
Platelet

Team Reference Material
(unlimited version)



凌皓煜
陈彤
顾逸

Contents

1	Graph Theory	4
1.1	2-SAT (ct)	4
1.2	割点与桥 (ct)	5
1.3	K 短路 (lhy)	6
1.4	最大团	10
1.5	一般图最大匹配	10
1.6	带花树	10
1.7	KM 算法	10
1.8	支配树	10
1.8.1	DAG (ct)	10
1.8.2	一般图	11
1.9	虚树 (ct)	11
1.10	树上点分治 (ct)	13
1.11	树上倍增 (ct)	14
1.12	Prufer 编码	14
1.13	Link-Cut Tree (ct)	14
1.14	仙人掌	16
1.14.1	圆方树	16
1.15	最小割	17
1.16	最大流 (ct)	17
1.17	费用流 (ct)	17
1.18	有上下界的网络流 (Durandal)	18
1.19	zkw 费用流	19
1.20	差分约束	19
1.21	图论知识	19
2	Math	20
2.1	int64 相乘取模 (Durandal)	20
2.2	扩展欧几里得 (gy)	20
2.3	中国剩余定理 (Durandal)	21
2.4	线性同余不等式 (Durandal)	21
2.5	组合数	21
2.6	高斯消元	21
2.7	Miller Rabin & Pollard Rho (gy)	21
2.8	$O(m^2 \log n)$ 线性递推	23
2.9	Polynomial	23
2.9.1	FFT	23
2.9.2	NTT & 多项式求逆	23
2.10	拉格朗日插值	23
2.11	杜教筛	23
2.12	BSGS (ct,gy)	23
2.12.1	BSGS	23
2.12.2	扩展 BSGS	24
2.13	直线下整点个数 (gy)	25

2.14	单纯形	25
2.15	辛普森积分	25
2.16	数学知识	25
3	Geometry	28
3.1	点、直线、圆 (gy)	28
3.2	平面最近点对 (Grimoire)	31
3.3	凸包游戏 (Grimoire)	32
3.4	半平面交 (Grimoire)	34
3.5	点在多边形内 (Grimoire)	35
3.6	最小圆覆盖 (Grimoire)	35
3.7	最小球覆盖 (Grimoire)	36
3.8	圆并 (Grimoire)	37
3.9	圆与多边形并 (Grimoire)	40
3.10	三角剖分 (Grimoire)	41
3.11	三维几何基础 (Grimoire)	43
3.12	三维凸包 (Grimoire)	44
3.13	三维绕轴旋转 (Grimoire)	45
3.14	几何知识 (gy)	46
4	String	48
4.1	KMP (ct)	48
4.2	AC 自动机 (ct)	49
4.3	后缀数组 (ct)	50
4.4	后缀自动机	51
4.5	Manacher (ct)	51
4.6	回文自动机 (ct)	51
4.7	最小表示法 (ct)	52
5	Data Structure	54
5.1	莫队 (ct)	54
5.2	ST 表 (ct)	54
5.3	带权并查集 (ct)	55
5.4	可并堆 (ct)	56
5.5	zkw 线段树 (ct)	56
5.6	Splay (ct)	57
5.7	Treap (ct)	61
5.8	可持久化平衡树 (ct)	62
5.9	CDQ 分治 (ct)	65
5.10	Bitset (ct)	66
6	Others	68
6.1	vimrc (gy)	68
6.2	STL 释放内存 (Durandal)	68
6.3	开栈 (Durandal)	68
6.4	O3 (gy)	69
6.5	Java Template (gy)	69
6.6	Big Fraction (gy)	71
6.7	模拟退火 (ct)	71
6.8	三分 (ct)	72
6.9	Zeller Congruence (gy)	72
6.10	博弈论模型 (gy)	72
6.11	积分表 (integral-table.com)	73

Chapter 1

Graph Theory

1.1 2-SAT (ct)

```
1 struct Edge {
2     Edge *next;
3     int to;
4 } *last[maxn << 1], e[maxn << 2], *ecnt = e;
5 inline void link(int a, int b)
6 {
7     *++ecnt = (Edge) {last[a], b}; last[a] = ecnt;
8 }
9 int dfn[maxn], low[maxn], timer, st[maxn], top, id[maxn], colcnt, n;
10 bool fail, used[maxn];
11 void tarjan(int x, int fa)
12 {
13     dfn[x] = low[x] = ++timer; st[++top] = x;
14     for (R Edge *iter = last[x]; iter; iter = iter -> next)
15         if (iter -> to != fa)
16             {
17                 if (!dfn[iter -> to])
18                     {
19                         tarjan(iter -> to, x);
20                         cmin(low[x], low[iter -> to]);
21                     }
22                 else if (!id[iter -> to]) cmin(low[x], dfn[iter -> to]);
23             }
24     if (dfn[x] == low[x])
25     {
26         ++colcnt; bool flag = 1;
27         for (; ;)
28             {
29                 int now = st[top--];
30                 id[now] = colcnt;
31                 if (now <= 2 * n)
32                     {
33                         flag &= !used[id[now <= n ? now + n : now - n]];
34                         now <= n ? fail |= (id[now + n] == id[now]) : fail |= (id[now - n] == id[now]);
35                     }
36                 if (now == x) break;
37             }
38         used[colcnt] = flag;
39     }
40 }
41 int ans[maxn], tot;
42 int main()
```

```

43 {
44     /*
45      build your graph here.
46     */
47     for (R int i = 1; !fail && i <= n; ++i) if (!dfn[i]) tarjan(i, 0);
48     if (fail)
49     {
50         puts("Impossible");
51         return 0;
52     }
53     for (R int i = 1; i <= n; ++i) if (used[id[i]]) ans[++tot] = i;
54     printf("%d\n", tot);
55     std::sort(ans + 1, ans + tot + 1);
56     for (R int i = 1; i <= tot; ++i) printf("%d ", ans[i]);
57     return 0;
58 }

```

1.2 割点与桥 (ct)

割点

```

1  int dfn[maxn], low[maxn], timer, ans, num;
2  void tarjan(int x, int fa)
3  {
4      dfn[x] = low[x] = ++timer;
5      for (Edge *iter = last[x]; iter; iter = iter -> next)
6          if (iter -> to != fa)
7              {
8                  if (!dfn[iter -> to])
9                      {
10                         tarjan(iter -> to, x);
11                         cmin(low[x], low[iter -> to]);
12                         if (dfn[x] <= low[iter -> to])
13                             {
14                                 cut[x] = 1;
15                                 if (!fa && dfn[x] < low[iter -> to]) num = 233;
16                                 else if (!fa) ++num;
17                             }
18                         }
19                     else cmin(low[x], dfn[iter -> to]);
20             }
21 }
22 int main()
23 {
24     for (int i = 1; i <= n; ++i)
25         if (!dfn[i])
26             {
27                 num = 0;
28                 tarjan(i, 0);
29                 if (num == 1) cut[i] = 0;
30             }
31 }

```

桥

```

1  int dfn[maxn], low[maxn], timer;
2  void tarjan(int x, int fa)
3  {

```

```

4   dfn[x] = low[x] = ++timer;
5   for (R Edge *iter = last[x]; iter; iter = iter -> next)
6       if (iter -> to != fa)
7           {
8               if (!dfn[iter -> to])
9                   {
10                      dfs(iter -> to, x);
11                      cmin(low[x], low[iter -> to]);
12                      if (dfn[x] < low[iter -> to]) ans[x][iter -> to] = ans[iter -> to][x] = 1;
13                  }
14              else cmin(low[x], dfn[iter -> to]);
15          }
16 }

```

1.3 K 短路 (lhy)

```

1   const int MAXNODE = MAXN + MAXM * 2;
2   bool used[MAXN];
3   int n, m, cnt, S, T, Kth, N, TT;
4   int rt[MAXN], seq[MAXN], adj[MAXN], from[MAXN], dep[MAXN];
5   LL dist[MAXN], w[MAXM], ans[MAXK];
6   struct GivenEdge{
7       int u, v, w;
8       GivenEdge() {};
9       GivenEdge(int _u, int _v, int _w) : u(_u), v(_v), w(_w){};
10  }edge[MAXM];
11  struct Edge{
12      int v, nxt, w;
13      Edge() {};
14      Edge(int _v, int _nxt, int _w) : v(_v), nxt(_nxt), w(_w) {};
15  }e[MAXM];
16  inline void addedge(int u, int v, int w)
17  {
18      e[++cnt] = Edge(v, adj[u], w); adj[u] = cnt;
19  }
20  void dij(int S)
21  {
22      for(int i = 1; i <= N; i++)
23          {
24              dist[i] = INF;
25              dep[i] = 0x3f3f3f3f;
26              used[i] = false;
27              from[i] = 0;
28          }
29      static priority_queue<pair<LL, int>, vector<pair<LL, int> >, greater<pair<LL, int> > > hp;
30      while(!hp.empty())hp.pop();
31      hp.push(make_pair(dist[S] = 0, S));
32      dep[S] = 1;
33      while(!hp.empty())
34          {
35              pair<LL, int> now = hp.top();
36              hp.pop();
37              int u = now.second;
38              if(used[u])continue;

```

```

39     else used[u] = true;
40     for(int p = adj[u]; p; p = e[p].nxt)
41     {
42         int v = e[p].v;
43         if(dist[u] + e[p].w < dist[v])
44         {
45             dist[v] = dist[u] + e[p].w;
46             dep[v] = dep[u] + 1;
47             from[v] = p;
48             hp.push(make_pair(dist[v], v));
49         }
50     }
51 }
52 for(int i = 1; i <= m; i++)    w[i] = 0;
53 for(int i = 1; i <= N; i++)
54     if(from[i])w[from[i]] = -1;
55 for(int i = 1; i <= m; i++)
56 {
57     if(~w[i] && dist[edge[i].u] < INF && dist[edge[i].v] < INF)
58     {
59         w[i] = -dist[edge[i].u] + (dist[edge[i].v] + edge[i].w);
60     }
61     else
62     {
63         w[i] = -1;
64     }
65 }
66 }

67 inline bool cmp_dep(int p, int q)
68 {
69     return dep[p] < dep[q];
70 }

71 struct Heap{
72     LL key;
73     int id, lc, rc, dist;
74     Heap() {};
75     Heap(LL k, int i, int l, int r, int d) : key(k), id(i), lc(l), rc(r), dist(d) {};
76     inline void clear()
77     {
78         key = 0;
79         id = lc = rc = dist = 0;
80     }
81 }hp[MAXNODE];

82 inline int merge_simple(int u, int v)
83 {
84     if(!u)return v;
85     if(!v)return u;
86     if(hp[u].key > hp[v].key)
87     {
88         swap(u, v);
89     }
90     hp[u].rc = merge_simple(hp[u].rc, v);
91     if(hp[hp[u].lc].dist < hp[hp[u].rc].dist)
92     {
93         swap(hp[u].lc, hp[u].rc);
94     }
95     hp[u].dist = hp[hp[u].rc].dist + 1;
96     return u;

```

```

97 }

98 inline int merge_full(int u, int v)
99 {
100     if(!u)return v;
101     if(!v)return u;
102     if(hp[u].key > hp[v].key)
103     {
104         swap(u, v);
105     }
106     int nownode = ++cnt;
107     hp[nownode] = hp[u];
108     hp[nownode].rc = merge_full(hp[nownode].rc, v);
109     if(hp[hp[nownode].lc].dist < hp[hp[nownode].rc].dist)
110     {
111         swap(hp[nownode].lc, hp[nownode].rc);
112     }
113     hp[nownode].dist = hp[hp[nownode].rc].dist + 1;
114     return nownode;
115 }

116 priority_queue<pair<LL, int>, vector<pair<LL, int> >, greater<pair<LL, int> > > Q;

117 int main()
118 {
119     while(scanf("%d%d", &n, &m) != EOF)
120     {
121         scanf("%d%d%d", &S, &T, &Kth, &TT);
122         for(int i = 1; i <= m; i++)
123         {
124             int u, v, w;
125             scanf("%d%d%d", &u, &v, &w);
126             edge[i] = {u, v, w};
127         }
128         N = n;
129         memset(adj, 0, sizeof(*adj) * (N + 1));
130         cnt = 0;
131         for(int i = 1; i <= m; i++)
132             addedge(edge[i].v, edge[i].u, edge[i].w);
133         dijkstra(T);
134         if(dist[S] > TT)
135         {
136             puts("Whitesnake!");
137             continue;
138         }
139         for(int i = 1; i <= N; i++)
140             seq[i] = i;
141         sort(seq + 1, seq + N + 1, cmp_dep);

142         cnt = 0;
143         memset(adj, 0, sizeof(*adj) * (N + 1));
144         memset(rt, 0, sizeof(*rt) * (N + 1));
145         for(int i = 1; i <= m; i++)
146             addedge(edge[i].u, edge[i].v, edge[i].w);
147         rt[T] = cnt = 0;
148         hp[0].dist = -1;
149         for(int i = 1; i <= N; i++)
150         {
151             int u = seq[i], v = edge[from[u]].v;
152             rt[u] = 0;
153             for(int p = adj[u]; p; p = e[p].nxt)

```



```

154     {
155         if(~w[p])
156         {
157             hp[++cnt] = Heap(w[p], p, 0, 0, 0);
158             rt[u] = merge_simple(rt[u], cnt);
159         }
160     }
161     if(i == 1)continue;
162     rt[u] = merge_full(rt[u], rt[v]);
163 }
164 while(!Q.empty())Q.pop();
165 Q.push(make_pair(dist[S], 0));
166 edge[0].v = S;
167 for(int kth = 1; kth <= Kth; kth++)
168 {
169     if(Q.empty())
170     {
171         ans[kth] = -1;
172         continue;
173     }
174     pair<LL, int> now = Q.top(); Q.pop();
175     ans[kth] = now.first;
176     int p = now.second;
177     if(hp[p].lc)
178     {
179         Q.push(make_pair(+hp[hp[p].lc].key + now.first - hp[p].key, hp[p].lc));
180     }
181     if(hp[p].rc)
182     {
183         Q.push(make_pair(+hp[hp[p].rc].key + now.first - hp[p].key, hp[p].rc));
184     }
185     if(rt[edge[hp[p].id].v])
186     {
187         Q.push(make_pair(hp[rt[edge[hp[p].id].v]].key + now.first, rt[edge[hp[p].id].v]));
188     }
189 }
190 if(ans[Kth] == -1 || ans[Kth] > TT)
191 {
192     puts("Whitesnake!");
193 }
194 else
195 {
196     puts("yareyaredawa");
197 }
198 }
199 }

```

1.4 最大团

1.5 一般图最大匹配

1.6 带花树

1.7 KM 算法

1.8 支配树

1.8.1 DAG (ct)

```

1 struct Edge {
2     Edge *next;
3     int to;
4 } ;
5 Edge *last[maxn], e[maxn], *ecnt = e; // original graph
6 Edge *rlast[maxn], re[maxn], *recnt = re; // reversed-edge graph
7 Edge *tlast[maxn], te[maxn], *tecnt = te; // dominate tree graph
8 int deg[maxn], q[maxn], fa[maxn][20], all_fa[maxn], fa_cnt, size[maxn], dep[maxn];
9 inline void link(int a, int b)
10 {
11     *++ecnt = (Edge) {last[a], b}; last[a] = ecnt; ++deg[b];
12 }
13 inline void link_rev(R int a, R int b)
14 {
15     *++recnt = (Edge) {rlast[a], b}; rlast[a] = recnt;
16 }
17 inline void link_tree(R int a, R int b)
18 {
19     *++tecnt = (Edge) {tlast[a], b}; tlast[a] = tecnt;
20 }
21 inline int getlca(R int a, R int b)
22 {
23     if (dep[a] < dep[b]) std::swap(a, b);
24     R int temp = dep[a] - dep[b];
25     for (R int i; temp; temp -= 1 << i)
26         a = fa[a][i = __builtin_ctz(temp)];
27     for (R int i = 16; ~i; --i)
28         if (fa[a][i] != fa[b][i])
29             a = fa[a][i], b = fa[b][i];
30     if (a == b) return a;
31     return fa[a][0];
32 }
33 void dfs(R int x)
34 {
35     size[x] = 1;
36     for (R Edge *iter = tlast[x]; iter; iter = iter -> next)
37         dfs(iter -> to), size[x] += size[iter -> to];
38 }
39 int main()
40 {
41     q[1] = 0;
42     R int head = 0, tail = 1;
43     while (head < tail)
44     {
45         R int now = q[++head];
46         fa_cnt = 0;

```

```

47     for (R Edge *iter = rlast[now]; iter; iter = iter -> next)
48         all_fa[++fa_cnt] = iter -> to;
49     for (; fa_cnt > 1; --fa_cnt)
50         all_fa[fa_cnt - 1] = getlca(all_fa[fa_cnt], all_fa[fa_cnt - 1]);
51     fa[now][0] = all_fa[fa_cnt];
52     dep[now] = dep[all_fa[fa_cnt]] + 1;
53     if (now) link_tree(fa[now][0], now);

54     for (R int i = 1; i <= 16; ++i)
55         fa[now][i] = fa[fa[now][i - 1]][i - 1];
56     for (R Edge *iter = last[now]; iter; iter = iter -> next)
57         if (--deg[iter -> to] == 0) q[++tail] = iter -> to;
58 }
59 dfs(0);
60 for (R int i = 1; i <= n; ++i) printf("%d\n", size[i] - 1);
61 return 0;
62 }

```

1.8.2 一般图

1.9 虚树 (ct)

```

1 struct Edge {
2     Edge *next;
3     int to;
4 } *last[maxn], e[maxn << 1], *ecnt = e;
5 inline void link(int a, int b)
6 {
7     *++ecnt = (Edge) {last[a], b}; last[a] = ecnt;
8     *++ecnt = (Edge) {last[b], a}; last[b] = ecnt;
9 }
10 int a[maxn], n, dfn[maxn], pos[maxn], timer, inv[maxn], st[maxn];
11 int fa[maxn], size[maxn], dep[maxn], son[maxn], top[maxn];
12 bool vis[maxn];
13 void dfs1(int x)
14 {
15     vis[x] = 1; size[x] = 1; dep[x] = dep[fa[x]] + 1;
16     for (R Edge *iter = last[x]; iter; iter = iter -> next)
17         if (!vis[iter -> to])
18             {
19                 fa[iter -> to] = x;
20                 dfs1(iter -> to);
21                 size[x] += size[iter -> to];
22                 size[son[x]] < size[iter -> to] ? son[x] = iter -> to : 0;
23             }
24 }
25 void dfs2(int x)
26 {
27     vis[x] = 0; top[x] = x == son[fa[x]] ? top[fa[x]] : x;
28     dfn[x] = ++timer; pos[timer] = x;
29     if (son[x]) dfs2(son[x]);
30     for (R Edge *iter = last[x]; iter; iter = iter -> next)
31         if (vis[iter -> to]) dfs2(iter -> to);
32     inv[x] = timer;
33 }
34 inline int getlca(int a, int b)
35 {
36     while (top[a] != top[b])
37         dep[top[a]] < dep[top[b]] ? b = fa[top[b]] : a = fa[top[a]];

```

```

38     return dep[a] < dep[b] ? a : b;
39 }
40 inline bool cmp(int a, int b)
41 {
42     return dfn[a] < dfn[b];
43 }
44 inline bool isson(int a, int b)
45 {
46     return dfn[a] <= dfn[b] && dfn[b] <= inv[a];
47 }
48 typedef long long ll;
49 bool imp[maxn];
50 struct sEdge {
51     sEdge *next;
52     int to, w;
53 } *slast[maxn], se[maxn << 1], *secnt = se;
54 inline void slink(int a, int b, int w)
55 {
56     *++secnt = (sEdge) {slast[a], b, w}; slast[a] = secnt;
57 }
58 int main()
59 {
60     scanf("%d", &n);
61     for (int i = 1; i < n; ++i)
62     {
63         int a, b; scanf("%d%d", &a, &b);
64         link(a, b);
65     }
66     int m; scanf("%d", &m);
67     dfs1(1); dfs2(1);
68     memset(size, 0, (n + 1) << 2);
69     for (; m; --m)
70     {
71         int top = 0; scanf("%d", &k);
72         for (int i = 1; i <= k; ++i) scanf("%d", &a[i]), vis[a[i]] = imp[a[i]] = 1;
73         std::sort(a + 1, a + k + 1, cmp);
74         int p = k;
75         for (int i = 1; i < k; ++i)
76         {
77             int lca = getlca(a[i], a[i + 1]);
78             if (!vis[lca]) vis[a[++p] = lca] = 1;
79         }
80         std::sort(a + 1, a + p + 1, cmp);
81         st[++top] = a[1];
82         for (int i = 2; i <= p; ++i)
83         {
84             while (!isson(st[top], a[i])) --top;
85             slink(st[top], a[i], dep[a[i]] - dep[st[top]]);
86             st[++top] = a[i];
87         }
88         /*
89         write your code here.
90         */
91         for (int i = 1; i <= p; ++i) vis[a[i]] = imp[a[i]] = 0, slast[a[i]] = 0;
92         secnt = se;
93     }
94     return 0;
95 }

