

# 目录

<b>1</b>	<b>数据结构</b>	<b>1</b>
1.1	单调栈	1
1.2	单调队列	1
1.3	线段树套平衡树	1
<b>2</b>	<b>图论</b>	<b>4</b>
2.1	点分治	4
2.1.1	两点间的距离是否为 3 的倍数（[国家集训队] 聪聪可可）	4
2.1.2	询问树上距离为 $k$ 的点对是否存在	6
2.1.3	两点间距离不超过 $l$ 距离权重和不超过 $w$	8
2.2	DSU	10
2.2.1	询问子树颜色种类数	10
2.3	无向图相邻边成对	11
2.4	BFS TREE(CF1496F)	12
<b>3</b>	<b>整体二分</b>	<b>13</b>
3.1	每次询问一个子矩阵的第 $k$ 小数	13
3.2	带修主席树	15
3.3	在右半边的整体二分	17
<b>4</b>	<b>计算几何</b>	<b>19</b>
4.1	凸包	19
4.2	四边形	20
4.2.1	四边形计数	20
4.2.2	四边形最小面积计数	21
<b>5</b>	<b>习题整理</b>	<b>23</b>
5.1	dfs+2019 银川 A	23
5.2	链哈 LCA 倍增	24
5.3	后缀自动机 +set	27
5.4	CF840D Destiny	28



# 1 数据结构

## 1.1 单调栈

```

1 int s[N], top=0;
2 for(int i=1;i<=n;i++) {
3     while(top&&a[s[top]]>a[i]) top--;
4     if(s[top]==a[i]) s[top]=i;//去重复操作，视情况而定
5     s[++top]=i;
6 }

```

## 1.2 单调队列

```

1 int l=0,r=1;
2 int q[N];
3 for(int i=1;i<=n;i++) {
4     while(l<=r&&a[q[r]]>a[i]) r--;
5     q[++r]=i;
6     while(l<=r&&q[l]<i-m+1) l++;//判断条件看情况
7     ans[i]=a[q[l]];
8 }

```

## 1.3 线段树套平衡树

**1 l r k** 查询  $k$  在区间内的排名  $O(\log^2 N)$

**2 l r k** 查询区间内排名为  $k$  的值  $O(\log^3 N)$

**3 pos k** 修改某一位值上的数值  $O(\log^2 N)$

**4 l r k** 查询  $k$  在区间内的前驱 (前驱定义为严格小于  $x$ ，且最大的数，若不存在输出 -2147483647)  $O(\log^3 N)$

**5 l r k** 查询  $k$  在区间内的后继 (后继定义为严格大于  $x$ ，且最小的数，若不存在输出 2147483647)  $O(\log^3 N)$

后面两个也有  $O(\log^2 N)$  的做法，求前驱/后继后取  $\max, \min$ 。

```

1 int a[MAXN];
2 class DS { public:
3     // treap begin
4     struct treapnode {
5         int ch[2], val, dat, siz, cnt;
6     } tnd[MAXN*30];
7     int tot;
8     inline int New_treapnode(int v) {
9         tot++;
10        tnd[tot].ch[0] = tnd[tot].ch[1] = 0;
11        tnd[tot].val = v, tnd[tot].dat = rand(), tnd[tot].siz = 1, tnd[tot].cnt = 1;
12        return tot;
13    }
14    #define lson tnd[rt].ch[0]
15    #define rson tnd[rt].ch[1]
16    inline void push_up(int rt) {
17        tnd[rt].siz = tnd[lson].siz + tnd[rson].siz + tnd[rt].cnt;
18    }
19    inline void rotate(int &rt, int d) {
20        int tmp = tnd[rt].ch[d ^ 1];
21        tnd[rt].ch[d ^ 1] = tnd[tmp].ch[d], tnd[tmp].ch[d] = rt, rt = tmp;

```

```

22     push_up(tnd[rt].ch[d]), push_up(rt);
23 }
24 void insert(int &rt, int v) {
25     if (!rt) {
26         rt = New_treapnode(v);
27         return;
28     }
29     if (v == tnd[rt].val) tnd[rt].cnt++;
30     else {
31         int d = v < tnd[rt].val ? 0 : 1;
32         insert(tnd[rt].ch[d], v);
33         if (tnd[rt].dat < tnd[tnd[rt].ch[d]].dat) rotate(rt, d ^ 1);
34     }
35     push_up(rt);
36 }
37 void remove(int &rt, int v) {
38     if (!rt) return;
39     if (v == tnd[rt].val) {
40         if (tnd[rt].cnt > 1) {
41             tnd[rt].cnt--; push_up(rt);
42             return;
43         }
44         if (lson || rson) {
45             if (!rson || tnd[lson].dat > tnd[rson].dat) {
46                 rotate(rt, 1), remove(rson, v);
47             } else rotate(rt, 0), remove(lson, v);
48             push_up(rt);
49         } else
50             rt = 0;
51     }
52     v < tnd[rt].val ? remove(lson, v) : remove(rson, v);
53     push_up(rt);
54 }
55
56 int get_rank(int rt, int v) {
57     if (!rt) return 0;
58     if (v == tnd[rt].val)
59         return tnd[lson].siz;
60     else if (v < tnd[rt].val)
61         return get_rank(lson, v);
62     else
63         return tnd[lson].siz + tnd[rt].cnt + get_rank(rson, v);
64 }
65
66 bool isexit(int rt, int v) {
67     if (!rt) return 0;
68     if (v == tnd[rt].val)
69         return 1;
70     else if (v < tnd[rt].val)
71         return isexit(lson, v);
72     else
73         return isexit(rson, v);
74 }
75
76 void init() {
77     tot = 0;

```

```

78     }
79     // treap end
80
81     struct segnode {
82         int l, r, root;
83     } T[MAXN << 2];
84
85     void build(int rt, int l, int r) {
86         T[rt].l = l, T[rt].r = r;
87         for (int i = l; i <= r; i++)
88             insert(T[rt].root, a[i]);
89         if (l == r) return;
90         int mid = (l + r) >> 1;
91         build(rt << 1, l, mid), build(rt << 1 | 1, mid + 1, r);
92     }
93
94     void modify(int rt, int pos, int v) { // a[pos] -> v;
95         remove(T[rt].root, a[pos]), insert(T[rt].root, v);
96         if (T[rt].l == T[rt].r) {
97             a[pos] = v;
98             return;
99         }
100         int mid = (T[rt].l + T[rt].r) >> 1;
101         if (pos <= mid) modify(rt << 1, pos, v);
102         else modify(rt << 1 | 1, pos, v);
103     }
104
105     int get_rank(int rt, int L, int R, int v) { // op1
106         if (L <= T[rt].l && T[rt].r <= R) return get_rank(T[rt].root, v);
107         int mid = (T[rt].l + T[rt].r) >> 1;
108         int ans = 0;
109         if (L <= mid) ans += get_rank(rt << 1, L, R, v);
110         if (R > mid) ans += get_rank(rt << 1 | 1, L, R, v);
111         return ans;
112     }
113
114     bool isexit(int rt, int L, int R, int v) {
115         if (L <= T[rt].l && T[rt].r <= R) return isexit(T[rt].root, v);
116         int mid = (T[rt].l + T[rt].r) >> 1;
117         bool ans = false;
118         if (L <= mid) ans |= isexit(rt << 1, L, R, v);
119         if (R > mid) ans |= isexit(rt << 1 | 1, L, R, v);
120         return ans;
121     }
122 } tree;
123
124 int main() {
125     int n, m; scanf("%d%d", &n, &m);
126     for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
127     tree.build(1, 1, n);
128     while (m--) {
129         int opt; scanf("%d", &opt);
130         if (opt == 1) {
131             int l, r, k; scanf("%d%d%d", &l, &r, &k);
132             printf("%d\n", tree.get_rank(1, l, r, k) + 1);
133         } else if (opt == 2) {

