

# XCPC - Templates

为什么不能 V 我 50

## Contents

<b>1</b>	<b>north-h</b>	<b>2</b>
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）	2
1.4	线段树（朴素）	3
1.5	线段树（lazy）	4
1.6	ST 表	5
1.7	单哈希	6
1.8	双哈希	6
1.9	自然溢出	7
1.10	快读快写	7
1.11	取模	8
1.12	二维树状数组	9
1.13	字典树	10

## 1 north-h

### 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) : n(n), tr(n) {}
6      void add(int x, T k) {
7          for(int i = x; i < n; i += (i & -i))
8              tr[i] += k;
9      }
10     T query(int x) {
11         T res = 0;
12         for(int i = x; i; i -= (i & -i))
13             res += tr[i];
14         return res;
15     }
16     T range_query(int l, int r) {
17         return query(r) - query(l - 1);
18     }
19 };
```

### 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) : n(n), tr(n), {}
6      void add(int x, T k) {
7          for(int i = x; i < n; i += (i & -i))
8              tr[i] += k;
9      }
10     void range_add(int l, int r) {
11         add(l, k);
12         add(r + 1, -k);
13     }
14     T query(int x) {
15         T res = 0;
16         for(int i = x; i; i -= (i & -i))
17             res += tr[i];
18         return res;
19     }
20 };
```

### 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(int n) : n(n), sum1(n + 1), sum2(n + 1) {}
6   void add(int x, T k) {
7       for(int i = x; i <= n; i += (i & -i))
8           sum1[i] += k, sum2[i] += x * k;
9   }
10  void range_add(int l, int r, T x) {
11      add(l, x), add(r + 1, -x);
12  }
13  T query(int x) {
14      T res = 0;
15      for(int i = x; i > 0; i -= (i & -i))
16          res += (x + 1) * sum1[i] - sum2[i];
17      return res;
18  }
19  T range_query(int l, int r) {
20      return query(r) - query(l - 1);
21  }
22  }
23  };

```

## 1.4 线段树 (朴素)

```

1   const int N = 100010;
2
3   struct Node {
4       int l, r, sum;
5   } tr[N * 4];
6
7   int a[N];
8
9   void pushup(int u) {
10      tr[u].sum = tr[u << 1].sum + tr[u << 1 | 1].sum;
11  }
12  void build(int u, int l, int r) {
13      tr[u] = {l, r, a[l], 0};
14      if (l == r) return;
15      int mid = l + r >> 1;
16      pushdown(u);
17      build(u << 1, l, mid);
18      build(u << 1 | 1, mid + 1, r);
19      pushup(u);
20  }
21  //单点修改
22  void modify(int u, int x, int k) {
23      if (tr[u].l == tr[u].r) {
24          tr[u].sum += k;
25          return;
26      }
27      int mid = tr[u].l + tr[u].r >> 1;
28      if (x <= mid) modify(u << 1, x, k);

```

```

29     else modify(u << 1 | 1, x, k);
30 }
31 //区间查询
32 int query(int u, int l, int r) {
33     if (tr[u].l >= l && tr[u].r <= r) return tr[u].sum;
34     int sum = 0;
35     int mid = tr[u].l + tr[u].r >> 1;
36     if (l <= mid) sum += query(u << 1, l, r);
37     if (r > mid) sum += query(u << 1 | 1, l, r);
38     return sum;
39 }

```

## 1.5 线段树 (lazy)

```

1  template<typename T, int N>
2  struct SegmentTree {
3      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) {
12         tr.resize(N * 4);
13         a = arr;
14         build(1, 1, a.size());
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 +
25                 1);
26             tr[u << 1].lazy += tr[u].lazy;
27             tr[u << 1 | 1].lazy += tr[u].lazy;
28             tr[u].lazy = 0;
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;
56     pushdown(u);
57     T sum = 0;
58     int mid = tr[u].l + tr[u].r >> 1;
59     if (l <= mid) sum += query(u << 1, l, r);
60     if (r > mid) sum += query(u << 1 | 1, l, r);
61     return sum;
62 }
63 };

```

## 1.6 ST 表

```

1  template<typename T>
2  struct RMQ {
3      int n;
4      vector<T> arr;
5      vector<vector<T>> f, g;
6      vector<int> lg2;
7
8      void init() {
9          for (int i = 0; i < n; i++) {
10             f[i][0] = arr[i];
11             g[i][0] = arr[i];
12         }
13         for (int j = 1; (1 << j) <= n; j++) {
14             for (int i = 0; i + (1 << j) - 1 < n; i++) {
15                 f[i][j] = max(f[i][j - 1], f[i + (1 << (j - 1))][j - 1]);
16                 g[i][j] = min(g[i][j - 1], g[i + (1 << (j - 1))][j - 1]);
17             }
18         }
19     }
20
21     RMQ(const vector<T>& arr) :
22         n(arr.size()), arr(arr),
23         f(n, vector<T>(log2(n) + 1)),

```