```

1.10 树上点分治 (ct)

```

1  int root, son[maxn], size[maxn], sum;
2  bool vis[maxn];
3  void dfs_root(int x, int fa)
4  {
5      size[x] = 1; son[x] = 0;
6      for (R Edge *iter = last[x]; iter; iter = iter -> next)
7      {
8          if (iter -> to == fa || vis[iter -> to]) continue;
9          dfs_root(iter -> to, x);
10         size[x] += size[iter -> to];
11         cmax(son[x], size[iter -> to]);
12     }
13     cmax(son[x], sum - size[x]);
14     if (!root || son[x] < son[root]) root = x;
15 }
16 void dfs_chain(int x, int fa, int st1, int st2)
17 {
18     /*
19      write your code here.
20     */
21     for (Edge *iter = last[x]; iter; iter = iter -> next)
22     {
23         if (vis[iter -> to] || iter -> to == fa) continue;
24         dfs_chain(iter -> to, x);
25     }
26 }
27 void calc(int x)
28 {
29     for (Edge *iter = last[x]; iter; iter = iter -> next)
30     {
31         if (vis[iter -> to]) continue;
32         dfs_chain(iter -> to, x);
33         /*
34          write your code here.
35         */
36     }
37 }
38 void work(int x)
39 {
40     vis[x] = 1;
41     calc(x);
42     for (R Edge *iter = last[x]; iter; iter = iter -> next)
43     {
44         if (vis[iter -> to]) continue;
45         root = 0;
46         sum = size[iter -> to];
47         dfs_root(iter -> to, 0);
48         work(root);
49     }
50 }
51 int main()
52 {
53     root = 0; sum = n;
54     dfs_root(1, 0);
55     work(root);
56     return 0;
57 }

```

1.11 树上倍增 (ct)

```

1  int fa[maxn][17], mn[maxn][17], dep[maxn];
2  bool vis[maxn];
3  void dfs(int x)
4  {
5      vis[x] = 1;
6      for (int i = 1; i <= 16; ++i)
7      {
8          if (dep[x] < (1 << i)) break;
9          fa[x][i] = fa[fa[x][i - 1]][i - 1];
10         mn[x][i] = dmin(mn[x][i - 1], mn[fa[x][i - 1]][i - 1]);
11     }
12     for (Edge *iter = last[x]; iter; iter = iter -> next)
13         if (!vis[iter -> to])
14         {
15             fa[iter -> to][0] = x;
16             mn[iter -> to][0] = iter -> w;
17             dep[iter -> to] = dep[x] + 1;
18             dfs(iter -> to);
19         }
20 }
21 inline int getlca(int x, int y)
22 {
23     if (dep[x] < dep[y]) std::swap(x, y);
24     int t = dep[x] - dep[y];
25     for (int i = 0; i <= 16 && t; ++i)
26         if ((1 << i) & t)
27             x = fa[x][i], t ^= 1 << i;
28     for (int i = 16; i >= 0; --i)
29         if (fa[x][i] != fa[y][i])
30         {
31             x = fa[x][i];
32             y = fa[y][i];
33         }
34     if (x == y) return x;
35     return fa[x][0];
36 }
37 inline int getans(int x, int f)
38 {
39     int ans = inf, t = dep[x] - dep[f];
40     for (int i = 0; i <= 16 && t; ++i)
41         if (t & (1 << i))
42         {
43             cmin(ans, mn[x][i]);
44             x = fa[x][i];
45             t ^= 1 << i;
46         }
47     return ans;
48 }

```

1.12 Prufer 编码

1.13 Link-Cut Tree (ct)

```

1  struct Node *null;
2  struct Node {
3      Node *ch[2], *fa, *pos;

```

```

4   int val, mn, l, len; bool rev;
5   // min_val in chain
6   inline bool type()
7   {
8       return fa -> ch[1] == this;
9   }
10  inline bool check()
11  {
12      return fa -> ch[type()] == this;
13  }
14  inline void pushup()
15  {
16      pos = this; mn = val;
17      ch[0] -> mn < mn ? mn = ch[0] -> mn, pos = ch[0] -> pos : 0;
18      ch[1] -> mn < mn ? mn = ch[1] -> mn, pos = ch[1] -> pos : 0;
19      len = ch[0] -> len + ch[1] -> len + 1;
20  }
21  inline void pushdown()
22  {
23      if (rev)
24      {
25          ch[0] -> rev ^= 1;
26          ch[1] -> rev ^= 1;
27          std::swap(ch[0], ch[1]);
28          rev ^= 1;
29      }
30  }
31  inline void pushdownall()
32  {
33      if (check()) fa -> pushdownall();
34      pushdown();
35  }
36  inline void rotate()
37  {
38      bool d = type(); Node *f = fa, *gf = f -> fa;
39      (fa = gf, f -> check()) ? fa -> ch[f -> type()] = this : 0;
40      (f -> ch[d] = ch[!d]) != null ? ch[!d] -> fa = f : 0;
41      (ch[!d] = f) -> fa = this;
42      f -> pushup();
43  }
44  inline void splay(bool need = 1)
45  {
46      if (need) pushdownall();
47      for (; check(); rotate())
48          if (fa -> check())
49              (type() == fa -> type() ? fa : this) -> rotate();
50      pushup();
51  }
52  inline Node *access()
53  {
54      Node *i = this, *j = null;
55      for (; i != null; i = (j = i) -> fa)
56      {
57          i -> splay();
58          i -> ch[1] = j;
59          i -> pushup();
60      }
61      return j;
62  }
63  inline void make_root()
64  {

```

```

65     access();
66     splay();
67     rev ^= 1;
68 }
69 inline void link(Node *that)
70 {
71     make_root();
72     fa = that;
73     splay(0);
74 }
75 inline void cut(Node *that)
76 {
77     make_root();
78     that -> access();
79     that -> splay(0);
80     that -> ch[0] = fa = null;
81     that -> pushup();
82 }
83 } mem[maxn];
84 inline Node *query(Node *a, Node *b)
85 {
86     a -> make_root(); b -> access(); b -> splay(0);
87     return b -> pos;
88 }
89 inline int dist(Node *a, Node *b)
90 {
91     a -> make_root(); b -> access(); b -> splay(0);
92     return b -> len;
93 }

```

1.14 仙人掌

1.14.1 圆方树

```

1  int dfn[maxn], low[maxn], timer, st[maxn], top, id[maxn], scc;
2  void dfs(int x)
3  {
4      dfn[x] = low[x] = ++timer; st[++top] = x;
5      for (Edge *iter = last[x]; iter; iter = iter -> next)
6          if (!dfn[iter -> to])
7              {
8                  dfs(iter -> to);
9                  cmin(low[x], low[iter -> to]);
10                 if (dfn[x] == low[iter -> to])
11                     {
12                         int now, elder = top, minn = c[x];
13                         ++scc;
14                         do
15                             {
16                                 now = st[top--];
17                                 cmin(minn, c[now]);
18                             }
19                         while (iter -> to != now);
20                         for (int i = top + 1; i <= elder; ++i)
21                             add(scc, st[i], minn);
22                         add(scc, x, minn);
23                     }
24             }
25     else if (!id[iter -> to]) cmin(low[x], dfn[iter -> to]);

```



```
26 }
```

1.15 最小割

1.16 最大流 (ct)

```
1 struct Edge {
2     Edge *next, *rev;
3     int to, cap;
4 } *last[maxn], *cur[maxn], e[maxn], *ecnt = e;
5 inline void link(R int a, R int b, R int w)
6 {
7     *++ecnt = (Edge) {last[a], ecnt + 1, b, w}; last[a] = ecnt;
8     *++ecnt = (Edge) {last[b], ecnt - 1, a, 0}; last[b] = ecnt;
9 }
10 int ans, s, t, q[maxn], dep[maxn];
11 inline bool bfs()
12 {
13     memset(dep, -1, (t + 1) << 2);
14     dep[q[1] = t] = 0; int head = 0, tail = 1;
15     while (head < tail)
16     {
17         int now = q[++head];
18         for (Edge *iter = last[now]; iter; iter = iter -> next)
19             if (dep[iter -> to] == -1 && iter -> rev -> cap)
20                 dep[q[++tail] = iter -> to] = dep[now] + 1;
21     }
22     return dep[s] != -1;
23 }
24 int dfs(int x, int f)
25 {
26     if (x == t) return f;
27     int used = 0;
28     for (Edge* &iter = cur[x]; iter; iter = iter -> next)
29         if (iter -> cap && dep[iter -> to] + 1 == dep[x])
30         {
31             int v = dfs(iter -> to, dmin(f - used, iter -> cap));
32             iter -> cap -= v;
33             iter -> rev -> cap += v;
34             used += v;
35             if (used == f) return f;
36         }
37     return used;
38 }
39 inline void dinic()
40 {
41     while (bfs())
42     {
43         memcpy(cur, last, sizeof cur);
44         ans += dfs(s, inf);
45     }
46 }
```

1.17 费用流 (ct)

```
1 struct Edge {
2     Edge *next, *rev;
```

```

3     int from, to, cap, cost;
4 } *last[maxn], *prev[maxn], e[maxn], *ecnt = e;
5 inline void link(int a, int b, int w, int c)
6 {
7     *++ecnt = (Edge) {last[a], ecnt + 1, a, b, w, c}; last[a] = ecnt;
8     *++ecnt = (Edge) {last[b], ecnt - 1, b, a, 0, -c}; last[b] = ecnt;
9 }
10 int s, t, q[maxn << 2], dis[maxn];
11 ll ans;
12 bool inq[maxn];
13 #define inf 0x7fffffff
14 inline bool spfa()
15 {
16     for (int i = 1; i <= t; ++i) dis[i] = inf;
17     int head = 0, tail = 1; dis[q[1] = s] = 0;
18     while (head < tail)
19     {
20         int now = q[++head]; inq[now] = 0;
21         for (Edge *iter = last[now]; iter; iter = iter -> next)
22             if (iter -> cap && dis[iter -> to] > dis[now] + iter -> cost)
23             {
24                 dis[iter -> to] = dis[now] + iter -> cost;
25                 prev[iter -> to] = iter;
26                 !inq[iter -> to] ? inq[q[++tail] = iter -> to] = 1 : 0;
27             }
28     }
29     return dis[t] != inf;
30 }
31 inline void mcmf()
32 {
33     int x = inf;
34     for (Edge *iter = prev[t]; iter; iter = prev[iter -> from]) cmin(x, iter -> cap);
35     for (Edge *iter = prev[t]; iter; iter = prev[iter -> from])
36     {
37         iter -> cap -= x;
38         iter -> rev -> cap += x;
39         ans += 1ll * x * iter -> cost;
40     }
41 }

```

1.18 有上下界的网络流 (Durandal)

$B(u, v)$ 表示边 (u, v) 流量的下界, $C(u, v)$ 表示边 (u, v) 流量的上界, 设 $F(u, v)$ 表示边 (u, v) 的实际流量 设 $G(u, v) = F(u, v) - B(u, v)$, 则 $0 \leq G(u, v) \leq C(u, v) - B(u, v)$

- 无源汇的上下界可行流
建立超级源点 S^* 和超级汇点 T^* , 对于原图每一条边 (u, v) 在新网络中连如下三条边: $S^* \rightarrow v$, 容量为 $B(u, v)$; $u \rightarrow T^*$, 容量为 $B(u, v)$; $u \rightarrow v$, 容量为 $C(u, v) - B(u, v)$ 。最后求新网络的最大流, 判断从超级源点 S^* 出发的边是否都满流即可, 边 (u, v) 的最终解中的实际流量为 $G(u, v) + B(u, v)$ 。
- 有源汇的上下界可行流
从汇点 T 到源点 S 连一条上界为 ∞ , 下界为 0 的边。按照无源汇的上下界可行流一样做即可, 流量即为 $T \rightarrow S$ 边上的流量。
- 有源汇的上下界最大流
 - 在有源汇的上下界可行流中, 从汇点 T 到源点 S 的边改为连一条上界为 ∞ , 下界为 x 的边。 x 满足二分性质, 找到最大的 x 使得新网络存在有源汇的上下界可行流即为原图的最大流。

- 从汇点 T 到源点 S 连一条上界为 ∞ ，下界为 0 的边，变成无源汇的网络。按照无源汇的上下界可行流的方法，建立超级源点 S^* 与超级汇点 T^* ，求一遍 $S^* \rightarrow T^*$ 的最大流，再将从汇点 T 到源点 S 的这条边拆掉，求一次 $S \rightarrow T$ 的最大流即可。
- 有源汇的上下界最小流
 - 在有源汇的上下界可行流中，从汇点 T 到源点 S 的边改为连一条上界为 x ，下界为 0 的边。 x 满足二分性质，找到最小的 x 使得新网络存在有源汇的上下界可行流即为原图的最大流。
 - 按照无源汇的上下界可行流的方法，建立超级源点 S^* 与超级汇点 T^* ，求一遍 $S^* \rightarrow T^*$ 的最大流，但是注意不加上汇点 T 到源点 S 的这条边，即不使之改为无源汇的网络去求解。求完后，再加上那条汇点 T 到源点 S 的边，上界为 ∞ 的边。因为这条边的下界为 0，所以 S^*, T^* 无影响，再求一次 $S^* \rightarrow T^*$ 的最大流。若超级源点 S^* 出发的边全部满流，则 $T \rightarrow S$ 边上的流量即为原图的最小流，否则无解。

1.19 zkw 费用流

1.20 差分约束

1.21 图论知识

弦图

树的计数

- 有根树计数

$$a_1 = 1$$

$$a_{n+1} = \frac{\sum_{j=1}^n j \cdot a_j \cdot S_{n,j}}{n}$$

$$S_{n,j} = \sum_{i=1}^{n/j} a_{n+1-ij} = S_{n-j,j} + a_{n+1-j}$$

- 无根树计数

$$\begin{cases} a_n - \sum_{i=1}^{n/2} a_i a_{n-i} & n \text{ is odd} \\ a_n - \sum_{i=1}^{n/2} a_i a_{n-i} + \frac{1}{2} a_{\frac{n}{2}} (a_{\frac{n}{2}} + 1) & n \text{ is even} \end{cases}$$

- 完全图生成树计数

$$n^{n-2}$$

- 矩阵-树定理

设 $\mathbf{A}[G]$ 为图 G 的邻接矩阵、 $\mathbf{D}[G]$ 为图 G 的度数矩阵，则图 G 的不同生成树的个数为 $\mathbf{C}[G] = \mathbf{D}[G] - \mathbf{A}[G]$ 的任意一个 $n-1$ 阶主子式的行列式值。

Chapter 2

Math

2.1 int64 相乘取模 (Durandal)

```
1 int64_t mul(int64_t x, int64_t y, int64_t p) {
2     int64_t t = (x * y - (int64_t) ((long double) x / p * y + 1e-3) * p) % p;
3     return t < 0 ? t + p : t;
4 }
```

2.2 扩展欧几里得 (gy)

```
1 // return gcd(a, b)
2 // ax+by=gcd(a,b)
3 int extend_gcd(int a, int b, int &x, int &y) {
4     if (b == 0) {
5         x = 1, y = 0;
6         return a;
7     }
8     int res = extend_gcd(b, a % b, x, y);
9     int t = y;
10    y = x - a / b * y;
11    x = t;
12    return res;
13 }
14 // return minimal positive integer x so that ax+by=c
15 // or -1 if such x does not exist
16 int solve_equ(int a, int b, int c) {
17     int x, y, d;
18     d = extend_gcd(a, b, x, y);
19     if (c % d)
20         return -1;
21     int t = c / d;
22     x *= t;
23     y *= t;
24     int k = b / d;
25     x = (x % k + k) % k;
26     return x;
27 }
28 // return minimal positive integer x so that ax==b(mod p)
29 // or -1 if such x does not exist
30 int solve(int a, int b, int p) {
31     a = (a % p + p) % p;
32     b = (b % p + p) % p;
```

```

33     return solve_equ(a, p, b);
34 }

```

2.3 中国剩余定理 (Durandal)

返回是否可行，余数和模数结果为 r_1, m_1

```

1 bool CRT(int &r1, int &m1, int r2, int m2) {
2     int x, y, g = extend_gcd(m1, m2, x, y);
3     if ((r2 - r1) % g != 0) return false;
4     x = 1ll * (r2 - r1) * x % m2;
5     if (x < 0) x += m2;
6     x /= g;
7     r1 += m1 * x;
8     m1 *= m2 / g;
9     return true;
10 }

```

2.4 线性同余不等式 (Durandal)

必须满足 $0 \leq d < m, 0 \leq l \leq r < m$, 返回 $\min\{x \geq 0 \mid l \leq x \cdot d \bmod m \leq r\}$, 无解返回 -1

```

1 int64_t calc(int64_t d, int64_t m, int64_t l, int64_t r) {
2     if (l == 0) return 0;
3     if (d == 0) return -1;
4     if (d * 2 > m) return calc(m - d, m, m - r, m - l);
5     if ((l - 1) / d < r / d) return (l - 1) / d + 1;
6     int64_t k = calc((-m % d + d) % d, d, l % d, r % d);
7     if (k == -1) return -1;
8     return (k * m + l - 1) / d + 1;
9 }

```

2.5 组合数

2.6 高斯消元

2.7 Miller Rabin & Pollard Rho (gy)

```

1 /*
2  * In Java, use BigInteger.isProbablePrime(int certainty) to replace miller_rabin(BigInteger
3  *   ↪ number)
4  * Test Set / First Wrong Answer
5  * 2 / 2,047
6  * 2, 3 / 1,373,653
7  * 31, 73 / 9,080,191
8  * 2, 3, 5 / 25,326,001
9  * 2, 3, 5, 7 / 3,215,031,751 (> Int.MAX_VALUE)
10 * 2, 7, 61 / 4,759,123,141
11 * 2, 13, 23, 1662803 / 1,122,004,669,633
12 * 2, 3, 5, 7, 11 / 2,152,302,898,747
13 * 2, 3, 5, 7, 11, 13 / 3,474,749,660,383
14 * 2, 3, 5, 7, 11, 13, 17 / 341,550,071,728,321
15 * 2, 3, 5, 7, 11, 13, 17, 19, 23 / 3,825,123,056,546,413,051
16 * 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37 / 318,665,857,834,031,151,167,461 (> Long.MAX_VALUE)
17 * 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41 / 3,317,044,064,679,887,385,961,981