```

```

134     int l, r, k; scanf("%d%d%d", &l, &r, &k);
135     int L = 0, R = 1e8;
136     while (L < R) {
137         int mid = (L + R + 1) >> 1;
138         if (tree.get_rank(1, l, r, mid) >= k) R = mid - 1;
139         else L = mid;
140     }
141     printf("%d\n", L);
142 } else if (opt == 3) {
143     int pos, k; scanf("%d%d", &pos, &k);
144     tree.modify(1, pos, k);
145 } else if (opt == 4) {
146     int l, r, k;
147     scanf("%d%d%d", &l, &r, &k);
148     int num = tree.get_rank(1, l, r, k); // the rank of k
149     if (num == 0) printf("-2147483647\n");
150     else {
151         int L = 0, R = 1e8;
152         while (L < R) {
153             int mid = (L + R + 1) >> 1;
154             if (tree.get_rank(1, l, r, mid) >= num) R = mid - 1;
155             else L = mid;
156         }
157         printf("%d\n", L);
158     }
159 } else if (opt == 5) {
160     int l, r, k; scanf("%d%d%d", &l, &r, &k);
161     int num = tree.get_rank(1, l, r, k + 1);
162     if (num == r - l + 1) printf("2147483647\n");
163     else {
164         num++;
165         int L = 0, R = 1e8;
166         while (L < R) {
167             int mid = (L + R + 1) >> 1;
168             if (tree.get_rank(1, l, r, mid) >= num) R = mid - 1;
169             else L = mid;
170         }
171         printf("%d\n", L);
172     }
173 }
174 }
175 }

```

## 2 图论

### 2.1 点分治

#### 2.1.1 两点间的距离是否为 3 的倍数（[国家集训队] 聪聪可可）

```

1  /* 5
2     1 2 1
3     1 3 2
4     1 4 1

```

```

5      2 5 3
6      res = 13/25
7      (1,1), (2,2), (2,3), (2,5), (3,2), (3,3), (3,4), (3,5), (4,3), (4,4), (5,2), (5,3), (5,5)*/
8  struct Edge {
9      int to, nex, w;
10 } e[MAXN << 1];
11 int head[MAXN], tol;
12 void addEdge(int u, int v, int w) {
13     e[tol].to = v, e[tol].w = w, e[tol].nex = head[u], head[u] = tol, tol++;
14 }
15 int son[MAXN], f[MAXN], vis[MAXN];
16 int t[5], dis[MAXN];
17 int main() {
18     int n; scanf("%d", &n);
19     for (int i = 1; i <= n; i++) head[i] = -1;
20     tol = 0;
21     for (int i = 2; i <= n; i++) {
22         int u, v, w;
23         scanf("%d%d%d", &u, &v, &w);
24         w %= 3;
25         addEdge(u, v, w), addEdge(v, u, w);
26     }
27     int root = 0, sum = n;
28     f[0] = n;
29     function<void(int, int)> get_root = [&](int u, int fa) {
30         son[u] = 1, f[u] = 0;
31         for (int i = head[u]; ~i; i = e[i].nex) {
32             int v = e[i].to;
33             if (vis[v] || v == fa) continue;
34             get_root(v, u);
35             son[u] += son[v];
36             f[u] = max(f[u], son[v]);
37         }
38         f[u] = max(f[u], sum - son[u]);
39         if (f[u] < f[root]) root = u;
40     };
41     get_root(1, 0);
42     function<void(int, int)> get_dis = [&](int u, int fa) {
43         t[dis[u]]++;
44         for (int i = head[u]; ~i; i = e[i].nex) {
45             int v = e[i].to, w = e[i].w;
46             if (v == fa || vis[v]) continue;
47             dis[v] = (dis[u] + w) % 3;
48             get_dis(v, u);
49         }
50     };
51     auto cal = [&](int u, int x) {
52         t[0] = t[1] = t[2] = 0;
53         dis[u] = x;
54         get_dis(u, 0);
55         return t[1] * t[2] * 2 + t[0] * t[0];
56     };
57     int res = 0;
58     function<void(int)> solve = [&](int u) {
59         res += cal(u, 0);
60         vis[u] = 1;

```

```

61     for (int i = head[u]; ~i; i = e[i].nex) {
62         int v = e[i].to, w = e[i].w;
63         if (vis[v]) continue;
64         res -= cal(v, w);
65         root = 0, sum = son[v];
66         get_root(v, 0);
67         solve(root);
68     }
69 };
70 solve(root);
71 int gcdd = __gcd(res, n * n);
72 printf("%d/%d", res / gcdd, n * n / gcdd);
73 }

```

### 2.1.2 询问树上距离为 $k$ 的点是否存在

时间复杂度:  $O(n \log^2 n + nm \log n)$

```

1  const int MAXN = 1e4 + 5;
2  const int MAXM = 105;
3  struct Edge {
4      int to, w, nex;
5  } e[MAXN << 1];
6  int head[MAXN], tol;
7  void addEdge(int u, int v, int w) {
8      e[tol].to = v, e[tol].w = w, e[tol].nex = head[u], head[u] = tol, tol++;
9  }
10 int son[MAXN], f[MAXN], vis[MAXN];
11 int dis[MAXN], top[MAXN];
12 int qs[MAXM], res[MAXM];
13 int main() {
14     int n, m; scanf("%d%d", &n, &m);
15     tol = 0;
16     for (int i = 1; i <= n; i++) head[i] = -1;
17     for (int i = 2; i <= n; i++) {
18         int u, v, w;
19         scanf("%d%d%d", &u, &v, &w);
20         addEdge(u, v, w), addEdge(v, u, w);
21     }
22     for (int i = 1; i <= m; i++) scanf("%d", &qs[i]);
23
24     int root = 0, sum = n;
25     f[0] = n;
26     function<void(int, int)> get_root = [&](int u, int fa) {
27         son[u] = 1, f[u] = 0;
28         for (int i = head[u]; ~i; i = e[i].nex) {
29             int v = e[i].to;
30             if (vis[v] || v == fa) continue;
31             get_root(v, u);
32             son[u] += son[v], f[u] = max(f[u], son[v]);
33         }
34         f[u] = max(f[u], sum - son[u]);
35         if (f[u] < f[root]) root = u;
36     };
37     get_root(1, 0);
38

```



```

39 vector<int> vec;
40 function<void(int, int, int)> get_dis = [&](int u, int fa, int topf) {
41     vec.pb(u), top[u] = topf;
42     for (int i = head[u]; ~i; i = e[i].nex) {
43         int v = e[i].to, w = e[i].w;
44         if (vis[v] || v == fa) continue;
45         dis[v] = dis[u] + w;
46         get_dis(v, u, topf);
47     }
48 };
49 auto cal = [&](int u) {
50     vec.clear(), vec.pb(u);
51     dis[u] = 0, top[u] = u;
52     for (int i = head[u]; ~i; i = e[i].nex) {
53         int v = e[i].to, w = e[i].w;
54         if (vis[v]) continue;
55         dis[v] = w;
56         get_dis(v, u, v);
57     }
58     sort(vec.begin(), vec.end(), [&](int ta, int tb) {
59         return dis[ta] < dis[tb];
60     });
61     for (int i = 1; i <= m; i++) {
62         int L = 0, R = SZ(vec) - 1;
63         if (res[i] || qs[i] == 0) {
64             res[i] = 1;
65         } else {
66             while (L < R) {
67                 if (dis[vec[L]] + dis[vec[R]] > qs[i]) R--;
68                 else if (dis[vec[L]] + dis[vec[R]] < qs[i]) L++;
69                 else if (top[vec[L]] == top[vec[R]]) {
70                     if (dis[vec[R]] == dis[vec[R - 1]]) R--;
71                     else L++;
72                 } else {
73                     res[i] = 1;
74                     break;
75                 }
76             }
77         }
78     }
79 };
80
81 function<void(int)> solve = [&](int u) {
82     vis[u] = 1;
83     cal(u);
84     for (int i = head[u]; ~i; i = e[i].nex) {
85         int v = e[i].to, w = e[i].w;
86         if (vis[v]) continue;
87         root = 0, sum = son[v];
88         get_root(v, 0);
89         solve(root);
90     }
91 };
92 solve(root);
93 for (int i = 1; i <= m; i++) {
94     printf("%s\n", res[i] ? "AYE" : "NAY");

