

ICPC Template

墨染空

2024 年 9 月 20 日

目录

1	Default	3
2	图论	4
2.1	Hall 定理	4
2.2	矩阵树定理	4
2.3	欧拉回路计数	4
2.4	圆方树	4
2.5	有向图 Tarjan	5
2.6	欧拉回路	6
2.7	最大流	6
2.8	Prufer	8
2.9	长链剖分	9
2.10	最小费用最大流	10
2.11	KM	11
2.12	有负圈/上下界费用流	13
2.13	虚树	15
2.14	重链剖分 + LCA	16
2.15	匈牙利	17
2.16	点分治	17
3	Poly 多项式	18
3.1	1e18 多项式乘法	18
3.2	正常多项式 + 线性递推	20

4	字符串	23
4.1	AC 自动机	23
4.2	KMP	23
4.3	Manacher	24
4.4	SA	25
4.5	哈希 Hash	26
4.6	最小表示法	26
4.7	Z 函数	26
4.8	SAM	27
4.9	广义 SAM	28
4.10	回文自动机	29
5	数学	30
5.1	单位根反演	30
5.2	积分表	30
5.3	扩域	42
5.4	原根	43
5.5	$O(n)$ 预处理逆元	45
5.6	Exgcd 扩展欧里几得	45
5.7	扩展中国剩余定理 exCRT	46
5.8	BSGS	47
5.9	杜教筛	48
5.10	Min25	49
5.11	FMT / FWT	51
5.12	子集卷积	52
6	数据结构	52
6.1	ST 表	52
6.2	Fhq Treap	53
6.3	线段树	54
6.4	主席树	56
6.5	树状数组: 区间加区间求和	56
6.6	LCT	57
6.7	左偏树	59

6.8	李超树	60
6.9	回滚莫队	61
6.10	动态凸包	62
6.11	珂朵莉树	64
6.12	HashMap	65
6.13	全局平衡二叉树	65
7	计算几何	68
7.1	Basic	68
7.2	点到线段距离	69
7.3	线段交	69
7.4	凸包	70
7.5	半平面交	71
7.6	最小圆覆盖	71
7.7	自适应辛普森积分	72
7.8	极角排序	72
7.9	Int 下凸包 + 闵可夫斯基和	73

1 Default

```

1 // Skyqwq
2 #include <bits/stdc++.h>
3
4 #define pb push_back
5 #define fi first
6 #define se second
7 #define mp make_pair
8
9 using namespace std;
10
11 typedef pair<int, int> PII;
12 typedef long long LL;
13
14 template <typename T> bool chkMax(T &x, T y) { return (y > x) ? x = y, 1 : 0; }
15 template <typename T> bool chkMin(T &x, T y) { return (y < x) ? x = y, 1 : 0; }
16

```

```

17 template <typename T> void inline read(T &x) {
18     int f = 1; x = 0; char s = getchar();
19     while (s < '0' || s > '9') { if (s == '-') f = -1; s = getchar(); }
20     while (s <= '9' && s >= '0') x = x * 10 + (s ^ 48), s = getchar();
21 }
22
23 int main() {
24
25     return 0;
26 }

```

2 图论

2.1 Hall 定理

完美匹配：集合 \leq 邻域的并

最大匹配：总数 - $\max(\text{集合} - \text{邻域并})$

2.2 矩阵树定理

度数 - 边

2.3 欧拉回路计数

Best 定理

内向生成树个数 $\prod (out_i - 1)$

2.4 圆方树

```

1 // 圆方树
2 int dfn[N], low[N], dfncnt, cnt;
3
4 int s[N], top;
5
6 void inline Add(int x, int y) {
7     g[x].pb(y), g[y].pb(x);
8 }

```

```

9
10 void tarjan(int u, int fa) {
11     dfn[u] = low[u] = ++dfncnt;
12     s[++top] = u;
13     for (int v: e[u]) {
14         if (v == fa) continue;
15         if (!dfn[v]) {
16             tarjan(v, u);
17             chkMin(low[u], low[v]);
18             if (low[v] >= dfn[u]) {
19                 int y; ++cnt;
20                 do {
21                     y = s[top--], Add(y, cnt);
22                 } while (y != v);
23                 Add(cnt, u);
24             }
25         } else {
26             chkMin(low[u], dfn[v]);
27         }
28     }
29 }

```

2.5 有向图 Tarjan

```

1
2
3 // 有向图 tarjan
4 void tarjan(int u) {
5     dfn[u] = low[u] = ++dfncnt;
6     s[++top] = u, ins[u] = true;
7     for (int i = head[u]; i; i = e[i].next) {
8         int v = e[i].v;
9         if (!dfn[v]) {
10             tarjan(v), low[u] = min(low[u], low[v]);
11         } else if (ins[v]) low[u] = min(low[u], dfn[v]);
12     }
13     if (low[u] == dfn[u]) {
14         int v; ++cnt;

```

```

15         do {
16             v = s[top--], ins[v] = false, col[v] = cnt;
17         } while (v != u);
18     }
19 }

```

2.6 欧拉回路

```

1 // 欧拉回路
2 void dfs(int u) {
3     for (int &i = head[u]; i; ) {
4         int v = e[i].v;
5         if(vis[i]) {
6             i = e[i].next;
7             continue;
8         }
9
10        vis[i] = true;
11        if(t == 1) vis[i ^ 1] = true;
12
13        i = e[i].next;
14        dfs(v);
15    }
16 }

```

2.7 最大流

```

1 // 最大流
2 namespace MF{
3     int n, m, s, t, pre[N], cur[N], q[N];
4     LL res, maxflow, d[N];
5     int head[N], numE = 1;
6     struct E{
7         int next, v, w;
8     } e[M << 1];
9
10    void inline add(int u, int v, int w) {
11        e[++numE] = (E) { head[u], v, w };

```

```

12     head[u] = numE;
13 }
14 void inline init(int v, int a, int b) {
15     for (int i = 1; i <= n; i++) head[i] = 0;
16     numE = 1;
17     n = v, s = a, t = b;
18 }
19
20 bool inline bfs() {
21     int hh = 0, tt = -1;
22     for (int i = 1; i <= n; i++) d[i] = 0;
23     q[++tt] = s, d[s] = 1, cur[s] = head[s];
24     while (hh <= tt) {
25         int u = q[hh++];
26         for (int i = head[u]; i; i = e[i].next) {
27             int v = e[i].v;
28             if (!d[v] && e[i].w) {
29                 cur[v] = head[v];
30                 q[++tt] = v, d[v] = d[u] + 1;
31                 if (v == t) return 1;
32             }
33         }
34     }
35     return 0;
36 }
37 LL dinic(int u, LL flow) {
38     if (u == t) return flow;
39     LL rest = flow;
40     for (int i = cur[u]; i && rest; i = e[i].next) {
41         cur[u] = i;
42         int v = e[i].v;
43         if (e[i].w && d[v] == d[u] + 1) {
44             int k = dinic(v, min((LL)e[i].w, rest));
45             if (!k) d[v] = 0;
46             rest -= k, e[i].w -= k, e[i ^ 1].w += k;
47         }
48     }
49     return flow - rest;
50 }

```

```

51 void inline addE(int u, int v, int w) {
52     add(u, v, w), add(v, u, 0);
53 }
54 LL inline work() {
55     maxflow = 0;
56     while (bfs())
57         while (res = dinic(s, INF)) maxflow += res;
58     return maxflow;
59 }
60 // Find min-cut
61 bool vis[N];
62
63 void dfs(int u) {
64     //cerr << u << " dfs\n";
65     vis[u] = 1;
66     for (int i = head[u]; i; i = e[i].next) {
67         int v = e[i].v;
68         if (!vis[v] && e[i].w) dfs(v);
69     }
70 }
71
72 void minCut() {
73     for (int i = 1; i <= n; i++) vis[i] = 0;
74     dfs(s);
75 }
76 }

```

2.8 Prufer

```

1 void inline fToP() {
2     for (int i = 1; i < n; i++) d[f[i]]++;
3     for (int i = 1, j = 1; i <= n - 2; j++) {
4         while (d[j]) j++;
5         p[i++] = f[j];
6         while (i <= n - 2 && --d[p[i - 1]] == 0 && p[i - 1] < j) p[i++] = f[p[i - 1]];
7     }
8 }
9

```



```

10 void inline pToF() {
11     for (int i = 1; i <= n - 2; i++) d[p[i]]++;
12     p[n - 1] = n;
13     for (int i = 1, j = 1; i < n; i++, j++) {
14         while (d[j]) j++;
15         f[j] = p[i];
16         while (i < n - 1 && --d[p[i]] == 0 && p[i] < j) f[p[i]] = p[i + 1], ++i;
17     }
18 }

```

2.9 长链剖分

```

1 int d[N], dep[N];
2 int g[N], son[N], fa[N][L], top[N];
3 LL res;
4 vector<int> U[N], D[N];
5 void dfs1(int u) {
6     dep[u] = d[u] = d[fa[u][0]] + 1;
7     for (int i = 1; fa[u][i - 1]; i++) fa[u][i] = fa[fa[u][i - 1]][i - 1];
8     for (int i = head[u]; i; i = e[i].next) {
9         int v = e[i].v;
10        dfs1(v);
11        if (dep[v] > dep[u]) dep[u] = dep[v], son[u] = v;
12    }
13 }
14
15 void dfs2(int u, int tp) {
16     top[u] = tp;
17     if (u == tp) {
18         for (int x = u, i = 0; i <= dep[u] - d[u]; i++)
19             U[u].push_back(x), x = fa[x][0];
20         for (int x = u, i = 0; i <= dep[u] - d[u]; i++)
21             D[u].push_back(x), x = son[x];
22     }
23     if (son[u]) dfs2(son[u], tp);
24     for (int i = head[u]; i; i = e[i].next) {
25         int v = e[i].v;
26         if (v != son[u]) dfs2(v, v);

```

```

27     }
28 }
29
30 int inline query(int x, int k) {
31     if (!k) return x;
32     x = fa[x][g[k]], k -= (1 << g[k]) + d[x] - d[top[x]], x = top[x];
33     return k < 0 ? D[x][-k] : U[x][k];
34 }

```

2.10 最小费用最大流

```

1  const int N = ?, M = ?;
2  const int INF = 0x3f3f3f3f;
3  int n, m, s, t, maxflow, cost, d[N], incf[N], pre[N];
4  int q[N];
5  int head[N], numE = 1;
6
7  bool vis[N];
8
9  struct E{
10     int next, v, w, c;
11 } e[M];
12
13 void inline add(int u, int v, int w, int c) {
14     e[++numE] = (E) { head[u], v, w, c };
15     head[u] = numE;
16 }
17
18 // Spfa ||
19 bool spfa() {
20     memset(vis, false, sizeof vis);
21     memset(d, 0x3f, sizeof d);
22     int hh = 0, tt = 1;
23     q[0] = s; d[s] = 0; incf[s] = 2e9;
24     while (hh != tt) {
25         int u = q[hh++]; vis[u] = false;
26         if (hh == N) hh = 0;
27         for (int i = head[u]; i; i = e[i].next) {

```

```

28         int v = e[i].v;
29         if (e[i].w && d[u] + e[i].c < d[v]) {
30             d[v] = d[u] + e[i].c;
31             pre[v] = i;
32             incf[v] = min(incf[u], e[i].w);
33             if (!vis[v]) {
34                 q[tt++] = v;
35                 vis[v] = true;
36                 if (tt == N) tt = 0;
37             }
38         }
39     }
40 }
41 return d[t] != INF;
42 }
43
44 void update() {
45     int x = t;
46     while (x != s) {
47         int i = pre[x];
48         e[i].w -= incf[t], e[i ^ 1].w += incf[t];
49         x = e[i ^ 1].v;
50     }
51     maxflow += incf[t];
52     cost += d[t] * incf[t];
53 }

```

2.11 KM

```

1 namespace KM{
2     int n, va[N], vb[N], match[N], last[N];
3     LL a[N], b[N], upd[N], w[N][N];
4     bool dfs(int u, int fa) {
5         va[u] = 1;
6         for (int v = 1; v <= n; v++) {
7             if (vb[v]) continue;
8             if (a[u] + b[v] == w[u][v]) {
9                 vb[v] = 1, last[v] = fa;

