
            while(!isroot()){
45
                 if(!fa->isroot())
46
47
                      d() == fa -> d() ? fa -> rot() : rot();
48
                 rot();
             }
49
50
            push_up();
51
             return this;
52
        inline Node* access(){
53
             for(Node *p = this, *q = null; p != null; q = p, p = p->fa){
54
55
                 p->splay();
                 if(p->ch[1] != null){
56
57
                      p->st.insert(p->ch[1]->ls);
58
                 if(q != null){
59
                      p->st.erase(p->st.find(q->ls));
60
61
                 }
62
                 p->setc(q,1);
63
                 p->push_up();
64
             }
65
             return splay();
66
        }
67
   };
68
   Node pool[MAXN],*tail;
69
   Node *node[MAXN];
70
   inline void init(int n){
71
        tail = pool;
        null = tail++;
72
73
        null \rightarrow fa = null \rightarrow ch[0] = null \rightarrow ch[1] = null;
        null->st.clear();
74
75
        null->ls = null->rs = INF;
76
        null->w0 = INF;
77
        null \rightarrow dd = null \rightarrow d0 = 0;
78
        for(int i = 1; i <= n; i++){
             node[i] = tail++;
79
80
             node[i]->clear();
        }
81
82
   }
   struct Edge{
83
84
        int to,next;
85
   }edge[MAXN*2];
   int head[MAXN],tot;
86
   inline void addedge(int u,int v){
87
        edge[tot].to = v;
88
89
        edge[tot].next = head[u];
90
        head[u] = tot++;
91
   inline void dfs(int u,int pre){
92
```

```
93
         for(int i = head[u]; i != -1; i = edge[i].next){
 94
             int v = edge[i].to;
 95
             if(v == pre)continue;
             node[v]->fa = node[u];
 96
 97
             node[v] -> d0 = 1;
             dfs(v,u);
 98
             node[u]->st.insert(node[v]->ls);
 99
100
         }
101
         node[u]->push_up();
102
103
    int main()
104
    {
105
         int n;
         while(scanf("%d",&n) == 1){
106
107
             init(n);
108
             for(int i = 1; i <= n; i++)head[i] = -1;
109
             tot = 0;
             int u,v;
110
             for(int i = 1;i < n;i++){</pre>
111
                  scanf("%d%d",&u,&v);
112
                 addedge(u,v);
113
114
                 addedge(v,u);
115
             }
             dfs(1,1);
116
             int Q;
117
             scanf("%d",&Q);
118
119
             int op;
             while(Q--){
120
121
                 scanf("%d%d",&op,&v);
122
                 if(op == 0){
123
                      node[v]->access();
                      node[v]->splay();
124
                      if(node[v]->w0 == 0)node[v]->w0 = INF;
125
126
                      else node[v]->w0 = 0;
                      node[v]->push_up();
127
128
                 }
129
                 else {
130
                      node[v]->access();
                      node[v]->splay();
131
                      if(node[v]->rs < INF)printf("%d\n",node[v]->rs);
132
                      else printf("-1\n");
133
134
                 }
135
             }
136
         }
137
         return 0;
138 | }
```

## 3.5.5 SPOJQTREE6

给定一棵 n 个结点的树,每个结点有黑白两色,初始时所有结点都是黑的。你被要求支持: 1. 对一个结点执行反色操作(白变黑,黑变白)

2. 询问有多少个点与 u 相连。两个结点 u,v 相连当且仅当 u,v 路径上所有点的颜色相同。其中 $n<=10^5$ ,查询数目不超过 $10^5$ .

```
//http://www.spoj.com/problems/QTREE6/
   const int MAXN = 100010;
   struct Node *null;
 4
   struct Node{
 5
       Node *fa,*ch[2];
 6
       int co;//0 is black, 1 is white
 7
       int lco,rco;
       int ls,rs;
 8
 9
       int s[2];
       int sum[2];//the sum of black and white
10
       inline void clear(){
11
            fa = ch[0] = ch[1] = null;
12
            co = lco = rco = 0;
13
            ls = rs = 1;
14
15
            s[0] = s[1] = 0;
            sum[0] = 1; sum[1] = 0;
16
17
       inline void push_up(){
18
19
            if(this == null)return;
            if(ch[0] != null)lco = ch[0]->lco;
20
21
            else lco = co;
            if(ch[1] != null)rco = ch[1]->rco;
22
23
            else rco = co;
24
            sum[0] = ch[0] -> sum[0] + ch[1] -> sum[0] + (co == 0);
            sum[1] = ch[0] -> sum[1] + ch[1] -> sum[1] + (co == 1);
25
26
            int ml = 1 + s[co] + (co==ch[0]->rco?ch[0]->rs:0);
            int mr = 1 + s[co] + (co==ch[1]->lco?ch[1]->ls:0);
27
28
            ls = ch[0] -> ls;
            if(lco == co && ch[0]->sum[!co] == 0)ls += mr;
29
            rs = ch[1]->rs;
30
31
            if(rco == co && ch[1]->sum[!co] == 0)rs += ml;
32
       inline void setc(Node *p,int d){
33
34
            ch[d] = p;
35
            p->fa = this;
36
       inline bool d(){
37
38
            return fa->ch[1] == this;
39
       inline bool isroot(){
40
            return fa == null || fa->ch[0] != this && fa->ch[1] != this
41
42
43
        inline void rot(){
            Node *f = fa, *ff = fa -> fa;
44
45
            int c = d(), cc = fa \rightarrow d();
            f->setc(ch[!c],c);
46
47
            this—>setc(f,!c);
```

```
48
             if(ff->ch[cc] == f)ff->setc(this,cc);
49
             else this->fa = ff;
             f->push_up();
50
51
        inline Node* splay(){
52
             while(!isroot()){
53
                  if(!fa->isroot())
54
                       d() == fa -> d() ? fa -> rot() : rot();
55
56
                  rot();
57
             }
58
             push_up();
59
             return this;
60
        inline Node* access(){
61
             for (Node *p = this, *q = null; p != null; q = p, p = p->fa){
62
63
                  p->splay();
                  if(p->ch[1] != null)
64
65
                       p->s[p->ch[1]->lco] += p->ch[1]->ls;
                  if(q != null)
66
                       p\rightarrow s[q\rightarrow lco] = q\rightarrow ls;
67
                  p\rightarrow setc(q,1);
68
69
                  p->push_up();
70
             }
71
             return splay();
72
        }
73
   };
74
   Node pool[MAXN],*tail;
75
   Node *node[MAXN];
76
   void init(int n){
77
        tail = pool;
        null = tail++;
78
79
        null \rightarrow fa = null \rightarrow ch[0] = null \rightarrow ch[1] = null;
        null \rightarrow s[0] = null \rightarrow s[1] = 0;
80
81
        null->ls = null->rs = 0;
82
        null \rightarrow sum[0] = null \rightarrow sum[1] = 0;
        null->co = null->lco = null->rco = 0;
83
        for(int i = 1;i <= n;i++){</pre>
84
             node[i] = tail++;
85
86
             node[i]->clear();
        }
87
88
   }
   struct Edge{
89
90
        int to,next;
   }edge[MAXN*2];
91
92
   int head[MAXN],tot;
   inline void addedge(int u,int v){
93
94
        edge[tot].to = v; edge[tot].next = head[u]; head[u] = tot++;
95
   void dfs(int u,int pre){
96
        for(int i = head[u];i != −1;i = edge[i].next){
97
98
             int v = edge[i].to;
```

```
99
             if(v == pre)continue;
100
             node[v]->fa = node[u];
             dfs(v,u);
101
             node[u]->s[node[v]->lco] += node[v]->ls;
102
         }
103
104
         node[u]->push_up();
105
    int main()
106
107
    {
         int n;
108
         while(scanf("%d",&n) == 1){
109
             init(n);
110
             for(int i = 1; i <= n; i++)head[i] = -1;
111
112
             tot = 0;
113
             int u,v;
             for(int i = 1;i < n;i++){</pre>
114
                 scanf("%d%d",&u,&v);
115
                 addedge(u,v);
116
117
                 addedge(v,u);
             }
118
             dfs(1,1);
119
120
             int Q;
121
             int op;
             scanf("%d",&Q);
122
             while(Q--){
123
                 scanf("%d%d",&op,&u);
124
                 if(op == 0){
125
                      node[u]->access();
126
127
                      node[u]->splay();
128
                      printf("%d\n",node[u]->rs);
129
                  }
                 else{
130
131
                      node[u]->access();
132
                      node[u]->splay();
133
                      node[u]->co ^= 1;
134
                      node[u]->push_up();
135
                 }
136
137
             return 0;
138
         }
139
         return 0;
140 }
```

## 3.5.6 SPOJQTREE7

给定一棵 n 个结点的树,每个结点有黑白两色和权值。三种操作:

- 1. 对一个结点执行反色操作(白变黑,黑变白)
- 2. 询问与 u 相连的点中点权的最大值。两个结点 u,v 相连当且仅当 u,v 路径上所有点的颜色相同。
- 3. 改变一个点的点权。

其中 $n <= 10^5$ ,查询数目不超过 $10^5$ .

```
1 //http://www.spoj.com/problems/QTREE7/
   const int MAXN = 100010;
   const int INF = 0x3f3f3f3f;
   struct Node *null;
 5
   struct Node{
 6
       Node *fa,*ch[2];
 7
       int co;
       int lco,rco;
 8
 9
       int ls,rs;
       int w0;
10
       multiset<int>st[2];
11
       int sum[2];
12
       inline void clear(int _co = 0, int _w0 = 0){
13
14
            fa = ch[0] = ch[1] = null;
            co = lco = rco = _co;
15
            w0 = _w0;
16
17
            ls = rs = _w0;
            st[0].clear(); st[1].clear();
18
19
            st[0].insert(-INF); st[1].insert(-INF);
            sum[0] = sum[1] = 0; sum[_co]++;
20
21
       }
       inline void push_up(){
22
            if(this == null)return;
23
            if(ch[0] != null)lco = ch[0]->lco;
24
25
            else lco = co;
            if(ch[1] != null)rco = ch[1]->rco;
26
27
            else rco = co;
28
            sum[0] = ch[0] -> sum[0] + ch[1] -> sum[0] + (co == 0);
29
            sum[1] = ch[0] -> sum[1] + ch[1] -> sum[1] + (co == 1);
            int ml = max(w0,max(*st[co].rbegin(),co==ch[0]->rco?ch[0]->
30
               rs:-INF));
31
            int mr = max(w0,max(*st[co].rbegin(),co==ch[1]->lco?ch[1]->
               ls:-INF));
32
            ls = ch[0] -> ls;
            if(lco == co \&\& ch[0] -> sum[!co] == 0)ls = max(ls,mr);
33
34
            rs = ch[1] - rs;
            if(rco == co && ch[1]->sum[!co] == 0)rs = max(rs,ml);
35
36
       }
       inline void setc(Node *p,int d){
37
            ch[d] = p;
38
39
            p->fa = this;
40
       inline bool d(){
41
42
            return fa->ch[1] == this;
43
        inline bool isroot(){
44
            return fa == null || fa->ch[0] != this && fa->ch[1] != this
45
46
47
       inline void rot(){
```

```
Node *f = fa, *ff = fa -> fa;
48
49
             int c = d(), cc = fa \rightarrow d();
             f->setc(ch[!c],c);
50
             this->setc(f,!c);
51
             if(ff->ch[cc] == f)ff->setc(this,cc);
52
             else this->fa = ff;
53
54
             f->push_up();
55
        }
56
        inline Node* splay(){
             while(!isroot()){
57
                  if(!fa->isroot())
58
                      d()==fa->d() ? fa->rot() : rot();
59
60
                  rot();
             }
61
62
             push_up();
63
             return this;
64
65
        inline Node* access(){
             for(Node *p = this, *q = null; p != null; q = p, p = p->fa){
66
67
                  p->splay();
                  if(p->ch[1] != null)
68
69
                      p\rightarrow st[p\rightarrow ch[1]\rightarrow lco].insert(p\rightarrow ch[1]\rightarrow ls);
70
                  if(q != null)
                      p->st[q->lco].erase(p->st[q->lco].find(q->ls));
71
72
                  p->setc(q,1);
73
                  p->push_up();
74
             }
75
             return splay();
76
        }
77
   };
   Node pool[MAXN],*tail;
78
   Node *node[MAXN];
   int color[MAXN], val[MAXN];
80
81
   void init(int n){
82
        tail = pool;
83
        null = tail++;
        null \rightarrow fa = null \rightarrow ch[0] = null \rightarrow ch[1] = null;
84
        null->st[0].clear(); null->st[1].clear();
85
        null->ls = null->rs = -INF;
86
87
        null \rightarrow sum[0] = null \rightarrow sum[1] = 0;
88
        null->co = null->lco = null->rco = 0;
        for(int i = 1;i <= n;i++){
89
90
             node[i] = tail++;
             node[i]->clear(color[i],val[i]);
91
92
        }
93
94
   struct Edge{
95
        int to,next;
   }edge[MAXN*2];
96
   int head[MAXN],tot;
   inline void addedge(int u,int v){
```

```
99
         edge[tot].to = v; edge[tot].next = head[u]; head[u] = tot++;
100
    void dfs(int u,int pre){
101
102
         for(int i = head[u];i != -1;i = edge[i].next){
             int v = edge[i].to;
103
             if(v == pre)continue;
104
             node[v]->fa = node[u];
105
106
             dfs(v,u);
107
             node[u]->st[node[v]->lco].insert(node[v]->ls);
108
109
         node[u]->push_up();
110
111
    int main()
112
    {
113
         int n;
114
         while(scanf("%d",&n) == 1){
             for(int i = 1; i <= n; i++)head[i] = -1;
115
             tot = 0;
116
117
             int u,v;
             for(int i = 1;i < n;i++){</pre>
118
                 scanf("%d%d",&u,&v);
119
120
                 addedge(u,v);
121
                 addedge(v,u);
122
             }
             for(int i = 1;i <= n;i++)scanf("%d",&color[i]);</pre>
123
             for(int i = 1;i <= n;i++)scanf("%d",&val[i]);</pre>
124
             init(n);
125
             dfs(1,1);
126
127
             int Q;
128
             int w,op;
             scanf("%d",&Q);
129
             while(Q--){
130
                 scanf("%d",&op);
131
132
                 if(op == 0){
                      scanf("%d",&u);
133
134
                      node[u]->access(); node[u]->splay();
135
                      printf("%d\n",node[u]->rs);
136
                 else if(op == 1){
137
138
                      scanf("%d",&u);
                      node[u]->access(); node[u]->splay();
139
140
                      node[u]->co ^= 1;
141
                      node[u]->push_up();
                 }
142
                 else {
143
                      scanf("%d%d",&u,&w);
144
                      node[u]->access(); node[u]->splay();
145
                      node[u]->w0 = w;
146
147
                      node[u]->push_up();
148
                 }
             }
149
```

```
150
        }
151
        return 0;
152 }
    3.5.7 HDU4010
    支持:
    1 \times y : 如果 x,y 不在同一颗子树中,则通过在 x,y 之间连边的方式,连接这两颗子树
    2 \times y : 如果 x,y 在同一颗子树中,且 x!=y, 则将 x 视为这颗子树的根以后,切断 y 与其父亲
    结点的连接
    3 \le x \le y: 如果 x,y 在同一颗子树中,则将 x,y 之间路径上所有点的点权增加 \le
    4 \times y :  如果 x,y 在同一颗子树中,输出 x,y 之间路径上点权的最大值
    非法操作输出 -1
  1 | const int MAXN = 300010;
   const int INF = 0x3f3f3f3f;
    struct Node *null;
    struct Node{
        Node *fa,*ch[2];
  5
        int Max, val;
  6
        int rev;//旋转标记
  7
        int add;
  8
        inline void clear(int _val){
  9
            fa = ch[0] = ch[1] = null;
 10
            val = Max = _val;
 11
 12
            rev = 0;
 13
            add = 0;
 14
 15
        inline void push_up(){
            Max = max(val,max(ch[0]->Max,ch[1]->Max));
 16
 17
        inline void setc(Node *p,int d){
 18
            ch[d] = p;
 19
            p->fa = this;
 20
21
        inline bool d(){
 22
 23
            return fa->ch[1] == this;
 24
        inline bool isroot(){
25
            return fa == null || fa->ch[0] != this && fa->ch[1] != this
 26
27
        }
 28
        //翻转
        inline void flip(){
 29
            if(this == null)return;
 30
            swap(ch[0],ch[1]);
31
            rev ^= 1;
 32
 33
        inline void update_add(int w){
 34
 35
            if(this == null)return;
            val += w;
 36
```

```
37
             add += w;
38
             Max += w;
39
        inline void push_down(){
40
41
             if(rev){
42
                  ch[0]->flip(); ch[1]->flip(); rev = 0;
43
44
             if(add){
45
                  ch[0]->update_add(add); ch[1]->update_add(add);
                  add = 0;
46
47
             }
        }
48
        //直接标记下放
49
        inline void go(){
50
             if(!isroot())fa->go();
51
52
             push down();
53
        inline void rot(){
54
             Node *f = fa, *ff = fa -> fa;
55
             int c = d(), cc = fa \rightarrow d();
56
             f->setc(ch[!c],c);
57
58
             this—>setc(f,!c);
             if(ff->ch[cc] == f)ff->setc(this,cc);
59
             else this->fa = ff;
60
             f->push_up();
61
        }
62
        inline Node* splay(){
63
64
             go();
65
             while(!isroot()){
66
                  if(!fa->isroot())
                      d()==fa->d() ? fa->rot() : rot();
67
68
                  rot();
             }
69
70
             push_up();
71
             return this;
72
73
        inline Node* access(){
74
             for (Node *p = this, *q = null; p != null; q = p, p = p \rightarrow fa) {
75
                  p\rightarrow splay()\rightarrow setc(q,1);
76
                  p->push_up();
77
             }
78
             return splay();
79
        }
80
        //找该点的根
81
        inline Node* find_root(){
82
             Node *x;
83
             for (x = access(); x \rightarrow bush_down(), x \rightarrow ch[0] != null; x = x \rightarrow bush_down()
                ch[0]);
84
             return x;
85
        }
        //变为树根 (换根操作)
86
```

```
void make_root(){
 87
 88
            access()->flip();
        }
89
90
        //切断该点和父亲结点的边
        void cut(){
91
            access();
 92
            ch[0] -> fa = null;
 93
            ch[0] = null;
 94
95
            push_up();
        }
96
        //切断该点以 x 为根时, 该点和父亲结点的根
 97
        //要求这个点和 x 在同一颗树而且不能相同
98
        //x 变为所在树的树根
 99
        void cut(Node* x){
100
            if(this == x || find_root() != x->find_root())
101
102
                puts("-1");
103
            else {
                x->make_root();
104
105
                cut();
            }
106
        }
107
108
        //该点连接到 x
109
        //假如是有虚边信息的,需要先 x->access() 再连接
110
        void link(Node *x){
            if(find_root() == x->find_root())
111
                puts("-1");
112
            else {
113
114
                make_root(); fa = x;
115
            }
116
        }
117
    };
    Node pool[MAXN],*tail;
118
119
    Node *node[MAXN];
120
    struct Edge{
121
        int to,next;
122
    }edge[MAXN*2];
123
    int head[MAXN],tot;
    inline void addedge(int u,int v){
124
125
        edge[tot].to = v;
126
        edge[tot].next = head[u];
127
        head[u] = tot++;
128
129
    void dfs(int u,int pre){
        for(int i = head[u];i != -1;i = edge[i].next){
130
131
            int v = edge[i].to;
            if(v == pre)continue;
132
            node[v]->fa = node[u];
133
            dfs(v,u);
134
135
        }
136
137
    |void ADD(Node *x,Node *y,int w){
```

```
138
         x->access();
139
         for(x = null; y != null; x = y, y = y\rightarrowfa){
              y->splay();
140
141
              if(y->fa == null){
                  y->ch[1]->update_add(w);
142
                  x->update_add(w);
143
144
                  y\rightarrow val += w;
145
                  y->push_up();
146
                  return;
147
148
              y \rightarrow setc(x,1);
149
              y->push_up();
         }
150
151
    int ask(Node *x,Node *y){
152
153
         x->access();
         for(x = null; y != null; x = y, y = y\rightarrowfa){
154
155
              y->splay();
              if(y->fa == null)
156
                  return max(y->val,max(y->ch[1]->Max,x->Max));
157
158
              y\rightarrow setc(x,1);
159
              y->push_up();
         }
160
161
162
    int main()
163
164
         int n;
         while(scanf("%d",&n) == 1){
165
166
              for(int i = 1; i \le n; i++)head[i] = -1;
167
              tot = 0;
168
              int u,v;
              for(int i = 1;i < n;i++){</pre>
169
                  scanf("%d%d",&u,&v);
170
                  addedge(u,v);
171
                  addedge(v,u);
172
173
              }
174
              tail = pool;
              null = tail++;
175
              null->clear(-INF);
176
              for(int i = 1;i <= n;i++){</pre>
177
                  node[i] = tail++;
178
                  scanf("%d",&v);
179
                  node[i]->clear(v);
180
              }
181
182
              dfs(1,1);
183
              int m,op;
              int x,y,w;
184
              scanf("%d",&m);
185
186
              while(m——){
187
                  scanf("%d",&op);
                  if(op == 1){
188
```

```
scanf("%d%d",&x,&y);
189
190
                    node[x]->link(node[y]);
                }
191
192
                else if(op == 2){
                    scanf("%d%d",&x,&y);
193
                    node[y]->cut(node[x]);
194
195
                else if(op == 3){
196
197
                    scanf("%d%d%d",&w,&x,&y);
                    if(node[x]->find_root() != node[y]->find_root())
198
                        printf("-1\n");
199
                    else ADD(node[x],node[y],w);
200
                }
201
                else{
202
203
                    scanf("%d%d",&x,&y);
204
                    if(node[x]->find_root() != node[y]->find_root())
                        printf("-1\n");
205
206
                    else printf("%d\n",ask(node[x],node[y]));
                }
207
208
            }
209
            printf("\n");
210
        }
211
        return 0;
212 |}
        主席树
    3.6
    3.6.1 查询区间多少个不同的数
    查询区间有多少个不同的数(SPOJ DQUERY)
 1
   /*
 2
    * 给出一个序列,查询区间内有多少个不相同的数
 3
    */
    const int MAXN = 30010;
    const int M = MAXN * 100;
    int n,q,tot;
    int a[MAXN];
    int build(int l,int r){
        int root = tot++;
 10
 11
        c[root] = 0;
 12
        if(l != r){
 13
            int mid = (l+r)>>1;
            lson[root] = build(l,mid);
 14
 15
            rson[root] = build(mid+1,r);
 16
 17
        return root;
 18
 19
    int update(int root,int pos,int val){
 20
        int newroot = tot++, tmp = newroot;
        c[newroot] = c[root] + val;
 21
```

```
22
        int l = 1, r = n;
23
        while(l < r){
24
            int mid = (l+r)>>1;
            if(pos <= mid){
25
26
                lson[newroot] = tot++; rson[newroot] = rson[root];
                newroot = lson[newroot]; root = lson[root];
27
                r = mid;
28
29
            }
            else{
30
31
                rson[newroot] = tot++; lson[newroot] = lson[root];
                newroot = rson[newroot]; root = rson[root];
32
33
                l = mid+1;
34
35
            c[newroot] = c[root] + val;
36
        }
37
        return tmp;
38
   int query(int root,int pos){
39
40
        int ret = 0;
        int l = 1, r = n;
41
        while(pos < r){</pre>
42
43
            int mid = (l+r)>>1;
44
            if(pos <= mid){
45
                r = mid;
46
                root = lson[root];
47
            }
48
            else{
49
                ret += c[lson[root]];
50
                root = rson[root];
51
                l = mid+1;
52
            }
53
        }
54
        return ret + c[root];
55
56
   int main(){
57
        while(scanf("%d",&n) == 1){
58
            tot = 0;
59
            for(int i = 1;i <= n;i++)
60
                scanf("%d",&a[i]);
            T[n+1] = build(1,n);
61
            map<int,int>mp;
62
            for(int i = n;i>= 1;i--){
63
64
                if(mp.find(a[i]) == mp.end()){
65
                     T[i] = update(T[i+1],i,1);
                }
66
                else{
67
                     int tmp = update(T[i+1], mp[a[i]], -1);
68
69
                     T[i] = update(tmp,i,1);
70
                }
71
                mp[a[i]] = i;
            }
72
```

```
73
            scanf("%d",&q);
74
            while (q--)
                int l,r;
75
                scanf("%d%d",&l,&r);
76
                printf("%d\n",query(T[l],r));
77
78
            }
79
        }
80
        return 0;
81 |}
   3.6.2 静态区间第 k 大
   POJ2104
 1 | const int MAXN = 100010;
   const int M = MAXN * 30;
   |int n,q,m,tot;
   int a[MAXN], t[MAXN];
   int T[MAXN], lson[M], rson[M], c[M];
   void Init_hash(){
 6
        for(int i = 1; i <= n;i++)</pre>
 7
 8
            t[i] = a[i];
        sort(t+1,t+1+n);
 9
10
        m = unique(t+1,t+1+n)-t-1;
11
12
   int build(int l,int r){
        int root = tot++;
13
14
        c[root] = 0;
        if(l != r){
15
            int mid = (l+r)>>1;
16
17
            lson[root] = build(l,mid);
18
            rson[root] = build(mid+1,r);
19
20
        return root;
21
22
   int hash(int x){
23
        return lower_bound(t+1,t+1+m,x) - t;
24
25
   int update(int root,int pos,int val){
        int newroot = tot++, tmp = newroot;
26
        c[newroot] = c[root] + val;
27
28
        int l = 1, r = m;
29
        while(l < r){</pre>
            int mid = (l+r)>>1;
30
31
            if(pos <= mid){
                lson[newroot] = tot++; rson[newroot] = rson[root];
32
                newroot = lson[newroot]; root = lson[root];
33
34
                r = mid;
35
            }
            else{
36
37
                rson[newroot] = tot++; lson[newroot] = lson[root];
38
                newroot = rson[newroot]; root = rson[root];
```

```
39
                l = mid+1;
40
            }
41
            c[newroot] = c[root] + val;
42
       }
43
       return tmp;
44
45
   int query(int left_root,int right_root,int k){
       int l = 1, r = m;
46
47
       while( l < r){
            int mid = (l+r)>>1;
48
49
            if(c[lson[left_root]]-c[lson[right_root]] >= k ){
                r = mid;
50
51
                left_root = lson[left_root];
                right_root = lson[right_root];
52
            }
53
            else{
54
                l = mid + 1;
55
56
                k -= c[lson[left_root]] - c[lson[right_root]];
                left_root = rson[left_root];
57
58
                right_root = rson[right_root];
            }
59
60
       }
61
       return l;
62
63
   int main(){
       while(scanf("%d%d",&n,&q) == 2){
64
65
            tot = 0;
            for(int i = 1;i <= n;i++)</pre>
66
67
                scanf("%d",&a[i]);
68
            Init_hash();
            T[n+1] = build(1,m);
69
70
            for(int i = n; i ; i—){
71
                int pos = hash(a[i]);
72
                T[i] = update(T[i+1], pos, 1);
            }
73
            while (q--)
74
75
                int l,r,k;
                scanf("%d%d%d",&l,&r,&k);
76
                printf("%d\n",t[query(T[l],T[r+1],k)]);
77
            }
78
79
       }
80
       return 0;
81 |}
   3.6.3 树上路径点权第 k 大
   树上路径点权第 k 大(SPOJ COT)
   LCA + 主席树
 1 //主席树部分 ************
   const int MAXN = 200010;
   const int M = MAXN * 40;
```

```
int n,q,m,TOT;
 5
   int a[MAXN], t[MAXN];
   int T[MAXN], lson[M], rson[M], c[M];
 7
   void Init hash(){
 8
       for(int i = 1; i <= n;i++)
 9
            t[i] = a[i];
       sort(t+1,t+1+n);
10
11
       m = unique(t+1,t+n+1)-t-1;
12
   int build(int l,int r){
13
14
       int root = TOT++;
       c[root] = 0;
15
       if(l != r){
16
17
            int mid = (l+r)>>1;
18
            lson[root] = build(l,mid);
19
            rson[root] = build(mid+1,r);
20
21
       return root;
22
23
   int hash(int x){
24
       return lower_bound(t+1,t+1+m,x) - t;
25
26
   int update(int root,int pos,int val){
       int newroot = TOT++, tmp = newroot;
27
       c[newroot] = c[root] + val;
28
29
       int l = 1, r = m;
       while( l < r){
30
            int mid = (l+r)>>1;
31
32
            if(pos <= mid){
33
                lson[newroot] = TOT++; rson[newroot] = rson[root];
                newroot = lson[newroot]; root = lson[root];
34
35
                r = mid;
            }
36
            else{
37
                rson[newroot] = TOT++; lson[newroot] = lson[root];
38
39
                newroot = rson[newroot]; root = rson[root];
40
                l = mid+1;
41
42
            c[newroot] = c[root] + val;
43
       }
44
       return tmp;
45
46
   int query(int left_root,int right_root,int LCA,int k){
47
       int lca_root = T[LCA];
       int pos = hash(a[LCA]);
48
       int l = 1, r = m;
49
       while(l < r){
50
51
            int mid = (l+r)>>1;
            int tmp = c[lson[left_root]] + c[lson[right_root]] - 2*c[
52
               lson[lca_root]] + (pos >= l && pos <= mid);</pre>
53
            if(tmp >= k){
```

```
left_root = lson[left_root];
 54
 55
                right_root = lson[right_root];
                lca_root = lson[lca_root];
 56
 57
                r = mid;
            }
 58
            else{
 59
                k = tmp;
 60
                left_root = rson[left_root];
 61
 62
                right_root = rson[right_root];
                lca_root = rson[lca_root];
 63
                l = mid + 1;
 64
            }
 65
 66
        }
 67
        return l;
 68
    }
 69
 70
    //LCA 部分
71
    int rmq[2*MAXN];//rmq 数组,就是欧拉序列对应的深度序列
    struct ST{
 72
73
        int mm[2*MAXN];
        int dp[2*MAXN][20];//最小值对应的下标
 74
 75
        void init(int n){
 76
            mm[0] = -1;
            for(int i = 1; i <= n; i++){
 77
 78
                mm[i] = ((i&(i-1)) == 0)?mm[i-1]+1:mm[i-1];
 79
                dp[i][0] = i;
 80
            }
            for(int j = 1; j <= mm[n];j++)
 81
 82
                for(int i = 1; i + (1 << j) - 1 <= n; i++)
                    dp[i][j] = rmq[dp[i][j-1]] < rmq[dp[i+(1<<(j-1))][j]
 83
                       -1]]?dp[i][j-1]:dp[i+(1<<(j-1))][j-1];
        }
 84
        //查询 [a,b] 之间最小值的下标
 85
 86
        int query(int a,int b){
 87
            if(a > b)swap(a,b);
 88
            int k = mm[b—a+1];
 89
            return rmq[dp[a][k]] <= rmq[dp[b-(1<<k)+1][k]]?dp[a][k]:dp[
               b-(1<< k)+1][k];
90
        }
 91
    };
 92
    //边的结构体定义
93
    struct Edge{
 94
        int to,next;
95
    };
    Edge edge[MAXN*2];
96
97
    int tot,head[MAXN];
98
 99
    int F[MAXN*2];//欧拉序列,就是 dfs 遍历的顺序,长度为 2*n-1,下标从 1 开始
    int P[MAXN];//P[i] 表示点 i 在 F 中第一次出现的位置
100
101
    int cnt;
102
```

```
103 | ST st;
104
    void init(){
105
        tot = 0;
106
        memset(head,-1,sizeof(head));
107
    //加边,无向边需要加两次
108
    void addedge(int u,int v){
109
        edge[tot].to = v;
110
111
        edge[tot].next = head[u];
112
        head[u] = tot++;
113
114
    void dfs(int u,int pre,int dep){
115
        F[++cnt] = u;
        rmq[cnt] = dep;
116
117
        P[u] = cnt;
118
        for(int i = head[u];i != -1;i = edge[i].next){
119
             int v = edge[i].to;
120
             if(v == pre)continue;
            dfs(v,u,dep+1);
121
             F[++cnt] = u;
122
123
             rmq[cnt] = dep;
124
        }
125
    }
    //查询 LCA 前的初始化
126
127
    void LCA_init(int root,int node_num){
        cnt = 0;
128
        dfs(root,root,0);
129
130
        st.init(2*node_num-1);
131
    }
132
    //查询 u,v 的 lca 编号
    int query_lca(int u,int v){
133
134
        return F[st.query(P[u],P[v])];
135
136
    void dfs_build(int u,int pre){
137
        int pos = hash(a[u]);
138
        T[u] = update(T[pre],pos,1);
139
        for(int i = head[u]; i != −1; i = edge[i].next){
             int v = edge[i].to;
140
141
             if(v == pre)continue;
142
             dfs_build(v,u);
        }
143
144
145
    int main(){
        while(scanf("%d%d",&n,&q) == 2){
146
             for(int i = 1;i <= n;i++)
147
                 scanf("%d",&a[i]);
148
             Init_hash();
149
             init();
150
            TOT = 0;
151
152
             int u,v;
153
             for(int i = 1;i < n;i++){
```

```
154
                scanf("%d%d",&u,&v);
155
                addedge(u,v);
                addedge(v,u);
156
157
            }
            LCA_init(1,n);
158
            T[n+1] = build(1,m);
159
            dfs_build(1,n+1);
160
            int k;
161
162
            while (q--)
                scanf("%d%d%d",&u,&v,&k);
163
164
                printf("%d\n",t[query(T[u],T[v],query_lca(u,v),k)]);
165
166
            return 0;
167
        }
168
        return 0;
169 }
    3.6.4 动态第 k 大
    树状数组套主席树 ZOJ 2112
  1 | const int MAXN = 60010;
    const int M = 2500010;
   int n,q,m,tot;
    int a[MAXN], t[MAXN];
    int S[MAXN];
  7
    struct Query{
        int kind;
  8
  9
        int l,r,k;
 10
    }query[10010];
 11
 12
    void Init_hash(int k){
 13
        sort(t,t+k);
 14
        m = unique(t,t+k) - t;
 15
 16
    int hash(int x){
 17
        return lower_bound(t,t+m,x)-t;
 18
 19
    int build(int l,int r){
        int root = tot++;
 20
        c[root] = 0;
 21
22
        if(l != r){
            int mid = (l+r)/2;
 23
 24
            lson[root] = build(l,mid);
 25
            rson[root] = build(mid+1,r);
26
 27
        return root;
 28
    }
29
 30
    int Insert(int root,int pos,int val){
 31
        int newroot = tot++, tmp = newroot;
```

```
32
       int l = 0, r = m-1;
33
       c[newroot] = c[root] + val;
       while(l < r){
34
            int mid = (l+r)>>1;
35
            if(pos <= mid){
36
                lson[newroot] = tot++; rson[newroot] = rson[root];
37
                newroot = lson[newroot]; root = lson[root];
38
                r = mid;
39
40
            }
            else{
41
42
                rson[newroot] = tot++; lson[newroot] = lson[root];
43
                newroot = rson[newroot]; root = rson[root];
44
                l = mid+1;
45
46
            c[newroot] = c[root] + val;
47
       }
48
       return tmp;
49
   }
50
   int lowbit(int x){
51
52
       return x&(-x);
53
54
   int use[MAXN];
   void add(int x,int pos,int val){
55
56
       while(x <= n){
57
            S[x] = Insert(S[x],pos,val);
            x += lowbit(x);
58
59
       }
60
   }
61
   int sum(int x){
       int ret = 0;
62
       while(x > 0){
63
            ret += c[lson[use[x]]];
64
65
            x = lowbit(x);
66
67
       return ret;
68
   int Query(int left,int right,int k){
69
70
       int left_root = T[left-1];
       int right_root = T[right];
71
72
       int l = 0, r = m-1;
       for(int i = left-1;i;i -= lowbit(i)) use[i] = S[i];
73
74
       for(int i = right;i ;i -= lowbit(i)) use[i] = S[i];
75
       while(l < r){
            int mid = (l+r)/2;
76
            int tmp = sum(right) - sum(left-1) + c[lson[right_root]] -
77
               c[lson[left_root]];
78
            if(tmp >= k){
79
                r = mid;
80
                for(int i = left-1; i ;i -= lowbit(i))
                    use[i] = lson[use[i]];
81
```

```
82
                 for(int i = right; i; i -= lowbit(i))
 83
                      use[i] = lson[use[i]];
                 left_root = lson[left_root];
 84
 85
                 right_root = lson[right_root];
             }
 86
             else{
 87
                 l = mid+1;
 88
 89
                 k = tmp;
 90
                 for(int i = left-1; i;i -= lowbit(i))
                      use[i] = rson[use[i]];
 91
                 for(int i = right;i ;i -= lowbit(i))
 92
                      use[i] = rson[use[i]];
 93
 94
                 left_root = rson[left_root];
 95
                 right_root = rson[right_root];
             }
 96
 97
         }
 98
         return l;
 99
100
    void Modify(int x,int p,int d){
        while(x <= n){
101
             S[x] = Insert(S[x],p,d);
102
103
             x += lowbit(x);
104
         }
    }
105
106
    int main(){
107
         int Tcase;
108
         scanf("%d",&Tcase);
109
110
         while(Tcase—){
111
             scanf("%d%d",&n,&q);
112
             tot = 0;
             m = 0;
113
114
             for(int i = 1; i <= n; i++){
                 scanf("%d",&a[i]);
115
                 t[m++] = a[i];
116
117
             }
             char op[10];
118
119
             for(int i = 0;i < q;i++){
                 scanf("%s",op);
120
121
                 if(op[0] == 'Q'){
                      query[i].kind = 0;
122
                      scanf("%d%d%d",&query[i].l,&query[i].r,&query[i].k)
123
                 }
124
                 else{
125
126
                      query[i].kind = 1;
                      scanf("%d%d",&query[i].l,&query[i].r);
127
                      t[m++] = query[i].r;
128
129
                 }
130
131
             Init_hash(m);
```

```
132
             T[0] = build(0,m-1);
             for(int i = 1;i <= n;i++)</pre>
133
                 T[i] = Insert(T[i-1],hash(a[i]),1);
134
             for(int i = 1;i <= n;i++)</pre>
135
                 S[i] = T[0];
136
             for(int i = 0;i < q;i++){
137
                 if(query[i].kind == 0)
138
                      printf("%d\n",t[Query(query[i].l,query[i].r,query[i
139
                         ].k)]);
                 else{
140
                      Modify(query[i].l,hash(a[query[i].l]),-1);
141
                     Modify(query[i].l,hash(query[i].r),1);
142
                      a[query[i].l] = query[i].r;
143
144
                 }
             }
145
146
         }
147
         return 0;
148 |}
    3.7
         Treap
    ZOJ3765
    long long gcd(long long a,long long b){
  2
         if(b == 0)return a;
  3
         else return gcd(b,a%b);
  4
    const int MAXN = 300010;
  5
    int num[MAXN],st[MAXN];
    struct Treap{
  7
  8
         int tot1;
  9
         int s[MAXN],tot2;//内存池和容量
         int ch[MAXN][2];
 10
         int key[MAXN],size[MAXN];
 11
         int sum0[MAXN],sum1[MAXN];
 12
         int status[MAXN];
 13
         void Init(){
 14
             tot1 = tot2 = 0;
 15
 16
             size[0] = 0;
 17
             ch[0][0] = ch[0][1] = 0;
 18
             sum0[0] = sum1[0] = 0;
 19
         bool random(double p){
 20
 21
             return (double)rand() / RAND_MAX < p;</pre>
 22
         int newnode(int val,int _status){
 23
 24
             int r;
 25
             if(tot2)r = s[tot2—];
 26
             else r = ++tot1;
             size[r] = 1;
 27
 28
             key[r] = val;
 29
             status[r] = _status;
```

```
30
            ch[r][0] = ch[r][1] = 0;
31
            sum0[r] = sum1[r] = 0;//需要push_up
32
            return r;
33
        }
34
        void del(int r){
            if(!r)return;
35
36
            s[++tot2] = r;
37
            del(ch[r][0]);
38
            del(ch[r][1]);
39
        void push_up(int r){
40
            int lson = ch[r][0], rson = ch[r][1];
41
            size[r] = size[lson] + size[rson] + 1;
42
            sum0[r] = gcd(sum0[lson],sum0[rson]);
43
            sum1[r] = gcd(sum1[lson],sum1[rson]);
44
45
            if(status[r] == 0)
                sum0[r] = gcd(sum0[r],key[r]);
46
47
            else sum1[r] = gcd(sum1[r],key[r]);
48
49
        void merge(int &p,int x,int y){
50
            if(!x || !y)
51
                p = x | y;
52
            else if(random((double)size[x]/(size[x]+size[y]))){
                merge(ch[x][1],ch[x][1],y);
53
54
                push_up(p=x);
55
            }
            else{
56
57
                merge(ch[y][0],x,ch[y][0]);
58
                push_up(p=y);
59
            }
60
61
        void split(int p,int &x,int &y,int k){
62
            if(!k){
                x = 0; y = p;
63
64
                return;
65
66
            if(size[ch[p][0]] >= k){
                y = p;
67
                split(ch[p][0],x,ch[y][0],k);
68
69
                push_up(y);
            }
70
            else{
71
72
73
                split(ch[p][1], ch[x][1], y, k - size[ch[p][0]] - 1);
                push_up(x);
74
75
            }
76
        void build(int &p,int l,int r){
77
            if(l > r)return;
78
79
            int mid = (l + r)/2;
            p = newnode(num[mid],st[mid]);
80
```

