Platelet

Team Reference Material

(unlimited version)

凌皓煜

陈　彤

顾　逸

Contents

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%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          dij(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[maxm], *ecnt = e; // original graph
6   Edge *rlast[maxn], re[maxm], *recnt = re; // reversed-edge graph
7   Edge *tlast[maxn], te[maxn << 1], *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
55        for (R int i = 1; i <= 16; ++i)
56            fa[now][i] = fa[fa[now][i - 1]][i - 1];
57        for (R Edge *iter = last[now]; iter; iter = iter -> next)
58            if (--deg[iter -> to] == 0) q[++tail] = iter -> to;
59    }
60    dfs(0);
61    for (R int i = 1; i <= n; ++i) printf("%d\n", size[i] - 1 );
62    return 0;
63 }
```

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)

```cpp
int root, son[maxn], size[maxn], sum;
bool vis[maxn];
void dfs_root(int x, int fa)
{
    size[x] = 1; son[x] = 0;
    for (R Edge *iter = last[x]; iter; iter = iter -> next)
    {
        if (iter -> to == fa || vis[iter -> to]) continue;
        dfs_root(iter -> to, x);
        size[x] += size[iter -> to];
        cmax(son[x], size[iter -> to]);
    }
    cmax(son[x], sum - size[x]);
    if (!root || son[x] < son[root]) root = x;
}
void dfs_chain(int x, int fa, int st1, int st2)
{
    /*
        write your code here.
    */
    for (Edge *iter = last[x]; iter; iter = iter -> next)
    {
        if (vis[iter -> to] || iter -> to == fa) continue;
        dfs_chain(iter -> to, x);
    }
}
void calc(int x)
{
    for (Edge *iter = last[x]; iter; iter = iter -> next)
    {
        if (vis[iter -> to]) continue;
        dfs_chain(iter -> to, x);
        /*
            write your code here.
        */
    }
}
void work(int x)
{
    vis[x] = 1;
    calc(x);
    for (R Edge *iter = last[x]; iter; iter = iter -> next)
    {
        if (vis[iter -> to]) continue;
        root = 0;
        sum = size[iter -> to];
        dfs_root(iter -> to, 0);
        work(root);
    }
}
int main()
{
    root = 0; sum = n;
    dfs_root(1, 0);
    work(root);
    return 0;
}
```

1.11   树上倍增 (ct)

```
int fa[maxn][17], mn[maxn][17], dep[maxn];
bool vis[maxn];
void dfs(int x)
{
    vis[x] = 1;
    for (int i = 1; i <= 16; ++i)
    {
        if (dep[x] < (1 << i)) break;
        fa[x][i] = fa[fa[x][i - 1]][i - 1];
        mn[x][i] = dmin(mn[x][i - 1], mn[fa[x][i - 1]][i - 1]);
    }
    for (Edge *iter = last[x]; iter; iter = iter -> next)
        if (!vis[iter -> to])
        {
            fa[iter -> to][0] = x;
            mn[iter -> to][0] = iter -> w;
            dep[iter -> to] = dep[x] + 1;
            dfs(iter -> to);
        }
}
inline int getlca(int x, int y)
{
    if (dep[x] < dep[y]) std::swap(x, y);
    int t = dep[x] - dep[y];
    for (int i = 0; i <= 16 && t; ++i)
        if ((1 << i) & t)
            x = fa[x][i], t ^= 1 << i;
    for (int i = 16; i >= 0; --i)
        if (fa[x][i] != fa[y][i])
        {
            x = fa[x][i];
            y = fa[y][i];
        }
    if (x == y) return x;
    return fa[x][0];
}
inline int getans(int x, int f)
{
    int ans = inf, t = dep[x] - dep[f];
    for (int i = 0; i <= 16 && t; ++i)
        if (t & (1 << i))
        {
            cmin(ans, mn[x][i]);
            x = fa[x][i];
            t ^= 1 << i;
        }
    return ans;
}
```

1.12   Prufer 编码

1.13   Link-Cut Tree (ct)

