XCPC - Templates

north-h

Contents

1 数据结构

1.1 树状数组（单点修改，区间查询）

```
template <class T>
struct BIT {
    vector<T> tr;
    int n;
    BIT(int N) {
        n = N;
        tr.resize(n + 1);
    }
    void add(int x, T k) {
        for(int i = x; i < n; i += (i & -i))
            tr[i] += k;
    }
    T query(int x) {
        T res = 0;
        for(int i = x; i; i -= (i & -i))
            res += tr[i];
        return res;
    }
    T range_query(int l, int r) {
        return query(r) - query(l - 1);
    }
};
```

1.2 树状数组（区间修改，单点查询）

```
template <class T>
struct BIT {
    vector<T> tr;
    int n;
    BIT(int N) {
        n = N;
        tr.resize(n + 1);
    }
    void add(int x, T k) {
        for(int i = x; i < n; i += (i & -i))
            tr[i] += k;
    }
    void range_add(int l, int r) {
        add(l, k);
        add(r + 1, -k);
    }
    T query(int x) {
        T res = 0;
        for(int i = x; i; i -= (i & -i))
            res += tr[i];
        return res;
    }
};
```

1.3 树状数组（区间修改，区间查询）

```
template <class T>
struct BIT {
    vector<T> sum1, sum2;
    int n;
    BIT() {}
    void init(int N) {
        n = N;
        sum1.resize(n);
        sum2.resize(n);
    }
    void add(int x, T k) {
        for(int i = x; i < n; i += (i & -i))
            sum1[i] += k, sum2[i] += x * k;
    }
    void add(int l, int r, T x) {
        add(l, x), add(r + 1, -x);
    }
    T query(int x) {
        T res = 0;
        for(int i = x; i > 0; i -= (i & -i))
            res += (x + 1) * sum1[i] - sum2[i];
        return res;
    }
    T query(int l, int r) {
        return query(r) - query(l - 1);

    }
};
```

1.4 线段树

```
template <class T>
struct Seg{
    struct Node{ int l, r, sum, lazy; };

    vector<Node> tr;
    vector<int> a;
    int n;

    Seg() {};

    void init(int N) {
        n = N;
        tr.resize(n * 4);
        a.resize(n);
    }

    void add(int x, int k) {
        a[x] = k;
    }
```

```cpp

    void pushup(int u) {
        tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
    }

    void pushdown(int u) {
        if (tr[u].lazy) {
            tr[u << 1].sum += tr[u].lazy * (tr[u << 1].r - tr[u << 1].l + 1);
            tr[u << 1 | 1].sum += tr[u].lazy * (tr[u << 1 | 1].r - tr[u << 1 | 1].l + 1);
            tr[u << 1].lazy += tr[u].lazy;
            tr[u << 1 | 1].lazy += tr[u].lazy;
            tr[u].lazy = 0;
        }
    }

    void build(int u, int l,int r) {
        tr[u] = {l, r, a[l], 0};
        if (l == r)return;
        int mid = l + r >> 1;
        pushdown(u);
        build(u << 1, l, mid);
        build(u << 1 | 1, mid + 1, r);
        pushup(u);
    }
    //区间修改
    void modify(int u,int l,int r,int k) {
        if (tr[u].l >= l && tr[u].r <= r) {
            tr[u].sum += (tr[u].r - tr[u].l + 1) * k;
            tr[u].lazy += k;
            return;
        }
        pushdown(u);
        int mid = tr[u].l + tr[u].r >> 1;
        if (l <= mid)modify(u << 1, l, r, k);
        if (r > mid) modify(u << 1 | 1, l, r, k);
        pushup(u);
    }
    //单点修改
    void modify(int u,int x,int k) {
        if (tr[u].l == tr[u].r) {
            tr[u].sum += k;
            return;
        }
        int mid = tr[u].l + tr[u].r >> 1;
        if (x <= mid)modify(u << 1, x, k);
        else modify(u << 1 | 1, x, k);
    }
    //区间查询
    int query(int u,int l,int r) {
        if (tr[u].l >= l && tr[u].r <= r) return tr[u].sum;
        pushdown(u);
        int sum = 0;
        int mid = tr[u].l + tr[u].r >> 1;
        if (l <= mid)sum += query(u << 1, l, r);
        if (r > mid) sum += query(u << 1 | 1, l, r);
        return sum;
    }
    //单点查询
    int query(int u,int x) {
        if (tr[u].l == tr[u].r)return tr[u].sum;
        pushdown(u);
        int mid = tr[u].l + tr[u].r >> 1;
        if (x <= mid)return query(u << 1, x);
        else return query(u << 1 | 1, x);
    }
};
```