```

```

95     }
96 }

```

### 2.1.3 两点间距离不超过 $l$ 距离权重和不超过 $w$

```

1  /* 4 4 6
2     1 3
3     1 4
4     1 3
5     res = 4
6     6 2 17
7     1 3
8     2 5
9     2 13
10    1 6
11    5 9
12    res = 9 */
13 class BIT { public:
14     int val[MAXN], n;
15     void init(int _n) {
16         n = _n;
17         for (int i = 1; i <= n; i++) val[i] = 0;
18     }
19     inline int lowbit(int x) { return x & (-x); }
20
21     void add(int pos, int v) {
22         for (int i = pos; i <= n; i += lowbit(i)) val[i] += v;
23     }
24     int query(int pos) {
25         int ans = 0;
26         for (int i = pos; i >= 1; i -= lowbit(i)) ans += val[i];
27         return ans;
28     }
29 } tree;
30 struct Edge {
31     int to, nex;
32     ll w;
33 } e[MAXN << 1];
34 int head[MAXN], tol;
35 void addEdge(int u, int v, ll w) {
36     e[tol].to = v, e[tol].w = w, e[tol].nex = head[u], head[u] = tol, tol++;
37 }
38 int son[MAXN], f[MAXN], vis[MAXN];
39 int dis[MAXN];
40 ll wis[MAXN];
41
42 int main() {
43     int n, l; ll w; scanf("%d%d%lld", &n, &l, &w);
44     tol = 0;
45     for (int i = 1; i <= n; i++) head[i] = -1;
46     for (int i = 2; i <= n; i++) {
47         int v; ll w; scanf("%d%lld", &v, &w);
48         addEdge(i, v, w), addEdge(v, i, w);
49     }

```

```

50  int root = 0, sum = n;
51  f[0] = n;
52  function<void(int, int)> get_root = [&](int u, int fa) {
53      son[u] = 1, f[u] = 0;
54      for (int i = head[u]; ~i; i = e[i].nex) {
55          int v = e[i].to;
56          if (vis[v] || v == fa) continue;
57          get_root(v, u);
58          son[u] += son[v], f[u] = max(f[u], son[v]);
59      }
60      f[u] = max(f[u], sum - son[u]);
61      if (f[u] < f[root]) root = u;
62  };
63  get_root(1, 0);
64
65  vector<int> vec;
66  function<void(int, int)> get_dis = [&](int u, int fa) {
67      vec.pb(u);
68      for (int i = head[u]; ~i; i = e[i].nex) {
69          int v = e[i].to;
70          ll w = e[i].w;
71          if (v == fa || vis[v]) continue;
72          dis[v] = dis[u] + 1, wis[v] = wis[u] + w;
73          get_dis(v, u);
74      }
75  };
76  auto cal = [&](int u, int x1, ll x2) {
77      vec.clear();
78      dis[u] = x1, wis[u] = x2, get_dis(u, 0);
79      sort(vec.begin(), vec.end(), [&](int ta, int tb) {
80          return wis[ta] < wis[tb];
81      });
82      tree.init(n+1);
83      for (int i = 0; i < SZ(vec); i++) tree.add(dis[vec[i]] + 1, 1);
84      ll ans = 0;
85      int L = 0, R = SZ(vec) - 1;
86      while (L < R) {
87          if (wis[vec[L]] + wis[vec[R]] <= w) {
88              tree.add(dis[vec[L]] + 1, -1);
89              ans += tree.query(1 - dis[vec[L]] + 1);
90              L++;
91          } else {
92              tree.add(dis[vec[R]] + 1, -1);
93              R--;
94          }
95      }
96      tree.add(dis[vec[L]] + 1, -1);
97      return ans;
98  };
99
100  ll res = 0;
101  function<void(int)> solve = [&](int u) {
102      res += cal(u, 0, 0);
103      vis[u] = 1;
104      for (int i = head[u]; ~i; i = e[i].nex) {
105          int v = e[i].to;

```

```

106         ll w = e[i].w;
107         if (vis[v]) continue;
108         res -= cal(v, 1, w);
109         root = 0, sum = son[v];
110         get_root(v, 0);
111         solve(root);
112     }
113 };
114 solve(root);
115 printf("%lld\n", res);
116 }

```

## 2.2 DSU

### 2.2.1 询问子树颜色种类数

```

1  const int MAXN = 1e5 + 5;
2  struct Edge {
3      int to, nex;
4  } e[MAXN << 1];
5  int head[MAXN], tol;
6  void addEdge(int u, int v) {
7      e[tol].to = v, e[tol].nex = head[u], head[u] = tol, tol++;
8  }
9  int son[MAXN], siz[MAXN];
10 int col[MAXN], cnt[MAXN], res[MAXN];
11 int main() {
12     int n; scanf("%d", &n);
13     tol = 0;
14     for (int i = 1; i <= n; i++) head[i] = -1;
15     for (int i = 2; i <= n; i++) {
16         int u, v; scanf("%d%d", &u, &v);
17         addEdge(u, v), addEdge(v, u);
18     }
19     for (int i = 1; i <= n; i++) scanf("%d", &col[i]);
20     function<void(int, int)> dfs1 = [&](int u, int f) {
21         siz[u] = 1;
22         int maxson = -1;
23         for (int i = head[u]; ~i; i = e[i].nex) {
24             int v = e[i].to;
25             if (v == f) continue;
26             dfs1(v, u);
27             siz[u] += siz[v];
28             if (siz[v] > maxson) son[u] = v, maxson = siz[v];
29         }
30     };
31     dfs1(1, 0);
32     int ans = 0, son_son;
33     function<void(int, int, int)> dfs3 = [&](int u, int f, int val) {
34         cnt[col[u]] += val;
35         if (val == 1 && cnt[col[u]] == 1) ans++;
36         else if (val == -1 && cnt[col[u]] == 0) ans--;
37         for (int i = head[u]; ~i; i = e[i].nex) {
38             int v = e[i].to;

```

```

39         if (v == f || v == son_son) continue;
40         dfs3(v, u, val);
41     }
42 };
43 function<void(int, int, bool)> dfs2 = [&](int u, int f, bool kp) {
44     for (int i = head[u]; ~i; i = e[i].nex) {
45         int v = e[i].to;
46         if (v == son[u] || v == f) continue;
47         dfs2(v, u, 0);
48     }
49     if (son[u]) dfs2(son[u], u, 1), son_son = son[u];
50     dfs3(u, f, 1), son_son = -1;
51     res[u] = ans;
52     if (!kp) dfs3(u, f, -1);
53 };
54 dfs2(1, 0, 0);
55 int m; scanf("%d", &m);
56 while (m--) {
57     int x; scanf("%d", &x);
58     printf("%d\n", res[x]);
59 }
60 }

```

## 2.3 无向图相邻边成对

```

1 vector<pii> res;
2 int vis[MAXN];
3 int dfs(int u) {
4     vis[u] = 1;
5     int cur = -1;
6     for (int i = head[u]; ~i; i = e[i].nex) {
7         int v = e[i].to, id = e[i].id;
8         if (vis[v] == 1) continue;
9         int tx1 = id;
10        if (!vis[v]) {
11            int tx2 = dfs(v);
12            if (tx2 != -1) {
13                res.pb(mp(tx1, tx2));
14                tx1 = -1;
15            }
16        }
17        if (tx1 != -1) {
18            if (cur != -1) {
19                res.pb(mp(tx1, cur));
20                cur = -1;
21            } else cur = tx1;
22        }
23    }
24    vis[u] = 2;
25    return cur;
26 }
27
28 int main() {
29     int n; scanf("%d", &n);

