

XCPC - Templates

north-h

Contents

1	数据结构	2
1.1	树状数组（单点修改，区间查询）	2
1.2	树状数组（区间修改，单点查询）	2
1.3	树状数组（区间修改，区间查询）	3
1.4	线段树	3
1.5	ST 表	5
1.6	二维树状数组	6
1.7	并查集	6
1.8	线段树维护区间和	7
1.9	线段树维护区间最大值	8
1.10	线段树维护区间最小值	9
1.11	线段树维护最大子段和	10
2	字符串	11
3	杂项	11
3.1	快读快写	11
3.2	取模	12
4	数学	13
4.1	组合数（递推）	13
4.2	组合数（逆元）	14

1 数据结构

1.1 树状数组（单点修改，区间查询）

```
1  template <class T>
2  struct BIT {
3      vector<T> tr;
4      int n;
5      BIT(int N) {
6          n = N;
7          tr.resize(n + 1);
8      }
9      void add(int x, T k) {
10         for(int i = x; i < n; i += (i & -i))
11             tr[i] += k;
12     }
13     T query(int x) {
14         T res = 0;
15         for(int i = x; i; i -= (i & -i))
16             res += tr[i];
17         return res;
18     }
19     T range_query(int l, int r) {
20         return query(r) - query(l - 1);
21     }
22 };
```

1.2 树状数组（区间修改，单点查询）

```
1  template <class T>
2  struct BIT {
3      vector<T> tr;
4      int n;
5      BIT(int N) {
6          n = N;
7          tr.resize(n + 1);
8      }
9      void add(int x, T k) {
10         for(int i = x; i < n; i += (i & -i))
11             tr[i] += k;
12     }
13     void range_add(int l, int r) {
14         add(l, k);
15         add(r + 1, -k);
16     }
17     T query(int x) {
18         T res = 0;
19         for(int i = x; i; i -= (i & -i))
20             res += tr[i];
21         return res;
22     }
23 };
```

1.3 树状数组（区间修改，区间查询）

```

1  template <class T>
2  struct BIT {
3      vector<T> sum1, sum2;
4      int n;
5      BIT() {}
6      void init(int N) {
7          n = N;
8          sum1.resize(n);
9          sum2.resize(n);
10     }
11     void add(int x, T k) {
12         for(int i = x; i < n; i += (i & -i))
13             sum1[i] += k, sum2[i] += x * k;
14     }
15     void add(int l, int r, T x) {
16         add(l, x), add(r + 1, -x);
17     }
18     T query(int x) {
19         T res = 0;
20         for(int i = x; i > 0; i -= (i & -i))
21             res += (x + 1) * sum1[i] - sum2[i];
22         return res;
23     }
24     T query(int l, int r) {
25         return query(r) - query(l - 1);
26     }
27 }
28 };

```

1.4 线段树

```

1  template <class T>
2  struct Seg{
3      struct Node{ int l, r, sum, lazy; };
4
5      vector<Node> tr;
6      vector<int> a;
7      int n;
8
9      Seg() {}
10
11     void init(int N) {
12         n = N;
13         tr.resize(n * 4);
14         a.resize(n);
15     }
16
17     void add(int x, int k) {
18         a[x] = k;
19     }