1.5 ST 表

```
template<class T>
struct ST {
    T n;
    vector<vector<T>> f, g;
    vector<int> lg2;
    ST(const vector<T> &a) {
        n = (int)a.size();
        lg2.resize(n + 1);
        lg2[0] = -1;
        for(int i = 1; i <= n; i ++) {
            lg2[i] = lg2[i >> 1] + 1;
        }
        f.resize(n + 1, vector<T>(lg2[n] + 1));
        g.resize(n + 1, vector<T>(lg2[n] + 1));
        for (int i = 1; i <= n; i++) {
            f[i][0] = a[i];
            g[i][0] = a[i];
        }
        for (int j = 1; (1 << j) <= n; j++) {
            for (int i = 1; i + (1 << j) - 1 <= n; i++) {
                f[i][j] = max(f[i][j - 1], f[i + (1 << (j - 1))][j - 1]);
                g[i][j] = min(g[i][j - 1], g[i + (1 << (j - 1))][j - 1]);
            }
        }
    }

    T query_max(int l, int r) {
        int k = lg2[r - l + 1];
        return max(f[l][k], f[r - (1 << k) + 1][k]);
    }

    T query_min(int l, int r) {
        int k = lg2[r - l + 1];
        return min(g[l][k], g[r - (1 << k) + 1][k]);
    }
};
```

1.6 二维树状数组

```
template <class T>
struct BIT_2D {
    vector<vector<T>> tr;
    int n, m;

    BIT_2D(int N, int M) {
        n = , m = M;
        tr.resize(n + 1, vector<T>(m + 1));
    }

    int lowbit(int x) { return x & (-x); }

    void add(int x, int y, T k) {
        for (int i = x; i <= n; i += lowbit(i))
            for (int j = y; j <= m; j += lowbit(j))
                tr[i][j] += k;
    }

    T query(int x, int y) {
        T res = 0;
        for (int i = x; i; i -= lowbit(i))
            for (int j = y; j; j -= lowbit(j))
                res += tr[i][j];
        return res;
    }

    T query(int x1, int y1, int x2, int y2) {
        return query(x2, y2) - query(x2, y1-1) - query(x1-1, y2) + query(x1-1, y1-1);
    }
};
```

1.7 并查集

```
struct DSU {
    vector<int> fa;

    DSU(int n) {
        fa.resize(n + 1);
        for (int i = 1; i <= n; i ++) fa[i] = i;
    }

    int find(int x) {
        if (fa[x] != x) fa[x] = find(fa[x]);
        return fa[x];
    }

    bool same(int x, int y) {
```

```cpp
        int px = find(x);
        int py = find(y);
        return x == y;
    }

    void merge(int x, int y) {
        int px = find(x);
        int py = find(y);
        if (px != py) {
            fa[px] = py;
        }
    }
};
```