```

```

10         if (!match[v] || dfs(match[v], v)) {
11             match[v] = u; return true;
12         }
13     } else if (a[u] + b[v] - w[u][v] < upd[v])
14         upd[v] = a[u] + b[v] - w[u][v], last[v] = fa;
15     }
16     return false;
17 }
18 void inline calc(int len, LL d[N][N]) {
19     n = len;
20     for (int i = 1; i <= n; i++)
21         for (int j = 1; j <= n; j++) w[i][j] = d[i][j];
22     for (int i = 1; i <= n; i++) {
23         a[i] = -1e18, b[i] = 0;
24         for (int j = 1; j <= n; j++)
25             a[i] = max(a[i], w[i][j]);
26     }
27     for (int i = 1; i <= n; i++) {
28         memset(va, 0, sizeof va);
29         memset(vb, 0, sizeof vb);
30         memset(upd, 0x3f, sizeof upd);
31         int st = 0; match[0] = i;
32         while (match[st]) {
33             LL delta = 1e18;
34             if (dfs(match[st], st)) break;
35             for (int j = 1; j <= n; j++) {
36                 if (!vb[j] && upd[j] < delta)
37                     delta = upd[j], st = j;
38             }
39             for (int j = 1; j <= n; j++) {
40                 if (va[j]) a[j] -= delta;
41                 if (vb[j]) b[j] += delta;
42                 else upd[j] -= delta;
43             }
44             vb[st] = true;
45         }
46         while (st) {
47             match[st] = match[last[st]];
48             st = last[st];

```

```

49         }
50     }
51 }
52 }

```

2.12 有负圈/上下界费用流

```

1 // 有负圈 / 上下界
2 struct MCMF2{
3     const int N = 205, M = 10005;
4     const int INF = 0x3f3f3f3f;
5     int n, m, s, t, maxflow, cost, d[N], incf[N], pre[N];
6     int q[N], in, S, T;
7     int head[N], a[N], numE = 1, a0, a1;
8     bool vis[N];
9     struct E{
10         int next, v, w, c;
11     } e[M << 2];
12     void inline add(int u, int v, int w, int c) {
13         e[++numE] = (E) { head[u], v, w, c };
14         head[u] = numE;
15     }
16     void inline addE(int u, int v, int w, int c) {
17         add(u, v, w, c), add(v, u, 0, -c);
18     }
19     bool spfa() {
20         memset(vis, false, sizeof vis);
21         memset(d, 0x3f, sizeof d);
22         int hh = 0, tt = 1;
23         q[0] = S; d[S] = 0; incf[S] = 2e9;
24         while (hh != tt) {
25             int u = q[hh++]; vis[u] = false;
26             if (hh == N) hh = 0;
27             for (int i = head[u]; i; i = e[i].next) {
28                 int v = e[i].v;
29                 if (e[i].w && d[u] + e[i].c < d[v]) {
30                     d[v] = d[u] + e[i].c;
31                     pre[v] = i;

```

```

32         incf[v] = min(incf[u], e[i].w);
33         if (!vis[v]) {
34             q[tt++] = v;
35             vis[v] = true;
36             if (tt == N) tt = 0;
37         }
38     }
39 }
40 }
41 return d[T] != INF;
42 }
43 void update() {
44     int x = T;
45     while (x != S) {
46         int i = pre[x];
47         e[i].w -= incf[T], e[i ^ 1].w += incf[T];
48         x = e[i ^ 1].v;
49     }
50     maxflow += incf[T];
51     cost += d[T] * incf[T];
52 }
53
54 void inline addEdge(int u, int v, int l, int d, int c) {
55     a[v] += l, a[u] -= l;
56     addE(u, v, d - l, c);
57 }
58
59 void inline work() {
60     while (spfa()) update();
61 }
62
63 void inline ADD(int u, int v, int w, int c) {
64     if (c >= 0) addEdge(u, v, 0, w, c);
65     else a[v] += w, a[u] -= w, addEdge(v, u, 0, w, -c), a1 += c * w;
66 }
67
68 void inline solve() {
69     for (int i = 1; i <= n; i++) {
70         if (!a[i]) continue;

```

```

71         if (a[i] > 0) addEdge(S, i, 0, a[i], 0);
72         else addEdge(i, T, 0, -a[i], 0);
73     }
74     addEdge(T, S, 0, INF, 0);
75     work();
76     S = s, T = t;
77     a1 += cost;
78     maxflow = cost = 0;
79     e[numE].w = e[numE - 1].w = 0;
80     work();
81     a0 += maxflow, a1 += cost;
82 }
83 }

```

2.13 虚树

```

1 void insert(int x) {
2     if (!top) { s[++top] = x; return; }
3     int p = lca(x, s[top]);
4     while (top > 1 && dep[s[top - 1]] >= dep[p]) e[s[top - 1]].pb(s[top]), top--;
5     if (s[top] != p) {
6         e[p].pb(s[top]);
7         s[top] = p;
8     }
9     s[++top] = x;
10 }
11
12
13 bool inline cmp(int x, int y) {
14     return dfn[x] < dfn[y];
15 }
16 int inline build(vector<int> &A) {
17     top = 0;
18     sort(A.begin(), A.end(), cmp);
19     for (int x: A) {
20         insert(x);
21     }
22     for (int i = 1; i < top; i++)

```

```

23         e[s[i]].pb(s[i + 1]);
24     return s[1];
25 }

```

2.14 重链剖分 + LCA

```

1  int sz[SZ], fa[SZ], dep[SZ], top[SZ], hson[SZ];
2
3  void dfs1(int u) {
4      sz[u] = 1;
5      for (int i = head[u]; i; i = e[i].next) {
6          int v = e[i].v;
7          if (v == fa[u]) continue;
8          dep[v] = dep[u] + 1, fa[v] = u;
9          dfs1(v);
10         sz[u] += sz[v];
11         if (sz[v] > sz[hson[u]]) hson[u] = v;
12     }
13 }
14
15 void dfs2(int u, int tp) {
16     top[u] = tp;
17     if (hson[u]) dfs2(hson[u], tp);
18     for (int i = head[u]; i; i = e[i].next) {
19         int v = e[i].v;
20         if (v == fa[u] || v == hson[u]) continue;
21         dfs2(v, v);
22     }
23 }
24
25 int lca(int x, int y) {
26     while (top[x] != top[y]) {
27         if (dep[top[x]] < dep[top[y]]) swap(x, y);
28         x = fa[top[x]];
29     }
30     if (dep[x] < dep[y]) swap(x, y);
31     return y;
32 }

```


2.15 匈牙利

```
1 int match[N];
2 bool vis[N];
3
4 bool find(int u) {
5     for (int i = head[u]; i; i = e[i].next) {
6         int v = e[i].v;
7         if (vis[v]) continue;
8         vis[v] = true;
9         if (!match[v] || find(match[v])) {
10             match[v] = u; return true;
11         }
12     }
13     return false;
14 }
```

2.16 点分治

```
1 int val;
2
3 void findRoot(int u, int last, int &rt) {
4     sz[u] = 1; int s = 0;
5     for (int i = head[u]; i; i = e[i].next) {
6         int v = e[i].v;
7         if (st[v] || v == last) continue;
8         findRoot(v, u, rt);
9         sz[u] += sz[v], s = max(s, sz[v]);
10    }
11    s = max(s, S - sz[u]);
12    if (s < val) val = s, rt = u;
13 }
14
15 void solve(int u) {
16     if (st[u]) return;
17     val = INF, findRoot(u, 0, u), st[u] = true;
18     for (int i = head[u], j = 0; i; i = e[i].next) {
19         int v = e[i].v;
20         if (st[v]) continue;
```

```

21         // Do sth
22     }
23     for (int i = head[u]; i; i = e[i].next) S = sz[e[i].v], solve(e[i].v);
24 }
25
26 S = n, solve(1);

```

3 Poly 多项式

3.1 1e18 多项式乘法

```

1 // 1e18 多项式乘法》。。。别用fft (mtt也不会写
2
3 #define I __int128_t
4 typedef vector<I> Poly;
5 const I P = 194555503902405427311, G = 5;
6 // p=1945555039024054273=27\times 2^{\{56\}+1},g=5
7
8
9 I A[N], rev[N];
10 I lim = 1, len = 0;
11 LL W[19][N];
12
13 I inline power(I a, I b, I Mod = P) {
14     I res = 1;
15     while (b) {
16         if (b & 1) res = res * a % Mod;
17         a = a * a % Mod;
18         b >>= 1;
19     }
20     return res;
21 }
22
23
24 void inline NTT(I c[], int lim, int o) {
25     for (int i = 0; i < lim; i++)
26         if (i < rev[i]) swap(c[i], c[rev[i]]);
27     for (int k = 1, t = 0; k < lim; k <= 1, t++) {

```

```

28         for (int i = 0; i < lim; i += (k << 1)) {
29             for (int j = 0; j < k; j++) {
30                 I u = c[i + j], v = (I)c[i + k + j] * W[t][j] % P;
31                 c[i + j] = u + v >= P ? u + v - P : u + v;
32                 c[i + j + k] = u - v < 0 ? u - v + P : u - v;
33             }
34         }
35     }
36     if (o == -1) {
37         reverse(c + 1, c + lim);
38         I inv = power(lim, P - 2, P);
39         for (int i = 0; i < lim; i++)
40             c[i] = c[i] * inv % P;
41     }
42 }
43
44 void inline setN(int n) {
45     lim = 1, len = 0;
46     while (lim < n) lim <= 1, len++;
47     for (int i = 0; i < lim; i++)
48         rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (len - 1));
49 }
50
51 Poly inline NTT(Poly a, int o) {
52     int n = a.size();
53     for (int i = 0; i < n; i++) A[i] = a[i];
54     NTT(A, lim, o);
55     a.clear();
56     for (int i = 0; i < lim; i++) a.push_back(A[i]), A[i] = 0;
57     return a;
58 }
59
60 Poly inline mul (Poly a, Poly b, int newn = -1) {
61     if (newn == -1) newn = a.size() + b.size() - 1;
62     setN(a.size() + b.size() - 1);
63     Poly c = NTT(a, 1), d = NTT(b, 1);
64     for (int i = 0; i < lim; i++) c[i] = (I)c[i] * d[i] % P;
65     d = NTT(c, -1); d.resize(newn);
66     return d;

```

```

67 }
68
69 // 用到的最大的 n
70 void inline init(int n) {
71     setN(n);
72     for (int k = 1, t = 0; k < lim; k <= 1, t++) {
73         I wn = power(G, (P - 1) / (k << 1));
74         W[t][0] = 1;
75         for (int j = 1; j < k; j++) W[t][j] = (I)W[t][j - 1] * wn % P;
76     }
77 }
78
79 // --

```

3.2 正常多项式 + 线性递推

```

1 typedef vector<int> Poly;
2
3 #define pb push_back
4
5 const int N = 8e5 + 5, P = 998244353, G = 3;
6
7 int A[N], rev[N], mod, inv[N], fact[N], infact[N];
8 int lim = 1, len = 0, W[20][N];
9
10 int inline power(int a, int b, int Mod = P) {
11     int res = 1;
12     while (b) {
13         if (b & 1) res = (LL)res * a % Mod;
14         a = (LL)a * a % Mod;
15         b >>= 1;
16     }
17     return res;
18 }
19
20 int Gi = power(G, P - 2, P), inv2 = power(2, P - 2, P);
21
22 void inline NTT(int c[], int lim, int o) {

```

```

23     for (int i = 0; i < lim; i++)
24         if (i < rev[i]) swap(c[i], c[rev[i]]);
25     for (int k = 1, t = 0; k < lim; k <= 1, t++) {
26         for (int i = 0; i < lim; i += (k < 1)) {
27             for (int j = 0; j < k; j++) {
28                 int u = c[i + j], v = (LL)c[i + k + j] * W[t][j] % P;
29                 c[i + j] = u + v >= P ? u + v - P : u + v;
30                 c[i + j + k] = u - v < 0 ? u - v + P : u - v;
31             }
32         }
33     }
34     if (o == -1) {
35         reverse(c + 1, c + lim);
36         int inv = power(lim, P - 2, P);
37         for (int i = 0; i < lim; i++)
38             c[i] = (LL)c[i] * inv % P;
39     }
40 }
41
42 void inline setN(int n) {
43     lim = 1, len = 0;
44     while (lim < n) lim <= 1, len++;
45     for (int i = 0; i < lim; i++)
46         rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (len - 1));
47 }
48
49 Poly inline NTT(Poly a, int o) {
50     int n = a.size();
51     for (int i = 0; i < n; i++) A[i] = a[i];
52     NTT(A, lim, o);
53     a.clear();
54     for (int i = 0; i < lim; i++) a.push_back(A[i]), A[i] = 0;
55     return a;
56 }
57
58 Poly inline mul (Poly a, Poly b, int newn = -1) {
59     if (newn == -1) newn = a.size() + b.size() - 1;
60     setN(a.size() + b.size() - 1);
61     Poly c = NTT(a, 1), d = NTT(b, 1);