```

```

20
21 void pushup(int u) {
22     tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
23 }
24
25 void pushdown(int u) {
26     if (tr[u].lazy) {
27         tr[u << 1].sum += tr[u].lazy * (tr[u << 1].r - tr[u << 1].l + 1);
28         tr[u << 1 | 1].sum += tr[u].lazy * (tr[u << 1 | 1].r - tr[u << 1 | 1].l +
29             1);
30         tr[u << 1].lazy += tr[u].lazy;
31         tr[u << 1 | 1].lazy += tr[u].lazy;
32         tr[u].lazy = 0;
33     }
34 }
35 void build(int u, int l, int r) {
36     tr[u] = {l, r, a[l], 0};
37     if (l == r) return;
38     int mid = l + r >> 1;
39     pushdown(u);
40     build(u << 1, l, mid);
41     build(u << 1 | 1, mid + 1, r);
42     pushup(u);
43 }
44 //区间修改
45 void modify(int u, int l, int r, int k) {
46     if (tr[u].l >= l && tr[u].r <= r) {
47         tr[u].sum += (tr[u].r - tr[u].l + 1) * k;
48         tr[u].lazy += k;
49         return;
50     }
51     pushdown(u);
52     int mid = tr[u].l + tr[u].r >> 1;
53     if (l <= mid) modify(u << 1, l, r, k);
54     if (r > mid) modify(u << 1 | 1, l, r, k);
55     pushup(u);
56 }
57 //单点修改
58 void modify(int u, int x, int k) {
59     if (tr[u].l == tr[u].r) {
60         tr[u].sum += k;
61         return;
62     }
63     int mid = tr[u].l + tr[u].r >> 1;
64     if (x <= mid) modify(u << 1, x, k);
65     else modify(u << 1 | 1, x, k);
66 }
67 //区间查询
68 int query(int u, int l, int r) {
69     if (tr[u].l >= l && tr[u].r <= r) return tr[u].sum;
70     pushdown(u);
71     int sum = 0;

```

```

72     int mid = tr[u].l + tr[u].r >> 1;
73     if (l <= mid) sum += query(u << 1, l, r);
74     if (r > mid) sum += query(u << 1 | 1, l, r);
75     return sum;
76 }
77 //单点查询
78 int query(int u, int x) {
79     if (tr[u].l == tr[u].r) return tr[u].sum;
80     pushdown(u);
81     int mid = tr[u].l + tr[u].r >> 1;
82     if (x <= mid) return query(u << 1, x);
83     else return query(u << 1 | 1, x);
84 }
85 };

```

1.5 ST 表

```

1  template<class T>
2  struct ST {
3      T n;
4      vector<vector<T>> f, g;
5      vector<int> lg2;
6      ST(const vector<T> &a) {
7          n = (int)a.size();
8          lg2.resize(n + 1);
9          lg2[0] = -1;
10         for(int i = 1; i <= n; i++) {
11             lg2[i] = lg2[i >> 1] + 1;
12         }
13         f.resize(n + 1, vector<T>(lg2[n] + 1));
14         g.resize(n + 1, vector<T>(lg2[n] + 1));
15         for (int i = 1; i <= n; i++) {
16             f[i][0] = a[i];
17             g[i][0] = a[i];
18         }
19         for (int j = 1; (1 << j) <= n; j++) {
20             for (int i = 1; i + (1 << j) - 1 <= n; i++) {
21                 f[i][j] = max(f[i][j - 1], f[i + (1 << (j - 1))][j - 1]);
22                 g[i][j] = min(g[i][j - 1], g[i + (1 << (j - 1))][j - 1]);
23             }
24         }
25     }
26
27     T query_max(int l, int r) {
28         int k = lg2[r - l + 1];
29         return max(f[l][k], f[r - (1 << k) + 1][k]);
30     }
31
32     T query_min(int l, int r) {
33         int k = lg2[r - l + 1];
34         return min(g[l][k], g[r - (1 << k) + 1][k]);
35     }

```

```
36 };
```

1.6 二维树状数组

```
1  template <class T>
2  struct BIT_2D {
3      vector<vector<T>> tr;
4      int n, m;
5
6      BIT_2D(int N, int M) {
7          n = N, m = M;
8          tr.resize(n + 1, vector<T>(m + 1));
9      }
10
11     int lowbit(int x) { return x & (-x); }
12
13     void add(int x, int y, T k) {
14         for (int i = x; i <= n; i += lowbit(i))
15             for (int j = y; j <= m; j += lowbit(j))
16                 tr[i][j] += k;
17     }
18
19     T query(int x, int y) {
20         T res = 0;
21         for (int i = x; i > 0; i -= lowbit(i))
22             for (int j = y; j > 0; j -= lowbit(j))
23                 res += tr[i][j];
24         return res;
25     }
26
27     T query(int x1, int y1, int x2, int y2) {
28         return query(x2, y2) - query(x2, y1-1) - query(x1-1, y2) + query(x1-1, y1-1);
29     }
30 };
```