```

```

17  */
18  const int test_case_size = 12;
19  const int test_cases[test_case_size] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};

20  int64_t multiply_mod(int64_t x, int64_t y, int64_t p) {
21      int64_t t = (x * y - (int64_t) ((long double) x / p * y + 1e-3) * p) % p;
22      return t < 0 ? t + p : t;
23  }

24  int64_t add_mod(int64_t x, int64_t y, int64_t p) {
25      return (0ull + x + y) % p;
26  }

27  int64_t power_mod(int64_t x, int64_t exp, int64_t p) {
28      int64_t ans = 1;
29      while (exp) {
30          if (exp & 1)
31              ans = multiply_mod(ans, x, p);
32          x = multiply_mod(x, x, p);
33          exp >>= 1;
34      }
35      return ans;
36  }

37  bool miller_rabin_check(int64_t prime, int64_t base) {
38      int64_t number = prime - 1;
39      for (; ~number & 1; number >>= 1)
40          continue;
41      int64_t result = power_mod(base, number, prime);
42      for (; number != prime - 1 && result != 1 && result != prime - 1; number <<= 1)
43          result = multiply_mod(result, result, prime);
44      return result == prime - 1 || (number & 1) == 1;
45  }

46  bool miller_rabin(int64_t number) {
47      if (number < 2)
48          return false;
49      if (number < 4)
50          return true;
51      if (~number & 1)
52          return false;
53      for (int i = 0; i < test_case_size && test_cases[i] < number; i++)
54          if (!miller_rabin_check(number, test_cases[i]))
55              return false;
56      return true;
57  }

58  int64_t gcd(int64_t x, int64_t y) {
59      return y == 0 ? x : gcd(y, x % y);
60  }

61  int64_t pollard_rho_test(int64_t number, int64_t seed) {
62      int64_t x = rand() % (number - 1) + 1, y = x;
63      int head = 1, tail = 2;
64      while (true) {
65          x = multiply_mod(x, x, number);
66          x = add_mod(x, seed, number);
67          if (x == y)
68              return number;
69          int64_t answer = gcd(std::abs(x - y), number);
70          if (answer > 1 && answer < number)

```

```

71         return answer;
72     if (++head == tail) {
73         y = x;
74         tail <<= 1;
75     }
76 }
77 }

78 void factorize(int64_t number, std::vector<int64_t> &divisor) {
79     if (number > 1) {
80         if (miller_rabin(number)) {
81             divisor.push_back(number);
82         } else {
83             int64_t factor = number;
84             while (factor >= number)
85                 factor = pollard_rho_test(number, rand() % (number - 1) + 1);
86             factorize(number / factor, divisor);
87             factorize(factor, divisor);
88         }
89     }
90 }

```

2.8 $O(m^2 \log n)$ 线性递推

2.9 Polynomial

2.9.1 FFT

2.9.2 NTT & 多项式求逆

2.10 拉格朗日插值

2.11 杜教筛

2.12 BSGS (ct,gy)

2.12.1 BSGS

p 是素数, 返回 $\min\{x \geq 0 \mid y^x \equiv z \pmod p\}$

```

1  const int mod = 19260817;
2  struct Hash
3  {
4      Hash *next;
5      int key, val;
6  } *last[mod], mem[100000], *tot = mem;
7  inline void insert(R int x, R int v)
8  {
9      *++tot = (Hash) {last[x % mod], x, v}; last[x % mod] = tot;
10 }
11 inline int query(R int x)
12 {
13     for (R Hash *iter = last[x % mod]; iter; iter = iter -> next)
14         if (iter -> key == x) return iter -> val;
15     return -1;
16 }
17 inline void del(R int x)

```

```

18 {
19     last[x % mod] = 0;
20 }
21 int main()
22 {
23     for (; T; --T)
24     {
25         R int y, z, p; scanf("%d%d%d", &y, &z, &p);
26         R int m = (int) sqrt(p * 1.0);
27         y %= p; z %= p;
28         if (!y && !z) {puts("0"); continue;}
29         if (!y) {puts("0rz, I cannot find x!"); continue;}
30         R int pw = 1;
31         for (R int i = 0; i < m; ++i, pw = 1ll * pw * y % p) insert(1ll * z * pw % p, i);
32         R int ans = -1;
33         for (R int i = 1, t, pw2 = pw; i <= p / m + 1; ++i, pw2 = 1ll * pw2 * pw % p)
34             if ((t = query(pw2)) != -1) {ans = i * m - t; break;}
35         if (ans == -1) puts("0rz, I cannot find x!");
36         else printf("%d\n", ans);
37         tot = mem; pw = 1;
38         for (R int i = 0; i < m; ++i, pw = 1ll * pw * y % p) del(1ll * z * pw % p);
39     }
40     return 0;
41 }

```

2.12.2 扩展 BSGS

必须满足 $0 \leq a < p$, $0 \leq b < p$, 返回 $\min\{x \geq 0 \mid a^x \equiv b \pmod p\}$

```

1  int64_t ex_bsgs(int64_t a, int64_t b, int64_t p) {
2      if (b == 1)
3          return 0;
4      int64_t t, d = 1, k = 0;
5      while ((t = std::__gcd(a, p)) != 1) {
6          if (b % t) return -1;
7          k++, b /= t, p /= t, d = d * (a / t) % p;
8          if (b == d) return k;
9      }
10     map.clear();
11     int64_t m = std::ceil(std::sqrt((long double) p));
12     int64_t a_m = pow_mod(a, m, p);
13     int64_t mul = b;
14     for (int j = 1; j <= m; j++) {
15         (mul *= a) %= p;
16         map[mul] = j;
17     }
18     for (int i = 1; i <= m; i++) {
19         (d *= a_m) %= p;
20         if (map.count(d))
21             return i * m - map[d] + k;
22     }
23     return -1;
24 }
25
26 int main() {
27     int64_t a, b, p;
28     while (scanf("%lld%lld%lld", &a, &b, &p) != EOF)
29         printf("%lld\n", ex_bsgs(a, b, p));
30     return 0;
31 }

```


2.13 直线下整点个数 (gy)

必须满足 $a \geq 0, b \geq 0, m > 0$, 返回 $\sum_{i=0}^{n-1} \frac{a+bi}{m}$

```

1 int64_t count(int64_t n, int64_t a, int64_t b, int64_t m) {
2     if (b == 0)
3         return n * (a / m);
4     if (a >= m)
5         return n * (a / m) + count(n, a % m, b, m);
6     if (b >= m)
7         return (n - 1) * n / 2 * (b / m) + count(n, a, b % m, m);
8     return count((a + b * n) / m, (a + b * n) % m, m, b);
9 }

```

2.14 单纯形

2.15 辛普森积分

2.16 数学知识

求和公式

- $\sum_{k=1}^n (2k-1)^2 = \frac{1}{3}n(4n^2-1)$
- $\sum_{k=1}^n k^3 = \frac{1}{4}n^2(n+1)^2$
- $\sum_{k=1}^n (2k-1)^3 = n^2(2n^2-1)$
- $\sum_{k=1}^n k^4 = \frac{1}{30}n(n+1)(2n+1)(3n^2+3n-1)$
- $\sum_{k=1}^n k^5 = \frac{1}{12}n^2(n+1)^2(2n^2+2n-1)$
- $\sum_{k=1}^n k(k+1) = \frac{1}{3}n(n+1)(n+2)$
- $\sum_{k=1}^n k(k+1)(k+2) = \frac{1}{4}n(n+1)(n+2)(n+3)$
- $\sum_{k=1}^n k(k+1)(k+2)(k+3) = \frac{1}{5}n(n+1)(n+2)(n+3)(n+4)$

错排公式

D_n 表示 n 个元素错位排列的方案数

$$D_1 = 0, D_2 = 1$$

$$D_n = (n-1)(D_{n-2} + D_{n-1}), n \geq 3$$

$$D_n = n! \cdot \left(1 - \frac{1}{1!} + \frac{1}{2!} - \cdots + (-1)^n \frac{1}{n!}\right)$$

Fibonacci sequence

$$\begin{aligned}
 F_0 &= 0, F_1 = 1 \\
 F_n &= F_{n-1} + F_{n-2} \\
 F_{n+1} \cdot F_{n-1} - F_n^2 &= (-1)^n \\
 F_{-n} &= (-1)^n F_n \\
 F_{n+k} &= F_k \cdot F_{n+1} + F_{k-1} \cdot F_n \\
 \gcd(F_m, F_n) &= F_{\gcd(m, n)} \\
 F_m \mid F_n^2 &\Leftrightarrow n F_n \mid m \\
 F_n &= \frac{\varphi^n - \Psi^n}{\sqrt{5}}, \varphi = \frac{1+\sqrt{5}}{2}, \Psi = \frac{1-\sqrt{5}}{2} \\
 F_n &= \lfloor \frac{\varphi^n}{\sqrt{5}} + \frac{1}{2} \rfloor, n \geq 0 \\
 n(F) &= \lfloor \log_{\varphi}(F \cdot \sqrt{5} + \frac{1}{2}) \rfloor
 \end{aligned}$$

第一类 Stirling number

用 $s(n, k) = (-1)^{n-k} \begin{bmatrix} n \\ k \end{bmatrix}$ 表示第一类 Stirling number

$$\begin{aligned}
 \begin{bmatrix} n+1 \\ k \end{bmatrix} &= n \begin{bmatrix} n \\ k \end{bmatrix} + \begin{bmatrix} n \\ k-1 \end{bmatrix}, k > 0 \\
 \begin{bmatrix} 0 \\ 0 \end{bmatrix} &= 1, \begin{bmatrix} n \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ n \end{bmatrix} = 0, n > 0 \\
 \begin{bmatrix} n \\ k \end{bmatrix} &\text{ 为将 } n \text{ 个元素分成 } k \text{ 个环的方案数}
 \end{aligned}$$

第二类 Stirling number

用 $S(n, k) = \left\{ \begin{smallmatrix} n \\ k \end{smallmatrix} \right\}$ 表示第二类 Stirling number

$$\begin{aligned}
 \left\{ \begin{smallmatrix} n+1 \\ k \end{smallmatrix} \right\} &= k \left\{ \begin{smallmatrix} n \\ k \end{smallmatrix} \right\} + \left\{ \begin{smallmatrix} n \\ k-1 \end{smallmatrix} \right\}, k > 0 \\
 \left\{ \begin{smallmatrix} 0 \\ 0 \end{smallmatrix} \right\} &= 1, \left\{ \begin{smallmatrix} n \\ 0 \end{smallmatrix} \right\} = \left\{ \begin{smallmatrix} 0 \\ n \end{smallmatrix} \right\} = 0, n > 0 \\
 \left\{ \begin{smallmatrix} n \\ k \end{smallmatrix} \right\} &= \frac{1}{k!} \sum_{j=0}^k (-1)^{k-j} \binom{k}{j} j^n \\
 \left\{ \begin{smallmatrix} n \\ k \end{smallmatrix} \right\} &\text{ 为将 } n \text{ 个元素划分成 } k \text{ 个非空集合的方案数}
 \end{aligned}$$

Catalan number

c_n 表示长度为 $2n$ 的合法括号序的数量

$$\begin{aligned}
 c_1 &= 1, c_{n+1} = \sum_{i=1}^n c_i \times c_{n+1-i} \\
 c_n &= \frac{\binom{2n}{n}}{n+1}
 \end{aligned}$$

Bell number

B_n 表示基数为 n 的集合的划分方案数

$$\begin{aligned}
 B_i &= \begin{cases} 1 & i = 0 \\ \sum_{k=0}^n \binom{n}{k} B_k & i > 0 \end{cases} \\
 B_n &= \sum_{k=0}^n \left\{ \begin{smallmatrix} n \\ k \end{smallmatrix} \right\}
 \end{aligned}$$

五边形数定理

$p(n)$ 表示将 n 划分为若干个正整数之和的方案数

$$p(n) = \sum_{k \in \mathbb{N}^*} (-1)^{k-1} p(n - \frac{k(3k-1)}{2})$$

Bernoulli number

$$\sum_{j=0}^m \binom{m+1}{j} B_j = 0, m > 0$$

$$B_i = \begin{cases} 1 & i = 0 \\ -\frac{\sum_{j=0}^{i-1} \binom{i+1}{j} B_j}{i+1} & i > 0 \end{cases}$$

$$\sum_{k=1}^n k^m = \frac{1}{m+1} \sum_{k=0}^m \binom{m+1}{k} B_k n^{m+1-k}$$

Möbius function

$$\mu(n) = \begin{cases} 1 & n \text{ is a square-free positive integer with an even number of prime factors} \\ -1 & n \text{ is a square-free positive integer with an odd number of prime factors} \\ 0 & n \text{ has a squared prime factor} \end{cases}$$

$$\sum_{d|n} \mu(d) = \begin{cases} 1 & n = 1 \\ 0 & n > 1 \end{cases}$$

$$g(n) = \sum_{d|n} f(d) \Leftrightarrow f(n) = \sum_{d|n} \mu(d) g\left(\frac{n}{d}\right)$$

Lagrange polynomial

给定次数为 n 的多项式函数 $L(x)$ 上的 $n+1$ 个点 $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$

$$\text{则 } L(x) = \sum_{j=0}^n y_j \prod_{0 \leq m \leq n, m \neq j} \frac{x - x_m}{x_j - x_m}$$

Chapter 3

Geometry

3.1 点、直线、圆 (gy)

```
1 using number = long double;
2 const number eps = 1e-8;

3 number _sqrt(number x) {
4     return std::sqrt(std::max(x, (number) 0));
5 }
6 number _asin(number x) {
7     x = std::min(x, (number) 1), x = std::max(x, (number) -1);
8     return std::asin(x);
9 }
10 number _acos(number x) {
11     x = std::min(x, (number) 1), x = std::max(x, (number) -1);
12     return std::acos(x);
13 }

14 int sgn(number x) {
15     return (x > eps) - (x < -eps);
16 }
17 int cmp(number x, number y) {
18     return sgn(x - y);
19 }

20 struct point {
21     number x, y;
22     point() {}
23     point(number x, number y) : x(x), y(y) {}

24     number len2() const {
25         return x * x + y * y;
26     }
27     number len() const {
28         return _sqrt(len2());
29     }
30     point unit() const {
31         return point(x / len(), y / len());
32     }
33     point rotate90() const {
34         return point(-y, x);
35     }

36     friend point operator+(const point &a, const point &b) {
37         return point(a.x + b.x, a.y + b.y);
```

```

38     }
39     friend point operator-(const point &a, const point &b) {
40         return point(a.x - b.x, a.y - b.y);
41     }
42     friend point operator*(const point &a, number b) {
43         return point(a.x * b, a.y * b);
44     }
45     friend point operator/(const point &a, number b) {
46         return point(a.x / b, a.y / b);
47     }
48     friend number dot(const point &a, const point &b) {
49         return a.x * b.x + a.y * b.y;
50     }
51     friend number det(const point &a, const point &b) {
52         return a.x * b.y - a.y * b.x;
53     }
54     friend number operator==(const point &a, const point &b) {
55         return cmp(a.x, b.x) == 0 && cmp(a.y, b.y) == 0;
56     }
57 };

58 number dis2(const point &a, const point &b) {
59     return (a - b).len2();
60 }
61 number dis(const point &a, const point &b) {
62     return (a - b).len();
63 }

64 struct line {
65     point a, b;
66     line() {}
67     line(point a, point b) : a(a), b(b) {}
68     point value() const {
69         return b - a;
70     }
71 };

72 bool point_on_line(const point &p, const line &l) {
73     return sgn(det(p - l.a, p - l.b)) == 0;
74 }
75 // including endpoint
76 bool point_on_ray(const point &p, const line &l) {
77     return sgn(det(p - l.a, p - l.b)) == 0 &&
78         sgn(dot(p - l.a, l.b - l.a)) >= 0;
79 }
80 // including endpoints
81 bool point_on_seg(const point &p, const line &l) {
82     return sgn(det(p - l.a, p - l.b)) == 0 &&
83         sgn(dot(p - l.a, l.b - l.a)) >= 0 &&
84         sgn(dot(p - l.b, l.a - l.b)) >= 0;
85 }
86 bool seg_has_intersection(const line &a, const line &b) {
87     if (point_on_seg(a.a, b) || point_on_seg(a.b, b) ||
88         point_on_seg(b.a, a) || point_on_seg(b.b, a))
89         return /* including endpoints */ true;
90     return sgn(det(a.a - b.a, b.b - b.a)) * sgn(det(a.b - b.a, b.b - b.a)) < 0
91         && sgn(det(b.a - a.a, a.b - a.a)) * sgn(det(b.b - a.a, a.b - a.a)) < 0;
92 }
93 point intersect(const line &a, const line &b) {
94     number s1 = det(a.b - a.a, b.a - a.a);
95     number s2 = det(a.b - a.a, b.b - a.a);

```

```

96     return (b.a * s2 - b.b * s1) / (s2 - s1);
97 }
98 point projection(const point &p, const line &l) {
99     return l.a + (l.b - l.a) * dot(p - l.a, l.b - l.a) / (l.b - l.a).len2();
100 }
101 number dis(const point &p, const line &l) {
102     return std::abs(dot(p - l.a, l.b - l.a)) / (l.b - l.a).len();
103 }
104 point symmetry_point(const point &a, const point &o) {
105     return o + o - a;
106 }
107 point reflection(const point &p, const line &l) {
108     return symmetry_point(p, projection(p, l));
109 }
110 struct circle {
111     point o;
112     number r;
113     circle() {}
114     circle(point o, number r) : o(o), r(r) {}
115 };
116 bool intersect(const line &l, const circle &a, point &p1, point &p2) {
117     number x = dot(l.a - a.o, l.b - l.a);
118     number y = (l.b - l.a).len2();
119     number d = x * x - y * ((l.a - a.o).len2() - a.r * a.r);
120     if (sgn(d) < 0) return false;
121     point p = l.a - (l.b - l.a) * (x / y), delta = (l.b - l.a) * (_sqrt(d) / y);
122     p1 = p + delta, p2 = p - delta;
123     return true;
124 }
125 bool intersect(const circle &a, const circle &b, point &p1, point &p2) {
126     if (a.o == b.o && cmp(a.r, b.r) == 0)
127         return /* value for coincident circles */ false;
128     number s1 = (b.o - a.o).len();
129     if (cmp(s1, a.r + b.r) > 0 || cmp(s1, std::abs(a.r - b.r)) < 0)
130         return false;
131     number s2 = (a.r * a.r - b.r * b.r) / s1;
132     number aa = (s1 + s2) / 2, bb = (s1 - s2) / 2;
133     point p = (b.o - a.o) * (aa / (aa + bb)) + a.o;
134     point delta = (b.o - a.o).unit().rotate90() * _sqrt(a.r * a.r - aa * aa);
135     p1 = p + delta, p2 = p - delta;
136     return true;
137 }
138 bool tangent(const point &p0, const circle &c, point &p1, point &p2) {
139     number x = (p0 - c.o).len2();
140     number d = x - c.r * c.r;
141     if (sgn(d) < 0) return false;
142     if (sgn(d) == 0)
143         return /* value for point_on_line */ false;
144     point p = (p0 - c.o) * (c.r * c.r / x);
145     point delta = ((p0 - c.o) * (-c.r * _sqrt(d) / x)).rotate90();
146     p1 = c.o + p + delta;
147     p2 = c.o + p - delta;
148     return true;
149 }
150 bool ex_tangent(const circle &a, const circle &b, line &l1, line &l2) {
151     if (cmp(std::abs(a.r - b.r), (b.o - a.o).len()) == 0) {
152         point p1, p2;
153         intersect(a, b, p1, p2);
154         l1 = l2 = line(p1, p1 + (a.o - p1).rotate90());

```

```

155     return true;
156 } else if (cmp(a.r, b.r) == 0) {
157     point dir = b.o - a.o;
158     dir = (dir * (a.r / dir.len())).rotate90();
159     l1 = line(a.o + dir, b.o + dir);
160     l2 = line(a.o - dir, b.o - dir);
161     return true;
162 } else {
163     point p = (b.o * a.r - a.o * b.r) / (a.r - b.r);
164     point p1, p2, q1, q2;
165     if (tangent(p, a, p1, p2) && tangent(p, b, q1, q2)) {
166         l1 = line(p1, q1);
167         l2 = line(p2, q2);
168         return true;
169     } else {
170         return false;
171     }
172 }
173 }
174 bool in_tangent(const circle &a, const circle &b, line &l1, line &l2) {
175     if (cmp(a.r + b.r, (b.o - a.o).len()) == 0) {
176         point p1, p2;
177         intersect(a, b, p1, p2);
178         l1 = l2 = line(p1, p1 + (a.o - p1).rotate90());
179         return true;
180     } else {
181         point p = (b.o * a.r + a.o * b.r) / (a.r + b.r);
182         point p1, p2, q1, q2;
183         if (tangent(p, a, p1, p2) && tangent(p, b, q1, q2)) {
184             l1 = line(p1, q1);
185             l2 = line(p2, q2);
186             return true;
187         } else {
188             return false;
189         }
190     }
191 }