```
struct Node *null;
struct Node {
    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 + l;
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[maxm], *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[maxm], *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 $0 \le G(u,v) \le C(u,v) - B(u,v)$

- 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：$S^* \to v$，容量为 $B(u,v)$；$u \to T^*$，容量为 $B(u,v)$；$u \to v$，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 $T \to S$ 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 $S^* \to T^*$ 的最大流，再将从汇点 T 到源点 S 的这条边拆掉，求一次 $S \to T$ 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 $S^* \to T^*$ 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 $S^* \to T^*$ 的最大流。若超级源点 S^* 出发的边全部满流，则 $T \to S$ 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**
  $a_1 = 1$
  $$a_{n+1} = \frac{\sum\limits_{j=1}^{n} j \cdot a_j \cdot S_{n,j}}{n}$$
  $$S_{n,j} = \sum_{i=1}^{n/j} a_{n+1-ij} = S_{n-j,j} + a_{n+1-j}$$

- **无根树计数**
  $$\begin{cases} a_n - \sum\limits_{i=1}^{n/2} a_i a_{n-i} & n \text{ is odd} \\ a_n - \sum\limits_{i=1}^{n/2} a_i a_{n-i} + \frac{1}{2} a_{\frac{n}{2}} \left(a_{\frac{n}{2}} + 1 \right) & n \text{ is even} \end{cases}$$

- **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)

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

2.2 扩展欧几里得 (gy)

```
// return gcd(a, b)
// ax+by=gcd(a,b)
int extend_gcd(int a, int b, int &x, int &y) {
    if (b == 0) {
        x = 1, y = 0;
        return a;
    }
    int res = extend_gcd(b, a % b, x, y);
    int t = y;
    y = x - a / b * y;
    x = t;
    return res;
}

// return minimal positive integer x so that ax+by=c
// or -1 if such x does not exist
int solve_equ(int a, int b, int c) {
    int x, y, d;
    d = extend_gcd(a, b, x, y);
    if (c % d)
        return -1;
    int t = c / d;
    x *= t;
    y *= t;
    int k = b / d;
    x = (x % k + k) % k;
    return x;
}

// return minimal positive integer x so that ax==b(mod p)
// or -1 if such x does not exist
int solve(int a, int b, int p) {
    a = (a % p + p) % p;
    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 $0 \le d < m, 0 \le l \le r < m$，返回 $\min\{x \ge 0 \mid l \le x \cdot d \bmod m \le r\}$，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
       ↪ number)
3   * Test Set / First Wrong Answer
4   * 2 / 2,047
5   * 2, 3 / 1,373,653
6   * 31, 73 / 9,080,191
7   * 2, 3, 5 / 25,326,001
8   * 2, 3, 5, 7 / 3,215,031,751 (> Int.MAX_VALUE)
9   * 2, 7, 61 / 4,759,123,141
10  * 2, 13, 23, 1662803 / 1,122,004,669,633
11  * 2, 3, 5, 7, 11 / 2,152,302,898,747
12  * 2, 3, 5, 7, 11, 13 / 3,474,749,660,383
13  * 2, 3, 5, 7, 11, 13, 17 / 341,550,071,728,321
14  * 2, 3, 5, 7, 11, 13, 17, 19, 23 / 3,825,123,056,546,413,051
15  * 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37 / 318,665,857,834,031,151,167,461 (> Long.MAX_VALUE)
16  * 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41 / 3,317,044,064,679,887,385,961,981
```

```cpp
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 $\min\{x \geq 0 \mid y^x \equiv z \mod p\}$

```
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("Orz, 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("Orz, 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 $0 \le a < p, 0 \le b < p$, 返回 $\min\{x \ge 0 \mid a^x \equiv b \mod p\}$

```
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  int main() {
26      int64_t a, b, p;
27      while (scanf("%lld%lld%lld", &a, &b, &p) != EOF)
28          printf("%lld\n", ex_bsgs(a, b, p));
29      return 0;
30  }
```