```
81
             build(ch[p][0],l,mid-1);
 82
             build(ch[p][1],mid+1,r);
 83
             push_up(p);
 84
        }
        void debug(int root){
 85
             if(root == 0)return;
 86
             printf("%duuu左儿子: %duu右儿子:u%duusizeu=u%duukeyu=u%d\n",
 87
                root,ch[root][0],ch[root][1],size[root],key[root]);
 88
             debug(ch[root][0]);
 89
             debug(ch[root][1]);
 90
        }
 91
    };
    Treap T;
 92
    char op[10];
 93
    int main(){
 94
 95
        int n,q;
        while(scanf("%d%d",&n,&q) == 2){
 96
 97
             int root = 0;
 98
             T.Init();
 99
             for(int i = 1;i <= n;i++)
                 scanf("%d%d",&num[i],&st[i]);
100
101
             T.build(root,1,n);
102
             while (q--)
                 scanf("%s",op);
103
                 if(op[0] == 'Q'){
104
                     int l,r,s;
105
                     scanf("%d%d%d",&l,&r,&s);
106
107
                     int x,y,z;
108
                     T.split(root,x,z,r);
                     T.split(x,x,y,l-1);
109
                     if(s == 0)
110
                          printf("%d\n",T.sum0[y] == 0? -1:T.sum0[y]);
111
112
                     else
113
                          printf("%d\n",T.sum1[y] == 0?-1:T.sum1[y]);
114
                     T.merge(x,x,y);
115
                     T.merge(root,x,z);
116
                 }
117
                 else if(op[0] == 'I'){
                     int v,s,loc;
118
119
                     scanf("%d%d%d",&loc,&v,&s);
120
                     int x,y;
121
                     T.split(root,x,y,loc);
                     T.merge(x,x,T.newnode(v,s));
122
123
                     T.merge(root,x,y);
124
                 else if(op[0] == 'D'){
125
                     int loc;
126
                     scanf("%d",&loc);
127
128
                     int x,y,z;
129
                     T.split(root,x,z,loc);
130
                     T.split(x,x,y,loc-1);
```

```
131
                     T.del(y);
132
                     T.merge(root,x,z);
                 }
133
                 else if(op[0] == 'R'){
134
                      int loc;
135
                      scanf("%d",&loc);
136
                      int x,y,z;
137
                      T.split(root,x,z,loc);
138
139
                     T.split(x,x,y,loc-1);
                      T.status[y] = 1-T.status[y];
140
141
                     T.push_up(y);
                     T.merge(x,x,y);
142
                     T.merge(root,x,z);
143
144
                 }
                 else{
145
                      int loc,v;
146
                      scanf("%d%d",&loc,&v);
147
148
                      int x,y,z;
                      T.split(root,x,z,loc);
149
                      T.split(x,x,y,loc-1);
150
                     T.key[y] = v;
151
152
                     T.push_up(y);
                     T.merge(x,x,y);
153
154
                      T.merge(root,x,z);
155
                 }
             }
156
157
         }
158
         return 0;
159 |}
    3.8
         KD 树
    3.8.1 HDU4347 K 近邻
    模板题, 求出最近的 K 个点。
  1 | const int MAXN = 50010;
    const int DIM = 10;
    inline double sqr(double x){return x*x;}
  4
    namespace KDTree{
         int K;//维数
  5
         struct Point{
  6
  7
             int x[DIM];
  8
             double distance(const Point &b)const{
                 double ret = 0;
  9
 10
                 for(int i = 0;i < K;i++)
 11
                      ret += sqr(x[i]-b.x[i]);
 12
                 return ret;
 13
             }
 14
             void input(){
 15
                 for(int i = 0;i < K;i++)scanf("%d",&x[i]);</pre>
 16
             }
```

```
void output(){
17
18
                 for(int i = 0;i < K;i++)</pre>
19
                     printf("%d%c",x[i],i < K-1?'':'\n');</pre>
20
            }
21
        };
        struct qnode{
22
23
            Point p;
24
            double dis;
25
            qnode(){}
            qnode(Point _p,double _dis){
26
27
                 p = p; dis = dis;
28
29
            bool operator <(const qnode &b)const{</pre>
                 return dis < b.dis;</pre>
30
            }
31
32
        };
33
        priority_queue<qnode>q;
34
        struct cmpx{
            int div;
35
            cmpx(const int &_div){div = _div;}
36
            bool operator()(const Point &a,const Point &b){
37
38
                 for(int i = 0;i < K;i++)</pre>
39
                     if(a.x[(div+i)%K] != b.x[(div+i)%K])
                          return a.x[(div+i)%K] < b.x[(div+i)%K];</pre>
40
41
                 return true;
            }
42
43
        };
44
        bool cmp(const Point &a,const Point &b,int div){
45
            cmpx cp = cmpx(div);
46
            return cp(a,b);
47
48
        struct Node{
49
            Point e;
50
            Node *lc,*rc;
51
            int div;
52
        }pool[MAXN],*tail,*root;
53
        void init(){
54
            tail = pool;
55
        Node* build(Point *a,int l,int r,int div){
56
57
            if(l >= r)return NULL;
            Node *p = tail++;
58
59
            p->div = div;
60
            int mid = (l+r)/2;
            nth_element(a+l,a+mid,a+r,cmpx(div));
61
            p\rightarrow e = a[mid];
62
            p->lc = build(a,l,mid,(div+1)%K);
63
            p->rc = build(a,mid+1,r,(div+1)%K);
64
65
            return p;
66
        void search(Point p, Node *x, int div, int m){
67
```

```
68
             if(!x)return;
 69
             if(cmp(p,x->e,div)){
                  search(p,x->lc,(div+1)%K,m);
 70
                  if(q.size() < m){
 71
 72
                      q.push(qnode(x->e,p.distance(x->e)));
 73
                      search(p,x->rc,(div+1)\%K,m);
 74
                  }
                  else {
 75
 76
                      if(p.distance(x->e) < q.top().dis){</pre>
 77
                           q.pop();
                           q.push(qnode(x\rightarrow e,p.distance(x\rightarrow e)));
 78
 79
                      if(sqr(x->e.x[div]-p.x[div]) < q.top().dis)</pre>
 80
                           search(p,x->rc,(div+1)\%K,m);
 81
                  }
 82
             }
 83
             else {
 84
 85
                  search(p,x->rc,(div+1)%K,m);
                  if(q.size() < m){
 86
 87
                      q.push(qnode(x->e,p.distance(x->e)));
 88
                      search(p,x->lc,(div+1)%K,m);
 89
                  }
                  else {
 90
 91
                      if(p.distance(x->e) < q.top().dis){</pre>
 92
                           q.pop();
 93
                           q.push(qnode(x->e,p.distance(x->e)));
 94
 95
                      if(sqr(x->e.x[div]-p.x[div]) < q.top().dis)</pre>
 96
                           search(p,x->lc,(div+1)%K,m);
 97
                  }
             }
 98
 99
         }
         void search(Point p,int m){
100
             while(!q.empty())q.pop();
101
             search(p,root,0,m);
102
103
         }
104
    };
105
    KDTree::Point p[MAXN];
106
    int main()
107
         int n,k;
108
         while(scanf("%d%d",&n,&k) == 2){
109
110
             KDTree::K = k;
             for(int i = 0;i < n;i++)p[i].input();</pre>
111
112
             KDTree::init();
113
             KDTree::root = KDTree::build(p,0,n,0);
             int 0;
114
             scanf("%d",&Q);
115
116
             KDTree::Point o;
117
             while(Q--){
118
                  o.input();
```

```
int m;
119
                  scanf("%d",&m);
120
121
                 KDTree::search(o,m);
                 printf("the_closest_%d_points_are:\n",m);
122
                 int cnt = 0;
123
                 while(!KDTree::q.empty()){
124
125
                      p[cnt++] = KDTree::q.top().p;
126
                      KDTree::q.pop();
127
                  }
                 for(int i = 0;i < m;i++)p[m-1-i].output();</pre>
128
129
             }
130
131
         return 0;
132 }
```

#### 3.8.2 CF44G

给定若干个靶子 (xl,xr,yl,yr,z),z 为该靶子离射击位置的距离,所有靶子都可以看成是二维平面上平行于坐标轴的矩形。然后按顺序给定若干个子弹的射击位置 (x,y),子弹射到一个靶子就会将靶子打碎,并掉落到地上。问每个子弹射到的靶子是谁。保证靶子的 z 值不相同。

找矩形内权值最小的点,支持删除操作

```
1 | const int MAXN = 100010;
   const int INF = 0x3f3f3f3f;
 3
   struct Point{
        int x,y,id;
 4
 5
        bool operator ==(const Point &b)const{
 6
            return x == b.x && y == b.y && id == b.id;
 7
        }
 8
   };
 9
   struct Node{
10
        Point e;
11
        Node *lc,*rc;
        bool div;
12
13
        int sub,cur;
        int size:
14
15
        bool exist;
16
        void push_up(){
            size = lc->size + rc->size + exist;
17
18
            sub = min(cur,min(lc->sub,rc->sub));
19
20
   }pool[MAXN],*root,*tail,*null;
21
   inline bool cmpX(const Point &a,const Point &b) {return a.x < b.x | |
        (a.x == b.x \&\& a.y < b.y) \mid | (a.x == b.x \&\& a.y == b.y \&\& a.id)
      < b.id);}
   inline bool cmpY(const Point &a,const Point &b){return a.y < b.y ||</pre>
22
        (a.y == b.y \&\& a.x < b.x) \mid | (a.y == b.y \&\& a.x == b.x \&\& a.id)
      < b.id);}
   inline bool cmp(const Point &a,const Point &b,bool div){return div?
23
      cmpY(a,b):cmpX(a,b);
24 | Node* build(Point *a, int l, int r, bool div){
```

```
25
        if(l >= r)return null;
26
        Node *p = tail++;
27
        p->div = div;
        int mid = (l+r)/2;
28
29
        nth_element(a+l,a+mid,a+r,div?cmpY:cmpX);
        p\rightarrow e = a[mid];
30
        p->lc = build(a,l,mid,!div);
31
32
        p->rc = build(a,mid+1,r,!div);
33
        p\rightarrow exist = 1;
34
        p\rightarrow cur = p\rightarrow e.id;
35
        p->push_up();
36
        return p;
37
38
   void remove(Node *p,Point o){
        if(p->e == o){
39
40
            p\rightarrow exist = 0;
41
            p->cur = INF;
42
            p->size--;
43
        }
        else {
44
45
            if(cmp(p->e,o,p->div))remove(p->rc,o);
46
            else remove(p->lc,o);
47
48
        p->push_up();
49
   int getMin(Node *p,int xl,int xr,int yl,int yr,int minx,int maxx,
50
      int miny,int maxy){
51
        if(p == null || p->size == 0)return INF;
52
        if(xl <= minx && xr >= maxx && yl <= miny && yr >= maxy)return
           p->sub;
        if(xl > maxx || xr < minx || yl > maxy || yr < miny)return INF;</pre>
53
        int ret = INF;
54
        if(p->e.x >= xl && p->e.x <= xr && p->e.y >= yl && p->e.y <= yr</pre>
55
56
            ret = min(ret,p->cur);
57
        if(p->div){
58
            if(yl <= p->e.y)
59
                ret = min(ret,getMin(p->lc,xl,xr,yl,min(yr,p->e.y),minx
                    ,maxx,miny,min(maxy,p->e.y)));
60
            if(yr >= p->e.y)
61
                 ret = min(ret,getMin(p->rc,xl,xr,max(yl,p->e.y),yr,minx
                    ,maxx,max(miny,p->e.y),maxy));
        }
62
63
        else {
            if(xl <= p->e.x)
64
                 ret = min(ret,getMin(p->lc,xl,min(xr,p->e.x),yl,yr,minx
65
                    ,min(maxx,p->e.x),miny,maxy));
66
            if(xr >= p->e.x)
                 ret = min(ret,getMin(p->rc,max(xl,p->e.x),xr,yl,yr,max(
67
                    minx,p->e.x),maxx,miny,maxy));
        }
68
```

```
69
         return ret;
70
 71
    Point pp[MAXN],pp2[MAXN];
    struct REC{
 72
 73
         int xl,xr,yl,yr,z;
 74
         int id;
 75
         void input(){
 76
             scanf("%d%d%d%d%d",&xl,&xr,&yl,&yr,&z);
 77
 78
         bool operator <(const REC &b)const{</pre>
 79
             return z < b.z;
 80
         }
    }rec[MAXN];
 81
    int ans[MAXN];
 82
    int main()
 83
 84
    {
 85
         int n,m;
 86
         while(scanf("%d",&n) == 1){
 87
             for(int i = 0;i < n;i++){</pre>
 88
                 rec[i].input();
                 rec[i].id = i+1;
 89
 90
             }
             sort(rec,rec+n);
 91
             scanf("%d",&m);
 92
             for(int i = 0;i < m;i++){</pre>
 93
 94
                 scanf("%d%d",&pp[i].x,&pp[i].y);
 95
                 pp[i].id = i;
 96
                 pp2[i] = pp[i];//备份
 97
             }
 98
             tail = pool;
 99
             null = tail++;
100
             null->size = 0;
             null->sub = null->cur = INF;
101
102
             null->lc = null->rc = null;
             root = build(pp,0,m,0);
103
104
             memset(ans,0,sizeof(ans));
105
             for(int i = 0; i < n; i++){
                 int tmp = getMin(root,rec[i].xl,rec[i].xr,rec[i].yl,rec
106
                     [i].yr,-INF,INF,-INF,INF);
                 if(tmp == INF)continue;
107
                 ans[tmp] = rec[i].id;
108
                 remove(root,pp2[tmp]);
109
110
             }
111
             for(int i = 0;i < m;i++)printf("%d\n",ans[i]);</pre>
112
113
         return 0;
114 }
```

#### 3.8.3 HDU4742

三维 LIS。即每个点有个三维坐标,两个点能放在一前一后当且仅当 xi < xj, yi < yj, zi < zj, 求最长的序列,并该条件下的方案数。

```
1 | const int MAXN = 100010;
   const int MOD = 1<<30;</pre>
   const int INF = 0x7ffffffff;//这个一定要够大
   struct Node{
       pair<int,int>e,sub,cur;
 5
 6
       bool div:
 7
       Node *lc,*rc;
   };
 8
   Node pool[MAXN],*tail;
   Node *root;
10
11 | bool cmpX(const pair<int,int> &a,const pair<int,int> &b){return a.
      first < b.first || (a.first == b.first && a.second < b.second)</pre>
12 | bool cmpY(const pair<int,int> &a,const pair<int,int> &b){return a.
      second < b.second || (a.second == b.second && a.first < b.first)</pre>
   bool cmp(const pair<int,int> &a,const pair<int,int> &b,bool div){
13
      return div?cmpY(a,b):cmpX(a,b);}
   Node* build(pair<int,int> *a,int l,int r,bool div){
14
       if(l >= r)return NULL;
15
16
       Node *p = tail++;
       p->div = div;
17
18
       int mid = (l+r)/2;
19
       nth_element(a+l,a+mid,a+r,div?cmpY:cmpX);
       p\rightarrow e = a[mid];
20
       p->cur = p->sub = make_pair(0,0);
21
22
       p->lc = build(a,l,mid,!div);
       p->rc = build(a,mid+1,r,!div);
23
24
       return p;
25
   inline void update(pair<int,int> &a,pair<int,int> b){
26
       if(a.first < b.first)a = b;</pre>
27
       else if(a.first == b.first){
28
            a.second += b.second;
29
30
            if(a.second >= MOD)a.second -= MOD;
       }
31
32
   void add(Node *p,pair<int,int> e,pair<int,int> v){
33
       update(p->sub,v);
34
       if(e == p->e){
35
36
            update(p->cur,v);
37
            return;
       }
38
39
       else {
40
            if(cmp(p->e,e,p->div))add(p->rc,e,v);
            else add(p->lc,e,v);
41
42
       }
```

```
43
   }
44
   pair<int,int>ans;
   //查询最大值
45
   void get(Node *p,pair<int,int>e,int maxx,int maxy){
46
47
        if(!p)return;
48
        if(p->sub.first < ans.first)return;</pre>
49
        if(maxx <= e.first && maxy <= e.second)</pre>
            update(ans,p->sub);
50
51
        else {
            if(p->e.first <= e.first && p->e.second <= e.second)update(</pre>
52
               ans,p->cur);
            if(p->div){
53
54
                if(p->e.second <= e.second)get(p->rc,e,maxx,maxy);
55
                get(p->lc,e,maxx,min(maxy,p->e.second));
56
            }
57
            else {
                if(p->e.first <= e.first)get(p->rc,e,maxx,maxy);
58
59
                get(p->lc,e,min(maxx,p->e.first),maxy);
60
            }
        }
61
62
   struct TNode{
63
64
        int x,y,z;
        void input(){
65
            scanf("%d%d%d",&x,&y,&z);
66
67
68
        bool operator < (const TNode &b)const{</pre>
69
            if(x != b.x)return x < b.x;
70
            else if(y != b.y)return y < b.y;</pre>
71
            else return z < b.z;
72
73
   }node[MAXN];
   pair<int,int>p[MAXN];
74
75
   pair<int,int>dp[MAXN];
76
   int main()
77
   {
78
        int ⊺;
79
        int n;
        scanf("%d",&T);
80
81
        while (T--)
82
            scanf("%d",&n);
83
            int cnt = 0;
            for(int i = 0;i < n;i++){
84
85
                node[i].input();
                p[cnt++] = make_pair(node[i].y,node[i].z);
86
87
            }
88
            sort(node,node+n);
89
            sort(p,p+cnt);
90
            cnt = unique(p,p+cnt)-p;
91
            tail = pool;
            root = build(p,0,cnt,0);
92
```

```
93
            for(int i = 0;i < n;i++)dp[i] = make_pair(1,1);</pre>
94
            for(int i = 0;i < n;i++){</pre>
95
                 ans = make_pair(0,0);
96
                 get(root, make_pair(node[i].y, node[i].z), INF, INF);
97
                 ans.first++;
                 update(dp[i],ans);
98
                 add(root,make_pair(node[i].y,node[i].z),dp[i]);
 99
100
101
            printf("%d<sub>\\\\\</sub>%d\n",root->sub.first,root->sub.second);
102
103
        return 0;
104 |}
    3.9
         替罪羊树 (ScapeGoat Tree)
          CF455D
    3.9.1
    http://codeforces.com/contest/455/problem/D
    题意:给了一个序列,1 操作把一个区间的末尾的数插入到头部,2 操作是询问一个区间里
    面等于某个数的个数。
    使用替罪羊树,里面套一个 map 来统计区间的个数。
  1 | const int MAXN = 200010;
    const double alpha = 0.75;
  3
    struct Node{
  4
        Node *ch[2];
  5
        int size,key,nodeCount;
  6
        bool exist;
        map<int,int>mp;
  7
        bool isBad(){
  8
  9
            return ch[0]->nodeCount > alpha*nodeCount+5 || ch[1]->
               nodeCount > alpha*nodeCount + 5;
 10
 11
        void push_up(){
            size = exist + ch[0]->size + ch[1]->size;
 12
            nodeCount = 1 + ch[0]->nodeCount + ch[1]->nodeCount;
 13
 14
            mp.clear();
 15
            if(exist)mp[key]++;
            for(map<int,int>::iterator it = ch[0]->mp.begin();it != ch
 16
               [0]->mp.end();it++)
                 mp[(*it).first] += (*it).second;
 17
            for(map<int,int>::iterator it = ch[1]->mp.begin();it != ch
 18
               [1]->mp.end();it++)
                 mp[(*it).first] += (*it).second;
 19
 20
        }
 21
    };
 22
    struct ScapeGoatTree{
        Node pool[MAXN];
23
        Node *tail,*root,*null;
 24
 25
        Node *bc[MAXN];//内存回收
        int bc_top;
 26
 27
        void init(){
```

```
28
            tail = pool;
29
            null = tail++;
            null->ch[0] = null->ch[1] = null;
30
            null->size = null->key = null->nodeCount = 0;
31
            null->mp.clear();
32
33
            root = null;
34
            bc_top = 0;
35
        }
36
        inline Node *newNode(int key){
            Node *p;
37
            if(bc_top)p = bc[--bc_top];
38
            else p = tail++;
39
            p\rightarrow ch[0] = p\rightarrow ch[1] = null;
40
            p->size = p->nodeCount = 1;
41
            p->key = key;
42
43
            p->exist = true;
44
            p->mp.clear();
45
            p->mp[key] = 1;
46
            return p;
47
        Node *buildTree(int *a,int l,int r){
48
49
            if(l >= r)return null;
            int mid = (l+r)>>1;
50
            Node *p = newNode(a[mid]);
51
52
            p->ch[0] = buildTree(a,l,mid);
            p->ch[1] = buildTree(a,mid+1,r);
53
            p->push_up();
54
            return p;
55
56
        }
57
        inline void Travel(Node *p,vector<Node *>&v){
            if(p == null)return;
58
59
            Travel(p \rightarrow ch[0], v);
            if(p->exist)v.push_back(p);
60
61
            else bc[bc_top++] = p;
62
            Travel(p \rightarrow ch[1], v);
63
64
        inline Node *divide(vector<Node *>&v,int l,int r){
            if(l >= r)return null;
65
            int mid = (l+r)/2;
66
            Node *p = v[mid];
67
            p->ch[0] = divide(v,l,mid);
68
            p->ch[1] = divide(v,mid+1,r);
69
70
            p->push_up();
71
            return p;
72
        //重构,注意 p 要引用
73
        inline void rebuild(Node *&p){
74
75
            vector<Node *>v;
            Travel(p,v);
76
77
            p = divide(v,0,v.size());
        }
78
```

```
79
         //删除第 id 个元素,返回第 id 个元素的值
 80
         inline int erase(Node *p,int id){
             if(p->exist && id == p->ch[0]->size + 1){
 81
 82
                 p\rightarrow exist = 0;
                 p->mp[p->key]--;
 83
                 p->size--;
 84
 85
                 return p->key;
 86
             }
 87
             p->size--;
 88
             int res;
 89
             if(p->ch[0]->size >= id)
                 res = erase(p->ch[0],id);
 90
 91
             else res = erase(p\rightarrow ch[1], id - p\rightarrow ch[0]\rightarrow size - p\rightarrow exist);
 92
             p->mp[res]--;
 93
             return res;
 94
         }
         //删除一定的点以后重构
 95
 96
         void check_erase(){
 97
             if(root->size < 0.5*root->nodeCount)
                 rebuild(root);
 98
 99
         }
100
         Node **insert(Node *&p,int id,int val){
101
             if(p == null){
                 p = newNode(val);
102
                 return &null;
103
             }
104
             else {
105
106
                 p->size++;
107
                 p->nodeCount++;
108
                 p->mp[val]++;
109
                 Node ** res;
                 if(id <= p->ch[0]->size+p->exist)
110
                      res = insert(p->ch[0],id,val);
111
                 else res = insert(p->ch[1],id-p->ch[0]->size-p->exist,
112
                    val);
113
                 if(p->isBad())res = &p;
114
                 return res;
115
             }
116
         //在第 id 个位置插入数 val
117
         void insert(int id,int val){
118
             Node **p = insert(root,id,val);
119
             if(*p != null)rebuild(*p);
120
         }
121
         //查询 [l,r] 之间值为 val 的数的个数
122
         int query(Node *p,int l,int r,int val){
123
             if(p == null)return 0;
124
             if(l <= 1 && p->size <= r)
125
                  return p->mp.count(val)?p->mp[val]:0;
126
127
             else {
128
                 int ans = 0;
```

```
129
                  if(l <= p->ch[0]->size)
130
                       ans += query(p->ch[0],l,r,val);
                  if(r > p->ch[0]->size+p->exist)
131
                       ans += query(p\rightarrow ch[1], l - p\rightarrow ch[0]\rightarrow size - p\rightarrow exist
132
                          , r - p \rightarrow ch[0] \rightarrow size - p \rightarrow exist, val);
                  if(p->exist && p->key == val && l <= p->ch[0]->size+1
133
                     && r >= p->ch[0]->size+1)
134
                       ans++;
135
                  return ans;
136
              }
137
         }
138
    }tree;
    int a[MAXN];
139
     int main()
140
141
     {
         int n;
142
         while(scanf("%d",&n) == 1){
143
144
              tree.init();
              for(int i = 0;i < n;i++)scanf("%d",&a[i]);</pre>
145
              tree.root = tree.buildTree(a,0,n);
146
              int m;
147
148
              int op,l,r,k;
149
              scanf("%d",&m);
              int ans = 0;
150
              while(m--){
151
                  scanf("%d",&op);
152
                  if(op == 1){
153
                       scanf("%d%d",&l,&r);
154
155
                       l = ((l+ans-1)%n)+1;
156
                       r = ((r+ans-1)%n)+1;
                       if(l > r)swap(l,r);
157
                       int v = tree.erase(tree.root,r);
158
                       //tree.check_erase(); //有时候可以加上删除重构
159
160
                       tree.insert(l,v);
161
                  }
162
                  else {
163
                       scanf("%d%d%d",&l,&r,&k);
                       l = ((l+ans-1)%n)+1;
164
                       r = ((r+ans-1)%n)+1;
165
                       k = ((k+ans-1)%n)+1;
166
                       if(l > r)swap(l,r);
167
                       ans = tree.query(tree.root,l,r,k);
168
                       printf("%d\n",ans);
169
                  }
170
171
              }
172
         }
173
         return 0;
174 |}
```

## 3.10 动态 KD 树

动态  $\mathrm{KD}$  树就是结合了  $\mathrm{KD}$  树和替罪羊树。支持  $\mathrm{KD}$  树的插入删除操作,用替罪羊树的思想来保存平衡。

UVALive6045

题意:给了二维平面上的 N 个整点( $N \leq 50000$ )。每次操作给了点  $(x_i, y_i)$ ,需要曼哈顿距离小于 E 的点进行一个变换。输出最后的点的坐标,保证变换次数不超过 50000.

```
const int MAXN = 100010;
   const double alpha = 0.75;
   struct Point{
        int x,y,id;
 5
   };
 6
   struct Node{
 7
        Point e;
 8
        int size,nodeCount;
 9
        Node *lc,*rc;
        bool div;
10
        bool exist;
11
        bool isBad(){
12
13
            return lc->nodeCount > alpha*nodeCount+5 || rc->nodeCount >
                alpha*nodeCount+5;
14
        inline void push_up(){
15
            size = exist + lc->size + rc->size;
16
17
            nodeCount = 1+lc->nodeCount+rc->nodeCount;
        }
18
19
   };
20
   Node pool[MAXN],*tail,*root,*null;
   Node *bc[MAXN];
   int bc_top;
22
23
   void init(){
        tail = pool;
24
        null = tail++;
25
        null->lc = null->rc = null;
26
        null->size = null->nodeCount = 0;
27
28
        root = null;
29
        bc_top = 0;
30
   Node *newNode(Point e){
31
32
       Node *p;
33
        if(bc_top)p = bc[--bc_top];
34
        else p = tail++;
        p->e = e;
35
        p\rightarrow lc = p\rightarrow rc = null;
36
        p->size = p->nodeCount = 1;
37
38
        p->exist = true;
39
        return p;
40
41
   inline bool cmpX(const Point &a,const Point &b){
        return a.x < b.x || (a.x == b.x && a.y < b.y) || (a.x == b.x &&
42
            a.y == b.y && a.id < b.id);
```

```
43
   }
44
   inline bool cmpY(const Point &a,const Point &b){
45
       return a.y < b.y || (a.y == b.y && a.x < b.x) || (a.y == b.y &&
           a.x == b.x && a.id < b.id);
46
47
   inline bool cmp(const Point &a,const Point &b,bool div){
48
       return div?cmpY(a,b):cmpX(a,b);
49
   //注意 a 需要备份, 否则就乱序
50
   Node *build(Point *a, int l, int r, bool div){
51
       if(l >= r)return null;
52
53
       int mid = (l+r)/2;
       nth_element(a+l,a+mid,a+r,div?cmpY:cmpX);
54
       Node *p = newNode(a[mid]);
55
       p->div = div;
56
       p->lc = build(a,l,mid,!div);
57
       p->rc = build(a,mid+1,r,!div);
58
59
       p->push_up();
60
       return p;
61
   void Travel(Node *p,vector<Point>&v){
62
63
       if(p == null)return;
64
       Travel(p->lc,v);
       if(p->exist)v.push_back(p->e);
65
66
       bc[bc_top++] = p;
67
       Travel(p->rc,v);
68
69
   Node *divide(vector<Point>&v, int l, int r, bool div){
70
       if(l >= r)return null;
71
       int mid = (l+r)/2;
       nth_element(v.begin()+l,v.begin()+mid,v.begin()+r,div?cmpY:cmpX
72
          );
       Node *p = newNode(v[mid]);
73
74
       p->div = div;
75
       p->lc = divide(v,l,mid,!div);
76
       p->rc = divide(v,mid+1,r,!div);
77
       p->push_up();
78
       return p;
79
80
   inline void rebuild(Node *&p){
81
       vector<Point>v;
82
       Travel(p,v);
83
       p = divide(v,0,v.size(),p->div);
84
85
   Node **insert(Node *&p,Point a,bool div){
       if(p == null){
86
87
            p = newNode(a);
            p->div = div;
88
89
            return &null;
90
       }
       else {
91
```

```
92
              p->nodeCount++;
 93
              p->size++;
              Node **res;
 94
 95
              if(cmp(a,p->e,div))
                  res = insert(p->lc,a,!div);
 96
 97
              else res = insert(p->rc,a,!div);
              if(p->isBad())res = &p;
 98
 99
              return res;
100
         }
101
102
    void insert(Point e){
         Node **p = insert(root,e,0);
103
         if(*p != null)rebuild(*p);
104
105
    vector<int>vec;
106
107
    void getvec(Node *p,int minx,int maxx,int miny,int maxy){
         if(p->size == 0)return;
108
109
         if(p->exist && minx <= p->e.x && p->e.x <= maxx && miny <= p->e
             .y && p->e.y <= maxy){
              vec.push_back(p->e.id);
110
              p\rightarrow exist = 0;
111
112
              p->size--;
113
         if(p\rightarrow div? miny <= p\rightarrow e.y : minx <= p\rightarrow e.x)getvec(p\rightarrow lc, minx,
114
            maxx,miny,maxy);
         if(p\rightarrow div? maxy >= p\rightarrow e.y : maxx >= p\rightarrow e.x)getvec(p\rightarrow rc, minx,
115
            maxx,miny,maxy);
         p->push_up();
116
117
118
    Point p[MAXN],p2[MAXN];
    Point p3[MAXN];
119
    int main()
120
121
122
         int T;
         scanf("%d",&T);
123
124
         int iCase = 0;
125
         int N,Q,W,H;
126
         while(T--){
127
              iCase++;
128
              scanf("%d%d%d%d",&N,&Q,&W,&H);
              init();
129
130
              for(int i = 0;i < N;i++){
                  scanf("%d%d",&p[i].x,&p[i].y);
131
                  p[i].id = p2[i].id = i;
132
133
                  p2[i].x = p[i].x+p[i].y;
                  p2[i].y = p[i].x-p[i].y;
134
135
                  p3[i] = p2[i];
              }
136
137
              root = build(p3,0,N,0);
138
              int X,Y,E,a,b,c,d,e,f;
139
              while(Q--){
```

```
scanf("%d%d%d%d%d%d%d%d%d",&X,&Y,&E,&a,&b,&c,&d,&e,&f);
140
141
                  vec.clear();
                  int minx = X+Y-E;
142
                  int maxx = X+Y+E;
143
                  int miny = X-Y-E;
144
                  int maxy = X-Y+E;
145
                  getvec(root, minx, maxx, miny, maxy);
146
                  int sz = vec.size();
147
148
                  for(int i = 0; i < sz; i++){
                      int id = vec[i];
149
150
                      long long tx = p[id].x;
                      long long ty = p[id].y;
151
                      p[id].x = (tx*a+ty*b+(long long)(id+1)*c)%W;
152
                      p[id].y = (tx*d+ty*e+(long long)(id+1)*f)%H;
153
                      p2[id].x = p[id].x+p[id].y;
154
                      p2[id].y = p[id].x-p[id].y;
155
                      insert(p2[id]);
156
                  }
157
158
             printf("Case<sub>□</sub>#%d:\n",iCase);
159
             for(int i = 0;i < N;i++)</pre>
160
161
                  printf("%d\lfloor%d\rfloorn",p[i].x,p[i].y);
162
         }
163
         return 0;
164 }
```

# 3.11 树套树

# 3.11.1 替罪羊树套 splay

BZOJ 3065: 带插入区间 K 小值

带插入、修改的区间 k 小值在线查询。

- 1. Q x y k: 询问从左至右第 x 只跳蚤到从左至右第 y 只跳蚤中,弹跳力第 k 小的跳蚤的弹跳力是多少。(1 <= x <= y <= m, 1 <= k <= y x + 1)
- 2. M x val: 将从左至右第 x 只跳蚤的弹跳力改为 val。 (1 <= x <= m)
- $3. \ I \times val:$  在从左至右第 x 只跳蚤的前面插入一只弹跳力为 val 的跳蚤。即插入后从左至右 第 x 只跳蚤是我刚插入的跳蚤。(1 <= x <= m+1)

```
const int MAXN = 70010;
 2
   namespace Splay{
 3
       struct Node *null;
 4
       struct Node{
 5
            Node *ch[2],*fa;
            int size,key,cnt;
 6
 7
            inline void setc(Node *p,int d){
 8
                ch[d] = p;
                p->fa = this;
 9
10
            inline bool d(){
11
12
                return fa->ch[1] == this;
13
            inline void push_up(){
14
```

```
size = ch[0]->size + ch[1]->size + cnt;
15
16
             }
             void clear(int _key){
17
                 size = cnt = 1;
18
                 key = _key;
19
                 ch[0] = ch[1] = fa = null;
20
21
             inline bool isroot(){
22
                 return fa == null || this != fa->ch[0] && this != fa->
23
                     ch[1];
24
             }
25
        };
26
        Node pool[MAXN*20],*tail;
        Node *bc[MAXN*20];
27
        int bc_top;//内存回收
28
        void init(){
29
             tail = pool;
30
31
             bc_top = 0;
             null = tail++;
32
             null->size = null->cnt = 0;
33
             null->ch[0] = null->ch[1] = null->fa = null;
34
35
36
        inline void rotate(Node *x){
             Node *f = x \rightarrow fa, *ff = x \rightarrow fa \rightarrow fa;
37
             int c = x->d(), cc = f->d();
38
             f->setc(x->ch[!c],c);
39
             x \rightarrow setc(f,!c);
40
41
             if(ff->ch[cc] == f)ff->setc(x,cc);
42
             else x->fa = ff;
43
             f->push_up();
44
        inline void splay(Node* &root, Node* x, Node* goal){
45
             while(x->fa != goal){
46
47
                 if(x->fa->fa == goal)rotate(x);
                 else {
48
49
                      bool f = x \rightarrow fa \rightarrow d();
50
                      x\rightarrow d() == f ? rotate(x\rightarrow fa) : rotate(x);
51
                      rotate(x);
52
                 }
53
             }
54
             x->push_up();
             if(goal == null)root = x;
55
56
        }
57
        //找到 r 子树里面的最左边那个
        Node* get_left(Node* r){
58
59
             Node* x = r;
             while (x\rightarrow ch[0] != null)x = x\rightarrow ch[0];
60
61
             return x;
62
        }
63
        //在 root 的树中删掉 x
        void erase(Node* &root,Node* x){
64
```

```
splay(root,x,null);
 65
 66
             Node* t = root;
             if(t->ch[1] != null){
 67
                 root = t->ch[1];
 68
                 splay(root,get_left(t->ch[1]),null);
 69
                 root->setc(t->ch[0],0);
 70
             }
 71
 72
             else root = root->ch[0];
 73
             bc[bc\_top++] = x;
             root->fa = null;
 74
 75
             if(root != null)root->push_up();
 76
        }
 77
        Node* newNode(int key){
             Node* p;
 78
             if(bc_top)p = bc[--bc_top];
 79
 80
             else p = tail++;
 81
             p->clear(key);
 82
             return p;
 83
        }
        //插入一个值 key
 84
        void insert(Node* &root,int key){
 85
 86
             if(root == null){
 87
                 root = newNode(key);
 88
                 return;
 89
             }
             Node* now = root;
 90
             Node* pre = root->fa;
 91
             while(now != null){
 92
 93
                 if(now->key == key){
 94
                     now->cnt++;
 95
                      splay(root,now,null);
 96
                     return;
                 }
 97
 98
                 pre = now;
 99
                 now = now->ch[key >= now->key];
100
             }
101
             Node *x = newNode(key);
             pre->setc(x,key >= pre->key);
102
             splay(root,x,null);
103
104
        }
         //删除一个值 key
105
        void erase(Node* &root,int key){
106
             Node* now = root;
107
             while(now->key != key){
108
                 now = now->ch[key >= now->key];
109
             }
110
             now->cnt--;
111
             if(now->cnt == 0)erase(root,now);
112
113
             else splay(root, now, null);
114
115
        void Travel(Node* r){
```

```
if(r == null)return;
116
             Travel(r->ch[0]);
117
             bc[bc_top++] = r;
118
             Travel(r->ch[1]);
119
         }
120
         void CLEAR(Node* &root){
121
122
             Travel(root);
123
             root = null;
124
         }
         //查询小于等于 val 的个数
125
         int query(Node *root,int val){
126
             int ans = 0;
127
             Node *x = root;
128
             while(x != null){
129
                 if(val < x->key)x = x->ch[0];
130
131
                 else{
132
                      ans += x->ch[0]->size + x->cnt;
                      x = x - ch[1];
133
134
                 }
             }
135
136
             return ans;
         }
137
138
    };
    namespace ScapeGoatTree{
139
140
         const double alpha = 0.75;
         struct Node{
141
142
             Node *ch[2];
             int size,nodeCount,key;
143
144
             Splay::Node *root;
145
             bool isBad(){
                 return ch[0]->nodeCount > alpha*nodeCount+5 || ch[1]->
146
                    nodeCount > alpha*nodeCount+5;
147
             void push_up(){
148
                 size = 1+ch[0]->size+ch[1]->size;
149
150
                 nodeCount = 1+ch[0]->nodeCount+ch[1]->nodeCount;
151
             }
152
         };
         Node pool[MAXN];
153
154
         Node *tail,*root,*null;
         Node *bc[MAXN];
155
         int bc_top;
156
         void init(){
157
             tail = pool;
158
159
             null = tail++;
             null \rightarrow ch[0] = null \rightarrow ch[1] = null;
160
             null->size = null->nodeCount = 0;
161
             null->root = Splay::null;
162
163
             bc_top = 0;
164
         inline Node* newNode(int key){
165
```

```
Node *p;
166
167
             if(bc_top)p = bc[--bc_top];
             else p = tail++;
168
             p\rightarrow key = key;
169
             p->ch[0] = p->ch[1] = null;
170
             p->size = p->nodeCount = 1;
171
             p->root = Splay::null;
172
173
             return p;
174
        }
        Node *buildTree(int *a,int l,int r){
175
             if(l >= r)return null;
176
             int mid = (l+r)/2;
177
             Node *p = newNode(a[mid]);
178
             for(int i = l;i < r;i++)
179
                 Splay::insert(p->root,a[i]);
180
181
             p->ch[0] = buildTree(a,l,mid);
             p->ch[1] = buildTree(a,mid+1,r);
182
             p->push_up();
183
184
             return p;
        }
185
        void Travel(Node *p,vector<int>&v){
186
187
             if(p == null)return;
188
             Travel(p->ch[0],v);
             v.push_back(p->key);
189
             Splay::CLEAR(p->root);
190
             bc[bc_top++] = p;
191
             Travel(p->ch[1],v);
192
193
194
        Node *divide(vector<int>&v,int l,int r){
195
             if(l == r)return null;
             int mid = (l+r)/2;
196
             Node *p = newNode(v[mid]);
197
             for(int i = l;i < r;i++)
198
                 Splay::insert(p->root,v[i]);
199
             p->ch[0] = divide(v,l,mid);
200
201
             p->ch[1] = divide(v,mid+1,r);
202
             p->push_up();
             return p;
203
204
205
        inline void rebuild(Node *&p){
             vector<int>v;
206
207
             Travel(p,v);
             p = divide(v,0,v.size());
208
        }
209
         //将第 id 个值修改为 val
210
        int Modify(Node *p,int id,int val){
211
             if(id == p->ch[0]->size+1){
212
                 int v = p->key;
213
                 Splay::erase(p->root,v);
214
215
                 Splay::insert(p->root,val);
216
                 p->key = val;
```

```
217
                 return v;
218
             }
             int res;
219
             if(p->ch[0]->size >= id)
220
                 res = Modify(p->ch[0],id,val);
221
             else res = Modify(p->ch[1],id-p->ch[0]->size-1,val);
222
             Splay::erase(p->root,res);
223
             Splay::insert(p->root,val);
224
225
             return res;
226
        }
        Node **insert(Node *&p,int id,int val){
227
             if(p == null){
228
                 p = newNode(val);
229
                 Splay::insert(p->root,val);
230
                 return &null;
231
232
             }
             else {
233
234
                 p->size++;
235
                 p->nodeCount++;
                 Splay::insert(p->root,val);
236
                 Node ** res;
237
                 if(id <= p->ch[0]->size+1)
238
239
                     res = insert(p->ch[0],id,val);
                 else res = insert(p->ch[1],id-p->ch[0]->size-1,val);
240
                 if(p->isBad())res = &p;
241
242
                 return res;
             }
243
244
245
        void insert(int id,int val){
246
             Node **p = insert(root,id,val);
             if(*p != null)rebuild(*p);
247
248
        }
         //查询在 [l,r] 区间,值小于等于 val 的个数
249
250
        int query(Node *p,int l,int r,int val){
             if(p == null)return 0;
251
252
             if(l <= 1 && p->size <= r)
253
                 return Splay::query(p->root,val);
254
             else {
                 int ans = 0;
255
256
                 if(l <= p->ch[0]->size)
                     ans += query(p->ch[0],l,r,val);
257
                 if(r > p->ch[0]->size+1)
258
                     ans += query(p->ch[1], l-p->ch[0]->size-1, r-p->ch
259
                        [0]->size-1,val);
                 if(p->key <= val && l <= p->ch[0]->size+1 && p->ch[0]->
260
                    size+1 <= r)
261
                     ans++;
262
                 return ans;
             }
263
264
        int query(int L,int R,int k){
265
```