1.7 并查集

```
1  struct DSU {
2      vector<int> fa;
3
4      DSU(int n) {
5          fa.resize(n + 1);
6          for (int i = 1; i <= n; i++) fa[i] = i;
7      }
8
9      int find(int x) {
10         if (fa[x] != x) fa[x] = find(fa[x]);
11         return fa[x];
12     }
13
14     bool same(int x, int y) {
```

```

15     int px = find(x);
16     int py = find(y);
17     return x == y;
18 }
19
20 void merge(int x, int y) {
21     int px = find(x);
22     int py = find(y);
23     if (px != py) {
24         fa[px] = py;
25     }
26 }
27 };

```

1.8 线段树维护区间和

```

1  template <class T>
2  struct Seg {
3      struct Node { int l, r; T lazy, mx, mn, sum; };
4      vector<Node> tr; vector<T> a; int n;
5      Seg(int N) { n = N + 1; tr.resize(n * 4); a.resize(n);}
6      void pushup(int u) {
7          tr[u] = merge(tr[u], tr[u << 1], tr[u << 1 | 1]);
8      }
9      Node merge(Node t, Node l, Node r) {
10         t.sum = l.sum + r.sum;
11         return t;
12     }
13     void pushdown(int u) {
14         if (tr[u].lazy) {
15             tr[u << 1].lazy += tr[u].lazy;
16             tr[u << 1].sum += (tr[u << 1].r - tr[u << 1].l + 1) * tr[u].lazy;
17             tr[u << 1 | 1].lazy += tr[u].lazy;
18             tr[u << 1 | 1].sum += (tr[u << 1 | 1].r - tr[u << 1 | 1].l + 1) * tr[u].
                lazy;
19             tr[u].lazy = 0;
20         }
21     }
22     void build(int u, int l, int r) {
23         tr[u] = {l, r, 0, a[l], a[l], a[l]};
24         if (l == r) return;
25         int mid = l + r >> 1;
26         pushdown(u);
27         build(u << 1, l, mid);
28         build(u << 1 | 1, mid + 1, r);
29         pushup(u);
30     }
31     void modify(int u, int l, int r, T k) {
32         if (tr[u].l >= l && tr[u].r <= r) {
33             tr[u].lazy += k;
34             tr[u].sum += (tr[u].r - tr[u].l + 1) * k;
35             return;

```

```

36     }
37     pushdown(u);
38     int mid = tr[u].l + tr[u].r >> 1;
39     if (l <= mid) modify(u << 1, l, r, k);
40     if (r > mid) modify(u << 1 | 1, l, r, k);
41     pushup(u);
42 }
43 Node query(int u, int l, int r) {
44     if (tr[u].l >= l && tr[u].r <= r) return tr[u];
45     pushdown(u);
46     int mid = tr[u].l + tr[u].r >> 1;
47     if (r <= mid) return query(u << 1, l, r);
48     if (l > mid) return query(u << 1 | 1, l, r);
49     Node t = merge(t, query(u << 1, l, r), query(u << 1 | 1, l, r));
50     return t;
51 }
52 };