2.13   直线下整点个数 (gy)

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

```cpp
int64_t count(int64_t n, int64_t a, int64_t b, int64_t m) {
    if (b == 0)
        return n * (a / m);
    if (a >= m)
        return n * (a / m) + count(n, a % m, b, m);
    if (b >= m)
        return (n - 1) * n / 2 * (b / m) + count(n, a, b % m, m);
    return count((a + b * n) / m, (a + b * n) % m, m, b);
}
```

2.14   单纯形

2.15   辛普森积分

2.16   数学知识

求和公式

- $\sum\limits_{k=1}^{n} (2k-1)^2 = \frac{1}{3}n(4n^2-1)$

- $\sum\limits_{k=1}^{n} k^3 = \frac{1}{4}n^2(n+1)^2$

- $\sum\limits_{k=1}^{n} (2k-1)^3 = n^2(2n^2-1)$

- $\sum\limits_{k=1}^{n} k^4 = \frac{1}{30}n(n+1)(2n+1)(3n^2+3m-1)$

- $\sum\limits_{k=1}^{n} k^5 = \frac{1}{12}n^2(n+1)^2(2n^2+2n-1)$

- $\sum\limits_{k=1}^{n} k(k+1) = \frac{1}{3}n(n+1)(n+2)$

- $\sum\limits_{k=1}^{n} k(k+1)(k+2) = \frac{1}{4}n(n+1)(n+2)(n+3)$

- $\sum\limits_{k=1}^{n} k(k+1)(k+2)(k+3) = \frac{1}{5}n(n+1)(n+2)(n+3)(n+4)$

错排公式

D_n 表示 n 个元素错位排列的方案数
$D_1 = 0, D_2 = 1$
$D_n = (n-1)(D_{n-2} + D_{n-1}), n \ge 3$
$D_n = n! \cdot (1 - \frac{1}{1!} + \frac{1}{2!} - \cdots + (-1)^n \frac{1}{n!})$

Fibonacci sequence

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

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

第二类 Stirling number

用 $S(n,k) = \begin{Bmatrix} n \\ k \end{Bmatrix}$ 表示第二类 Stirling number
$\begin{Bmatrix} n+1 \\ k \end{Bmatrix} = k \begin{Bmatrix} n \\ k \end{Bmatrix} + \begin{Bmatrix} n \\ k-1 \end{Bmatrix}, k > 0$
$\begin{Bmatrix} 0 \\ 0 \end{Bmatrix} = 1, \begin{Bmatrix} n \\ 0 \end{Bmatrix} = \begin{Bmatrix} 0 \\ n \end{Bmatrix} = 0, n > 0$
$\begin{Bmatrix} n \\ k \end{Bmatrix} = \frac{1}{k!} \sum\limits_{j=0}^{k} (-1)^{k-j} \binom{k}{j} j^n$
$\begin{Bmatrix} n \\ k \end{Bmatrix}$ 为将 n 个元素划分成 k 个非空集合的方案数

Catalan number

c_n 表示长度为 $2n$ 的合法括号序的数量
$c_1 = 1, c_{n+1} = \sum\limits_{i=1}^{n} c_i \times c_{n+1-i}$
$c_n = \frac{\binom{2n}{n}}{n+1}$

Bell number

B_n 表示基数为 n 的集合的划分方案数
$B_i = \begin{cases} 1 & i = 0 \\ \sum\limits_{k=0}^{n} \binom{n}{k} B_k & i > 0 \end{cases}$
$B_n = \sum\limits_{k=0}^{n} \begin{Bmatrix} n \\ k \end{Bmatrix}$

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

Bernoulli number

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

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

$\sum\limits_{k=1}^{n} k^m = \dfrac{1}{m+1} \sum\limits_{k=0}^{m} \binom{m+1}{k} B_k n^{m+1-k}$

Möbius function

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

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

$g(n) = \sum\limits_{d|n} f(d) \Leftrightarrow f(n) = \sum\limits_{d|n} \mu(d) g(\frac{n}{d})$