```

```

62     for (int i = 0; i < lim; i++) c[i] = (LL)c[i] * d[i] % P;
63     d = NTT(c, -1); d.resize(newn);
64     return d;
65 }
66
67 // 用到的最大的 n
68 void inline init(int n) {
69     setN(2 * n);
70     for (int k = 1, t = 0; k < lim; k <= 1, t++) {
71         int wn = power(G, (P - 1) / (k << 1));
72         W[t][0] = 1;
73         for (int j = 1; j < k; j++) W[t][j] = (LL)W[t][j - 1] * wn % P;
74     }
75 }
76
77 // f[0 ... n] 线性递推第 b 项
78 // g[1 ~ k] 为递推多项式
79
80 int inline LRS(int b, Poly f, Poly g) {
81     int k = g.size() - 1;
82     g[0] = 1;
83     for (int i = 1; i <= k; i++) g[i] = (P - g[i]) % P;
84     Poly h = mul(f, g, k);
85     while (b) {
86         Poly g2 = g;
87         for (int i = 0; i < g2.size(); i += 2)
88             g2[i] = (P - g2[i]) % P;
89         Poly t = mul(g2, g); g.clear();
90         for (int i = 0; i < t.size(); i += 2)
91             g.pb(t[i]);
92         t = mul(g2, h); h.clear();
93         for (int i = (b & 1); i < t.size(); i += 2)
94             h.pb(t[i]);
95         b >>= 1;
96     }
97     return (LL)h[0] * power(g[0], P - 2) % P;
98 }

```

4 字符串

4.1 AC 自动机

```
1 struct ACAutomation{
2     int tr[SZ][26], nxt[SZ], idx, q[SZ];
3     void inline insert(char s[]) {
4         int p = 0;
5         for (int j = 0; s[j]; j++) {
6             int ch = s[j] - 'a';
7             if(!tr[p][ch]) tr[p][ch] = ++idx;
8             p = tr[p][ch];
9         }
10    }
11    void build() {
12        int hh = 0, tt = -1;
13        for (int i = 0; i < 26; i++)
14            if (tr[0][i]) q[++tt] = tr[0][i];
15        while (hh <= tt) {
16            int u = q[hh++];
17            for (int i = 0; i < 26; i++) {
18                int v = tr[u][i];
19                if (!v) tr[u][i] = tr[nxt[u]][i];
20                else nxt[v] = tr[nxt[u]][i], q[++tt] = v;
21            }
22        }
23    }
24 }
```

4.2 KMP

```
1 struct KMP{
2     int n, nxt[SZ];
3     void inline build(char s[]) {
4         n = strlen(s + 1);
5         nxt[1] = 0;
6         for (int i = 2, j = 0; i <= n; i++) {
7             while (j && s[j + 1] != s[i]) j = nxt[j];
```

```

8             if (s[j + 1] == s[i]) j++;
9             nxt[i] = j;
10        }
11    }
12    void inline match(char a[], int m) {
13        for (int i = 1, j = 0; i <= m; i++) {
14            while (j && s[j + 1] != a[i]) j = nxt[j];
15            if (s[j + 1] == a[i]) j++;
16            if (j == n) {
17                j = nxt[j];
18            }
19        }
20    }
21 } kmp;

```

4.3 Manacher

```

1 // 中间添加 #
2 char s[N], g[N];
3
4 void change() {
5     n = strlen(s + 1) * 2;
6     g[0] = 0;
7     for (int i = 1; i <= n; i++) {
8         if (i % 2) g[i] = 1;
9         else g[i] = s[i >> 1];
10    }
11    g[++n] = 1, g[n + 1] = 2;
12    manacher();
13 }
14
15 void manacher() {
16     int r = 0, mid = 0;
17     for (int i = 1; i <= n; i++) {
18         p[i] = i <= r ? min(r - i + 1, p[2 * mid - i]) : 1;
19         while (g[i - p[i]] == g[i + p[i]]) ++p[i];
20         if (i + p[i] - 1 > r) mid = i, r = i + p[i] - 1;
21         ans = max(ans, p[i] - 1);

```



```

22     }
23 }

```

4.4 SA

```

1 struct SA{
2     int rk[SZ], sa[SZ], cnt[SZ], oldrk[SZ], id[SZ], n, m, p, height[SZ];
3     bool inline cmp(int i, int j, int k) {
4         return oldrk[i] == oldrk[j] && oldrk[i + k] == oldrk[j + k];
5     }
6     void inline build(char s[]) {
7         n = strlen(s + 1), m = 221;
8         for (int i = 1; i <= n; i++) cnt[rk[i] = s[i]]++;
9         for (int i = 1; i <= m; i++) cnt[i] += cnt[i - 1];
10        for (int i = n; i; i--) sa[cnt[rk[i]]--] = i;
11        for (int w = 1; w < n; w <= 1, m = p) {
12            p = 0;
13            for (int i = n; i > n - w; i--) id[++p] = i;
14            for (int i = 1; i <= n; i++)
15                if (sa[i] > w) id[++p] = sa[i] - w;
16            for (int i = 1; i <= m; i++) cnt[i] = 0;
17            for (int i = 1; i <= n; i++) cnt[rk[i]]++, oldrk[i] = rk[i];
18            for (int i = 1; i <= m; i++) cnt[i] += cnt[i - 1];
19            for (int i = n; i; i--) sa[cnt[rk[id[i]]]--] = id[i];
20            p = 0;
21            for (int i = 1; i <= n; i++) {
22                rk[sa[i]] = cmp(sa[i], sa[i - 1], w) ? p : ++p;
23            }
24            if (p == n) break;
25        }
26        for (int i = 1; i <= n; i++) {
27            int j = sa[rk[i] - 1], k = max(0, height[rk[i] - 1] - 1);
28            while (s[i + k] == s[j + k]) k++;
29            height[rk[i]] = k;
30        }
31    }
32 };

```

4.5 哈希 Hash

```
1 // 哈希
2
3 struct Hash{
4     int b, P, p[N], h[N];
5     int inline get(int l, int r){
6         return (h[r] - (LL)h[l - 1] * p[r - l + 1] % P + P) % P;
7     }
8     void inline build(int n, int tb, int tp) {
9         b = tb, P = tp;
10        p[0] = 1;
11        for(int i = 1; i <= n; i++){
12            p[i] = (LL)p[i - 1] * b % P;
13            h[i] = ((LL)h[i - 1] * b + s[i]) % P;
14        }
15    }
16 }
```

4.6 最小表示法

```
1 // 切记复制一倍到后面，最小表示法，返回开始下标
2 int inline minExp(int a[], int n) {
3     int i = 1, j = 2;
4     while (i <= n && j <= n) {
5         int k;
6         for (k = 0; k < n && a[i + k] == a[j + k]; k++);
7         if (k == n) break;
8         if (a[i + k] < a[j + k]) j += k + 1;
9         else i += k + 1;
10        if (i == j) i++;
11    }
12    return min(i, j);
13 }
```

4.7 Z 函数

```
1 // z 函数
```

```

2 z[1] = n;
3 for (int i = 2, r = 0, j = 0; i <= n; i++) {
4     if (i <= r) z[i] = min(r - i + 1, z[i - j + 1]);
5     while (i + z[i] <= n && a[i + z[i]] == a[1 + z[i]]) z[i]++;
6     if (i + z[i] - 1 > r) r = i + z[i] - 1, j = i;
7 }
8
9 for (int i = 1, r = 0, j = 0; i <= m; i++) {
10     if (i <= r) p[i] = min(r - i + 1, z[i - j + 1]);
11     while (i + p[i] <= m && b[i + p[i]] == a[1 + p[i]]) p[i]++;
12     if (i + p[i] - 1 > r) r = i + p[i] - 1, j = i;
13 }

```

4.8 SAM

```

1 struct SAM{
2     int idx, last;
3     struct SAM_ {
4         int nxt[26], len, link;
5     } t[N];
6     void inline init() {
7         last = idx = 1;
8     }
9
10    void inline extend(int c) {
11        int x = ++idx, p = last; sz[x] = 1;
12        t[x].len = t[p].len + 1;
13        while (p && !t[p].nxt[c])
14            t[p].nxt[c] = x, p = t[p].link;
15        if (!p) t[x].link = 1;
16        else {
17            int q = t[p].nxt[c];
18            if (t[p].len + 1 == t[q].len) t[x].link = q;
19            else {
20                int y = ++idx;
21                t[y] = t[q], t[y].len = t[p].len + 1;
22                while (p && t[p].nxt[c] == q)
23                    t[p].nxt[c] = y, p = t[p].link;

```

```

24             t[q].link = t[x].link = y;
25         }
26     }
27     last = x;
28 }
29 } t;

```

4.9 广义 SAM

```

1 struct GSAM{
2     int idx, last;
3     struct SAM{
4         int ch[26], len, link;
5     } t[N];
6     void inline init() {
7         last = idx = 1;
8     }
9     void inline insert(int c) {
10         int p = last;
11         if (t[p].ch[c]) {
12             int q = t[p].ch[c];
13             if (t[q].len == t[p].len + 1) last = q;
14             else {
15                 int y = ++idx; t[y] = t[q];
16                 t[y].len = t[p].len + 1;
17                 while (p && t[p].ch[c] == q)
18                     t[p].ch[c] = y, p = t[p].link;
19                 t[q].link = y;
20                 last = y;
21             }
22             return;
23         }
24         int x = ++idx; t[x].len = t[p].len + 1;
25         while (p && !t[p].ch[c]) t[p].ch[c] = x, p = t[p].link;
26         int q, y;
27         if (!p) t[x].link = 1;
28         else {
29             q = t[p].ch[c];

```

```

30         if (t[q].len == t[p].len + 1) t[x].link = q;
31     else {
32         int y = ++idx; t[y] = t[q];
33         t[y].len = t[p].len + 1;
34         while (p && t[p].ch[c] == q)
35             t[p].ch[c] = y, p = t[p].link;
36         t[q].link = t[x].link = y;
37         last = y;
38     }
39 }
40 last = x;
41 }
42 } t;

```

4.10 回文自动机

```

1 // 回文自动机
2 struct PAM{
3     int n, ch[SZ][26], fail[SZ], len[SZ], sz[SZ], idx = -1, lastans, last;
4
5     char s[SZ];
6
7     int inline newNode(int x) { len[++idx] = x; return idx; }
8     int inline getFail(int x) {
9         while (s[n - len[x] - 1] != s[n]) x = fail[x];
10        return x;
11    }
12
13    int inline insert(char c) {
14        int k = c - 'a';
15        s[++n] = c;
16        int p = getFail(last), x;
17        if (!ch[p][k]) {
18            x = newNode(len[p] + 2);
19            fail[x] = ch[getFail(fail[p])][k];
20            ch[p][k] = x, sz[x] = 1 + sz[fail[x]];
21        } else x = ch[p][k];
22        last = x;

```

```

23         return sz[x];
24     }
25
26     void inline build() {
27         newNode(0), newNode(-1);
28         s[0] = '$', fail[0] = 1, last = 0;
29     }
30 }

