XCPC - Templates

north-h

Contents

1 数据结构

1.1 树状数组（单点修改，区间查询）

```cpp
template <class T>
struct BIT {
    vector<T> tr;
    int n;
    BIT(int n) : n(n), tr(n) {}
    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）

```cpp
template <class T>
struct BIT {
    vector<T> tr;
    int n;
    BIT(int n) : n(n), tr(n), {}
    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）

```cpp
template <class T>
struct BIT {
```

```
    vector<T> sum1, sum2;
    int n;
    BIT(int n) : n(n), sum1(n + 1), sum2(n + 1) {}
    void add(int x, T k) {
        for(int i = x; i <= n; i += (i & -i))
            sum1[i] += k, sum2[i] += x * k;
    }
    void range_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 range_query(int l, int r) {
        return query(r) - query(l - 1);

    }
};
```

1.4 线段树（朴素）

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

    vector<Node> tr;
    vector<T> a;

    SegmentTree(const vector<T> &arr, const int n) {
        tr.resize(n * 4);
        a = arr;
        build(1, 1, n);
    }

    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
    T 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;
    }
};
```

1.5 线段树（lazy）

```
template<class T>
struct SegmentTree {
    struct Node {
```

```
4       int l, r;
5       T sum, lazy;
6   };
7
8   vector<Node> tr;
9   vector<T> a;
10
11  SegmentTree(const vector<T> &arr, const int n) {
12      tr.resize(n * 4);
13      a = arr;
14      build(1, 1, n);
15  }
16
17  void pushup(int u) {
18      tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
19  }
20
21  void pushdown(int u) {
22      if (tr[u].lazy) {
23          tr[u << 1].sum += tr[u].lazy * (tr[u << 1].r - tr[u << 1].l + 1);
24          tr[u << 1 | 1].sum += tr[u].lazy * (tr[u << 1 | 1].r - tr[u << 1 | 1].l +
                1);
25          tr[u << 1].lazy += tr[u].lazy;
26          tr[u << 1 | 1].lazy += tr[u].lazy;
27          tr[u].lazy = 0;
28      }
29  }
30
31  void build(int u, int l, int r) {
32      tr[u] = {l, r, a[l], 0};
33      if (l == r) return;
34      int mid = l + r >> 1;
35      pushdown(u);
36      build(u << 1, l, mid);
37      build(u << 1 | 1, mid + 1, r);
38      pushup(u);
39  }
40
41  void modify(int u, int l, int r, T k) {
42      if (tr[u].l >= l && tr[u].r <= r) {
43          tr[u].sum += (tr[u].r - tr[u].l + 1) * k;
44          tr[u].lazy += k;
45          return;
46      }
47      pushdown(u);
48      int mid = tr[u].l + tr[u].r >> 1;
49      if (l <= mid) modify(u << 1, l, r, k);
50      if (r > mid) modify(u << 1 | 1, l, r, k);
51      pushup(u);
52  }
53
54  T query(int u, int l, int r) {
55      if (tr[u].l >= l && tr[u].r <= r) return tr[u].sum;
```

```
        pushdown(u);
        T 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;
    }
};
```

1.6 ST 表

```
template<class T>
struct RMQ {
    T n;
    vector<T> arr;
    vector<vector<T>> f, g;
    vector<int> lg2;
    RMQ(const vector<T> &a) :
        n(a.size()), arr(a),
        f(n, vector<T>(log2(n) + 1)),
        g(n, vector<T>(log2(n) + 1)),
        lg2(n + 1) {
        init();
    }
    void init() {
        lg2[0] = -1;
        for(int i = 1; i <= n; i ++)
            lg2[i] = lg2[i >> 1] + 1;
        for (int i = 0; i < n; i++) {
            f[i][0] = arr[i];
            g[i][0] = arr[i];
        }
        for (int j = 1; (1 << j) <= n; j++) {
            for (int i = 0; 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]);
    }
};
```

1.7 二维树状数组

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

    BIT_2D(int n, int m) : n(n), m(m), tr(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);
    }
};
```

2 字符串

2.1 单哈希

```cpp
template<class T, int P, int mod>
struct SH {
    vector<T> p, h;
    string s;
    SH(const string &str) :
        s(" " + str), p(str.size() + 1),
        h(str.size() + 1) {
        p[0] = 1;
        for (int i = 1; i < s.size(); i++) {
            p[i] = (p[i - 1] * P) % mod;
            h[i] = (h[i - 1] * P + s[i]) % mod;
        }
    }
    T get(int l, int r) {
        return (h[r] - h[l - 1] * p[r - l + 1] % mod + mod) % mod;
    }
};
```