```

```

30     int cnt = 0;
31     for (int i = 1; i <= n; i++) {
32         ll a, b, c, d;
33         scanf("%lld%lld%lld%lld", &a, &b, &c, &d);
34         ll gcdd1 = __gcd(c * b, d * (a + b)), gcdd2 = __gcd(b * (c + d), d * a);
35         pll ks1 = mp(c * b / gcdd1, d * (a + b) / gcdd1);
36         pll ks2 = mp(b * (c + d) / gcdd2, d * a / gcdd2);
37
38         if (ma.find(ks1) == ma.end()) {
39             ma[ks1] = ++cnt, head[cnt] = -1;
40         }
41         if (ma.find(ks2) == ma.end()) {
42             ma[ks2] = ++cnt, head[cnt] = -1;
43         }
44         addEdge(ma[ks1], ma[ks2], i), addEdge(ma[ks2], ma[ks1], i);
45     }
46     for (int i = 1; i <= cnt; i++) {
47         if (!vis[i]) dfs(i);
48     }
49     printf("%d\n", SZ(res));
50     for (auto e: res) { printf("%d %d\n", e.first, e.second); }
51
52 }

```

## 2.4 BFS TREE(CF1496F)

给定一张无向图，取任意两点进行如下操作：

以这两点  $x, y$  为源构造生成树，使得对于任意点  $k$ ，有  $dis[x][k] = \min(dis[x][k])$  且  $dis[y][k] = \min(dis[y][k])$

即  $x, y$  点与  $k$  点的距离即为  $x, y$  与  $k$  的最短路径（之一），对于每一对  $x, y$  求能构造的生成树有多少

记  $dist(x, y)$  为从点  $x$  到点  $y$  所经过的点的个数。

有两点性质：

1. 对于点  $z$ ，当  $dist(x, z) + dist(y, z) - 1 = dist(x, y)$ ，那么点  $z$  应当是在从  $x$  到  $y$  的最短路上。特别的，当这样的点的个数超过  $dist(x, y)$  个时，那么  $x$  和  $y$  作为根节点的 BFS 树同构必定不存在。
2. 对于其他不在从  $x$  到  $y$  的最短路的点  $u$ ，要存在相邻的点  $v$ ，使得  $dist(x, v) = dist(x, u) - 1$  并且  $dist(y, v) = dist(y, u) - 1$ 。

```

1  int dis[MAXN][MAXN];
2  void bfs(int s) {
3      queue<int> q; dis[s][s] = 1; q.push(s);
4      while (!q.empty()) {
5          int u = q.front(); q.pop();
6          for (int i = head[u]; ~i; i = e[i].nex) {
7              int v = e[i].to;
8              if (!dis[s][v]) {
9                  dis[s][v] = dis[s][u] + 1;
10                 q.push(v);
11             }
12         }
13     }
14 }
15 ll res[MAXN][MAXN];
16 int main() {
17     int n, m; scanf("%d%d", &n, &m);
18     for (int i = 1; i <= n; i++) head[i] = -1;
19     while (m--) {
20         int u, v;

```

```

21     scanf("%d%d", &u, &v);
22     addEdge(u, v), addEdge(v, u);
23 }
24 for (int i = 1; i <= n; i++) bfs(i);
25 for (int x = 1; x <= n; x++) {
26     for (int y = x; y <= n; y++) {
27         int cnt = 0;
28         for (int i = 1; i <= n; i++) {
29             if (dis[x][i] + dis[y][i] - 1 == dis[x][y]) cnt++; // i int the path of x to y
30         }
31         ll ans = 1;
32         if (cnt > dis[x][y]) ans = 0;
33         for (int u = 1; u <= n; u++) {
34             if (dis[x][u] + dis[y][u] - 1 != dis[x][y]) {
35                 int fg = 0;
36                 for (int i = head[u]; ~i; i = e[i].nex) {
37                     int v = e[i].to;
38                     if (dis[x][v] == dis[x][u] - 1 && dis[y][v] == dis[y][u] - 1) fg++;
39                 }
40                 ans = ans * fg % mod;
41                 if (!ans) break;
42             }
43         }
44         res[x][y] = res[y][x] = ans;
45     }
46 }
47 for (int i = 1; i <= n; i++) {
48     for (int j = 1; j <= n; j++) {
49         printf("%d ", res[i][j]);
50     }
51     printf("\n");
52 }
53 }

```

### 3 整体二分

#### 3.1 每次询问一个子矩阵的第 $k$ 小数

```

1  /* [input]          [output]
2     2 2
3     2 1
4     3 4
5     1 2 1 2 1      1
6     1 1 2 2 3      3      */
7  class BIT { public:
8      int val[MAXN][MAXN]; int n, m;
9      void init(int _n, int _m) {
10         n = _n, m = _m;
11         for (int i = 1; i <= n; i++)
12             for (int j = 1; j <= m; j++)
13                 val[i][j] = 0;
14     }
15     inline int lowbit(int x) { return x & (-x); }

```

```

16 void add(int x, int y, int v) {
17     for (int i = x; i <= n; i += lowbit(i))
18         for (int j = y; j <= m; j += lowbit(j))
19             val[i][j] += v;
20 }
21 inline int query(int x, int y) {
22     int ans = 0;
23     for (int i = x; i >= 1; i -= lowbit(i))
24         for (int j = y; j >= 1; j -= lowbit(j))
25             ans += val[i][j];
26     return ans;
27 }
28 int query(int x1, int y1, int x2, int y2) {
29     return query(x2, y2) - query(x1 - 1, y2) - query(x2, y1 - 1) + query(x1 - 1, y1 - 1);
30 }
31 } bit;
32
33 struct Query {
34     int x1, y1, x2, y2, k, val, id, type;
35 } q[MAXM], q1[MAXM], q2[MAXM];
36 int res[MAXM];
37 int a[MAXN][MAXN];
38
39 int main() {
40     int n, m; scanf("%d%d", &n, &m);
41     for (int i = 1; i <= n; i++) {
42         for (int j = 1; j <= n; j++)
43             scanf("%d", &a[i][j]);
44     }
45     int qcnt = 0;
46     for (int i = 1; i <= n; i++) {
47         for (int j = 1; j <= n; j++) {
48             q[++qcnt] = {i, j, i, j, 0, a[i][j], 0, 1};
49         }
50     }
51     for (int i = 1; i <= m; i++) {
52         int x1, y1, x2, y2, k;
53         scanf("%d%d%d%d", &x1, &y1, &x2, &y2, &k);
54         q[++qcnt] = {x1, y1, x2, y2, k, 0, i, 2};
55     }
56     bit.init(n, n);
57     function<void(int, int, int, int)> solve = [&](int l, int r, int ql, int qr) {
58         if (ql > qr) return;
59         if (l == r) {
60             for (int i = ql; i <= qr; i++) {
61                 if (q[i].type == 2) res[q[i].id] = l;
62             }
63             return;
64         }
65         int mid = (l + r) >> 1;
66         int cnt1 = 0, cnt2 = 0;
67         for (int i = ql; i <= qr; i++) {
68             if (q[i].type == 1) {
69                 if (q[i].val <= mid) {
70                     bit.add(q[i].x1, q[i].y1, 1);
71                     q1[++cnt1] = q[i];

```



```

72         } else q2[++cnt2] = q[i];
73     } else {
74         int d = bit.query(q[i].x1, q[i].y1, q[i].x2, q[i].y2);
75         if (q[i].k <= d) q1[++cnt1] = q[i];
76         else {
77             q[i].k -= d;
78             q2[++cnt2] = q[i];
79         }
80     }
81 }
82 for (int i = 1; i <= cnt1; i++) {
83     if (q1[i].type == 1)
84         bit.add(q1[i].x1, q1[i].y1, -1);
85 }
86 for (int i = 1; i <= cnt1; i++) q[ql + i - 1] = q1[i];
87 for (int i = 1; i <= cnt2; i++) q[ql + cnt1 + i - 1] = q2[i];
88 solve(l, mid, ql, ql + cnt1 - 1);
89 solve(mid + 1, r, ql + cnt1, qr);
90
91 };
92 solve(0, 1e9, 1, qcnt);
93 for (int i = 1; i <= m; i++) printf("%d\n", res[i]);
94 }