```

5 数学

5.1 单位根反演

$$[n|k] = \frac{1}{n} \sum_{i=1}^{n-1} w_n^{ik}$$

5.2 积分表

Basic Forms

$$\int x^n dx = \frac{1}{n+1} x^{n+1} \quad (1)$$

$$\int \frac{1}{x} dx = \ln |x| \quad (2)$$

$$\int u dv = uv - \int v du \quad (3)$$

$$\int \frac{1}{ax+b} dx = \frac{1}{a} \ln |ax+b| \quad (4)$$

Integrals of Rational Functions

$$\int \frac{1}{(x+a)^2} dx = -\frac{1}{x+a} \quad (5)$$

$$\int (x+a)^n dx = \frac{(x+a)^{n+1}}{n+1}, n \neq -1 \quad (6)$$

$$\int x(x+a)^n dx = \frac{(x+a)^{n+1}((n+1)x-a)}{(n+1)(n+2)} \quad (7)$$

$$\int \frac{1}{1+x^2} dx = \tan^{-1} x \quad (8)$$

$$\int \frac{1}{a^2+x^2} dx = \frac{1}{a} \tan^{-1} \frac{x}{a} \quad (9)$$

$$\int \frac{x}{a^2+x^2} dx = \frac{1}{2} \ln |a^2+x^2| \quad (10)$$

$$\int \frac{x^2}{a^2+x^2} dx = x - a \tan^{-1} \frac{x}{a} \quad (11)$$

$$\int \frac{x^3}{a^2+x^2} dx = \frac{1}{2}x^2 - \frac{1}{2}a^2 \ln |a^2+x^2| \quad (12)$$

$$\int \frac{1}{ax^2+bx+c} dx = \frac{2}{\sqrt{4ac-b^2}} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}} \quad (13)$$

$$\int \frac{1}{(x+a)(x+b)} dx = \frac{1}{b-a} \ln \frac{a+x}{b+x}, a \neq b \quad (14)$$

$$\int \frac{x}{(x+a)^2} dx = \frac{a}{a+x} + \ln |a+x| \quad (15)$$

$$\begin{aligned} \int \frac{x}{ax^2+bx+c} dx &= \frac{1}{2a} \ln |ax^2+bx+c| \\ &\quad - \frac{b}{a\sqrt{4ac-b^2}} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}} \end{aligned} \quad (16)$$

Integrals with Roots

$$\int \sqrt{x-a} dx = \frac{2}{3}(x-a)^{3/2} \quad (17)$$

$$\int \frac{1}{\sqrt{x \pm a}} dx = 2\sqrt{x \pm a} \quad (18)$$

$$\int \frac{1}{\sqrt{a-x}} dx = -2\sqrt{a-x} \quad (19)$$

$$\int x\sqrt{x-a} dx = \frac{2}{3}a(x-a)^{3/2} + \frac{2}{5}(x-a)^{5/2} \quad (20)$$

$$\int \sqrt{ax+b} dx = \left(\frac{2b}{3a} + \frac{2x}{3} \right) \sqrt{ax+b} \quad (21)$$

$$\int (ax+b)^{3/2} dx = \frac{2}{5a}(ax+b)^{5/2} \quad (22)$$

$$\int \frac{x}{\sqrt{x \pm a}} dx = \frac{2}{3}(x \mp 2a)\sqrt{x \pm a} \quad (23)$$

$$\int \sqrt{\frac{x}{a-x}} dx = -\sqrt{x(a-x)} - a \tan^{-1} \frac{\sqrt{x(a-x)}}{x-a} \quad (24)$$

$$\int \sqrt{\frac{x}{a+x}} dx = \sqrt{x(a+x)} - a \ln [\sqrt{x} + \sqrt{x+a}] \quad (25)$$

$$\int x\sqrt{ax+b} dx = \frac{2}{15a^2}(-2b^2 + abx + 3a^2x^2)\sqrt{ax+b} \quad (26)$$

$$\begin{aligned} \int \sqrt{x(ax+b)} dx &= \frac{1}{4a^{3/2}} \left[(2ax+b)\sqrt{ax(ax+b)} \right. \\ &\quad \left. - b^2 \ln \left| a\sqrt{x} + \sqrt{a(ax+b)} \right| \right] \end{aligned} \quad (27)$$

$$\begin{aligned} \int \sqrt{x^3(ax+b)} dx &= \left[\frac{b}{12a} - \frac{b^2}{8a^2x} + \frac{x}{3} \right] \sqrt{x^3(ax+b)} \\ &\quad + \frac{b^3}{8a^{5/2}} \ln \left| a\sqrt{x} + \sqrt{a(ax+b)} \right| \end{aligned} \quad (28)$$

$$\int \sqrt{x^2 \pm a^2} dx = \frac{1}{2} x \sqrt{x^2 \pm a^2} \pm \frac{1}{2} a^2 \ln \left| x + \sqrt{x^2 \pm a^2} \right| \quad (29)$$

$$\int \sqrt{a^2 - x^2} dx = \frac{1}{2} x \sqrt{a^2 - x^2} + \frac{1}{2} a^2 \tan^{-1} \frac{x}{\sqrt{a^2 - x^2}} \quad (30)$$

$$\int x \sqrt{x^2 \pm a^2} dx = \frac{1}{3} (x^2 \pm a^2)^{3/2} \quad (31)$$

$$\int \frac{1}{\sqrt{x^2 \pm a^2}} dx = \ln \left| x + \sqrt{x^2 \pm a^2} \right| \quad (32)$$

$$\int \frac{1}{\sqrt{a^2 - x^2}} dx = \sin^{-1} \frac{x}{a} \quad (33)$$

$$\int \frac{x}{\sqrt{x^2 \pm a^2}} dx = \sqrt{x^2 \pm a^2} \quad (34)$$

$$\int \frac{x}{\sqrt{a^2 - x^2}} dx = -\sqrt{a^2 - x^2} \quad (35)$$

$$\int \frac{x^2}{\sqrt{x^2 \pm a^2}} dx = \frac{1}{2} x \sqrt{x^2 \pm a^2} \mp \frac{1}{2} a^2 \ln \left| x + \sqrt{x^2 \pm a^2} \right| \quad (36)$$

$$\begin{aligned} \int \sqrt{ax^2 + bx + c} dx &= \frac{b + 2ax}{4a} \sqrt{ax^2 + bx + c} \\ &+ \frac{4ac - b^2}{8a^{3/2}} \ln \left| 2ax + b + 2\sqrt{a(ax^2 + bx + c)} \right| \end{aligned} \quad (37)$$

$$\begin{aligned} \int x \sqrt{ax^2 + bx + c} &= \frac{1}{48a^{5/2}} \left(2\sqrt{a} \sqrt{ax^2 + bx + c} \right. \\ &\times (-3b^2 + 2abx + 8a(c + ax^2)) \\ &\left. + 3(b^3 - 4abc) \ln \left| b + 2ax + 2\sqrt{a} \sqrt{ax^2 + bx + c} \right| \right) \end{aligned} \quad (38)$$

$$\int \frac{1}{\sqrt{ax^2 + bx + c}} dx = \frac{1}{\sqrt{a}} \ln \left| 2ax + b + 2\sqrt{a(ax^2 + bx + c)} \right| \quad (39)$$

$$\int \frac{x}{\sqrt{ax^2 + bx + c}} dx = \frac{1}{a} \sqrt{ax^2 + bx + c} - \frac{b}{2a^{3/2}} \ln \left| 2ax + b + 2\sqrt{a(ax^2 + bx + c)} \right| \quad (40)$$

$$\int \frac{dx}{(a^2 + x^2)^{3/2}} = \frac{x}{a^2 \sqrt{a^2 + x^2}} \quad (41)$$

Integrals with Logarithms

$$\int \ln ax dx = x \ln ax - x \quad (42)$$

$$\int \frac{\ln ax}{x} dx = \frac{1}{2} (\ln ax)^2 \quad (43)$$

$$\int \ln(ax + b) dx = \left(x + \frac{b}{a} \right) \ln(ax + b) - x, a \neq 0 \quad (44)$$

$$\int \ln(x^2 + a^2) dx = x \ln(x^2 + a^2) + 2a \tan^{-1} \frac{x}{a} - 2x \quad (45)$$

$$\int \ln(x^2 - a^2) dx = x \ln(x^2 - a^2) + a \ln \frac{x + a}{x - a} - 2x \quad (46)$$

$$\begin{aligned} \int \ln(ax^2 + bx + c) dx &= \frac{1}{a} \sqrt{4ac - b^2} \tan^{-1} \frac{2ax + b}{\sqrt{4ac - b^2}} \\ &- 2x + \left(\frac{b}{2a} + x \right) \ln(ax^2 + bx + c) \end{aligned} \quad (47)$$

$$\begin{aligned} \int x \ln(ax + b) dx &= \frac{bx}{2a} - \frac{1}{4} x^2 \\ &+ \frac{1}{2} \left(x^2 - \frac{b^2}{a^2} \right) \ln(ax + b) \end{aligned} \quad (48)$$

$$\begin{aligned}\int x \ln(a^2 - b^2 x^2) dx &= -\frac{1}{2}x^2 + \\ &\quad \frac{1}{2} \left(x^2 - \frac{a^2}{b^2} \right) \ln(a^2 - b^2 x^2)\end{aligned}\quad (49)$$

Integrals with Exponentials

$$\int e^{ax} dx = \frac{1}{a} e^{ax} \quad (50)$$

$$\begin{aligned}\int \sqrt{x} e^{ax} dx &= \frac{1}{a} \sqrt{x} e^{ax} + \frac{i\sqrt{\pi}}{2a^{3/2}} \operatorname{erf}(i\sqrt{ax}), \\ \text{where } \operatorname{erf}(x) &= \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt\end{aligned}\quad (51)$$

$$\int x e^x dx = (x - 1) e^x \quad (52)$$

$$\int x e^{ax} dx = \left(\frac{x}{a} - \frac{1}{a^2} \right) e^{ax} \quad (53)$$

$$\int x^2 e^x dx = (x^2 - 2x + 2) e^x \quad (54)$$

$$\int x^2 e^{ax} dx = \left(\frac{x^2}{a} - \frac{2x}{a^2} + \frac{2}{a^3} \right) e^{ax} \quad (55)$$

$$\int x^3 e^x dx = (x^3 - 3x^2 + 6x - 6) e^x \quad (56)$$

$$\int x^n e^{ax} dx = \frac{x^n e^{ax}}{a} - \frac{n}{a} \int x^{n-1} e^{ax} dx \quad (57)$$

$$\begin{aligned}\int x^n e^{ax} dx &= \frac{(-1)^n}{a^{n+1}} \Gamma[1 + n, -ax], \\ \text{where } \Gamma(a, x) &= \int_x^\infty t^{a-1} e^{-t} dt\end{aligned}\quad (58)$$

$$\int e^{ax^2} dx = -\frac{i\sqrt{\pi}}{2\sqrt{a}} \operatorname{erf}(ix\sqrt{a}) \quad (59)$$

$$\int e^{-ax^2} dx = \frac{\sqrt{\pi}}{2\sqrt{a}} \operatorname{erf}(x\sqrt{a}) \quad (60)$$

$$\int x e^{-ax^2} dx = -\frac{1}{2a} e^{-ax^2} \quad (61)$$

$$\int x^2 e^{-ax^2} dx = \frac{1}{4} \sqrt{\frac{\pi}{a^3}} \operatorname{erf}(x\sqrt{a}) - \frac{x}{2a} e^{-ax^2} \quad (62)$$

Integrals with Trigonometric Functions

$$\int \sin ax dx = -\frac{1}{a} \cos ax \quad (63)$$

$$\int \sin^2 ax dx = \frac{x}{2} - \frac{\sin 2ax}{4a} \quad (64)$$

$$\begin{aligned} \int \sin^n ax dx = \\ -\frac{1}{a} \cos ax {}_2F_1 \left[\frac{1}{2}, \frac{1-n}{2}, \frac{3}{2}, \cos^2 ax \right] \end{aligned} \quad (65)$$

$$\int \sin^3 ax dx = -\frac{3 \cos ax}{4a} + \frac{\cos 3ax}{12a} \quad (66)$$

$$\int \cos ax dx = \frac{1}{a} \sin ax \quad (67)$$

$$\int \cos^2 ax dx = \frac{x}{2} + \frac{\sin 2ax}{4a} \quad (68)$$

$$\begin{aligned} \int \cos^p ax dx = -\frac{1}{a(1+p)} \cos^{1+p} ax \times \\ {}_2F_1 \left[\frac{1+p}{2}, \frac{1}{2}, \frac{3+p}{2}, \cos^2 ax \right] \end{aligned} \quad (69)$$

$$\int \cos^3 ax dx = \frac{3 \sin ax}{4a} + \frac{\sin 3ax}{12a} \quad (70)$$

$$\int \cos ax \sin bxdx = \frac{\cos[(a-b)x]}{2(a-b)} - \frac{\cos[(a+b)x]}{2(a+b)}, a \neq b \quad (71)$$

$$\begin{aligned} \int \sin^2 ax \cos bxdx &= -\frac{\sin[(2a-b)x]}{4(2a-b)} \\ &+ \frac{\sin bx}{2b} - \frac{\sin[(2a+b)x]}{4(2a+b)} \end{aligned} \quad (72)$$