```
int ans;
266
             int l = 0, r = 100000;
267
             while(l <= r){
268
                 int mid = (l+r)/2;
269
                 if(query(root,L,R,mid) >= k){
270
                      ans = mid;
271
272
                      r = mid-1;
273
                 }
274
                 else l = mid+1;
275
276
             return ans;
277
         }
278
    };
    int a[MAXN];
279
    int main()
280
281
    {
         int n;
282
283
         while(scanf("%d",&n) == 1){
             Splay::init();
284
             ScapeGoatTree::init();
285
             for(int i = 0;i < n;i++)scanf("%d",&a[i]);</pre>
286
287
             ScapeGoatTree::root = ScapeGoatTree::buildTree(a,0,n);
288
             int m;
             char op[10];
289
             scanf("%d",&m);
290
             int ans = 0;
291
             while(m——){
292
                 scanf("%s",op);
293
294
                 if(op[0] == 'Q'){
295
                      int x,y,k;
                      scanf("%d%d%d",&x,&y,&k);
296
                      x ^= ans; y ^= ans; k ^= ans;
297
298
                      ans = ScapeGoatTree::query(x,y,k);
299
                      printf("%d\n",ans);
300
301
                 else if(op[0] == 'M'){
302
                      int x,val;
                      scanf("%d%d",&x,&val);
303
                      x ^= ans; val ^= ans;
304
305
                      ScapeGoatTree::Modify(ScapeGoatTree::root,x,val);
306
                 }
                 else if(op[0] == 'I'){
307
                      int x, val;
308
                      scanf("%d%d",&x,&val);
309
                      x ^= ans; val ^= ans;
310
                      ScapeGoatTree::insert(x,val);
311
                 }
312
             }
313
314
         }
315
         return 0;
316 |}
```

# 4 图论

# 4.1 最短路

## 4.1.1 Dijkstra 单源最短路

权值必须是非负

```
/*
 1
    * 单源最短路径, Dijkstra 算法, 邻接矩阵形式, 复杂度为O(n^2)
 2
    * 求出源 beg 到所有点的最短路径,传入图的顶点数,和邻接矩阵 cost[][]
 3
    * 返回各点的最短路径 lowcost[], 路径 pre[].pre[i] 记录 beg 到 i 路径上的
       父结点, pre[beg]=-1
    * 可更改路径权类型,但是权值必须为非负
 5
 6
    */
   const int MAXN=1010;
 7
   #define typec int
   const typec INF=0x3f3f3f3f;//防止后面溢出,这个不能太大
   bool vis[MAXN];
   int pre[MAXN];
11
12
   void Dijkstra(typec cost[][MAXN],typec lowcost[],int n,int beg){
13
       for(int i=0;i<n;i++){</pre>
14
           lowcost[i]=INF;vis[i]=false;pre[i]=-1;
15
       lowcost[beg]=0;
16
       for(int j=0;j<n;j++){</pre>
17
           int k=-1;
18
19
           int Min=INF;
20
           for(int i=0;i<n;i++)</pre>
21
               if(!vis[i]&&lowcost[i]<Min){</pre>
                   Min=lowcost[i];
22
23
                   k=i;
24
               }
           if(k==-1)break;
25
           vis[k]=true;
26
27
           for(int i=0;i<n;i++)</pre>
               if(!vis[i]&&lowcost[k]+cost[k][i]<lowcost[i]){</pre>
28
                   lowcost[i]=lowcost[k]+cost[k][i];
29
                   pre[i]=k;
30
31
               }
32
       }
33 |}
   4.1.2 Dijkstra 算法 + 堆优化
   使用优先队列优化,复杂度 O (E log E)
 1
 2
    * 使用优先队列优化 Dijkstra 算法
    * 复杂度 O(ElogE)
```

kuangbin 130

\* 注意对 vector<Edge>E[MAXN] 进行初始化后加边

const int INF=0x3f3f3f3f;

5

\*/

```
const int MAXN=1000010;
 8
   struct qnode{
 9
       int v;
10
       int c;
       qnode(int _v=0,int _c=0):v(_v),c(_c){}
11
12
       bool operator <(const gnode &r)const{</pre>
13
           return c>r.c;
       }
14
15
   };
   struct Edge{
16
17
       int v,cost;
       Edge(int _v=0,int _cost=0):v(_v),cost(_cost){}
18
19
   };
20
   vector<Edge>E[MAXN];
21 |bool vis[MAXN];
22
   |int dist[MAXN];
   //点的编号从 1 开始
23
   void Dijkstra(int n,int start){
24
25
       memset(vis, false, sizeof(vis));
       for(int i=1;i<=n;i++)dist[i]=INF;</pre>
26
       priority_queue<qnode>que;
27
28
       while(!que.empty())que.pop();
29
       dist[start]=0;
       que.push(qnode(start,0));
30
       anode tmp:
31
       while(!que.empty()){
32
33
           tmp=que.top();
34
           que.pop();
35
           int u=tmp.v;
36
           if(vis[u])continue;
37
           vis[u]=true;
           for(int i=0;i<E[u].size();i++){</pre>
38
39
                int v=E[tmp.v][i].v;
                int cost=E[u][i].cost;
40
41
                if(!vis[v]&&dist[v]>dist[u]+cost){
42
                    dist[v]=dist[u]+cost;
43
                    que.push(qnode(v,dist[v]));
44
                }
45
           }
       }
46
47
48
   void addedge(int u,int v,int w){
49
       E[u].push_back(Edge(v,w));
50
   }
   4.1.3 单源最短路 bellman ford 算法
   /*
 1
 2
    * 单源最短路 bellman ford 算法,复杂度 O(VE)
 3
    * 可以处理负边权图。
      可以判断是否存在负环回路。返回 true, 当且仅当图中不包含从源点可达的负权回路
```

```
* vector<Edge>E; 先 E.clear() 初始化, 然后加入所有边
 5
 6
    * 点的编号从 1 开始 (从 0 开始简单修改就可以了)
 7
   */
   const int INF=0x3f3f3f3f;
 8
   const int MAXN=550;
9
  int dist[MAXN];
11
   struct Edge{
12
       int u, v;
13
       int cost;
       Edge(int u=0, int v=0, int cost=0):u(u),v(v),cost(cost){}
14
15
   };
   vector<Edge>E;
16
   //点的编号从 1 开始
17
   bool bellman_ford(int start,int n){
18
       for(int i=1;i<=n;i++)dist[i]=INF;</pre>
19
       dist[start]=0;
20
       //最多做 n-1 次
21
       for(int i=1;i<n;i++){</pre>
22
           bool flag=false;
23
           for(int j=0;j<E.size();j++){</pre>
24
               int u=E[j].u;
25
26
               int v=E[j].v;
27
               int cost=E[j].cost;
               if(dist[v]>dist[u]+cost){
28
29
                   dist[v]=dist[u]+cost;
30
                   flag=true;
31
               }
32
33
           if(!flag)return true;//没有负环回路
34
       for(int j=0;j<E.size();j++)</pre>
35
           if(dist[E[j].v]>dist[E[j].u]+E[j].cost)
36
37
               return false;//有负环回路
38
       return true; //没有负环回路
39 |}
   4.1.4 单源最短路 SPFA
  /*
 1
 2
    * 单源最短路 SPFA
 3
    * 时间复杂度 0(kE)
   * 这个是队列实现,有时候改成栈实现会更加快,很容易修改
    * 这个复杂度是不定的
 5
 7
   const int MAXN=1010;
   const int INF=0x3f3f3f3f;
 9
   struct Edge{
10
       int ∨;
11
       int cost;
12
       Edge(int _v=0,int _cost=0):v(_v),cost(_cost){}
13
   };
14 | vector<Edge>E[MAXN];
```

```
void addedge(int u,int v,int w){
16
       E[u].push_back(Edge(v,w));
17
   bool vis[MAXN];//在队列标志
18
   |int cnt[MAXN];//每个点的入队列次数
19
20
   int dist[MAXN];
21
   bool SPFA(int start,int n){
22
       memset(vis, false, sizeof(vis));
23
       for(int i=1;i<=n;i++)dist[i]=INF;</pre>
24
       vis[start]=true;
       dist[start]=0;
25
26
       queue<int>que;
       while(!que.empty())que.pop();
27
       que.push(start);
28
       memset(cnt,0,sizeof(cnt));
29
30
       cnt[start]=1;
       while(!que.empty()){
31
           int u=que.front();
32
33
           que.pop();
           vis[u]=false;
34
           for(int i=0;i<E[u].size();i++){</pre>
35
36
                int v=E[u][i].v;
37
               if(dist[v]>dist[u]+E[u][i].cost){
                    dist[v]=dist[u]+E[u][i].cost;
38
39
                    if(!vis[v]){
40
                        vis[v]=true;
41
                        que.push(v);
42
                        if(++cnt[v]>n)return false;
43
                        //cnt[i] 为入队列次数,用来判定是否存在负环回路
44
                    }
45
               }
46
           }
47
       }
48
       return true;
49 |}
   4.2
        最小生成树
   4.2.1 Prim 算法
 1 /*
 2
    * Prim 求 MST
    * 耗费矩阵 cost[][], 标号从 0 开始, 0~n-1
 3
    * 返回最小生成树的权值,返回 -1 表示原图不连通
 4
 5
    */
   const int INF=0x3f3f3f3f;
 7
   const int MAXN=110;
 8 | bool vis[MAXN];
   int lowc[MAXN];
 9
   |//点是 0 n-1
10
   int Prim(int cost[][MAXN],int n){
11
12
       int ans=0;
```

```
13
       memset(vis, false, sizeof(vis));
14
       vis[0]=true;
15
       for(int i=1;i<n;i++)lowc[i]=cost[0][i];</pre>
       for(int i=1;i<n;i++){</pre>
16
           int minc=INF;
17
           int p=-1;
18
           for(int j=0;j<n;j++)</pre>
19
                if(!vis[j]&&minc>lowc[j]){
20
21
                    minc=lowc[j];
22
                    p=j;
23
24
           if(minc==INF)return -1;//原图不连通
25
           ans+=minc;
26
           vis[p]=true;
           for(int j=0;j<n;j++)
27
28
                if(!vis[j]&&lowc[j]>cost[p][j])
29
                    lowc[j]=cost[p][j];
30
       }
31
       return ans;
32
   }
   4.2.2 Kruskal 算法
 1
  /*
 2
    * Kruskal 算法求 MST
 3
    */
   |const int MAXN=110;//最大点数
   const int MAXM=10000;//最大边数
   |int F[MAXN];//并查集使用
   struct Edge{
 7
 8
       int u, v, w;
   |}edge[MAXM];//存储边的信息,包括起点/终点/权值
   int tol;//边数,加边前赋值为 0
10
   void addedge(int u,int v,int w){
11
12
       edge[tol].u=u;
13
       edge[tol].v=v;
       edge[tol++].w=w;
14
15
   //排序函数,讲边按照权值从小到大排序
16
17
   bool cmp(Edge a, Edge b) {
       return a.w<b.w;</pre>
18
19
   int find(int x){
20
       if(F[x] == -1)return x;
21
22
       else return F[x]=find(F[x]);
23
   }
   //传入点数,返回最小生成树的权值,如果不连通返回 -1
24
   int Kruskal(int n){
25
26
       memset(F,-1,sizeof(F));
27
       sort(edge,edge+tol,cmp);
       int cnt=0;//计算加入的边数
28
```

29

int ans=0;

```
30
       for(int i=0;i<tol;i++){</pre>
           int u=edge[i].u;
31
           int v=edge[i].v;
32
           int w=edge[i].w;
33
           int t1=find(u);
34
           int t2=find(v);
35
           if(t1!=t2){
36
37
                ans+=w;
38
                F[t1]=t2;
39
                cnt++;
40
41
           if(cnt==n-1)break;
42
43
       if(cnt<n-1) return -1; //不连通
44
       else return ans;
45 |}
       次小生成树
   4.3
 1
   /*
 2
    * 次小生成树
 3
    * 求最小生成树时, 用数组 Max[i][j] 来表示 MST 中 i 到 j 最大边权
    * 求完后,直接枚举所有不在 MST 中的边,替换掉最大边权的边,更新答案
 5
    * 点的编号从 0 开始
 6
    */
   const int MAXN=110;
   const int INF=0x3f3f3f3f;
9 bool vis[MAXN];
   |int lowc[MAXN];
   int pre[MAXN];
11
12 | int Max[MAXN][MAXN];//Max[i][j] 表示在最小生成树中从 i 到 j 的路径中的最
      大边权
13 | bool used[MAXN][MAXN];
   int Prim(int cost[][MAXN],int n){
14
15
       int ans=0;
       memset(vis, false, sizeof(vis));
16
       memset(Max,0,sizeof(Max));
17
       memset(used, false, sizeof(used));
18
       vis[0]=true;
19
20
       pre[0]=-1;
       for(int i=1;i<n;i++){</pre>
21
22
           lowc[i]=cost[0][i];
           pre[i]=0;
23
24
25
       lowc[0]=0;
       for(int i=1;i<n;i++){</pre>
26
           int minc=INF;
27
28
           int p=-1;
           for(int j=0;j<n;j++)</pre>
29
                if(!vis[j]&&minc>lowc[j]){
30
31
                    minc=lowc[j];
32
                    p=j;
```

```
33
                }
34
            if(minc==INF)return -1;
            ans+=minc;
35
36
            vis[p]=true;
            used[p][pre[p]]=used[pre[p]][p]=true;
37
            for(int j=0;j<n;j++){</pre>
38
                if(vis[j] && j != p)Max[j][p]=Max[p][j]=max(Max[j][pre[
39
                   p]],lowc[p]);
40
                if(!vis[j]&&lowc[j]>cost[p][j]){
41
                    lowc[j]=cost[p][j];
42
                    pre[j]=p;
43
                }
44
            }
45
46
       return ans;
47 }
   4.4
        有向图的强连通分量
   4.4.1 Tarjan
 1 | / *
 2
    * Tarjan 算法
 3
    * 复杂度 O(N+M)
 4
    */
 5
   |const int MAXN = 20010;//点数
   |const int MAXM = 50010;//边数
 7
   struct Edge{
 8
       int to,next;
 9
   }edge[MAXM];
   int head[MAXN],tot;
11 | int Low[MAXN], DFN[MAXN], Stack[MAXN], Belong[MAXN]; //Belong 数组的值是
      1 \sim \text{scc}
12 | int Index, top;
   int scc;//强连通分量的个数
13
14 bool Instack[MAXN];
   | {f int}| | {f num[MAXN];//各个强连通分量包含点的个数,数组编号 1 <math>\sim | {f scc}|
15
   //num 数组不一定需要,结合实际情况
16
17
   void addedge(int u,int v){
18
       edge[tot].to = v;edge[tot].next = head[u];head[u] = tot++;
19
20
   void Tarjan(int u){
21
22
       int ∨;
23
       Low[u] = DFN[u] = ++Index;
24
       Stack[top++] = u;
25
       Instack[u] = true;
26
        for(int i = head[u]; i != -1; i = edge[i].next){
27
            v = edge[i].to;
            if(!DFN[v]){
28
29
                Tarjan(v);
                if( Low[u] > Low[v] )Low[u] = Low[v];
30
```

```
31
           }
32
           else if(Instack[v] && Low[u] > DFN[v])
33
               Low[u] = DFN[\vee];
34
35
       if(Low[u] == DFN[u]){
           scc++;
36
           do{
37
38
               v = Stack[--top];
39
               Instack[v] = false;
40
               Belong[v] = scc;
41
               num[scc]++;
42
43
           while( v != u);
44
       }
45
   }
46
   void solve(int N){
47
       memset(DFN,0,sizeof(DFN));
       memset(Instack, false, sizeof(Instack));
48
49
       memset(num,0,sizeof(num));
       Index = scc = top = 0;
50
       for(int i = 1;i <= N;i++)</pre>
51
52
           if(!DFN[i])
53
               Tarjan(i);
54
55
   void init(){
56
       tot = 0;
57
       memset(head, -1, sizeof(head));
58 | }
   4.4.2 Kosaraju
 1
 2
   * Kosaraju 算法, 复杂度 O(N+M)
 3
   */
   const int MAXN = 20010;
   const int MAXM = 50010;
  struct Edge{
 7
       int to,next;
  }edge1[MAXM],edge2[MAXM];
 8
   //edge1 是原图 G, edge2 是逆图 GT
  int head1[MAXN],head2[MAXN];
11 | bool mark1[MAXN], mark2[MAXN];
  int tot1,tot2;
   int cnt1,cnt2;
   15 | int Belong[MAXN];//每个点属于哪个连通分量 (0~cnt2-1)
   |int num;//中间变量,用来数某个连通分量中点的个数
   int setNum[MAXN];//强连通分量中点的个数,编号 0~cnt2-1
17
   void addedge(int u,int v){
18
19
       edge1[tot1].to = v;edge1[tot1].next = head1[u];head1[u] = tot1
          ++;
```

```
20
       edge2[tot2].to = u;edge2[tot2].next = head2[v];head2[v] = tot2
          ++;
21
   void DFS1(int u){
22
23
       mark1[u] = true;
       for(int i = head1[u];i != -1;i = edge1[i].next)
24
25
            if(!mark1[edge1[i].to])
                DFS1(edge1[i].to);
26
27
       st[cnt1++] = u;
28
   void DFS2(int u){
29
       mark2[u] = true;
30
31
       num++;
32
       Belong[u] = cnt2;
       for(int i = head2[u];i != -1;i = edge2[i].next)
33
            if(!mark2[edge2[i].to])
34
                DFS2(edge2[i].to);
35
36
   //点的编号从 1 开始
37
   void solve(int n){
38
       memset(mark1, false, sizeof(mark1));
39
       memset(mark2, false, sizeof(mark2));
40
       cnt1 = cnt2 = 0;
41
       for(int i = 1;i <= n;i++)
42
            if(!mark1[i])
43
44
                DFS1(i);
       for(int i = cnt1-1;i >= 0; i---)
45
            if(!mark2[st[i]]){
46
                num = 0;
47
48
                DFS2(st[i]);
49
                setNum[cnt2++] = num;
50
            }
51 |}
```

# 4.5 图的割点、桥和双连通分支的基本概念

[点连通度与边连通度] 在一个无向连通图中,如果有一个顶点集合,删除这个顶点集合,以及这个集合中所有顶点相关联的边以后,原图变成多个连通块,就称这个点集为割点集合。 一个图的点连通度的定义为,最小割点集合中的顶点数。

类似的,如果有一个边集合,删除这个边集合以后,原图变成多个连通块,就称这个点集为 割边集合。一个图的边连通度的定义为,最小割边集合中的边数。

## [双连通图、割点与桥]

如果一个无向连通图的点连通度大于 1,则称该图是点双连通的 (point biconnected),简称双连通或重连通。一个图有割点,当且仅当这个图的点连通度为 1,则割点集合的唯一元素被称为割点 (cut point),又叫关节点 (articulation point)。

如果一个无向连通图的边连通度大于  $\hat{1}$ ,则称该图是边双连通的 (edge biconnected),简称双连通或重连通。一个图有桥,当且仅当这个图的边连通度为  $\hat{1}$ ,则割边集合的唯一元素被称为桥 (bridge),又叫关节边 (articulation edge)。

可以看出,点双连通与边双连通都可以简称为双连通,它们之间是有着某种联系的,下文中提到的双连通,均既可指点双连通,又可指边双连通。

#### [双连通分支]

在图 G 的所有子图 G' 中,如果 G' 是双连通的,则称 G' 为双连通子图。如果一个双连

通子图 G' 它不是任何一个双连通子图的真子集,则 G' 为极大双连通子图。双连通分支 (biconnected component),或重连通分支,就是图的极大双连通子图。特殊的,点双连通分支又叫做块。[求割点与桥]

该算法是 R. Tarjan 发明的。对图深度优先搜索,定义 DFS(u) 为 u 在搜索树(以下简称为树)中被遍历到的次序号。定义 Low(u) 为 u 或 u 的子树中能通过非父子边追溯到的最早的节点,即 DFS 序号最小的节点。根据定义,则有:

 $Low(u)=Min\ DFS(u)\ DFS(v)\ (u,v)$  为后向边 (返祖边) 等价于 DFS(v)<DFS(u) 且 v 不为 u 的父亲节点  $Low(v)\ (u,v)$  为树枝边 (父子边) 一个顶点 u 是割点,当且仅当满足 (1) 或 (2) (1) u 为树根,且 u 有多于一个子树。(2) u 不为树根,且满足存在 (u,v) 为树枝边 (或称父子边,即 u 为 v 在搜索树中的父亲),使得 DFS(u)<=Low(v)。

一条无向边 (u,v) 是桥,当且仅当 (u,v) 为树枝边,且满足 DFS(u) < Low(v)。

## [求双连通分支]

下面要分开讨论点双连通分支与边双连通分支的求法。

对于点双连通分支,实际上在求割点的过程中就能顺便把每个点双连通分支求出。建立一个栈,存储当前双连通分支,在搜索图时,每找到一条树枝边或后向边 (非横叉边),就把这条边加入栈中。如果遇到某时满足 DFS(u) <= Low(v),说明 u 是一个割点,同时把边从栈顶一个个取出,直到遇到了边 (u,v),取出的这些边与其关联的点,组成一个点双连通分支。割点可以属于多个点双连通分支,其余点和每条边只属于且属于一个点双连通分支。

对于边双连通分支,求法更为简单。只需在求出所有的桥以后,把桥边删除,原图变成了多个连通块,则每个连通块就是一个边双连通分支。桥不属于任何一个边双连通分支,其余的边和每个顶点都属于且只属于一个边双连通分支。

### [构造双连通图]

一个有桥的连通图,如何把它通过加边变成边双连通图?方法为首先求出所有的桥,然后删除这些桥边,剩下的每个连通块都是一个双连通子图。把每个双连通子图收缩为一个顶点,再把桥边加回来,最后的这个图一定是一棵树,边连通度为 $\,1
m_{o}$ 。

统计出树中度为 1 的节点的个数,即为叶节点的个数,记为 leaf。则至少在树上添加 (leaf+1)/2 条边,就能使树达到边二连通,所以至少添加的边数就是 (leaf+1)/2。具体方法为,首先把两个最近公共祖先最远的两个叶节点之间连接一条边,这样可以把这两个点到祖先的路径上所有点收缩到一起,因为一个形成的环一定是双连通的。然后再找两个最近公共祖先最远的两个叶节点,这样一对一对找完,恰好是 (leaf+1)/2 次,把所有点收缩到了一起。

# 4.6 割点与桥

### 4.6.1 模板

```
/*
1
2
     求无向图的割点和桥
     可以找出割点和桥, 求删掉每个点后增加的连通块。
     需要注意重边的处理,可以先用矩阵存,再转邻接表,或者进行判重
5
  */
  const int MAXN = 10010;
7
  const int MAXM = 100010;
  struct Edge{
8
9
      int to,next;
      bool cut; //是否为桥的标记
10
  }edge[MAXM];
  int head[MAXN],tot;
  int Low[MAXN],DFN[MAXN],Stack[MAXN];
13
14
  int Index,top;
  bool Instack[MAXN];
15
```

```
16 bool cut[MAXN];
17
   int add_block[MAXN];//删除一个点后增加的连通块
18
   int bridge;
   void addedge(int u,int v){
19
       edge[tot].to = v;edge[tot].next = head[u];edge[tot].cut = false
20
21
       head[u] = tot++;
22
23
   void Tarjan(int u,int pre){
24
       int v;
25
       Low[u] = DFN[u] = ++Index;
26
       Stack[top++] = u;
       Instack[u] = true;
27
       int son = 0;
28
       int pre_cnt = 0; //处理重边,如果不需要可以去掉
29
       for(int i = head[u]; i != -1; i = edge[i].next){
30
           v = edge[i].to;
31
32
           if(v == pre && pre_cnt == 0){pre_cnt++;continue;}
33
           if( !DFN[∨] ){
34
               son++;
               Tarjan(v,u);
35
36
               if(Low[u] > Low[v])Low[u] = Low[v];
37
               //一条无向边 (u,v) 是桥,当且仅当 (u,v) 为树枝边,且满足
38
                 DFS(u) < Low(v).
39
               if(Low[v] > DFN[u]){
40
                   bridge++;
41
                   edge[i].cut = true;
42
                   edge[i^1].cut = true;
43
               }
               //割点
44
               //一个顶点 u 是割点, 当且仅当满足(1)或(2)(1) u 为树根, 且
45
                 u 有多于一个子树。
46
               //(2) u 不为树根,且满足存在 (u,v) 为树枝边 (或称父子边,
47
               //即 u 为 v 在搜索树中的父亲), 使得 DFS(u)<=Low(v)
48
               if(u != pre && Low[v] >= DFN[u]){//不是树根
49
                   cut[u] = true;
                   add block[u]++;
50
51
               }
52
           }
53
           else if( Low[u] > DFN[v])
                Low[u] = DFN[v];
54
55
       }
       //树根,分支数大于 1
56
       if(u == pre && son > 1)cut[u] = true;
57
       if(u == pre)add_block[u] = son - 1;
58
       Instack[u] = false;
59
60
       top--;
61 |}
```

#### 4.6.2 调用

1) UVA 796 Critical Links 给出一个无向图,按顺序输出桥

```
void solve(int N){
 1
 2
        memset(DFN,0,sizeof(DFN));
 3
        memset(Instack, false, sizeof(Instack));
 4
        memset(add_block,0,sizeof(add_block));
        memset(cut, false, sizeof(cut));
 5
 6
        Index = top = 0;
 7
        bridge = 0;
        for(int i = 1;i <= N;i++)</pre>
 8
 9
            if(!DFN[i])
10
                Tarjan(i,i);
11
        printf("%d_critical_links\n",bridge);
12
        vector<pair<int,int> >ans;
        for(int u = 1;u <= N;u++)</pre>
13
            for(int i = head[u];i != -1;i = edge[i].next)
14
15
                if(edge[i].cut && edge[i].to > u)
16
                {
17
                     ans.push_back(make_pair(u,edge[i].to));
18
        sort(ans.begin(),ans.end());
19
20
        //按顺序输出桥
        for(int i = 0;i < ans.size();i++)</pre>
21
22
            printf("%d<sub>L</sub>-__%d\n",ans[i].first-1,ans[i].second-1);
23
        printf("\n");
24
   }
25
   void init(){
26
        tot = 0;
27
        memset(head,-1,sizeof(head));
28
   }
29
   //处理重边
   map<int,int>mapit;
30
   inline bool isHash(int u,int v){
31
32
        if(mapit[u*MAXN+v])return true;
        if(mapit[v*MAXN+u])return true;
33
        mapit[u*MAXN+v] = mapit[v*MAXN+u] = 1;
34
35
        return false;
36
   int main(){
37
38
        int n;
39
        while(scanf("%d",&n) == 1){
            init();
40
            int u;
41
42
            int k;
43
            int ∨;
44
            //mapit.clear();
45
            for(int i = 1;i <= n;i++){
46
                scanf("%d<sub>□</sub>(%d)",&u,&k);
47
                u++;
                //这样加边,要保证正边和反边是相邻的,建无向图
48
```

```
while(k--){
49
50
                     scanf("%d",&v);
51
                     v++;
52
                     if(v <= u)continue;</pre>
                     //if(isHash(u,v))continue;
53
54
                     addedge(u,v);
55
                     addedge(v,u);
                }
56
57
            }
58
            solve(n);
59
        }
60
        return 0;
61
   }
   2) POJ 2117 求删除一个点后,图中最多有多少个连通块
   void solve(int N){
 1
 2
        memset(DFN,0,sizeof(DFN));
 3
        memset(Instack,0,sizeof(Instack));
        memset(add_block,0,sizeof(add_block));
 4
 5
        memset(cut, false, sizeof(cut));
 6
        Index = top = 0;
 7
        int cnt = 0; / /原来的连通块数
        for(int i = 1;i <= N;i++)</pre>
 8
 9
           if(!DFN[i]){
10
               Tarjan(i,i);//找割点调用必须是 Tarjan(i,i)
11
               cnt++;
12
           }
13
        int ans = 0;
        for(int i = 1;i <= N;i++)</pre>
14
           ans = max(ans,cnt+add_block[i]);
15
16
        printf("%d\n",ans);
17
18
   void init(){
19
        tot = 0;
20
        memset(head,-1,sizeof(head));
21
   int main(){
22
23
        int n,m;
24
        int u,v;
        while(scanf("%d%d",&n,&m)==2){
25
            if(n==0 && m == 0)break;
26
27
            init();
            while(m——){
28
29
                scanf("%d%d",&u,&v);
30
                u++; v++;
31
                addedge(u,v);
32
                addedge(v,u);
33
            }
34
            solve(n);
35
        }
36
        return 0;
37
   }
```

# 4.7 边双连通分支

去掉桥,其余的连通分支就是边双连通分支了。一个有桥的连通图要变成边双连通图的话,把双连通子图收缩为一个点,形成一颗树。需要加的边为 (leaf+1)/2 (leaf 为叶子结点个数) POJ 3177 给定一个连通的无向图 G,至少要添加几条边,才能使其变为双连通图。

```
1 const int MAXN = 5010; //点数
  struct Edge{
       int to,next;
 4
 5
       bool cut;//是否是桥标记
 6 | }edge[MAXM];
 7
   int head[MAXN],tot;
   |int Low[MAXN],DFN[MAXN],Stack[MAXN],Belong[MAXN];//Belong 数组的值是
      1 \sim block
9 | int Index, top;
  11
  bool Instack[MAXN];
12 int bridge; //桥的数目
13
   void addedge(int u,int v){
       edge[tot].to = v;edge[tot].next = head[u];edge[tot].cut=false;
14
15
       head[u] = tot++;
16
   }
   void Tarjan(int u,int pre){
17
18
       int v;
19
       Low[u] = DFN[u] = ++Index;
20
       Stack[top++] = u;
21
       Instack[u] = true;
       int pre_cnt = 0;
22
23
       for(int i = head[u];i != -1;i = edge[i].next){
24
           v = edge[i].to;
25
           if(v == pre && pre_cnt == 0){pre_cnt++; continue;}
           if(!DFN[v]){
26
27
              Tarjan(v,u);
              if( Low[u] > Low[v] )Low[u] = Low[v];
28
              if(Low[v] > DFN[u]){
29
30
                  bridge++;
31
                  edge[i].cut = true;
                  edge[i^1].cut = true;
32
              }
33
34
           }
          else if( Instack[v] && Low[u] > DFN[v] )
35
36
              Low[u] = DFN[v];
37
       if(Low[u] == DFN[u]){
38
           block++;
39
40
           do{
41
              v = Stack[—top];
42
              Instack[v] = false;
              Belong[v] = block;
43
44
           }
```

```
45
            while( v!=u );
46
       }
47
   }
48
   void init(){
49
       tot = 0;
50
       memset(head,-1,sizeof(head));
51
52
   int du[MAXN];//缩点后形成树,每个点的度数
53
   void solve(int n){
54
       memset(DFN,0,sizeof(DFN));
       memset(Instack, false, sizeof(Instack));
55
       Index = top = block = 0;
56
       Tarjan(1,0);
57
       int ans = 0;
58
       memset(du,0,sizeof(du));
59
       for(int i = 1;i <= n;i++)</pre>
60
           for(int j = head[i];j != -1;j = edge[j].next)
61
62
              if(edge[j].cut)
                 du[Belong[i]]++;
63
       for(int i = 1;i <= block;i++)</pre>
64
           if(du[i]==1)
65
66
              ans++;
67
        //找叶子结点的个数 ans,构造边双连通图需要加边 (ans+1)/2
       printf("%d\n",(ans+1)/2);
68
69
70
   int main(){
71
       int n,m;
72
       int u,v;
73
       while(scanf("%d%d",&n,&m)==2){
74
            init();
75
            while(m——){
                scanf("%d%d",&u,&v);
76
77
                addedge(u,v);
78
                addedge(v,u);
79
            }
80
            solve(n);
81
       }
82
       return 0;
83 |}
```

# 4.8 点双连通分支

对于点双连通分支,实际上在求割点的过程中就能顺便把每个点双连通分支求出。建立一个栈,存储当前双连通分支,在搜索图时,每找到一条树枝边或后向边 (非横叉边),就把这条边加入栈中。如果遇到某时满足 DFS(u) <= Low(v),说明 u 是一个割点,同时把边从栈顶一个个取出,直到遇到了边 (u,v),取出的这些边与其关联的点,组成一个点双连通分支。割点可以属于多个点双连通分支,其余点和每条边只属于且属于一个点双连通分支。  $POJ\ 2942$ 

奇圈,二分图判断的染色法,求点双连通分支

```
1 |/*
2 | POJ 2942 Knights of the Round Table
```

```
3 |亚瑟王要在圆桌上召开骑士会议,为了不引发骑士之间的冲突,
4 并且能够让会议的议题有令人满意的结果,每次开会前都必须对出席会议的骑士有如下要
     求:
  1、相互憎恨的两个骑士不能坐在直接相邻的 2 个位置;
  2、出席会议的骑士数必须是奇数,这是为了让投票表决议题时都能有结果。
7
  注意: 1、所给出的憎恨关系一定是双向的,不存在单向憎恨关系。
  2、由于是圆桌会议,则每个出席的骑士身边必定刚好有 2 个骑士。
  |即每个骑士的座位两边都必定各有一个骑士。
11 3、一个骑士无法开会,就是说至少有 3 个骑士才可能开会。
12
  |首先根据给出的互相憎恨的图中得到补图。
13
14 | 然后就相当于找出不能形成奇圈的点。
  利用下面两个定理:
15
  |(1)如果一个双连通分量内的某些顶点在一个奇圈中(即双连通分量含有奇圈),
17 那么这个双连通分量的其他顶点也在某个奇圈中;
18 (2) 如果一个双连通分量含有奇圈,则他必定不是一个二分图。反过来也成立,这是一个
     充要条件。
19
20 所以本题的做法,就是对补图求点双连通分量。
21 | 然后对于求得的点双连通分量,使用染色法判断是不是二分图,不是二分图,这个双连通分
     量的点是可以存在的
22 | */
23
24 | const int MAXN = 1010;
25 const int MAXM = 2000010;
26 struct Edge{
27
      int to.next;
28 | }edge[MAXM];
  int head[MAXN],tot;
30 | int Low[MAXN], DFN[MAXN], Stack[MAXN], Belong[MAXN];
31 | int Index, top;
  |int block;//点双连通分量的个数
32
33 bool Instack[MAXN];
34 bool can[MAXN];
35 | bool ok[MAXN];//标记
  |int tmp[MAXN];//暂时存储双连通分量中的点
37
  |int cc;//tmp 的计数
38 | int color[MAXN];//染色
  void addedge(int u,int v){
39
40
      edge[tot].to = v;edge[tot].next = head[u];head[u] = tot++;
41
  //染色判断二分图
42
  bool dfs(int u,int col){
43
      color[u] = col;
44
      for(int i = head[u]; i != -1; i = edge[i].next){
45
         int v = edge[i].to;
46
```

if(color[v]==col)return false;

if( !ok[v] )continue;

**if**(color[v] != −1){

continue:

47

48

49

50

```
51
             }
 52
             if(!dfs(v,!col))return false;
 53
 54
         return true;
 55
 56
    void Tarjan(int u,int pre){
 57
         int v;
 58
         Low[u] = DFN[u] = ++Index;
 59
         Stack[top++] = u;
         Instack[u] = true;
 60
 61
         int pre_cnt = 0;
         for(int i = head[u];i != -1;i = edge[i].next){
 62
 63
             v = edge[i].to;
             if(v == pre && pre_cnt == 0){pre_cnt++; continue;}
 64
             if(!DFN[v]){
 65
                 Tarjan(v,u);
 66
                 if(Low[u] > Low[v])Low[u] = Low[v];
 67
                 if( Low[v] >= DFN[u]){
 68
 69
                      block++;
 70
                      int vn;
 71
                      cc = 0;
                      memset(ok, false, sizeof(ok));
 72
 73
                      do{
                          vn = Stack[--top];
 74
 75
                          Belong[vn] = block;
 76
                          Instack[vn] = false;
 77
                          ok[vn] = true;
 78
                          tmp[cc++] = vn;
 79
                      }
 80
                      while( vn!=v );
 81
                      ok[u] = 1;
                      memset(color,-1,sizeof(color));
 82
 83
                      if(!dfs(u,0)){
 84
                          can[u] = true;
 85
                          while(cc—)can[tmp[cc]]=true;
 86
                      }
                 }
 87
 88
 89
             else if(Instack[v] && Low[u] > DFN[v])
                Low[u] = DFN[v];
 90
         }
 91
 92
 93
    void solve(int n){
 94
         memset(DFN,0,sizeof(DFN));
         memset(Instack, false, sizeof(Instack));
 95
         Index = block = top = 0;
 96
         memset(can, false, sizeof(can));
 97
         for(int i = 1;i <= n;i++)</pre>
 98
            if(!DFN[i])
 99
100
               Tarjan(i,-1);
101
         int ans = n;
```

```
102
         for(int i = 1;i <= n;i++)</pre>
103
              if(can[i])
104
                ans—;
105
         printf("%d\n",ans);
106
107
    void init(){
108
         tot = 0;
         memset(head,-1,sizeof(head));
109
110
    }
111
    int g[MAXN][MAXN];
    int main(){
112
113
         int n,m;
114
         int u,v;
         while(scanf("%d%d",&n,&m)==2){
115
              if(n==0 && m==0)break;
116
117
              init();
             memset(g, 0, sizeof(g));
118
119
             while(m——){
                  scanf("%d%d",&u,&v);
120
121
                  g[u][v]=g[v][u]=1;
              }
122
123
              for(int i = 1;i <= n;i++)</pre>
                for(int j = 1; j <= n; j++)</pre>
124
                   if(i != j && g[i][j]==0)
125
126
                      addedge(i,j);
127
             solve(n);
128
         }
129
         return 0;
130 |}
```

### 4.9 最小树形图

```
/*
1
2
    * 最小树形图
3
    * int 型
   * 复杂度 O(NM)
4
5
    * 点从 0 开始
6
    */
   const int INF = 0x3f3f3f3f;
   const int MAXN = 1010;
   const int MAXM = 40010;
   struct Edge{
10
11
       int u,v,cost;
12
   };
   Edge edge[MAXM];
13
   int pre[MAXN],id[MAXN],visit[MAXN],in[MAXN];
14
   int zhuliu(int root,int n,int m,Edge edge[]){
15
       int res = 0,u,v;
16
       while(1){
17
           for(int i = 0;i < n;i++)</pre>
18
19
                in[i] = INF;
```

```
20
            for(int i = 0;i < m;i++)
21
                if(edge[i].u != edge[i].v && edge[i].cost < in[edge[i].</pre>
                   \]){
22
                     pre[edge[i].v] = edge[i].u;
23
                     in[edge[i].v] = edge[i].cost;
24
25
            for(int i = 0;i < n;i++)</pre>
                if(i != root && in[i] == INF)
26
27
                     return -1; // 不存在最小树形图
28
            int tn = 0;
            memset(id, -1, sizeof(id));
29
30
            memset(visit,-1,sizeof(visit));
            in[root] = 0;
31
            for(int i = 0;i < n;i++){</pre>
32
                res += in[i];
33
34
                v = i;
                while( visit[v] != i && id[v] == −1 && v != root){
35
36
                     visit[v] = i;
37
                     v = pre[v];
38
39
                if( v != root \&\& id[v] == -1){
40
                     for(int u = pre[v]; u != v ;u = pre[u])
41
                         id[u] = tn;
42
                     id[v] = tn++;
43
                }
44
45
            if(tn == 0)break;//没有有向环
46
            for(int i = 0;i < n;i++)</pre>
47
                if(id[i] == -1)
48
                     id[i] = tn++;
49
            for(int i = 0;i < m;){</pre>
50
                v = edge[i].v;
                edge[i].u = id[edge[i].u];
51
52
                edge[i].v = id[edge[i].v];
53
                if(edge[i].u != edge[i].v)
54
                     edge[i++].cost -= in[v];
55
                else
56
                     swap(edge[i],edge[--m]);
57
            }
58
            n = tn;
59
            root = id[root];
60
61
        return res;
62
63
   int g[MAXN][MAXN];
   int main(){
64
65
        int n,m;
        int iCase = 0;
66
        int T;
67
        scanf("%d",&T);
68
        while( T— ){
69
```