```

### 3.2 带修主席树

```

1  class BIT { public:
2      int val[MAXN], n;
3      void init(int _n) {
4          n = _n;
5          for (int i = 1; i <= n; i++) val[i] = 0;
6      }
7      inline int lowbit(int x) { return x & (-x); }
8      void add(int pos, int v) {
9          for (int i = pos; i <= n; i += lowbit(i)) val[i] += v;
10     }
11     int query(int pos) {
12         int ans = 0;
13         for (int i = pos; i >= 1; i -= lowbit(i)) ans += val[i];
14         return ans;
15     }
16     int query(int l, int r) { return query(r) - query(l - 1); }
17 } tree;
18 struct Query {
19     int l, r, k, val, id, type;
20 } q[MAXN << 1], q1[MAXN << 1], q2[MAXN << 1];
21 int a[MAXN], res[MAXN], type[MAXN];
22 int main() {
23     int n, m; scanf("%d%d", &n, &m);
24     for (int i = 1; i <= n; i++) scanf("%d", &a[i]);
25     int qcnt = 0;
26     for (int i = 1; i <= m; i++) {
27         q[++qcnt] = {0, 0, 0, a[i], i, 1};
28     }

```

```

29  for (int i = 1; i <= m; i++) {
30      char opt[3];
31      scanf("%s", opt);
32      if (opt[0] == 'Q') {
33          int l, r, k;
34          scanf("%d%d%d", &l, &r, &k);
35          q[++qcnt] = {l, r, k, 0, i, 2};
36          type[i] = 1;
37      } else {
38          int x, y;
39          scanf("%d%d", &x, &y);
40          q[++qcnt] = {0, 0, 0, a[x], x, 3};
41          q[++qcnt] = {0, 0, 0, y, x, 1};
42          a[x] = y;
43      }
44  }
45
46  tree.init(n);
47  function<void(int, int, int, int)> solve = [&](int l, int r, int ql, int qr) {
48      if (ql > qr) return;
49      if (l == r) {
50          for (int i = ql; i <= qr; i++) {
51              if (q[i].type == 2) res[q[i].id] = l;
52          }
53          return;
54      }
55      int mid = (l + r) >> 1;
56      int cnt1 = 0, cnt2 = 0;
57      for (int i = ql; i <= qr; i++) {
58          if (q[i].type == 1) {
59              if (q[i].val <= mid) {
60                  tree.add(q[i].id, 1);
61                  q1[++cnt1] = q[i];
62              } else {
63                  q2[++cnt2] = q[i];
64              }
65          } else if (q[i].type == 3) {
66              if (q[i].val <= mid) {
67                  tree.add(q[i].id, -1);
68                  q1[++cnt1] = q[i];
69              } else {
70                  q2[++cnt2] = q[i];
71              }
72          } else {
73              int d = tree.query(q[i].l, q[i].r);
74              if (q[i].k <= d) {
75                  q1[++cnt1] = q[i];
76              } else {
77                  q[i].k -= d;
78                  q2[++cnt2] = q[i];
79              }
80          }
81      }
82      for (int i = 1; i <= cnt1; i++) {
83          if (q1[i].type == 1) tree.add(q1[i].id, -1);
84          else if (q1[i].type == 3) tree.add(q1[i].id, 1);

```

```

85     }
86     for (int i = 1; i <= cnt1; i++) q[ql + i - 1] = q1[i];
87     for (int i = 1; i <= cnt2; i++) q[ql + cnt1 + i - 1] = q2[i];
88     solve(l, mid, ql, ql + cnt1 - 1);
89     solve(mid + 1, r, ql + cnt1, qr);
90 };
91 solve(0, 1e9, 1, qcnt);
92 for (int i = 1; i <= m; i++) {
93     if (type[i]) printf("%d\n", res[i]);
94 }
95 }

```

### 3.3 在右半边的整体二分

```

1  class BIT_LL { public:
2      __int128 val[MAXN], n;
3      void init(int _n) {
4          n = _n;
5          for (int i = 1; i <= n; i++) val[i] = 0;
6      }
7      inline int lowbit(int x) { return x & (-x); }
8      void add(int pos, __int128 v) {
9          for (int i = pos; i <= n; i += lowbit(i)) val[i] += v;
10     }
11     __int128 query(int pos) {
12         __int128 ans = 0;
13         for (int i = pos; i >= 1; i -= lowbit(i)) ans += val[i];
14         return ans;
15     }
16     __int128 query(int L, int R) {
17         return query(R) - query(L - 1);
18     }
19 } bit1, bit2;
20 struct Query {
21     int d, p, l;
22     ll g, L;
23     int id;
24 } q[MAXN << 1], q1[MAXN << 1], q2[MAXN << 1];
25 int res[MAXN], type[MAXN];
26 int main() {
27     int n, m; scanf("%d%d", &n, &m);
28     bit1.init(1e5), bit2.init(1e5);
29     int qcnt = 0;
30     for (int i = 1; i <= n; i++) {
31         int d, p, l;
32         scanf("%d%d%d", &d, &p, &l);
33         q[++qcnt] = {d, p, l, 0, 0, 0};
34         bit1.add(q[i].p, (ll) l);
35         bit2.add(q[i].p, (ll) l * p);
36     }
37     for (int i = 1; i <= m; i++) {
38         ll g, L;
39         scanf("%lld%lld", &g, &L);
40         ll tmp = bit1.query(1e5);

```

```

41     if (L > tmp) {
42         type[i] = 0;
43     } else {
44         int posL = 1, posR = 1e5;
45         while (posL < posR) {
46             int mid = (posL + posR) >> 1;
47             if (bit1.query(mid) < L) posL = mid + 1;
48             else posR = mid;
49         }
50         int pos = posL;
51
52         ll momo = bit2.query(pos - 1) +
53             (L - bit1.query(pos - 1)) * (bit2.query(pos, pos) / bit1.query(pos,
54                 pos));
55         if (momo > g) type[i] = 0;
56         else type[i] = 1, q[++qcnt] = {0, 0, 0, g, L, i};
57     }
58 }
59 bit1.init(1e5), bit2.init(1e5);
60 function<void(int, int, int, int)> solve = [&](int l, int r, int ql, int qr) {
61     if (ql > qr) return;
62     if (l == r) {
63         for (int i = ql; i <= qr; i++) {
64             if (q[i].id && type[q[i].id]) {
65                 res[q[i].id] = l;
66             }
67         }
68         return;
69     }
70     int mid = (l + r + 1) >> 1;
71     int cnt1 = 0, cnt2 = 0;
72     for (int i = ql; i <= qr; i++) {
73         if (q[i].id == 0) {
74             if (q[i].d >= mid) {
75                 bit1.add(q[i].p, (ll) q[i].l);
76                 bit2.add(q[i].p, (ll) q[i].l * q[i].p);
77                 q2[++cnt2] = q[i];
78             } else q1[++cnt1] = q[i];
79         } else {
80             ll tmp = bit1.query(1e5);
81             if (q[i].L > tmp) {
82                 q1[++cnt1] = q[i];
83             } else {
84                 int posL = 1, posR = 1e5;
85                 while (posL < posR) {
86                     int mid = (posL + posR) >> 1;
87                     if (bit1.query(mid) < q[i].L) posL = mid + 1;
88                     else posR = mid;
89                 }
90                 int pos = posL;
91                 __int128 momo = (__int128) bit2.query(pos - 1) +
92                     ((__int128) q[i].L - (__int128) bit1.query(pos - 1)) *
93                     (__int128) ((__int128) bit2.query(pos, pos) / (__int128)
94                         bit1.query(pos, pos));
95                 if (momo > q[i].g) q1[++cnt1] = q[i];
96                 else q2[++cnt2] = q[i];

```

```

95     }
96 }
97 }
98 for (int i = 1; i <= cnt1; i++) q[ql + i - 1] = q1[i];
99 for (int i = 1; i <= cnt2; i++) q[ql + cnt1 + i - 1] = q2[i];
100 solve(l, mid - 1, ql, ql + cnt1 - 1);
101 for (int i = 1; i <= cnt2; i++) {
102     if (q[ql + cnt1 + i - 1].id == 0) {
103         bit1.add(q[ql + cnt1 + i - 1].p, 0ll - (1ll) q[ql + cnt1 + i - 1].l);
104         bit2.add(q[ql + cnt1 + i - 1].p, 0ll - (1ll) q[ql + cnt1 + i - 1].l * q[ql +
105             cnt1 + i - 1].p);
106     }
107 }
108 solve(mid, r, ql + cnt1, qr);
109 };
110 solve(1, 1e5, 1, qcnt);
111 for (int i = 1; i <= m; i++)
112     if (type[i]) printf("%d\n", res[i]);
113     else printf("-1\n");
114 }