1.8 线段树维护区间和

```cpp
template <class T>
struct Seg {
    struct Node { int l, r; T lazy, mx, mn, sum; };
    vector<Node> tr; vector<T> a; int n;
    Seg(int N) { n = N + 1; tr.resize(n * 4); a.resize(n);}
    void pushup(int u) {
        tr[u] = merge(tr[u], tr[u << 1], tr[u << 1 | 1]);
    }
    Node merge(Node t, Node l, Node r) {
        t.sum = l.sum + r.sum;
        return t;
    }
    void pushdown(int u) {
        if (tr[u].lazy) {
            tr[u << 1].lazy += tr[u].lazy;
            tr[u << 1].sum += (tr[u << 1].r - tr[u << 1].l + 1) * tr[u].lazy;
            tr[u << 1 | 1].lazy += tr[u].lazy;
            tr[u << 1 | 1].sum += (tr[u << 1 | 1].r - tr[u << 1 | 1].l + 1) * tr[u].lazy;
            tr[u].lazy = 0;
        }
    }
    void build(int u, int l, int r) {
        tr[u] = {l, r, 0, a[l], a[l], a[l]};
        if (l == r) return;
        int mid = l + r >> 1;
        pushdown(u);
        build(u << 1, l, mid);
        build(u << 1 | 1, mid + 1, r);
        pushup(u);
    }
    void modify(int u, int l, int r, T k) {
        if (tr[u].l >= l && tr[u].r <= r) {
            tr[u].lazy += k;
            tr[u].sum += (tr[u].r - tr[u].l + 1) * k;
            return;
        }
        pushdown(u);
        int mid = tr[u].l + tr[u].r >> 1;
        if (l <= mid) modify(u << 1, l, r, k);
        if (r > mid) modify(u << 1 | 1, l, r, k);
        pushup(u);
    }
    Node query(int u, int l, int r) {
        if (tr[u].l >= l && tr[u].r <= r) return tr[u];
        pushdown(u);
        int mid = tr[u].l + tr[u].r >> 1;
        if (r <= mid) return query(u << 1, l, r);
        if (l > mid) return query(u << 1 | 1, l, r);
        Node t = merge(t, query(u << 1, l, r), query(u << 1 | 1, l, r));
        return t;
    }
};
```

1.9 线段树维护区间最大值

```
template <class T>
struct Seg {
    struct Node { int l, r; T lazy, mx, mn; };
    vector<Node> tr; vector<T> a; int n;
    Seg(int N) { n = N + 1; tr.resize(n * 4); a.resize(n); }
    void pushup(int u) {
        tr[u] = merge(tr[u], tr[u << 1], tr[u << 1 | 1]);
    }
    Node merge(Node t, Node l, Node r) {
        t.mx = max(l.mx, r.mx);
        return t;
    }
    void pushdown(int u) {
        if (tr[u].lazy) {
            tr[u << 1].mx = max(tr[u << 1].mx, tr[u].lazy);
            tr[u << 1].lazy = max(tr[u].lazy, tr[u << 1].lazy);
            tr[u << 1 | 1].mx = max(tr[u << 1 | 1].mx, tr[u].lazy);
            tr[u << 1 | 1].lazy = max(tr[u].lazy, tr[u << 1 | 1].lazy);
            tr[u].lazy = 0;
        }
    }
    void build(int u, int l, int r) {
        tr[u] = {l, r, 0, a[l], a[l]};
        if (l == r) return;
        int mid = l + r >> 1;
        pushdown(u);
        build(u << 1, l, mid);
        build(u << 1 | 1, mid + 1, r);
        pushup(u);
    }
    void modify(int u, int l, int r, T k) {
        if (tr[u].l >= l && tr[u].r <= r) {
            tr[u].lazy = max(tr[u].lazy, k);
            tr[u].mx = max(tr[u].mx, k);
            return;
        }
        pushdown(u);
        int mid = tr[u].l + tr[u].r >> 1;
        if (l <= mid) modify(u << 1, l, r, k);
        if (r > mid) modify(u << 1 | 1, l, r, k);
        pushup(u);
    }
    Node query(int u, int l, int r) {
        if (tr[u].l >= l && tr[u].r <= r) return tr[u];
        pushdown(u);
        int mid = tr[u].l + tr[u].r >> 1;
        if (r <= mid) return query(u << 1, l, r);
        if (l > mid) return query(u << 1 | 1, l, r);
        Node t = merge(t, query(u << 1, l, r), query(u << 1 | 1, l, r));
        return t;
    }
};
```