```
iCase ++;
70
71
          scanf("%d%d",&n,&m);
          for(int i = 0;i < n;i++)
72
              for(int j = 0; j < n; j++)
73
                  g[i][j] = INF;
74
75
          int u,v,cost;
          while(m——){
76
              scanf("%d%d%d",&u,&v,&cost);
77
78
              if(u == v)continue;
79
              g[u][v] = min(g[u][v], cost);
80
          int L = 0;
81
          for(int i = 0;i < n;i++)</pre>
82
              for(int j = 0; j < n; j++)
83
                  if(g[i][j] < INF){
84
                     edge[L].u = i;
85
86
                     edge[L].v = j;
                     edge[L++].cost = g[i][j];
87
88
                  }
89
          int ans = zhuliu(0,n,L,edge);
          printf("Case_#%d:_",iCase);
90
91
          if(ans == -1)printf("Possums!\n");
92
          else printf("%d\n",ans);
93
      }
94
      return 0;
95 | }
        二分图匹配
   4.10
         邻接矩阵(匈牙利算法)
   4.10.1
1 /* ******************************
  |//二分图匹配(匈牙利算法的 DFS 实现)(邻接矩阵形式)
  |//初始化:g[][] 两边顶点的划分情况
  |//建立 g[i][j] 表示 i->j 的有向边就可以了,是左边向右边的匹配
  //g 没有边相连则初始化为 0
  //uN 是匹配左边的顶点数,vN 是匹配右边的顶点数
  |//调用: res=hungary(); 输出最大匹配数
   //优点: 适用于稠密图, DFS 找增广路, 实现简洁易于理解
  |//时间复杂度:O(VE)
  //**********************
10
  //顶点编号从 0 开始的
11
  const int MAXN = 510;
  |int uN,vN;//u,v 的数目,使用前面必须赋值
  |int g[MAXN][MAXN];//邻接矩阵
15
   int linker[MAXN];
16
  bool used[MAXN];
17
   bool dfs(int u){
      for(int \vee = 0; \vee < \vee N;\vee ++)
18
19
          if(g[u][v] && !used[v]){
20
              used[v] = true;
              if(linker[v] == -1 || dfs(linker[v])){
21
```

```
22
                    linker[v] = u;
23
                    return true;
24
                }
25
26
       return false;
27
28
   int hungary(){
29
       int res = 0;
       memset(linker,-1,sizeof(linker));
30
31
       for(int u = 0; u < uN; u++){}
            memset(used, false, sizeof(used));
32
33
            if(dfs(u))res++;
34
       }
35
       return res;
36 | }
   4.10.2 邻接表(匈牙利算法)
 1
  /*
 2
    * 匈牙利算法邻接表形式
 3
    * 使用前用 init() 进行初始化, 给 uN 赋值
    * 加边使用函数 addedge(u,v)
 4
 5
    *
 6
    */
   const int MAXN = 5010;//点数的最大值
   | const int MAXM = 50010;//边数的最大值
9
   struct Edge{
       int to,next;
10
11
   }edge[MAXM];
12
   | int head[MAXN],tot;
   void init(){
13
14
       tot = 0;
15
       memset(head,-1,sizeof(head));
16
17
   void addedge(int u,int v){
       edge[tot].to = v; edge[tot].next = head[u];
18
19
       head[u] = tot++;
20
   |int linker[MAXN];
21
22 bool used[MAXN];
23
   int uN;
   bool dfs(int u){
24
       for(int i = head[u]; i != -1; i = edge[i].next){
25
            int v = edge[i].to;
26
            if(!used[v]){
27
28
                used[v] = true;
                if(linker[v] == -1 || dfs(linker[v])){
29
                    linker[v] = u;
30
31
                    return true;
32
                }
33
            }
34
       }
```

```
35
        return false;
36
37
   int hungary(){
38
        int res = 0;
39
        memset(linker,-1,sizeof(linker));
40
        //点的编号 0~uN-1
41
        for(int u = 0; u < uN;u++){}
42
            memset(used, false, sizeof(used));
43
            if(dfs(u))res++;
44
45
        return res;
46 |}
   4.10.3 Hopcroft-Karp 算法
   /* ****************
    * 二分图匹配(Hopcroft-Karp 算法)
 2
 3
   * 复杂度 O(sqrt(n)*E)
 4
    * 邻接表存图, vector 实现
    * vector 先初始化,然后假如边
 6
    * uN 为左端的顶点数,使用前赋值 (点编号 0 开始)
 7
    */
   const int MAXN = 3000;
   const int INF = 0x3f3f3f3f;
10 | vector<int>G[MAXN];
11 | int uN;
12
   |int Mx[MAXN],My[MAXN];
13
   int dx[MAXN],dy[MAXN];
   int dis;
14
   bool used[MAXN];
15
   bool SearchP(){
16
17
       queue<int>Q;
18
        dis = INF;
        memset(dx, -1, sizeof(dx));
19
        memset(dy, -1, sizeof(dy));
20
        for(int i = 0 ; i < uN; i++)</pre>
21
22
            \mathbf{if}(\mathsf{M} \times \lceil \mathsf{i} \rceil == -1)
23
                Q.push(i);
                dx[i] = 0;
24
25
            }
        while(!Q.empty()){
26
            int u = Q.front();
27
28
            Q.pop();
            if(dx[u] > dis)break;
29
30
            int sz = G[u].size();
31
            for(int i = 0;i < sz;i++){</pre>
32
                int v = G[u][i];
                if(dy[v] == -1){
33
34
                     dy[v] = dx[u] + 1;
35
                     if(My[v] == -1)dis = dy[v];
36
                     else{
37
                         dx[My[v]] = dy[v] + 1;
```

```
38
                          Q.push(My[v]);
39
                     }
40
                 }
41
            }
42
        }
43
        return dis != INF;
44
45
   bool DFS(int u){
46
        int sz = G[u].size();
47
        for(int i = 0;i < sz;i++){</pre>
48
            int v = G[u][i];
49
            if(!used[v] && dy[v] == dx[u] + 1){
50
                 used[v] = true;
                 if (My[v] != -1 \&\& dy[v] == dis) continue;
51
52
                 if(My[v] == -1 || DFS(My[v])){
53
                     My[v] = u;
54
                     Mx[u] = v;
55
                     return true;
                 }
56
57
            }
58
59
        return false;
60
61
   int MaxMatch(){
62
        int res = 0;
63
        memset(Mx, -1, sizeof(Mx));
        memset(My, -1, sizeof(My));
64
        while(SearchP()){
65
66
            memset(used, false, sizeof(used));
67
            for(int i = 0;i < uN;i++)</pre>
                 if(Mx[i] == -1 \&\& DFS(i))
68
69
                     res++;
70
71
        return res;
72 | }
          二分图多重匹配
   4.11
 1 | const int MAXN = 1010;
   const int MAXM = 510;
   int uN, vN;
   int g[MAXN][MAXM];
   int linker[MAXM][MAXN];
   bool used[MAXM];
 7
   int num[MAXM];//右边最大的匹配数
   bool dfs(int u){
 9
        for(int \vee = 0; \vee < \vee N;\vee ++)
10
            if(g[u][v] && !used[v]){
                 used[v] = true;
11
12
                 if(linker[v][0] < num[v]){
13
                     linker[v][++linker[v][0]] = u;
14
                     return true;
```

```
15
               }
16
               for(int i = 1;i <= num[v];i++)</pre>
                    if(dfs(linker[v][i])){
17
                        linker[v][i] = u;
18
19
                        return true;
20
                    }
21
22
       return false;
23
   }
   int hungary(){
24
       int res = 0;
25
       for(int i = 0;i < vN;i++)</pre>
26
           linker[i][0] = 0;
27
       for(int u = 0; u < uN; u++){
28
           memset(used, false, sizeof(used));
29
30
           if(dfs(u))res++;
31
32
       return res;
33 | }
   4.12 二分图最大权匹配(KM 算法)
   /*
       KM 算法
 1
 2
        复杂度 O(nx*nx*ny)
 3
    *
       求最大权匹配
        若求最小权匹配,可将权值取相反数,结果取相反数
 4
 5
       点的编号从 0 开始
    *
 6
    */
   const int N = 310;
   const int INF = 0x3f3f3f3f;
   |int nx,ny;//两边的点数
   |int g[N][N];//二分图描述
   |int linker[N],lx[N],ly[N];//y 中各点匹配状态, x,y 中的点标号
11
12
   int slack[N];
   bool visx[N], visy[N];
13
   bool DFS(int x){
15
       visx[x] = true;
       for(int y = 0; y < ny; y++){
16
           if(visy[y])continue;
17
           int tmp = lx[x] + ly[y] - g[x][y];
18
19
           if(tmp == 0){
20
               visy[y] = true;
21
               if(linker[y] == -1 || DFS(linker[y])){
                    linker[y] = x;
22
23
                    return true;
24
               }
25
26
           else if(slack[y] > tmp)
27
               slack[y] = tmp;
28
29
       return false;
30
   }
```

```
31
   int KM(){
32
        memset(linker,-1,sizeof(linker));
33
        memset(ly,0,sizeof(ly));
        for(int i = 0;i < nx;i++){</pre>
34
35
            lx[i] = -INF;
            for(int j = 0;j < ny;j++)</pre>
36
                 if(g[i][j] > lx[i])
37
38
                     lx[i] = g[i][j];
39
        }
40
        for(int x = 0; x < nx; x++){
            for(int i = 0;i < ny;i++)</pre>
41
42
                 slack[i] = INF;
43
            while(true){
44
                 memset(visx, false, sizeof(visx));
                 memset(visy, false, sizeof(visy));
45
46
                 if(DFS(x))break;
                 int d = INF;
47
48
                 for(int i = 0;i < ny;i++)
49
                     if(!visy[i] && d > slack[i])
                          d = slack[i];
50
                 for(int i = 0;i < nx;i++)</pre>
51
52
                     if(visx[i])
53
                          lx[i] = d;
                 for(int i = 0; i < ny; i++){
54
55
                     if(visy[i])ly[i] += d;
56
                     else slack[i] -= d;
57
                 }
            }
58
59
        int res = 0;
60
        for(int i = 0;i < ny;i++)</pre>
61
62
            if(linker[i] != -1)
63
                 res += g[linker[i]][i];
64
        return res;
65
66
   //HDU 2255
67
   int main(){
68
        int n;
69
        while(scanf("%d",&n) == 1){
            for(int i = 0;i < n;i++)</pre>
70
71
                 for(int j = 0; j < n; j++)
                     scanf("%d",&g[i][j]);
72
73
            nx = ny = n;
74
            printf("%d\n",KM());
75
76
        return 0;
77 | }
          一般图匹配带花树
   4.13
   URAL 1099
 1 | const int MAXN = 250;
```

```
2 | int N; //点的个数,点的编号从 1 到 N
   bool Graph[MAXN][MAXN];
   int Match[MAXN];
 5 | bool InQueue[MAXN], InPath[MAXN], InBlossom[MAXN];
   int Head, Tail;
   int Queue[MAXN];
   int Start,Finish;
   int NewBase;
   int Father[MAXN],Base[MAXN];
10
   int Count;//匹配数, 匹配对数是 Count/2
11
   void CreateGraph(){
12
13
       int u,v;
14
       memset(Graph, false, sizeof(Graph));
       scanf("%d",&N);
15
       while(scanf("%d%d",&u,&v) == 2){
16
17
            Graph[u][v] = Graph[v][u] = true;
       }
18
19
20
   void Push(int u){
21
       Queue[Tail] = u;
22
       Tail++;
23
       InQueue[u] = true;
24
25
   int Pop(){
26
       int res = Queue[Head];
27
       Head++;
28
       return res;
29
30
   int FindCommonAncestor(int u,int v){
31
       memset(InPath, false, sizeof(InPath));
       while(true){
32
           u = Base[u];
33
            InPath[u] = true;
34
35
            if(u == Start) break;
            u = Father[Match[u]];
36
37
       }
38
       while(true){
39
           v = Base[v];
40
            if(InPath[v])break;
41
            v = Father[Match[v]];
42
       }
43
       return v;
44
45
   void ResetTrace(int u){
46
       int ∨;
47
       while(Base[u] != NewBase){
48
            v = Match[u];
49
            InBlossom[Base[u]] = InBlossom[Base[v]] = true;
50
            u = Father[v];
51
            if(Base[u] != NewBase) Father[u] = v;
       }
52
```

```
53
    }
 54
    void BloosomContract(int u,int v){
 55
         NewBase = FindCommonAncestor(u,v);
         memset(InBlossom, false, sizeof(InBlossom));
 56
         ResetTrace(u);
 57
 58
         ResetTrace(v);
         if(Base[u] != NewBase) Father[u] = v;
 59
         if(Base[v] != NewBase) Father[v] = u;
 60
         for(int tu = 1; tu <= N; tu++)</pre>
 61
             if(InBlossom[Base[tu]]){
 62
                  Base[tu] = NewBase;
 63
                 if(!InQueue[tu]) Push(tu);
 64
             }
 65
 66
    void FindAugmentingPath(){
 67
         memset(InQueue, false, sizeof(InQueue));
 68
 69
         memset(Father, 0, size of (Father));
         for(int i = 1;i <= N;i++)</pre>
 70
 71
             Base[i] = i;
         Head = Tail = 1;
 72
         Push(Start);
 73
 74
         Finish = 0;
         while(Head < Tail){</pre>
 75
             int u = Pop();
 76
 77
             for(int v = 1; v \le N; v++)
                  if(Graph[u][v] && (Base[u] != Base[v]) && (Match[u] !=
 78
                     v)){
 79
                      if((v == Start) || ((Match[v] > 0) && Father[Match[
                         V] > 0))
 80
                          BloosomContract(u,v);
                      else if(Father[v] == 0){
 81
 82
                          Father[v] = u;
 83
                          if(Match[v] > 0)
 84
                               Push(Match[v]);
 85
                          else{
 86
                               Finish = v;
 87
                               return;
 88
                          }
 89
                      }
                 }
 90
 91
         }
 92
 93
    void AugmentPath(){
 94
         int u,v,w;
 95
         u = Finish;
         while (u > 0)
 96
 97
             v = Father[u];
 98
             w = Match[v];
 99
             Match[v] = u;
100
             Match[u] = v;
101
             u = w;
```

```
102
        }
103
    }
104
    void Edmonds(){
        memset(Match, 0, sizeof(Match));
105
        for(int u = 1; u <= N; u++)</pre>
106
            if(Match[u] == 0){
107
                Start = u;
108
                FindAugmentingPath();
109
110
                if(Finish > 0)AugmentPath();
            }
111
112
    void PrintMatch(){
113
        Count = 0;
114
        for(int u = 1; u <= N;u++)</pre>
115
            if(Match[u] > 0)
116
117
                Count++;
        printf("%d\n",Count);
118
        for(int u = 1; u <= N; u++)
119
            if(u < Match[u])</pre>
120
121
                122
123
    int main(){
124
        CreateGraph();//建图
        Edmonds();//进行匹配
125
        PrintMatch();//输出匹配数和匹配
126
127
        return 0;
128 |}
```

# 4.14 一般图最大加权匹配

```
1 //一般图的最大加权匹配模板
  //注意 G 的初始化, 需要偶数个点, 刚好可以形成 n/2 个匹配
  //如果要求最小权匹配,可以取相反数,或者稍加修改就可以了
  │//点的编号从 0 开始的
5 const int MAXN = 110;
  const int INF = 0x3f3f3f3f;
  |int G[MAXN][MAXN];
7
8 int cnt_node; //点的个数
  int dis[MAXN];
10 | int match[MAXN];
11 bool vis[MAXN];
  |int sta[MAXN],top;//堆栈
  bool dfs(int u){
13
14
       sta[top++] = u;
15
       if(vis[u])return true;
16
       vis[u] = true;
17
       for(int i = 0;i < cnt_node;i++)</pre>
           if(i != u && i != match[u] && !vis[i]){
18
19
              int t = match[i];
20
              if(dis[t] < dis[u] + G[u][i] - G[i][t]){
21
                  dis[t] = dis[u] + G[u][i] - G[i][t];
```

```
22
                     if(dfs(t))return true;
23
                 }
24
25
        top--;
26
        vis[u] = false;
27
        return false;
28
29
   int P[MAXN];
   //返回最大匹配权值
30
31
   int get_Match(int N){
        cnt_node = N;
32
33
        for(int i = 0;i < cnt_node;i++)P[i] = i;</pre>
34
        for(int i = 0;i < cnt_node;i += 2){</pre>
35
            match[i] = i+1;
36
            match[i+1] = i;
37
        }
        int cnt = 0;
38
        while(1){
39
40
            memset(dis,0,sizeof(dis));
            memset(vis, false, sizeof(vis));
41
42
            top = 0;
43
            bool update = false;
            for(int i = 0;i < cnt_node;i++)</pre>
44
45
                 if(dfs(P[i])){
46
                     update = true;
47
                     int tmp = match[sta[top-1]];
48
                     int j = top-2;
49
                     while(sta[j] != sta[top-1]){
50
                          match[tmp] = sta[j];
51
                          swap(tmp,match[sta[j]]);
52
                          j--;
53
                     }
54
                     match[tmp] = sta[j];
55
                     match[sta[j]] = tmp;
                     break;
56
57
                 }
58
            if(!update){
59
                 cnt++;
60
                 if(cnt >= 3)break;
                 random_shuffle(P,P+cnt_node);
61
            }
62
63
64
        int ans = 0;
65
        for(int i = 0;i < cnt_node;i++){</pre>
            int v = match[i];
66
            if(i < v)ans += G[i][v];
67
68
69
        return ans;
70 |}
```

### 4.15 生成树计数

Matrix-Tree 定理 (Kirchhoff 矩阵 -树定理)

- $1 \times G$  的度数矩阵 D[G] 是一个 n\*n 的矩阵,并且满足:当 i j 时,dij=0; 当 i=j 时, dij 等于 vi 的度数。
- $2 \times G$  的邻接矩阵 A[G] 也是一个 n\*n 的矩阵,并且满足: 如果  $vi \times vj$  之间有边直接相连,则 aij=1,否则为 0。

我们定义 G 的 Kirchhoff 矩阵 (也称为拉普拉斯算子)C[G] 为 C[G]=D[G]-A[G], 则 Matrix-Tree 定理可以描述为:

G 的所有不同的生成树的个数等于其 Kirchhoff 矩阵 C[G] 任何一个 n-1 阶主子式的行列式的 绝对值。

所谓 n-1 阶主子式,就是对于  $r(1\ r\ n)$ ,将 C[G] 的第 r 行、第 r 列同时去掉后得到的新矩阵,用 Cr[G] 表示。

```
//HDU 4305
   //求生成树计数部分代码,计数对 10007 取模
  const int MOD = 10007;
 2
   int INV[MOD];
   //求 ax = 1( mod m) 的 x 值, 就是逆元 (0<a<m)
 3
   long long inv(long long a,long long m){
 5
       if(a == 1)return 1;
 6
       return inv(m%a,m)*(m-m/a)%m;
 7
   }
   struct Matrix{
 9
       int mat[330][330];
       void init(){
10
11
           memset(mat,0,sizeof(mat));
12
       //求行列式的值模上,需要使用逆元MOD
13
       int det(int n){
14
15
           for(int i = 0;i < n;i++)
                for(int j = 0; j < n; j++)
16
                    mat[i][j] = (mat[i][j]%MOD+MOD)%MOD;
17
           int res = 1;
18
           for(int i = 0;i < n;i++){</pre>
19
                for(int j = i; j < n; j++)
20
                    if(mat[j][i]!=0){
21
                        for(int k = i;k < n;k++)
22
23
                            swap(mat[i][k],mat[j][k]);
24
                        if(i != j)
25
                            res = (-res+MOD)%MOD;
26
                        break;
27
                    }
28
                if(mat[i][i] == 0){
29
                    res = -1; // 不存在也就是行列式值为(0)
30
                    break;
31
32
                for(int j = i+1; j < n; j++){
                    //int mut = (mat[j][i]*INV[mat[i][i]])%MOD打表逆
33
                       元;//
34
                    int mut = (mat[j][i]*inv(mat[i][i],MOD))%MOD;
```

```
for(int k = i;k < n;k++)
35
36
                         mat[j][k] = (mat[j][k]-(mat[i][k]*mut)%MOD+MOD)
                            %MOD;
37
                 }
38
                res = (res * mat[i][i])%MOD;
39
            }
40
            return res;
41
        }
   };
42
43
44
            Matrix ret;
45
            ret.init();
            for(int i = 0;i < n;i++)</pre>
46
47
                 for(int j = 0; j < n; j++)
48
                     if(i != j && g[i][j]){
49
                         ret.mat[i][j] = -1;
50
                         ret.mat[i][i]++;
51
52
            printf("%d\n",ret.det(n-1));
   计算生成树个数,不取模,SPOJ 104
   const double eps = 1e-8;
 2
   const int MAXN = 110;
 3
   int sgn(double x){
 4
        if(fabs(x) < eps)return 0;</pre>
 5
        if(x < 0) return -1;
 6
        else return 1;
 7
   }
   double b[MAXN][MAXN];
 9
   double det(double a[][MAXN],int n){
10
        int i, j, k, sign = 0;
11
        double ret = 1;
12
        for(i = 0;i < n;i++)
13
            for(j = 0; j < n; j++)
                b[i][j] = a[i][j];
14
15
        for(i = 0;i < n;i++){</pre>
            if(sgn(b[i][i]) == 0){
16
17
                 for(j = i + 1; j < n; j++)
                     if(sgn(b[j][i]) != 0)
18
19
                         break;
20
                if(j == n)return 0;
21
                 for(k = i;k < n;k++)
22
                     swap(b[i][k],b[j][k]);
23
                sign++;
24
            }
25
            ret *= b[i][i];
            for(k = i + 1; k < n; k++)
26
27
                b[i][k]/=b[i][i];
28
            for(j = i+1; j < n; j++)
                 for(k = i+1; k < n; k++)
29
30
                     b[j][k] -= b[j][i]*b[i][k];
31
        }
```

```
32
       if(sign & 1)ret = -ret;
33
       return ret;
34
   }
35
   double a[MAXN][MAXN];
   int g[MAXN][MAXN];
36
37
   int main(){
       int T;
38
39
       int n,m;
40
       int u,v;
41
       scanf("%d",&T);
42
       while(T--){
43
            scanf("%d%d",&n,&m);
44
            memset(g,0,sizeof(g));
45
            while(m——){
46
                scanf("%d%d",&u,&v);
47
                u--;v--;
48
                g[u][v] = g[v][u] = 1;
49
            }
            memset(a,0,sizeof(a));
50
            for(int i = 0;i < n;i++)
51
                for(int j = 0; j < n; j++)
52
53
                    if(i != j && g[i][j]){
54
                         a[i][i]++;
55
                         a[i][j] = -1;
56
                     }
57
            double ans = det(a, n-1);
            printf("%.0lf\n",ans);
58
59
60
       return 0;
61 |}
   4.16
          最大流
   4.16.1 SAP 邻接矩阵形式
 1 | / *
 2
    * SAP 算法(矩阵形式)
 3
    * 结点编号从 0 开始
 4
    */
   const int MAXN=1100;
   int maze[MAXN][MAXN];
 7
   int gap[MAXN],dis[MAXN],pre[MAXN],cur[MAXN];
 8
   int sap(int start,int end,int nodenum){
       memset(cur,0,sizeof(cur));
 9
       memset(dis,0,sizeof(dis));
10
       memset(gap,0,sizeof(gap));
11
12
       int u=pre[start]=start,maxflow=0,aug=-1;
13
       gap[0]=nodenum;
14
       while(dis[start]<nodenum){</pre>
15
            loop:
16
              for(int v=cur[u];v<nodenum;v++)</pre>
```

```
if(maze[u][v] && dis[u]==dis[v]+1){
17
18
                    if(aug==-1 || aug>maze[u][v])aug=maze[u][v];
                    pre[v]=u;
19
                    u=cur[u]=v;
20
21
                    if(v==end){
22
                        maxflow+=aug;
23
                        for(u=pre[u];v!=start;v=u,u=pre[u]){
24
                            maze[u][v]—=aug;
25
                            maze[v][u]+=aug;
26
                        }
27
                        aug=-1;
28
                    }
29
                    goto loop;
30
               int mindis=nodenum-1;
31
                for(int v=0;v<nodenum;v++)</pre>
32
                   if(maze[u][v]&&mindis>dis[v]){
33
34
                       cur[u]=v;
35
                       mindis=dis[v];
36
                   }
               if((--gap[dis[u]])==0)break;
37
38
               gap[dis[u]=mindis+1]++;
39
               u=pre[u];
40
41
       return maxflow;
42
   4.16.2 SAP 邻接矩阵形式 2
   保留原矩阵,可用于多次使用最大流
 1 |/*
 2
    * SAP 邻接矩阵形式
 3
    * 点的编号从 0 开始
   * 增加个 flow 数组,保留原矩阵 maze,可用于多次使用最大流
 5
   */
   const int MAXN=1100;
 6
   int maze[MAXN][MAXN];
   int gap[MAXN],dis[MAXN],pre[MAXN],cur[MAXN];
   |int flow[MAXN][MAXN];//存最大流的容量
   int sap(int start,int end,int nodenum){
10
       memset(cur,0,sizeof(cur));
11
       memset(dis,0,sizeof(dis));
12
       memset(gap,0,sizeof(gap));
13
       memset(flow,0,sizeof(flow));
14
       int u=pre[start]=start,maxflow=0,aug=-1;
15
16
       gap[0]=nodenum;
       while(dis[start]<nodenum) {</pre>
17
18
           loop:
19
              for(int v=cur[u];v<nodenum;v++)</pre>
                if(maze[u][v]-flow[u][v] && dis[u]==dis[v]+1){
20
                    if(aug==-1 || aug>maze[u][v]-flow[u][v])aug=maze[u
21
                       ][v]-flow[u][v];
```

```
22
                   pre[v]=u;
23
                   u=cur[u]=v;
                   if(v==end){
24
25
                       maxflow+=aug;
26
                       for(u=pre[u];v!=start;v=u,u=pre[u]){
27
                            flow[u][v]+=aug;
28
                            flow[v][u]—=aug;
29
                       }
30
                       aug=-1;
31
32
                   goto loop;
33
34
               int mindis=nodenum-1;
               for(int v=0;v<nodenum;v++)</pre>
35
                  if(maze[u][v]-flow[u][v]&&mindis>dis[v]){
36
37
                      cur[u]=v;
38
                      mindis=dis[v];
39
                  }
40
               if((--gap[dis[u]])==0)break;
41
               gap[dis[u]=mindis+1]++;
42
               u=pre[u];
43
       }
44
       return maxflow;
45 |}
   4.16.3 ISAP 邻接表形式
 1 const int MAXN = 100010; //点数的最大值
   const int MAXM = 400010;//边数的最大值
  const int INF = 0x3f3f3f3f;
   struct Edge{
       int to,next,cap,flow;
   |}edge[MAXM];//注意是 MAXM
 7
  int tol;
   |int head[MAXN];
   int gap[MAXN],dep[MAXN],pre[MAXN],cur[MAXN];
   void init(){
10
11
       tol = 0;
12
       memset(head,-1,sizeof(head));
13
   //加边,单向图三个参数,双向图四个参数
14
15
   void addedge(int u,int v,int w,int rw=0){
       edge[tol].to = v;edge[tol].cap = w;edge[tol].next = head[u];
16
17
       edge[tol].flow = 0;head[u] = tol++;
       edge[tol].to = u;edge[tol].cap = rw;edge[tol].next = head[v];
18
19
       edge[tol].flow = 0;head[v]=tol++;
20
21
   |//输入参数:起点、终点、点的总数
22
   //点的编号没有影响,只要输入点的总数
   int sap(int start,int end,int N){
23
       memset(gap,0,sizeof(gap));
24
```

```
25
        memset(dep,0,sizeof(dep));
        memcpy(cur,head,sizeof(head));
26
27
        int u = start;
        pre[u] = -1;
28
29
        gap[0] = N;
30
        int ans = 0;
31
        while(dep[start] < N){</pre>
            if(u == end){
32
                int Min = INF;
33
34
                for(int i = pre[u];i != -1; i = pre[edge[i^1].to])
                     if(Min > edge[i].cap - edge[i].flow)
35
                             Min = edge[i].cap - edge[i].flow;
36
                for(int i = pre[u]; i != -1; i = pre[edge[i^1].to]){
37
                     edge[i].flow += Min;
38
                     edge[i^1].flow -= Min;
39
40
                }
41
                u = start;
42
                ans += Min;
43
                continue;
44
            }
45
            bool flag = false;
46
            int v;
47
            for(int i = cur[u]; i != -1; i = edge[i].next){
                v = edge[i].to;
48
49
                if(edge[i].cap - edge[i].flow && dep[v]+1 == dep[u])
50
51
                     flag = true;
52
                     cur[u] = pre[v] = i;
53
                    break;
54
                }
55
            if(flag){
56
57
                u = v;
58
                continue;
59
60
            int Min = N;
            for(int i = head[u]; i != -1;i = edge[i].next)
61
                if(edge[i].cap - edge[i].flow && dep[edge[i].to] < Min)</pre>
62
63
                     Min = dep[edge[i].to];
64
                     cur[u] = i;
65
66
            gap[dep[u]]--;
67
            if(!gap[dep[u]])return ans;
            dep[u] = Min+1;
68
            gap[dep[u]]++;
69
70
            if(u != start) u = edge[pre[u]^1].to;
71
        }
72
        return ans;
73 |}
```

#### 4.16.4 ISAP+bfs 初始化 + 栈优化

```
|const int MAXN = 100010;//点数的最大值
 2
   |const int MAXM = 400010;//边数的最大值
 3
   const int INF = 0x3f3f3f3f;
   struct Edge{
       int to,next,cap,flow;
   |}edge[MAXM];//注意是 MAXM
   int tol;
 7
   int head[MAXN];
9
   int gap[MAXN],dep[MAXN],cur[MAXN];
   void init(){
10
11
       tol = 0;
12
       memset(head,-1,sizeof(head));
13
14
   void addedge(int u,int v,int w,int rw = 0){
       edge[tol].to = v; edge[tol].cap = w; edge[tol].flow = 0;
15
       edge[tol].next = head[u]; head[u] = tol++;
16
       edge[tol].to = u; edge[tol].cap = rw; edge[tol].flow = 0;
17
18
       edge[tol].next = head[v]; head[v] = tol++;
19
20
   int Q[MAXN];
   void BFS(int start,int end){
21
22
       memset(dep, -1, sizeof(dep));
23
       memset(gap,0,sizeof(gap));
24
       gap[0] = 1;
25
       int front = 0, rear = 0;
26
       dep[end] = 0;
27
       Q[rear++] = end;
       while(front != rear){
28
29
            int u = Q[front++];
            for(int i = head[u]; i != -1; i = edge[i].next){
30
31
                int v = edge[i].to;
                if (dep[v] != -1) continue;
32
                Q[rear++] = v;
33
34
                dep[v] = dep[u] + 1;
35
                gap[dep[v]]++;
36
            }
37
       }
38
39
   int S[MAXN];
   int sap(int start,int end,int N){
40
41
       BFS(start,end);
       memcpy(cur,head,sizeof(head));
42
43
       int top = 0;
       int u = start;
44
45
       int ans = 0;
46
       while(dep[start] < N){</pre>
47
            if(u == end){
48
                int Min = INF;
49
                int inser;
```

```
50
                for(int i = 0;i < top;i++)
                    if(Min > edge[S[i]].cap - edge[S[i]].flow){
51
52
                         Min = edge[S[i]].cap - edge[S[i]].flow;
53
                         inser = i;
54
                    }
55
                for(int i = 0;i < top;i++){</pre>
                    edge[S[i]].flow += Min;
56
57
                    edge[S[i]^1].flow -= Min;
58
                }
59
                ans += Min;
                top = inser;
60
                u = edge[S[top]^1].to;
61
62
                continue;
63
            bool flag = false;
64
65
            int ∨;
            for(int i = cur[u]; i != -1; i = edge[i].next){
66
67
                v = edge[i].to;
                if(edge[i].cap - edge[i].flow && dep[v]+1 == dep[u]){
68
                     flag = true;
69
                    cur[u] = i;
70
71
                    break;
72
                }
73
            if(flag){
74
75
                S[top++] = cur[u];
76
                u = v;
77
                continue;
78
            }
79
            int Min = N;
            for(int i = head[u]; i != -1; i = edge[i].next)
80
81
                if(edge[i].cap - edge[i].flow && dep[edge[i].to] < Min)</pre>
82
                    Min = dep[edge[i].to];
83
                    cur[u] = i;
84
                }
85
            gap[dep[u]]--;
            if(!gap[dep[u]])return ans;
86
87
            dep[u] = Min + 1;
            gap[dep[u]]++;
88
89
            if(u != start)u = edge[S[--top]^1].to;
90
91
       return ans;
92 |}
   4.16.5 dinic
 1 const int MAXN = 2010; // 点数的最大值
   |const int MAXM = 1200010;//边数的最大值
   const int INF = 0x3f3f3f3f;
   struct Edge{
 5
       int to,next,cap,flow;
```

```
|}edge[MAXM];//注意是 MAXM
 7
   int tol;
 8
   int head[MAXN];
 9
   void init(){
       tol = 2;
10
11
       memset(head,-1,sizeof(head));
12
13
   void addedge(int u,int v,int w,int rw = 0){
14
       edge[tol].to = v; edge[tol].cap = w; edge[tol].flow = 0;
       edge[tol].next = head[u]; head[u] = tol++;
15
       edge[tol].to = u; edge[tol].cap = rw; edge[tol].flow = 0;
16
       edge[tol].next = head[v]; head[v] = tol++;
17
18
   int Q[MAXN];
19
   int dep[MAXN],cur[MAXN],sta[MAXN];
   bool bfs(int s,int t,int n){
21
       int front = 0,tail = 0;
22
       memset(dep, -1, sizeof(dep[0])*(n+1));
23
24
       dep[s] = 0;
       Q[tail++] = s;
25
       while(front < tail){</pre>
26
27
            int u = Q[front++];
            for(int i = head[u]; i != -1;i = edge[i].next){
28
                int v = edge[i].to;
29
30
                if(edge[i].cap > edge[i].flow && dep[v] == -1){
                    dep[v] = dep[u] + 1;
31
                    if(v == t)return true;
32
                    0[tail++] = v;
33
34
                }
35
            }
36
37
       return false;
38
39
   int dinic(int s,int t,int n){
40
       int maxflow = 0;
41
       while(bfs(s,t,n)){
            for(int i = 0;i < n;i++)cur[i] = head[i];</pre>
42
            int u = s, tail = 0;
43
            while (cur[s] != -1){
44
                if(u == t){
45
46
                    int tp = INF;
47
                    for(int i = tail-1;i >= 0;i--)
48
                        tp = min(tp,edge[sta[i]].cap-edge[sta[i]].flow)
                            ;
49
                    maxflow += tp;
                    for(int i = tail-1;i >= 0;i--){
50
                        edge[sta[i]].flow += tp;
51
                        edge[sta[i]^1].flow -= tp;
52
                        if(edge[sta[i]].cap-edge[sta[i]].flow == 0)
53
54
                             tail = i;
                    }
55
```

```
56
                    u = edge[sta[tail]^1].to;
57
                }
                else if(cur[u] != −1 && edge[cur[u]].cap > edge[cur[u
58
                   ]].flow && dep[u] + 1 == dep[edge[cur[u]].to]){
59
                    sta[tail++] = cur[u];
                    u = edge[cur[u]].to;
60
61
62
                else {
63
                    while(u != s && cur[u] == -1)
                        u = edge[sta[--tail]^1].to;
64
                    cur[u] = edge[cur[u]].next;
65
                }
66
67
            }
68
       }
69
       return maxflow;
70 |}
   4.16.6 最大流判断多解
   //判断最大流多解就是在残留网络中找正环
   |bool vis[MAXN],no[MAXN];
   int Stack[MAXN],top;
   bool dfs(int u,int pre,bool flag){
       vis[u] = 1;
 5
 6
       Stack[top++] = u;
 7
       for(int i = head[u];i != -1;i = edge[i].next)
 8
            int v = edge[i].to;
 9
            if(edge[i].cap <= edge[i].flow)continue;</pre>
10
            if(v == pre)continue;
11
            if(!vis[v]){
12
13
                if(dfs(v,u,edge[i^1].flow < edge[i^1].cap))return true;</pre>
14
15
            else if(!no[v])return true;
16
       if(!flag){
17
           while(1){
18
19
                int v = Stack[--top];
20
                no[v] = true;
21
                if(v == u)break;
22
23
       }
24
       return false;
25
   //执行完最大流后可进行调用
26
27
           memset(vis, false, sizeof(vis));
28
            memset(no, false, sizeof(no));
29
            top = 0;
30
            bool flag = dfs(end,end,0);
```

### 4.17 最小费用最大流

### 4.17.1 SPFA 版费用流

最小费用最大流,求最大费用只需要取相反数,结果取相反数即可。 点的总数为 N,点的编号  $0 \sim N-1$ 

```
const int MAXN = 10000;
 2
   const int MAXM = 100000;
   const int INF = 0x3f3f3f3f;
   struct Edge{
 5
       int to,next,cap,flow,cost;
   }edge[MAXM];
   int head[MAXN],tol;
 7
   |int pre[MAXN],dis[MAXN];
   bool vis[MAXN];
   int N; / / 节点总个数,节点编号从 0~N-1
10
   void init(int n){
11
12
       N = n;
13
       tol = 0;
14
       memset(head,-1,sizeof(head));
15
16
   void addedge(int u,int v,int cap,int cost){
17
       edge[tol].to = v;
18
       edge[tol].cap = cap;
       edge[tol].cost = cost;
19
       edge[tol].flow = 0;
20
       edge[tol].next = head[u];
21
       head[u] = tol++;
22
23
       edge[tol].to = u;
       edge[tol].cap = 0;
24
25
       edge[tol].cost = -cost;
       edge[tol].flow = 0;
26
       edge[tol].next = head[v];
27
28
       head[v] = tol++;
29
30
   |bool spfa(int s,int t){
31
       queue<int>q;
32
       for(int i = 0;i < N;i++){</pre>
            dis[i] = INF;
33
            vis[i] = false;
34
            pre[i] = -1;
35
       }
36
37
       dis[s] = 0;
       vis[s] = true;
38
39
       q.push(s);
40
       while(!q.empty()){
41
            int u = q.front();
42
            q.pop();
43
            vis[u] = false;
44
            for(int i = head[u]; i != -1;i = edge[i].next){
45
                int v = edge[i].to;
```

```
if(edge[i].cap > edge[i].flow && dis[v] > dis[u] + edge
46
                   [i].cost )
47
                {
48
                    dis[v] = dis[u] + edge[i].cost;
49
                    pre[v] = i;
                    if(!vis[v]){
50
51
                        vis[v] = true;
52
                        q.push(v);
53
                    }
54
                }
            }
55
56
57
       if(pre[t] == -1)return false;
       else return true;
58
59
   }
   //返回的是最大流, cost 存的是最小费用
60
   int minCostMaxflow(int s,int t,int &cost){
61
       int flow = 0;
62
       cost = 0;
63
       while(spfa(s,t)){
64
            int Min = INF;
65
66
            for(int i = pre[t];i != -1;i = pre[edge[i^1].to]){
67
                if(Min > edge[i].cap - edge[i].flow)
                    Min = edge[i].cap - edge[i].flow;
68
69
70
            for(int i = pre[t];i != -1;i = pre[edge[i^1].to]){
                edge[i].flow += Min;
71
                edge[i^1].flow -= Min;
72
73
                cost += edge[i].cost * Min;
74
            flow += Min;
75
76
       }
77
       return flow;
78 | }
   4.17.2 zkw 费用流
   对于二分图类型的比较高效
 1 const int MAXN = 100;
   const int MAXM = 20000;
   const int INF = 0x3f3f3f3f;
   struct Edge{
 5
       int to,next,cap,flow,cost;
 6
       Edge(int _to = 0,int _next = 0,int _cap = 0,int _flow = 0,int
          _cost = 0):
 7
            to(_to),next(_next),cap(_cap),flow(_flow),cost(_cost){}
   }edge[MAXM];
 8
 9
   struct ZKW_MinCostMaxFlow{
       int head[MAXN],tot;
10
11
       int cur[MAXN];
       int dis[MAXN];
12
```

```
13
       bool vis[MAXN];
14
       int ss,tt,N;//源点、汇点和点的总个数(编号是 0 \sim N-1),不需要额外赋值,
          调用会直接赋值
15
       int min_cost, max_flow;
       void init(){
16
17
           tot = 0;
           memset(head,-1,sizeof(head));
18
19
20
       void addedge(int u,int v,int cap,int cost){
21
            edge[tot] = Edge(v,head[u],cap,0,cost);
            head[u] = tot++;
22
            edge[tot] = Edge(u,head[v],0,0,-cost);
23
           head[v] = tot++;
24
25
       int aug(int u,int flow){
26
            if(u == tt)return flow;
27
            vis[u] = true;
28
            for(int i = cur[u];i != -1;i = edge[i].next){
29
                int v = edge[i].to;
30
                if(edge[i].cap > edge[i].flow && !vis[v] && dis[u] ==
31
                   dis[v] + edge[i].cost){
                    int tmp = aug(v,min(flow,edge[i].cap-edge[i].flow))
32
                    edge[i].flow += tmp;
33
                    edge[i^1].flow -= tmp;
34
                    cur[u] = i;
35
                    if(tmp)return tmp;
36
37
                }
38
            }
39
            return 0;
40
41
       bool modify_label(){
            int d = INF;
42
43
            for(int u = 0; u < N; u++)
44
                if(vis[u])
45
                    for(int i = head[u];i != -1;i = edge[i].next){
                        int v = edge[i].to;
46
                        if(edge[i].cap>edge[i].flow && !vis[v])
47
                            d = min(d,dis[v]+edge[i].cost-dis[u]);
48
                    }
49
            if(d == INF)return false;
50
51
            for(int i = 0;i < N;i++)</pre>
52
                if(vis[i]){
53
                    vis[i] = false;
                    dis[i] += d;
54
55
56
57
            return true;
58
       }
59
       /*
        * 直接调用获取最小费用和最大流
60
```