$$\int \sin^2 x \cos x dx = \frac{1}{3} \sin^3 x \quad (73)$$

$$\begin{aligned} \int \cos^2 ax \sin bxdx &= \frac{\cos[(2a-b)x]}{4(2a-b)} - \frac{\cos bx}{2b} \\ &- \frac{\cos[(2a+b)x]}{4(2a+b)} \end{aligned} \quad (74)$$

$$\int \cos^2 ax \sin ax dx = -\frac{1}{3a} \cos^3 ax \quad (75)$$

$$\begin{aligned} \int \sin^2 ax \cos^2 bxdx &= \frac{x}{4} - \frac{\sin 2ax}{8a} - \frac{\sin[2(a-b)x]}{16(a-b)} \\ &+ \frac{\sin 2bx}{8b} - \frac{\sin[2(a+b)x]}{16(a+b)} \end{aligned} \quad (76)$$

$$\int \sin^2 ax \cos^2 ax dx = \frac{x}{8} - \frac{\sin 4ax}{32a} \quad (77)$$

$$\int \tan ax dx = -\frac{1}{a} \ln \cos ax \quad (78)$$

$$\int \tan^2 ax dx = -x + \frac{1}{a} \tan ax \quad (79)$$

$$\int \tan^n ax dx = \frac{\tan^{n+1} ax}{a(1+n)} \times {}_2F_1\left(\frac{n+1}{2}, 1, \frac{n+3}{2}, -\tan^2 ax\right) \quad (80)$$

$$\int \tan^3 ax dx = \frac{1}{a} \ln \cos ax + \frac{1}{2a} \sec^2 ax \quad (81)$$

$$\int \sec x dx = \ln |\sec x + \tan x| = 2 \tanh^{-1} \left(\tan \frac{x}{2} \right) \quad (82)$$

$$\int \sec^2 ax dx = \frac{1}{a} \tan ax \quad (83)$$

$$\int \sec^3 x dx = \frac{1}{2} \sec x \tan x + \frac{1}{2} \ln |\sec x + \tan x| \quad (84)$$

$$\int \sec x \tan x dx = \sec x \quad (85)$$

$$\int \sec^2 x \tan x dx = \frac{1}{2} \sec^2 x \quad (86)$$

$$\int \sec^n x \tan x dx = \frac{1}{n} \sec^n x, n \neq 0 \quad (87)$$

$$\int \csc x dx = \ln \left| \tan \frac{x}{2} \right| = \ln |\csc x - \cot x| + C \quad (88)$$

$$\int \csc^2 ax dx = -\frac{1}{a} \cot ax \quad (89)$$

$$\int \csc^3 x dx = -\frac{1}{2} \cot x \csc x + \frac{1}{2} \ln |\csc x - \cot x| \quad (90)$$

$$\int \csc^n x \cot x dx = -\frac{1}{n} \csc^n x, n \neq 0 \quad (91)$$

$$\int \sec x \csc x dx = \ln |\tan x| \quad (92)$$

Products of Trigonometric Functions and Monomials

$$\int x \cos x dx = \cos x + x \sin x \quad (93)$$

$$\int x \cos ax dx = \frac{1}{a^2} \cos ax + \frac{x}{a} \sin ax \quad (94)$$

$$\int x^2 \cos x dx = 2x \cos x + (x^2 - 2) \sin x \quad (95)$$

$$\int x^2 \cos ax dx = \frac{2x \cos ax}{a^2} + \frac{a^2 x^2 - 2}{a^3} \sin ax \quad (96)$$

$$\begin{aligned} \int x^n \cos x dx = & -\frac{1}{2}(i)^{n+1} [\Gamma(n+1, -ix) \\ & + (-1)^n \Gamma(n+1, ix)] \end{aligned} \quad (97)$$

$$\begin{aligned} \int x^n \cos ax dx = & \frac{1}{2}(ia)^{1-n} [(-1)^n \Gamma(n+1, -iax) \\ & - \Gamma(n+1, ixa)] \end{aligned} \quad (98)$$

$$\int x \sin x dx = -x \cos x + \sin x \quad (99)$$

$$\int x \sin ax dx = -\frac{x \cos ax}{a} + \frac{\sin ax}{a^2} \quad (100)$$

$$\int x^2 \sin x dx = (2 - x^2) \cos x + 2x \sin x \quad (101)$$

$$\int x^2 \sin ax dx = \frac{2 - a^2 x^2}{a^3} \cos ax + \frac{2x \sin ax}{a^2} \quad (102)$$

$$\int x^n \sin x dx = -\frac{1}{2}(i)^n [\Gamma(n+1, -ix) - (-1)^n \Gamma(n+1, ix)] \quad (103)$$

Products of Trigonometric Functions and Exponentials

$$\int e^x \sin x dx = \frac{1}{2} e^x (\sin x - \cos x) \quad (104)$$

$$\int e^{bx} \sin ax dx = \frac{1}{a^2 + b^2} e^{bx} (b \sin ax - a \cos ax) \quad (105)$$

$$\int e^x \cos x dx = \frac{1}{2} e^x (\sin x + \cos x) \quad (106)$$

$$\int e^{bx} \cos ax dx = \frac{1}{a^2 + b^2} e^{bx} (a \sin ax + b \cos ax) \quad (107)$$

$$\int x e^x \sin x dx = \frac{1}{2} e^x (\cos x - x \cos x + x \sin x) \quad (108)$$

$$\int x e^x \cos x dx = \frac{1}{2} e^x (x \cos x - \sin x + x \sin x) \quad (109)$$

Integrals of Hyperbolic Functions

$$\int \cosh ax dx = \frac{1}{a} \sinh ax \quad (110)$$

$$\int e^{ax} \cosh bxdx = \begin{cases} \frac{e^{ax}}{a^2 - b^2} [a \cosh bx - b \sinh bx] & a \neq b \\ \frac{e^{2ax}}{4a} + \frac{x}{2} & a = b \end{cases} \quad (111)$$

$$\int \sinh ax dx = \frac{1}{a} \cosh ax \quad (112)$$

$$\int e^{ax} \sinh bxdx = \begin{cases} \frac{e^{ax}}{a^2 - b^2} [-b \cosh bx + a \sinh bx] & a \neq b \\ \frac{e^{2ax}}{4a} - \frac{x}{2} & a = b \end{cases} \quad (113)$$

$$\int e^{ax} \tanh bx dx = \begin{cases} \frac{e^{(a+2b)x}}{(a+2b)} {}_2F_1 \left[1 + \frac{a}{2b}, 1, 2 + \frac{a}{2b}, -e^{2bx} \right] \\ \quad - \frac{1}{a} e^{ax} {}_2F_1 \left[\frac{a}{2b}, 1, 1E, -e^{2bx} \right] & a \neq b \\ \frac{e^{ax} - 2 \tan^{-1}[e^{ax}]}{a} & a = b \end{cases} \quad (114)$$

$$\int \tanh ax dx = \frac{1}{a} \ln \cosh ax \quad (115)$$

$$\int \cos ax \cosh bxdx = \frac{1}{a^2 + b^2} [a \sin ax \cosh bx + b \cos ax \sinh bx] \quad (116)$$

$$\int \cos ax \sinh bxdx = \frac{1}{a^2 + b^2} [b \cos ax \cosh bx + a \sin ax \sinh bx] \quad (117)$$

$$\int \sin ax \cosh bxdx = \frac{1}{a^2 + b^2} [-a \cos ax \cosh bx + b \sin ax \sinh bx] \quad (118)$$

$$\int \sin ax \sinh bxdx = \frac{1}{a^2 + b^2} [b \cosh bx \sin ax - a \cos ax \sinh bx] \quad (119)$$

$$\int \sinh ax \cosh ax dx = \frac{1}{4a} [-2ax + \sinh 2ax] \quad (120)$$

$$\int \sinh ax \cosh bxdx = \frac{1}{b^2 - a^2} [b \cosh bx \sinh ax - a \cosh ax \sinh bx] \quad (121)$$

5.3 扩域

```
1 // 扩域
2 struct C{
3     int x, y;
4     // x + y * sqrt(o);
5 };
6
7 int o = 2;
8
9 // fn = Aa^n + Bb^n
10
11 int inline power(int a, int b) {
12     int ret = 1;
13     while (b) {
14         if (b & 1) ret = 1ll * ret * a % P;
15         a = 1ll * a * a % P;
16         b >>= 1;
17     }
18     return ret;
19 }
20
21
22
23
24 int mod(int x) {
25     return x >= P ? x - P : x;
26 }
27
28 C operator + (const C &a, const C &b) {
29     return (C) { mod(a.x + b.x), mod(a.y + b.y) };
30 };
31
32 C operator * (const C &a, const C &b) {
33     C c;
34     c.x = (1ll * a.x * b.x + 1ll * a.y * b.y % P * o) % P;
35     c.y = (1ll * a.x * b.y + 1ll * a.y * b.x) % P;
36     return c;
37 };
```

```

38
39 C operator * (const C &a, const int &b) {
40     C c;
41     c.x = 1ll * a.x * b % P;
42     c.y = 1ll * a.y * b % P;
43
44     return c;
45 };
46
47
48 C inline power(C a, int b) {
49     C ret = (C) { 1, 0 } ;
50     while (b) {
51         if (b & 1) ret = ret * a;
52         a = a * a;
53         b >>= 1;
54     }
55     return ret;
56 }
57
58 C operator / (const C &a, const C &b) {
59     C c, d;
60     c = a;
61     d = b;
62     d.y = mod(P - d.y);
63     c = c * d;
64     int I = (((LL)b.x * b.x - (LL)b.y * b.y * o) % P + P) % P;
65     I = power(I, P - 2);
66     c = c * I;
67     return c;
68 };

```

5.4 原根

```

1 // 原根 / 封装不太好
2
3
4 int n, D, phi[N], primes[N], tot, d[N], len;

```

```

5  int ans[N], cnt;
6
7  bool st[N], pr[N];
8
9  void inline init() {
10     phi[1] = 1, pr[2] = pr[4] = true;
11     for (int i = 2; i < N; i++) {
12         if (!st[i]) primes[tot++] = i, phi[i] = i - 1;
13         for (int j = 0; i * primes[j] < N; j++) {
14             st[i * primes[j]] = true;
15             if (i % primes[j] == 0) {
16                 phi[i * primes[j]] = phi[i] * primes[j];
17                 break;
18             }
19             phi[i * primes[j]] = phi[i] * (primes[j] - 1);
20         }
21     }
22     for (int i = 1; i < tot; i++) {
23         for (LL j = primes[i]; j < N; j *= primes[i]) pr[j] = true;
24         for (LL j = 2 * primes[i]; j < N; j *= primes[i]) pr[j] = true;
25     }
26 }
27
28
29 void inline factor(int m) {
30     len = 0;
31     for (int i = 0; i < tot && primes[i] * primes[i] <= m; i++) {
32         int j = primes[i];
33         if (m % j == 0) {
34             d[len++] = j;
35             while (m % j == 0) m /= j;
36         }
37     }
38     if (m > 1) d[len++] = m;
39 }
40
41 int inline power(int a, int b, int P) {
42     int res = 1;
43     while (b) {

```

```

44         if (b & 1) res = (LL)res * a % P;
45         a = (LL)a * a % P;
46         b >>= 1;
47     }
48     return res;
49 }
50
51 bool inline check(int x, int P) {
52     if (power(x, phi[P], P) != 1) return false;
53     for (int i = 0; i < len; i++)
54         if (power(x, phi[P] / d[i], P) == 1) return false;
55     return true;
56 }
57
58 // 输入 P, 返回最小原根
59
60 int inline get(int P) {
61     for (int i = 1; i < P; i++)
62         if (check(i, P)) return i;
63     return 0;
64 }
65 //-

```

5.5 $O(n)$ 预处理逆元

```

1 void inline preInv(int n) {
2     inv[1] = 1;
3     for (int i = 2; i <= n; i++)
4         inv[i] = ((LL)P - P / i) * inv[P % i] % P;
5 }

```

5.6 Exgcd 扩展欧里几得

```

1 LL inline exgcd(LL a, LL b, LL &x, LL &y) {
2     if (b == 0) {
3         x = 1, y = 0;
4         return a;
5     }

```

```

6         LL d = exgcd(b, a % b, y, x);
7         y -= a / b * x;
8         return d;
9     }