1.10 线段树维护区间最小值

```
template <class T>
struct Seg {
   struct Node { int l, r; T lazy, mx, mn; };
   vector<Node> tr; vector<T> a; int n;
   Seg(int N) { n = N + 1; tr.resize(n * 4); a.resize(n); }
   void pushup(int u) {
      tr[u] = merge(tr[u], tr[u << 1], tr[u << 1 | 1]);
   }
   Node merge(Node t, Node l, Node r) {
      t.mx = min(l.mx, r.mx);
      return t;
   }
   void pushdown(int u) {
      if (tr[u].lazy) {
         tr[u << 1].mx = min(tr[u << 1].mx, tr[u].lazy);
         tr[u << 1].lazy = min(tr[u].lazy, tr[u << 1].lazy);
         tr[u << 1 | 1].mx = min(tr[u << 1 | 1].mx, tr[u].lazy);
         tr[u << 1 | 1].lazy = min(tr[u].lazy, tr[u << 1 | 1].lazy);
         tr[u].lazy = 0;
      }
   }
   void build(int u, int l, int r) {
      tr[u] = {l, r, 0, a[l], a[l]};
      if (l == r) return;
      int mid = l + r >> 1;
      pushdown(u);
      build(u << 1, l, mid);
      build(u << 1 | 1, mid + 1, r);
      pushup(u);
```

```cpp
    }
    void modify(int u, int l, int r, T k) {
        if (tr[u].l >= l && tr[u].r <= r) {
            tr[u].lazy = min(tr[u].lazy, k);
            tr[u].mx = min(tr[u].mx, k);
            return;
        }
        pushdown(u);
        int mid = tr[u].l + tr[u].r >> 1;
        if (l <= mid) modify(u << 1, l, r, k);
        if (r > mid) modify(u << 1 | 1, l, r, k);
        pushup(u);
    }
    Node query(int u, int l, int r) {
        if (tr[u].l >= l && tr[u].r <= r) return tr[u];
        pushdown(u);
        int mid = tr[u].l + tr[u].r >> 1;
        if (r <= mid) return query(u << 1, l, r);
        if (l > mid) return query(u << 1 | 1, l, r);
        Node t = merge(t, query(u << 1, l, r), query(u << 1 | 1, l, r));
        return t;
    }
};
```

1.11 线段树维护最大子段和

```cpp
template <class T>
struct Seg {
    struct Node { int l, r; T lazy, mx, lmx, rmx, sum; };
    vector<Node> tr; vector<T> a; int n;
    Seg(int n) { n = N + 1; tr.resize(n * 4); a.resize(n); }
    void pushup(int u) {
        tr[u] = merge(tr[u], tr[u << 1], tr[u << 1 | 1]);
    }
    Node merge(Node t, Node l, Node r) {
        t.lmx = max(l.lmx, l.sum + r.lmx);
        t.rmx = max(r.rmx, r.sum + l.rmx);
        t.sum = l.sum + r.sum;
        t.mx = max(max(l.mx, r.mx), l.rmx + r.lmx);
        return t;
    }
    void pushdown(int u) {
        if (tr[u].lazy) {
            tr[u << 1].lazy = tr[u].lazy;
            tr[u << 1 | 1].lazy = tr[u].lazy;
            tr[u].lazy = 0;
        }
    }
    void build(int u, int l, int r) {
        tr[u] = {l, r, 0, a[l], a[l], a[l], a[l]};
        if (l == r) return;
        int mid = l + r >> 1;
        pushdown(u);
        build(u << 1, l, mid);
        build(u << 1 | 1, mid + 1, r);
        pushup(u);
    }
    void modify(int u, int l, int r, T k) {
        if (tr[u].l >= l && tr[u].r <= r) {
            tr[u].lazy = k;
            tr[u].mx = tr[u].lmx = tr[u].rmx = tr[u].sum = k;
            return;
        }
        pushdown(u);
        int mid = tr[u].l + tr[u].r >> 1;
        if (l <= mid) modify(u << 1, l, r, k);
        if (r > mid) modify(u << 1 | 1, l, r, k);
        pushup(u);
    }
    Node query(int u, int l, int r) {
        // debug1(u);
        if (tr[u].l >= l && tr[u].r <= r) return tr[u];
        pushdown(u);
        int mid = tr[u].l + tr[u].r >> 1;
        if (r <= mid) return query(u << 1, l, r);
        if (l > mid) return query(u << 1 | 1, l, r);
        Node t = merge(t, query(u << 1, l, r), query(u << 1 | 1, l, r));
        return t;
    }
};
```