```
61
        * 输入: start-源点, end-汇点, n-点的总个数(编号从 0 开始)
        * 返回值: pair<int,int> 第一个是最小费用,第二个是最大流
62
63
        */
       pair<int,int> mincostmaxflow(int start,int end,int n){
64
           ss = start, tt = end, N = n;
65
           min_cost = max_flow = 0;
66
           for(int i = 0;i < n;i++)dis[i] = 0;</pre>
67
           while(1){
68
69
                for(int i = 0;i < n;i++)cur[i] = head[i];</pre>
               while(1){
70
                    for(int i = 0;i < n;i++)vis[i] = false;</pre>
71
                    int tmp = aug(ss,INF);
72
73
                    if(tmp == 0)break;
74
                    max_flow += tmp;
75
                    min_cost += tmp*dis[ss];
76
               if(!modify_label())break;
77
78
           }
79
           return make_pair(min_cost,max_flow);
80
81 | solve;
   4.18
        2	ext{-SAT}
   4.18.1 染色法(可以得到字典序最小的解)
   HDU 1814
 1 | const int MAXN = 20020;
   const int MAXM = 100010;
 3
   struct Edge
 4
   {
 5
       int to,next;
   }edge[MAXM];
   int head[MAXN],tot;
 7
   void init()
 9
   {
10
       tot = 0:
11
       memset(head,-1,sizeof(head));
12
13
   void addedge(int u,int v)
14
   {
15
       edge[tot].to = v;edge[tot].next = head[u];head[u] = tot++;
16
17
   bool vis[MAXN];//染色标记,为 true 表示选择
   |int S[MAXN],top;//栈
19
   bool dfs(int u)
20
   {
21
       if(vis[u^1])return false;
22
       if(vis[u])return true;
23
       vis[u] = true;
24
       S[top++] = u;
```

```
for(int i = head[u];i != -1;i = edge[i].next)
25
26
           if(!dfs(edge[i].to))
27
                return false;
28
       return true;
29
30
   bool Twosat(int n)
31
   {
32
       memset(vis, false, sizeof(vis));
33
       for(int i = 0; i < n; i += 2)
34
       {
           if(vis[i] || vis[i^1])continue;
35
           top = 0;
36
           if(!dfs(i))
37
38
           {
                while(top)vis[S[--top]] = false;
39
40
                if(!dfs(i^1)) return false;
41
42
       }
43
       return true;
44
   int main()
45
46
   {
47
       int n,m;
48
       int u, v;
49
       while(scanf("%d%d",&n,&m) == 2)
50
51
           init();
           while(m——)
52
53
           {
54
                scanf("%d%d",&u,&v);
55
                u--;v--;
56
                addedge(u, v^1);
                addedge(v,u^1);
57
58
           }
           if(Twosat(2*n))
59
60
           {
61
                for(int i = 0;i < 2*n;i++)
                    if(vis[i])
62
63
                       printf("%d\n",i+1);
64
           }
65
           else printf("NIE\n");
66
67
       return 0;
68 |}
          强连通缩点法(拓扑排序只能得到任意解)
   4.18.2
   POJ 3648 Wedding
 1 //***********
 2
   |//2-SAT 强连通缩点
 3 const int MAXN = 1010;
```

```
4 | const int MAXM = 100010;
 5
   struct Edge
 6
 7
       int to,next;
 8
   }edge[MAXM];
   int head[MAXN],tot;
   void init()
10
11
   {
12
       tot = 0;
13
       memset(head,-1,sizeof(head));
14
15
   void addedge(int u,int v)
16
17
       edge[tot].to = v; edge[tot].next = head[u]; head[u] = tot++;
18
   |int Low[MAXN],DFN[MAXN],Stack[MAXN],Belong[MAXN];//Belong 数组的值 1
      \sim scc
20 | int Index, top;
21
   int scc;
22 bool Instack[MAXN];
   int num[MAXN];
   void Tarjan(int u)
25
   {
26
       int ∨;
27
       Low[u] = DFN[u] = ++Index;
       Stack[top++] = u;
28
29
       Instack[u] = true;
       for(int i = head[u];i != -1;i = edge[i].next)
30
31
        {
32
            v = edge[i].to;
            if( !DFN[v] )
33
            {
34
35
                Tarjan(v);
36
                if(Low[u] > Low[v])Low[u] = Low[v];
37
38
            else if(Instack[v] && Low[u] > DFN[v])
39
                Low[u] = DFN[v];
40
       if(Low[u] == DFN[u])
41
42
43
            scc++;
44
            do
45
            {
46
                v = Stack[--top];
47
                Instack[v] = false;
48
                Belong[v] = scc;
49
                num[scc]++;
50
51
            while(v != u);
52
       }
53 |}
```

```
|bool solvable(int n)//n 是总个数, 需要选择一半
 55
    {
 56
        memset(DFN,0,sizeof(DFN));
        memset(Instack, false, sizeof(Instack));
 57
        memset(num,0,sizeof(num));
 58
 59
        Index = scc = top = 0;
        for(int i = 0;i < n;i++)</pre>
 60
            if(!DFN[i])
 61
 62
                Tarjan(i);
        for(int i = 0; i < n; i += 2)
 63
 64
            if(Belong[i] == Belong[i^1])
 65
 66
                return false;
 67
 68
        return true;
 69
 70
    //*************
71
    //拓扑排序求任意一组解部分
72
    queue<int>q1,q2;
73
    vector<vector<int> > dag;//缩点后的逆向 DAG 图
74
    |char color[MAXN];//染色,为'R' 是选择的
    int indeg[MAXN];//入度
76
    int cf[MAXN];
77
    void solve(int n)
78
 79
    {
 80
        dag.assign(scc+1,vector<int>());
 81
        memset(indeg,0,sizeof(indeg));
 82
        memset(color,0,sizeof(color));
        for(int u = 0; u < n; u++)
 83
            for(int i = head[u];i != -1;i = edge[i].next)
 84
 85
            {
                int v = edge[i].to;
 86
 87
                if(Belong[u] != Belong[v])
 88
                {
 89
                    dag[Belong[v]].push_back(Belong[u]);
90
                    indeg[Belong[u]]++;
 91
                }
92
        for(int i = 0;i < n;i += 2)
 93
 94
        {
95
            cf[Belong[i]] = Belong[i^1];
 96
            cf[Belong[i^1]] = Belong[i];
97
        while(!q1.empty())q1.pop();
98
        while(!q2.empty())q2.pop();
99
        for(int i = 1;i <= scc;i++)</pre>
100
            if(indeg[i] == 0)
101
102
                q1.push(i);
103
        while(!q1.empty())
104
```

```
105
             int u = q1.front();
106
             q1.pop();
             if(color[u] == 0)
107
             {
108
                 color[u] = 'R';
109
                 color[cf[u]] = 'B';
110
111
             int sz = dag[u].size();
112
113
             for(int i = 0;i < sz;i++)
114
             {
115
                 indeg[dag[u][i]]--;
                 if(indeg[dag[u][i]] == 0)
116
                      q1.push(dag[u][i]);
117
118
             }
         }
119
    }
120
121
122
    int change(char s[])
123
124
         int ret = 0;
125
         int i = 0;
126
         while(s[i]>='0' && s[i]<='9')</pre>
127
128
             ret *= 10;
129
             ret += s[i]-'0';
130
             j++;
131
         }
         if(s[i] == 'w')return 2*ret;
132
133
         else return 2*ret+1;
134
    }
135
    int main()
136
    {
137
         int n,m;
138
         char s1[10],s2[10];
139
         while(scanf("%d%d",&n,&m) == 2)
140
         {
141
             if(n == 0 && m == 0)break;
142
             init();
             while(m——)
143
144
                 scanf("%s%s",s1,s2);
145
146
                 int u = change(s1);
147
                 int v = change(s2);
                 addedge(u^1,v);
148
149
                 addedge(v^1,u);
150
             }
             addedge(1,0);
151
152
             if(solvable(2*n))
153
             {
154
                 solve(2*n);
155
                 for(int i = 1;i < n;i++)
```

```
156
                   {
157
                        //注意这一定是判断 color[Belong]
                       if(color[Belong[2*i]] == 'R')printf("%dw",i);
158
                       else printf("%dh",i);
159
                       if(i < n−1)printf("<sub>\(\)</sub>");
160
                       else printf("\n");
161
162
                   }
163
164
              else printf("bad<sub>□</sub>luck\n");
165
166
         return 0;
167 |}
```

### 4.19 曼哈顿最小生成树

POJ 3241 求曼哈顿最小生成树上第 k 大的边

```
1 | const int MAXN = 100010;
   const int INF = 0x3f3f3f3f;
 3
   struct Point{
        int x,y,id;
 4
   |}p[MAXN];
   bool cmp(Point a,Point b){
 7
        if(a.x != b.x) return a.x < b.x;
 8
        else return a.y < b.y;</pre>
 9
   //树状数组,找 y-x 大于当前的,但是 y+x 最小的
10
11
   struct BIT{
12
        int min_val,pos;
        void init()
13
14
        {
15
            min_val = INF;
            pos = -1;
16
17
        }
   |}bit[MAXN];
18
   //所有有效边
19
   struct Edge{
20
21
        int u,v,d;
22 | } edge [MAXN < < 2];
   bool cmpedge(Edge a, Edge b) {
        return a.d < b.d;</pre>
24
25
   }
26
   int tot;
27
   int n;
28
   int F[MAXN];
   int find(int x){
29
        if(F[x] == -1) return x;
30
31
        else return F[x] = find(F[x]);
32
33
   void addedge(int u,int v,int d){
34
        edge[tot].u = u;
35
        edge[tot].v = v;
```

```
36
        edge[tot++].d = d;
37
38
   int lowbit(int x){
39
        return x&(-x);
40
41
   void update(int i,int val,int pos){
42
        while(i > 0){
43
            if(val < bit[i].min_val){</pre>
44
                bit[i].min_val = val;
45
                bit[i].pos = pos;
46
            }
47
            i -= lowbit(i);
48
        }
49
   //查询 [i,m] 的最小值位置
50
51
   int ask(int i,int m){
        int min_val = INF,pos = -1;
52
53
        while(i <= m){
            if(bit[i].min val < min val){</pre>
54
55
                min_val = bit[i].min_val;
                pos = bit[i].pos;
56
57
            }
58
            i += lowbit(i);
59
60
        return pos;
61
62
   int dist(Point a,Point b){
63
        return abs(a.x - b.x) + abs(a.y - b.y);
64
   }
65
   void Manhattan_minimum_spanning_tree(int n,Point p[]){
        int a[MAXN],b[MAXN];
66
67
        tot = 0;
        for(int dir = 0; dir < 4;dir++){</pre>
68
69
            //4 种坐标变换
70
            if(dir == 1 || dir == 3){
71
                for(int i = 0;i < n;i++)
72
                     swap(p[i].x,p[i].y);
73
74
            else if(dir == 2){
                for(int i = 0; i < n; i++)
75
76
                     p[i].x = -p[i].x;
77
78
            sort(p,p+n,cmp);
79
            for(int i = 0;i < n;i++)</pre>
                a[i] = b[i] = p[i].y - p[i].x;
80
            sort(b,b+n);
81
            int m = unique(b,b+n) - b;
82
            for(int i = 1;i <= m;i++)</pre>
83
                bit[i].init();
84
85
            for(int i = n-1; i >= 0; i--){
                int pos = lower_bound(b,b+m,a[i]) - b + 1;
86
```

```
87
                 int ans = ask(pos,m);
                 if(ans !=-1)
 88
89
                     addedge(p[i].id,p[ans].id,dist(p[i],p[ans]));
90
                 update(pos,p[i].x+p[i].y,i);
91
             }
 92
        }
 93
    int solve(int k){
 94
95
        Manhattan_minimum_spanning_tree(n,p);
        memset(F,-1,sizeof(F));
96
        sort(edge,edge+tot,cmpedge);
97
        for(int i = 0;i < tot;i++){
98
             int u = edge[i].u;
99
             int v = edge[i].v;
100
             int t1 = find(u), t2 = find(v);
101
             if(t1 != t2){
102
103
                 F[t1] = t2;
104
                 k---;
105
                 if(k == 0)return edge[i].d;
106
             }
        }
107
108
    int main()
109
110
    {
111
        //freopen("in.txt","r",stdin);
        //freopen("out.txt","w",stdout);
112
        int k;
113
        while(scanf("%d%d",&n,&k)==2 && n){
114
115
             for(int i = 0;i < n;i++){</pre>
116
                 scanf("%d%d",&p[i].x,&p[i].y);
117
                 p[i].id = i;
118
             }
             printf("%d\n",solve(n-k));
119
120
        }
121
        return 0;
122 |}
    4.20
          LCA
    4.20.1 dfs+ST 在线算法
  1
   /*
  2
     * LCA (POJ 1330)
     * 在线算法 DFS + ST
  4
     */
  5
    const int MAXN = 10010;
    |int rmq[2*MAXN];//rmq 数组,就是欧拉序列对应的深度序列
  7
    struct ST{
  8
        int mm[2*MAXN];
  9
        int dp[2*MAXN][20];//最小值对应的下标
        void init(int n){
 10
```

```
mm[0] = -1;
11
            for(int i = 1;i <= n;i++){</pre>
12
13
                mm[i] = ((i&(i-1)) == 0)?mm[i-1]+1:mm[i-1];
14
                dp[i][0] = i;
15
            for(int j = 1; j <= mm[n];j++)
16
                for(int i = 1; i + (1 << j) - 1 <= n; i++)
17
                    dp[i][j] = rmq[dp[i][j-1]] < rmq[dp[i+(1<<(j-1))][j]
18
                       -1]]?dp[i][j-1]:dp[i+(1<<(j-1))][j-1];
19
       }
       //查询 [a,b] 之间最小值的下标
20
21
       int query(int a,int b)
22
       {
23
            if(a > b)swap(a,b);
            int k = mm[b-a+1];
24
25
            return rmq[dp[a][k]] <= rmq[dp[b-(1<<k)+1][k]]?dp[a][k]:dp[
              b-(1<< k)+1\rceil\lceil k\rceil;
26
       }
27
   };
28
   //边的结构体定义
29
   struct Edge{
30
       int to,next;
   };
31
   Edge edge[MAXN*2];
32
33
   int tot,head[MAXN];
34
   |int F[MAXN*2];//欧拉序列,就是 dfs 遍历的顺序,长度为 2*n-1,下标从 1 开始
   int P[MAXN];//P[i] 表示点 i 在 F 中第一次出现的位置
36
37
   int cnt;
38
   ST st;
39
   void init(){
40
       tot = 0;
41
       memset(head,-1,sizeof(head));
42
43
   //加边,无向边需要加两次
   void addedge(int u,int v){
       edge[tot].to = v;
45
       edge[tot].next = head[u];
46
47
       head[u] = tot++;
48
49
   void dfs(int u,int pre,int dep){
50
       F[++cnt] = u;
51
       rmq[cnt] = dep;
52
       P[u] = cnt;
       for(int i = head[u]; i != -1; i = edge[i].next){
53
            int v = edge[i].to;
54
            if(v == pre)continue;
55
56
            dfs(v,u,dep+1);
            F[++cnt] = u;
57
58
            rmq[cnt] = dep;
59
       }
```

```
60
   }
61
   //查询 LCA 前的初始化
   void LCA_init(int root,int node_num){
62
63
       cnt = 0;
       dfs(root,root,0);
64
65
       st.init(2*node_num-1);
66
   //查询 u,v 的 lca 编号
67
   int query_lca(int u,int v){
68
69
       return F[st.query(P[u],P[v])];
70
71
   bool flag[MAXN];
   int main(){
72
       int T;
73
       int N;
74
75
       int u,v;
       scanf("%d",&T);
76
77
       while (T--)
            scanf("%d",&N);
78
79
            init();
            memset(flag, false, sizeof(flag));
80
81
            for(int i = 1; i < N;i++){</pre>
82
                scanf("%d%d",&u,&v);
                addedge(u,v);
83
84
                addedge(v,u);
                flag[v] = true;
85
            }
86
            int root;
87
88
            for(int i = 1; i <= N;i++)
                if(!flag[i]){
89
90
                    root = i;
91
                    break;
                }
92
93
            LCA_init(root,N);
            scanf("%d%d",&u,&v);
94
95
            printf("%d\n",query_lca(u,v));
96
       }
97
       return 0;
  }
98
   4.20.2 离线 Tarjan 算法
 1 |/*
 2
    * POJ 1470
    * 给出一颗有向树, Q 个查询
 4
    * 输出查询结果中每个点出现次数
 5
    */
 6
   /*
 7
    * 离线算法, LCATarjan
    * 复杂度O(n+Q);
 8
 9
    */
10 | const int MAXN = 1010;
```

```
11 const int MAXQ = 500010; //查询数的最大值
12
13
   //并查集部分
   |int F[MAXN];//需要初始化为 -1
14
15
   int find(int x){
16
       if(F[x] == -1)return x;
17
       return F[x] = find(F[x]);
18
19
   void bing(int u,int v){
20
       int t1 = find(u);
       int t2 = find(v);
21
22
       if(t1 != t2)
23
           F[t1] = t2;
24
   //*******
26 | bool vis[MAXN];//访问标记
   |int ancestor[MAXN];//祖先
27
28
   struct Edge{
29
       int to,next;
   }edge[MAXN*2];
30
   int head[MAXN],tot;
31
32
   void addedge(int u,int v){
33
       edge[tot].to = v;
       edge[tot].next = head[u];
34
35
       head[u] = tot++;
   }
36
37
38
   struct Query{
39
       int q,next;
40
       int index;//查询编号
41
   |}query[MAXQ*2];
   |int answer[MAXQ];//存储最后的查询结果,下标 0 Q-1
   int h[MAXQ];
43
44
   int tt;
45
   int Q;
46
47
   void add_query(int u,int v,int index){
48
       query[tt].q = v;
49
       query[tt].next = h[u];
50
       query[tt].index = index;
51
       h[u] = tt++;
52
       query[tt].q = u;
53
       query[tt].next = h[v];
54
       query[tt].index = index;
55
       h[v] = tt++;
   }
56
57
58
   void init(){
59
       tot = 0;
60
       memset(head,-1,sizeof(head));
61
       tt = 0;
```

```
62
         memset(h,-1,sizeof(h));
 63
         memset(vis, false, sizeof(vis));
 64
         memset(F,-1,sizeof(F));
 65
         memset(ancestor,0,sizeof(ancestor));
 66
 67
    void LCA(int u){
 68
         ancestor[u] = u;
 69
         vis[u] = true;
 70
         for(int i = head[u];i != −1;i = edge[i].next){
 71
             int v = edge[i].to;
 72
             if(vis[v])continue;
 73
             LCA(v);
 74
             bing(u,v);
 75
             ancestor[find(u)] = u;
 76
 77
         for(int i = h[u]; i != -1; i = query[i].next){
 78
             int v = query[i].q;
 79
             if(vis[v]){
 80
                  answer[query[i].index] = ancestor[find(v)];
 81
             }
         }
 82
 83
 84
    bool flag[MAXN];
    int Count_num[MAXN];
 85
 86
    int main(){
 87
         int n;
         int u,v,k;
 88
 89
         while(scanf("%d",&n) == 1){
 90
             init();
 91
             memset(flag, false, sizeof(flag));
             for(int i = 1;i <= n;i++){</pre>
 92
                  scanf("%d:(%d)",&u,&k);
 93
 94
                  while(k--){
 95
                      scanf("%d",&v);
 96
                      flag[v] = true;
 97
                      addedge(u,v);
 98
                      addedge(v,u);
 99
                  }
100
             }
             scanf("%d",&Q);
101
             for(int i = 0;i < Q;i++){
102
103
                  char ch;
104
                  cin>>ch;
                  scanf("%d<sub>\(\)</sub>%d)",&u,&v);
105
106
                  add_query(u,v,i);
107
             int root;
108
             for(int i = 1;i <= n;i++)</pre>
109
                  if(!flag[i]){
110
111
                      root = i;
112
                      break;
```

```
113
                 }
114
             LCA(root);
             memset(Count_num,0,sizeof(Count_num));
115
             for(int i = 0;i < Q;i++)
116
                 Count_num[answer[i]]++;
117
             for(int i = 1;i <= n;i++)
118
                 if(Count_num[i] > 0)
119
                     printf("%d:%d\n",i,Count_num[i]);
120
121
        }
122
        return 0;
123 |}
    4.20.3 LCA 倍增法
  1
  2
     * POJ 1330
     * LCA 在线算法
    */
  5
    const int MAXN = 10010;
    const int DEG = 20;
  7
  8
    struct Edge{
  9
        int to,next;
 10
    }edge[MAXN*2];
 11
   int head[MAXN],tot;
 12
    void addedge(int u,int v){
 13
        edge[tot].to = v;
 14
        edge[tot].next = head[u];
 15
        head[u] = tot++;
 16
 17
    void init(){
 18
        tot = 0;
 19
        memset(head,-1,sizeof(head));
20
 21
    |int fa[MAXN][DEG];//fa[i][j] 表示结点 i 的第2^j个祖先
 22
    int deg[MAXN];//深度数组
23
    void BFS(int root){
 24
 25
        queue<int>que;
        deg[root] = 0;
 26
 27
        fa[root][0] = root;
        que.push(root);
 28
 29
        while(!que.empty()){
             int tmp = que.front();
 30
 31
             que.pop();
 32
             for(int i = 1;i < DEG;i++)</pre>
 33
                 fa[tmp][i] = fa[fa[tmp][i-1]][i-1];
             for(int i = head[tmp]; i != -1; i = edge[i].next){
 34
 35
                 int v = edge[i].to;
                 if(v == fa[tmp][0])continue;
 36
                 deg[v] = deg[tmp] + 1;
 37
 38
                 fa[v][0] = tmp;
```

```
39
                que.push(v);
            }
40
41
42
        }
43
   int LCA(int u,int v){
44
45
        if(deg[u] > deg[v])swap(u,v);
46
        int hu = deg[u], hv = deg[v];
47
        int tu = u, tv = v;
48
        for(int det = hv-hu, i = 0; det ;det>>=1, i++)
49
            if(det&1)
                tv = fa[tv][i];
50
        if(tu == tv)return tu;
51
52
        for(int i = DEG-1; i >= 0; i--){
53
            if(fa[tu][i] == fa[tv][i])
54
                continue;
            tu = fa[tu][i];
55
56
            tv = fa[tv][i];
        }
57
58
        return fa[tu][0];
59
60
   bool flag[MAXN];
61
   int main(){
62
        int T;
63
        int n;
64
        int u,v;
        scanf("%d",&T);
65
66
        while(T--){
67
            scanf("%d",&n);
68
            init();
            memset(flag, false, sizeof(flag));
69
70
            for(int i = 1;i < n;i++){</pre>
71
                 scanf("%d%d",&u,&v);
72
                addedge(u,v);
73
                addedge(v,u);
74
                flag[v] = true;
75
            }
            int root;
76
77
            for(int i = 1;i <= n;i++)
78
                if(!flag[i]){
79
                     root = i;
80
                     break;
81
                }
82
            BFS(root);
83
            scanf("%d%d",&u,&v);
84
            printf("%d\n",LCA(u,v));
85
        }
86
        return 0;
87 |}
```

## 4.21 欧拉路

欧拉回路:每条边只经过一次,而且回到起点欧拉路径:每条边只经过一次,不要求回到起点

#### 欧拉回路判断:

无向图:连通(不考虑度为0的点),每个顶点度数都为偶数。

有向图:基图连通(把边当成无向边,同样不考虑度为 0 的点),每个顶点出度等于入度。 混合图(有无向边和有向边):首先是基图连通(不考虑度为 0 的点),然后需要借助网络流

首先给原图中的每条无向边随便指定一个方向(称为初始定向),将原图改为有向图 G', 然后的任务就是改变 G'中某些边的方向(当然是无向边转化来的,原混合图中的有向边不能动)使其满足每个点的入度等于出度。

设 D[i] 为 G' 中 (点 i 的出度 - 点 i 的入度 )。可以发现,在改变 G' 中边的方向的过程中,任何点的 D 值的奇偶性都不会发生改变(设将边 < i, j > 改为 < j, i >,则 i 入度加 1 出度减 1, j 入度减 1 出度加 1,两者之差加 2 或减 2,奇偶性不变 i 是。而最终要求的是每个点的入度等于出度,即每个点的 i 位都为 i 0,是偶数,故可得:若初始定向得到的 i i 中任意一个点的 i 位是奇数,那么原图中一定不存在欧拉环!

若初始 D 值都是偶数,则将 G' 改装成网络:设立源点 S 和汇点 T,对于每个 D[i]>0 的点 i,连边  $\langle S, i \rangle$ ,容量为 D[i]/2;对于每个 D[j]<0 的点 j,连边  $\langle j, T \rangle$ ,容量为 D[j]/2; G' 中的每条边在网络中仍保留,容量为 1(表示该边最多只能被改变方向一次)。求这个网络的最大流,若 S 引出的所有边均满流,则原混合图是欧拉图,将网络中所有流量为 1 的中间边(就是不与 S 或 T 关联的边)在 G' 中改变方向,形成的新图 G'' 一定是有向欧拉图;若 S 引出的边中有的没有满流,则原混合图不是欧拉图。

#### 欧拉路径的判断:

无向图:连通(不考虑度为 0 的点),每个顶点度数都为偶数或者仅有两个点的度数为偶数。有向图:基图连通(把边当成无向边,同样不考虑度为 0 的点),每个顶点出度等于入度或者有且仅有一个点的出度比入度多 1,有且仅有一个点的出度比入度少 1,其余出度等于入度。

混合图:如果存在欧拉回路,一点存在欧拉路径了。否则如果有且仅有两个点的(出度 -入度)是奇数,那么给这个两个点加边,判断是否存在欧拉回路。

#### 4.21.1 有向图

POJ 2337

给出 n 个小写字母组成的单词,要求将 n 个单词连接起来,使得前一个单词的最后一个字母和后一个单词的第一个字母相同。输出字典序最小的解。

```
struct Edge{
 1
 2
       int to,next;
 3
       int index;
       bool flag;
 4
 5
   }edge[2010];
   int head[30],tot;
 7
   void init(){
 8
       tot = 0;
 9
       memset(head,-1,sizeof(head));
10
   void addedge(int u,int v,int index){
11
12
       edge[tot].to = v;
```

```
13
       edge[tot].next = head[u];
14
       edge[tot].index = index;
15
       edge[tot].flag = false;
16
       head[u] = tot++;
17
18
   string str[1010];
   int in[30],out[30];
19
   int cnt;
21
   int ans[1010];
   void dfs(int u){
22
       for(int i = head[u]; i != -1; i = edge[i].next)
23
24
            if(!edge[i].flag)
25
            {
26
                edge[i].flag = true;
                dfs(edge[i].to);
27
                ans[cnt++] = edge[i].index;
28
29
            }
30
   int main(){
31
32
       int T,n;
       scanf("%d",&T);
33
34
       while (T--)
            scanf("%d",&n);
35
            for(int i = 0;i < n;i++)</pre>
36
37
                cin>>str[i];
            sort(str,str+n);//要输出字典序最小的解,先按照字典序排序
38
            init();
39
            memset(in,0,sizeof(in));
40
41
            memset(out,0,sizeof(out));
42
            int start = 100;
            for(int i = n-1;i >= 0;i--)//字典序大的先加入
43
44
            {
45
                int u = str[i][0] - 'a';
46
                int v = str[i][str[i].length() - 1] - 'a';
47
                addedge(u,v,i);
48
                out[u]++;
49
                in[v]++;
                if(u < start)start = u;</pre>
50
                if(v < start)start = v;</pre>
51
52
53
            int cc1 = 0, cc2 = 0;
            for(int i = 0;i < 26;i++){</pre>
54
55
                if(out[i] - in[i] == 1){
56
                    cc1++;
                    start = i; // 如果有一个出度比入度大 1 的点,就从这个点出发,
57
                       否则从最小的点出发
58
59
                else if(out[i] - in[i] == -1)
60
                    cc2++;
61
                else if(out[i] - in[i] != 0)
62
                    cc1 = 3;
```

```
}
63
64
            if(! ( (cc1 == 0 && cc2 == 0) || (cc1 == 1 && cc2 == 1) )){
                printf("***\n");
65
66
                continue;
            }
67
            cnt = 0;
68
            dfs(start);
69
            if(cnt != n)//判断是否连通
70
71
                printf("***\n");
72
                continue;
73
74
            for(int i = cnt-1; i >= 0;i--){
75
                cout<<str[ans[i]];</pre>
76
                if(i > 0)printf(".");
77
                else printf("\n");
78
79
            }
80
        }
81
        return 0;
82 |}
   4.21.2 无向图
   SGU 101
   struct Edge{
 1
 2
        int to,next;
 3
        int index;
 4
        int dir;
 5
        bool flag;
   }edge[220];
 7
   int head[10],tot;
 8
   void init(){
 9
        memset(head,-1,sizeof(head));
10
        tot = 0;
11
12
   void addedge(int u,int v,int index){
13
        edge[tot].to = v;
14
        edge[tot].next = head[u];
15
        edge[tot].index = index;
        edge[tot].dir = 0;
16
17
        edge[tot].flag = false;
18
        head[u] = tot++;
19
        edge[tot].to = u;
20
        edge[tot].next = head[v];
        edge[tot].index = index;
21
22
        edge[tot].dir = 1;
23
        edge[tot].flag = false;
24
        head[v] = tot++;
25
26
   int du[10];
27
   vector<int>ans;
```

```
28
   void dfs(int u){
        for(int i = head[u]; i != -1;i = edge[i].next)
29
30
             if(!edge[i].flag ){
                 edge[i].flag = true;
31
                 edge[i^1].flag = true;
32
                 dfs(edge[i].to);
33
34
                 ans.push_back(i);
35
             }
36
   }
37
   int main(){
        int n;
38
39
        while(scanf("%d",&n) == 1){
40
             init();
41
             int u,v;
            memset(du,0,sizeof(du));
42
             for(int i = 1;i <= n;i++){
43
                 scanf("%d%d",&u,&v);
44
45
                 addedge(u,v,i);
46
                 du[u]++;
47
                 du[v]++;
48
             }
49
             int s = -1;
50
             int cnt = 0;
             for(int i = 0;i <= 6;i++){
51
52
                 if(du[i]&1) {cnt++; s = i;}
53
                 if(du[i] > 0 \&\& s == -1)
54
                      s = i;
55
            bool ff = true;
56
57
             if(cnt != 0 && cnt != 2){
                 printf("No<sub>□</sub>solution\n");
58
59
                 continue;
             }
60
            ans.clear();
61
            dfs(s);
62
63
             if(ans.size() != n){
64
                 printf("No<sub>□</sub>solution\n");
                 continue;
65
66
             for(int i = 0;i < ans.size();i++){</pre>
67
                 printf("%d<sub>\upsi</sub>",edge[ans[i]].index);
68
69
                 if(edge[ans[i]].dir == 0)printf("-\n");
70
                 else printf("+\n");
71
             }
72
        }
73
        return 0;
74 }
```

### 4.21.3 混合图

POJ 1637 (本题保证了连通,故不需要判断连通,否则要判断连通)

```
1 const int MAXN = 210;
 2
   //最大流部分ISAP
   const int MAXM = 20100;
   const int INF = 0x3f3f3f3f;
   struct Edge
 6
   {
 7
       int to,next,cap,flow;
   }edge[MAXM];
 8
 9
   int tol;
   int head[MAXN];
10
   int gap[MAXN],dep[MAXN],pre[MAXN],cur[MAXN];
11
   void init()
12
13
   {
       tol = 0;
14
15
       memset(head,-1,sizeof(head));
16
   void addedge(int u,int v,int w,int rw = 0)
17
18
   {
19
       edge[tol].to = v;
20
       edge[tol].cap = w;
21
       edge[tol].next = head[u];
22
       edge[tol].flow = 0;
23
       head[u] = tol++;
       edge[tol].to = u;
24
25
       edge[tol].cap = rw;
26
       edge[tol].next = head[v];
27
       edge[tol].flow = 0;
28
       head[v] = tol++;
29
30
   int sap(int start,int end,int N)
31
32
       memset(gap,0,sizeof(gap));
33
       memset(dep,0,sizeof(dep));
34
       memcpy(cur,head,sizeof(head));
35
       int u = start;
36
       pre[u] = -1;
37
       gap[0] = N;
38
       int ans = 0;
39
       while(dep[start] < N)</pre>
40
41
            if(u == end)
42
            {
43
                int Min = INF;
44
                for(int i = pre[u]; i != -1;i = pre[edge[i^1].to])
                    if(Min > edge[i].cap - edge[i].flow)
45
46
                        Min = edge[i].cap - edge[i].flow;
                for(int i = pre[u];i != -1;i = pre[edge[i^1].to])
47
48
49
                    edge[i].flow += Min;
50
                    edge[i^1].flow -= Min;
51
                }
```

```
52
                 u = start;
 53
                 ans += Min;
 54
                 continue;
 55
             }
             bool flag = false;
 56
 57
             int ∨;
             for(int i = cur[u];i != -1;i = edge[i].next)
 58
 59
             {
 60
                 v = edge[i].to;
                 if(edge[i].cap - edge[i].flow && dep[v] + 1 == dep[u])
 61
 62
                      flag = true;
 63
                      cur[u] = pre[v] = i;
 64
 65
                     break;
                 }
 66
 67
             }
             if(flag)
 68
 69
             {
 70
                 u = v;
 71
                 continue;
 72
 73
             int Min = N;
             for(int i = head[u];i != −1;i = edge[i].next)
 74
                 if(edge[i].cap - edge[i].flow && dep[edge[i].to] < Min)</pre>
 75
 76
                 {
 77
                      Min = dep[edge[i].to];
 78
                      cur[u] = i;
 79
 80
             gap[dep[u]]--;
 81
             if(!gap[dep[u]])return ans;
             dep[u] = Min+1;
 82
 83
             gap[dep[u]]++;
             if(u != start)u = edge[pre[u]^1].to;
 84
 85
         }
 86
         return ans;
 87
 88
    //the end of 最大流部分
 89
 90
    int in[MAXN],out[MAXN];//每个点的出度和入度
 91
 92
    int main()
 93
    {
 94
         //freopen("in.txt","r",stdin);
         //freopen("out.txt","w",stdout);
 95
 96
         int T;
         int n,m;
 97
         scanf("%d",&T);
 98
         while(T--)
 99
100
         {
101
             scanf("%d%d",&n,&m);
102
             init();
```

```
int u, v, w;
103
             memset(in,0,sizeof(in));
104
             memset(out,0,sizeof(out));
105
             while(m--)
106
107
             {
                 scanf("%d%d%d",&u,&v,&w);
108
                 out[u]++; in[v]++;
109
                 if(w == 0)//双向
110
111
                      addedge(u, v, 1);
112
             bool flag = true;
113
             for(int i = 1;i <= n;i++)
114
115
             {
                 if(out[i] - in[i] > 0)
116
                      addedge(0,i,(out[i] - in[i])/2);
117
                 else if(in[i] - out[i] > 0)
118
                      addedge(i,n+1,(in[i] - out[i])/2);
119
120
                 if((out[i] - in[i]) & 1)
                      flag = false;
121
122
             if(!flag)
123
124
125
                 printf("impossible\n");
                 continue;
126
127
             sap(0,n+1,n+2);
128
             for(int i = head[0]; i != -1; i = edge[i].next)
129
                 if(edge[i].cap > 0 && edge[i].cap > edge[i].flow)
130
131
132
                      flag = false;
133
                     break;
134
             if(flag)printf("possible\n");
135
136
             else printf("impossible\n");
137
         }
138
         return 0;
139 |}
```

## 4.22 树分治

### 4.22.1 点分治 -HDU5016

HDU 5016 给定一个边权树,初始时,一些结点上已经建立了市场。每个结点会被距离自己最近的市场所支配(距离相同时,被标号最小的市场支配)可以新建一个市场,问新建的市场最多可以支配多少点。

```
const int MAXN = 100010;
const int INF = 0x3f3f3f3f;
struct Edge{
   int to,next,w;
}edge[MAXN*2];
int head[MAXN],tot;
```

```
7
   void init(){
 8
       tot = 0;
 9
       memset(head,-1,sizeof(head));
10
11
   |inline void addedge(int u,int v,int w){
12
       edge[tot].to = v;
13
       edge[tot].w = w;
       edge[tot].next = head[u];
14
15
       head[u] = tot++;
16
17
   | int size[MAXN],vis[MAXN],fa[MAXN],que[MAXN];
   |int TT;//时间戳
18
   //找重心
19
20
   inline int getroot(int u){
       int Min = MAXN, root = 0;
21
       int l,r;
22
       que[l = r = 1] = u;
23
       fa[u] = 0;
24
25
       for(;l <= r;l++)
            for(int i = head[que[l]]; i != -1;i = edge[i].next){
26
                int v = edge[i].to;
27
28
                if(v == fa[que[l]] || vis[v] == TT)continue;
29
                que[++r] = v;
                fa[v] = que[l];
30
31
32
       for(l---;l;l---){
33
            int x = que[l], Max = 0;
34
            size[x] = 1;
35
            for(int i = head[x];i != -1;i = edge[i].next){
36
                int v = edge[i].to;
                if(v == fa[x] || vis[v] == TT)continue;
37
                Max = max(Max, size[v]);
38
39
                size[x] += size[v];
40
            }
41
            Max = max(Max,r - size[x]);
42
            if(Max < Min){</pre>
43
                Min = Max; root = x;
44
            }
45
       }
46
       return root;
47
   }
48
49
   |int ans[MAXN];
50
   pair<int,int>pp[MAXN];
   pair<int,int>np[MAXN];
51
   int dis[MAXN];
52
   int type[MAXN];
53
   inline void go(int u,int pre,int w,int tt){
54
55
       int l,r;
56
       que[l = r = 1] = u;
       fa[u] = pre; dis[u] = w;
57
```

```
58
        for(;l <= r;l++)
             for(int i = head[que[l]]; i != -1; i = edge[i].next){
 59
                 int v = edge[i].to;
 60
                 if(v == fa[que[l]] || vis[v] == TT)continue;
 61
                 que[++r] = v;
 62
 63
                 fa[v] = que[l];
                 dis[v] = dis[que[l]] + edge[i].w;
 64
 65
             }
        int cnt = 0;
 66
 67
        for(int i = 1;i <= r;i++){
 68
             int x = que[i];
 69
             pp[cnt++] = make_pair(np[x].first-dis[x],np[x].second);
 70
 71
        sort(pp,pp+cnt);
        for(int i = 1;i <= r;i++){
 72
             int x = que[i];
 73
             if(type[x])continue;
 74
 75
             int id = lower_bound(pp,pp+cnt,make_pair(dis[x],x)) - pp;
 76
             ans[x] += (cnt-id)*tt;
        }
 77
 78
 79
    void solve(int u){
 80
        int root = getroot(u);
 81
        vis[root] = TT;
 82
        go(root,0,0,1);
 83
        for(int i = head[root];i != -1;i = edge[i].next){
             int v = edge[i].to;
 84
 85
             if(vis[v] == TT)continue;
 86
             go(v,root,edge[i].w,-1);
 87
        }
        for(int i = head[root];i != -1;i = edge[i].next){
 88
 89
             int v = edge[i].to;
 90
             if(vis[v] == TT)continue;
 91
             solve(v);
 92
        }
 93
 94
    bool ff[MAXN];
    int main()
 95
 96
    {
 97
        int n;
        memset(vis,0,sizeof(vis));
 98
 99
        TT = 0;
100
        while(scanf("%d",&n) == 1){
101
             init();
             int u,v,w;
102
             for(int i = 1;i < n;i++){
103
                 scanf("%d%d%d",&u,&v,&w);
104
                 addedge(u, v, w);
105
106
                 addedge(v,u,w);
107
             for(int i = 1;i <= n;i++)scanf("%d",&type[i]);</pre>
108
```

```
109
            queue<int>q;
             for(int i = 1;i <= n;i++){</pre>
110
                 if(type[i]){
111
                     np[i] = make_pair(0,i);
112
                     ff[i] = true;
113
                     q.push(i);
114
                 }
115
116
                 else{
117
                     np[i] = make_pair(INF,0);
                     ff[i] = false;
118
119
                 }
120
             }
            while(!q.empty()){
121
                 u = q.front();
122
                 q.pop();
123
                 ff[u] = false;
124
                 for(int i = head[u];i != −1;i = edge[i].next){
125
126
                     v = edge[i].to;
                     pair<int,int>tmp = make_pair(np[u].first+edge[i].w,
127
                        np[u].second);
                     if(tmp < np[v]){
128
129
                         np[v] = tmp;
130
                         if(!ff[v]){
                             ff[v] = true;
131
                             q.push(v);
132
                         }
133
                     }
134
135
                 }
136
             }
137
            TT++;
             for(int i = 1;i <= n;i++)ans[i] = 0;</pre>
138
             solve(1);
139
             int ret = 0;
140
             for(int i = 1;i <= n;i++)ret = max(ret,ans[i]);</pre>
141
             printf("%d\n",ret);
142
143
        }
144
        return 0;
145 |}
    4.22.2 * 点分治 -HDU4918
    HDU 4918
    题意:给出一颗 n 个点的树,每个点有一个权值,有两种操作,一种是将某个点的权值修改
    为 v, 另一种是查询距离点 u 不超过 d 的点的权值和。
  1 | const int MAXN = 100010;
  2 const int MAXD = 40;
  3 int cc[MAXN*MAXD];
  4 | int *cc_tail; //记得初始化 cc_tail = cc
    //0-based BinaryIndexTree
    struct BIT{
  6
  7
        int *c;
```