Lagrange polynomial

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

则 $L(x) = \sum\limits_{j=0}^{n} y_j \prod\limits_{0 \le m \le n, m \ne j} \dfrac{x - x_m}{x_j - x_m}$

Chapter 3

Geometry

3.1 点、直线、圆 (gy)

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

number _sqrt(number x) {
    return std::sqrt(std::max(x, (number) 0));
}
number _asin(number x) {
    x = std::min(x, (number) 1), x = std::max(x, (number) -1);
    return std::asin(x);
}
number _acos(number x) {
    x = std::min(x, (number) 1), x = std::max(x, (number) -1);
    return std::acos(x);
}

int sgn(number x) {
    return (x > eps) - (x < -eps);
}
int cmp(number x, number y) {
    return sgn(x - y);
}

struct point {
    number x, y;
    point() {}
    point(number x, number y) : x(x), y(y) {}

    number len2() const {
        return x * x + y * y;
    }
    number len() const {
        return _sqrt(len2());
    }
    point unit() const {
        return point(x / len(), y / len());
    }
    point rotate90() const {
        return point(-y, x);
    }

    friend point operator+(const point &a, const point &b) {
        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);
```

```cpp
 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   点到凸包切线

3.3   直线凸包交点

3.4   凸包游戏

3.5   半平面交

3.6   旋转卡壳

3.7   判断圆是否有交

3.8   最小圆覆盖

3.9   最小球覆盖

3.10 $O(n^2 \log n)$ 圆交面积和重心

3.11   圆与多边形交

3.12 $O(n \log n)$ 凸多边形内的最大圆

3.13   三维凸包

3.14   三维绕轴旋转

3.15   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**
  半周长 $p = \frac{a+b+c}{2}$
  面积 $S = \frac{1}{2}aH_a = \frac{1}{2}ab \cdot \sin C = \sqrt{p(p-a)(p-b)(p-c)} = pr = \frac{abc}{4R}$
  中线长 $M_a = \frac{1}{2}\sqrt{2(b^2+c^2)-a^2} = \frac{1}{2}\sqrt{b^2+c^2+2bc \cdot \cos A}$
  角平分线长 $T_a = \frac{\sqrt{bc((b+c)^2-a^2)}}{b+c} = \frac{2bc}{b+c}\cos\frac{A}{2}$
  高 $H_a = b\sin C = \sqrt{b^2 - (\frac{a^2+b^2-c^2}{2a})^2}$
  内切圆半径 $r = \frac{S}{p} = 4R\sin\frac{A}{2}\sin\frac{B}{2}\sin\frac{C}{2} = \sqrt{\frac{(p-a)(p-b)(p-c)}{p}} = p\tan\frac{A}{2}\tan\frac{B}{2}\tan\frac{C}{2}$

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

旁切圆半径 $r_A = \frac{2S}{-a+b+c}$

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

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

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

垂心 $\left(\dfrac{\begin{vmatrix} x_2x_3+y_2y_3 & 1 & y_1 \\ x_3x_1+y_3y_1 & 1 & y_2 \\ x_1x_2+y_1y_2 & 1 & y_3 \end{vmatrix}}{2\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}, \dfrac{\begin{vmatrix} x_2x_3+y_2y_3 & x_1 & 1 \\ x_3x_1+y_3y_1 & x_2 & 1 \\ x_1x_2+y_1y_2 & x_3 & 1 \end{vmatrix}}{2\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}\right)$

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

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

- **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 $\cos a = \cos b \cdot \cos c + \sin a \cdot \sin b \cdot \cos A$
  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 $V = \frac{1}{6}\left|\overrightarrow{AB} \cdot (\overrightarrow{AC} \times \overrightarrow{AD})\right|$