2 字符串

3 杂项

3.1 快读快写

```
template <typename T>
T read() {
    T x = 0;
    char ch = getchar();
    while (ch < '0' || ch > '9') ch = getchar();
    while (ch >= '0' && ch <= '9') x = (x << 3) + (x << 1) + ch - '0', ch = getchar();
    return x;
}

template <typename T>
void write(T x) {
    if (x < 0) putchar('-'), x = -x;
    if (x > 9) write(x / 10);
    putchar(x % 10 + '0');
}
```

3.2 取模

```cpp
template<const int T>
struct ModInt {
    const static int mod = T;
    int x;
    ModInt(int x = 0) : x(x % mod) {}
    ModInt(long long x) : x(int(x % mod)) {}
    int val() {
        return x;
    }
    ModInt operator + (const ModInt &a) const {
        int x0 = x + a.x;
        return ModInt(x0 < mod ? x0 : x0 - mod);
    }
    ModInt operator - (const ModInt &a) const {
        int x0 = x - a.x;
        return ModInt(x0 < 0 ? x0 + mod : x0);
    }
    ModInt operator * (const ModInt &a) const {
        return ModInt(1LL * x * a.x % mod);
    }
    ModInt operator / (const ModInt &a) const {
        return *this * a.inv();
    }
    bool operator == (const ModInt &a) const {
        return x == a.x;
    };
    bool operator != (const ModInt &a) const {
        return x != a.x;
    };
    void operator += (const ModInt &a) {
        x += a.x;
        if (x >= mod) x -= mod;
    }
    void operator -= (const ModInt &a) {
        x -= a.x;
        if (x < 0) x += mod;
    }
    void operator *= (const ModInt &a) {
        x = 1LL * x * a.x % mod;
    }
    void operator /= (const ModInt &a) {
        *this = *this / a;
    }
    friend ModInt operator + (int y, const ModInt &a) {
        int x0 = y + a.x;
        return ModInt(x0 < mod ? x0 : x0 - mod);
    }
    friend ModInt operator - (int y, const ModInt &a) {
        int x0 = y - a.x;
        return ModInt(x0 < 0 ? x0 + mod : x0);
    }
    friend ModInt operator * (int y, const ModInt &a) {
        return ModInt(1LL * y * a.x % mod);
    }
    friend ModInt operator / (int y, const ModInt &a) {
        return ModInt(y) / a;
    }
    friend ostream &operator<<(ostream &os, const ModInt &a) {
        return os << a.x;
    }
    friend istream &operator>>(istream &is, ModInt &t) {
        return is >> t.x;
    }

    ModInt pow(int64_t n) const {
        ModInt res(1), mul(x);
        while(n) {
            if (n & 1) res *= mul;
            mul *= mul;
            n >>= 1;
        }
        return res;
    }

    ModInt inv() const {
        int a = x, b = mod, u = 1, v = 0;
        while (b) {
            int t = a / b;
            a -= t * b;
            swap(a, b);
            u -= t * v;
            swap(u, v);
        }
        if (u < 0) u += mod;
        return u;
    }

};
using mint = ModInt<998244353>;
```

4 数学

4.1 组合数（递推）

```
int C[N][N];

void comb(){
    for(int i = 0; i < N; i++) {
        for(int j = 0; j <= i; j++) {
            if(!j) {
                C[i][j] = 1;
                continue;
            }
```

```
         C[i][j] = C[i - 1][j] + C[i - 1][j - 1];
         C[i][j] %= mod;
      }
   }
}
```

4.2 组合数（逆元）

```
struct comb2{
   vector<int> fac,inf;

   comb2(int n) {
      fac.resize(n + 1);
      inf.resize(n + 1);
      fac[0] = inf[0] = 1;
      for(int i = 1; i <= n; i++){
         fac[i] = fac[i - 1] * i % mod;
         inf[i] = inf[i - 1] * ksm(i, mod - 2) % mod;
      }
   }

   int ksm(int a, int b){ // 快速幂
      int res = 1;
      while(b){
         if(b & 1) res = res * a % mod;
         a = a * a % mod;
         b >>= 1;
      }
      return res;
   }

   int query(int n, int m) {
      int ans;
      ans = fac[n] * inf[n - m] % mod * inf[m] % mod;
      return ans;
   }
 };
```