```
8
        int n;
 9
        void init(int _n){
10
            n = _n;
11
            c = cc_tail;
12
            cc_tail = cc_tail + n;
13
            memset(c,0,sizeof(int)*n);
14
        }
        void add(int i,int val){
15
16
            while(i < n){
17
                c[i] += val;
                i += ~i & i+1;
18
19
            }
20
        }
        int sum(int i){
21
            i = min(i, n-1);
22
23
            int s = 0;
            while(i >= 0){
24
25
                s += c[i];
                i -= ~i & i+1;
26
27
            }
28
            return s;
29
        }
30
   }bits[MAXN<<1];</pre>
   struct Edge{
31
32
        int to,next;
33
   }edge[MAXN*2];
34
   int head[MAXN],tot;
35
   void init(){
36
        tot = 0;
37
        memset(head,-1,sizeof(head));
38
39
   inline void addedge(int u,int v){
40
        edge[tot].to = v;
        edge[tot].next = head[u];
41
42
        head[u] = tot++;
43
44
   int size[MAXN],vis[MAXN],fa[MAXN],que[MAXN];
45
   int TT;
46
   inline int getroot(int u,int &tot){
47
        int Min = MAXN, root = 0;
        int l,r;
48
        que[l = r = 1] = u;
49
50
        fa[u] = 0;
51
        for(;l <= r;l++)
52
            for(int i = head[que[l]]; i != -1; i = edge[i].next){
                int v = edge[i].to;
53
                if(v == fa[que[l]] || vis[v] == TT)continue;
54
55
                que[++r] = v;
                fa[v] = que[l];
56
57
        tot = r;
58
```

```
59
         for(l--;l;l--){
 60
             int x = que[l], Max = 0;
             size[x] = 1;
 61
             for(int i = head[x]; i != -1; i = edge[i].next){
 62
                 int v = edge[i].to;
 63
                 if(v == fa[x] || vis[v] == TT)continue;
 64
                 Max = max(Max,size[v]);
 65
                 size[x] += size[v];
 66
 67
             }
             Max = max(Max,r - size[x]);
 68
 69
             if(Max < Min){</pre>
 70
                 Min = Max, root = x;
 71
             }
 72
         }
 73
         return root;
 74
    }
 75
    struct Node{
 76
         int root, subtree, dis;
 77
         Node(int _root = 0, int _subtree = 0,int _dis = 0){
 78
             root = _root;
 79
             subtree = _subtree;
 80
             dis = _dis;
 81
         }
 82
    };
 83
    vector<Node>vec[MAXN];
    |int id[MAXN];
    |int dist[MAXN];
    int val[MAXN];
 86
 87
    int cnt;
 88
    inline void go(int u,int root,int subtree){
 89
         int l,r;
 90
         que[l = r = 1] = u;
 91
         fa[u] = 0; dist[u] = 1;
 92
         for(; l <= r;l++){
 93
             u = que[l];
             vec[u].push_back(Node(id[root], subtree, dist[u]));
 94
             for(int i = head[u];i != −1;i = edge[i].next){
 95
                 int v = edge[i].to;
 96
 97
                 if(v == fa[u] || vis[v] == TT)continue;
 98
                 que[++r] = v;
 99
                 fa[v] = u;
                 dist[v] = dist[u] + 1;
100
101
             }
         }
102
         bits[subtree].init(r+1);
103
         for(int i = 1;i <= r;i++){
104
             u = que[i];
105
             bits[id[root]].add(dist[u],val[u]);
106
             bits[subtree].add(dist[u],val[u]);
107
108
         }
109 |}
```

```
110
    void solve(int u){
111
         int tot;
112
         int root = getroot(u,tot);
         vis[root] = TT;
113
         id[root] = cnt++;
114
         vec[root].push_back(Node(id[root],-1,0));
115
         bits[id[root]].init(tot);
116
         bits[id[root]].add(0,val[root]);
117
118
         for(int i = head[root];i != -1;i = edge[i].next){
             int v = edge[i].to;
119
             if(vis[v] == TT)continue;
120
             go(v,root,cnt);
121
             cnt++;
122
123
         for(int i = head[root];i != -1;i = edge[i].next){
124
             int v = edge[i].to;
125
             if(vis[v] == TT)continue;
126
127
             solve(v);
         }
128
129
130
    int main(){
131
         int n,m;
132
         memset(vis,0,sizeof(vis));
133
         TT = 0;
         while(scanf("%d%d",&n,&m) == 2){
134
135
             init();
136
             TT++;
             cc_tail = cc;
137
138
             cnt = 0;
139
             for(int i = 1;i <= n;i++)vec[i].clear();</pre>
             for(int i = 1;i <= n;i++)scanf("%d",&val[i]);</pre>
140
             int u,v;
141
             for(int i = 1;i < n;i++){</pre>
142
143
                 scanf("%d%d",&u,&v);
                 addedge(u,v);
144
145
                 addedge(v,u);
146
             }
147
             solve(1);
             char op[10];
148
             int d;
149
             while(m——){
150
                 scanf("%s%d%d",op,&u,&d);
151
                 if(op[0] == '!'){
152
                      int dv = d - val[u];
153
154
                      int sz = vec[u].size();
                      for(int i = 0;i < sz;i++){
155
                          Node tmp = vec[u][i];
156
                          bits[tmp.root].add(tmp.dis,dv);
157
158
                          if(tmp.subtree !=-1)
159
                              bits[tmp.subtree].add(tmp.dis,dv);
                      }
160
```

```
161
                      val[u] += dv;
162
                  }
                  else {
163
                      int ans = 0;
164
                      int sz = vec[u].size();
165
                      for(int i = 0;i < sz;i++){</pre>
166
                          Node tmp = vec[u][i];
167
                          ans += bits[tmp.root].sum(d-tmp.dis);
168
169
                          if(tmp.subtree !=-1)
                               ans -= bits[tmp.subtree].sum(d-tmp.dis);
170
171
172
                      printf("%d\n",ans);
173
                  }
174
             }
175
         }
176
         return 0;
177 |}
```

#### 4.22.3 链分治 -HDU5039

HDU~5039 一颗树,每条边的属性为 0 或 1,求有多少条路径经过奇数条属性为 1 的边。一种是查询操作,一种是修改边的属性。

虽然这题有更加简单的方法,但是用来练习链分治还是不错的。

```
1 | const int MAXN = 30010;
   const int INF = 0x3f3f3f3f;
   struct Edge{
 3
 4
       int to,next;
       int f;
 5
   }edge[MAXN*2];
   int head[MAXN],tot;
   void init(){
 8
 9
       tot = 0;
10
       memset(head,-1,sizeof(head));
11
12
   void addedge(int u,int v,int f){
13
       edge[tot].to = v;
14
       edge[tot].next = head[u];
       edge[tot].f = f;
15
       head[u] = tot++;
16
17
18
   long long ans;
   int num0[MAXN],num1[MAXN];
20
   long long tnum[MAXN];
   struct Node{
21
       int l0,l1;
22
       int r0,r1;
23
24
       int cc;
25
       long long sum;
       Node gao(int u){
26
27
            l0 = r0 = num0[u];
28
            l1 = r1 = num1[u];
```

```
29
            sum = tnum[u];
30
            cc = 0;
31
            return *this;
32
        }
33
   };
34
   int pos[MAXN];
35
   int val[MAXN];
   int fa[MAXN];
37
   int cnt[MAXN];
   int col[MAXN];
38
   int link[MAXN];
39
40
   int CHANGEU;
41
   struct chain{
42
        vector<int>uu;
43
        vector<Node>nde;
44
        int n;
45
        void init(){
46
            n = uu.size();
47
            nde.resize(n << 2);</pre>
            for(int i = 0;i < n;i++)pos[uu[i]] = i;</pre>
48
49
            build(0,n-1,1);
50
        }
51
        void up(int l,int r,int p){
            int mid = (l+r)>>1;
52
            nde[p].cc = nde[p<<1].cc ^ <math>nde[(p<<1)|1].cc ^ val[uu[mid]];
53
            nde[p].l0 = nde[p << 1].l0;
54
            nde[p].l1 = nde[p << 1].l1;
55
            if(nde[p<<1].cc^val[uu[mid]]){</pre>
56
57
                 nde[p].l0 += nde[(p << 1)|1].l1;
58
                 nde[p].l1 += nde[(p << 1)|1].l0;
59
            }
            else {
60
61
                 nde[p].lo += nde[(p << 1)|1].lo;
62
                 nde[p].l1 += nde[(p << 1)|1].l1;
63
64
            nde[p].r0 = nde[(p << 1)|1].r0;
65
            nde[p].r1 = nde[(p << 1)|1].r1;
66
            if(nde[(p<<1)|1].cc^val[uu[mid]]){
                 nde[p].r0 += nde[p<<1].r1;
67
68
                 nde[p].r1 += nde[p<<1].r0;
            }
69
            else {
70
71
                 nde[p].r0 += nde[p<<1].r0;
72
                 nde[p].r1 += nde[p<<1].r1;
73
            if(val[uu[mid]] == 0){
74
75
                 nde[p].sum = nde[p << 1].sum + nde[(p << 1)|1].sum +
76
                     (long long) nde[p<<1].r0*nde[(p<<1)|1].l1 +
77
                     (long long) nde[p<<1].r1*nde[(p<<1)|1].l0;
78
            }
            else {
79
```

```
80
                 nde[p].sum = nde[p << 1].sum + nde[(p << 1)|1].sum +
 81
                      (long long) nde[p<<1].r0*nde[(p<<1)|1].l0 +
 82
                      (long long) nde[p<<1].r1*nde[(p<<1)|1].l1;
 83
             }
 84
         }
 85
         void build(int l,int r,int p){
             if(l == r){
 86
 87
                 nde[p].gao(uu[l]);
 88
                 return;
 89
             int mid = (l+r)/2;
 90
 91
             build(l,mid,p<<1);</pre>
             build(mid+1,r,(p<<1)|1);
 92
 93
             up(l,r,p);
 94
         }
 95
         void update(int k,int l,int r,int p){
             if(l == r){
 96
 97
                 nde[p].gao(uu[k]);
 98
                 return;
 99
             }
             int mid = (l+r)/2;
100
101
             if(k <= mid)update(k,l,mid,p<<1);</pre>
102
             else update(k,mid+1,r,(p<<1)|1);
103
             up(l,r,p);
104
         int change(int y){
105
             int x = uu.back();
106
             int p = fa[x];
107
108
             if(p){
109
                 if(x == CHANGEU)val[x] ^= 1;
110
                 if(val[x]){
                      tnum[p] = (long long)nde[1].r0*(num0[p]-nde[1].r1)
111
                      tnum[p] -= (long long)nde[1].r1*(num1[p]-nde[1].r0)
112
113
                      num0[p] = nde[1].r1;
114
                      num1[p] -= nde[1].r0;
115
                 }
                 else {
116
117
                      tnum[p] = (long long)nde[1].r1*(num0[p]-nde[1].r0)
                      tnum[p] -= (long long)nde[1].r0*(num1[p]-nde[1].r1)
118
119
                      num0[p] -= nde[1].r0;
120
                      num1[p] -= nde[1].r1;
121
                 if(x == CHANGEU)val[x] ^= 1;
122
             }
123
124
             ans -= nde[1].sum;
125
             update(pos[y],0,n-1,1);
126
             if(p){
```

```
127
                 if(val[x]){
128
                      tnum[p] += (long long)nde[1].r0*num0[p];
                      tnum[p] += (long long)nde[1].r1*num1[p];
129
130
                      num0[p] += nde[1].r1;
131
                      num1[p] += nde[1].r0;
                 }
132
                 else {
133
134
                      tnum[p] += (long long)nde[1].r0*num1[p];
135
                      tnum[p] += (long long)nde[1].r1*num0[p];
                      num0[p] += nde[1].r0;
136
137
                      num1[p] += nde[1].r1;
                 }
138
139
             }
140
             ans += nde[1].sum;
             return p;
141
142
    }ch[MAXN];
143
144
    void dfs1(int u,int pre){
145
         chain &c = ch[u];
         c.uu.clear();
146
         int \vee, \times = 0;
147
148
         cnt[u] = 1;
149
         for(int i = head[u]; i != -1; i = edge[i].next){
             v = edge[i].to;
150
151
             if(v == pre)continue;
             dfs1(v,u);
152
             link[i/2] = v;
153
             val[v] = edge[i].f;
154
155
             cnt[u] += cnt[v];
156
             fa[v] = u;
157
             if(cnt[v] > cnt[x]) x = v;
158
159
         if(!x)col[u] = u;
         else col[u] = col[x];
160
         ch[col[u]].uu.push_back(u);
161
162
         num0[u] = 1;
163
         num1[u] = 0;
164
         tnum[u] = 0;
165
166
167
    void dfs2(int x){
168
         x = col[x];
169
         chain &c = ch[x];
         int n = c.uu.size();
170
         int u,v;
171
         for(int i = 1;i < n;i++){
172
             u = c.uu[i];
173
174
             for(int j = head[u]; j != -1; j = edge[j].next){
                 v = edge[j].to;
175
176
                 if(v == c.uu[i-1] || fa[u] == v)continue;
177
                 dfs2(v);
```

```
178
                 if(val[v]){
179
                      tnum[u] += (long long)num0[u]*ch[col[v]].nde[1].r0
                         + (long long)num1[u]*ch[col[v]].nde[1].r1;
                      num0[u] += ch[col[v]].nde[1].r1;
180
                      num1[u] += ch[col[v]].nde[1].r0;
181
                 }
182
                 else {
183
                      tnum[u] += (long long)num1[u]*ch[col[v]].nde[1].r0
184
                         + (long long)num0[u]*ch[col[v]].nde[1].r1;
                      num0[u] += ch[col[v]].nde[1].r0;
185
                      num1[u] += ch[col[v]].nde[1].r1;
186
                 }
187
             }
188
         }
189
190
         c.init();
191
         ans += c.nde[1].sum;
192
193
    char str[100];
    char str1[100],str2[100];
194
    int main()
195
196
    {
197
         int T;
198
         int iCase = 0;
         scanf("%d",&T);
199
200
         int n;
         while(T--){
201
             ans = 0;
202
203
             iCase++;
204
             scanf("%d",&n);
205
             map<string,int>mp;
206
             init();
             for(int i = 1;i <= n;i++){
207
                 scanf("%s",str);
208
                 mp[str] = i;
209
             }
210
211
             int u, v, f;
212
             for(int i = 1;i < n;i++){
                 scanf("%s%s%d",str1,str2,&f);
213
                 addedge(mp[str1],mp[str2],f);
214
                 addedge(mp[str2],mp[str1],f);
215
             }
216
             int Q;
217
             char op[10];
218
             scanf("%d",&Q);
219
220
             printf("Case<sub>□</sub>#%d:\n",iCase);
             val[1] = 0;
221
222
             fa[1] = 0;
             dfs1(1,1);
223
224
             dfs2(1);
225
             while(Q--){
                 scanf("%s",op);
226
```

```
if(op[0] == 'Q'){
227
                     printf("%I64d\n",ans*2);
228
                 }
229
                 else {
230
                     int id ;
231
                     scanf("%d",&id);
232
                     id--;
233
                     u = link[id];
234
                     val[u] ^= 1;
235
                     CHANGEU = u;
236
                     while(u)
237
                          u = ch[col[u]].change(u);
238
239
                 }
240
             }
241
         }
        return 0;
242
243 }
```

# 5 搜索

# 5.1 Dancing Links

#### 5.1.1 精确覆盖

```
1
 2
   * P0J3074
    */
   const int MaxN = N*N*N + 10;
   const int MaxM = N*N*4 + 10;
 7
   const int maxnode = MaxN*4 + MaxM + 10;
   char g[MaxN];
   struct DLX{
 9
10
       int n,m,size;
11
       int U[maxnode],D[maxnode],R[maxnode],L[maxnode],Row[maxnode],
          Col[maxnode];
12
       int H[MaxN],S[MaxM];
13
       int ansd,ans[MaxN];
       void init(int _n,int _m){
14
15
           n = _n;
           m = _m;
16
17
            for(int i = 0;i <= m;i++){
18
                S[i] = 0;
19
                U[i] = D[i] = i;
20
                L[i] = i-1;
                R[i] = i+1;
21
            }
22
23
           R[m] = 0; L[0] = m;
24
           size = m;
25
            for(int i = 1; i <= n; i++)H[i] = -1;
26
27
       void Link(int r,int c){
28
            ++S[Col[++size]=c];
29
           Row[size] = r;
30
           D[size] = D[c];
           U[D[c]] = size;
31
32
           U[size] = c;
33
           D[c] = size;
            if(H[r] < 0)H[r] = L[size] = R[size] = size;
34
35
            else{
36
                R[size] = R[H[r]];
37
                L[R[H[r]]] = size;
38
                L[size] = H[r];
39
                R[H[r]] = size;
40
            }
41
       }
42
       void remove(int c){
            L[R[c]] = L[c]; R[L[c]] = R[c];
43
            for(int i = D[c];i != c;i = D[i])
44
45
                for(int j = R[i]; j != i; j = R[j]){
```

```
U[D[j]] = U[j];
46
47
                    D[U[j]] = D[j];
                    ---S[Col[i]];
48
49
                }
50
51
       void resume(int c){
52
            for(int i = U[c];i != c;i = U[i])
53
                for(int j = L[i]; j != i; j = L[j])
54
                    ++S[Col[U[D[j]]=D[U[j]]=j]];
55
            L[R[c]] = R[L[c]] = c;
56
       bool Dance(int d){
57
            if(R[0] == 0){
58
59
                for(int i = 0; i < d; i++)g[(ans[i]-1)/9] = (ans[i]-1)%9
                   + '1';
60
                for(int i = 0;i < N*N;i++)printf("%c",g[i]);</pre>
61
                printf("\n");
62
                return true;
63
            }
            int c = R[0];
64
            for(int i = R[0];i != 0;i = R[i])
65
66
                if(S[i] < S[c])
67
                    c = i;
68
            remove(c);
69
            for(int i = D[c];i != c;i = D[i]){
                ans[d] = Row[i];
70
71
                for(int j = R[i]; j != i; j = R[j])remove(Col[j]);
72
                if(Dance(d+1))return true;
73
                for(int j = L[i]; j != i; j = L[j])resume(Col[j]);
74
            }
75
            resume(c);
76
            return false;
77
       }
78
79
   void place(int &r,int &c1,int &c2,int &c3,int &c4,int i,int j,int k
      ) {
80
       r = (i*N+j)*N + k; c1 = i*N+j+1; c2 = N*N+i*N+k;
81
       c3 = N*N*2+j*N+k; c4 = N*N*3+((j/3))*N+k;
82
83
   DLX dlx;
   int main(){
84
       while(scanf("%s",g) == 1){
85
86
            if(strcmp(g,"end") == 0)break;
87
            dlx.init(N*N*N,N*N*4);
            int r,c1,c2,c3,c4;
88
89
            for(int i = 0;i < N;i++)
                for(int j = 0; j < N; j++)
90
91
                    for(int k = 1;k <= N;k++)
                         if(g[i*N+j] == '.' || g[i*N+j] == '0'+k){
92
93
                             place(r,c1,c2,c3,c4,i,j,k);
94
                             dlx.Link(r,c1);
```

```
95
                              dlx.Link(r,c2);
 96
                              dlx.Link(r,c3);
 97
                              dlx.Link(r,c4);
 98
 99
             dlx.Dance(0);
100
101
         return 0;
102 |}
    5.1.2 可重复覆盖
  1
    /*
  2
     * FZU1686
  3
     */
    const int MaxM = 15*15+10;
    const int MaxN = 15*15+10;
    const int maxnode = MaxN * MaxM;
  7
    const int INF = 0x3f3f3f3f;
  8
    struct DLX{
  9
         int n,m,size;
 10
         int U[maxnode],D[maxnode],R[maxnode],L[maxnode],Row[maxnode],
            Col[maxnode];
         int H[MaxN],S[MaxM];
 11
 12
         int ansd;
 13
         void init(int _n,int _m){
 14
             n = _n;
             m = _m;
 15
             for(int i = 0;i <= m;i++){</pre>
 16
 17
                 S[i] = 0;
                 U[i] = D[i] = i;
 18
 19
                 L[i] = i-1;
 20
                 R[i] = i+1;
 21
             }
 22
             R[m] = 0; L[0] = m;
 23
             size = m;
 24
             for(int i = 1; i <= n; i++)H[i] = -1;
 25
 26
         void Link(int r,int c){
             ++S[Col[++size]=c];
 27
             Row[size] = r;
 28
 29
             D[size] = D[c];
             U[D[c]] = size;
 30
 31
             U[size] = c;
 32
             D[c] = size;
 33
             if(H[r] < 0)H[r] = L[size] = R[size] = size;
 34
             else{
 35
                 R[size] = R[H[r]];
                 L[R[H[r]]] = size;
 36
 37
                 L[size] = H[r];
                 R[H[r]] = size;
 38
 39
             }
 40
         }
```

```
41
        void remove(int c){
            for(int i = D[c];i != c;i = D[i])
42
43
                 L[R[i]] = L[i], R[L[i]] = R[i];
44
        }
45
        void resume(int c){
46
            for(int i = U[c];i != c;i = U[i])
47
                L[R[i]] = R[L[i]] = i;
48
49
        bool v[MaxM];
50
        int f(){
51
            int ret = 0;
            for(int c = R[0]; c != 0;c = R[c])v[c] = true;
52
            for(int c = R[0]; c != 0;c = R[c])
53
                 if(v[c])
54
55
                 {
56
                     ret++;
57
                     v[c] = false;
58
                     for(int i = D[c];i != c;i = D[i])
59
                         for(int j = R[i]; j != i; j = R[j])
                              v[Col[j]] = false;
60
61
                 }
62
            return ret;
63
        }
        void Dance(int d){
64
65
            if(d + f() >= ansd)return;
            \mathbf{if}(R[0] == 0)
66
                if(d < ansd) ansd = d;</pre>
67
68
                return;
69
            }
70
            int c = R[0];
            for(int i = R[0];i != 0;i = R[i])
71
72
                 if(S[i] < S[c])
73
                     c = i;
            for(int i = D[c];i != c;i = D[i]){
74
75
                 remove(i);
76
                for(int j = R[i]; j != i; j = R[j])remove(j);
77
                Dance(d+1);
78
                for(int j = L[i]; j != i; j = L[j])resume(j);
79
                 resume(i);
80
            }
81
        }
82
   };
83
   DLX g;
   int a[20][20];
   int id[20][20];
85
86
   int main(){
87
        int n,m;
        while(scanf("%d%d",&n,&m) == 2){
88
89
            int sz = 0;
90
            memset(id,0,sizeof(id));
            for(int i = 0;i < n;i++)</pre>
91
```

```
92
                for(int j = 0; j < m; j++){
93
                     scanf("%d",&a[i][j]);
94
                     if(a[i][j] == 1)id[i][j] = (++sz);
95
            g.init(n*m,sz);
96
97
            sz = 1;
            int n1, m1;
 98
            scanf("%d%d",&n1,&m1);
99
100
            for(int i = 0;i < n;i++)
                for(int j = 0; j < m; j++) {
101
                     for(int x = 0; x < n1 && i + x < n; x++)
102
                         for(int y = 0; y < m1 && j + y < m; y++)
103
                             if(id[i+x][i+y])
104
                                 g.Link(sz,id[i+x][j+y]);
105
106
                     sz++;
107
                }
108
            g.ansd = INF;
109
            g.Dance(0);
            printf("%d\n",g.ansd);
110
111
        }
112
        return 0;
113 |}
    5.2
        八数码
    5.2.1 HDU1043 反向搜索
  1 | / *
   HDU 1043 Eight
    八数码,输出路径
    思路:反向搜索,从目标状态找回状态对应的路径
  5
    用康托展开判重
  6
    */
    |const int MAXN=1000000;//最多是 9!/2
    int fac[]={1,1,2,6,24,120,720,5040,40320,362880};//康拖展开判重
               0!1!2!3! 4! 5! 6! 7!
 9
    //
                                       8!
                                               9!
    |bool vis[MAXN];//标记
 10
    string path[MAXN];//记录路径
    //康拖展开求该序列的 hash 值
 12
 13
    int cantor(int s[]){
        int sum=0;
 14
 15
        for(int i=0;i<9;i++){</pre>
 16
            int num=0;
 17
            for(int j=i+1;j<9;j++)
              if(s[j]<s[i])num++;
 18
 19
            sum+=(num*fac[9-i-1]);
 20
        }
 21
        return sum+1;
 22
 23
    struct Node{
        int s[9];
 24
        int loc;//'0' 的位置
 25
```

```
26
        int status;//康拖展开的 hash 值
27
        string path;//路径
28
   };
   int move[4][2]=\{\{-1,0\},\{1,0\},\{0,-1\},\{0,1\}\}\};//u,d,l,r
29
   char indexs[5]="durl";//和上面的要相反,因为是反向搜索
30
31
   int aim=46234;//123456780 对应的康拖展开的 hash 值
32
   void bfs(){
33
        memset(vis, false, sizeof(vis));
34
        Node cur, next;
35
        for(int i=0;i<8;i++)cur.s[i]=i+1;</pre>
36
        cur.s[8]=0;
37
        cur.loc=8;
38
        cur.status=aim;
39
        cur.path="";
40
        queue<Node>q;
41
        q.push(cur);
        path[aim]="";
42
43
        while(!q.empty()){
            cur=q.front();
44
45
            q.pop();
            int x=cur.loc/3;
46
47
            int y=cur.loc%3;
48
            for(int i=0;i<4;i++){</pre>
49
                int tx=x+move[i][0];
50
                int ty=y+move[i][1];
                if(tx<0||tx>2||ty<0||ty>2)continue;
51
52
                next=cur;
                next.loc=tx*3+ty;
53
54
                next.s[cur.loc]=next.s[next.loc];
55
                next.s[next.loc]=0;
                next.status=cantor(next.s);
56
57
                if(!vis[next.status]){
58
                     vis[next.status]=true;
59
                     next.path=indexs[i]+next.path;
60
                     q.push(next);
61
                     path[next.status]=next.path;
62
                }
63
            }
        }
64
65
66
67
   int main(){
68
        char ch;
69
        Node cur;
70
        bfs();
        while(cin>>ch){
71
            if(ch=='x') {cur.s[0]=0;cur.loc=0;}
72
            else cur.s[0]=ch-'0';
73
            for(int i=1;i<9;i++){</pre>
74
75
                cin>>ch;
                if(ch=='x'){
76
```

```
cur.s[i]=0;
77
                      cur.loc=i;
78
                 }
79
                 else cur.s[i]=ch-'0';
80
            }
81
            cur.status=cantor(cur.s);
82
            if(vis[cur.status]){
83
                 cout<<path[cur.status]<<endl;</pre>
84
            }
85
            else cout<<"unsolvable"<<endl;</pre>
86
87
        }
        return 0;
88
89 }
```

# 6 动态规划

# 6.1 最长上升子序列 O(nlogn)

```
1 | const int MAXN=500010;
 2
   int a[MAXN],b[MAXN];
 3
   |//用二分查找的方法找到一个位置,使得 num>b[i-1] 并且 num<b[i], 并用 num 代
      替 b[i]
   int Search(int num,int low,int high){
 5
 6
       int mid;
       while(low<=high){</pre>
 7
           mid=(low+high)/2;
 8
           if(num>=b[mid])
 9
                           low=mid+1;
                  high=mid-1;
10
11
       }
12
       return low;
13
   int DP(int n){
14
       int i,len,pos;
15
16
       b[1]=a[1];
17
       len=1;
       for(i=2;i<=n;i++){
18
           if(a[i]>=b[len])//如果 a[i] 比 b[] 数组中最大还大直接插入到后面即
19
              可
20
           {
               len=len+1;
21
               b[len]=a[i];
22
23
           }
           else//用二分的方法在 b[] 数组中找出第一个比 a[i] 大的位置并且让
24
              a[i] 替代这个位置
25
           {
26
               pos=Search(a[i],1,len);
27
               b[pos]=a[i];
28
           }
29
30
       return len;
31 | }
        背包
   6.2
  int nValue,nKind;
 2
   //0-1 背包, 代价为 cost, 获得的价值为 weight
 3
 4
   void ZeroOnePack(int cost,int weight){
       for(int i=nValue;i>=cost;i---)
 5
         dp[i]=max(dp[i],dp[i-cost]+weight);
 6
   }
 7
 8
   //完全背包,代价为 cost, 获得的价值为 weight
   void CompletePack(int cost,int weight){
10
11
       for(int i=cost;i<=nValue;i++)</pre>
```

```
12
         dp[i]=max(dp[i],dp[i-cost]+weight);
13
   }
14
15
   //多重背包
   void MultiplePack(int cost,int weight,int amount){
16
17
       if(cost*amount>=nValue) CompletePack(cost,weight);
       else{
18
           int k=1;
19
20
           while(k<amount) {</pre>
               ZeroOnePack(k*cost,k*weight);
21
22
               amount-=k;
23
               k<<=1;
24
           ZeroOnePack(amount*cost,amount*weight);//这个不要忘记了,经常
25
26
       }
27 |}
   分组背包:
   for k = 1 to K
   for v = V to 0
   for item i in group k
   F[v] = \max F[v], F[v-C_i] + W_i
        插头 DP
   6.3
   6.3.1 HDU 4285
   求 K 个回路的方案数。而且不能是环套环。
   增加个标志位来记录形成的回路个数。而且注意避免环套环的情况。不形成环套环的话就是
   在形成新的回路时,两边的插头个数要为偶数。
 1 | /*
 2
  HDU 4285
   |要形成刚好 K 条回路的方法数
  要避免环套环的情况。
 5 所以形成回路时,要保证两边的插头数是偶数
   G++ 11265ms 11820K
 7
   C++ 10656ms 11764K
 8
   */
   const int MAXD=15;
   const int STATE=1000010;
   | const int HASH=300007;//这个大一点可以防止        TLE, 但是容易        MLE
   const int MOD=10000000007;
   int N,M,K;
  int maze[MAXD][MAXD];
   int code[MAXD];
15
16 int ch[MAXD];
   |int num;//圈的个数
   struct HASHMAP
18
19
   {
20
       int head[HASH],next[STATE],size;
       long long state[STATE];
21
```

```
22
        int f[STATE];
23
        void init()
24
        {
25
            size=0;
26
            memset(head,-1,sizeof(head));
27
        void push(long long st,int ans)
28
29
        {
30
            int i;
            int h=st%HASH;
31
            for(i=head[h];i!=-1;i=next[i])
32
33
              if(state[i]==st)
34
              {
35
                   f[i]+=ans;
36
                   f[i]%=MOD;
37
                   return;
              }
38
            state[size]=st;
39
40
            f[size]=ans;
41
            next[size]=head[h];
42
            head[h]=size++;
43
        }
   }hm[2];
44
45
   void decode(int *code,int m,long long
46
   {
47
        num=st&63;
48
        st>>=6;
49
        for(int i=m;i>=0;i—)
50
51
            code[i]=st&7;
52
            st>>=3;
53
        }
54
55
   long long encode(int *code,int m)//最小表示法
56
   {
57
        int cnt=1;
        memset(ch,-1,sizeof(ch));
58
        ch[0]=0;
59
60
        long long st=0;
        for(int i=0;i<=m;i++)</pre>
61
62
        {
            if(ch[code[i]]==-1)ch[code[i]]=cnt++;
63
64
            code[i]=ch[code[i]];
65
            st<<=3;
            st|=code[i];
66
        }
67
68
        st<<=6;
69
        st|=num;
70
        return st;
71
72 void shift(int *code,int m)
```

```
73
    {
74
         for(int i=m;i>0;i—)code[i]=code[i-1];
 75
         code[0]=0;
 76
    void dpblank(int i,int j,int cur)
 77
 78
 79
         int k,left,up;
 80
         for(k=0;k<hm[cur].size;k++)</pre>
 81
             decode(code,M,hm[cur].state[k]);
 82
             left=code[j-1];
 83
             up=code[j];
 84
             if(left&&up)
 85
 86
                 if(left==up)
 87
 88
                 {
                      if(num>=K)continue;
 89
                      int t=0;
 90
 91
                      //要避免环套环的情况,需要两边插头数为偶数
 92
                      for(int p=0;p<j-1;p++)
                        if(code[p])t++;
 93
 94
                      if(t&1)continue;
                      if(num<K)</pre>
 95
                      {
 96
 97
                          num++;
                          code[j-1]=code[j]=0;
 98
 99
                          hm[cur^1].push(encode(code,j==M?M-1:M),hm[cur].
                             f[k]);
                      }
100
101
                 }
                 else
102
103
                 {
104
                      code[j-1]=code[j]=0;
105
                      for(int t=0;t<=M;t++)
                        if(code[t]==up)
106
107
                          code[t]=left;
                      hm[cur^1].push(encode(code,j==M?M-1:M),hm[cur].f[k
108
                         ]);
109
                 }
110
             }
             else if(left||up)
111
112
             {
113
                 int t;
                 if(left)t=left;
114
115
                 else t=up;
                 if(maze[i][j+1])
116
117
                 {
118
                      code[j-1]=0;
119
                      code[j]=t;
                      hm[cur^1].push(encode(code,M),hm[cur].f[k]);
120
                 }
121
```

```
122
                  if(maze[i+1][j])
123
                  {
124
                      code[j]=0;
125
                      code[j-1]=t;
126
                      hm[cur^1].push(encode(code,j==M?M-1:M),hm[cur].f[k
                  }
127
             }
128
129
             else
130
             {
                  if(maze[i][j+1]&&maze[i+1][j])
131
132
                      code[j-1]=code[j]=13;
133
                      hm[cur^1].push(encode(code,j==M?M-1:M),hm[cur].f[k
134
                         ]);
135
                  }
             }
136
         }
137
138
    void dpblock(int i,int j,int cur)
139
140
    {
141
         int k;
         for(k=0;k<hm[cur].size;k++)</pre>
142
143
         {
144
             decode(code,M,hm[cur].state[k]);
145
             code[i-1]=code[i]=0;
             hm[cur^1].push(encode(code,j==M?M-1:M),hm[cur].f[k]);
146
147
         }
148
149
    char str[20];
150
    void init()
151
    {
         scanf("%d%d%d",&N,&M,&K);
152
153
         memset(maze,0,sizeof(maze));
154
         for(int i=1;i<=N;i++)</pre>
155
         {
156
             scanf("%s",&str);
             for(int j=1;j<=M;j++)</pre>
157
158
                if(str[j-1]=='.')
159
                  maze[i][j]=1;
         }
160
161
162
    void solve()
163
    {
164
         int i,j,cur=0;
         hm[cur].init();
165
         hm[cur].push(0,1);
166
         for(i=1;i<=N;i++)</pre>
167
           for(j=1;j<=M;j++)
168
169
           {
               hm[cur^1].init();
170
```

```
if(maze[i][j])dpblank(i,j,cur);
171
                else dpblock(i,j,cur);
172
173
                cur^=1;
            }
174
         int ans=0;
175
         for(i=0;i<hm[cur].size;i++)</pre>
176
177
            if(hm[cur].state[i]==K)
178
            {
179
                ans+=hm[cur].f[i];
180
                ans%=MOD;
181
         printf("%d\n",ans);
182
183
184
185
    int main()
186
     {
187
         int T;
         scanf("%d",&T);
188
         while(T--)
189
190
              init();
191
192
              solve();
193
194
         return 0;
195
    }
196
197
    /*
198
    Sample Input
199
    4 4 1
200
    **..
201
     . . . .
202
203
204
    4 1
205
     . . . .
206
    . . . .
207
     . . . .
208
     . . . .
209
210
    Sample Output
211
    6
212
213 |*/
```

## 7 计算几何

## 7.1 二维几何

```
1 // 计算几何模板
   const double eps = 1e-8;
 3 const double inf = 1e20;
   const double pi = acos(-1.0);
   const int maxp = 1010;
   //Compares a double to zero
   int sgn(double x){
 7
 8
       if(fabs(x) < eps)return 0;</pre>
 9
       if(x < 0) return -1;
10
       else return 1;
   }
11
   //square of a double
   inline double sqr(double x){return x*x;}
13
14
15
   * Point
    * Point()

    Empty constructor

16
    * Point(double _x,double _y) - constructor
17
    * input()
                           - double input
18
                           - %.2f output
    * output()
19
20
    * operator ==
                          compares x and y
                           - compares first by x, then by y
21
    * operator <
22
    * operator -

    return new Point after subtracting

       curresponging x and y
    * operator ^
23

    cross product of 2d points

    * operator *
                           – dot product
24
    * len()

    gives length from origin

25
    * len2()

    gives square of length from origin

26
    * distance(Point p) — gives distance from p
27
    * operator + Point b - returns new Point after adding
28
       curresponging x and y
    \star operator \star double k - returns new Point after multiplieing x and
29
        y by k
30
    * operator / double k - returns new Point after divideing x and y
31
    * rad(Point a, Point b)— returns the angle of Point a and Point b
       from this Point
    * trunc(double r)
                           - return Point that if truncated the
32
       distance from center to r
                          - returns 90 degree ccw rotated point
    * rotleft()
33
    * rotright()

    returns 90 degree cw rotated point

34
    * rotate(Point p, double angle) — returns Point after rotateing the
35
        Point centering at p by angle radian ccw
36
    */
37
   struct Point{
38
       double x,y;
39
       Point(){}
40
       Point(double _x,double _y){
```

```
41
            x = _x;
42
            y = _y;
43
44
        void input(){
45
            scanf("%lf%lf",&x,&y);
46
47
        void output(){
48
            printf("%.2f_{\perp}%.2f_{\mid}n",x,y);
49
        bool operator == (Point b)const{
50
            return sgn(x-b.x) == 0 \&\& sgn(y-b.y) == 0;
51
52
        }
53
        bool operator < (Point b)const{</pre>
54
            return sgn(x-b.x) == 0?sgn(y-b.y) < 0:x < b.x;
55
        }
56
        Point operator -(const Point &b)const{
57
            return Point(x-b.x,y-b.y);
58
        }
        //叉积
59
        double operator ^(const Point &b)const{
60
            return x*b.y - y*b.x;
61
62
        }
        //点积
63
        double operator *(const Point &b)const{
64
65
            return x*b.x + y*b.y;
        }
66
        //返回长度
67
        double len(){
68
69
            return hypot(x,y);//库函数
70
        }
        //返回长度的平方
71
        double len2(){
72
73
            return x*x + y*y;
74
        }
75
        //返回两点的距离
76
        double distance(Point p){
            return hypot(x-p.x,y-p.y);
77
78
79
        Point operator +(const Point &b)const{
80
            return Point(x+b.x,y+b.y);
81
        }
        Point operator *(const double &k)const{
82
            return Point(x*k,y*k);
83
84
        }
85
        Point operator /(const double &k)const{
86
            return Point(x/k,y/k);
87
        }
        //计算 pa 和 pb 的夹角
88
        //就是求这个点看 a,b 所成的夹角
89
90
        //测试 Light0J1203
        double rad(Point a,Point b){
91
```

```
92
            Point p = *this;
93
             return fabs(atan2( fabs((a-p)^(b-p)),(a-p)*(b-p));
        }
94
        //化为长度为 r 的向量
95
        Point trunc(double r){
96
 97
            double l = len();
             if(!sgn(l))return *this;
 98
99
             r /= l;
100
             return Point(x*r,y*r);
        }
101
        //逆时针旋转 90 度
102
        Point rotleft(){
103
            return Point(-y,x);
104
105
        //顺时针旋转 90 度
106
107
        Point rotright(){
108
             return Point(y,-x);
109
        }
        //绕着 p 点逆时针旋转 angle
110
        Point rotate(Point p,double angle){
111
            Point v = (*this) - p;
112
113
            double c = cos(angle), s = sin(angle);
114
             return Point(p.x + v.x*c - v.y*s,p.y + v.x*s + v.y*c);
115
        }
116
    };
117
    /*
118
    * Stores two points
     * Line()
119

    Empty constructor

120
     * Line(Point _s,Point _e)
                                        Line through _s and _e
121
     * operator ==

    checks if two points are same

     * Line(Point p,double angle)
                                       - one end p , another end at
122
        angle degree
     * Line(double a, double b, double c) — Line of equation ax + by + c
123
        = 0
     * input()
                                        inputs s and e
124
125
     * adjust()
                                        — orders in such a way that s < e</p>
126
     * length()
                                        – distance of se
127
     * angle()
                                        - return 0 <= angle < pi</pre>
128
     * relation(Point p)
                                        — 3 if point is on line
                                          1 if point on the left of line
129
130
                                          2 if point on the right of line

    return true if point on segment

131
     * pointonseg(double p)
132
     * parallel(Line v)

    return true if they are

        parallel
133
     * segcrossseg(Line v)
                                        - returns 0 if does not intersect
                                          returns 1 if non-standard
134
        intersection
                                          returns 2 if intersects
135
     * linecrossseg(Line v)
136

    line and seg

     * linecrossline(Line v)
                                       - 0 if parallel
137
                                          1 if coincides
138
```

```
139
                                           2 if intersects
                                         - returns intersection point
140
     * crosspoint(Line v)
141
     * dispointtoline(Point p)
                                         — distance from point p to the
        line
142
     * dispointtoseg(Point p)

    distance from p to the segment

143
     * dissegtoseg(Line v)

    distance of two segment

144
     * lineprog(Point p)

    returns projected point p on se

         line
145
     * symmetrypoint(Point p)

    returns reflection point of p

        over se
146
     *
147
     */
148
    struct Line{
149
         Point s,e;
150
         Line(){}
151
         Line(Point _s,Point _e){
             s = _s;
152
153
             e = _e;
154
         bool operator ==(Line v){
155
             return (s == v.s)&&(e == v.e);
156
157
         //根据一个点和倾斜角 angle 确定直线,0<=angle<pi
158
         Line(Point p,double angle){
159
             s = p;
160
             if(sgn(angle-pi/2) == 0){
161
                 e = (s + Point(0,1));
162
             }
163
164
             else{
165
                 e = (s + Point(1,tan(angle)));
166
             }
         }
167
         //ax+by+c=0
168
         Line(double a,double b,double c){
169
             if(sgn(a) == 0){
170
                 s = Point(0, -c/b);
171
172
                 e = Point(1,-c/b);
173
             else if(sgn(b) == 0){
174
                 s = Point(-c/a, 0);
175
                 e = Point(-c/a, 1);
176
             }
177
             else{
178
                 s = Point(0, -c/b);
179
180
                 e = Point(1, (-c-a)/b);
             }
181
         }
182
         void input(){
183
             s.input();
184
185
             e.input();
186
         }
```