```

3.2 平面最近点对 (Grimoire)

```

1 bool byY(P a, P b) { return a.y < b.y; }
2 LL solve(P *p, int l, int r) {
3     LL d = 1LL << 62;
4     if (l == r)
5         return d;
6     if (l + 1 == r)
7         return dis2(p[l], p[r]);
8     int mid = (l + r) >> 1;
9     d = min(solve(l, mid), d);
10    d = min(solve(mid + 1, r), d);
11    vector<P> tmp;
12    for (int i = l; i <= r; i++)
13        if (sqr(p[mid].x - p[i].x) <= d)
14            tmp.push_back(p[i]);
15    sort(tmp.begin(), tmp.end(), byY);
16    for (int i = 0; i < tmp.size(); i++)
17        for (int j = i + 1; j < tmp.size() && j - i < 10; j++)
18            d = min(d, dis2(tmp[i], tmp[j]));
19    return d;

```

20 }

3.3 凸包游戏 (Grimoire)

给定凸包, $O(n \log n)$ 完成询问:

- 点在凸包内
- 凸包外的点到凸包的两个切点
- 向量关于凸包的切点
- 直线与凸包的交点

传入凸包要求 1 号点为 $Pair(x, y)$ 最小的

```

1  const int INF = 1000000000;
2  struct Convex
3  {
4      int n;
5      vector<Point> a, upper, lower;
6      Convex(vector<Point> _a) : a(_a) {
7          n = a.size();
8          int ptr = 0;
9          for(int i = 1; i < n; ++ i) if (a[ptr] < a[i]) ptr = i;
10         for(int i = 0; i <= ptr; ++ i) lower.push_back(a[i]);
11         for(int i = ptr; i < n; ++ i) upper.push_back(a[i]);
12         upper.push_back(a[0]);
13     }
14     int sign(long long x) { return x < 0 ? -1 : x > 0; }
15     pair<long long, int> get_tangent(vector<Point> &convex, Point vec) {
16         int l = 0, r = (int)convex.size() - 2;
17         for( ; l + 1 < r; ) {
18             int mid = (l + r) / 2;
19             if (sign((convex[mid + 1] - convex[mid]).det(vec)) > 0) r = mid;
20             else l = mid;
21         }
22         return max(make_pair(vec.det(convex[r]), r)
23             , make_pair(vec.det(convex[0]), 0));
24     }
25     void update_tangent(const Point &p, int id, int &i0, int &i1) {
26         if ((a[i0] - p).det(a[id] - p) > 0) i0 = id;
27         if ((a[i1] - p).det(a[id] - p) < 0) i1 = id;
28     }
29     void binary_search(int l, int r, Point p, int &i0, int &i1) {
30         if (l == r) return;
31         update_tangent(p, l % n, i0, i1);
32         int sl = sign((a[l % n] - p).det(a[(l + 1) % n] - p));
33         for( ; l + 1 < r; ) {
34             int mid = (l + r) / 2;
35             int smid = sign((a[mid % n] - p).det(a[(mid + 1) % n] - p));
36             if (smid == sl) l = mid;
37             else r = mid;
38         }
39         update_tangent(p, r % n, i0, i1);
40     }
41     int binary_search(Point u, Point v, int l, int r) {
42         int sl = sign((v - u).det(a[l % n] - u));
43         for( ; l + 1 < r; ) {
44             int mid = (l + r) / 2;
45             int smid = sign((v - u).det(a[mid % n] - u));

```



```

46         if (smid == s1) l = mid;
47         else r = mid;
48     }
49     return l % n;
50 }
51 // 判定点是否在凸包内, 在边界返回 true
52 bool contain(Point p) {
53     if (p.x < lower[0].x || p.x > lower.back().x) return false;
54     int id = lower_bound(lower.begin(), lower.end()
55         , Point(p.x, -INF)) - lower.begin();
56     if (lower[id].x == p.x) {
57         if (lower[id].y > p.y) return false;
58     } else if ((lower[id - 1] - p).det(lower[id] - p) < 0) return false;
59     id = lower_bound(upper.begin(), upper.end(), Point(p.x, INF)
60         , greater<Point>()) - upper.begin();
61     if (upper[id].x == p.x) {
62         if (upper[id].y < p.y) return false;
63     } else if ((upper[id - 1] - p).det(upper[id] - p) < 0) return false;
64     return true;
65 }
66 // 求点 p 关于凸包的两个切点, 如果在凸包外则有序返回编号
67 // 共线的多个切点返回任意一个, 否则返回 false
68 bool get_tangent(Point p, int &i0, int &i1) {
69     if (contain(p)) return false;
70     i0 = i1 = 0;
71     int id = lower_bound(lower.begin(), lower.end(), p) - lower.begin();
72     binary_search(0, id, p, i0, i1);
73     binary_search(id, (int)lower.size(), p, i0, i1);
74     id = lower_bound(upper.begin(), upper.end(), p
75         , greater<Point>()) - upper.begin();
76     binary_search((int)lower.size() - 1, (int)lower.size() - 1 + id, p, i0, i1);
77     binary_search((int)lower.size() - 1 + id
78         , (int)lower.size() - 1 + (int)upper.size(), p, i0, i1);
79     return true;
80 }
81 // 求凸包上和向量 vec 叉积最大的点, 返回编号, 共线的多个切点返回任意一个
82 int get_tangent(Point vec) {
83     pair<long long, int> ret = get_tangent(upper, vec);
84     ret.second = (ret.second + (int)lower.size() - 1) % n;
85     ret = max(ret, get_tangent(lower, vec));
86     return ret.second;
87 }
88 // 求凸包和直线 u,v 的交点, 如果无严格相交返回 false.
89 // 如果有则是和 (i, next(i)) 的交点, 两个点无序, 交在点上不确定返回前后两条线段其中之一
90 bool get_intersection(Point u, Point v, int &i0, int &i1) {
91     int p0 = get_tangent(u - v), p1 = get_tangent(v - u);
92     if (sign((v - u).det(a[p0] - u)) * sign((v - u).det(a[p1] - u)) < 0) {
93         if (p0 > p1) swap(p0, p1);
94         i0 = binary_search(u, v, p0, p1);
95         i1 = binary_search(u, v, p1, p0 + n);
96         return true;
97     } else {
98         return false;
99     }
100 }
101 };

```

3.4 半平面交 (Grimoire)

```

1 struct P{
2     int quad() const { return sgn(y) == 1 || (sgn(y) == 0 && sgn(x) >= 0); }
3 };
4 struct L{
5     bool onLeft(const P &p) const { return sgn((b - a)*(p - a)) > 0; }
6     L push() const { // push out eps
7         const double eps = 1e-10;
8         P delta = (b - a).turn90().norm() * eps;
9         return L(a - delta, b - delta);
10    }
11 };
12 bool sameDir(const L &l0, const L &l1) {
13     return parallel(l0, l1) && sgn((l0.b - l0.a)^(l1.b - l1.a)) == 1;
14 }
15 bool operator < (const P &a, const P &b) {
16     if (a.quad() != b.quad())
17         return a.quad() < b.quad();
18     else
19         return sgn((a*b)) > 0;
20 }
21 bool operator < (const L &l0, const L &l1) {
22     if (sameDir(l0, l1))
23         return l1.onLeft(l0.a);
24     else
25         return (l0.b - l0.a) < (l1.b - l1.a);
26 }
27 bool check(const L &u, const L &v, const L &w) {
28     return w.onLeft(intersect(u, v));
29 }
30 vector<P> intersection(vector<L> &l) {
31     sort(l.begin(), l.end());
32     deque<L> q;
33     for (int i = 0; i < (int)l.size(); ++i) {
34         if (i && sameDir(l[i], l[i - 1])) {
35             continue;
36         }
37         while (q.size() > 1
38             && !check(q[q.size() - 2], q[q.size() - 1], l[i]))
39             q.pop_back();
40         while (q.size() > 1
41             && !check(q[1], q[0], l[i]))
42             q.pop_front();
43         q.push_back(l[i]);
44     }
45     while (q.size() > 2
46         && !check(q[q.size() - 2], q[q.size() - 1], q[0]))
47         q.pop_back();
48     while (q.size() > 2
49         && !check(q[1], q[0], q[q.size() - 1]))
50         q.pop_front();
51     vector<P> ret;
52     for (int i = 0; i < (int)q.size(); ++i)
53         ret.push_back(intersect(q[i], q[(i + 1) % q.size()]));
54     return ret;
55 }

```

3.5 点在多边形内 (Grimoire)

```

1 bool inPoly(P p,vector<P>poly){
2     int cnt=0;
3     for(int i=0;i<poly.size();i++){
4         P a=poly[i],b=poly[(i+1)%poly.size()];
5         if(onSeg(p,L(a,b)))
6             return false;
7         int x=sgn(det(a,p,b));
8         int y=sgn(a.y-p.y);
9         int z=sgn(b.y-p.y);
10        cnt+=(x>0&&y<=0&&z>0);
11        cnt-=(x<0&&z<=0&&y>0);
12    }
13    return cnt;
14 }

```

3.6 最小圆覆盖 (Grimoire)

```

1 struct line{
2     point p,v;
3 };
4 point Rev(point v){return point(-v.y,v.x);}
5 point operator*(line A,line B){
6     point u=B.p-A.p;
7     double t=(B.v*u)/(B.v*A.v);
8     return A.p+A.v*t;
9 }
10 point get(point a,point b){
11     return (a+b)/2;
12 }
13 point get(point a,point b,point c){
14     if(a==b)return get(a,c);
15     if(a==c)return get(a,b);
16     if(b==c)return get(a,b);
17     line ABO=(line){(a+b)/2,Rev(a-b)};
18     line BCO=(line){(c+b)/2,Rev(b-c)};
19     return ABO*BCO;
20 }
21 int main(){
22     scanf("%d",&n);
23     for(int i=1;i<=n;i++)scanf("%lf%lf",&p[i].x,&p[i].y);
24     random_shuffle(p+1,p+1+n);
25     O=p[1];r=0;
26     for(int i=2;i<=n;i++){
27         if(dis(p[i],O)<r+1e-6)continue;
28         O=get(p[1],p[i]);r=dis(O,p[i]);
29         for(int j=1;j<i;j++){
30             if(dis(p[j],O)<r+1e-6)continue;
31             O=get(p[i],p[j]);r=dis(O,p[i]);
32             for(int k=1;k<j;k++){
33                 if(dis(p[k],O)<r+1e-6)continue;
34                 O=get(p[i],p[j],p[k]);r=dis(O,p[i]);
35             }
36         }
37     }printf("%.21f %.21f %.21f\n",O.x,O.y,r);
38     return 0;
39 }

```

3.7 最小球覆盖 (Grimoire)

```

1  bool equal(const double & x, const double & y) {
2      return x + eps > y and y + eps > x;
3  }
4  double operator % (const Point & a, const Point & b) {
5      return a.x * b.x + a.y * b.y + a.z * b.z;
6  }
7  Point operator * (const Point & a, const Point & b) {
8      return Point(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);
9  }
10 struct Circle {
11     double r; Point o;
12 };
13 struct Plane {
14     Point nor;
15     double m;
16     Plane(const Point & nor, const Point & a) : nor(nor){
17         m = nor % a;
18     }
19 };
20 Point intersect(const Plane & a, const Plane & b, const Plane & c) {
21     Point c1(a.nor.x, b.nor.x, c.nor.x), c2(a.nor.y, b.nor.y, c.nor.y), c3(a.nor.z, b.nor.z,
22         ↪ c.nor.z), c4(a.m, b.m, c.m);
23     return 1 / ((c1 * c2) % c3) * Point((c4 * c2) % c3, (c1 * c4) % c3, (c1 * c2) % c4);
24 }
25 bool in(const Point & a, const Circle & b) {
26     return sign((a - b.o).len() - b.r) <= 0;
27 }
28 bool operator < (const Point & a, const Point & b) {
29     if(!equal(a.x, b.x)) {
30         return a.x < b.x;
31     }
32     if(!equal(a.y, b.y)) {
33         return a.y < b.y;
34     }
35     if(!equal(a.z, b.z)) {
36         return a.z < b.z;
37     }
38     return false;
39 }
40 bool operator == (const Point & a, const Point & b) {
41     return equal(a.x, b.x) and equal(a.y, b.y) and equal(a.z, b.z);
42 }
43 vector<Point> vec;
44 Circle calc() {
45     if(vec.empty()) {
46         return Circle(Point(0, 0, 0), 0);
47     }else if(1 == (int)vec.size()) {
48         return Circle(vec[0], 0);
49     }else if(2 == (int)vec.size()) {
50         return Circle(0.5 * (vec[0] + vec[1]), 0.5 * (vec[0] - vec[1]).len());
51     }else if(3 == (int)vec.size()) {
52         double r((vec[0] - vec[1]).len() * (vec[1] - vec[2]).len() * (vec[2] - vec[0]).len() / 2 /
53             ↪ fabs(((vec[0] - vec[2]) * (vec[1] - vec[2])).len()));
54         return Circle(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])),
55             Plane(vec[2] - vec[1], 0.5 * (vec[2] + vec[1])),
56             Plane((vec[1] - vec[0]) * (vec[2] - vec[0]), vec[0])), r);
57     }else {
58         Point o(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])),

```

```

57         Plane(vec[2] - vec[0], 0.5 * (vec[2] + vec[0])),
58         Plane(vec[3] - vec[0], 0.5 * (vec[3] + vec[0]))));
59     return Circle(o, (o - vec[0]).len());
60 }
61 }
62 Circle miniBall(int n) {
63     Circle res(calc());
64     for(int i(0); i < n; i++) {
65         if(!in(a[i], res)) {
66             vec.push_back(a[i]);
67             res = miniBall(i);
68             vec.pop_back();
69             if(i) {
70                 Point tmp(a[i]);
71                 memmove(a + 1, a, sizeof(Point) * i);
72                 a[0] = tmp;
73             }
74         }
75     }
76     return res;
77 }
78 int main() {
79     int n;
80     sort(a, a + n);
81     n = unique(a, a + n) - a;
82     vec.clear();
83     printf("%.10f\n", miniBall(n).r);
84 }

```

3.8 圓并 (Grimoire)

```

1  double ans[2001];
2  struct Point {
3      double x, y;
4      Point(){}
5      Point(const double & x, const double & y) : x(x), y(y) {}
6      void scan() {scanf("%lf%lf", &x, &y);}
7      double sqrlen() {return sqr(x) + sqr(y);}
8      double len() {return sqrt(sqrlen());}
9      Point rev() {return Point(y, -x);}
10     void print() {printf("%f %f\n", x, y);}
11     Point zoom(const double & d) {double lambda = d / len(); return Point(lambda * x, lambda * y);}
12 } dvd, a[2001];
13 Point centre[2001];
14 double atan2(const Point & x) {
15     return atan2(x.y, x.x);
16 }
17 Point operator - (const Point & a, const Point & b) {
18     return Point(a.x - b.x, a.y - b.y);
19 }
20 Point operator + (const Point & a, const Point & b) {
21     return Point(a.x + b.x, a.y + b.y);
22 }
23 double operator * (const Point & a, const Point & b) {
24     return a.x * b.y - a.y * b.x;
25 }
26 Point operator * (const double & a, const Point & b) {
27     return Point(a * b.x, a * b.y);
28 }

```

```

29 double operator % (const Point & a, const Point & b) {
30     return a.x * b.x + a.y * b.y;
31 }
32 struct circle {
33     double r; Point o;
34     circle() {}
35     void scan() {
36         o.scan();
37         scanf("%lf", &r);
38     }
39 } cir[2001];
40 struct arc {
41     double theta;
42     int delta;
43     Point p;
44     arc() {};
45     arc(const double & theta, const Point & p, int d) : theta(theta), p(p), delta(d) {}
46 } vec[4444];
47 int nV;
48 inline bool operator < (const arc & a, const arc & b) {
49     return a.theta + eps < b.theta;
50 }
51 int cnt;
52 inline void psh(const double t1, const Point p1, const double t2, const Point p2) {
53     if(t2 + eps < t1)
54         cnt++;
55     vec[nV++] = arc(t1, p1, 1);
56     vec[nV++] = arc(t2, p2, -1);
57 }
58 inline double cub(const double & x) {
59     return x * x * x;
60 }
61 inline void combine(int d, const double & area, const Point & o) {
62     if(sign(area) == 0) return;
63     centre[d] = 1 / (ans[d] + area) * (ans[d] * centre[d] + area * o);
64     ans[d] += area;
65 }
66 bool equal(const double & x, const double & y) {
67     return x + eps > y and y + eps > x;
68 }
69 bool equal(const Point & a, const Point & b) {
70     return equal(a.x, b.x) and equal(a.y, b.y);
71 }
72 bool equal(const circle & a, const circle & b) {
73     return equal(a.o, b.o) and equal(a.r, b.r);
74 }
75 bool f[2001];
76 int main() {
77     int n, m, index;
78     while(EOF != scanf("%d%d%d", &m, &n, &index)) {
79         index--;
80         for(int i(0); i < m; i++) {
81             a[i].scan();
82         }
83         for(int i(0); i < n; i++) {
84             cir[i].scan(); //n 个圆
85         }
86         for(int i(0); i < n; i++) { //这一段在去重圆 能加速 删掉不会错
87             f[i] = true;
88             for(int j(0); j < n; j++) if(i != j) {

```

```

89         if(equal(cir[i], cir[j]) and i < j or !equal(cir[i], cir[j]) and cir[i].r <
          ↪ cir[j].r + eps and (cir[i].o - cir[j].o).sqrln() < sqr(cir[i].r - cir[j].r) +
          ↪ eps) {
90             f[i] = false;
91             break;
92         }
93     }
94 }
95 int n1(0);
96 for(int i(0); i < n; i++)
97     if(f[i])
98         cir[n1++] = cir[i];
99 n = n1; //去重圆结束
100 fill(ans, ans + n + 1, 0); //ans[i] 表示被圆覆盖至少 i 次的面积
101 fill(centre, centre + n + 1, Point(0, 0)); //centre[i] 表示上面 ans[i] 部分的重心
102 for(int i(0); i < m; i++)
103     combine(0, a[i] * a[(i + 1) % m] * 0.5, 1. / 3 * (a[i] + a[(i + 1) % m]));
104 for(int i(0); i < n; i++) {
105     dvd = cir[i].o - Point(cir[i].r, 0);
106     nV = 0;
107     vec[nV++] = arc(-pi, dvd, 1);
108     cnt = 0;
109     for(int j(0); j < n; j++) if(j != i) {
110         double d = (cir[j].o - cir[i].o).sqrln();
111         if(d < sqr(cir[j].r - cir[i].r) + eps) {
112             if(cir[i].r + i * eps < cir[j].r + j * eps)
113                 psh(-pi, dvd, pi, dvd);
114             }else if(d + eps < sqr(cir[j].r + cir[i].r)) {
115                 double lambda = 0.5 * (1 + (sqr(cir[i].r) - sqr(cir[j].r)) / d);
116                 Point cp(cir[i].o + lambda * (cir[j].o - cir[i].o));
117                 Point nor((cir[j].o - cir[i].o).rev().zoom(sqrt(sqr(cir[i].r) - (cp -
                  ↪ cir[i].o).sqrln())));
118                 Point frm(cp + nor);
119                 Point to(cp - nor);
120                 psh(atan2(frm - cir[i].o), frm, atan2(to - cir[i].o), to);
121             }
122         }
123     sort(vec + 1, vec + nV);
124     vec[nV++] = arc(pi, dvd, -1);
125     for(int j = 0; j + 1 < nV; j++) {
126         cnt += vec[j].delta;
127         //if(cnt == 1) { //如果只算 ans[1] 和 centre[1], 可以加这个 if 加速.
128             double theta(vec[j + 1].theta - vec[j].theta);
129             double area(sqr(cir[i].r) * theta * 0.5);
130             combine(cnt, area, cir[i].o + 1. / area / 3 * cub(cir[i].r) * Point(sin(vec[j +
              ↪ 1].theta) - sin(vec[j].theta), cos(vec[j].theta) - cos(vec[j + 1].theta)));
131             combine(cnt, -sqr(cir[i].r) * sin(theta) * 0.5, 1. / 3 * (cir[i].o + vec[j].p +
              ↪ vec[j + 1].p));
132             combine(cnt, vec[j].p * vec[j + 1].p * 0.5, 1. / 3 * (vec[j].p + vec[j +
              ↪ 1].p));
133         } //}
134     }
135 }
136 combine(0, -ans[1], centre[1]);
137 for(int i = 0; i < m; i++) {
138     if(i != index)
139         (a[index] - Point((a[i] - a[index]) * (centre[0] - a[index]), (a[i] - a[index]) %
          ↪ (centre[0] - a[index])).zoom((a[i] - a[index]).len()))).print();
140     else
141         a[i].print();
142 }