```

5.7 扩展中国剩余定理 exCRT

```

1 // 扩展中国剩余定理 exCRT
2 typedef pair<LL, LL> PLL;
3
4 LL gcd(LL a, LL b) {
5     return b ? gcd(b, a % b) : a;
6 }
7
8 LL exgcd(LL a, LL b, LL &x, LL &y) {
9     if (!b) {
10         x = 1, y = 0;
11         return a;
12     }
13     LL d = exgcd(b, a % b, y, x);
14     y -= a / b * x;
15     return d;
16 }
17
18 LL mul(LL x, LL y, LL P) {
19     return (__int128)x * y % P;
20 //     return x * y % P;
21 }
22
23
24 // x mod m = a (m1, a1) (m2, a2) return x
25
26
27 PLL inline merge(PLL A, PLL B) {
28     LL a1 = A.fi, b1 = A.se;
29     LL a2 = B.fi, b2 = B.se;
30     LL a = a1 / gcd(a1, a2) * a2;
31     LL x, y;

```

```

32     LL d = exgcd(a1, a2, x, y);
33     assert((b2 - b1) % d == 0);
34     x = mul(x, (b2 - b1) / d, a);
35     if (x < 0) x += a;
36     LL o = mul(x, a1, a) + b1;
37     if (o >= a) o -= a;
38     PLL c = mp(a, o);
39     return c;
40 }

```

5.8 BSGS

```

1  // BSGS
2
3  unordered_map<int, int> mp;
4
5  int BSGS(int a, int b, int P) {
6      int t = sqrt(P) + 1; mp.clear(); b %= P;
7      for (int j = 0, s = b; j < t; j++)
8          mp[s] = j, s = (LL)s * a % P;
9      a = power(a, t, P);
10     for (int i = 1, s = 1; i <= t; i++) {
11         s = (LL)s * a % P;
12         if (mp.count(s) && i * t - mp[s] >= 0)
13             return i * t - mp[s];
14     }
15     return -1;
16 }
17
18 int exBSGS(int a, int b, int P) {
19     int x, y, d, A = 1, k = 0;
20     while ((d = gcd(a, P)) > 1) {
21         if (b % d) return -1;
22         b /= d, P /= d, k++, A = (LL)A * (a / d) % P;
23         if (A == b) return k;
24     }
25     exgcd(A, P, x, y); x = (x % P + P) % P;
26     int res = BSGS(a, (LL)b * x % P, P);

```

```

27         return res == -1 ? -1 : res + k;
28     }

```

5.9 杜教筛

```

1  const int N = 5000005, S = 3000;
2  const LL INF = 9e18;
3
4  LL p1[N], p2[S], m1[N], m2[S];
5
6  int n, primes[N], tot;
7
8  bool vis[N];
9
10 // 杜教筛 phi
11 LL s1(int x) {
12     if (x < N) return p1[x];
13     else if (p2[n / x] != INF) return p2[n / x];
14     LL res = x * (x + 1) / 2;
15     for (LL l = 2, r; l <= x; l = r + 1) {
16         r = x / (x / l);
17         res -= (r - l + 1) * s1(x / l);
18     }
19     return p2[n / x] = res;
20 }
21
22 // 杜教筛 mu
23
24 LL s2(int x) {
25     if (x < N) return m1[x];
26     else if (m2[n / x] != INF) return m2[n / x];
27     LL res = 1;
28     for (LL l = 2, r; l <= x; l = r + 1) {
29         r = x / (x / l);
30         res -= (r - l + 1) * s2(x / l);
31     }
32     return m2[n / x] = res;
33 }

```


5.10 Min25

```
1 // Min25
2
3 int inv2 = power(2, P - 2), inv6 = power(6, P - 2);
4
5 // 求 g_k 函数: <= x 的和
6 int inline getS(LL x, int k) {
7     if (k == 1) return (x % P * (x % P + 111) % P * inv2 + P - 111)
8     % P;
9     if (k == 2) return (P - 111 + x % P * (x % P + 111) % P * (211 * x % P + 1) % P *
10 }
11
12 int inline getV(LL x, int k) {
13     if (k == 1) return x % P;
14     if (k == 2) return (LL)x % P * x % P;
15 }
16
17 bool vis[M];
18
19 int primes[M], tot;
20
21 void inline linear(int n) {
22     for (int i = 2; i <= n; i++) {
23         if (!vis[i]) primes[++tot] = i;
24         for (int j = 1; primes[j] <= n / i; j++) {
25             vis[i * primes[j]] = true;
26             if (i % primes[j] == 0) break;
27         }
28     }
29 }
30
31 // 预处理 g_k 处所有 n / i 形式的质数前缀和
32 struct MP1{
33     int m, g[M], pos1[M], pos2[M], len, id;
34     LL n, d[M];
35     int inline getPos(LL x) {
36         return x <= m ? pos1[x] : pos2[n / x];
37     }
38 }
```

```

37     void inline add(LL v) {
38         d[++len] = v;
39         g[len] = getS(v, id);
40         if (v <= m) pos1[v] = len;
41         else pos2[n / v] = len;
42     }
43     void build(LL sum, int t) {
44         m = sqrt(n = sum); id = t;
45         for (LL i = 1, j; i <= n; i = j + 1) {
46             LL v = n / i; j = n / v;
47             if (v <= m) break;
48             add(v);
49         }
50         for (int i = m; i; i--) add(i);
51         for (int i = 1; i <= tot && (LL)primes[i] * primes[i] <= n; i++) {
52             LL pr = primes[i];
53             for (int j = 1; j <= len && pr * pr <= d[j]; j++) {
54                 int k = getPos(d[j] / pr);
55                 g[j] = (g[j] - (LL)getV(pr, id) * (g[k] - g[getPos(primes
56                     }
57             }
58         }
59         int inline s(LL x) { return g[getPos(x)]; }
60 } t1, t2;
61
62 int inline get(LL x) {
63     return (t2.s(x) - t1.s(x) + P) % P;
64 }
65
66 int inline calc(LL x) {
67     return x % P * (x % P - 111 + P) % P;
68 }
69
70 void inline add(int &x, int y) {
71     (x += y) %= P;
72 }
73
74 int inline s(LL n, int t) {
75     if (primes[t] >= n) return 0;

```

```

76     int ans = (get(n) - get(primes[t]) + P) % P;
77     for (int i = t + 1; i <= tot && (LL)primes[i] * primes[i] <= n; i++) {
78         int pr = primes[i];
79         LL v = pr;
80         for (int j = 1; v <= n; v = v * pr, j++) {
81             add(ans, (LL)calc(v) * ((j != 1) + s(n / v, i)) % P);
82         }
83     }
84     return ans;
85 }

```

5.11 FMT / FWT

```

1  // FMT / FWT
2
3  void inline OR(int n, int a[], int o) {
4      for (int w = 1; w < n; w <= 1)
5          for (int i = 0; i < n; i += (w << 1))
6              for (int j = 0; j < w; j++)
7                  add(a[i + j + w], o * a[i + j]);
8  }
9
10 void inline AND(int n, int a[], int o) {
11     for (int w = 1; w < n; w <= 1)
12         for (int i = 0; i < n; i += (w << 1))
13             for (int j = 0; j < w; j++)
14                 add(a[i + j], o * a[i + j + w]);
15 }
16
17
18 // 反向传 1/2
19 void inline XOR(int n, int a[], int o) {
20     for (int w = 1; w < n; w <= 1)
21         for (int i = 0; i < n; i += (w << 1))
22             for (int j = 0; j < w; j++) {
23                 int u = a[i + j], v = a[i + j + w];
24                 a[i + j] = ((LL)u + v + P) * o % P;
25                 a[i + j + w] = ((LL)u - v + P) * o % P;

```

```

26         }
27     }

```

5.12 子集卷积

```

1 // 子集卷积
2
3 void inline SubConv(int n, int a[], int b[], int c[]) {
4     for (int i = 0; i < (1 << n); i++) {
5         f[get(i)][i] = a[i];
6         g[get(i)][i] = b[i];
7     }
8     for (int i = 0; i <= n; i++)
9         OR(1 << n, f[i], 1), OR(1 << n, g[i], 1);
10    for (int i = 0; i <= n; i++)
11        for (int j = 0; j <= i; j++)
12            for (int k = 0; k < (1 << n); k++)
13                add(h[i][k], (LL)f[j][k] * g[i - j][k] % P);
14    for (int i = 0; i <= n; i++) OR(1 << n, h[i], -1);
15    for (int i = 0; i < (1 << n); i++) c[i] = h[get(i)][i];
16 }

```

6 数据结构

6.1 ST 表

```

1 // ST 表
2 struct ST{
3     void inline STPrework(int n) {
4         g[0] = -1;
5         for (int i = 1; i <= n; i++)
6             f[i][0] = a[i], g[i] = g[i >> 1] + 1;
7         for (int j = 1; j <= g[n]; j++)
8             for (int i = 1; i + (1 << j) - 1 <= n; i++)
9                 f[i][j] = max(f[i][j - 1], f[i + (1 << (j - 1))][j - 1]);
10    }
11 }

```

```

12         int inline query(int l, int r) {
13             int k = g[r - l + 1];
14             return max(f[l][k], f[r - (1 << k) + 1][k]);
15         }
16     }

```

6.2 Fhq Treap

```

1  // 用来动态开点的池
2  struct T{
3      int l, r, val, rnd, sz;
4  } t[SZ];
5  int idx;
6
7  struct Fhq{
8      int rt;
9      void pushup(int p) {
10
11      }
12      // value(A) < value(B)
13      int merge(int A, int B) {
14          if (!A || !B) return A + B;
15          else if(t[A].rnd > t[B].rnd) {
16              t[A].r = merge(t[A].r, B);
17              pushup(A);
18              return A;
19          } else {
20              t[B].l = merge(A, t[B].l);
21              pushup(B);
22              return B;
23          }
24      }
25
26      // 按值分裂
27      void split(int p, int k, int &x, int &y) {
28          if (!p) x = y = 0;
29          else {
30              if (t[p].val <= k)

```

```

31         x = p, split(t[p].r, k, t[p].r, y);
32         else y = p, split(t[p].l, k, x, t[p].l);
33         pushup(p);
34     }
35 }
36 int getNode(int val) {
37     t[++idx] = (T) { 0, 0, val, rand(), 1 };
38     return idx;
39 }
40
41 void insert(int val) {
42     int x, y;
43     split(rt, val, x, y);
44     rt = merge(merge(x, getNode(val)), y);
45 }
46
47 int get(int l, int r) {
48     int x, y, z;
49     split(rt, l - 1, x, y);
50     split(y, r, y, z);
51     int res = t[y].N;
52     rt = merge(x, merge(y, z));
53     return res;
54 }
55
56 void del(int val) {
57     int x, y, z;
58     split(rt, val - 1, x, y);
59     split(y, val, y, z);
60     y = merge(t[y].l, t[y].r);
61     rt = merge(x, merge(y, z));
62 }
63 }

```

6.3 线段树

```
1 // 普通线段树
```

```
2
```

```

3 struct Seg{
4     #define ls (p << 1)
5     #define rs (p << 1 | 1)
6     void inline pu(int p) {
7
8     }
9
10    void inline pd(int p) {
11
12    }
13
14    void bd(int p, int l, int r) {
15        if(l == r) {
16            return;
17        }
18        int mid = (l + r) >> 1;
19        bd(ls, l, mid);
20        bd(rs, mid + 1, r);
21        pu(p);
22    }
23    void chg(int p, int l, int r, int x, int y, int k, int c) {
24        if(x <= l && r <= y) {
25            return ;
26        }
27        int mid = (l + r) >> 1;
28        pd(p);
29        if(x <= mid) chg(ls, l, mid, x, y, k, c);
30        if(mid + 1 <= y) chg(rs, mid + 1, r, x, y, k, c);
31        pu(p);
32    }
33
34    int qry(int p, int l, int r, int x, int y) {
35        if(x <= l && r <= y) return ?;
36        int mid = (l + r) >> 1, s = 0;
37        pd(p);
38        if(x <= mid) s += qry(ls, l, mid, x, y);
39        if(mid + 1 <= y) s += qry(rs, mid + 1, r, x, y);
40        return s % P;
41    }