```
void adjust(){
187
188
            if(e < s)swap(s,e);
        }
189
        //求线段长度
190
        double length(){
191
            return s.distance(e);
192
193
        //返回直线倾斜角 0<=angle<pi
194
195
        double angle(){
            double k = atan2(e.y-s.y,e.x-s.x);
196
197
            if(sgn(k) < 0)k += pi;
            if(sgn(k-pi) == 0)k -= pi;
198
            return k;
199
        }
200
        //点和直线关系
201
202
        //1 在左侧
        //2 在右侧
203
        //3 在直线上
204
        int relation(Point p){
205
            int c = sgn((p-s)^{(e-s)});
206
            if(c < 0) return 1;
207
208
            else if(c > 0)return 2;
209
            else return 3;
        }
210
        // 点在线段上的判断
211
212
        bool pointonseg(Point p){
213
            return sgn((p-s)^{(e-s)}) == 0 \&\& sgn((p-s)*(p-e)) <= 0;
214
215
        //两向量平行 (对应直线平行或重合)
216
        bool parallel(Line v){
217
            return sgn((e-s)^{(v.e-v.s)}) == 0;
218
        }
219
        //两线段相交判断
        //2 规范相交
220
        //1 非规范相交
221
222
        //0 不相交
223
        int segcrossseg(Line v){
224
            int d1 = sgn((e-s)^{(v.s-s)});
225
            int d2 = sgn((e-s)^{(v.e-s)});
226
            int d3 = sgn((v.e-v.s)^(s-v.s));
            int d4 = sgn((v.e-v.s)^{(e-v.s)});
227
            if( (d1^d2)==-2 \&\& (d3^d4)==-2 ) return 2;
228
            return (d1==0 && sgn((v.s-s)*(v.s-e))<=0) ||
229
                 (d2==0 \&\& sgn((v.e-s)*(v.e-e))<=0) | |
230
                 (d3==0 \&\& sgn((s-v.s)*(s-v.e))<=0) | |
231
                 (d4==0 \&\& sgn((e-v.s)*(e-v.e))<=0);
232
        }
233
        //直线和线段相交判断
234
235
        //-*this line -v seg
236
        //2 规范相交
        //1 非规范相交
237
```

```
238
        //0 不相交
        int linecrossseg(Line v){
239
            int d1 = sgn((e-s)^{(v.s-s)});
240
241
            int d2 = sgn((e-s)^(v.e-s));
            if((d1^d2)==-2) return 2;
242
            return (d1==0||d2==0);
243
244
        }
        //两直线关系
245
        //0 平行
246
        //1 重合
247
248
        //2 相交
        int linecrossline(Line v){
249
            if((*this).parallel(v))
250
251
                return v.relation(s)==3;
252
            return 2;
253
        }
        //求两直线的交点
254
255
        //要保证两直线不平行或重合
256
        Point crosspoint(Line v){
            double a1 = (v.e-v.s)^{(s-v.s)};
257
            double a2 = (v.e-v.s)^{(e-v.s)};
258
259
            return Point((s.x*a2-e.x*a1)/(a2-a1),(s.y*a2-e.y*a1)/(a2-a1
               ));
        }
260
        //点到直线的距离
261
        double dispointtoline(Point p){
262
            return fabs((p-s)^(e-s))/length();
263
264
        //点到线段的距离
265
266
        double dispointtoseg(Point p){
            if(sgn((p-s)*(e-s))<0 \mid | sgn((p-e)*(s-e))<0)
267
                return min(p.distance(s),p.distance(e));
268
269
            return dispointtoline(p);
270
        }
        //返回线段到线段的距离
271
272
        //前提是两线段不相交,相交距离就是 0 了
273
        double dissegtoseg(Line v){
            return min(min(dispointtoseg(v.s), dispointtoseg(v.e)), min(v
274
               .dispointtoseg(s), v.dispointtoseg(e)));
275
        }
        //返回点 p 在直线上的投影
276
277
        Point lineprog(Point p){
            return s + (((e-s)*((e-s)*(p-s)))/((e-s).len2()));
278
279
        }
        //返回点 p 关于直线的对称点
280
        Point symmetrypoint(Point p){
281
            Point q = lineprog(p);
282
            return Point(2*q.x-p.x,2*q.y-p.y);
283
284
        }
285
    };
286 | / /圆
```

```
287
    struct circle{
288
        Point p;//圆心
        double r;//半径
289
290
        circle(){}
        circle(Point _p,double _r){
291
292
             p = p;
293
             r = _r;
294
        }
295
        circle(double x,double y,double _r){
296
             p = Point(x,y);
297
             r = _r;
298
        }
        //三角形的外接圆
299
         //需要 Point 的 + / rotate() 以及 Line 的 crosspoint()
300
        //利用两条边的中垂线得到圆心
301
302
         //测试: UVA12304
        circle(Point a, Point b, Point c){
303
             Line u = Line((a+b)/2,((a+b)/2)+((b-a).rotleft()));
304
             Line v = Line((b+c)/2,((b+c)/2)+((c-b).rotleft()));
305
306
             p = u.crosspoint(v);
             r = p.distance(a);
307
308
        }
309
        //三角形的内切圆
        //参数 bool t 没有作用,只是为了和上面外接圆函数区别
310
        //测试: UVA12304
311
        circle(Point a, Point b, Point c, bool t){
312
             Line u,v;
313
314
             double m = atan2(b.y-a.y,b.x-a.x), n = atan2(c.y-a.y,c.x-a.
                x);
315
            u.s = a;
316
             u.e = u.s + Point(cos((n+m)/2), sin((n+m)/2));
317
             v.s = b;
             m = atan2(a.y-b.y,a.x-b.x), n = atan2(c.y-b.y,c.x-b.x);
318
            v.e = v.s + Point(cos((n+m)/2), sin((n+m)/2));
319
             p = u.crosspoint(v);
320
321
             r = Line(a,b).dispointtoseg(p);
322
        }
        //输入
323
        void input(){
324
325
             p.input();
             scanf("%lf",&r);
326
327
        }
        //输出
328
        void output(){
329
             printf("%.2lf_{\perp}%.2lf_{\parallel}%.2lf_{\mid}n",p.x,p.y,r);
330
        }
331
332
        bool operator == (circle v){
             return (p==v.p) && sgn(r-v.r)==0;
333
334
335
        bool operator < (circle v)const{</pre>
336
             return ((p<v.p)||((p==v.p)&&sgn(r-v.r)<0));
```

```
}
337
338
        //面积
        double area(){
339
340
             return pi*r*r;
        }
341
        //周长
342
        double circumference(){
343
344
             return 2*pi*r;
345
        }
        //点和圆的关系
346
        //0 圆外
347
        //1 圆上
348
        //2 圆内
349
        int relation(Point b){
350
            double dst = b.distance(p);
351
352
             if(sgn(dst-r) < 0)return 2;</pre>
             else if(sgn(dst-r)==0)return 1;
353
354
             return 0;
355
        }
        //线段和圆的关系
356
        //比较的是圆心到线段的距离和半径的关系
357
358
        int relationseg(Line v){
359
            double dst = v.dispointtoseg(p);
             if(sgn(dst-r) < 0)return 2;</pre>
360
             else if(sgn(dst-r) == 0)return 1;
361
362
             return 0;
        }
363
        //直线和圆的关系
364
365
        //比较的是圆心到直线的距离和半径的关系
366
        int relationline(Line v){
            double dst = v.dispointtoline(p);
367
             if(sgn(dst-r) < 0)return 2;</pre>
368
             else if(sgn(dst-r) == 0)return 1;
369
370
             return 0;
        }
371
372
        //两圆的关系
373
        //5 相离
        //4 外切
374
375
        //3 相交
        //2 内切
376
        //1 内含
377
        //需要 Point 的 distance
378
        //测试: UVA12304
379
        int relationcircle(circle v){
380
            double d = p.distance(v.p);
381
             if(sgn(d-r-v.r) > 0) return 5;
382
             if(sgn(d-r-v.r) == 0)return 4;
383
            double l = fabs(r-v.r);
384
             if(sgn(d-r-v.r)<0 && sgn(d-l)>0)return 3;
385
386
             if(sgn(d-l)==0)return 2;
             if(sgn(d-l)<0)return 1;</pre>
387
```

```
388
        }
389
        //求两个圆的交点,返回 0 表示没有交点,返回 1 是一个交点,2 是两个交点
        //需要 relationcircle
390
        //测试: UVA12304
391
        int pointcrosscircle(circle v,Point &p1,Point &p2){
392
            int rel = relationcircle(v);
393
            if(rel == 1 || rel == 5)return 0;
394
            double d = p.distance(v.p);
395
396
            double l = (d*d+r*r-v.r*v.r)/(2*d);
            double h = sqrt(r*r-l*l);
397
398
            Point tmp = p + (v.p-p).trunc(l);
            p1 = tmp + ((v.p-p).rotleft().trunc(h));
399
            p2 = tmp + ((v.p-p).rotright().trunc(h));
400
            if(rel == 2 || rel == 4)
401
402
                return 1;
403
            return 2;
        }
404
        //求直线和圆的交点,返回交点个数
405
        int pointcrossline(Line v,Point &p1,Point &p2){
406
            if(!(*this).relationline(v))return 0;
407
            Point a = v.lineprog(p);
408
409
            double d = v.dispointtoline(p);
410
            d = sqrt(r*r-d*d);
            if(sgn(d) == 0){
411
                p1 = a;
412
                p2 = a;
413
414
                return 1;
            }
415
416
            p1 = a + (v.e-v.s).trunc(d);
417
            p2 = a - (v.e-v.s).trunc(d);
418
            return 2;
        }
419
        //得到过 a,b 两点, 半径为 r1 的两个圆
420
421
        int gercircle(Point a, Point b, double r1, circle &c1, circle &c2){
422
            circle x(a,r1),y(b,r1);
423
            int t = x.pointcrosscircle(y,c1.p,c2.p);
424
            if(!t)return 0;
425
            c1.r = c2.r = r;
426
            return t;
427
        }
        //得到与直线 u 相切,过点 q,半径为 r1 的圆
428
        //测试: UVA12304
429
430
        int getcircle(Line u,Point q,double r1,circle &c1,circle &c2){
            double dis = u.dispointtoline(q);
431
            if(sgn(dis-r1*2)>0)return 0;
432
            if(sgn(dis) == 0){
433
                c1.p = q + ((u.e-u.s).rotleft().trunc(r1));
434
                c2.p = q + ((u.e-u.s).rotright().trunc(r1));
435
                c1.r = c2.r = r1;
436
437
                return 2;
438
            }
```

```
439
            Line u1 = Line((u.s + (u.e-u.s).rotleft().trunc(r1)),(u.e +
                (u.e-u.s).rotleft().trunc(r1)));
            Line u2 = Line((u.s + (u.e-u.s).rotright().trunc(r1)),(u.e
440
               + (u.e-u.s).rotright().trunc(r1)));
            circle cc = circle(q,r1);
441
442
            Point p1,p2;
443
            if(!cc.pointcrossline(u1,p1,p2))cc.pointcrossline(u2,p1,p2)
444
            c1 = circle(p1,r1);
            if(p1 == p2){
445
446
                c2 = c1;
447
                return 1;
448
            }
449
            c2 = circle(p2,r1);
450
            return 2;
451
        }
        //同时与直线 u,v 相切,半径为 r1 的圆
452
453
        //测试: UVA12304
        int getcircle(Line u,Line v,double r1,circle &c1,circle &c2,
454
           circle &c3,circle &c4){
            if(u.parallel(v))return 0;//两直线平行
455
456
            Line u1 = Line(u.s + (u.e-u.s).rotleft().trunc(r1),u.e + (u
                .e-u.s).rotleft().trunc(r1));
            Line u2 = Line(u.s + (u.e-u.s).rotright().trunc(r1),u.e + (
457
               u.e-u.s).rotright().trunc(r1));
            Line v1 = Line(v.s + (v.e-v.s).rotleft().trunc(r1), v.e + (v.e-v.s)
458
               .e-v.s).rotleft().trunc(r1));
            Line v2 = Line(v.s + (v.e-v.s).rotright().trunc(r1),v.e + (
459
               v.e-v.s).rotright().trunc(r1));
460
            c1.r = c2.r = c3.r = c4.r = r1;
461
            c1.p = u1.crosspoint(v1);
            c2.p = u1.crosspoint(v2);
462
463
            c3.p = u2.crosspoint(v1);
464
            c4.p = u2.crosspoint(v2);
465
            return 4;
466
        }
467
        //同时与不相交圆 cx,cy 相切, 半径为 r1 的圆
        //测试: UVA12304
468
        int getcircle(circle cx,circle cy,double r1,circle &c1,circle &
469
           c2){
            circle x(cx.p,r1+cx.r),y(cy.p,r1+cy.r);
470
471
            int t = x.pointcrosscircle(y,c1.p,c2.p);
472
            if(!t)return 0;
473
            c1.r = c2.r = r1;
474
            return t;
        }
475
476
        //过一点作圆的切线 (先判断点和圆的关系)
477
        //测试: UVA12304
478
479
        int tangentline(Point q,Line &u,Line &v){
            int x = relation(q);
480
```

```
if(x == 2)return 0;
481
482
             if(x == 1){
                 u = Line(q,q + (q-p).rotleft());
483
484
                 v = u;
485
                 return 1;
             }
486
            double d = p.distance(q);
487
             double l = r*r/d;
488
489
            double h = sqrt(r*r-l*l);
             u = Line(q,p + ((q-p).trunc(l) + (q-p).rotleft().trunc(h)))
490
             v = Line(q,p + ((q-p).trunc(l) + (q-p).rotright().trunc(h))
491
                );
492
             return 2;
493
        }
494
        //求两圆相交的面积
        double areacircle(circle v){
495
             int rel = relationcircle(v);
496
             if(rel >= 4)return 0.0;
497
             if(rel <= 2)return min(area(),v.area());</pre>
498
             double d = p.distance(v.p);
499
500
             double hf = (r+v.r+d)/2.0;
501
             double ss = 2*sqrt(hf*(hf-r)*(hf-v.r)*(hf-d));
             double a1 = acos((r*r+d*d-v.r*v.r)/(2.0*r*d));
502
503
             a1 = a1*r*r;
504
             double a2 = acos((v.r*v.r+d*d-r*r)/(2.0*v.r*d));
505
             a2 = a2*v.r*v.r;
506
             return a1+a2-ss;
507
        }
508
        //求圆和三角形 pab 的相交面积
        //测试: POJ3675 HDU3982 HDU2892
509
        double areatriangle(Point a, Point b){
510
511
             if(sgn((p-a)^{(p-b)}) == 0)return 0.0;
             Point q[5];
512
             int len = 0;
513
514
             q[len++] = a;
515
             Line l(a,b);
516
             Point p1,p2;
             if(pointcrossline(l,q[1],q[2])==2){
517
518
                 if(sgn((a-q[1])*(b-q[1]))<0)q[len++] = q[1];
                 if(sgn((a-q[2])*(b-q[2]))<0)q[len++] = q[2];
519
520
             }
             q[len++] = b;
521
             if(len == 4 && sgn((q[0]-q[1])*(q[2]-q[1]))>0)swap(q[1],q[0])
522
                [2]);
            double res = 0;
523
             for(int i = 0;i < len-1;i++){</pre>
524
                 if(relation(q[i])==0||relation(q[i+1])==0){
525
                     double arg = p.rad(q[i],q[i+1]);
526
527
                     res += r*r*arg/2.0;
528
                 }
```

```
529
                 else{
530
                      res += fabs((q[i]-p)^{(q[i+1]-p)}/2.0;
531
532
             }
533
             return res;
         }
534
    };
535
536
537
    /*
     * n,p Line l for each side
538
     * input(int _n)
539
                                               inputs _n size polygon
     * add(Point q)

    adds a point at end of

540
        the list
     * getline()
541

    populates line array

542
     * cmp
                                                comparision in
        convex hull order
543
     * norm()
                                               sorting in convex_hull
        order
544
     * getconvex(polygon &convex)

    returns convex hull in

        convex
     * Graham(polygon &convex)

    returns convex hull in

545
        convex
546
     * isconvex()
                                               – checks if convex
     * relationpoint(Point q)
547
                                               returns 3 if q is a
        vertex
548
                                                          2 if on a side
     *
                                                          1 if inside
549
     *
550
                                                          0 if outside
551
     * convexcut(Line u,polygon &po)

    left side of u in po

552
     * gercircumference()
                                               - returns side length
553
     * getarea()
                                               - returns area
554
     * getdir()
                                               - returns 0 for cw, 1 for
        CCW
     * getbarycentre()
                                               returns barycenter
555
556
     *
557
     */
558
    struct polygon{
559
         int n;
         Point p[maxp];
560
561
         Line l[maxp];
         void input(int _n){
562
563
             n = _n;
564
             for(int i = 0;i < n;i++)
565
                 p[i].input();
566
         void add(Point q){
567
             p[n++] = q;
568
569
         void getline(){
570
             for(int i = 0;i < n;i++){</pre>
571
                 l[i] = Line(p[i],p[(i+1)%n]);
572
```

```
}
573
574
        }
        struct cmp{
575
            Point p;
576
            cmp(const Point &p0){p = p0;}
577
            bool operator()(const Point &aa,const Point &bb){
578
                Point a = aa, b = bb;
579
                int d = sgn((a-p)^{(b-p)});
580
581
                if(d == 0){
                     return sgn(a.distance(p)-b.distance(p)) < 0;</pre>
582
583
584
                return d > 0;
            }
585
586
        };
        //进行极角排序
587
588
        //首先需要找到最左下角的点
        //需要重载号好 Point 的 < 操作符 (min 函数要用)
589
590
        void norm(){
            Point mi = p[0];
591
            for(int i = 1;i < n;i++)mi = min(mi,p[i]);</pre>
592
            sort(p,p+n,cmp(mi));
593
594
        }
        //得到凸包
595
        //得到的凸包里面的点编号是 0∼n-1 的
596
597
        //两种凸包的方法
        //注意如果有影响,要特判下所有点共点,或者共线的特殊情况
598
        //测试 Light0J1203 Light0J1239
599
        void getconvex(polygon &convex){
600
601
            sort(p,p+n);
602
            convex.n = n;
            for(int i = 0;i < min(n,2);i++){</pre>
603
                convex.p[i] = p[i];
604
605
            if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n
606
               ---;//特
               判
607
            if(n <= 2)return;</pre>
608
            int &top = convex.n;
609
            top = 1;
610
            for(int i = 2;i < n;i++){
                while(top && sgn((convex.p[top]-p[i])^(convex.p[top-1]-
611
                   p[i])) <= 0)
612
                     top--;
613
                convex.p[++top] = p[i];
614
            int temp = top;
615
            convex.p[++top] = p[n-2];
616
            for(int i = n-3; i >= 0; i---)
617
                while(top != temp && sgn((convex.p[top]-p[i])^(convex.p
618
                    [top-1]-p[i])) <= 0)
619
                     top--;
```

```
convex.p[++top] = p[i];
620
621
            }
622
            if(convex.n == 2 \&\& (convex.p[0] == convex.p[1]))convex.n
               ---;//特
               紃
            convex.norm();//原来得到的是顺时针的点,排序后逆时针
623
        }
624
        //得到凸包的另外一种方法
625
626
        //测试 LightOJ1203 LightOJ1239
        void Graham(polygon &convex){
627
628
            norm();
            int &top = convex.n;
629
            top = 0;
630
            if(n == 1){
631
632
                top = 1;
633
                convex.p[0] = p[0];
634
                return;
            }
635
            if(n == 2){
636
637
                top = 2;
                convex.p[0] = p[0];
638
639
                convex.p[1] = p[1];
640
                if(convex.p[0] == convex.p[1])top—;
641
                return;
            }
642
643
            convex.p[0] = p[0];
644
            convex.p[1] = p[1];
645
            top = 2;
646
            for(int i = 2;i < n;i++){
647
                while( top > 1 && sgn((convex.p[top-1]-convex.p[top-2])
                    ^(p[i]-convex.p[top-2])) <= 0 )
648
                     top--;
                convex.p[top++] = p[i];
649
650
651
            if(convex.n == 2 && (convex.p[0] == convex.p[1]))convex.n
                 -;//特
               剕
652
        }
        //判断是不是凸的
653
654
        bool isconvex(){
            bool s[2];
655
            memset(s, false, sizeof(s));
656
            for(int i = 0;i < n;i++){
657
                int j = (i+1)%n;
658
                int k = (j+1)%n;
659
                s[sgn((p[j]-p[i])^(p[k]-p[i]))+1] = true;
660
                if(s[0] && s[2])return false;
661
            }
662
663
            return true;
664
        //判断点和任意多边形的关系
665
```

```
// 3 点上
666
         // 2 边上
667
         // 1 内部
668
         // 0 外部
669
         int relationpoint(Point q){
670
             for(int i = 0;i < n;i++){
671
                 if(p[i] == q)return 3;
672
673
             }
674
             getline();
             for(int i = 0;i < n;i++){
675
676
                 if(l[i].pointonseg(q))return 2;
677
             int cnt = 0;
678
             for(int i = 0;i < n;i++){</pre>
679
680
                 int j = (i+1)\%n;
681
                 int k = sgn((q-p[j])^(p[i]-p[j]));
682
                 int u = sgn(p[i].y-q.y);
                 int v = sgn(p[j].y-q.y);
683
                 if(k > 0 \&\& u < 0 \&\& v >= 0)cnt++;
684
                 if(k < 0 \&\& v < 0 \&\& u >= 0)cnt—;
685
686
687
             return cnt != 0;
688
         }
         //直线 u 切割凸多边形左侧
689
690
         //注意直线方向
         //测试: HDU3982
691
         void convexcut(Line u,polygon &po){
692
             int &top = po.n;//注意引用
693
694
             top = 0;
695
             for(int i = 0;i < n;i++){
                 int d1 = sgn((u.e-u.s)^(p[i]-u.s));
696
                 int d2 = sgn((u.e-u.s)^{(p[(i+1)\%n]-u.s));
697
                 if(d1 >= 0)po.p[top++] = p[i];
698
                 if(d1*d2 < 0)po.p[top++] = u.crosspoint(Line(p[i],p[(i</pre>
699
                    +1)%n]));
700
             }
701
         }
         //得到周长
702
         //测试 Light0J1239
703
704
         double getcircumference(){
             double sum = 0;
705
706
             for(int i = 0;i < n;i++){
707
                 sum += p[i].distance(p[(i+1)%n]);
708
             }
709
             return sum;
710
         }
         //得到面积
711
         double getarea(){
712
             double sum = 0;
713
714
             for(int i = 0;i < n;i++){
715
                 sum += (p[i]^p[(i+1)%n]);
```

```
716
            }
717
            return fabs(sum)/2;
        }
718
        //得到方向
719
        // 1 表示逆时针, 0 表示顺时针
720
        bool getdir(){
721
            double sum = 0;
722
            for(int i = 0;i < n;i++)
723
724
                 sum += (p[i]^p[(i+1)%n]);
            if(sgn(sum) > 0)return 1;
725
726
            return 0;
        }
727
        //得到重心
728
729
        Point getbarycentre(){
            Point ret(0,0);
730
731
            double area = 0;
            for(int i = 1;i < n-1;i++){
732
733
                double tmp = (p[i]-p[0])^{(p[i+1]-p[0])};
                if(sgn(tmp) == 0)continue;
734
                area += tmp;
735
                 ret.x += (p[0].x+p[i].x+p[i+1].x)/3*tmp;
736
737
                 ret.y += (p[0].y+p[i].y+p[i+1].y)/3*tmp;
738
            if(sgn(area)) ret = ret/area;
739
740
            return ret;
        }
741
        //多边形和圆交的面积
742
        //测试: POJ3675 HDU3982 HDU2892
743
744
        double areacircle(circle c){
745
            double ans = 0;
            for(int i = 0;i < n;i++){</pre>
746
                int j = (i+1)%n;
747
                 if(sgn( (p[j]-c.p)^(p[i]-c.p) ) >= 0)
748
                     ans += c.areatriangle(p[i],p[j]);
749
750
                else ans -= c.areatriangle(p[i],p[j]);
751
            }
752
            return fabs(ans);
753
        }
        //多边形和圆关系
754
        // 2 圆完全在多边形内
755
        // 1 圆在多边形里面,碰到了多边形边界
756
        // 0 其它
757
        int relationcircle(circle c){
758
            getline();
759
            int x = 2;
760
            if(relationpoint(c.p) != 1)return 0;//圆心不在内部
761
            for(int i = 0;i < n;i++){</pre>
762
                if(c.relationseg(l[i])==2)return 0;
763
                if(c.relationseg(l[i])==1)x = 1;
764
765
766
            return x;
```

```
767
        }
768 | };
769
    //AB X AC
770
    double cross(Point A,Point B,Point C){
771
        return (B-A)^{(C-A)};
772
    //AB*AC
773
    double dot(Point A,Point B,Point C){
774
775
        return (B-A)*(C-A);
776
    //最小矩形面积覆盖
777
778
    // A 必须是凸包 (而且是逆时针顺序)
    // 测试 UVA 10173
779
    double minRectangleCover(polygon A){
780
781
        //要特判 A.n < 3 的情况
        if(A.n < 3)return 0.0;
782
        A.p[A.n] = A.p[0];
783
        double ans = -1;
784
        int r = 1, p = 1, q;
785
        for(int i = 0;i < A.n;i++){
786
            //卡出离边 A.p[i] - A.p[i+1] 最远的点
787
788
            while( sgn( cross(A.p[i],A.p[i+1],A.p[r+1]) - cross(A.p[i],
               A.p[i+1], A.p[r]) >= 0
                r = (r+1)%A.n;
789
790
            //卡出 A.p[i] - A.p[i+1] 方向上正向 n 最远的点
791
            while(sgn( dot(A.p[i],A.p[i+1],A.p[p+1]) - dot(A.p[i],A.p[i
               +1],A.p[p]) >= 0)
                p = (p+1)%A.n;
792
793
            if(i == 0)q = p;
            //卡出 A.p[i] - A.p[i+1] 方向上负向最远的点
794
            while(sgn(dot(A.p[i],A.p[i+1],A.p[q+1]) - dot(A.p[i],A.p[i
795
               +1],A.p[q])) <= 0)
796
                q = (q+1)%A.n;
797
            double d = (A.p[i] - A.p[i+1]).len2();
            double tmp = cross(A.p[i],A.p[i+1],A.p[r]) *
798
799
                (dot(A.p[i],A.p[i+1],A.p[p]) - dot(A.p[i],A.p[i+1],A.p[
                   q]))/d;
800
            if(ans < 0 \mid | ans > tmp)ans = tmp;
801
        }
802
        return ans;
    }
803
804
805
    //直线切凸多边形
    //多边形是逆时针的, 在 q1q2 的左侧
806
    //测试:HDU3982
807
    vector<Point> convexCut(const vector<Point> &ps,Point q1,Point q2){
808
        vector<Point>qs:
809
        int n = ps.size();
810
        for(int i = 0;i < n;i++){
811
812
            Point p1 = ps[i], p2 = ps[(i+1)%n];
            int d1 = sgn((q2-q1)^{(p1-q1)}), d2 = sgn((q2-q1)^{(p2-q1)});
813
```

```
814
             if(d1 >= 0)
815
                 qs.push_back(p1);
             if(d1 * d2 < 0)
816
                 qs.push_back(Line(p1,p2).crosspoint(Line(q1,q2)));
817
818
        }
819
        return qs;
820
    //半平面交
821
822
    //测试 POJ3335 POJ1474 POJ1279
    //********
823
    struct halfplane:public Line{
824
825
        double angle;
        halfplane(){}
826
        //表示向量 s->e 逆时针 (左侧) 的半平面
827
828
        halfplane(Point _s,Point _e){
829
             s = _s;
830
             e = _e;
        }
831
832
        halfplane(Line v){
833
            s = v.s;
834
             e = v.e;
835
        }
836
        void calcangle(){
             angle = atan2(e.y-s.y,e.x-s.x);
837
838
        bool operator <(const halfplane &b)const{</pre>
839
             return angle < b.angle;</pre>
840
        }
841
842
    };
843
    struct halfplanes{
844
        int n;
845
        halfplane hp[maxp];
        Point p[maxp];
846
        int que[maxp];
847
        int st,ed;
848
849
        void push(halfplane tmp){
850
             hp[n++] = tmp;
        }
851
        //去重
852
        void unique(){
853
             int m = 1;
854
             for(int i = 1;i < n;i++){</pre>
855
                 if(sgn(hp[i].angle-hp[i-1].angle) != 0)
856
                     hp[m++] = hp[i];
857
                 else if(sgn( (hp[m-1].e-hp[m-1].s)^(hp[i].s-hp[m-1].s)
858
                    ) > 0
                     hp[m-1] = hp[i];
859
             }
860
861
             n = m;
862
        bool halfplaneinsert(){
863
```

```
for(int i = 0;i < n;i++)hp[i].calcangle();</pre>
864
865
            sort(hp,hp+n);
            unique();
866
            que[st=0] = 0;
867
            que[ed=1] = 1;
868
            p[1] = hp[0].crosspoint(hp[1]);
869
            for(int i = 2;i < n;i++){
870
                 while(st<ed && sgn((hp[i].e-hp[i].s)^(p[ed]-hp[i].s))</pre>
871
                    <0)ed—;
                 while(st<ed && sgn((hp[i].e-hp[i].s)^(p[st+1]-hp[i].s))</pre>
872
                    <0)st++;
                 que[++ed] = i;
873
                 if(hp[i].parallel(hp[que[ed-1]]))return false;
874
                 p[ed]=hp[i].crosspoint(hp[que[ed-1]]);
875
876
            }
877
            while(st<ed && sgn((hp[que[st]].e-hp[que[st]].s)^(p[ed]-hp[</pre>
               que[st]].s))<0)ed---;
            while(st<ed && sgn((hp[que[ed]].e-hp[que[ed]].s)^(p[st+1]-</pre>
878
               hp[que[ed]].s))<0)st++;
            if(st+1>=ed)return false;
879
            return true;
880
881
        }
        //得到最后半平面交得到的凸多边形
882
        //需要先调用 halfplaneinsert() 且返回 true
883
        void getconvex(polygon &con){
884
            p[st] = hp[que[st]].crosspoint(hp[que[ed]]);
885
            con.n = ed-st+1;
886
            for(int j = st,i = 0;j <= ed;i++,j++)</pre>
887
888
                 con.p[i] = p[j];
889
        }
890
891
    //*********
892
893
    const int maxn = 1010;
894
    struct circles{
895
        circle c[maxn];
896
        double ans[maxn];//ans[i] 表示被覆盖了 i 次的面积
        double pre[maxn];
897
        int n;
898
899
        circles(){}
        void add(circle cc){
900
            c[n++] = cc;
901
        }
902
        //x 包含在 y 中
903
        bool inner(circle x,circle y){
904
            if(x.relationcircle(y) != 1)return 0;
905
            return sgn(x.r-y.r)<=0?1:0;
906
907
        //圆的面积并去掉内含的圆
908
909
        void init or(){
            bool mark[maxn] = {0};
910
```

```
int i,j,k=0;
911
912
            for(i = 0;i < n;i++){
                for(j = 0;j < n;j++)
913
914
                    if(i != j && !mark[j]){
                        if( (c[i]==c[j])||inner(c[i],c[j]) )break;
915
916
                if(j < n) mark[i] = 1;
917
918
919
            for(i = 0;i < n;i++)
                if(!mark[i])
920
921
                    c[k++] = c[i];
            n = k;
922
923
        }
        //圆的面积交去掉内含的圆
924
        void init_add(){
925
926
            int i,j,k;
            bool mark[maxn] = {0};
927
            for(i = 0;i < n;i++){
928
                for(j = 0; j < n; j++)
929
                    if(i != j && !mark[j]){
930
                        if( (c[i]==c[j])||inner(c[j],c[i]) )break;
931
932
933
                if(j < n)mark[i] = 1;
934
            for(i = 0;i < n;i++)
935
                if(!mark[i])
936
937
                    c[k++] = c[i];
938
            n = k;
939
        }
940
        //半径为 r 的圆, 弧度为 th 对应的弓形的面积
        double areaarc(double th,double r){
941
            return 0.5*r*r*(th-sin(th));
942
        }
943
        //测试 SPOJVCIRCLES SPOJCIRUT
944
        //SPOJVCIRCLES 求 n 个圆并的面积,需要加上 init or() 去掉重复圆(否则
945
           WA)
946
        //SPOJCIRUT 是求被覆盖 k 次的面积,不能加 init_or()
        //对于求覆盖多少次面积的问题,不能解决相同圆,而且不能 init or()
947
        //求多圆面积并,需要 init_or, 其中一个目的就是去掉相同圆
948
949
        void getarea(){
            memset(ans,0,sizeof(ans));
950
951
            vector<pair<double,int> >v;
            for(int i = 0;i < n;i++){
952
                v.clear();
953
                v.push_back(make_pair(-pi,1));
954
955
                v.push_back(make_pair(pi,-1));
                for(int j = 0; j < n; j++)
956
                    if(i != j){
957
958
                        Point q = (c[j].p - c[i].p);
959
                        double ab = q.len(),ac = c[i].r, bc = c[j].r;
960
                        if(sgn(ab+ac-bc)<=0){
```

```
961
                              v.push_back(make_pair(-pi,1));
962
                              v.push_back(make_pair(pi,-1));
                              continue;
963
964
                          }
                          if(sgn(ab+bc-ac)<=0)continue;</pre>
965
                          if(sgn(ab-ac-bc)>0)continue;
966
                          double th = atan2(q.y,q.x), fai = acos((ac*ac+
967
                             ab*ab-bc*bc)/(2.0*ac*ab));
968
                          double a0 = th—fai;
                          if(sgn(a0+pi)<0)a0+=2*pi;
969
                          double a1 = th+fai;
970
                          if(sgn(a1-pi)>0)a1-=2*pi;
971
                          if(sgn(a0-a1)>0){
972
                              v.push_back(make_pair(a0,1));
973
                              v.push_back(make_pair(pi,-1));
974
975
                              v.push_back(make_pair(-pi,1));
                              v.push_back(make_pair(a1,-1));
976
977
                          }
978
                          else{
                              v.push_back(make_pair(a0,1));
979
                              v.push_back(make_pair(a1,-1));
980
981
                          }
                      }
982
                 sort(v.begin(),v.end());
983
984
                 int cur = 0;
                 for(int j = 0; j < v.size(); j++){</pre>
985
                      if(cur && sgn(v[j].first-pre[cur])){
986
                          ans[cur] += areaarc(v[j].first-pre[cur],c[i].r)
987
988
                          ans[cur] += 0.5*(Point(c[i].p.x+c[i].r*cos(pre[
                             cur]),c[i].p.y+c[i].r*sin(pre[cur]))^Point(c
                             [i].p.x+c[i].r*cos(v[j].first),c[i].p.y+c[i
                             ].r*sin(v[j].first)));
989
                      }
                      cur += v[j].second;
990
991
                      pre[cur] = v[j].first;
992
                 }
993
             for(int i = 1;i < n;i++)</pre>
994
995
                 ans[i] = ans[i+1];
996
         }
997 | };
    7.2
         三维几何
  1 | const double eps = 1e-8;
    int sgn(double x){
  2
  3
         if(fabs(x) < eps)return 0;</pre>
  4
         if(x < 0) return -1;
  5
         else return 1;
  6
    struct Point3{
```

```
8
        double x,y,z;
 9
        Point3(double _x = 0, double _y = 0, double _z = 0)
10
            x = _x;
11
            y = _y;
            z = _z;
12
13
        void input(){
14
15
            scanf("%lf%lf%lf",&x,&y,&z);
16
        void output(){
17
18
            scanf("\%.2lf_{\square}\%.2lf_{\square}\%.2lf_{\square}",x,y,z);
19
20
        bool operator ==(const Point3 &b)const{
21
            return sgn(x-b.x) == 0 \&\& sgn(y-b.y) == 0 \&\& sgn(z-b.z) ==
               0;
22
23
        bool operator <(const Point3 &b)const{</pre>
24
            return sgn(x-b.x)==0?(sgn(y-b.y)==0?sgn(z-b.z)<0:y<b.y):x<b
               .x;
25
        double len(){
26
27
            return sqrt(x*x+y*y+z*z);
28
        double len2(){
29
30
            return x*x+y*y+z*z;
31
        double distance(const Point3 &b)const{
32
            return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y)+(z-b.z)*(z-b.z)
33
               );
34
35
        Point3 operator -(const Point3 &b)const{
36
            return Point3(x-b.x,y-b.y,z-b.z);
37
38
        Point3 operator +(const Point3 &b)const{
39
            return Point3(x+b.x,y+b.y,z+b.z);
40
41
        Point3 operator *(const double &k)const{
42
            return Point3(x*k,y*k,z*k);
43
44
        Point3 operator /(const double &k)const{
45
            return Point3(x/k,y/k,z/k);
46
47
        //点乘
        double operator *(const Point3 &b)const{
48
49
            return x*b.x+y*b.y+z*b.z;
50
        }
        //叉乘
51
52
        Point3 operator ^(const Point3 &b)const{
53
            return Point3(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
54
55
        double rad(Point3 a,Point3 b){
```

```
56
             Point3 p = (*this);
 57
             return acos((a-p)*(b-p))/(a.distance(p)*b.distance(p))
 58
        }
 59
         //变换长度
        Point3 trunc(double r){
 60
             double l = len();
 61
             if(!sgn(l))return *this;
 62
 63
             r /= l;
 64
             return Point3(x*r,y*r,z*r);
        }
 65
 66
    };
    struct Line3
 67
 68
    {
 69
        Point3 s,e;
 70
        Line3(){}
 71
        Line3(Point3 _s,Point3 _e)
 72
        {
 73
             s = _s;
 74
             e = _e;
 75
 76
        bool operator ==(const Line3 v)
 77
             return (s==v.s)&&(e==v.e);
 78
 79
        }
        void input()
 80
 81
         {
 82
             s.input();
 83
             e.input();
 84
        }
 85
        double length()
 86
        {
 87
             return s.distance(e);
 88
        }
 89
         //点到直线距离
 90
        double dispointtoline(Point3 p)
 91
        {
             return ((e-s)^(p-s)).len()/s.distance(e);
 92
 93
        }
         //点到线段距离
 94
 95
        double dispointtoseg(Point3 p)
 96
        {
 97
             if(sgn((p-s)*(e-s)) < 0 \mid | sgn((p-e)*(s-e)) < 0)
 98
                 return min(p.distance(s),e.distance(p));
 99
             return dispointtoline(p);
100
        }
         //返回点 p 在直线上的投影
101
        Point3 lineprog(Point3 p)
102
103
        {
             return s + (((e-s)*((e-s)*(p-s)))/((e-s).len2()));
104
105
        }
```

```
//p 绕此向量逆时针 arg 角度
106
        Point3 rotate(Point3 p,double ang)
107
108
             if(sgn(((s-p)^(e-p)).len()) == 0)return p;
109
            Point3 f1 = (e-s)^{(p-s)};
110
            Point3 f2 = (e-s)^{(f1)};
111
            double len = ((s-p)^(e-p)).len()/s.distance(e);
112
             f1 = f1.trunc(len); f2 = f2.trunc(len);
113
114
            Point3 h = p+f2;
            Point3 pp = h+f1;
115
             return h + ((p-h)*cos(ang)) + ((pp-h)*sin(ang));
116
        }
117
        //点在直线上
118
        bool pointonseg(Point3 p)
119
120
        {
121
             return sgn((s-p)^{(e-p)}).len()) == 0 && <math>sgn((s-p)*(e-p))
               == 0;
        }
122
123
    };
    struct Plane
124
125
    {
126
        Point3 a,b,c,o;//平面上的三个点,以及法向量
127
        Plane(){}
        Plane(Point3 _a,Point3 _b,Point3 _c)
128
129
        {
130
            a = _a;
131
            b = b;
            c = _c;
132
133
            o = pvec();
134
        }
135
        Point3 pvec()
136
        {
             return (b-a)^{(c-a)};
137
138
        }
        //ax+by+cz+d = 0
139
140
        Plane(double _a,double _b,double _c,double _d)
141
142
            o = Point3(_a,_b,_c);
143
             if(sgn(_a) != 0)
144
                 a = Point3((-_d-_c-_b)/_a,1,1);
            else if(sgn(b)!= 0)
145
146
                 a = Point3(1,(-_d-_c-_a)/_b,1);
            else if(sgn(_c) != 0)
147
148
                 a = Point3(1,1,(-_d-_a-_b)/_c);
149
        //点在平面上的判断
150
151
        bool pointonplane(Point3 p)
152
        {
             return sgn((p-a)*o) == 0;
153
154
        //两平面夹角
155
```