```

```

143     }
144     return 0;
145 }

```

3.9 圆与多边形并 (Grimoire)

```

1  double form(double x){
2      while(x>=2*pi)x-=2*pi;
3      while(x<0)x+=2*pi;
4      return x;
5  }
6  double calcCir(C cir){
7      vector<double>ang;
8      ang.push_back(0);
9      ang.push_back(pi);
10     double ans=0;
11     for(int i=1;i<=n;i++){
12         if(cir==c[i])continue;
13         P p1,p2;
14         if(intersect(cir,c[i],p1,p2)){
15             ang.push_back(form(cir.ang(p1)));
16             ang.push_back(form(cir.ang(p2)));
17         }
18     }
19
20     for(int i=1;i<=m;i++){
21         vector<P>tmp;
22         tmp=intersect(poly[i],cir);
23         for(int j=0;j<tmp.size();j++){
24             ang.push_back(form(cir.ang(tmp[j])));
25         }
26     }
27     sort(ang.begin(),ang.end());
28     for(int i=0;i<ang.size();i++){
29         double t1=ang[i],t2=(i+1==ang.size()?ang[0]+2*pi:ang[i+1]);
30         P p=cir.at((t1+t2)/2);
31         int ok=1;
32         for(int j=1;j<=n;j++){
33             if(cir==c[j])continue;
34             if(inC(p,c[j],true)){
35                 ok=0;
36                 break;
37             }
38         }
39         for(int j=1;j<=m&&ok;j++){
40             if(inPoly(p,poly[j],true)){
41                 ok=0;
42                 break;
43             }
44         }
45         if(ok){
46             double r=cir.r,x0=cir.o.x,y0=cir.o.y;
47             ans+=(r*r*(t2-t1)+r*x0*(sin(t2)-sin(t1))-r*y0*(cos(t2)-cos(t1)))/2;
48         }
49     }
50     return ans;
51 }
P st;

```



```

52 bool bySt(P a,P b){
53     return dis(a,st)<dis(b,st);
54 }
55 double calcSeg(L l){
56     double ans=0;
57     vector<P>pt;
58     pt.push_back(l.a);
59     pt.push_back(l.b);
60     for(int i=1;i<=n;i++){
61         P p1,p2;
62         if(intersect(c[i],l,p1,p2)){
63             if(onSeg(p1,l))
64                 pt.push_back(p1);
65             if(onSeg(p2,l))
66                 pt.push_back(p2);
67         }
68     }
69     st=l.a;
70     sort(pt.begin(),pt.end(),bySt);
71     for(int i=0;i+1<pt.size();i++){
72         P p1=pt[i],p2=pt[i+1];
73         P p=(p1+p2)/2;
74         int ok=1;
75         for(int j=1;j<=n;j++){
76             if(sgn(dis(p,c[j].o),c[j].r)<0){
77                 ok=0;
78                 break;
79             }
80         }
81         if(ok){
82             double x1=p1.x,y1=p1.y,x2=p2.x,y2=p2.y;
83             double res=(x1*y2-x2*y1)/2;
84             ans+=res;
85         }
86     }
87     return ans;
88 }

```

3.10 三角剖分 (Grimoire)

Triangulation :: *find* 返回包含某点的三角形

Triangulation :: *add_point* 将某点加入三角剖分

某个 *Triangle* 在三角剖分中当且仅当它的 *has_children* 为 0

如果要找到三角形 *u* 的邻域, 则枚举它的所有 *u.edge[i].tri*, 该条边的两个点为 *u.p[(i+1)%3]*, *u.p[(i+2)%3]*

通过三角剖分构造 *V* 图: 连接相邻三角形外接圆圆心

注意初始化内存池和 *Triangulation* :: *LOTS*

复杂度 $O(n \log n)$

```

1  const int N = 100000 + 5, MAX_TRIS = N * 6;
2  const double eps = 1e-6, PI = acos(-1.0);
3  struct P {
4      double x,y; P():x(0),y(0){}
5      P(double x, double y):x(x),y(y){}
6      bool operator==(P const& that)const {return x==that.x&&y==that.y;}
7  };
8  inline double sqr(double x) { return x*x; }
9  double dist_sqr(P const& a, P const& b){return sqr(a.x-b.x)+sqr(a.y-b.y);}
10 bool in_circumcircle(P const& p1, P const& p2, P const& p3, P const& p4) { //p4 in C(p1,p2,p3)
11     double u11 = p1.x - p4.x, u21 = p2.x - p4.x, u31 = p3.x - p4.x;

```

```

12     double u12 = p1.y - p4.y, u22 = p2.y - p4.y, u32 = p3.y - p4.y;
13     double u13 = sqrt(p1.x) - sqrt(p4.x) + sqrt(p1.y) - sqrt(p4.y);
14     double u23 = sqrt(p2.x) - sqrt(p4.x) + sqrt(p2.y) - sqrt(p4.y);
15     double u33 = sqrt(p3.x) - sqrt(p4.x) + sqrt(p3.y) - sqrt(p4.y);
16     double det = -u13*u22*u31 + u12*u23*u31 + u13*u21*u32 - u11*u23*u32 - u12*u21*u33 +
        ↪ u11*u22*u33;
17     return det > eps;
18 }
19 double side(P const& a, P const& b, P const& p) { return (b.x-a.x)*(p.y-a.y) -
        ↪ (b.y-a.y)*(p.x-a.x);}
20 typedef int SideRef; struct Triangle; typedef Triangle* TriangleRef;
21 struct Edge {
22     TriangleRef tri; SideRef side; Edge() : tri(0), side(0) {}
23     Edge(TriangleRef tri, SideRef side) : tri(tri), side(side) {}
24 };
25 struct Triangle {
26     P p[3]; Edge edge[3]; TriangleRef children[3]; Triangle() {}
27     Triangle(P const& p0, P const& p1, P const& p2) {
28         p[0] = p0; p[1] = p1; p[2] = p2;
29         children[0] = children[1] = children[2] = 0;
30     }
31     bool has_children() const { return children[0] != 0; }
32     int num_children() const {
33         return children[0] == 0 ? 0
34             : children[1] == 0 ? 1
35             : children[2] == 0 ? 2 : 3;
36     }
37     bool contains(P const& q) const {
38         double a=side(p[0],p[1],q), b=side(p[1],p[2],q), c=side(p[2],p[0],q);
39         return a >= -eps && b >= -eps && c >= -eps;
40     }
41 } triange_pool[MAX_TRIS], *tot_triangles;
42 void set_edge(Edge a, Edge b) {
43     if (a.tri) a.tri->edge[a.side] = b;
44     if (b.tri) b.tri->edge[b.side] = a;
45 }
46 class Triangulation {
47 public:
48     Triangulation() {
49         const double LOTS = 1e6; //初始为极大三角形
50         the_root = new(tot_triangles++) Triangle(P(-LOTS,-LOTS),P(+LOTS,-LOTS),P(0,+LOTS));
51     }
52     TriangleRef find(P p) const { return find(the_root,p); }
53     void add_point(P const& p) { add_point(find(the_root,p),p); }
54 private:
55     TriangleRef the_root;
56     static TriangleRef find(TriangleRef root, P const& p) {
57         for( ; ; ) {
58             if (!root->has_children()) return root;
59             else for (int i = 0; i < 3 && root->children[i] ; ++i)
60                 if (root->children[i]->contains(p))
61                     {root = root->children[i]; break;}
62         }
63     }
64     void add_point(TriangleRef root, P const& p) {
65         TriangleRef tab,tbc,tca;
66         tab = new(tot_triangles++) Triangle(root->p[0], root->p[1], p);
67         tbc = new(tot_triangles++) Triangle(root->p[1], root->p[2], p);
68         tca = new(tot_triangles++) Triangle(root->p[2], root->p[0], p);
69         set_edge(Edge(tab,0),Edge(tbc,1)); set_edge(Edge(tbc,0),Edge(tca,1));
70         set_edge(Edge(tca,0),Edge(tab,1)); set_edge(Edge(tab,2),root->edge[2]);

```

```

71     set_edge(Edge(tbc,2),root->edge[0]); set_edge(Edge(tca,2),root->edge[1]);
72     root->children[0]=tab; root->children[1]=tbc; root->children[2]=tca;
73     flip(tab,2); flip(tbc,2); flip(tca,2);
74 }
75 void flip(TriangleRef tri, SideRef pi) {
76     TriangleRef trj = tri->edge[pi].tri; int pj = tri->edge[pi].side;
77     if(!trj || !in_circumcircle(tri->p[0],tri->p[1],tri->p[2],trj->p[pj])) return;
78     TriangleRef trk = new(tot_triangles++) Triangle(tri->p[(pi+1)%3], trj->p[pj],
79         ↪ tri->p[pi]);
80     TriangleRef trl = new(tot_triangles++) Triangle(trj->p[(pj+1)%3], tri->p[pi],
81         ↪ trj->p[pj]);
82     set_edge(Edge(trk,0), Edge(trl,0));
83     set_edge(Edge(trk,1), tri->edge[(pi+2)%3]); set_edge(Edge(trk,2), trj->edge[(pj+1)%3]);
84     set_edge(Edge(trl,1), trj->edge[(pj+2)%3]); set_edge(Edge(trl,2), tri->edge[(pi+1)%3]);
85     tri->children[0]=trk; tri->children[1]=trl; tri->children[2]=0;
86     trj->children[0]=trk; trj->children[1]=trl; trj->children[2]=0;
87     flip(trk,1); flip(trk,2); flip(trl,1); flip(trl,2);
88 }
89 };
90 int n; P ps[N];
91 void build(){
92     tot_triangles = triange_pool; cin >> n;
93     for(int i = 0; i < n; ++ i) scanf("%lf%lf",&ps[i].x,&ps[i].y);
94     random_shuffle(ps, ps + n); Triangulation tri;
95     for(int i = 0; i < n; ++ i) tri.add_point(ps[i]);
96 }

```

3.11 三维几何基础 (Grimoire)

```

1 struct P {
2     double x, y, z;
3     P(){}
4     P(double _x,double _y,double _z):x(_x),y(_y),z(_z){}
5     double len2(){
6         return (x*x+y*y+z*z);
7     }
8     double len(){
9         return sqrt(x*x+y*y+z*z);
10    }
11 };
12 bool operator==(P a,P b){
13     return sgn(a.x-b.x)==0 && sgn(a.y-b.y)==0 && sgn(a.z-b.z)==0 ;
14 }
15 bool operator<(P a,P b){
16     return sgn(a.x-b.x) ? a.x<b.x : (sgn(a.y-b.y)?a.y<b.y :a.z<b.z);
17 }
18 P operator+(P a,P b){
19     return P(a.x+b.x,a.y+b.y,a.z+b.z);
20 }
21 P operator-(P a,P b){
22     return P(a.x-b.x,a.y-b.y,a.z-b.z);
23 }
24 P operator*(P a,double b){
25     return P(a.x*b,a.y*b,a.z*b);
26 }
27 P operator/(P a,double b){
28     return P(a.x/b,a.y/b,a.z/b);
29 }
30 P operator*(const P &a, const P &b) {

```

```

31     return P(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);
32 }
33 double operator^(const P &a, const P &b) {
34     return a.x*b.x+a.y*b.y+a.z*b.z;
35 }
36 double dis(P a,P b){return (b-a).len();}
37 double dis2(P a,P b){return (b-a).len2();}
38 // 3D line intersect
39 P intersect(const P &a0, const P &b0, const P &a1, const P &b1) {
40     double t = ((a0.x - a1.x) * (a1.y - b1.y) - (a0.y - a1.y) * (a1.x - b1.x)) / ((a0.x - b0.x) *
41         ↪ (a1.y - b1.y) - (a0.y - b0.y) * (a1.x - b1.x));
42     return a0 + (b0 - a0) * t;
43 }
44 // area-line intersect
45 P intersect(const P &a, const P &b, const P &c, const P &l0, const P &l1) {
46     P p = (b-a)*(c-a); // 平面法向量
47     double t = (p^(a-l0)) / (p^(l1-l0));
48     return l0 + (l1 - l0) * t;
49 }

```

3.12 三维凸包 (Grimoire)

```

1  int mark[1005][1005],n, cnt;;
2  double mix(const P &a, const P &b, const P &c) {
3      return a^(b*c);
4  }
5  double area(int a, int b, int c) {
6      return ((info[b] - info[a])*(info[c] - info[a])).len();
7  }
8  double volume(int a, int b, int c, int d) {
9      return mix(info[b] - info[a], info[c] - info[a], info[d] - info[a]);
10 }
11 struct Face {
12     int a, b, c; Face() {}
13     Face(int a, int b, int c): a(a), b(b), c(c) {}
14     int &operator [] (int k) {
15         if (k == 0) return a; if (k == 1) return b; return c;
16     }
17 };
18 vector <Face> face;
19 inline void insert(int a, int b, int c) {
20     face.push_back(Face(a, b, c));
21 }
22 void add(int v) {
23     vector <Face> tmp; int a, b, c; cnt++;
24     for (int i = 0; i < SIZE(face); i++) {
25         a = face[i][0]; b = face[i][1]; c = face[i][2];
26         if (sgn(volume(v, a, b, c)) < 0)
27             mark[a][b] = mark[b][a] = mark[b][c] = mark[c][b] = mark[c][a] = mark[a][c] = cnt;
28         else tmp.push_back(face[i]);
29     } face = tmp;
30     for (int i = 0; i < SIZE(tmp); i++) {
31         a = face[i][0]; b = face[i][1]; c = face[i][2];
32         if (mark[a][b] == cnt) insert(b, a, v);
33         if (mark[b][c] == cnt) insert(c, b, v);
34         if (mark[c][a] == cnt) insert(a, c, v);

```

```

35     }
36 }
37 int Find() {
38     for (int i = 2; i < n; i++) {
39         P ndir = (info[0] - info[i])*(info[1] - info[i]);
40         if (ndir == P()) continue; swap(info[i], info[2]);
41         for (int j = i + 1; j < n; j++) if (sgn(volume(0, 1, 2, j)) != 0) {
42             swap(info[j], info[3]); insert(0, 1, 2); insert(0, 2, 1); return 1;
43         }
44     }
45     return 0;
46 }
47 //find the weight center
48 double calcDist(const P &p, int a, int b, int c) {
49     return fabs(mix(info[a] - p, info[b] - p, info[c] - p) / area(a, b, c));
50 }
51 //compute the minimal distance of center of any faces
52 P findCenter() { //compute center of mass
53     double totalWeight = 0;
54     P center(.0, .0, .0);
55     P first = info[face[0][0]];
56     for (int i = 0; i < SIZE(face); ++i) {
57         P p = (info[face[i][0]]+info[face[i][1]]+info[face[i][2]]+first)*.25;
58         double weight = mix(info[face[i][0]] - first, info[face[i][1]] - first, info[face[i][2]] -
59             ↪ first);
60         totalWeight += weight; center = center + p * weight;
61     }
62     center = center / totalWeight;
63     return center;
64 }
65 double minDis(P p) {
66     double res = 1e100; //compute distance
67     for (int i = 0; i < SIZE(face); ++i)
68         res = min(res, calcDist(p, face[i][0], face[i][1], face[i][2]));
69     return res;
70 }
71 void findConvex(P *info, int n) {
72     sort(info, info + n); n = unique(info, info + n) - info;
73     face.clear(); random_shuffle(info, info + n);
74     if (!Find()) return abort();
75     memset(mark, 0, sizeof(mark)); cnt = 0;
76     for (int i = 3; i < n; i++) add(i);
77 }

```

3.13 三维绕轴旋转 (Grimoire)

右手大拇指指向 *axis* 方向，四指弯曲方向旋转 *w* 弧度

```

1 P rotate(const P& s, const P& axis, double w) {
2     double x = axis.x, y = axis.y, z = axis.z;
3     double s1 = x * x + y * y + z * z, ss1 = msqrt(s1),
4         cosw = cos(w), sinw = sin(w);
5     double a[4][4];
6     memset(a, 0, sizeof a);
7     a[3][3] = 1;
8     a[0][0] = ((y * y + z * z) * cosw + x * x) / s1;
9     a[0][1] = x * y * (1 - cosw) / s1 + z * sinw / ss1;
10    a[0][2] = x * z * (1 - cosw) / s1 - y * sinw / ss1;
11    a[1][0] = x * y * (1 - cosw) / s1 - z * sinw / ss1;

```

```

12  a[1][1] = ((x * x + z * z) * cosw + y * y) / s1;
13  a[1][2] = y * z * (1 - cosw) / s1 + x * sinw / ss1;
14  a[2][0] = x * z * (1 - cosw) / s1 + y * sinw / ss1;
15  a[2][1] = y * z * (1 - cosw) / s1 - x * sinw / ss1;
16  a[2][2] = ((x * x + y * y) * cos(w) + z * z) / s1;
17  double ans[4] = {0, 0, 0, 0}, c[4] = {s.x, s.y, s.z, 1};
18  for (int i = 0; i < 4; ++ i)
19      for (int j = 0; j < 4; ++ j)
20          ans[i] += a[j][i] * c[j];
21  return P(ans[0], ans[1], ans[2]);
22 }

```

3.14 几何知识 (gy)

Pick theorem

顶点为整点的简单多边形，其面积 A ，内部格点数 i ，边上格点数 b 满足：

$$A = i + \frac{b}{2} - 1$$

欧拉示性数

- 三维凸包的顶点个数 V ，边数 E ，面数 F 满足：
 $V - E + F = 2$
- 平面图的顶点个数 V ，边数 E ，平面被划分的区域数 F ，组成图形的连通部分的数目 C 满足：
 $V - E + F = C + 1$

几何公式

• 三角形

$$\text{半周长 } p = \frac{a+b+c}{2}$$

$$\text{面积 } S = \frac{1}{2}aH_a = \frac{1}{2}ab \cdot \sin C = \sqrt{p(p-a)(p-b)(p-c)} = pr = \frac{abc}{4R}$$

$$\text{中线长 } M_a = \frac{1}{2}\sqrt{2(b^2 + c^2) - a^2} = \frac{1}{2}\sqrt{b^2 + c^2 + 2bc \cdot \cos A}$$

$$\text{角平分线长 } T_a = \frac{\sqrt{bc((b+c)^2 - a^2)}}{b+c} = \frac{2bc}{b+c} \cos \frac{A}{2}$$

$$\text{高 } H_a = b \sin C = \sqrt{b^2 - \left(\frac{a^2 + b^2 - c^2}{2a}\right)^2}$$

$$\text{内切圆半径 } r = \frac{S}{p} = 4R \sin \frac{A}{2} \sin \frac{B}{2} \sin \frac{C}{2} = \sqrt{\frac{(p-a)(p-b)(p-c)}{p}} = p \tan \frac{A}{2} \tan \frac{B}{2} \tan \frac{C}{2}$$

$$\text{外接圆半径 } R = \frac{abc}{4S} = \frac{a}{2 \sin A}$$

$$\text{旁切圆半径 } r_A = \frac{2S}{-a+b+c}$$

$$\text{重心 } \left(\frac{x_1+x_2+x_3}{3}, \frac{y_1+y_2+y_3}{3} \right)$$

$$\text{外心 } \left(\frac{\begin{vmatrix} x_1^2+y_1^2 & y_1 & 1 \\ x_2^2+y_2^2 & y_2 & 1 \\ x_3^2+y_3^2 & y_3 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}, \frac{\begin{vmatrix} x_1 & x_1^2+y_1^2 & 1 \\ x_2 & x_2^2+y_2^2 & 1 \\ x_3 & x_3^2+y_3^2 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}} \right)$$

$$\text{内心 } \left(\frac{ax_1+bx_2+cx_3}{a+b+c}, \frac{ay_1+by_2+cy_3}{a+b+c} \right)$$

$$\text{垂心 } \left(\frac{\begin{vmatrix} x_2x_3+y_2y_3 & 1 & y_1 \\ x_3x_1+y_3y_1 & 1 & y_2 \\ x_1x_2+y_1y_2 & 1 & y_3 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}, \frac{\begin{vmatrix} x_2x_3+y_2y_3 & x_1 & 1 \\ x_3x_1+y_3y_1 & x_2 & 1 \\ x_1x_2+y_1y_2 & x_3 & 1 \end{vmatrix}}{\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}} \right)$$

$$\text{旁心 } \left(\frac{-ax_1+bx_2+cx_3}{-a+b+c}, \frac{-ay_1+by_2+cy_3}{-a+b+c} \right)$$