```

```
42 }
```

6.4 主席树

```
1 // 主席树
2 struct PersisSeg{
3     struct T{
4         int l, r;
5         LL v;
6     } t[SZ];
7
8     int rt[SZ], idx;
9
10    void inline update(int &p, int q, int l, int r, int x, int k) {
11        t[p = ++idx] = t[q];
12        t[p].v += k;
13        if (l == r) return;
14        int mid = (l + r) >> 1;
15        if (x <= mid) update(t[p].l, t[q].l, l, mid, x, k);
16        else update(t[p].r, t[q].r, mid + 1, r, x, k);
17    }
18
19    LL inline query(int p, int l, int r, int x, int y) {
20        if (!p || x > y) return 0;
21        if (x <= l && r <= y) return t[p].v;
22        int mid = (l + r) >> 1; LL res = 0;
23        if (x <= mid) res += query(t[p].l, l, mid, x, y);
24        if (mid < y) res += query(t[p].r, mid + 1, r, x, y);
25        return res;
26    }
27 }
```

6.5 树状数组：区间加区间求和

```
1 // 区间加 区间查的树状数组
2 struct exBIT{
3     BIT t1, t2;
4     int n;
```



```

5     void inline init(int len, int a[]) {
6         n = len;
7         for (int i = 1; i <= n; i++)
8             b[i] = a[i] - a[i - 1];
9         t1.init(n, b);
10        for (int i = 1; i <= n; i++) b[i] *= i;
11        t2.init(n, b);
12    }
13    void inline add(int l, int r, LL c) {
14        t1.add(l, c), t1.add(r + 1, -c);
15        t2.add(l, c * l), t2.add(r + 1, -c * (r + 1));
16    }
17    LL inline ask(int x) {
18        return (x + 1) * t1.ask(x) - t2.ask(x);
19    }
20    LL inline ask(int x, int y) { return ask(y) - ask(x - 1); }
21 };

```

6.6 LCT

```

1 struct LCT{
2     #define get(x) (ch[fa[x]][1] == x)
3     #define isRoot(x) (ch[fa[x]][0] != x && ch[fa[x]][1] != x)
4     #define ls ch[p][0]
5     #define rs ch[p][1]
6
7     int ch[N][2], fa[N], mx[N], w[N], rev[N];
8
9     void inline pushup(int p) {
10
11    }
12
13    void inline pushdown(int p) {
14        if (rev[p]) { swap(ls, rs), rev[ls] ^= 1, rev[rs] ^= 1, rev[p] = 0; }
15    }
16
17    void inline rotate(int x) {
18        int y = fa[x], z = fa[y], k = get(x);

```

```

19         if (!isRoot(y)) ch[z][get(y)] = x;
20         ch[y][k] = ch[x][!k], fa[ch[y][k]] = y;
21         ch[x][!k] = y, fa[y] = x, fa[x] = z;
22         pushup(y); pushup(x);
23     }
24
25     void inline update(int p) {
26         if (!isRoot(p)) update(fa[p]);
27         pushdown(p);
28     }
29
30     void inline splay(int p) {
31         update(p);
32         for (int f = fa[p]; !isRoot(p); rotate(p), f = fa[p])
33             if (!isRoot(f)) rotate(get(p) == get(f) ? f : p);
34     }
35
36     void inline access(int x) {
37         for (int p = 0; x; p = x, x = fa[x]) {
38             splay(x), ch[x][1] = p, pushup(x);
39         }
40     }
41
42     int inline find(int p) {
43         access(p), splay(p);
44         while (ls) pushdown(p), p = ls;
45         splay(p);
46         return p;
47     }
48
49     void inline makeRoot(int x) {
50         access(x), splay(x), rev[x] ^= 1;
51     }
52
53     void inline split(int x, int y) {
54         makeRoot(x), access(y), splay(y);
55     }
56
57     void inline link(int x, int y) {

```

```

58         makeRoot(x), fa[x] = y;
59     }
60
61     void inline cut(int x, int y) {
62         split(x, y);
63         ch[y][0] = 0, fa[x] = 0;
64         pushup(y);
65     }
66
67 }

```

6.7 左偏树

```

1 // 左偏树
2 struct LeftistTree{
3     struct T{
4         int l, r, v, d, f;
5         // l, r 表示左右儿子, v 表示值
6         // d 表示从当前节点到最近叶子节点的距离, f 表示当前节点的父亲
7     } t[SZ];
8
9     int find(int x) {
10         return t[x].f == x ? x : t[x].f = find(t[x].f);
11     }
12
13     int merge(int x, int y) { // 递归合并函数
14         if (!x || !y) return x + y;
15         if (t[x].v > t[y].v || (t[x].v == t[y].v && x > y)) swap(x, y);
16         rs = merge(rs, y);
17         if (t[ls].d < t[rs].d) swap(ls, rs);
18         t[x].d = t[rs].d + 1;
19         return x;
20     }
21
22     int work(int x, int y) { // 合并 x, y 两个堆。
23         if (x == y) return 0;
24         if (!x || !y) return t[x + y].f = x + y;
25         if (t[x].v > t[y].v || (t[x].v == t[y].v && x > y)) swap(x, y);

```

```

26         t[x].f = t[y].f = x;
27         merge(x, y); return x;
28     }
29
30     void del(int x) {
31         t[x].f = work(ls, rs), t[x].v = -1;
32     }
33 }

```

6.8 李超树

```

1  // 李超树
2
3  struct LC{
4      struct Tree{
5          int l, r;
6          Line v;
7      } t[N << 2];
8      LL inline calc(Line e, LL x) {
9          return e.k * x + e.b;
10     }
11     int idx, rt;
12     void inline clr() {
13         idx = 0; rt = 0;
14     }
15     // 这里写法非常简洁的原因是，让计算机人工帮你判断了单调 / 需要 upd 的位置，事实上
16     void inline ins(int &p, int l, int r, Line e) {
17         if (!p) {
18             t[p = ++idx] = (Tree) { 0, 0, e };
19             return;
20         }
21         int mid = (l + r) >> 1;
22         if (calc(t[p].v, mid) > calc(e, mid)) swap(e, t[p].v);
23         if (calc(e, l) < calc(t[p].v, l)) ins(t[p].l, l, mid, e);
24         if (calc(e, r) < calc(t[p].v, r)) ins(t[p].r, mid + 1, r, e);
25     }
26     LL ask(int p, int l, int r, int x) {
27         if (!p) return INF;

```

```

28         if (l == r) return calc(t[p].v, x);
29         int mid = (l + r) >> 1; LL ret = calc(t[p].v, x);
30         if (x <= mid) chkMin(ret, ask(t[p].l, l, mid, x));
31         else chkMin(ret, ask(t[p].r, mid + 1, r, x));
32         return ret;
33     }
34
35 } ;

```

6.9 回滚莫队

```

1  // 莫队
2
3  int pos[N], L[N], R[N], t;
4
5  struct Q {
6      int l, r, id;
7      bool operator < (const Q &b) const {
8          if (pos[l] != pos[b.l]) return pos[l] < pos[b.l];
9          return r < b.r;
10     }
11 } q[N];
12
13 t = sqrt(n);
14 for (int i = 1; i <= n; i++) {
15     pos[i] = (i - 1) / t + 1;
16     if (!L[pos[i]]) L[pos[i]] = i;
17     R[pos[i]] = i;
18 }
19
20 sort(q + 1, q + 1 + m);
21
22 // 回滚
23
24 int l = 1, r = 0, last = -1;
25 for (int i = 1; i <= m; i++) {
26     if (pos[q[i].l] == pos[q[i].r]) {
27         // 块内暴力

```

```

28         continue;
29     }
30     if (pos[q[i].l] != last) {
31         // 新的左块
32         res = 0, top = 0, r = R[pos[q[i].l]], l = r + 1;
33         last = pos[q[i].l];
34     }
35     while (r < q[i].r) {
36         ++r;
37         // insert r
38     }
39     int bl = l, tp = res; // 记录
40     while (l > q[i].l) {
41         --l;
42         // insert l
43     }
44     // 恢复
45     ans[q[i].id] = res; res = tp;
46 }

```

6.10 动态凸包

```

1 typedef pair<LL, LL> PII;
2 typedef set<PII>::iterator SIT;
3 typedef set<PII> SI;
4
5 PII operator - (const PII &a, const PII &b) {
6     return mp(a.x - b.x, a.y - b.y);
7 }
8
9 LL inline cross(PII a, PII b) {
10     return a.x * b.y - a.y * b.x;
11 }
12
13 LL inline cross(PII a, PII b, PII c) {
14     PII u = b - a, v = c - a;
15     return cross(u, v);
16 }

```

```

17
18 // 动态凸包
19
20 struct Hull {
21     SI su, sd;
22     bool inline query(SI &s, PII u, int o) {
23         SIT l = s.upper_bound(u), r = s.lower_bound(u);
24         if (r == s.end() || l == s.begin()) return false;
25         l--;
26         return cross(*l, u, *r) * o <= 0;
27     }
28     void inline insert(SI &s, PII u, int o) {
29         if (query(s, u, o)) return;
30         SIT it = s.insert(u).first;
31         while (1) {
32             SIT mid = it;
33             if (mid == s.begin()) break; --mid;
34             SIT l = mid;
35             if (l == s.begin()) break; --l;
36             if (cross(*l, *mid, u) * o >= 0) break;
37             s.erase(mid);
38         }
39         while (1) {
40             SIT mid = it; ++mid;
41             if (mid == s.end()) break;
42             SIT r = mid; ++r;
43             if (r == s.end()) break;
44             if (cross(u, *mid, *r) * o >= 0) break;
45             s.erase(mid);
46         }
47     }
48     void inline ins(PII u) {
49         insert(su, u, 1), insert(sd, u, -1);
50     }
51     int inline chk(PII u) {
52         return query(su, u, 1) && query(sd, u, -1);
53     }
54 } t;

```

6.11 珂朵莉树

```
1 // 珂朵莉树??
2
3 struct E{
4     int l, r, v;
5     bool operator < (const E &b) const {
6         return r < b.r;
7     }
8 };
9
10 set<E> s;
11
12 typedef set<E>::iterator SIT;
13
14 void split(int i) {
15     SIT u = s.lower_bound((E){ 0, i + 1, 0 });
16     if (u == s.end()) return;
17     if (u -> r > i && u -> l <= i) {
18         E t = *u;
19         s.erase(u);
20         s.insert((E){ t.l, i, t.v });
21         s.insert((E){ i + 1, t.r, t.v });
22     }
23 }
24
25 void inline ins(int l, int r, int v) {
26     split(l - 1), split(r);
27     while (1) {
28         SIT u = s.lower_bound((E){ 0, l, 0, 0 });
29         if (u == s.end()) break;
30         if (u -> r > r) break;
31
32         s.erase(u);
33     }
34     s.insert((E){ l, r, v });
35 }
```


6.12 HashMap

```
1 // Hashmap
2
3
4 struct E{
5     int next, v, w;
6 };
7
8 const int MOD = 999997;
9
10 struct Hash{
11     E e[MOD];
12     int numE, head[MOD];
13     void inline clear() {
14         for (int i = 1; i <= numE; i++)
15             head[e[i].v % MOD] = 0;
16         numE = 0;
17     }
18     int &operator[] (int x) {
19         int t = x % MOD;
20         for (int i = head[t]; i; i = e[i].next) {
21             if (e[i].v == x) {
22                 return e[i].w;
23             }
24         }
25         e[++numE] = (E) { head[t], x, 0 };
26         head[t] = numE;
27         return e[numE].w;
28     }
29 } t
```

6.13 全局平衡二叉树

```
1 // 全局平衡二叉树
2
3
4 vector<int> g[N];
5
```

```

6  int lim[N];
7
8  bool vis[N];
9
10 int fa[N], sz[N], son[N], d[N];
11
12 struct Mat{
13     // 定义矩阵的地方
14     Mat operator * (const Mat &b) const {
15
16     }
17 };
18
19 void dfs1(int u) {
20     sz[u] = 1;
21     for (int v: g[u]) {
22         if (v == fa[u]) continue;
23         fa[v] = u;
24         d[v] = d[u] + 1;
25         dfs1(v);
26         sz[u] += sz[v];
27         if (sz[v] > sz[son[u]]) son[u] = v;
28     }
29 }
30
31 int len, b[N], val[N], rt[N], ps[N];
32
33 struct T{
34     int l, r, f;
35     Mat v, s;
36 } t[N];
37
38 int inline getM(int x, int y) {
39     int mn = 2e9, p = -1;
40     for (int i = x; i <= y; i++)
41         if (chkMin(mn, max(val[i] - 1] - val[x - 1], val[y] - val[i]))) p = i;
42     return p;
43 }
44