```

## 4 计算几何

### 4.1 凸包

```

1 struct Point {
2     double x, y;
3     Point(double _x = 0, double _y = 0) : x(_x), y(_y) {}
4     Point operator-(const Point &tb) { return Point(x - tb.x, y - tb.y); }
5     double dis(const Point &tb) {
6         return sqrt((x - tb.x) * (x - tb.x) + (y - tb.y) * (y - tb.y));
7     }
8 };
9 double cross(const Point &ta, const Point &tb) {
10     return ta.x * tb.y - ta.y * tb.x;
11 }
12 namespace Convex {
13     vector<Point> GetConvexHull(const vector<Point> &P) {
14         vector<Point> ans, p(P);
15         int n = SZ(p), t = -1;
16         sort(p.begin(), p.end(), [&](const Point &ta, const Point &tb) {
17             if (ta.x != tb.x) return ta.x < tb.x;
18             else return ta.y < tb.y;
19         }); ans.assign(n * 2, Point(0, 0));
20         auto ins = [&](int pre, const Point &u) {
21             while (t > pre && cross(ans[t - 1] - u, ans[t] - u) >= 0) t--;
22             ans[++t] = u;
23         };
24         for (auto u:p) ins(0, u);
25         for (int i = n - 2, pre = t; i >= 0; i--) ins(pre, p[i]);
26         ans.resize(t);
27         return ans;
28     }

```

```

29 }
30 vector<Point> vec;
31 int main() {
32     int n; scanf("%d", &n);
33     vec.resize(n);
34     for (int i = 0; i < n; i++) {
35         scanf("%lf%lf", &vec[i].x, &vec[i].y);
36     }
37     vector<Point> v = Convex::GetConvexHull(vec);
38     double res = 0.0;
39     for (int i = 0; i < SZ(v); i++) {
40         if (i == 0) res += v[i].dis(v[SZ(v) - 1]);
41         else res += v[i].dis(v[i - 1]);
42     }
43     printf("%.2lf\n", res);
44 }

```

## 4.2 四边形

### 4.2.1 四边形计数

```

1 struct Point {
2     ll x, y;
3     Point() {}
4     Point(ll _x, ll _y) : x(_x), y(_y) {}
5     Point operator-(const Point &tb) const {
6         return Point(x - tb.x, y - tb.y);
7     }
8     ll operator^(const Point &tb) const {
9         return x * tb.y - y * tb.x;
10    }
11 } p[MAXN];
12
13 struct node {
14     Point p; int a, b;
15     node() {}
16     node(Point _p, int _a, int _b) : p(_p), a(_a), b(_b) {}
17 } nd[MAXN * MAXN];
18
19 int rk[MAXN], id[MAXN];
20 int main() {
21     int T; scanf("%d", &T);
22     while (T--) {
23         int n; scanf("%d", &n);
24         for (int i = 1; i <= n; i++) scanf("%lld%lld", &p[i].x, &p[i].y);
25         sort(p + 1, p + 1 + n, [&](const Point &ta, const Point &tb) {
26             if (ta.x != tb.x) return ta.x < tb.x;
27             else return ta.y < tb.y;
28         });
29         int nd_tot = 0;
30         for (int i = 1; i <= n; i++) {
31             for (int j = 1; j < i; j++) nd[++nd_tot] = node(p[i] - p[j], i, j);
32         }
33         sort(nd + 1, nd + 1 + nd_tot, [&](const node &ta, const node &tb) {

```

```

34         ll tmp = ta.p ^tb.p;
35         return tmp < 0;
36     });
37     ll uconv = (ll) n * (n - 1) * (n - 2) * (n - 3) / 6;
38     for (int i = 1; i <= n; i++) rk[i] = id[i] = i;
39     for (int i = 1; i <= nd_tot; i++) {
40         int a = nd[i].a, b = nd[i].b;
41         if (rk[a] > rk[b]) swap(a, b);
42         int na = rk[a] - 1, nb = n - rk[b];
43         uconv -= (ll) na * (na - 1) / 2;
44         uconv -= (ll) nb * (nb - 1) / 2;
45         swap(rk[a], rk[b]);
46         swap(id[rk[a]], id[rk[b]]);
47     }
48     ll conv = (ll) n * (n - 1) * (n - 2) * (n - 3) / 24 - uconv;
49     printf("%lld\n", conv);
50 }
51
52 }
```

#### 4.2.2 四边形最小面积计数

```

1 struct Point {
2     ll x, y;
3     Point() {}
4     Point(ll _x, ll _y) : x(_x), y(_y) {}
5     Point operator-(const Point &tb) const {
6         return Point(x - tb.x, y - tb.y);
7     }
8     ll operator^(const Point &tb) const {
9         return x * tb.y - y * tb.x;
10    }
11 } p[MAXN];
12 struct node {
13     Point p; int a, b;
14     node() {}
15     node(Point _p, int _a, int _b) : p(_p), a(_a), b(_b) {}
16 } nd[MAXN * MAXN];
17 int rk[MAXN], id[MAXN];
18 int main() {
19     int n; scanf("%d", &n);
20     for (int i = 1; i <= n; i++) scanf("%lld%lld", &p[i].x, &p[i].y);
21     sort(p + 1, p + 1 + n, [&](const Point &ta, const Point &tb) {
22         if (ta.x != tb.x) return ta.x < tb.x;
23         else return ta.y < tb.y;
24     });
25     int nd_tot = 0;
26     for (int i = 1; i <= n; i++) {
27         for (int j = 1; j < i; j++) nd[++nd_tot] = node(p[i] - p[j], i, j);
28     }
29     sort(nd + 1, nd + 1 + nd_tot, [&](const node &ta, const node &tb) {
30         ll tmp = ta.p ^tb.p;
31         return tmp < 0;
32     });
33 }
```

```

33 ll uconv = (ll) n * (n - 1) * (n - 2) * (n - 3) / 6;
34 ll area = LLONG_MAX; // llinf? dog never use!
35 ll cnt_uconv = 0, cnt_conv = 0;
36 for (int i = 1; i <= n; i++) rk[i] = id[i] = i;
37 for (int i = 1; i <= nd_tot; i++) {
38     Point cen = nd[i].p;
39     int a = nd[i].a, b = nd[i].b;
40     if (rk[a] > rk[b]) swap(a, b);
41     int na = rk[a] - 1, nb = n - rk[b];
42     uconv -= (ll) na * (na - 1) / 2;
43     uconv -= (ll) nb * (nb - 1) / 2;
44     if (1 <= rk[a] - 1 && rk[b] + 1 <= n) {
45         ll left = abs(cen ^ (p[a] - p[id[rk[a] - 1]])), right = abs(cen ^ (p[b] -
46             p[id[rk[b] + 1]]));
47         ll cur_area = left + right;
48         if (cur_area < area) area = cur_area, cnt_uconv = cnt_conv = 0;
49         if (cur_area == area) {
50             vi l, r;
51             l.pb(id[rk[a] - 1]), r.pb(id[rk[b] + 1]);
52             for (int j = rk[a] - 2; j >= 1; j--) {
53                 if (abs(cen ^ (p[id[j]] - p[a])) == left) l.pb(id[j]);
54             }
55             for (int j = rk[b] + 2; j <= n; j++) {
56                 if (abs(cen ^ (p[id[j]] - p[b])) == right) r.pb(id[j]);
57             }
58             for (auto &ea:l) {
59                 for (auto &eb:r) {
60                     vector<Point> tmp{p[ea], p[a], p[eb], p[b]};
61                     int s = signbit((tmp[1] - tmp[0]) ^ (tmp[2] - tmp[0]));
62                     bool f = 0;
63                     for (int j = 1; j < 4; j++) {
64                         if (s != signbit((tmp[(j + 1) % 4] - tmp[j]) ^ (tmp[(j + 2) % 4] -
65                             tmp[j]))) {
66                             f = 1; break;
67                         }
68                     }
69                     if (f) cnt_uconv++;
70                     else cnt_conv++;
71                 }
72             }
73             swap(rk[a], rk[b]); swap(id[rk[a]], id[rk[b]]);
74         }
75     }
76 ll conv = (ll) n * (n - 1) * (n - 2) * (n - 3) / 24 - uconv;
77 cnt_conv /= 2;
78 ll res = 4ll * cnt_conv + 3ll * cnt_uconv + 2ll * (conv - cnt_conv) + 1ll * (uconv * 3 -
79     cnt_uconv);
80 printf("%lld\n", res);
81 }