Chapter 4

String

Chapter 5

Data Structure

5.1 莫队 (ct)

```
int size;
struct Query {
    int l, r, id;
    inline bool operator < (const Queuy &that) const {return l / size != that.l / size ? l < that.l
        : ((l / size) & 1 ? r < that.r : r > that.r);}
} q[maxn];
int main()
{
    size = (int) sqrt(n * 1.0);
    std::sort(q + 1, q + m + 1);
    int l = 1, r = 0;
    for (int i = 1; i <= m; ++i)
    {
        for (; r < q[i].r; ) add(++r);
        for (; r > q[i].r; ) del(r--);
        for (; l < q[i].l; ) del(l++);
        for (; l > q[i].l; ) add(--l);
        /*
            write your code here.
        */
    }
    return 0;
}
```

5.2 ST 表 (ct)

```
int a[maxn], f[20][maxn], n;
int Log[maxn];

void build()
{
    for (int i = 1; i <= n; ++i) f[0][i] = a[i];

    int lim = Log[n];
    for (int j = 1; j <= lim; ++j)
    {
        int *fj = f[j], *fj1 = f[j - 1];
        for (int i = 1; i <= n - (1 << j) + 1; ++i)
            fj[i] = dmax(fj1[i], fj1[i + (1 << (j - 1))]);
    }
}
```

```
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     return now;
37 }
38 inline Treap *Find_kth(R Treap *now, R int k)
39 {
40     if (!k) return mem;
41     if (now -> ls -> size >= k) return Find_kth(now -> ls, k);
42     else if (now -> ls -> size + 1 == k) return now;
43     else return Find_kth(now -> rs, k - now -> ls -> size - 1);
44 }
45 Treap *merge(R Treap *a, R Treap *b)
46 {
47     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      char *s1 = new char[n + 1];
21      char *s2 = new char[n + 1];
22      scanf("%s%s", s1, s2);

23      uint *anss = new uint[q];

24      for (R int i = 1; i < 256; ++i) popcnt[i] = popcnt[i >> 1] + (i & 1);

25      #define modify(x, _p)\
26      {\
27          for (R int j = 0; j < 32 && j <= _p; ++j)\
28              set(b##x[j], _p - j);\
29      }
30      for (R int i = 0; i < n; ++i)
31          if (s1[i] == '0') modify(1, 3 * i)
32          else if (s1[i] == '1') modify(1, 3 * i + 1)
33          else modify(1, 3 * i + 2)

34      for (R int i = 0; i < n; ++i)
35          if (s2[i] == '1') modify(2, 3 * i)
36          else if (s2[i] == '2') modify(2, 3 * i + 1)
37          else modify(2, 3 * i + 2)

38      for (int Q = 0; Q < q; ++Q) {
39          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
60      output_arr(anss, q * sizeof(uint));
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)

```java
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
```

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

4     operator fun plus(o: Fraction): Fraction {
5         var gcd = b.gcd(o.b)
6         val tempProduct = (b / gcd) * (o.b / gcd)
7         var ansA = a * (o.b / gcd) + o.a * (b / gcd)
8         val gcd2 = ansA.gcd(gcd)
9         ansA /= gcd2
10        gcd /= gcd2
11        return Fraction(ansA, gcd * tempProduct)
12    }

13    operator fun minus(o: Fraction): Fraction {
14        var gcd = b.gcd(o.b)
15        val tempProduct = (b / gcd) * (o.b / gcd)
16        var ansA = a * (o.b / gcd) - o.a * (b / gcd)
17        val gcd2 = ansA.gcd(gcd)
18        ansA /= gcd2
19        gcd /= gcd2
20        return Fraction(ansA, gcd * tempProduct)
21    }

22    operator fun times(o: Fraction): Fraction {
23        val gcd1 = a.gcd(o.b)
24        val gcd2 = b.gcd(o.a)
25        return Fraction((a / gcd1) * (o.a / gcd2), (b / gcd2) * (o.b / gcd1))
26    }
27 }
```