```

```

45 #define ls t[p].l
46 #define rs t[p].r
47
48 void pu(int p) {
49     if (ls && rs) t[p].s = t[rs].s * t[p].v * t[ls].s;
50     else if (ls) t[p].s = t[p].v * t[ls].s;
51     else if (rs) t[p].s = t[rs].s * t[p].v;
52     else t[p].s = t[p].v;
53 }
54
55 void inline bd(int &p, int l, int r, int F) {
56     if (l > r) return;
57     int mid = getM(l, r);
58     p = b[mid];
59     t[p].f = F;
60     bd(ls, l, mid - 1, p), bd(rs, mid + 1, r, p);
61     pu(p);
62 }
63
64 void inline remake(int u) {
65     // 更新 u 的子树了, 更新矩阵
66 }
67
68 void inline updF(int v) {
69     // u 的轻儿子 v 变了, 更新轻儿子对自己的影响
70 }
71
72 void inline bd(int tp) {
73     int x = tp; vector<int> z;
74     while (x) z.pb(x), x = son[x];
75     for (int u: z) {
76         for (int v: g[u])
77             if (v != fa[u] && v != son[u]) bd(v), updF(v);
78         remake(u);
79     }
80     len = 0;
81     for (int v: z) b[++len] = v, val[len] = sz[v] - sz[son[v]];
82     for (int i = 1; i <= len; i++) val[i] += val[i - 1];
83     bd(rt[tp], 1, len, 0);

```

```

84         ps[rt[tp]] = tp;
85     }
86
87     void inline sop(int x) {
88         while (x) {
89             remake(x); int p = x, y = 0;
90             while (p) y = ps[p], pu(p), p = t[p].f;
91             if (!fa[y]) break;
92             updF(y), x = fa[y];
93         }
94     }

```

7 计算几何

7.1 Basic

```

1  const double eps = 1e-4;
2  typedef pair<double, double> PDD;
3  struct Line{
4      PDD s, t;
5  };
6
7  int inline cmp(double x, double y) {
8      if (fabs(x - y) < eps) return 0;
9      return x < y ? -1 : 1;
10 }
11
12 double inline cross(PDD a, PDD b) { return a.fi * b.se - a.se * b.fi; }
13 PDD operator - (const PDD &a, const PDD &b) { return make_pair(a.fi - b.fi, a.se - b.se); }
14 PDD operator + (const PDD &a, const PDD &b) { return make_pair(a.fi + b.fi, a.se + b.se); }
15 PDD operator / (const PDD &a, double b) { return make_pair(a.fi / b, a.se / b); }
16 PDD operator * (const PDD &a, double b) { return make_pair(a.fi * b, a.se * b); }
17 double inline area(PDD a, PDD b, PDD c) { return cross(b - a, c - a); }
18 double inline dot(PDD a, PDD b) { return a.fi * b.fi + a.se * b.se; }
19 double inline len(PDD a) { return sqrt(dot(a, a)); }
20 double inline project(PDD a, PDD b, PDD c) { return dot(b - a, c - a) / len(b - a); }
21 double inline dist(PDD a, PDD b) { return sqrt((a.fi - b.fi) * (a.fi - b.fi) + (a.se - b.se) * (a.se - b.se)); }
22 // 顺时针转 x

```

```

23 PDD inline rotate(PDD a, double x) { return make_pair ( cos(x) * a.fi + sin(x) * a.se, -s
24 PDD inline norm(PDD a) { return a / len(a); }
25 double angle(PDD a, PDD b) {
26     return acos(dot(a, b) / len(a) / len(b));
27 }
28 int sign(double fi) {
29     if (fabs(fi) < eps) return 0;
30     if (fi < 0) return -1;
31     return 1;
32 }

```

7.2 点到线段距离

```

1 LD getD(PDD a, PDD u, PDD v) {
2     LD w = min(dis(a, u), dis(a, v));
3     LD c = dot(a - u, v - u);
4     LD t = dis(u, v);
5     c /= t;
6     if (cmp(c, 0) >= 0 && cmp(c, t) <= 0) {
7         LD z = norm(u - a);
8         LD val = sqrt(z - c * c);
9         w = val;
10    }
11    return w;
12 }

```

7.3 线段交

```

1 bool segInter(PDD a1, PDD a2, PDD b1, PDD b2) {
2     double c1 = cross(a2 - a1, b1 - a1), c2 = cross(a2 - a1, b2 - a1);
3     double c3 = cross(b2 - b1, a2 - b1), c4 = cross(b2 - b1, a1 - b1);
4     return sign(c1) * sign(c2) <= 0 && sign(c3) * sign(c4) <= 0;
5 }
6
7 bool cmp2 (const Line &a, const Line &b) {
8     double A = getAngle(a), B = getAngle(b);
9     if (A != B) return A < B;
10    else return area(a.s, a.t, b.t) < 0;

```

```

11 }
12
13 PDD getInter(PDD p, PDD v, PDD q, PDD w) {
14     PDD u = p - q;
15     double t = cross(w, u) / cross(v, w);
16     return make_pair(p.fi + t * v.fi, p.se + t * v.se);
17 }
18
19 PDD getInter(Line a, Line b) { return getInter(a.s, a.t - a.s, b.s, b.t - b.s); }
20
21 bool inline Right(Line a, Line b, Line c) {
22     PDD u = getInter(b, c);
23     return area(a.s, a.t, u) <= 0;
24 }

```

7.4 凸包

```

1 void inline andrew() {
2     sort(p + 1, p + 1 + n);
3     for (int i = 1; i <= n; i++) {
4         while (top > 1 && area(p[s[top - 1]], p[s[top]], p[i]) < 0) {
5             if (area(p[s[top - 1]], p[s[top]], p[i]) <= 0) st[s[top--]] = false;
6             else top--;
7         }
8         st[i] = true, s[++top] = i;
9     }
10    st[1] = false;
11    for (int i = n; i; i--) {
12        if (!st[i]) {
13            while (top > 1 && area(p[s[top - 1]], p[s[top]], p[i]) <= 0)
14                st[s[top--]] = false;
15            st[i] = true, s[++top] = i;
16        }
17    }
18    for (int i = 0; i < top; i++) s[i] = s[i + 1];
19    top--;
20 }

```

7.5 半平面交

```
1 struct Line{
2     PDD s, t;
3     int id;
4 } e[N];
5
6 // 半平面交
7 double HPI() {
8     sort(e + 1, e + 1 + n, cmp2);
9     int hh = 0, tt = -1;
10    for (int i = 1; i <= n; i++) {
11        if (i && getAngle(e[i]) == getAngle(e[i - 1])) continue;
12        while (hh < tt && Right(e[i], e[q[tt - 1]], e[q[tt]])) tt--;
13        while (hh < tt && Right(e[i], e[q[hh]], e[q[hh + 1]])) hh++;
14        q[++tt] = i;
15    }
16    while (hh < tt && Right(e[q[hh]], e[q[tt - 1]], e[q[tt]])) tt--;
17    while (hh < tt && Right(e[q[tt]], e[q[hh]], e[q[hh + 1]])) hh++;
18    q[++tt] = q[hh];
19    tot = 0;
20    for (int i = hh; i < tt; i++)
21        p[++tot] = getInter(e[q[i]], e[q[i + 1]]);
22    double res = 0;
23    for (int i = 1; i < tot; i++)
24        res += area(p[1], p[i], p[i + 1]);
25    return res / 2;
26 }
```

7.6 最小圆覆盖

```
1 Point inline getCircle(Point a, Point b, Point c) {
2     return Inter((a + b) / 2, rotate(b - a, PI / 2), (a + c) / 2, rotate(c - a, PI / 2));
3 }
4
5 // 最小圆覆盖
6
7 void inline minCircle(PDD a[]) {
8     random_shuffle(a + 1, a + 1 + n);
```

```

9     double r = 0; Point u = a[1];
10    for (int i = 2; i <= n; i++) {
11        if (cmp(r, len(u - a[i])) == -1) {
12            r = 0, u = a[i];
13            for (int j = 1; j < i; j++) {
14                if (cmp(r, len(u - a[j])) == -1) {
15                    r = len(a[i] - a[j]) / 2, u = (a[i] + a[j]) / 2;
16                    for (int k = 1; k < j; k++) {
17                        if (cmp(r, len(u - a[k])) == -1) {
18                            u = getCircle(a[i], a[j], a[k]), r = len(a[i] - u);
19                        }
20                    }
21                }
22            }
23        }
24    }
25 }

```

7.7 自适应辛普森积分

```

1 // 自适应辛普森积分
2 double inline f(double fi) {
3     return ?;
4 }
5 double inline s(double l, double r) {
6     double mid = (l + r) / 2;
7     return (r - l) * (f(l) + 4 * f(mid) + f(r)) / 6;
8 }
9
10 double inline asr(double l, double r) {
11     double mid = (l + r) / 2, v = s(l, r);
12     double a = s(l, mid), b = s(mid, r);
13     if (fabs(a + b - v) < eps) return v;
14     else return asr(l, mid) + asr(mid, r);
15 }

```

7.8 极角排序


```

1 // https://codeforces.com/contest/1284/problem/E 的怨念 不丢精度的极角排序
2
3 LL inline cross(PII x, PII y) {
4     return 1ll * x.fi * y.se - 1ll * x.se * y.fi;
5 }
6
7 int inline quad(PII x) {
8     if (x.fi >= 0 && x.se >= 0) return 1;
9     if (x.fi <= 0 && x.se >= 0) return 2;
10    if (x.fi <= 0 && x.se <= 0) return 3;
11    if (x.fi >= 0 && x.se <= 0) return 4;
12    return 0;
13 }

```

7.9 Int 下凸包 + 闵可夫斯基和

```

1 // PII andrew + mincowf
2
3
4 LL operator * (PII a, PII b) {
5     return (LL)a.fi * b.se - (LL)a.se * b.fi;
6 }
7
8 PII operator + (PII a, PII b) {
9     return mp(a.fi + b.fi, a.se + b.se);
10 }
11
12 PII operator - (PII a, PII b) {
13     return mp(a.fi - b.fi, a.se - b.se);
14 }
15
16 LL dot (PII a, PII b) {
17     return (LL)a.fi * a.se + (LL)b.fi * b.se;
18 }
19
20 vector<PII> inline andrew(vector<PII> a) {
21     int n = a.size();
22     top = 0;

```

```

23     sort(a.begin(), a.end());
24     for (int i = 0; i < n; i++) {
25         while (top > 1 && (a[i] - a[s[top - 1]]) * (a[s[top]] - a[s[top - 1]]) > 0) {
26             vis[s[top--]] = 0;
27         }
28         vis[i] = 1, s[++top] = i;
29     }
30     vis[0] = 0;
31     for (int i = n - 1; i >= 0; i--) {
32         if (!vis[i]) {
33             while (top > 1 && (a[i] - a[s[top - 1]]) * (a[s[top]] - a[s[top - 1]]) > 0) {
34                 vis[s[top--]] = 0;
35                 vis[i] = 1, s[++top] = i;
36             }
37         }
38         --top;
39         vector<PII> ret;
40         for (int i = 1; i <= top; i++) ret.pb(a[s[i]]);
41         for (int i = 0; i < n; i++) vis[i] = 0;
42         return ret;
43     }
44
45     // 有
46
47     vector<PII> calc(vector<PII> a, vector<PII> b) {
48         vector<PII> c;
49         c.pb(a[0] + b[0]);
50         vector<PII> dx, dy;
51         for (int i = 1; i < a.size(); i++) dx.pb(a[i] - a[i - 1]);
52         dx.pb(a[0] - a.back());
53         for (int i = 1; i < b.size(); i++) dy.pb(b[i] - b[i - 1]);
54         dy.pb(b[0] - b.back());
55         int i = 0, j = 0;
56         while (i < dx.size() && j < dy.size()) {
57             if (dx[i] * dy[j] > 0)
58                 c.pb(c.back() + dx[i++]);
59             else if (dx[i] * dy[j] == 0 && c.size() > 1) {
60                 // 共线放一起不然是错的!!!!
61                 if (dot(c.back() - c[c.size() - 2], dx[i]) > 0)

```

```

62         c.pb(c.back() + dx[i++]);
63         else c.pb(c.back() + dy[j++]);
64     } else {
65         c.pb(c.back() + dy[j++]);
66     }
67 }
68 while (i < dx.size()) c.pb(c.back() + dx[i++]);
69 while (j < dy.size()) c.pb(c.back() + dy[j++]);
70 assert(c.back() == c[0]);
71 c.pop_back();
72 return c;
73 }

```