```



## 5 习题整理

### 5.1 dfs+2019 银川 A

```

1  const int MAXN = 1e5 + 5;
2  string name[MAXN], color[MAXN], bo_name[6], bo_color;
3  int val[MAXN];
4  vector<int> vec[4], v2[4];
5  int main() {
6      cin.tie(0), cin.sync_with_stdio(0);
7      int T; cin >> T;
8      while (T--) {
9          int n; cin >> n;
10         for (int i = 1; i <= n; i++) {
11             cin >> name[i] >> color[i] >> val[i];
12         }
13         for (int i = 1; i <= 5; i++) cin >> bo_name[i];
14         cin >> bo_color;
15         for (int i = 0; i < 4; i++) vec[i].clear(), v2[i].clear();
16         for (int i = 1; i <= n; i++) {
17             int fg1 = 0, fg2 = 0;
18             for (int j = 1; j <= 5; j++) {
19                 if (name[i] == bo_name[j]) {
20                     fg1 = 1;
21                     break;
22                 }
23             }
24             if (color[i] == bo_color) fg2 = 1;
25             if (fg1 == 0 && fg2 == 0) vec[0].pb(i);
26             else if (fg1 == 0 && fg2 == 1) vec[1].pb(i);
27             else if (fg1 == 1 && fg2 == 0) vec[2].pb(i);
28             else vec[3].pb(i);
29         }
30         for (int i = 0; i < 4; i++) {
31             sort(vec[i].begin(), vec[i].end(), [&](int ta, int tb) {
32                 if (name[ta] != name[tb]) return name[ta] < name[tb];
33                 return val[ta] > val[tb];
34             });
35             string pre = "";
36             for (auto e: vec[i]) {
37                 if (name[e] != pre) {
38                     v2[i].pb(e);
39                     pre = name[e];
40                 }
41             }
42             sort(v2[i].begin(), v2[i].end(), [&](int ta, int tb) {
43                 return val[ta] > val[tb];
44             });
45         }
46         int res = 0;
47         set<string> st;
48         function<void(int, int, int, int, int, int, int, int)> dfs = [&](int pos, int ri1, int
            ri2, int base, int t0,

```

```

49                                     int t1, int t2, int
50                                     t3) {
51
52     if (pos == 6) {
53         double ri = 1.0 + 0.1 * ri1 + 0.2 * ri2;
54         res = max(res, (int) floor(ri * base));
55         return;
56     }
57     if (t0 < SZ(v2[0])) {
58         if (!st.count(name[v2[0][t0]])) {
59             st.insert(name[v2[0][t0]]);
60             dfs(pos + 1, ri1, ri2, base + val[v2[0][t0]], t0 + 1, t1, t2, t3);
61             st.erase(name[v2[0][t0]]);
62         } else dfs(pos, ri1, ri2, base, t0 + 1, t1, t2, t3);
63     }
64     if (t1 < SZ(v2[1])) {
65         if (!st.count(name[v2[1][t1]])) {
66             st.insert(name[v2[1][t1]]);
67             dfs(pos + 1, ri1, ri2 + 1, base + val[v2[1][t1]], t0, t1 + 1, t2, t3);
68             st.erase(name[v2[1][t1]]);
69         } else dfs(pos, ri1, ri2, base, t0, t1 + 1, t2, t3);
70     }
71     if (t2 < SZ(v2[2])) {
72         if (!st.count(name[v2[2][t2]])) {
73             st.insert(name[v2[2][t2]]);
74             dfs(pos + 1, ri1 + 1, ri2, base + val[v2[2][t2]], t0, t1, t2 + 1, t3);
75             st.erase(name[v2[2][t2]]);
76         } else dfs(pos, ri1, ri2, base, t0, t1, t2 + 1, t3);
77     }
78     if (t3 < SZ(v2[3])) {
79         if (!st.count(name[v2[3][t3]])) {
80             st.insert(name[v2[3][t3]]);
81             dfs(pos + 1, ri1 + 1, ri2 + 1, base + val[v2[3][t3]], t0, t1, t2, t3 + 1);
82             st.erase(name[v2[3][t3]]);
83         } else dfs(pos, ri1, ri2, base, t0, t1, t2, t3 + 1);
84     }
85     dfs(1, 0, 0, 0, 0, 0, 0, 0);
86     cout << res << '\n';
87 }

```

## 5.2 链哈 LCA 倍增

```

1 struct LCA {
2     int fa[MAXN][22], dep[MAXN], n, limt, bin[22], len[MAXN];
3     ll sd[2], p[2], ha[MAXN][2], tmp[MAXN][2];
4     vector<int> mp[MAXN];
5     void init(int _n) {
6         n = _n;
7         for (limt = 1; 1 << (limt - 1) <= n; limt++);
8         for (int i = bin[0] = 1; 1 << (i - 1) <= n; i++) bin[i] = (bin[i - 1] << 1);
9         sd[0] = 13331, sd[1] = 23333;
10        p[0] = 1e9 + 7, p[1] = 998244353;
11        mem(ha[0], 0);

```

```

12     tmp[0][0] = tmp[0][1] = 1;
13     mem(fa[0], 0);
14     for (int i = 1; i <= n; i++) {
15         mp[i].clear();
16         mem(fa[i], 0);
17         tmp[i][0] = tmp[i - 1][0] * sd[0] % p[0];
18         tmp[i][1] = tmp[i - 1][1] * sd[1] % p[1];
19     }
20 }
21 void add_edge(int a, int b) { mp[a].pb(b); }
22 void dfs(int x, int pre) {
23     ha[x][0] = (ha[pre][0] * sd[0] + len[x]) % p[0];
24     ha[x][1] = (ha[pre][1] * sd[1] + len[x]) % p[1];
25     for (int i = 1; bin[i] <= dep[x]; i++) fa[x][i] = fa[fa[x][i - 1]][i - 1];
26     for (int i = 0; i < sz(mp[x]); i++) {
27         int to = mp[x][i];
28         if (to == pre) continue;
29         dep[to] = dep[x] + 1;
30         fa[to][0] = x;
31         dfs(to, x);
32     }
33 }
34 void work(int rt) {
35     dep[rt] = 0;
36     dfs(rt, 0);
37 }
38 int find(int a, int L) {
39     if (len[a] == L) return a;
40     for (int i = limt; ~i; i--) {
41         if (len[fa[a][i]] > L) a = fa[a][i];
42     }
43     assert(len[fa[a][0]] == L);
44     return fa[a][0];
45 }
46 ll get(int l, int r, int f) {
47     int LEN = dep[r] - dep[l] + 1;
48     if (l == 0) return ha[r][f];
49     return ((ha[r][f] - ha[fa[l][0]][f] * tmp[LEN][f]) % p[f] + p[f]) % p[f];
50 }
51 void comp(int x, int y) {
52     int tx, ty;
53     tx = x, ty = y;
54     if (len[tx] != len[ty]) {
55         if (len[tx] < len[ty]) puts("sjfnb");
56         else if (len[tx] > len[ty]) puts("cslnb");
57         return;
58     }
59     for (int i = limt; ~i; i--) {
60         if (get(fa[tx][i], tx, 0) == get(fa[ty][i], ty, 0) &&
61             get(fa[tx][i], tx, 1) == get(fa[ty][i], ty, 1)) {
62             tx = fa[tx][i];
63             ty = fa[ty][i];
64         }
65     }
66     tx = fa[tx][0];
67     ty = fa[ty][0];