```

1.9 线段树维护区间最大值

```

1  template <class T>
2  struct Seg {
3      struct Node { int l, r; T lazy, mx, mn; };
4      vector<Node> tr; vector<T> a; int n;
5      Seg(int N) { n = N + 1; tr.resize(n * 4); a.resize(n); }
6      void pushup(int u) {
7          tr[u] = merge(tr[u], tr[u << 1], tr[u << 1 | 1]);
8      }
9      Node merge(Node t, Node l, Node r) {
10         t.mx = max(l.mx, r.mx);
11         return t;
12     }
13     void pushdown(int u) {
14         if (tr[u].lazy) {
15             tr[u << 1].mx = max(tr[u << 1].mx, tr[u].lazy);
16             tr[u << 1].lazy = max(tr[u].lazy, tr[u << 1].lazy);
17             tr[u << 1 | 1].mx = max(tr[u << 1 | 1].mx, tr[u].lazy);
18             tr[u << 1 | 1].lazy = max(tr[u].lazy, tr[u << 1 | 1].lazy);
19             tr[u].lazy = 0;
20         }
21     }
22     void build(int u, int l, int r) {
23         tr[u] = {l, r, 0, a[l], a[l]};
24         if (l == r) return;
25         int mid = l + r >> 1;
26         pushdown(u);
27         build(u << 1, l, mid);
28         build(u << 1 | 1, mid + 1, r);
29         pushup(u);
30     }
31     void modify(int u, int l, int r, T k) {
32         if (tr[u].l >= l && tr[u].r <= r) {

```



```

33         tr[u].lazy = max(tr[u].lazy, k);
34         tr[u].mx = max(tr[u].mx, k);
35         return;
36     }
37     pushdown(u);
38     int mid = tr[u].l + tr[u].r >> 1;
39     if (l <= mid) modify(u << 1, l, r, k);
40     if (r > mid) modify(u << 1 | 1, l, r, k);
41     pushup(u);
42 }
43 Node query(int u, int l, int r) {
44     if (tr[u].l >= l && tr[u].r <= r) return tr[u];
45     pushdown(u);
46     int mid = tr[u].l + tr[u].r >> 1;
47     if (r <= mid) return query(u << 1, l, r);
48     if (l > mid) return query(u << 1 | 1, l, r);
49     Node t = merge(t, query(u << 1, l, r), query(u << 1 | 1, l, r));
50     return t;
51 }
52 };

```

1.10 线段树维护区间最小值

```

1  template <class T>
2  struct Seg {
3      struct Node { int l, r; T lazy, mx, mn; };
4      vector<Node> tr; vector<T> a; int n;
5      Seg(int N) { n = N + 1; tr.resize(n * 4); a.resize(n); }
6      void pushup(int u) {
7          tr[u] = merge(tr[u], tr[u << 1], tr[u << 1 | 1]);
8      }
9      Node merge(Node t, Node l, Node r) {
10         t.mx = min(l.mx, r.mx);
11         return t;
12     }
13     void pushdown(int u) {
14         if (tr[u].lazy) {
15             tr[u << 1].mx = min(tr[u << 1].mx, tr[u].lazy);
16             tr[u << 1].lazy = min(tr[u].lazy, tr[u << 1].lazy);
17             tr[u << 1 | 1].mx = min(tr[u << 1 | 1].mx, tr[u].lazy);
18             tr[u << 1 | 1].lazy = min(tr[u].lazy, tr[u << 1 | 1].lazy);
19             tr[u].lazy = 0;
20         }
21     }
22     void build(int u, int l, int r) {
23         tr[u] = {l, r, 0, a[l], a[l]};
24         if (l == r) return;
25         int mid = l + r >> 1;
26         pushdown(u);
27         build(u << 1, l, mid);
28         build(u << 1 | 1, mid + 1, r);
29         pushup(u);