- **圆**

弧长 $l = rA$

弦长 $a = 2\sqrt{2hr - h^2} = 2r \cdot \sin \frac{A}{2}$

弓形高 $h = r - \sqrt{r^2 - \frac{a^2}{4}} = r(1 - \cos \frac{A}{2})$

扇形面积 $S_1 = \frac{1}{2}lr = \frac{1}{2}Ar^2$

弓形面积 $S_2 = \frac{1}{2}r^2(A - \sin A)$

- **Circles of Apollonius**

已知三个两两相切的圆, 半径为 r_1, r_2, r_3

与它们外切的圆半径为 $\left| \frac{r_1 r_2 r_3}{r_1 r_2 + r_2 r_3 + r_3 r_1 - 2\sqrt{r_1 r_2 r_3 (r_1 + r_2 + r_3)}} \right|$

与它们内切的圆半径为 $\frac{r_1 r_2 r_3}{r_1 r_2 + r_2 r_3 + r_3 r_1 + 2\sqrt{r_1 r_2 r_3 (r_1 + r_2 + r_3)}}$

- **棱台**

体积 $V = \frac{1}{3}h(A_1 + A_2 + \sqrt{A_1 A_2})$

正棱台侧面积 $S = \frac{1}{2}(p_1 + p_2)l$, l 为侧高

- **球**

体积 $V = \frac{4}{3}\pi r^3$

表面积 $S = 4\pi r^2$

- **球台**

侧面积 $S = 2\pi rh$

体积 $V = \frac{1}{6}\pi h(3(r_1^2 + r_2^2) + h_h)$

- **球扇形**

球面面积 $S = 2\pi rh$

体积 $V = \frac{2}{3}\pi r^2 h = \frac{2}{3}\pi r^3 h(1 - \cos \varphi)$

- **球面三角形**

考虑单位球上的球面三角形, a, b, c 表示三边长 (弧所对球心角), A, B, C 表示三角大小 (切线夹角)

余弦定理 $\cos a = \cos b \cdot \cos c + \sin b \cdot \sin c \cdot \cos A$

正弦定理 $\frac{\sin A}{\sin a} = \frac{\sin B}{\sin b} = \frac{\sin C}{\sin c}$

球面面积 $S = A + B + C - \pi$

- **四面体**

体积 $V = \frac{1}{6} |\vec{AB} \cdot (\vec{AC} \times \vec{AD})|$

Chapter 4

String

4.1 KMP (ct)

KMP

```
1 int main()
2 {
3     for (int i = 2, j = 0; i <= n; ++i)
4     {
5         for ( ; j && s[j + 1] != s[i]; j = fail[j]) ;
6         s[i] == s[j + 1] ? ++j : 0;
7         fail[i] = j;
8     }
9     return 0;
10 }
```

exKMP

$extend_i$ 表示 T 与 $S_{i,n}$ 的最长公共前缀

```
1 int next[maxn], extend[maxn], fail[maxn];
2 void getnext(R char *s, R int len)
3 {
4     fail[1] = 0;
5     R int p = 0;
6     memset(next, 0, (len + 2) << 2);
7     for (R int i = 2; i <= len; ++i)
8     {
9         while (p && s[p + 1] != s[i]) p = fail[p];
10        s[p + 1] == s[i] ? ++p : 0;
11        fail[i] = p;
12        p ? cmax(next[i - p + 1], p) : 0;
13    }
14 }
15 void getextend(R char *s, R int lens, R char *t, R int lent)
16 {
17     getnext(t, lent);
18     R int a = 1, p = 0;
19
20     for (R int i = 1; i <= lens; ++i)
21     {
22         if (i + next[i - a + 1] - 1 >= p)
23         {
24             cmax(p, i - 1);
25         }
26     }
27 }
```



```

24         while (p < lens && p - i + 1 < lent && s[p + 1] == t[p - i + 2]) ++p;
25         a = i;
26         extend[i] = p - i + 1;
27     }
28     else extend[i] = next[i - a + 1];
29 }
30 }

```

4.2 AC 自动机 (ct)

AC 自动机

```

1 struct SAM {
2     SAM *next[26], *fa;
3     int val;
4 } mem[maxn], *last = mem, *tot = mem;
5 void extend(int c)
6 {
7     R SAM *p = last, *np;
8     last = np = ++tot; np -> val = p -> val + 1;
9     for (; p && !p -> next[c]; p = p -> fa) p -> next[c] = np;
10    if (!p) np -> fa = rt[id];
11    else
12    {
13        SAM *q = p -> next[c];
14        if (q -> val == p -> val + 1) np -> fa = q;
15        else
16        {
17            SAM *nq = ++tot;
18            memcpy(nq -> next, q -> next, sizeof nq -> next);
19            nq -> val = p -> val + 1;
20            nq -> fa = q -> fa;
21            q -> fa = np -> fa = nq;
22            for (; p && p -> next[c] == q; p = p -> fa) p -> next[c] = nq;
23        }
24    }
25 }

```

广义 AC 自动机

```

1 struct sam {
2     sam *next[26], *fa;
3     int val;
4 } mem[maxn << 1], *tot = mem;
5 inline sam *extend(R sam *p, R int c)
6 {
7     if (p -> next[c])
8     {
9         R sam *q = p -> next[c];
10        if (q -> val == p -> val + 1)
11            return q;
12        else
13        {
14            R sam *nq = ++tot;
15            memcpy(nq -> next, q -> next, sizeof nq -> next);
16            nq -> val = p -> val + 1;
17            nq -> fa = q -> fa;
18            q -> fa = nq;

```

```

19         for ( ; p && p -> next[c] == q; p = p -> fa)
20             p -> next[c] = nq;
21         return nq;
22     }
23 }
24 R sam *np = ++tot;
25 np -> val = p -> val + 1;
26 for ( ; p && !p -> next[c]; p = p -> fa) p -> next[c] = np;
27 if (!p)
28     np -> fa = mem;
29 else
30 {
31     R sam *q = p -> next[c];
32     if (q -> val == p -> val + 1)
33         np -> fa = q;
34     else
35     {
36         R sam *nq = ++tot;
37         memcpy(nq -> next, q -> next, sizeof nq -> next);
38         nq -> val = p -> val + 1;
39         nq -> fa = q -> fa;
40         q -> fa = np -> fa = nq;
41         for ( ; p && p -> next[c] == q; p = p -> fa)
42             p -> next[c] = nq;
43     }
44 }
45 return np;
46 }

```

4.3 后缀数组 (ct)

```

1 char s[maxn];
2 int sa[maxn], rank[maxn], wa[maxn], wb[maxn], cnt[maxn], height[maxn];
3 inline void build(int n, int m)
4 {
5     int *x = wa, *y = wb, *t;
6     for (int i = 1; i <= n; ++i) cnt[x[i] = s[i] - 'a' + 1]++;
7     for (int i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
8     for (int i = n; i; --i) sa[cnt[x[i]]--] = i;
9
10    for (int j = 1; j < n || (j == 1 && m < n); j <= 1, t = x, x = y, y = t)
11    {
12        memset(cnt + 1, 0, m << 2);
13        int p = 0;
14        for (int i = n - j + 1; i <= n; ++i) y[++p] = i;
15        for (int i = 1; i <= n; ++i)
16        {
17            ++cnt[x[i]];
18            sa[i] > j ? y[++p] = sa[i] - j : 0;
19        }
20        for (int i = 1; i <= m; ++i) cnt[i] += cnt[i - 1];
21        for (int i = n; i; --i) sa[cnt[x[y[i]]]--] = y[i];
22        m = 0;
23        for (int i = 1; i <= n; ++i)
24            y[sa[i]] = (i == 1 || x[sa[i]] != x[sa[i - 1]] || x[sa[i - 1] + j] != x[sa[i] + j]) ?
25                ++m : m;
26    }
27    for (int i = 1; i <= n; ++i) rank[sa[i]] = i;
28    for (int i = 1, j, k = 0; i <= n; height[rank[i++]] = k)

```

```

27     for (k ? --k : 0, j = sa[rank[i] - 1]; s[i + k] == s[j + k]; ++k);
28 }

```

4.4 后缀自动机

4.5 Manacher (ct)

```

1  char str[maxn];
2  int p1[maxn], p2[maxn], n;
3  void manacher1()
4  {
5      int mx = 0, id;
6      for(int i = 1; i <= n; ++i)
7      {
8          if (mx >= i) p1[i] = dmin(mx - i, p1[(id << 1) - i]);
9          else p1[i] = 1;
10         for (; str[i + p1[i]] == str[i - p1[i]]; ++p1[i]);
11         if (p1[i] + i - 1 > mx) id = i, mx = p1[i] + i - 1;
12     }
13 }
14 void manacher2()
15 {
16     int mx = 0, id;
17     for(int i = 1; i <= n; i++)
18     {
19         if (mx >= i) p2[i] = dmin(mx - i, p2[(id << 1) - i]);
20         else p2[i] = 0;
21         for (; str[i + p2[i] + 1] == str[i - p2[i]]; ++p2[i]);
22         if (p2[i] + i > mx) id = i, mx = p2[i] + i;
23     }
24 }
25 int main()
26 {
27     scanf("%s", str + 1);
28     n = strlen(str + 1);
29     str[0] = '#';
30     str[n + 1] = '$';
31     manacher1();
32     manacher2();
33     return 0;
34 }

```

4.6 回文自动机 (ct)

```

1  char str[maxn];
2  int next[maxn][26], fail[maxn], len[maxn], cnt[maxn], last, tot, n;
3  inline int new_node(int l)
4  {
5      len[++tot] = l;
6      return tot;
7  }
8  inline void init()
9  {
10     tot = -1;
11     new_node(0);
12     new_node(-1);
13     str[0] = -1;

```

```

14     fail[0] = 1;
15 }
16 inline int get_fail(int x)
17 {
18     while (str[n - len[x] - 1] != str[n]) x = fail[x];
19     return x;
20 }
21 inline void extend(int c)
22 {
23     ++n;
24     int cur = get_fail(last);
25     if (!next[cur][c])
26     {
27         int now = new_node(len[cur] + 2);
28         fail[now] = next[get_fail(fail[cur])][c];
29         next[cur][c] = now;
30     }
31     last = next[cur][c];
32     ++cnt[last];
33 }
34 long long ans;
35 inline void count()
36 {
37     for (int i = tot; i; --i)
38     {
39         cnt[fail[i]] += cnt[i];
40         cmax(ans, 1ll * len[i] * cnt[i]);
41     }
42 }
43 int main()
44 {
45     scanf("%s", str + 1);
46     init();
47     for (int i = 1; str[i]; ++i)
48         extend(str[i] - 'a');
49     count();
50     printf("%lld\n", ans);
51     return 0;
52 }

```

4.7 最小表示法 (ct)

```

1 int main()
2 {
3     int i = 0, j = 1, k = 0;
4     while (i < n && j < n && k < n)
5     {
6         int tmp = a[(i + k) % n] - a[(j + k) % n];
7         if (!tmp) k++;
8         else
9         {
10             if (tmp > 0) i += k + 1;
11             else j += k + 1;
12             if (i == j) ++j;
13             k = 0;
14         }
15     }
16     j = dmin(i, j);
17     for (int i = j; i < n; ++i) printf("%d ", a[i]);

```

```
18 |     for (int i = 0; i < j - 1; ++i) printf("%d ", a[i]);  
19 |     if (j > 0) printf("%d\n", a[j - 1]);  
20 |     return 0;  
21 | }
```

Chapter 5

Data Structure

5.1 莫队 (ct)

```
1 int size;
2 struct Query {
3     int l, r, id;
4     inline bool operator < (const Query &that) const {return l / size != that.l / size ? l < that.l
5         ↪ : ((l / size) & 1 ? r < that.r : r > that.r);}
6 } q[maxn];
7 int main()
8 {
9     size = (int) sqrt(n * 1.0);
10    std::sort(q + 1, q + m + 1);
11    int l = 1, r = 0;
12    for (int i = 1; i <= m; ++i)
13    {
14        for (; r < q[i].r; ) add(++r);
15        for (; r > q[i].r; ) del(r--);
16        for (; l < q[i].l; ) del(l++);
17        for (; l > q[i].l; ) add(--l);
18        /*
19         * write your code here.
20         */
21    }
22    return 0;
23 }
```

5.2 ST 表 (ct)

```
1 int a[maxn], f[20][maxn], n;
2 int Log[maxn];
3 void build()
4 {
5     for (int i = 1; i <= n; ++i) f[0][i] = a[i];
6
7     int lim = Log[n];
8     for (int j = 1; j <= lim; ++j)
9     {
10        int *fj = f[j], *fj1 = f[j - 1];
11        for (int i = 1; i <= n - (1 << j) + 1; ++i)
12            fj[i] = dmax(fj1[i], fj1[i + (1 << (j - 1))]);
13    }
14 }
```

```

14 int Query(int l, int r)
15 {
16     int k = Log[r - l + 1];
17     return dmax(f[k][l], f[k][r - (1 << k) + 1]);
18 }
19 int main()
20 {
21     scanf("%d", &n);
22     Log[0] = -1;
23     for (int i = 1; i <= n; ++i)
24     {
25         scanf("%d", &a[i]);
26         Log[i] = Log[i >> 1] + 1;
27     }
28     build();
29     int q;
30     scanf("%d", &q);
31     for (; q; --q)
32     {
33         int l, r; scanf("%d%d", &l, &r);
34         printf("%d\n", Query(l, r));
35     }
36 }

```

5.3 带权并查集 (ct)

```

1 struct edge
2 {
3     int a, b, w;
4     inline bool operator < (const edge &that) const {return w > that.w;}
5 } e[maxm];
6 int fa[maxn], f1[maxn], f2[maxn], f1cnt, f2cnt, val[maxn], size[maxn];
7 int main()
8 {
9     int n, m; scanf("%d%d", &n, &m);
10    for (int i = 1; i <= m; ++i)
11        scanf("%d%d%d", &e[i].a, &e[i].b, &e[i].w);
12    for (int i = 1; i <= n; ++i) size[i] = 1;
13    std::sort(e + 1, e + m + 1);
14    for (int i = 1; i <= m; ++i)
15    {
16        int x = e[i].a, y = e[i].b;
17        for (; fa[x]; x = fa[x]);
18        for (; fa[y]; y = fa[y]);
19        if (x != y)
20        {
21            if (size[x] < size[y]) std::swap(x, y);
22            size[x] += size[y];
23            val[y] = e[i].w;
24            fa[y] = x;
25        }
26    }
27    int q; scanf("%d", &q);
28    for (; q; --q)
29    {
30        int a, b; scanf("%d%d", &a, &b); f1cnt = f2cnt = 0;
31        for (; fa[a]; a = fa[a]) f1[++f1cnt] = a;
32        for (; fa[b]; b = fa[b]) f2[++f2cnt] = b;

```

```

33     if (a != b) {puts("-1"); continue;}
34     while (f1cnt && f2cnt && f1[f1cnt] == f2[f2cnt]) --f1cnt, --f2cnt;
35     int ret = 0x7fffffff;
36     for (; f1cnt; --f1cnt) cmin(ret, val[f1[f1cnt]]);
37     for (; f2cnt; --f2cnt) cmin(ret, val[f2[f2cnt]]);
38     printf("%d\n", ret);
39 }
40 return 0;
41 }

```

5.4 可并堆 (ct)

```

1 struct Node {
2     Node *ch[2];
3     ll val; int size;
4     inline void update()
5     {
6         size = ch[0] -> size + ch[1] -> size + 1;
7     }
8 } mem[maxn], *rt[maxn];
9 Node *merge(Node *a, Node *b)
10 {
11     if (a == mem) return b;
12     if (b == mem) return a;
13     if (a -> val < b -> val) std::swap(a, b);
14     // a -> pushdown();
15     std::swap(a -> ch[0], a -> ch[1]);
16     a -> ch[1] = merge(a -> ch[1], b);
17     a -> update();
18     return a;
19 }

```

5.5 zkw 线段树 (ct)

```

1 // must be 0-based !
2 inline void build()
3 {
4     for (int i = M - 1; i; --i) tr[i] = dmax(tr[i << 1], tr[i << 1 | 1]);
5 }
6 inline void Change(int x, int v)
7 {
8     x += M; tr[x] = v; x >>= 1;
9     for (; x; x >>= 1) tr[x] = dmax(tr[x << 1], tr[x << 1 | 1]);
10 }
11 inline int Query(int s, int t)
12 {
13     int ret = -0x7fffffff;
14     for (s = s + M - 1, t = t + M + 1; s ^ t ^ 1; s >>= 1, t >>= 1)
15     {
16         if (~s & 1) cmax(ret, tr[s ^ 1]);
17         if (t & 1) cmax(ret, tr[t ^ 1]);
18     }
19     return ret;
20 }
21 int main()
22 {
23     int n; scanf("%d", &n);
24     for (M = 1; M < n; M <<= 1) ;

```



```

25     for (int i = 0; i < n; ++i)
26         scanf("%d", &tr[i + M]);
27     for (int i = n; i < M; ++i) tr[i + M] = -0x7fffffff;
28     build();
29     int q; scanf("%d", &q);
30     for (; q; --q)
31     {
32         int l, r; scanf("%d%d", &l, &r); --l, --r;
33         printf("%d\n", Query(l, r));
34     }
35     return 0;
36 }

```

5.6 Splay (ct)

指针版

```

1  struct Node *null;
2  struct Node {
3      Node *ch[2], *fa;
4      int val; bool rev;
5      inline bool type()
6      {
7          return fa -> ch[1] == this;
8      }
9      inline void pushup()
10     {
11     }
12     inline void pushdown()
13     {
14         if (rev)
15         {
16             ch[0] -> rev ^= 1;
17             ch[1] -> rev ^= 1;
18             std::swap(ch[0], ch[1]);
19             rev ^= 1;
20         }
21     }
22     inline void rotate()
23     {
24         bool d = type(); Node *f = fa, *gf = f -> fa;
25         (fa = gf, f -> fa != null) ? fa -> ch[f -> type()] = this : 0;
26         (f -> ch[d] = ch[!d]) != null ? ch[!d] -> fa = f : 0;
27         (ch[!d] = f) -> fa = this;
28         f -> pushup();
29     }
30     inline void splay()
31     {
32         for (; fa != null; rotate())
33             if (fa -> fa != null)
34                 (type() == fa -> type() ? fa : this) -> rotate();
35         pushup();
36     }
37 } mem[maxn];