2.2 双哈希

```cpp
template<class T, int P1, int mod1, int P2, int mod2>
struct SH {
    vector<T> p1, p2, h1, h2;
    string s;
    SH(const string &str) :
        s(" " + str),
        p1(str.size() + 1),
        h1(str.size() + 1),
        p2(str.size() + 1),
        h2(str.size() + 1) {
        p1[0] = p2[0] = 1;
        for (int i = 1; i < s.size(); i++) {
            p1[i] = (p1[i - 1] * P1) % mod1;
            h1[i] = (h1[i - 1] * P1 % mod1 + s[i]) % mod1;
            p2[i] = (p2[i - 1] * P2) % mod2;
            h2[i] = (h2[i - 1] * P2 % mod2 + s[i]) % mod2;
        }
    }

    T get1(int l, int r) {
        return (h1[r] - (h1[l - 1] * p1[r - l + 1] % mod1) + mod1) % mod1;
    }

    T get2(int l, int r) {
        return (h2[r] - (h2[l - 1] * p2[r - l + 1] % mod2) + mod2) % mod2;
    }

    bool query(int sl, int sr, int el, int er) {
        if(get1(sl, sr) == get1(el, er) && get2(sl, sr) == get2(el, er))return true;
        return false;
    }
};
```

2.3 自然溢出

```cpp
template<class T, int P>
struct SH {
    vector<T> p, h;
    string s;
    SH(const string &str) :
        s(" " + str), p(str.size() + 1),
        h(str.size() + 1) {
        p[0] = 1;
        for (int i = 1; i < s.size(); i++) {
            p[i] = p[i - 1] * P;
            h[i] = h[i - 1] * P + s[i];
        }
    }
    T get(int l, int r) {
        return h[r] - h[l - 1] * p[r - l + 1];
```

```
    }
};
```

2.4 字典树

```
template<class T>
struct Trie {
    T idx = 0;
    vector<vector<T>> s;
    vector<T>cnt;
    Trie(T n): cnt(n, 0), s(n, vector<T>(26, 0)) {}
    void insert(string str) {
        T p = 0;
        for (int i = 0; i < str.size(); i++) {
            T u = str[i] - 'a';
            if (!s[p][u]) s[p][u] = ++idx;
            p = s[p][u];
        }
        cnt[p]++;
    }

    T query(string str) {
        T p = 0;
        for (int i = 0; i < str.size(); i++) {
            T u = str[i] - 'a';
            if (!s[p][u]) return 0;
            p = s[p][u];
        }
        return cnt[p];
    }
};
```

2.5 马拉车

```
template <class T>
struct Manacher {
    vector<T> d;
    string str = "$#";
    Manacher(string s) : d((s.size() + 1) * 2) {
        for(auto i : s) {
            str += i;
            str += '#';
        }
        d[1] = 1;
        for(int i = 2, l, r = 1; i < str.size(); i++) {
            if(i <= r)d[i] = min(d[r + l - i], r - i + 1);
            while(str[i - d[i]] == str[i + d[i]])d[i]++;
            if(i + d[i] - 1 > r)l = i - d[i] + 1, r = i + d[i] - 1;
        }
    }

```

```
    T query() {
        T len = 0;
        for(auto i : d)len = max(len, i);
        return len - 1;
    }
};
```

2.6 正反哈希

```
template <class T, int P, int mod>
struct Palin {
    vector<T> pre, suf, p;
    string str = "$#";
    T n;
    Palin(const string &s) {
        for(auto i : s) {
            str += i;
            str += '#';
        }
        n = (int)str.size() - 1;
        // pre((s.size() + 1) * 2),
        // suf((s.size() + 1) * 2),
        // p((s.size() + 1) * 2)
        pre(n + 1), suf(n + 1), p(n + 1);
        p[0] = 1;
        for(int i = 1, j = n; i <= n; i++, j--) {
            pre[i] = (pre[i - 1] * P % mod + str[i]) % mod;
            suf[i] = (suf[i - 1] * P % mod + str[j]) % mod;
            p[i] = p[i - 1] * P % mod;
        }

    }

    T get(T h[], int l, int r) {
        return (h[r] - h[l - 1] * p[r - l + 1] % mod + mod) % mod;
    }

    bool query(int l, int r) {
        T x = get(pre, l, r);
        T y = get(suf, n + 1 - r, n + 1 - l);
        if(x == y)return true;
        return false;
    }
};
```

3 杂项

3.1 快读快写

```
inline int read() {
    int 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;
}
inline void write(int x) {
    if (x < 0) putchar('-'), x = -x;
    if (x > 9) write(x / 10);
    putchar(x % 10 + '0');
}
```

3.2 取模

```
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) {
```

```
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>;
```