```

```

30     }
31     void modify(int u, int l, int r, T k) {
32         if (tr[u].l >= l && tr[u].r <= r) {
33             tr[u].lazy = min(tr[u].lazy, k);
34             tr[u].mx = min(tr[u].mx, k);
35             return;
36         }
37         pushdown(u);
38         int mid = tr[u].l + tr[u].r >> 1;
39         if (l <= mid) modify(u << 1, l, r, k);
40         if (r > mid) modify(u << 1 | 1, l, r, k);
41         pushup(u);
42     }
43     Node query(int u, int l, int r) {
44         if (tr[u].l >= l && tr[u].r <= r) return tr[u];
45         pushdown(u);
46         int mid = tr[u].l + tr[u].r >> 1;
47         if (r <= mid) return query(u << 1, l, r);
48         if (l > mid) return query(u << 1 | 1, l, r);
49         Node t = merge(t, query(u << 1, l, r), query(u << 1 | 1, l, r));
50         return t;
51     }
52 };

```

1.11 线段树维护最大子段和

```

1  template <class T>
2  struct Seg {
3      struct Node { int l, r; T lazy, mx, lmx, rmx, sum; };
4      vector<Node> tr; vector<T> a; int n;
5      Seg(int n) { n = N + 1; tr.resize(n * 4); a.resize(n); }
6      void pushup(int u) {
7          tr[u] = merge(tr[u], tr[u << 1], tr[u << 1 | 1]);
8      }
9      Node merge(Node t, Node l, Node r) {
10         t.lmx = max(l.lmx, l.sum + r.lmx);
11         t.rmx = max(r.rmx, r.sum + l.rmx);
12         t.sum = l.sum + r.sum;
13         t.mx = max(max(l.mx, r.mx), l.rmx + r.lmx);
14         return t;
15     }
16     void pushdown(int u) {
17         if (tr[u].lazy) {
18             tr[u << 1].lazy = tr[u].lazy;
19             tr[u << 1 | 1].lazy = tr[u].lazy;
20             tr[u].lazy = 0;
21         }
22     }
23     void build(int u, int l, int r) {
24         tr[u] = {l, r, 0, a[l], a[l], a[l], a[l]};
25         if (l == r) return;
26         int mid = l + r >> 1;

```

```

27     pushdown(u);
28     build(u << 1, l, mid);
29     build(u << 1 | 1, mid + 1, r);
30     pushup(u);
31 }
32 void modify(int u, int l, int r, T k) {
33     if (tr[u].l >= l && tr[u].r <= r) {
34         tr[u].lazy = k;
35         tr[u].mx = tr[u].lmx = tr[u].rmx = tr[u].sum = k;
36         return;
37     }
38     pushdown(u);
39     int mid = tr[u].l + tr[u].r >> 1;
40     if (l <= mid) modify(u << 1, l, r, k);
41     if (r > mid) modify(u << 1 | 1, l, r, k);
42     pushup(u);
43 }
44 Node query(int u, int l, int r) {
45     // debug1(u);
46     if (tr[u].l >= l && tr[u].r <= r) return tr[u];
47     pushdown(u);
48     int mid = tr[u].l + tr[u].r >> 1;
49     if (r <= mid) return query(u << 1, l, r);
50     if (l > mid) return query(u << 1 | 1, l, r);
51     Node t = merge(t, query(u << 1, l, r), query(u << 1 | 1, l, r));
52     return t;
53 }
54 };

```

2 字符串

3 杂项

3.1 快读快写

```

1  template <typename T>
2  T read() {
3      T x = 0;
4      char ch = getchar();
5      while (ch < '0' || ch > '9') ch = getchar();
6      while (ch >= '0' && ch <= '9') x = (x << 3) + (x << 1) + ch - '0', ch = getchar
7      ();
8      return x;
9  }
10 template <typename T>
11 void write(T x) {
12     if (x < 0) putchar('-'), x = -x;
13     if (x > 9) write(x / 10);
14     putchar(x % 10 + '0');
15 }