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：
  $\lfloor (b - a) \times \phi \rfloor = a, (a \le b, \phi = \frac{\sqrt{5}+1}{2})$
  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)

$$\int x^n \, dx = \frac{1}{n+1} x^{n+1}, \ n \neq -1$$

$$\int \frac{1}{x} \, dx = \ln |x|$$

$$\int u \, dv = uv - \int v \, du$$

$$\int \frac{1}{ax+b} \, dx = \frac{1}{a} \ln |ax+b|$$

$$\int \frac{1}{(x+a)^2} \, dx = -\frac{1}{x+a}$$

$$\int (x+a)^n \, dx = \frac{(x+a)^{n+1}}{n+1}, \, n \neq -1$$

$$\int x(x+a)^n \, dx = \frac{(x+a)^{n+1}((n+1)x-a)}{(n+1)(n+2)}$$

$$\int \frac{1}{1+x^2} \, dx = \tan^{-1} x$$

$$\int \frac{1}{a^2+x^2} \, dx = \frac{1}{a} \tan^{-1} \frac{x}{a}$$

$$\int \frac{x}{a^2+x^2} \, dx = \frac{1}{2} \ln |a^2+x^2|$$

$$\int \frac{x^2}{a^2+x^2} \, dx = x - a \tan^{-1} \frac{x}{a}$$

$$\int \frac{x^3}{a^2+x^2} \, dx = \frac{1}{2} x^2 - \frac{1}{2} a^2 \ln |a^2+x^2|$$

$$\int \frac{1}{ax^2+bx+c} \, dx = \frac{2}{\sqrt{4ac-b^2}} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}}$$

$$\int \frac{1}{(x+a)(x+b)} \, dx = \frac{1}{b-a} \ln \frac{a+x}{b+x}, \ a \neq b$$

$$\int \frac{x}{(x+a)^2} \, dx = \frac{a}{a+x} + \ln |a+x|$$

$$\int \frac{x}{ax^2+bx+c} \, dx = \frac{1}{2a} \ln |ax^2+bx+c| - \frac{b}{a\sqrt{4ac-b^2}} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}}$$

$$\int \sqrt{x-a} \, dx = \frac{2}{3}(x-a)^{3/2}$$

$$\int \frac{1}{\sqrt{x \pm a}} \, dx = 2\sqrt{x \pm a}$$

$$\int \frac{1}{\sqrt{a-x}} \, dx = -2\sqrt{a-x}$$

$$\int x\sqrt{x-a} \, dx = \begin{cases} \frac{2a}{3}(x-a)^{3/2} + \frac{2}{5}(x-a)^{5/2}, \text{ or} \\ \frac{2}{3}x(x-a)^{3/2} - \frac{4}{15}(x-a)^{5/2}, \text{ or} \\ \frac{2}{15}(2a+3x)(x-a)^{3/2} \end{cases}$$

$$\int \sqrt{ax+b} \, dx = \left(\frac{2b}{3a} + \frac{2x}{3} \right) \sqrt{ax+b}$$

$$\int (ax+b)^{3/2} \, dx = \frac{2}{5a}(ax+b)^{5/2}$$

$$\int \frac{x}{\sqrt{x \pm a}} \, dx = \frac{2}{3}(x \mp 2a)\sqrt{x \pm a}$$

$$\int \sqrt{\frac{x}{a-x}} \, dx = -\sqrt{x(a-x)} - a \tan^{-1} \frac{\sqrt{x(a-x)}}{x-a}$$

$$\int \sqrt{\frac{x}{a+x}} \, dx = \sqrt{x(a+x)} - a \ln \left(\sqrt{x} + \sqrt{x+a} \right)$$

$$\int x\sqrt{ax+b} \, dx = \frac{2}{15a^2}(-2b^2 + abx + 3a^2 x^2)\sqrt{ax+b}$$

$$\int \sqrt{x(ax+b)} \, dx = \frac{1}{4a^{3/2}} \left((2ax+b)\sqrt{ax(ax+b)} - b^2 \ln \left| a\sqrt{x} + \sqrt{a(ax+b)} \right| \right)$$

$$\int \sqrt{x^3(ax+b)} \, dx = \left(\frac{b}{12a} - \frac{b^2}{8a^2 x} + \frac{x}{3} \right) \sqrt{x^3(ax+b)} + \frac{b^3}{8a^{5/2}} \ln \left| a\sqrt{x} + \sqrt{a(ax+b)} \right|$$