```

数组版

```

1 // BZOJ - 1500 维修数列
2 int fa[maxn], ch[maxn][2], a[maxn], size[maxn], cnt;
3 int sum[maxn], lmx[maxn], rmx[maxn], mx[maxn], v[maxn], id[maxn], root;
4 bool rev[maxn], tag[maxn];
5 inline void update(R int x)
6 {
7     R int ls = ch[x][0], rs = ch[x][1];
8     size[x] = size[ls] + size[rs] + 1;
9     sum[x] = sum[ls] + sum[rs] + v[x];
10    mx[x] = gmax(mx[ls], mx[rs]);
11    cmax(mx[x], lmx[rs] + rmx[ls] + v[x]);
12    lmx[x] = gmax(lmx[ls], sum[ls] + v[x] + lmx[rs]);
13    rmx[x] = gmax(rmx[rs], sum[rs] + v[x] + rmx[ls]);
14 }
15 inline void pushdown(R int x)
16 {
17     R int ls = ch[x][0], rs = ch[x][1];
18     if (tag[x])
19     {
20         rev[x] = tag[x] = 0;
21         if (ls) tag[ls] = 1, v[ls] = v[x], sum[ls] = size[ls] * v[x];
22         if (rs) tag[rs] = 1, v[rs] = v[x], sum[rs] = size[rs] * v[x];
23         if (v[x] >= 0)
24         {
25             if (ls) lmx[ls] = rmx[ls] = mx[ls] = sum[ls];
26             if (rs) lmx[rs] = rmx[rs] = mx[rs] = sum[rs];
27         }
28         else
29         {
30             if (ls) lmx[ls] = rmx[ls] = 0, mx[ls] = v[x];
31             if (rs) lmx[rs] = rmx[rs] = 0, mx[rs] = v[x];
32         }
33     }
34     if (rev[x])
35     {
36         rev[x] ^= 1; rev[ls] ^= 1; rev[rs] ^= 1;
37         swap(lmx[ls], rmx[ls]); swap(lmx[rs], rmx[rs]);
38         swap(ch[ls][0], ch[ls][1]); swap(ch[rs][0], ch[rs][1]);
39     }
40 }
41 inline void rotate(R int x)
42 {
43     R int f = fa[x], gf = fa[f], d = ch[f][1] == x;
44     if (f == root) root = x;
45     (ch[f][d] = ch[x][d ^ 1]) > 0 ? fa[ch[f][d]] = f : 0;
46     (fa[x] = gf) > 0 ? ch[gf][ch[gf][1] == f] = x : 0;
47     fa[ch[x][d ^ 1] = f] = x;
48     update(f);
49 }
50 inline void splay(R int x, R int rt)
51 {
52     while (fa[x] != rt)
53     {
54         R int f = fa[x], gf = fa[f];
55         if (gf != rt) rotate((ch[gf][1] == f) ^ (ch[f][1] == x) ? x : f);
56         rotate(x);
57     }
58     update(x);
59 }

```

```

60 void build(R int l, R int r, R int rt)
61 {
62     if (l > r) return ;
63     R int mid = l + r >> 1, now = id[mid], last = id[rt];
64     if (l == r)
65     {
66         sum[now] = a[l];
67         size[now] = 1;
68         tag[now] = rev[now] = 0;
69         if (a[l] >= 0) lmx[now] = rmx[now] = mx[now] = a[l];
70         else lmx[now] = rmx[now] = 0, mx[now] = a[l];
71     }
72     else
73     {
74         build(l, mid - 1, mid);
75         build(mid + 1, r, mid);
76     }
77     v[now] = a[mid];
78     fa[now] = last;
79     update(now);
80     ch[last][mid >= rt] = now;
81 }
82 int find(R int x, R int rank)
83 {
84     if (tag[x] || rev[x]) pushdown(x);
85     R int ls = ch[x][0], rs = ch[x][1], lsize = size[ls];
86     if (lsize + 1 == rank) return x;
87     if (lsize >= rank)
88         return find(ls, rank);
89     else
90         return find(rs, rank - lsize - 1);
91 }
92 inline int prepare(R int l, R int tot)
93 {
94     R int x = find(root, l - 1), y = find(root, l + tot);
95     splay(x, 0);
96     splay(y, x);
97     return ch[y][0];
98 }
99 std::queue <int> q;
100 inline void Insert(R int left, R int tot)
101 {
102     for (R int i = 1; i <= tot; ++i) a[i] = FastIn();
103     for (R int i = 1; i <= tot; ++i)
104         if (!q.empty()) id[i] = q.front(), q.pop();
105         else id[i] = ++cnt;
106     build(1, tot, 0);
107     R int z = id[(1 + tot) >> 1];
108     R int x = find(root, left), y = find(root, left + 1);
109     splay(x, 0);
110     splay(y, x);
111     fa[z] = y;
112     ch[y][0] = z;
113     update(y);
114     update(x);
115 }
116 void rec(R int x)
117 {
118     if (!x) return ;
119     R int ls = ch[x][0], rs = ch[x][1];
120     rec(ls); rec(rs); q.push(x);

```

```

121     fa[x] = ch[x][0] = ch[x][1] = 0;
122     tag[x] = rev[x] = 0;
123 }
124 inline void Delete(R int l, R int tot)
125 {
126     R int x = prepare(l, tot), f = fa[x];
127     rec(x); ch[f][0] = 0;
128     update(f); update(fa[f]);
129 }
130 inline void Makesame(R int l, R int tot, R int val)
131 {
132     R int x = prepare(l, tot), y = fa[x];
133     v[x] = val; tag[x] = 1; sum[x] = size[x] * val;
134     if (val >= 0) lmx[x] = rmx[x] = mx[x] = sum[x];
135     else lmx[x] = rmx[x] = 0, mx[x] = val;
136     update(y); update(fa[y]);
137 }
138 inline void Reverse(R int l, R int tot)
139 {
140     R int x = prepare(l, tot), y = fa[x];
141     if (!tag[x])
142     {
143         rev[x] ^= 1;
144         swap(ch[x][0], ch[x][1]);
145         swap(lmx[x], rmx[x]);
146         update(y); update(fa[y]);
147     }
148 }
149 inline void Query(R int l, R int tot)
150 {
151     R int x = prepare(l, tot);
152     printf("%d\n", sum[x] );
153 }
154 #define inf ((1 << 30))
155 int main()
156 {
157     R int n = FastIn(), m = FastIn(), l, tot, val;
158     R char op, op2;
159     mx[0] = a[1] = a[n + 2] = -inf;
160     for (R int i = 2; i <= n + 1; i++)
161     {
162         a[i] = FastIn();
163     }
164     for (R int i = 1; i <= n + 2; ++i) id[i] = i;
165     n += 2; cnt = n; root = (n + 1) >> 1;
166     build(1, n, 0);
167     for (R int i = 1; i <= m; i++)
168     {
169         op = getc();
170         while (op < 'A' || op > 'Z') op = getc();
171         getc(); op2 = getc(); getc(); getc(); getc(); getc();
172         if (op == 'M' && op2 == 'X')
173         {
174             printf("%d\n", mx[root] );
175         }
176         else
177         {
178             l = FastIn() + 1; tot = FastIn();
179             if (op == 'I') Insert(l, tot);
180             if (op == 'D') Delete(l, tot);
181             if (op == 'M') val = FastIn(), Makesame(l, tot, val);

```

```

182         if (op == 'R')
183             Reverse(l, tot);
184         if (op == 'G')
185             Query(l, tot);
186     }
187 }
188 return 0;
189 }

```

5.7 Treap (ct)

```

1 struct Treap {
2     Treap *ls, *rs;
3     int size;
4     bool rev;
5     inline void update()
6     {
7         size = ls -> size + rs -> size + 1;
8     }
9     inline void set_rev()
10    {
11        rev ^= 1;
12        std::swap(ls, rs);
13    }
14    inline void pushdown()
15    {
16        if (rev)
17        {
18            ls -> set_rev();
19            rs -> set_rev();
20            rev = 0;
21        }
22    }
23 } mem[maxn], *root, *null = mem;
24 struct Pair {
25     Treap *fir, *sec;
26 };
27 Treap *build(R int l, R int r)
28 {
29     if (l > r) return null;
30     R int mid = l + r >> 1;
31     R Treap *now = mem + mid;
32     now -> rev = 0;
33     now -> ls = build(l, mid - 1);
34     now -> rs = build(mid + 1, r);
35     now -> update();
36
37     return now;
38 }
39 inline Treap *Find_kth(R Treap *now, R int k)
40 {
41     if (!k) return mem;
42     if (now -> ls -> size >= k) return Find_kth(now -> ls, k);
43     else if (now -> ls -> size + 1 == k) return now;
44     else return Find_kth(now -> rs, k - now -> ls -> size - 1);
45 }
46 Treap *merge(R Treap *a, R Treap *b)
47 {
48     if (a == null) return b;

```

```

48     if (b == null) return a;
49     if (rand() % (a -> size + b -> size) < a -> size)
50     {
51         a -> pushdown();
52         a -> rs = merge(a -> rs, b);
53         a -> update();
54         return a;
55     }
56     else
57     {
58         b -> pushdown();
59         b -> ls = merge(a, b -> ls);
60         b -> update();
61         return b;
62     }
63 }
64 Pair split(R Treap *now, R int k)
65 {
66     if (now == null) return (Pair) {null, null};
67     R Pair t = (Pair) {null, null};
68     now -> pushdown();
69     if (k <= now -> ls -> size)
70     {
71         t = split(now -> ls, k);
72         now -> ls = t.sec;
73         now -> update();
74         t.sec = now;
75     }
76     else
77     {
78         t = split(now -> rs, k - now -> ls -> size - 1);
79         now -> rs = t.fir;
80         now -> update();
81         t.fir = now;
82     }
83     return t;
84 }
85 inline void set_rev(int l, int r)
86 {
87     R Pair x = split(root, l - 1);
88     R Pair y = split(x.sec, r - l + 1);
89     y.fir -> set_rev();
90     root = merge(x.fir, merge(y.fir, y.sec));
91 }

```

5.8 可持久化平衡树 (ct)

```

1  char str[maxn];
2  struct Treap
3  {
4      Treap *ls, *rs;
5      char data; int size;
6      inline void update()
7      {
8          size = ls -> size + rs -> size + 1;
9      }
10 } *root[maxn], mem[maxcnt], *tot = mem, *last = mem, *null = mem;
11 inline Treap* new_node(char ch)
12 {

```

```

13     *++tot = (Treap) {null, null, ch, 1};
14     return tot;
15 }
16 struct Pair
17 {
18     Treap *fir, *sec;
19 };
20 inline Treap *copy(Treap *x)
21 {
22     if (x == null) return null;
23     if(x > last) return x;
24     *++tot = *x;
25     return tot;
26 }
27 Pair Split(Treap *x, int k)
28 {
29     if (x == null) return (Pair) {null, null};
30     Pair y;
31     Treap *nw = copy(x);
32     if (nw -> ls -> size >= k)
33     {
34         y = Split(nw -> ls, k);
35         nw -> ls = y.sec;
36         nw -> update();
37         y.sec = nw;
38     }
39     else
40     {
41         y = Split(nw -> rs, k - nw -> ls -> size - 1);
42         nw -> rs = y.fir;
43         nw -> update();
44         y.fir = nw;
45     }
46     return y;
47 }
48 Treap *Merge(Treap *a, Treap *b)
49 {
50     if (a == null) return b;
51     if (b == null) return a;
52     Treap *nw;
53     if (rand() % (a -> size + b -> size) < a -> size)
54     {
55         nw = copy(a);
56         nw -> rs = Merge(nw -> rs, b);
57     }
58     else
59     {
60         nw = copy(b);
61         nw -> ls = Merge(a, nw -> ls);
62     }
63     nw -> update();
64     return nw;
65 }
66 Treap *Build(int l, int r)
67 {
68     if (l > r) return null;
69     R int mid = l + r >> 1;
70     Treap *nw = new_node(str[mid]);
71     nw -> ls = Build(l, mid - 1);
72     nw -> rs = Build(mid + 1, r);
73     nw -> update();

```

```

74     return nw;
75 }
76 int now;
77 inline void Insert(int k, char ch)
78 {
79     Pair x = Split(root[now], k);
80     Treap *nw = new_node(ch);
81     root[++now] = Merge(Merge(x.fir, nw), x.sec);
82 }
83 inline void Del(int l, int r)
84 {
85     Pair x = Split(root[now], l - 1);
86     Pair y = Split(x.sec, r - l + 1);
87     root[++now] = Merge(x.fir, y.sec);
88 }
89 inline void Copy(int l, int r, int ll)
90 {
91     Pair x = Split(root[now], l - 1);
92     Pair y = Split(x.sec, r - l + 1);
93     Pair z = Split(root[now], ll);
94     Treap *ans = y.fir;
95     root[++now] = Merge(Merge(z.fir, ans), z.sec);
96 }
97 void Print(Treap *x, int l, int r)
98 {
99     if (!x) return ;
100     if (l > r) return;
101     R int mid = x -> ls -> size + 1;
102     if (r < mid)
103     {
104         Print(x -> ls, l, r);
105         return ;
106     }
107     if (l > mid)
108     {
109         Print(x -> rs, l - mid, r - mid);
110         return ;
111     }
112     Print(x -> ls, l, mid - 1);
113     printf("%c", x -> data );
114     Print(x -> rs, 1, r - mid);
115 }
116 void Printtree(Treap *x)
117 {
118     if (!x) return;
119     Printtree(x -> ls);
120     printf("%c", x -> data );
121     Printtree(x -> rs);
122 }
123 int main()
124 {
125     srand(time(0) + clock());
126     null -> ls = null -> rs = null; null -> size = 0; null -> data = 0;
127     int n = F();
128     gets(str + 1);
129     int len = strlen(str + 1);
130     root[0] = Build(1, len);
131     while (1)
132     {
133         last = tot;
134         R char opt = getc();

```



```

135     while (opt < 'A' || opt > 'Z')
136     {
137         if (opt == EOF) return 0;
138         opt = getc();
139     }
140     if (opt == 'I')
141     {
142         R int x = F();
143         R char ch = getc();
144         Insert(x, ch);
145     }
146     else if (opt == 'D')
147     {
148         R int l = F(), r = F();
149         Del(l, r);
150     }
151     else if (opt == 'C')
152     {
153         R int x = F(), y = F(), z = F();
154         Copy(x, y, z);
155     }
156     else if (opt == 'P')
157     {
158         R int x = F(), y = F(), z = F();
159         Print(root[now - x], y, z);
160         puts("");
161     }
162 }
163 return 0;
164 }

```

5.9 CDQ 分治 (ct)

```

1 struct event
2 {
3     int x, y, id, opt, ans;
4 } t[maxn], q[maxn];
5 void cdq(int left, int right)
6 {
7     if (left == right) return ;
8     R int mid = left + right >> 1;
9     cdq(left, mid);
10    cdq(mid + 1, right);
11    //分成若干个子问题
12    ++now;
13    for (int i = left, j = mid + 1; j <= right; ++j)
14    {
15        for (; i <= mid && q[i].x <= q[j].x; ++i)
16            if (!q[i].opt)
17                add(q[i].y, q[i].ans);
18        //考虑前面的修改操作对后面的询问的影响
19        if (q[j].opt)
20            q[j].ans += query(q[j].y);
21    }
22    R int i, j, k = 0;
23    //以下相当于归并排序
24    for (i = left, j = mid + 1; i <= mid && j <= right; )
25    {
26        if (q[i].x <= q[j].x)

```

```

27         t[k++] = q[i++];
28     else
29         t[k++] = q[j++];
30 }
31 for (; i <= mid; )
32     t[k++] = q[i++];
33 for (; j <= right; )
34     t[k++] = q[j++];
35 for (int i = 0; i < k; ++i)
36     q[left + i] = t[i];
37 }

```

5.10 Bitset (ct)

```

1  namespace Game {
2  #define maxn 300010
3  #define maxs 30010
4  uint b1[32][maxs], b2[32][maxs];
5  int popcnt[256];
6  inline void set(R uint *s, R int pos)
7  {
8      s[pos >> 5] |= 1u << (pos & 31);
9  }
10 inline int popcount(R uint x)
11 {
12     return popcnt[x >> 24 & 255]
13         + popcnt[x >> 16 & 255]
14         + popcnt[x >> 8 & 255]
15         + popcnt[x & 255];
16 }
17 void main() {
18     int n, q;
19     scanf("%d%d", &n, &q);
20
21     char *s1 = new char[n + 1];
22     char *s2 = new char[n + 1];
23     scanf("%s%s", s1, s2);
24
25     uint *anss = new uint[q];
26
27     for (R int i = 1; i < 256; ++i) popcnt[i] = popcnt[i >> 1] + (i & 1);
28
29     #define modify(x, _p)\
30     {\
31         for (R int j = 0; j < 32 && j <= _p; ++j)\
32             set(b##x[j], _p - j);\
33     }
34
35     for (R int i = 0; i < n; ++i)
36         if (s1[i] == '0') modify(1, 3 * i)
37         else if (s1[i] == '1') modify(1, 3 * i + 1)
38         else modify(1, 3 * i + 2)
39
40     for (R int i = 0; i < n; ++i)
41         if (s2[i] == '1') modify(2, 3 * i)
42         else if (s2[i] == '2') modify(2, 3 * i + 1)
43         else modify(2, 3 * i + 2)
44
45     for (int Q = 0; Q < q; ++Q) {
46         R int x, y, l;

```

```

40     scanf("%d%d%d", &x, &y, &l); x *= 3; y *= 3; l *= 3;
41     uint *f1 = b1[x & 31], *f2 = b2[y & 31], ans = 0;
42     R int i = x >> 5, j = y >> 5, p, lim;
43     for (p = 0, lim = l >> 5; p + 8 < lim; p += 8, i += 8, j += 8)
44     {
45         ans += popcount(f1[i + 0] & f2[j + 0]);
46         ans += popcount(f1[i + 1] & f2[j + 1]);
47         ans += popcount(f1[i + 2] & f2[j + 2]);
48         ans += popcount(f1[i + 3] & f2[j + 3]);
49         ans += popcount(f1[i + 4] & f2[j + 4]);
50         ans += popcount(f1[i + 5] & f2[j + 5]);
51         ans += popcount(f1[i + 6] & f2[j + 6]);
52         ans += popcount(f1[i + 7] & f2[j + 7]);
53     }
54     for (; p < lim; ++p, ++i, ++j) ans += popcount(f1[i] & f2[j]);
55     R uint S = (1u << (l & 31)) - 1;
56     ans += popcount(f1[i] & f2[j] & S);
57     anss[Q] = ans;
58 }
59 output_arr(anss, q * sizeof(uint));
60 }
61 }

```

Chapter 6

Others

6.1 vimrc (gy)

```
1 se et ts=4 sw=4 sts=4 nu sc sm lbr is hls mouse=a
2 sy on
3 ino <tab> <c-n>
4 ino <s-tab> <tab>
5 au bufwinenter * winc L

6 nm <f6> ggVG"+y
7 nm <f7> :w<cr>:make<cr>
8 nm <f8> :!@<cr>
9 nm <f9> :!@ < in<cr>
10 nm <s-f9> :!(time @ < in &> out) &>> out<cr>:sp out<cr>

11 au filetype cpp cm @ ./a.out | se cin fdm=syntax mp=g++\ \% -std=c++11\ -Wall\ -Wextra\
    ↪ -Wconversion\ -O2

12 map <c-p> :ha<cr>
13 se pheader=%N@%F popt=number:y

14 au filetype java cm @ java %< | se cin fdm=syntax mp=javac\ \%
15 au filetype python cm @ python % | se si fdm=indent
16 au bufenter *.kt setf kotlin
17 au filetype kotlin cm @ kotlin _%<Kt | se si mp=kotlinc\ %
```

6.2 STL 释放内存 (Durandal)

```
1 template <typename T>
2 __inline void clear(T &container) {
3     container.clear();
4     T(container).swap(container);
5 }
```

6.3 开栈 (Durandal)

```
1 register char *_sp __asm__("rsp");
2 int main() {
3     const int size = 400 << 20; // 400 MB
4     static char *sys, *mine(new char[size] + size - 4096);
5     sys = _sp; _sp = mine;
6     _main(); // main method
```

```

7  _sp = sys;
8  return 0;
9  }