```

3.2 取模

```

1  template<const int T>
2  struct ModInt {
3      const static int mod = T;
4      int x;
5      ModInt(int x = 0) : x(x % mod) {}
6      ModInt(long long x) : x(int(x % mod)) {}
7      int val() {
8          return x;
9      }
10     ModInt operator + (const ModInt &a) const {
11         int x0 = x + a.x;
12         return ModInt(x0 < mod ? x0 : x0 - mod);
13     }
14     ModInt operator - (const ModInt &a) const {
15         int x0 = x - a.x;
16         return ModInt(x0 < 0 ? x0 + mod : x0);
17     }
18     ModInt operator * (const ModInt &a) const {
19         return ModInt(1LL * x * a.x % mod);
20     }
21     ModInt operator / (const ModInt &a) const {
22         return *this * a.inv();
23     }
24     bool operator == (const ModInt &a) const {
25         return x == a.x;
26     };
27     bool operator != (const ModInt &a) const {
28         return x != a.x;
29     };
30     void operator += (const ModInt &a) {
31         x += a.x;
32         if (x >= mod) x -= mod;
33     }
34     void operator -= (const ModInt &a) {
35         x -= a.x;
36         if (x < 0) x += mod;
37     }
38     void operator *= (const ModInt &a) {
39         x = 1LL * x * a.x % mod;
40     }
41     void operator /= (const ModInt &a) {
42         *this = *this / a;
43     }
44     friend ModInt operator + (int y, const ModInt &a) {
45         int x0 = y + a.x;
46         return ModInt(x0 < mod ? x0 : x0 - mod);
47     }
48     friend ModInt operator - (int y, const ModInt &a) {
49         int x0 = y - a.x;
50         return ModInt(x0 < 0 ? x0 + mod : x0);
51     }

```

```

52     friend ModInt operator * (int y, const ModInt &a) {
53         return ModInt(1LL * y * a.x % mod);
54     }
55     friend ModInt operator / (int y, const ModInt &a) {
56         return ModInt(y) / a;
57     }
58     friend ostream &operator<<(ostream &os, const ModInt &a) {
59         return os << a.x;
60     }
61     friend istream &operator>>(istream &is, ModInt &t) {
62         return is >> t.x;
63     }
64
65     ModInt pow(int64_t n) const {
66         ModInt res(1), mul(x);
67         while(n) {
68             if (n & 1) res *= mul;
69             mul *= mul;
70             n >>= 1;
71         }
72         return res;
73     }
74
75     ModInt inv() const {
76         int a = x, b = mod, u = 1, v = 0;
77         while (b) {
78             int t = a / b;
79             a -= t * b;
80             swap(a, b);
81             u -= t * v;
82             swap(u, v);
83         }
84         if (u < 0) u += mod;
85         return u;
86     }
87
88 };
89 using mint = ModInt<998244353>;

```

4 数学

4.1 组合数 (递推)

```

1  int C[N][N];
2
3  void comb(){
4      for(int i = 0; i < N; i++) {
5          for(int j = 0; j <= i; j++) {
6              if(!j) {
7                  C[i][j] = 1;
8                  continue;
9              }

```

```
10         C[i][j] = C[i - 1][j] + C[i - 1][j - 1];
11         C[i][j] %= mod;
12     }
13 }
14 }
```

4.2 组合数（逆元）

```
1 struct comb2{
2     vector<int> fac,inf;
3
4     comb2(int n) {
5         fac.resize(n + 1);
6         inf.resize(n + 1);
7         fac[0] = inf[0] = 1;
8         for(int i = 1; i <= n; i++){
9             fac[i] = fac[i - 1] * i % mod;
10            inf[i] = inf[i - 1] * ksm(i, mod - 2) % mod;
11        }
12    }
13
14    int ksm(int a, int b){ // 快速幂
15        int res = 1;
16        while(b){
17            if(b & 1) res = res * a % mod;
18            a = a * a % mod;
19            b >>= 1;
20        }
21        return res;
22    }
23
24    int query(int n, int m) {
25        int ans;
26        ans = fac[n] * inf[n - m] % mod * inf[m] % mod;
27        return ans;
28    }
29 };
```