```

```

68     if (len[tx] < len[ty]) puts("sjfnb");
69     else if (len[tx] > len[ty]) puts("cslnb");
70     else puts("draw");
71 }
72 } lca;
73 int pos[MAXN];
74 struct Palindrome_Tree {
75     struct node {
76         int ch[MAXC];
77         int fail, len;
78     } T[MAXN];
79     int las, tol;
80     int c[MAXN]; // cnt 计数, pos 记录位置
81     int get_fail(int x, int pos) {
82         while (c[pos - T[x].len - 1] != c[pos]) {
83             x = T[x].fail;
84         }
85         return x;
86     }
87     void init() { // 传入字符串长度
88         memset(T[0].ch, 0, sizeof(T[0].ch));
89         memset(T[1].ch, 0, sizeof(T[1].ch));
90         T[0].len = 0, T[1].len = -1;
91         T[0].fail = 1, T[1].fail = 0;
92         las = 0; tol = 1;
93     }
94     void insert(char s[], int len) {
95         c[0] = -1;
96         for (int i = 1; i <= len; i++) {
97             c[i] = s[i] - 'a';
98             int p = get_fail(las, i);
99             if (!T[p].ch[c[i]]) {
100                 T[++tol].len = T[p].len + 2;
101                 memset(T[tol].ch, 0, sizeof(T[tol].ch));
102                 int u = get_fail(T[p].fail, i);
103                 T[tol].fail = T[u].ch[c[i]];
104                 T[p].ch[c[i]] = tol;
105             }
106             las = T[p].ch[c[i]];
107             pos[i] = las;
108         }
109     }
110     void dfs(int u) {
111         printf("u = %d T[u].fail = %d T[u].len = %d\n", u, T[u].fail, T[u].len);
112         for (int i = 0; i < 26; i++) {
113             if (T[u].ch[i]) {
114                 printf("%d %d %d\n", u, i, T[u].ch[i]);
115                 dfs(T[u].ch[i]);
116             }
117         }
118     }
119     void build() {
120         lca.init(tol + 1);
121         for (int i = 1; i <= tol; i++) lca.add_edge(T[i].fail + 1, i + 1);
122         for (int i = 0; i <= tol; i++) lca.len[i + 1] = T[i].len;
123         lca.work(1);

```

```

124     }
125 } tree;
126 char s[MAXN];
127 int main() {
128     int T; scanf("%d", &T);
129     while (T--) {
130         int len;
131         scanf("%d", &len);
132         scanf("%s", s + 1);
133         tree.init();
134         tree.insert(s, len);
135         tree.build();
136         int q; scanf("%d", &q);
137         while (q--) {
138             int a, b, c, d;
139             scanf("%d%d%d%d", &a, &b, &c, &d);
140             int x = lca.find(pos[b] + 1, b - a + 1);
141             int y = lca.find(pos[d] + 1, d - c + 1);
142             lca.comp(x, y);
143         }
144     }
145 }

```

### 5.3 后缀自动机 +set

给定一个长度为  $n$  的字符串，对于前缀  $1..i$ ，找到最短的字符串，使其在整个长度为  $n$  的字符串中只出现一次，输出长度。

对于后缀自动机，其实是有三种情况进行分类讨论的：

1. 直接连到起始节点，这种情况不会出现重复。
2. 直连，类似 aa，这种情况会出现重复。
3. 之前出现过的状态，需要复制节点信息，这种情况会出现重复。

```

1 struct node {
2     int id, len;
3     node(int _id = 0, int _len = 0) : id(_id), len(_len) {}
4     bool operator<(const node &tb) const {
5         if (len != tb.len) return len < tb.len;
6         else return id < tb.id;
7     }
8 };
9 set<node> st;
10
11 class SAM {
12 public:
13     int rt, link[MAXN], maxlen[MAXN], trans[MAXN][MAXC];
14     int val[MAXN];
15     void init() {
16         rt = 1;
17         link[1] = maxlen[1] = 0;
18         memset(trans[1], 0, sizeof(trans[1]));
19     }
20     int insert(int ch, int last) {
21         int z = ++rt, p = last;
22         val[z] = 1;
23         memset(trans[z], 0, sizeof(trans[z]));
24         maxlen[z] = maxlen[last] + 1;

```

```

25 while (p && !trans[p][ch])trans[p][ch] = z, p = link[p];
26 if (!p) link[z] = 1;
27 else {
28     int x = trans[p][ch];
29     if (maxlen[p] + 1 == maxlen[x]) {
30         link[z] = x;
31         if (val[x]) val[x] = 0, st.erase(node(x, maxlen[link[x]] + 1));
32     } else {
33         int y = ++rt;
34         maxlen[y] = maxlen[p] + 1;
35         if (val[x]) st.erase(node(x, maxlen[link[x]] + 1));
36         for (int i = 0; i < MAXC; i++) trans[y][i] = trans[x][i];
37         while (p && trans[p][ch] == x) trans[p][ch] = y, p = link[p];
38         link[y] = link[x], link[z] = link[x] = y;
39         if (val[x]) st.insert(node(x, maxlen[link[x]] + 1));
40     }
41 }
42 st.insert(node(z, maxlen[link[z]] + 1));
43 return z;
44 }
45 } sa;
46
47 char s[MAXN];
48
49 int main() {
50     int n;
51     scanf("%d", &n);
52     sa.init();
53     scanf("%s", s + 1);
54     int last = 1;
55     for (int i = 1; i <= n; i++) {
56         last = sa.insert(s[i] - 'a', last);
57         set<node>::iterator it = st.begin();
58         printf("%d\n", it->len);
59     }
60 }

```

## 5.4 CF840D Destiny

给定  $n$  个元素,  $m$  次询问。

每次给出三个参数  $l, r, k$ , 询问区间  $[l, r]$  内是否存在出现次数严格大于  $\frac{r-l+1}{k}$  的数。如果存在就输出最小的那个  $ans$ , 否则输出  $-1$ 。

时间复杂度:  $O(nk \log n)$

```

1 class HJT { public:
2     int ch[MAXN * 70][2], sum[MAXN * 70];
3     int tot = 0;
4     inline void push_up(int rt) {
5         sum[rt] = sum[ch[rt][0]] + sum[ch[rt][1]];
6     }
7     int change(int rt, int pos, int val, int be, int en) {
8         int nrt = ++tot;
9         ch[nrt][0] = ch[nrt][1] = sum[nrt] = 0;
10        if (be == en) {
11            sum[nrt] = sum[rt] + val;

```

```

12         return nrt;
13     }
14     int mid = (be + en) >> 1;
15     if (pos <= mid) {
16         ch[nrt][0] = change(ch[rt][0], pos, val, be, mid);
17         ch[nrt][1] = ch[rt][1];
18     } else {
19         ch[nrt][0] = ch[rt][0];
20         ch[nrt][1] = change(ch[rt][1], pos, val, mid+1, en);
21     }
22     push_up(nrt);
23     return nrt;
24 }
25 int query(int lrt, int rrt, int k, int be, int en) {
26     if (be >= en) return be;
27     int delta = sum[ch[rrt][0]] - sum[ch[lrt][0]];
28     int mid = (be + en) >> 1;
29     int ans = -1;
30     if (delta > k) ans = query(ch[lrt][0], ch[rrt][0], k, be, mid);
31     delta = sum[ch[rrt][1]] - sum[ch[lrt][1]];
32     if (ans == -1 && delta > k) ans = query(ch[lrt][1], ch[rrt][1], k, mid + 1, en);
33     return ans;
34 }
35 } tree;
36 int ai[MAXN], root[MAXN];
37 int main() {
38     int n, q; scanf("%d%d", &n, &q);
39     for (int i = 1; i <= n; i++) scanf("%d", &ai[i]);
40     root[0] = 0;
41     for (int i = 1; i <= n; i++) {
42         root[i] = tree.change(root[i - 1], ai[i], 1, 1, n);
43     }
44     while (q--) {
45         int l, r, k; scanf("%d%d%d", &l, &r, &k);
46         int K = (r - l + 1) / k;
47         int res = tree.query(root[l-1], root[r], K, 1, n);
48         printf("%d\n", res);
49     }
50 }

```