```

6.4 O3 (gy)

```

1  __attribute__((optimize("-O3"))) int main() { return 0; }

```

6.5 Java Template (gy)

```

1  import java.io.*;
2  import java.math.*;
3  import java.util.*;

4  public class Template {
5      // Input
6      private static BufferedReader reader;
7      private static StringTokenizer tokenizer;
8      private static String next() {
9          try {
10             while (tokenizer == null || !tokenizer.hasMoreTokens())
11                 tokenizer = new StringTokenizer(reader.readLine());
12             } catch (IOException e) {
13                 // do nothing
14             }
15             return tokenizer.nextToken();
16         }
17         private static int nextInt() {
18             return Integer.parseInt(next());
19         }
20         private static double nextDouble() {
21             return Double.parseDouble(next());
22         }
23         private static BigInteger nextBigInteger() {
24             return new BigInteger(next());
25         }

26         public static void main(String[] args) {
27             reader = new BufferedReader(new InputStreamReader(System.in));
28             Scanner scanner = new Scanner(System.in);
29             while (scanner.hasNext())
30                 scanner.next();
31         }

32         // BigInteger & BigDecimal
33         private static void bigDecimal() {
34             BigDecimal a = BigDecimal.valueOf(1.0);
35             BigDecimal b = a.setScale(50, RoundingMode.HALF_EVEN);
36             BigDecimal c = b.abs();
37             // if scale omitted, b.scale is used
38             BigDecimal d = c.divide(b, 50, RoundingMode.HALF_EVEN);
39             // since Java 9
40             BigDecimal e = d.sqrt(new MathContext(50, RoundingMode.HALF_EVEN));
41             BigDecimal x = new BigDecimal(BigInteger.ZERO);
42             BigInteger y = BigDecimal.ZERO.toBigInteger(); // RoundingMode.DOWN
43             y = BigDecimal.ZERO.setScale(0, RoundingMode.HALF_EVEN).unscaledValue();
44         }
45         // sqrt for Java 8

```

```

46 // can solve scale=100 for 10000 times in about 1 second
47 private static BigDecimal sqrt(BigDecimal a, int scale) {
48     if (a.compareTo(BigDecimal.ZERO) < 0)
49         return BigDecimal.ZERO.setScale(scale, RoundingMode.HALF_EVEN);
50     int length = a.precision() - a.scale();
51     BigDecimal ret = new BigDecimal(BigInteger.ONE, -length / 2);
52     for (int i = 1; i <= Integer.highestOneBit(scale) + 10; i++)
53         ret = ret.add(a.divide(ret, scale,
54             ↳ RoundingMode.HALF_EVEN)).divide(BigDecimal.valueOf(2), scale,
55             ↳ RoundingMode.HALF_EVEN);
56     return ret;
57 }
58 // can solve a=2^10000 for 100000 times in about 1 second
59 private static BigInteger sqrt(BigInteger a) {
60     int length = a.bitLength() - 1;
61     BigInteger l = BigInteger.ZERO.setBit(length / 2), r = BigInteger.ZERO.setBit(length / 2);
62     while (!l.equals(r)) {
63         BigInteger m = l.add(r).shiftRight(1);
64         if (m.multiply(m).compareTo(a) < 0)
65             l = m.add(BigInteger.ONE);
66         else
67             r = m;
68     }
69     return l;
70 }
71 // Collections
72 private static void arrayList() {
73     List<Integer> list = new ArrayList<>();
74     // Generic array is banned
75     List[] lists = new List[100];
76     lists[0] = new ArrayList<Integer>();
77     // for List<Integer>, remove(Integer) stands for element, while remove(int) stands for
78     // ↳ index
79     list.remove(list.get(1));
80     list.remove(list.size() - 1);
81     list.clear();
82     Queue<Integer> queue = new LinkedList<>();
83     // return the value without popping
84     queue.peek();
85     // pop and return the value
86     queue.poll();
87     Queue<Integer> priorityQueue = new PriorityQueue<>();
88     Deque<Integer> deque = new ArrayDeque<>();
89     deque.peekFirst();
90     deque.peekLast();
91     deque.pollFirst();
92     TreeSet<Integer> set = new TreeSet<>();
93     TreeSet<Integer> anotherSet = new TreeSet<>(Comparator.reverseOrder());
94     set.ceiling(1);
95     set.floor(1);
96     set.lower(1);
97     set.higher(1);
98     set.contains(1);
99     HashSet<Integer> hashSet = new HashSet<>();
100     HashMap<String, Integer> map = new HashMap<>();
101     map.put("", 1);
102     map.get("");
103     map.forEach((string, integer) -> System.out.println(string + integer));
104     TreeMap<String, Integer> treeMap = new TreeMap<>();
105     Arrays.sort(new int[10]);

```

```

103     Arrays.sort(new Integer[10], (a, b) -> {
104         if (a.equals(b)) return 0;
105         if (a > b) return -1;
106         return 1;
107     });
108     Arrays.sort(new Integer[10], Comparator.comparingInt((a) -> (int) a).reversed());
109     long a = 1_000_000_000_000_000L;
110     int b = Integer.MAX_VALUE;
111     int c = 'a';
112 }
113 }

```

6.6 Big Fraction (gy)

```

1 fun gcd(a: Long, b: Long): Long = if (b == 0L) a else gcd(b, a % b)
2
3 class Fraction(val a: BigInteger, val b: BigInteger) {
4     constructor(a: Long, b: Long) : this(BigInteger.valueOf(a / gcd(a, b)), BigInteger.valueOf(b /
5         ↳ gcd(a, b)))
6
7     operator fun plus(o: Fraction): Fraction {
8         var gcd = b.gcd(o.b)
9         val tempProduct = (b / gcd) * (o.b / gcd)
10        var ansA = a * (o.b / gcd) + o.a * (b / gcd)
11        val gcd2 = ansA.gcd(gcd)
12        ansA /= gcd2
13        gcd /= gcd2
14        return Fraction(ansA, gcd * tempProduct)
15    }
16
17    operator fun minus(o: Fraction): Fraction {
18        var gcd = b.gcd(o.b)
19        val tempProduct = (b / gcd) * (o.b / gcd)
20        var ansA = a * (o.b / gcd) - o.a * (b / gcd)
21        val gcd2 = ansA.gcd(gcd)
22        ansA /= gcd2
23        gcd /= gcd2
24        return Fraction(ansA, gcd * tempProduct)
25    }
26
27    operator fun times(o: Fraction): Fraction {
28        val gcd1 = a.gcd(o.b)
29        val gcd2 = b.gcd(o.a)
30        return Fraction((a / gcd1) * (o.a / gcd2), (b / gcd2) * (o.b / gcd1))
31    }
32 }

```

6.7 模拟退火 (ct)

```

1 db ans_x, fans;
2 inline double rand01() {return rand() / 2147483647.0;}
3 inline double randp() {return (rand() & 1 ? 1 : -1) * rand01();}
4 inline double f(double x)
5 {
6     /*
7      * write your function here.
8      */
9     if (maxx < fans) {fans = maxx; ans_x = x;}

```

```

10     return maxx;
11 }
12 int main()
13 {
14     srand(time(NULL) + clock());
15     db x = 0, fnow = f(x);
16     fans = 1e30;
17     for (db T = 1e4; T > 1e-4; T *= 0.997)
18     {
19         db nx = x + randp() * T, fnext = f(nx);
20         db delta = fnext - fnow;
21         if (delta < 1e-9 || exp(-delta / T) > rand01())
22         {
23             x = nx;
24             fnow = fnext;
25         }
26     }
27     return 0;
28 }

```

6.8 三分 (ct)

```

1 inline db cubic_search()
2 {
3     double l = -1e4, r = 1e4;
4     for (int i = 1; i <= 100; ++i)
5     {
6         double ll = (l + r) * 0.5;
7         double rr = (ll + r) * 0.5;
8         if (check(ll) < check(rr)) r = rr;
9         else l = ll;
10    }
11    return (l + r) * 0.5;
12 }

```

6.9 Zeller Congruence (gy)

```

1 int day_in_week(int year, int month, int day) {
2     if (month == 1 || month == 2)
3         month += 12, year--;
4     int c = year / 100, y = year % 100, m = month, d = day;
5     int ret = (y + y / 4 + c / 4 + 5 * c + 13 * (m + 1) / 5 + d + 6) % 7;
6     return ret >= 0 ? ret : ret + 7;
7 }

```

6.10 博弈论模型 (gy)

- Wythoff's game
 给定两堆石子，每次可以从任意一堆中取至少一个石子，或从两堆中取相同的至少一个石子，取走最后石子的胜
 先手胜当且仅当石子数满足：
 $\lfloor (b - a) \times \phi \rfloor = a, (a \leq b, \phi = \frac{\sqrt{5}+1}{2})$
 先手胜对应的石子数构成两个序列：
 Lower Wythoff sequence: $a_n = \lfloor n \times \phi \rfloor$
 Upper Wythoff sequence: $b_n = \lfloor n \times \phi^2 \rfloor$

- Fibonacci nim

给定一堆石子，第一次可以取至少一个、少于石子总数数量的石子，之后每次可以取至少一个、不超过上次取石子数量两倍的石子，取走最后石子的胜
先手胜当且仅当石子数为斐波那契数

6.11 积分表 (integral-table.com)

$$\begin{aligned}
 \int x^n dx &= \frac{1}{n+1} x^{n+1}, \quad n \neq -1 \\
 \int \frac{1}{x} dx &= \ln |x| \\
 \int u dv &= uv - \int v du \\
 \int \frac{1}{ax+b} dx &= \frac{1}{a} \ln |ax+b| \\
 \int \frac{1}{(x+a)^2} dx &= -\frac{1}{x+a} \\
 \int (x+a)^n dx &= \frac{(x+a)^{n+1}}{n+1}, \quad n \neq -1 \\
 \int x(x+a)^n dx &= \frac{(x+a)^{n+1}((n+1)x-a)}{(n+1)(n+2)} \\
 \int \frac{1}{1+x^2} dx &= \tan^{-1} x \\
 \int \frac{1}{a^2+x^2} dx &= \frac{1}{a} \tan^{-1} \frac{x}{a} \\
 \int \frac{x}{a^2+x^2} dx &= \frac{1}{2} \ln |a^2+x^2| \\
 \int \frac{x^2}{a^2+x^2} dx &= x - a \tan^{-1} \frac{x}{a} \\
 \int \frac{x^3}{a^2+x^2} dx &= \frac{1}{2} x^2 - \frac{1}{2} a^2 \ln |a^2+x^2| \\
 \int \frac{1}{ax^2+bx+c} dx &= \frac{2}{\sqrt{4ac-b^2}} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}} \\
 \int \frac{1}{(x+a)(x+b)} dx &= \frac{1}{b-a} \ln \frac{a+x}{b+x}, \quad a \neq b \\
 \int \frac{x}{(x+a)^2} dx &= \frac{a}{a+x} + \ln |a+x| \\
 \int \frac{x}{ax^2+bx+c} dx &= \frac{1}{2a} \ln |ax^2+bx+c| - \frac{b}{a\sqrt{4ac-b^2}} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}} \\
 \int \sqrt{x-a} dx &= \frac{2}{3} (x-a)^{3/2} \\
 \int \frac{1}{\sqrt{x \pm a}} dx &= 2\sqrt{x \pm a} \\
 \int \frac{1}{\sqrt{a-x}} dx &= -2\sqrt{a-x} \\
 \int x\sqrt{x-a} dx &= \begin{cases} \frac{2a}{3} (x-a)^{3/2} + \frac{2}{5} (x-a)^{5/2}, \text{ or} \\ \frac{2}{3} x(x-a)^{3/2} - \frac{4}{15} (x-a)^{5/2}, \text{ or} \\ \frac{2}{15} (2a+3x)(x-a)^{3/2} \end{cases} \\
 \int \sqrt{ax+b} dx &= \left(\frac{2b}{3a} + \frac{2x}{3} \right) \sqrt{ax+b} \\
 \int (ax+b)^{3/2} dx &= \frac{2}{5a} (ax+b)^{5/2} \\
 \int \frac{x}{\sqrt{x \pm a}} dx &= \frac{2}{3} (x \pm 2a) \sqrt{x \pm a} \\
 \int \sqrt{\frac{x}{a-x}} dx &= -\sqrt{x(a-x)} - a \tan^{-1} \frac{\sqrt{x(a-x)}}{x-a} \\
 \int \sqrt{\frac{x}{a+x}} dx &= \sqrt{x(a+x)} - a \ln (\sqrt{x} + \sqrt{a+x}) \\
 \int x\sqrt{ax+b} dx &= \frac{2}{15a^2} (-2b^2+abx+3a^2x^2)\sqrt{ax+b} \\
 \int \sqrt{x(ax+b)} dx &= \frac{1}{4a^{3/2}} \left((2ax+b)\sqrt{ax(ax+b)} - b^2 \ln \left| a\sqrt{x} + \sqrt{a(ax+b)} \right| \right) \\
 \int \sqrt{x^3(ax+b)} dx &= \left(\frac{b}{12a} - \frac{b^2}{8a^2x} + \frac{x}{3} \right) \sqrt{x^3(ax+b)} + \frac{b^3}{8a^{5/2}} \ln \left| a\sqrt{x} + \sqrt{a(ax+b)} \right| \\
 \int \sqrt{x^2 \pm a^2} dx &= \frac{1}{2} x \sqrt{x^2 \pm a^2} \pm \frac{1}{2} a^2 \ln \left| x + \sqrt{x^2 \pm a^2} \right| \\
 \int \sqrt{a^2 - x^2} dx &= \frac{1}{2} x \sqrt{a^2 - x^2} + \frac{1}{2} a^2 \tan^{-1} \frac{x}{\sqrt{a^2 - x^2}} \\
 \int x\sqrt{x^2 \pm a^2} dx &= \frac{1}{3} (x^2 \pm a^2)^{3/2} \\
 \int \frac{1}{\sqrt{x^2 \pm a^2}} dx &= \ln \left| x + \sqrt{x^2 \pm a^2} \right| \\
 \int \frac{1}{\sqrt{a^2 - x^2}} dx &= \sin^{-1} \frac{x}{a} \\
 \int \frac{x}{\sqrt{x^2 \pm a^2}} dx &= \sqrt{x^2 \pm a^2} \\
 \int \frac{x}{\sqrt{a^2 - x^2}} dx &= -\sqrt{a^2 - x^2} \\
 \int \frac{x^2}{\sqrt{x^2 \pm a^2}} dx &= \frac{1}{2} x \sqrt{x^2 \pm a^2} \mp \frac{1}{2} a^2 \ln \left| x + \sqrt{x^2 \pm a^2} \right| \\
 \int \sqrt{ax^2+bx+c} dx &= \frac{b+2ax}{4a} \sqrt{ax^2+bx+c} + \frac{4ac-b^2}{8a^{3/2}} \ln \left| 2ax+b+2\sqrt{a(ax^2+bx+c)} \right| \\
 \int x\sqrt{ax^2+bx+c} dx &= \frac{1}{48a^{5/2}} \left(2\sqrt{a}\sqrt{ax^2+bx+c} (-3b^2+2abx+8a(c+ax^2)) \right. \\
 &\quad \left. + 3(b^3-4abc) \ln \left| b+2ax+2\sqrt{a}\sqrt{ax^2+bx+c} \right| \right) \\
 \int \frac{1}{\sqrt{ax^2+bx+c}} dx &= \frac{1}{\sqrt{a}} \ln \left| 2ax+b+2\sqrt{a(ax^2+bx+c)} \right| \\
 \int \frac{x}{\sqrt{ax^2+bx+c}} dx &= \frac{1}{a} \sqrt{ax^2+bx+c} - \frac{b}{2a^{3/2}} \ln \left| 2ax+b+2\sqrt{a(ax^2+bx+c)} \right| \\
 \int \frac{dx}{(a^2+x^2)^{3/2}} &= \frac{x}{a^2\sqrt{a^2+x^2}} \\
 \int \sin ax dx &= -\frac{1}{a} \cos ax \\
 \int \sin^2 ax dx &= \frac{x}{2} - \frac{\sin 2ax}{4a}
 \end{aligned}$$

$$\begin{aligned}
\int \sin^3 ax \, dx &= -\frac{3 \cos ax}{4a} + \frac{\cos 3ax}{12a} \\
\int \cos ax \, dx &= \frac{1}{a} \sin ax \\
\int \cos^2 ax \, dx &= \frac{x}{2} + \frac{\sin 2ax}{4a} \\
\int \cos^3 ax \, dx &= \frac{3 \sin ax}{4a} + \frac{\sin 3ax}{12a} \\
\int \cos x \sin x \, dx &= \frac{1}{2} \sin^2 x + c_1 = -\frac{1}{2} \cos^2 x + c_2 = -\frac{1}{4} \cos 2x + c_3 \\
\int \cos ax \sin bx \, dx &= \frac{\cos[(a-b)x]}{2(a-b)} - \frac{\cos[(a+b)x]}{2(a+b)}, a \neq b \\
\int \sin^2 ax \cos bx \, dx &= -\frac{\sin[(2a-b)x]}{4(2a-b)} + \frac{\sin bx}{2b} - \frac{\sin[(2a+b)x]}{4(2a+b)} \\
\int \sin^2 x \cos x \, dx &= \frac{1}{3} \sin^3 x \\
\int \cos^2 ax \sin bx \, dx &= \frac{\cos[(2a-b)x]}{4(2a-b)} - \frac{\cos bx}{2b} - \frac{\cos[(2a+b)x]}{4(2a+b)} \\
\int \cos^2 ax \sin ax \, dx &= -\frac{1}{3a} \cos^3 ax \\
\int \sin^2 ax \cos^2 bx \, dx &= \frac{x}{4} - \frac{\sin 2ax}{8a} - \frac{\sin[2(a-b)x]}{16(a-b)} + \frac{\sin 2bx}{8b} - \frac{\sin[2(a+b)x]}{16(a+b)} \\
\int \sin^2 ax \cos^2 ax \, dx &= \frac{x}{8} - \frac{\sin 4ax}{32a} \\
\int \tan ax \, dx &= -\frac{1}{a} \ln \cos ax \\
\int \tan^2 ax \, dx &= -x + \frac{1}{a} \tan ax \\
\int \tan^3 ax \, dx &= \frac{1}{a} \ln \cos ax + \frac{1}{2a} \sec^2 ax \\
\int \sec x \, dx &= \ln |\sec x + \tan x| = 2 \tanh^{-1} \left(\tan \frac{x}{2} \right) \\
\int \sec^2 ax \, dx &= \frac{1}{a} \tan ax \\
\int \sec^3 x \, dx &= \frac{1}{2} \sec x \tan x + \frac{1}{2} \ln |\sec x + \tan x| \\
\int \sec x \tan x \, dx &= \sec x \\
\int \sec^2 x \tan x \, dx &= \frac{1}{2} \sec^2 x \\
\int \sec^n x \tan x \, dx &= \frac{1}{n} \sec^n x, n \neq 0 \\
\int \csc x \, dx &= \ln \left| \tan \frac{x}{2} \right| = \ln |\csc x - \cot x| + C \\
\int \csc^2 ax \, dx &= -\frac{1}{a} \cot ax \\
\int \csc^3 x \, dx &= -\frac{1}{2} \cot x \csc x + \frac{1}{2} \ln |\csc x - \cot x| \\
\int \csc^n x \cot x \, dx &= -\frac{1}{n} \csc^n x, n \neq 0 \\
\int \sec x \csc x \, dx &= \ln |\tan x| \\
\int x \cos x \, dx &= \cos x + x \sin x \\
\int x \cos ax \, dx &= \frac{1}{a^2} \cos ax + \frac{x}{a} \sin ax \\
\int x^2 \cos x \, dx &= 2x \cos x + (x^2 - 2) \sin x \\
\int x^2 \cos ax \, dx &= \frac{2x \cos ax}{a^2} + \frac{a^2 x^2 - 2}{a^3} \sin ax \\
\int x \sin x \, dx &= -x \cos x + \sin x \\
\int x \sin ax \, dx &= -\frac{x \cos ax}{a} + \frac{\sin ax}{a^2} \\
\int x^2 \sin x \, dx &= (2 - x^2) \cos x + 2x \sin x \\
\int x^2 \sin ax \, dx &= \frac{2 - a^2 x^2}{a^3} \cos ax + \frac{2x \sin ax}{a^2} \\
\int x \cos^2 x \, dx &= \frac{x^2}{4} + \frac{1}{8} \cos 2x + \frac{1}{4} x \sin 2x \\
\int x \sin^2 x \, dx &= \frac{x^2}{4} - \frac{1}{8} \cos 2x - \frac{1}{4} x \sin 2x
\end{aligned}$$