$$\int \sqrt{x^2 \pm a^2} \, dx = \frac{1}{2} x\sqrt{x^2 \pm a^2} \pm \frac{1}{2} a^2 \ln \left| x + \sqrt{x^2 \pm a^2} \right|$$

$$\int \sqrt{a^2-x^2} \, dx = \frac{1}{2} x\sqrt{a^2-x^2} + \frac{1}{2} a^2 \tan^{-1} \frac{x}{\sqrt{a^2-x^2}}$$

$$\int x\sqrt{x^2 \pm a^2} \, dx = \frac{1}{3} \left(x^2 \pm a^2 \right)^{3/2}$$

$$\int \frac{1}{\sqrt{x^2 \pm a^2}} \, dx = \ln \left| x + \sqrt{x^2 \pm a^2} \right|$$

$$\int \frac{1}{\sqrt{a^2-x^2}} \, dx = \sin^{-1} \frac{x}{a}$$

$$\int \frac{x}{\sqrt{x^2 \pm a^2}} \, dx = \sqrt{x^2 \pm a^2}$$

$$\int \frac{x}{\sqrt{a^2-x^2}} \, dx = -\sqrt{a^2-x^2}$$

$$\int \frac{x^2}{\sqrt{x^2 \pm a^2}} \, dx = \frac{1}{2} x\sqrt{x^2 \pm a^2} \mp \frac{1}{2} a^2 \ln \left| x + \sqrt{x^2 \pm a^2} \right|$$

$$\int \sqrt{ax^2+bx+c} \, dx = \frac{b+2ax}{4a} \sqrt{ax^2+bx+c} + \frac{4ac-b^2}{8a^{3/2}} \ln \left| 2ax+b + 2\sqrt{a(ax^2+bx+c)} \right|$$

$$\int x\sqrt{ax^2+bx+c} \, dx = \frac{1}{48a^{5/2}} \left(2\sqrt{a}\sqrt{ax^2+bx+c} \left(-3b^2 + 2abx + 8a(c+ax^2) \right) + 3(b^3 - 4abc) \ln \left| b + 2ax + 2\sqrt{a}\sqrt{ax^2+bx+c} \right| \right)$$

$$\int \frac{1}{\sqrt{ax^2+bx+c}} \, dx = \frac{1}{\sqrt{a}} \ln \left| 2ax+b + 2\sqrt{a(ax^2+bx+c)} \right|$$

$$\int \frac{x}{\sqrt{ax^2+bx+c}} \, dx = \frac{1}{a} \sqrt{ax^2+bx+c} - \frac{b}{2a^{3/2}} \ln \left| 2ax+b + 2\sqrt{a(ax^2+bx+c)} \right|$$

$$\int \frac{dx}{(a^2+x^2)^{3/2}} = \frac{x}{a^2\sqrt{a^2+x^2}}$$

$$\int \sin ax \, dx = -\frac{1}{a} \cos ax$$

$$\int \sin^2 ax \, dx = \frac{x}{2} - \frac{\sin 2ax}{4a}$$

$$\int \sin^3 ax \ dx = -\frac{3 \cos ax}{4a} + \frac{\cos 3ax}{12a}$$

$$\int \cos ax \ dx = \frac{1}{a} \sin ax$$

$$\int \cos^2 ax \ dx = \frac{x}{2} + \frac{\sin 2ax}{4a}$$

$$\int \cos^3 ax dx = \frac{3 \sin ax}{4a} + \frac{\sin 3ax}{12a}$$

$$\int \cos x \sin x \ dx = \frac{1}{2} \sin^2 x + c_1 = -\frac{1}{2} \cos^2 x + c_2 = -\frac{1}{4} \cos 2x + c_3$$

$$\int \cos ax \sin bx \ dx = \frac{\cos[(a-b)x]}{2(a-b)} - \frac{\cos[(a+b)x]}{2(a+b)}, a \neq b$$