```
156
        double angleplane(Plane f)
157
        {
            return acos(o*f.o)/(o.len()*f.o.len());
158
159
        }
        //平面和直线的交点,返回值是交点个数
160
        int crossline(Line3 u,Point3 &p)
161
162
            double x = o*(u.e-a);
163
164
            double y = o*(u.s-a);
            double d = x-y;
165
            if(sgn(d) == 0)return 0;
166
            p = ((u.s*x)-(u.e*y))/d;
167
            return 1;
168
        }
169
        //点到平面最近点 (也就是投影)
170
171
        Point3 pointtoplane(Point3 p)
172
            Line3 u = Line3(p,p+o);
173
174
            crossline(u,p);
175
            return p;
176
        }
177
        //平面和平面的交线
178
        int crossplane(Plane f,Line3 &u)
179
        {
            Point3 oo = o^f.o;
180
            Point3 v = o^o;
181
            double d = fabs(f.o*v);
182
            if(sgn(d) == 0)return 0;
183
184
            Point3 q = a + (v*(f.o*(f.a-a))/d);
185
            u = Line3(q,q+oo);
186
            return 1;
        }
187
188 | };
    7.3 平面最近点对
    HDU1007/ZOJ2107
  1 | const int MAXN = 100010;
  2 | const double eps = 1e-8;
    const double INF = 1e20;
  4
    struct Point{
        double x,y;
  5
  6
        void input(){
  7
            scanf("%lf%lf",&x,&y);
  8
        }
  9
    };
    double dist(Point a, Point b){
 10
 11
        return sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
    }
 12
 13 | Point p[MAXN];
    Point tmpt[MAXN];
 15 | bool cmpx(Point a, Point b) {
```

```
return a.x < b.x || (a.x == b.x && a.y < b.y);
16
17
18
   bool cmpy(Point a, Point b){
19
       return a.y < b.y || (a.y == b.y && a.x < b.x);
20
21
   double Closest_Pair(int left,int right){
22
       double d = INF;
23
       if(left == right)return d;
24
       if(left+1 == right)return dist(p[left],p[right]);
25
       int mid = (left+right)/2;
       double d1 = Closest_Pair(left,mid);
26
27
       double d2 = Closest_Pair(mid+1,right);
       d = min(d1,d2);
28
29
       int cnt = 0;
       for(int i = left;i <= right;i++){</pre>
30
31
            if(fabs(p[mid].x - p[i].x) <= d)
                tmpt[cnt++] = p[i];
32
33
       }
34
       sort(tmpt,tmpt+cnt,cmpy);
35
       for(int i = 0;i < cnt;i++){
            for(int j = i+1; j < cnt && tmpt[j].y - tmpt[i].y < d; j++)</pre>
36
                d = min(d,dist(tmpt[i],tmpt[j]));
37
38
39
       return d;
40
   int main(){
41
42
       int n;
43
       while(scanf("%d",&n) == 1 && n){
44
            for(int i = 0;i < n;i++)p[i].input();
45
            sort(p,p+n,cmpx);
            printf("%.2lf\n",Closest_Pair(0,n-1));
46
47
48
       return 0;
49 }
        三维凸包
   7.4
   7.4.1 HDU4273
   HDU 4273 给一个三维凸包, 求重心到表面的最短距离。
 1 | const double eps = 1e-8;
 2
   const int MAXN = 550;
 3
   int sgn(double x){
       if(fabs(x) < eps)return 0;</pre>
 4
 5
       if(x < 0) return -1;
 6
       else return 1;
 7
   }
 8
   struct Point3{
 9
       double x,y,z;
       Point3(double _x = 0, double _y = 0, double _z = 0)
10
11
           x = _x;
```

```
12
           y = _y;
13
           z = _z;
14
15
       void input(){
           scanf("%lf%lf%lf",&x,&y,&z);
16
17
       bool operator ==(const Point3 &b)const{
18
            return sgn(x-b.x) == 0 \&\& sgn(y-b.y) == 0 \&\& sgn(z-b.z) ==
19
20
       }
       double len(){
21
22
            return sqrt(x*x+y*y+z*z);
23
24
       double len2(){
            return x*x+y*y+z*z;
25
26
       double distance(const Point3 &b)const{
27
28
            return sqrt((x-b.x)*(x-b.x)+(y-b.y)*(y-b.y)+(z-b.z)*(z-b.z)
              );
29
30
       Point3 operator -(const Point3 &b)const{
31
            return Point3(x-b.x,y-b.y,z-b.z);
32
       Point3 operator +(const Point3 &b)const{
33
            return Point3(x+b.x,y+b.y,z+b.z);
34
35
36
       Point3 operator *(const double &k)const{
            return Point3(x*k,y*k,z*k);
37
38
39
       Point3 operator /(const double &k)const{
40
            return Point3(x/k,y/k,z/k);
41
       }
       //点乘
42
43
       double operator *(const Point3 &b)const{
44
            return x*b.x + y*b.y + z*b.z;
45
       }
46
       //叉乘
47
       Point3 operator ^(const Point3 &b)const{
48
            return Point3(y*b.z-z*b.y,z*b.x-x*b.z,x*b.y-y*b.x);
       }
49
50
   };
   struct CH3D{
51
       struct face{
52
53
            //表示凸包一个面上的三个点的编号
54
            int a,b,c;
            //表示该面是否属于最终的凸包上的面
55
56
            bool ok;
57
       };
       //初始顶点数
58
59
       int n;
60
       Point3 P[MAXN];
```

```
61
        //凸包表面的三角形数
 62
        int num;
        //凸包表面的三角形
 63
 64
        face F[8*MAXN];
        int g[MAXN][MAXN];
 65
        //叉乘
 66
 67
        Point3 cross(const Point3 &a,const Point3 &b,const Point3 &c){
            return (b-a)^{(c-a)};
 68
 69
        }
        //三角形面积 *2
 70
71
        double area(Point3 a,Point3 b,Point3 c){
 72
            return ((b-a)^(c-a)).len();
73
        }
        //四面体有向面积 *6
 74
        double volume(Point3 a,Point3 b,Point3 c,Point3 d){
 75
 76
            return ((b-a)^{(c-a)})*(d-a);
        }
 77
        //正: 点在面同向
78
        double dblcmp(Point3 &p,face &f){
 79
            Point3 p1 = P[f.b] - P[f.a];
 80
            Point3 p2 = P[f.c] - P[f.a];
 81
 82
            Point3 p3 = p - P[f.a];
 83
            return (p1^p2)*p3;
 84
 85
        void deal(int p,int a,int b){
            int f = g[a][b];
 86
            face add;
 87
            if(F[f].ok){
 88
 89
                 if(dblcmp(P[p],F[f]) > eps)
                     dfs(p,f);
90
91
                 else {
 92
                     add.a = b;
 93
                     add.b = a;
 94
                     add.c = p;
95
                     add.ok = true;
 96
                     g[p][b] = g[a][p] = g[b][a] = num;
97
                     F[num++] = add;
98
                 }
99
            }
        }
100
        //递归搜索所有应该从凸包内删除的面
101
        void dfs(int p,int now){
102
            F[now].ok = false;
103
            deal(p,F[now].b,F[now].a);
104
            deal(p,F[now].c,F[now].b);
105
            deal(p,F[now].a,F[now].c);
106
107
        bool same(int s,int t){
108
            Point3 &a = P[F[s].a];
109
110
            Point3 &b = P[F[s].b];
111
            Point3 &c = P[F[s].c];
```

```
112
            return fabs(volume(a,b,c,P[F[t].a])) < eps &&</pre>
113
                 fabs(volume(a,b,c,P[F[t].b])) < eps &&</pre>
                 fabs(volume(a,b,c,P[F[t].c])) < eps;</pre>
114
115
        }
        //构建三维凸包
116
        void create(){
117
118
             num = 0;
119
             face add;
120
121
             //*********
122
             //此段是为了保证前四个点不共面
             bool flag = true;
123
             for(int i = 1;i < n;i++){</pre>
124
                 if(!(P[0] == P[i])){
125
                     swap(P[1],P[i]);
126
127
                     flag = false;
128
                     break;
129
                 }
130
             if(flag)return;
131
             flag = true;
132
133
             for(int i = 2;i < n;i++){</pre>
134
                 if( ((P[1]-P[0])^(P[i]-P[0])).len() > eps ){
                     swap(P[2],P[i]);
135
                     flag = false;
136
137
                     break;
138
                 }
139
140
             if(flag)return;
141
             flag = true;
             for(int i = 3;i < n;i++){</pre>
142
                 if(fabs( ((P[1]-P[0])^{(P[2]-P[0]))*(P[i]-P[0]) > eps)
143
                     swap(P[3],P[i]);
144
                     flag = false;
145
146
                     break;
147
                 }
148
             if(flag)return;
149
150
             //*********
151
             for(int i = 0;i < 4;i++){
152
                 add.a = (i+1)\%4;
153
                 add.b = (i+2)\%4;
154
155
                 add.c = (i+3)\%4;
                 add.ok = true;
156
                 if(dblcmp(P[i],add) > 0)swap(add.b,add.c);
157
                 g[add.a][add.b] = g[add.b][add.c] = g[add.c][add.a] =
158
                    num;
159
                 F[num++] = add;
             }
160
```

```
161
             for(int i = 4;i < n;i++)</pre>
162
                  for(int j = 0; j < num; j++)</pre>
                      if(F[j].ok && dblcmp(P[i],F[j]) > eps){
163
164
                           dfs(i,j);
                           break;
165
                      }
166
167
             int tmp = num;
168
             num = 0;
169
             for(int i = 0;i < tmp;i++)</pre>
                  if(F[i].ok)
170
                      F[num++] = F[i];
171
         }
172
         //表面积
173
         //测试: HDU3528
174
         double area(){
175
176
             double res = 0;
             if(n == 3){
177
                  Point3 p = cross(P[0], P[1], P[2]);
178
179
                  return p.len()/2;
180
             for(int i = 0;i < num;i++)</pre>
181
182
                  res += area(P[F[i].a],P[F[i].b],P[F[i].c]);
183
             return res/2.0;
         }
184
         double volume(){
185
186
             double res = 0;
             Point3 tmp = Point3(0,0,0);
187
             for(int i = 0;i < num;i++)</pre>
188
189
                  res += volume(tmp,P[F[i].a],P[F[i].b],P[F[i].c]);
190
             return fabs(res/6);
         }
191
         //表面三角形个数
192
193
         int triangle(){
194
             return num;
         }
195
196
         //表面多边形个数
         //测试: HDU3662
197
         int polygon(){
198
             int res = 0;
199
             for(int i = 0;i < num;i++){</pre>
200
                  bool flag = true;
201
                  for(int j = 0; j < i; j++)</pre>
202
203
                      if(same(i,j)){
204
                           flag = 0;
205
                           break;
206
207
                  res += flag;
             }
208
209
             return res;
210
         }
         //重心
211
```

```
//测试: HDU4273
212
213
         Point3 barycenter(){
214
             Point3 ans = Point3(0,0,0);
             Point3 o = Point3(0,0,0);
215
             double all = 0;
216
             for(int i = 0;i < num;i++){</pre>
217
                 double vol = volume(o,P[F[i].a],P[F[i].b],P[F[i].c]);
218
                 ans = ans + (((o+P[F[i].a]+P[F[i].b]+P[F[i].c])/4.0)*
219
                    vol);
                 all += vol;
220
             }
221
222
             ans = ans/all;
223
             return ans;
224
         //点到面的距离
225
         //测试: HDU4273
226
         double ptoface(Point3 p,int i){
227
             double tmp1 = fabs(volume(P[F[i].a],P[F[i].b],P[F[i].c],p))
228
             double tmp2 = ((P[F[i].b]-P[F[i].a])^(P[F[i].c]-P[F[i].a]))
229
                .len();
230
             return tmp1/tmp2;
231
         }
232
    };
233
    CH3D hull;
234
    int main()
235
    {
236
         while(scanf("%d",&hull.n) == 1){
             for(int i = 0;i < hull.n;i++)hull.P[i].input();</pre>
237
             hull.create();
238
             Point3 p = hull.barycenter();
239
240
             double ans = 1e20;
241
             for(int i = 0;i < hull.num;i++)</pre>
                 ans = min(ans,hull.ptoface(p,i));
242
243
             printf("%.3lf\n",ans);
244
         }
245
         return 0;
246 }
```

## 8 其他

## 8.1 高精度

高精度,支持乘法和加法

```
1
 2
   * 高精度,支持乘法和加法
 3
    */
 4
   struct BigInt{
        const static int mod = 10000;
 5
 6
        const static int DLEN = 4;
 7
        int a[600],len;
 8
        BigInt(){
 9
            memset(a,0,sizeof(a));
10
            len = 1;
11
        }
        BigInt(int v){
12
            memset(a,0,sizeof(a));
13
14
            len = 0;
15
            do{
16
                a[len++] = v mod;
                v /= mod;
17
            }while(v);
18
19
        }
20
        BigInt(const char s[]){
21
            memset(a,0,sizeof(a));
22
            int L = strlen(s);
23
            len = L/DLEN;
            if(L%DLEN)len++;
24
            int index = 0;
25
            for(int i = L-1;i >= 0;i -= DLEN){
26
                int t = 0;
27
                int k = i - DLEN + 1;
28
29
                if(k < 0)k = 0;
                for(int j = k; j <= i; j++)
30
                     t = t*10 + s[j] - '0';
31
32
                a[index++] = t;
33
            }
34
35
        BigInt operator +(const BigInt &b)const{
36
            BigInt res;
            res.len = max(len,b.len);
37
            for(int i = 0;i <= res.len;i++)</pre>
38
39
                res.a[i] = 0;
            for(int i = 0;i < res.len;i++){</pre>
40
41
                res.a[i] += ((i < len)?a[i]:0)+((i < b.len)?b.a[i]:0);
42
                res.a[i+1] += res.a[i]/mod;
43
                res.a[i] %= mod;
44
45
            if(res.a[res.len] > 0)res.len++;
46
            return res;
```

```
47
        }
48
        BigInt operator *(const BigInt &b)const{
49
            BigInt res;
            for(int i = 0; i < len;i++){</pre>
50
                int up = 0;
51
                for(int j = 0; j < b.len; j++){</pre>
52
                     int temp = a[i]*b.a[j] + res.a[i+j] + up;
53
                     res.a[i+j] = temp%mod;
54
55
                     up = temp/mod;
56
                if(up != 0)
57
                     res.a[i + b.len] = up;
58
59
60
            res.len = len + b.len;
            while (res.a[res.len - 1] == 0 \& res.len > 1) res.len—;
61
62
            return res;
63
        void output(){
64
            printf("%d",a[len-1]);
65
            for(int i = len-2;i >=0 ;i--)
66
                printf("%04d",a[i]);
67
68
            printf("\n");
69
        }
70
   |};
        完全高精度
   8.2
   HDU 1134 求卡特兰数
 1
 2
    * 完全大数模板
 3
```

```
* 输入 cin>>a
   * 输出 a.print();
   * 注意这个输入不能自动去掉前导 0 的,可以先读入到 char 数组,去掉前导 0,再用
5
      构造函数。
6
  */
  #define MAXN 9999
7
8
  #define MAXSIZE 1010
  #define DLEN 4
9
  class BigNum{
10
  private:
11
      int a[500];
                  //可以控制大数的位数
12
      int len;
13
14
  public:
15
      BigNum(){len=1;memset(a,0,sizeof(a));} //构造函数
                       //将一个 int 类型的变量转化成大数
      BigNum(const int);
16
                          //将一个字符串类型的变量转化为大数
      BigNum(const char*);
17
      BigNum(const BigNum &); //拷贝构造函数
18
      BigNum & operator = (const BigNum &); //重载赋值运算符, 大数之间进行赋
19
         值运算
20
      friend istream& operator>>(istream&,BigNum&); //重载输入运算符
21
      friend ostream& operator<<(ostream&,BigNum&); //重载输出运算符
```

```
22
23
       BigNum operator+(const BigNum &)const;
                                             //重载加法运算符,两个大数
         之间的相加运算
       BigNum operator—(const BigNum &)const;
24
                                             //重载减法运算符,两个大数
         之间的相减运算
       BigNum operator*(const BigNum &)const;
                                             //重载乘法运算符,两个大数
25
         之间的相乘运算
       BigNum operator/(const int &)const;
                                             //重载除法运算符,大数对一
26
         个整数进行相除运算
27
       BigNum operator^(const int &)const;
                                             //大数的 n 次方运算
28
29
       int operator%(const int &)const;
                                             //大数对一个类型的变量进行
         取模运算int
       bool operator>(const BigNum &T)const;
                                             //大数和另一个大数的大小比
30
31
       bool operator>(const int &t)const;
                                             //大数和一个 int 类型的变
         量的大小比较
32
33
                           //输出大数
       void print();
34
   };
   //将一个 int 类型的变量转化为大数
35
36
   BigNum::BigNum(const int b){
37
       int c,d=b;
       len=0;
38
39
       memset(a,0,sizeof(a));
40
       while(d>MAXN){
41
           c=d-(d/(MAXN+1))*(MAXN+1);
42
           d=d/(MAXN+1);
43
          a[len++]=c;
44
       }
45
       a[len++]=d;
46
   //将一个字符串类型的变量转化为大数
47
48
   BigNum::BigNum(const char *s){
49
       int t,k,index,L,i;
50
       memset(a,0,sizeof(a));
51
       L=strlen(s);
52
       len=L/DLEN;
       if(L%DLEN)len++;
53
54
       index=0;
55
       for(i=L-1;i>=0;i-=DLEN){
56
           t=0;
57
          k=i-DLEN+1;
58
           if(k<0)k=0;
59
           for(int j=k;j<=i;j++)
              t=t*10+s[j]-'0';
60
           a[index++]=t;
61
       }
62
63
64
   //拷贝构造函数
65
   BigNum::BigNum(const BigNum &T):len(T.len){
```

```
66
         int i;
 67
         memset(a,0,sizeof(a));
         for(i=0;i<len;i++)</pre>
 68
 69
             a[i]=T.a[i];
 70
 71
    //重载赋值运算符,大数之间赋值运算
 72
    BigNum & BigNum::operator=(const BigNum &n){
 73
         int i;
 74
         len=n.len;
 75
         memset(a,0,sizeof(a));
         for(i=0;i<len;i++)</pre>
 76
 77
             a[i]=n.a[i];
 78
         return *this;
 79
 80
    istream& operator>>(istream &in,BigNum &b){
 81
         char ch[MAXSIZE*4];
         int i=-1;
 82
 83
         in>>ch;
 84
         int L=strlen(ch);
 85
         int count=0,sum=0;
         for(i=L-1;i>=0;){
 86
 87
             sum=0;
 88
             int t=1;
             for(int j=0;j<4&&i>=0;j++,i--,t*=10){
 89
 90
                 sum+=(ch[i]-'0')*t;
             }
 91
 92
             b.a[count]=sum;
 93
             count++;
 94
         }
 95
         b.len=count++;
 96
         return in;
 97
    //重载输出运算符
 98
 99
    ostream& operator<<(ostream& out,BigNum& b){</pre>
         int i;
100
         cout<<b.a[b.len-1];</pre>
101
102
         for(i=b.len-2;i>=0;i---){
             printf("%04d",b.a[i]);
103
104
         }
105
         return out;
106
    //两个大数之间的相加运算
107
    BigNum BigNum::operator+(const BigNum &T)const{
108
109
         BigNum t(*this);
         int i,big;
110
         big=T.len>len?T.len:len;
111
         for(i=0;i<big;i++){</pre>
112
             t.a[i]+=T.a[i];
113
114
             if(t.a[i]>MAXN){
115
                 t.a[i+1]++;
116
                 t.a[i]—=MAXN+1;
```

```
117
             }
118
         }
         if(t.a[big]!=0)
119
120
            t.len=big+1;
         else t.len=big;
121
122
         return t;
123
    //两个大数之间的相减运算
124
125
    BigNum BigNum::operator-(const BigNum &T)const{
         int i,j,big;
126
         bool flag;
127
         BigNum t1,t2;
128
         if(*this>T){
129
130
             t1=*this;
131
             t2=T;
132
             flag=0;
         }
133
         else{
134
135
             t1=T;
             t2=*this;
136
137
             flag=1;
138
         }
         big=t1.len;
139
         for(i=0;i<big;i++){</pre>
140
141
             if(t1.a[i]<t2.a[i]){
142
                 j=i+1;
                 while(t1.a[j]==0)
143
144
                      j++;
145
                 t1.a[j---]---;
146
                 while(j>i)
147
                      t1.a[j--]+=MAXN;
                 t1.a[i]+=MAXN+1—t2.a[i];
148
             }
149
150
             else t1.a[i]-=t2.a[i];
         }
151
152
         t1.len=big;
153
         while(t1.a[t1.len-1]==0 && t1.len>1){
154
             t1.len--;
155
             big—-;
156
         if(flag)
157
158
             t1.a[big-1]=0-t1.a[big-1];
159
         return t1;
160
    }
    //两个大数之间的相乘
161
    BigNum BigNum::operator*(const BigNum &T)const{
162
         BigNum ret;
163
         int i,j,up;
164
         int temp,temp1;
165
166
         for(i=0;i<len;i++){</pre>
167
             up=0;
```

```
for(j=0;j<T.len;j++){</pre>
168
169
                 temp=a[i]*T.a[j]+ret.a[i+j]+up;
170
                 if(temp>MAXN){
                     temp1=temp-temp/(MAXN+1)*(MAXN+1);
171
                     up=temp/(MAXN+1);
172
                     ret.a[i+j]=temp1;
173
                 }
174
                 else{
175
176
                     up=0;
                     ret.a[i+j]=temp;
177
                 }
178
179
             if(up!=0)
180
                ret.a[i+j]=up;
181
182
        }
183
        ret.len=i+j;
        while(ret.a[ret.len-1]==0 && ret.len>1)ret.len--;
184
185
        return ret;
186
    }
    //大数对一个整数进行相除运算
187
    BigNum BigNum::operator/(const int &b)const{
188
189
        BigNum ret;
190
        int i,down=0;
        for(i=len-1;i>=0;i--){
191
             ret.a[i]=(a[i]+down*(MAXN+1))/b;
192
             down=a[i]+down*(MAXN+1)-ret.a[i]*b;
193
194
        }
        ret.len=len:
195
        while(ret.a[ret.len-1]==0 && ret.len>1)
196
197
             ret.len—;
198
        return ret;
199
    //大数对一个 int 类型的变量进行取模
200
201
    int BigNum::operator%(const int &b)const{
202
        int i,d=0;
203
        for(i=len-1;i>=0;i--)
204
             d=((d*(MAXN+1))%b+a[i])%b;
205
        return d;
206
    //大数的 n 次方运算
207
    BigNum BigNum::operator^(const int &n)const{
208
        BigNum t,ret(1);
209
210
        int i;
        if(n<0) exit(-1);
211
212
        if(n==0)return 1;
        if(n==1)return *this;
213
        int m=n;
214
        while(m>1){
215
216
             t=*this;
217
             for(i=1;(i<<1)<=m;i<<=1)
218
                t=t*t;
```

```
219
            m-=i;
220
            ret=ret*t;
            if(m==1)ret=ret*(*this);
221
222
223
        return ret;
224
    }
225
    //大数和另一个大数的大小比较
    bool BigNum::operator>(const BigNum &T)const{
226
227
        int ln;
        if(len>T.len)return true;
228
        else if(len==T.len){
229
230
            ln=len-1;
            while(a[ln]==T.a[ln]&&ln>=0)
231
232
            if(ln>=0 && a[ln]>T.a[ln])
233
234
                return true;
235
            else
236
                return false;
237
        }
        else
238
239
           return false;
240
    //大数和一个 int 类型的变量的大小比较
241
    bool BigNum::operator>(const int &t)const{
242
243
        BigNum b(t);
244
        return *this>b;
    }
245
    //输出大数
246
247
    void BigNum::print(){
248
        int i;
        printf("%d",a[len-1]);
249
250
        for(i=len-2;i>=0;i--)
251
          printf("%04d",a[i]);
252
        printf("\n");
253
    BigNum f[110];//卡特兰数
254
255
    int main(){
256
257
        f[0]=1;
258
        for(int i=1;i<=100;i++)</pre>
259
            f[i]=f[i-1]*(4*i-2)/(i+1);//卡特兰数递推式
        int n;
260
        while(scanf("%d",&n)==1){
261
            if(n==-1)break;
262
263
            f[n].print();
264
265
        return 0;
266 |}
```

### 8.3 strtok 和 sscanf 结合输入

空格作为分隔输入,读取一行的整数:

```
gets(buf);
int v;
char *p = strtok(buf,"_");
while(p)

{
    sscanf(p,"%d",&v);
    p = strtok(NULL,"_");
}
```

# 8.4 解决爆栈,手动加栈

防止爆栈最好方法是改变写法,弄成 bfs,或者模拟栈。加栈都是旁门左道,需谨慎! C++

放在头文件前面

```
1 | #pragma comment(linker, "/STACK:1024000000,1024000000")
G++
放在主函数里面(汇编开栈不一定适用,和系统有关。需谨慎!)
```

```
1 | int __size__ = 256<<20;
2 | char *_p__ = (char *)malloc(__size__)+__size__;
3 | __asm__("movl_\%0,\%esp\n"::"r"(__p__));</pre>
```

### 8.5 STL

#### 8.5.1 优先队列 priority queue

```
empty() 如果队列为空返回真
```

pop() 删除对顶元素

push() 加入一个元素

size() 返回优先队列中拥有的元素个数

top() 返回优先队列队顶元素

在默认的优先队列中,优先级高的先出队。在默认的 int 型中先出队的为较大的数。

```
1 | priority_queue<int>q1;//大的先出对
2 | priority_queue<int,vector<int>,greater<int> >q2; //小的先出队 自定义比较函数:
```

```
struct cmp
1
2
  {
3
     bool operator ()(int x, int y)
4
     {
         return x > y; // x 小的优先级高
5
6
       //也可以写成其他方式,如: return p[x] > p[y]; 表示 p[i] 小的优先级高
  }
7
8
  priority_queue<int, vector<int>, cmp>q;//定义方法
9
 //其中,第二个参数为容器类型。第三个参数为比较函数。
```

```
struct node
1
2
  {
3
      int x, y;
      friend bool operator < (node a, node b)</pre>
4
5
          return a.x > b.x; //结构体中, x 小的优先级高
6
7
      }
8
  };
9
  |priority_queue<node>q;//定义方法
  |//在该结构中,y 为值, x 为优先级。
  |//通过自定义 operator< 操作符来比较元素中的优先级。
11
12 //在重载"<"时,最好不要重载">",可能会发生编译错误
  8.5.2 set 和 multiset
  set 和 multiset 用法一样,就是 multiset 允许重复元素。
   元素放入容器时,会按照一定的排序法则自动排序,默认是按照 less<> 排序规则来排序。
   不能修改容器里面的元素值,只能插入和删除。
   自定义 int 排序函数: (默认的是从小到大的,下面这个从大到小)
1 struct classcomp {
2
    bool operator() (const int& lhs, const int& rhs) const
3
    {return lhs>rhs;}
  };//这里有个逗号的,注意
5 multiset<int,classcomp> fifth;
                                            // class as Compare
   上面这样就定义成了从大到小排列了。
  结构体自定义排序函数:
   (定义 set 或者 multiset 的时候定义了排序函数,定义迭代器时一样带上排序函数)
1 struct Node
2
  {
3
      int x,y;
  };
  struct classcomp//先按照 x 从小到大排序, x 相同则按照 y 从大到小排序
6
  {
7
      bool operator()(const Node &a,const Node &b)const
8
9
          if(a.x!=b.x)return a.x<b.x;</pre>
          else return a.y>b.y;
10
11
  |}; //注意这里有个逗号
12
  multiset<Node,classcomp>mt;
14 | multiset<Node,classcomp>::iterator it;
1
2
  主要函数:
  |begin() 返回指向第一个元素的迭代器
  clear() 清除所有元素
  count() 返回某个值元素的个数
  |empty() 如果集合为空,返回 true
```

```
7 end() 返回指向最后一个元素的迭代器
8 erase() 删除集合中的元素 (参数是一个元素值,或者迭代器)
9 find() 返回一个指向被查找到元素的迭代器
10 insert() 在集合中插入元素
11 size() 集合中元素的数目
12 lower_bound() 返回指向大于(或等于)某值的第一个元素的迭代器
13 upper_bound() 返回大于某个值元素的迭代器
14 equal_range() 返回集合中与给定值相等的上下限的两个迭代器
15 (注意对于 multiset 删除操作之间删除值会把所以这个值的都删掉,删除一个要用迭代
器)
```

## 8.6 输入输出外挂

```
1
   //适用于正负整数
   template <class T>
   inline bool scan d(T &ret) {
      char c; int sgn;
 5
      if(c=getchar(),c==EOF) return 0; //EOF
 6
 7
      while(c!='-'&&(c<'0'||c>'9')) c=getchar();
      sgn=(c=='-')?-1:1;
 8
      ret=(c=='-')?0:(c-'0');
 9
      while(c=getchar(),c>='0'&&c<='9') ret=ret*10+(c-'0');</pre>
10
11
      ret*=sgn;
      return 1;
12
13
   }
14
15
   inline void out(int x) {
      if(x>9) out(x/10);
16
      putchar(x%10+'0');
17
18 | }
```

## 8.7 莫队算法

莫队算法,可以解决一类静态,离线区间查询问题。

BZOJ 2038: [2009 国家集训队] 小 Z 的袜子 (hose)

Description

作为一个生活散漫的人,小 Z 每天早上都要耗费很久从一堆五颜六色的袜子中找出一双来穿。终于有一天,小 Z 再也无法忍受这恼人的找袜子过程,于是他决定听天由命……具体来说,小 Z 把这 N 只袜子从 1 到 N 编号,然后从编号 L 到 R(L

Input

输入文件第一行包含两个正整数 N 和 M 。 N 为袜子的数量, M 为小 N 所提的询问的数量。接下来一行包含 N 个正整数 N 八 其中 N 表示第 N ,以袜子的颜色,相同的颜色用相同的数字表示。再接下来 M 行,每行两个正整数 N ,N 表示一个询问。

Output

包含 M 行,对于每个询问在一行中输出分数 A/B 表示从该询问的区间 [L,R] 中随机抽出两只袜子颜色相同的概率。若该概率为 0 则输出 0/1,否则输出的 A/B 必须为最简分数。(详见样例)

Sample Input

6 4

1 2 3 3 3 2

```
26
   13
   3 5
   16
   Sample Output
   2/5
   0/1
   1/1
   4/15
   题解:
   只需要统计区间内各个数出现次数的平方和
   莫队算法,两种方法,一种是直接分成 sqrt(n) 块,分块排序。
   另外一种是求得曼哈顿距离最小生成树,根据 manhattan MST 的 dfs 序求解。
   8.7.1 分块
 1 | const int MAXN = 50010;
 2 | const int MAXM = 50010;
 3
   struct Query
 4
   {
 5
       int L,R,id;
   }node[MAXM];
 7
   long long gcd(long long a,long long b){
       if(b == 0)return a;
       return gcd(b,a%b);
10
11
   struct Ans{
12
       long long a,b;//分数 a/b
13
       void reduce()//分数化简
14
       {
15
           long long d = gcd(a,b);
           a /= d; b /= d;
       }
17
18 | ans[MAXM];
   int a[MAXN];
19
   |int num[MAXN];
   int n,m,unit;
21
22
   bool cmp(Query a,Query b){
23
       if(a.L/unit != b.L/unit)return a.L/unit < b.L/unit;</pre>
24
       else return a.R < b.R;</pre>
25
   }
26
   void work(){
       long long temp = 0;
       memset(num,0,sizeof(num));
       int L = 1;
       int R = 0;
       for(int i = 0;i < m;i++){</pre>
```

8

9

16

27

28

29

30

31

32

kuangbin 259

while(R < node[i].R){</pre>

```
33
                R++;
34
                temp -= (long long)num[a[R]]*num[a[R]];
35
                num[a[R]]++;
36
                temp += (long long)num[a[R]]*num[a[R]];
37
            }
            while(R > node[i].R){
38
39
                temp -= (long long)num[a[R]]*num[a[R]];
40
                num[a[R]]--;
41
                temp += (long long)num[a[R]]*num[a[R]];
42
                R---;
43
            }
44
            while(L < node[i].L){</pre>
45
                temp -= (long long)num[a[L]]*num[a[L]];
46
                num[a[L]]--;
47
                temp += (long long)num[a[L]]*num[a[L]];
48
                L++;
49
50
            while(L > node[i].L){
51
52
                temp -= (long long)num[a[L]]*num[a[L]];
53
                num[a[L]]++;
54
                temp += (long long)num[a[L]]*num[a[L]];
55
            }
            ans[node[i].id].a = temp - (R-L+1);
56
57
            ans[node[i].id].b = (long long)(R-L+1)*(R-L);
            ans[node[i].id].reduce();
58
59
        }
60
61
   int main(){
62
        while(scanf("%d%d",&n,&m) == 2){
            for(int i = 1;i <= n;i++)</pre>
63
                scanf("%d",&a[i]);
64
            for(int i = 0;i < m;i++){</pre>
65
66
                node[i].id = i;
67
                scanf("%d%d",&node[i].L,&node[i].R);
68
            }
69
            unit = (int)sqrt(n);
70
            sort(node, node+m, cmp);
71
            work();
72
            for(int i = 0;i < m;i++)
73
                printf("%lld/%lld\n",ans[i].a,ans[i].b);
74
75
        return 0;
76 |}
   8.7.2 Manhattan MST 的 dfs 顺序求解
 1 | const int MAXN = 50010;
   const int MAXM = 50010;
   const int INF = 0x3f3f3f3f;
   struct Point{
 4
        int x,y,id;
 5
```

```
6 | p [MAXN], pp [MAXN];
 7
   bool cmp(Point a, Point b)
 8
 9
        if(a.x != b.x) return a.x < b.x;
10
        else return a.y < b.y;</pre>
11
   //树状数组,找 y-x 大于当前的,但是 y+x 最小的
12
13
   struct BIT{
14
        int min_val,pos;
15
        void init()
16
        {
17
            min_val = INF;
18
            pos = -1;
19
20 | }bit[MAXN];
21 | struct Edge{
22
        int u,v,d;
23
   |}edge[MAXN<<2];
24 bool cmpedge(Edge a, Edge b){
25
       return a.d < b.d;</pre>
   }
26
27
   int tot;
28
   int n;
   int F[MAXN];
29
30
   int find(int x){
31
        if(F[x] == -1) return x;
32
        else return F[x] = find(F[x]);
33
   void addedge(int u,int v,int d){
35
        edge[tot].u = u;
        edge[tot].v = v;
36
        edge[tot++].d = d;
37
38
   }
39
   struct Graph{
40
        int to,next;
41
   }e[MAXN<<1];
   int total,head[MAXN];
42
   void _addedge(int u,int v){
43
44
        e[total].to = v;
45
        e[total].next = head[u];
46
        head[u] = total++;
47
48
   | int lowbit(int x){
49
        return x&(-x);
50
   void update(int i,int val,int pos){
51
52
        while(i > 0){
53
            if(val < bit[i].min_val){</pre>
                bit[i].min_val = val;
54
55
                bit[i].pos = pos;
56
            }
```

```
i -= lowbit(i);
 57
 58
         }
 59
 60
    int ask(int i,int m){
         int min_val = INF,pos = -1;
 61
 62
         while(i <= m){
 63
             if(bit[i].min_val < min_val){</pre>
                 min_val = bit[i].min_val;
 64
 65
                 pos = bit[i].pos;
 66
             i += lowbit(i);
 67
 68
         }
 69
         return pos;
 70
 71
    int dist(Point a,Point b){
 72
         return abs(a.x - b.x) + abs(a.y - b.y);
 73
 74
    void Manhattan_minimum_spanning_tree(int n,Point p[]){
 75
         int a[MAXN],b[MAXN];
 76
         tot = 0;
         for(int dir = 0;dir < 4;dir++){</pre>
 77
 78
             if(dir == 1 || dir == 3){
 79
                 for(int i = 0;i < n;i++)
                      swap(p[i].x,p[i].y);
 80
 81
             else if(dir == 2){
 82
                 for(int i = 0;i < n;i++)</pre>
 83
                      p[i].x = -p[i].x;
 84
 85
             }
 86
             sort(p,p+n,cmp);
             for(int i = 0;i < n;i++)</pre>
 87
                 a[i] = b[i] = p[i].y - p[i].x;
 88
 89
             sort(b,b+n);
 90
             int m = unique(b,b+n) - b;
 91
             for(int i = 1;i <= m;i++)
 92
                 bit[i].init();
             for(int i = n-1; i >= 0; i—){
 93
                 int pos = lower_bound(b,b+m,a[i]) - b + 1;
 94
 95
                 int ans = ask(pos,m);
                 if(ans !=-1)
 96
 97
                      addedge(p[i].id,p[ans].id,dist(p[i],p[ans]));
 98
                 update(pos,p[i].x+p[i].y,i);
 99
             }
100
         }
         memset(F,-1,sizeof(F));
101
         sort(edge,edge+tot,cmpedge);
102
         total = 0;
103
         memset(head,-1,sizeof(head));
104
         for(int i = 0;i < tot;i++){
105
106
             int u = edge[i].u, v = edge[i].v;
             int t1 = find(u), t2 = find(v);
107
```

```
if(t1 != t2){
108
109
                 F[t1] = t2;
110
                 _addedge(u,v);
111
                 _addedge(v,u);
             }
112
113
        }
114
    int m;
115
116
    int a[MAXN];
117
    struct Ans{
118
        long long a,b;
119
    }ans[MAXM];
    long long temp ;
120
121
    int num[MAXN];
    void add(int l,int r){
122
123
        for(int i = l;i <= r;i++){
124
             temp -= (long long)num[a[i]]*num[a[i]];
125
             num[a[i]]++;
126
             temp += (long long)num[a[i]]*num[a[i]];
        }
127
128
129
    void del(int l,int r){
130
        for(int i = l;i <= r;i++){
131
             temp -= (long long)num[a[i]]*num[a[i]];
132
             num[a[i]]--;
133
             temp += (long long)num[a[i]]*num[a[i]];
        }
134
135
    void dfs(int l1,int r1,int l2,int r2,int idx,int pre){
136
137
        if(l2 < l1) add(l2,l1-1);
        if(r2 > r1) add(r1+1,r2);
138
        if(l2 > l1) del(l1,l2-1);
139
        if(r2 < r1) del(r2+1,r1);
140
        ans[pp[idx].id].a = temp - (r2-l2+1);
141
        ans[pp[idx].id].b = (long long)(r2-l2+1)*(r2-l2);
142
        for(int i = head[idx]; i != -1; i = e[i].next){
143
144
             int v = e[i].to;
145
             if(v == pre) continue;
146
             dfs(l2,r2,pp[v].x,pp[v].y,v,idx);
147
        }
        if(l2 < l1)del(l2,l1-1);
148
        if(r2 > r1)del(r1+1,r2);
149
150
        if(l2 > l1)add(l1,l2-1);
151
        if(r2 < r1)add(r2+1,r1);
152
153
    long long gcd(long long a,long long b){
        if(b == 0) return a;
154
155
        else return gcd(b,a%b);
156
157
    int main(){
158
        while(scanf("%d%d",&n,&m) == 2){
```

```
159
             for(int i = 1;i <= n;i++)
160
                 scanf("%d",&a[i]);
             for(int i = 0;i < m;i++){</pre>
161
                 scanf("%d%d",&p[i].x,&p[i].y);
162
                 p[i].id = i;
163
                 pp[i] = p[i];
164
165
166
             Manhattan_minimum_spanning_tree(m,p);
             memset(num,0,sizeof(num));
167
             temp = 0;
168
169
             dfs(1,0,pp[0].x,pp[0].y,0,-1);
             for(int i = 0;i < m;i++){
170
                 long long d = gcd(ans[i].a,ans[i].b);
171
                 printf("%lld/%lld\n",ans[i].a/d,ans[i].b/d);
172
             }
173
174
         }
175
         return 0;
176 |}
         VIM 配置
    8.8
  1
    set nu
  2
    set history=1000000
  3
  4
    set tabstop=4
    set shiftwidth=4
  6
    set smarttab
  7
    set cindent
  9
 10
    colo evening
 11
 12
    set nobackup
 13
    set noswapfile
 14
 15
    set mouse=a
    map <F6> :call CR()<CR>
 16
    func! CR()
 17
 18 exec "w"
    exec "!g++_%_-o_%<"
 19
    exec "!u./%<"
 20
 21
    endfunc
 22
 23
    |imap <c-]> {<cr>}<c-o>0<left><right>
 24 map <F2> :call SetTitle()<CR>
    func SetTitle()
 25
    let l = 0
 26
    let l = l + 1 | call setline(l,'#include<sub>□</sub><stdio.h>')
 27
    let l = l + 1 | call setline(l,'#include<sub>□</sub><string.h>')
    let l = l + 1 | call setline(l,'#include⊔<iostream>')
    let l = l + 1 | call setline(l,'#include_<algorithm>')
 30
 31 |let l = l + 1 | call setline(l,'#include<sub>□</sub><vector>')
```

```
let l = l + 1 | call setline(l,'#include<sub>□</sub><set>')
   let l = l + 1 | call setline(l,'#include

<map>')
34
   let l = l + 1 | call setline(l,'#include<sub>□</sub><string>')
35
   let l = l + 1 | call setline(l,'#include<sub>□</sub><math.h>')
36
   let l = l + 1 | call setline(l, '#include <stdlib.h>')
37
   let l = l + 1 | call setline(l,'#include
<time.h>')
38
   let l = l + 1 | call setline(l,'using⊔namespace⊔std;')
   let l = l + 1 | call setline(l,'')
   let l = l + 1 | call setline(l,'int<sub>□</sub>main()')
41
   let l = l + 1 | call setline(l,'{')
42
   let l = l + 1 | call setline(l, 'uuuu//freopen("in.txt", "r", stdin);'
43
     )
44
   let l = l + 1 | call setline(l,'____//freopen("out.txt","w",stdout)
     ;')
   let l = l + 1 | call setline(l,'____')
45
   let l = l + 1 | call setline(l,'}')
47
48 endfunc
   现场赛配置:
  syntax on
 2
  set nu
   set tabstop=4
   set shiftwidth=4
 5
   set cin
   colo evening
 6
  set mouse=a
```