$$\int \sin^2 ax \cos bx \ dx = -\frac{\sin[(2a-b)x]}{4(2a-b)} + \frac{\sin bx}{2b} - \frac{\sin[(2a+b)x]}{4(2a+b)}$$

$$\int \sin^2 x \cos x \ dx = \frac{1}{3} \sin^3 x$$

$$\int \cos^2 ax \sin bx \ dx = \frac{\cos[(2a-b)x]}{4(2a-b)} - \frac{\cos bx}{2b} - \frac{\cos[(2a+b)x]}{4(2a+b)}$$

$$\int \cos^2 ax \sin ax \ dx = -\frac{1}{3a} \cos^3 ax$$

$$\int \sin^2 ax \cos^2 bx dx =$$
$$\frac{x}{4} - \frac{\sin 2ax}{8a} - \frac{\sin[2(a-b)x]}{16(a-b)} + \frac{\sin 2bx}{8b} - \frac{\sin[2(a+b)x]}{16(a+b)}$$

$$\int \sin^2 ax \cos^2 ax \ dx = \frac{x}{8} - \frac{\sin 4ax}{32a}$$

$$\int \tan ax \ dx = -\frac{1}{a} \ln \cos ax$$

$$\int \tan^2 ax \ dx = -x + \frac{1}{a} \tan ax$$

$$\int \tan^3 ax dx = \frac{1}{a} \ln \cos ax + \frac{1}{2a} \sec^2 ax$$

$$\int \sec x \ dx = \ln|\sec x + \tan x| = 2 \tanh^{-1}\left(\tan \frac{x}{2}\right)$$

$$\int \sec^2 ax \ dx = \frac{1}{a} \tan ax$$

$$\int \sec^3 x \ dx = \frac{1}{2} \sec x \tan x + \frac{1}{2} \ln|\sec x + \tan x|$$

$$\int \sec x \tan x \ dx = \sec x$$

$$\int \sec^2 x \tan x \ dx = \frac{1}{2} \sec^2 x$$

$$\int \sec^n x \tan x \ dx = \frac{1}{n} \sec^n x, n \neq 0$$

$$\int \csc x \ dx = \ln\left|\tan \frac{x}{2}\right| = \ln|\csc x - \cot x| + C$$

$$\int \csc^2 ax \ dx = -\frac{1}{a} \cot ax$$

$$\int \csc^3 x \ dx = -\frac{1}{2} \cot x \csc x + \frac{1}{2} \ln|\csc x - \cot x|$$

$$\int \csc^n x \cot x \ dx = -\frac{1}{n} \csc^n x, n \neq 0$$

$$\int \sec x \csc x \ dx = \ln|\tan x|$$

$$\int x \cos x \ dx = \cos x + x \sin x$$

$$\int x \cos ax \ dx = \frac{1}{a^2} \cos ax + \frac{x}{a} \sin ax$$

$$\int x^2 \cos x \ dx = 2x \cos x + \left(x^2 - 2\right) \sin x$$

$$\int x^2 \cos ax \ dx = \frac{2x \cos ax}{a^2} + \frac{a^2 x^2 - 2}{a^3} \sin ax$$

$$\int x \sin x \ dx = -x \cos x + \sin x$$

$$\int x \sin ax \ dx = -\frac{x \cos ax}{a} + \frac{\sin ax}{a^2}$$

$$\int x^2 \sin x \ dx = \left(2 - x^2\right) \cos x + 2x \sin x$$

$$\int x^2 \sin ax \ dx = \frac{2 - a^2 x^2}{a^3} \cos ax + \frac{2x \sin ax}{a^2}$$

$$\int x \cos^2 x \ dx = \frac{x^2}{4} + \frac{1}{8} \cos 2x + \frac{1}{4} x \sin 2x$$

$$\int x \sin^2 x \ dx = \frac{x^2}{4} - \frac{1}{8} \cos 2x - \frac{1}{4} x \sin 2x$$