```

24         g(n, vector<T>(log2(n) + 1)),
25         lg2(n + 1) {
26     lg2[0] = -1;
27     for(int i = 1; i <= n; i++)
28         lg2[i] = lg2[i >> 1] + 1;
29     init();
30 }
31
32 T query_max(int l, int r) {
33     int k = lg2[r - l + 1];
34     return max(f[l][k], f[r - (1 << k) + 1][k]);
35 }
36
37 T query_min(int l, int r) {
38     int k = lg2[r - l + 1];
39     return min(g[l][k], g[r - (1 << k) + 1][k]);
40 }
41 };

```

## 1.7 单哈希

```

1  template<int P, int mod, int N>
2  struct Hash {
3      long long p[N], h[N];
4      string s;
5
6      Hash(const string &str) : s(" " + str) {
7          p[0] = 1;
8          for (int i = 1; i < s.size(); i++) {
9              p[i] = (p[i - 1] * P) % mod;
10             h[i] = (h[i - 1] * P % mod + s[i]) % mod;
11         }
12     }
13
14     long long get(int l, int r) {
15         return (h[r] - (h[l - 1] * p[r - l + 1] % mod) + mod) % mod;
16     }
17 };

```

## 1.8 双哈希

```

1  template<int P1, int mod1, int P2, int mod2, int N>
2  struct Hash {
3      long long p1[N], p2[N], h1[N], h2[N];
4      string s;
5
6      Hash(const string &str) : s(" " + str) {
7          p1[0] = p2[0] = 1;
8          for (int i = 1; i <= s.size(); i++) {
9              p1[i] = (p1[i - 1] * P1) % mod1;
10             h1[i] = (h1[i - 1] * P1 % mod1 + s[i]) % mod1;

```

```

11         p2[i] = (p2[i - 1] * P2) % mod2;
12         h2[i] = (h2[i - 1] * P2 % mod2 + s[i]) % mod2;
13     }
14 }
15
16 long long get1(int l, int r) {
17     return (h1[r] - (h1[l - 1] * p1[r - l + 1] % mod1) + mod1) % mod1;
18 }
19
20 long long get2(int l, int r) {
21     return (h2[r] - (h2[l - 1] * p2[r - l + 1] % mod2) + mod2) % mod2;
22 }
23 };

```

## 1.9 自然溢出

```

1 template<int P, int mod, int N>
2 struct Hash {
3     typedef unsigned long long ULL;
4     ULL p[N], h[N];
5     string s;
6
7     Hash(const string &str) : s(" " + str) {
8         p[0] = 1;
9         for (int i = 1; i <= s.size(); i++) {
10             p[i] = p[i - 1] * P % mod;
11             h[i] = h[i - 1] * P % mod + s[i];
12             h[i] %= mod;
13         }
14     }
15
16     ULL get(int l, int r) {
17         return (h[r] - h[l - 1] * p[r - l + 1] % mod + mod) % mod;
18     }
19 };

```

## 1.10 快读快写

```

1 inline int read() {
2     int x = 0;
3     char ch = getchar();
4     while (ch < '0' || ch > '9') ch = getchar();
5     while (ch >= '0' && ch <= '9') x = (x << 3) + (x << 1) + ch - '0', ch = getchar();
6     return x;
7 }
8 inline void write(int x) {
9     if (x < 0) putchar('-'), x = -x;
10    if (x > 9) write(x / 10);
11    putchar(x % 10 + '0');
12 }

```

## 1.11 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>;

```

## 1.12 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) : n(n), m(m), tr(n + 1, vector<T>(m + 1)) {}
7
8      int lowbit(int x) { return x & (-x); }
9
10     void add(int x, int y, T k) {
11         for (int i = x; i <= n; i += lowbit(i))

```

```

12         for (int j = y; j <= m; j += lowbit(j))
13             tr[i][j] += k;
14     }
15
16     T query(int x, int y) {
17         T res = 0;
18         for (int i = x; i; i -= lowbit(i))
19             for (int j = y; j; j -= lowbit(j))
20                 res += tr[i][j];
21         return res;
22     }
23
24     T query(int x1, int y1, int x2, int y2) {
25         return query(x2, y2) - query(x2, y1-1) - query(x1-1, y2) + query(x1-1, y1-1);
26     }
27 };

```

### 1.13 字典树

```

1  template<class T>
2  struct Trie {
3      T idx = 0;
4      vector<vector<T>> s;
5      vector<T>cnt;
6      Trie(T n): cnt(n, 0), s(n, vector<T>(26, 0)) {}
7      void insert(string str) {
8          T p = 0;
9          for (int i = 0; i < str.size(); i++) {
10             T u = str[i] - 'a';
11             if (!s[p][u]) s[p][u] = ++idx;
12             p = s[p][u];
13         }
14         cnt[p]++;
15     }
16
17     T query(string str) {
18         T p = 0;
19         for (int i = 0; i < str.size(); i++) {
20             T u = str[i] - 'a';
21             if (!s[p][u]) return 0;
22             p = s[p][u];
23         }
24         return cnt[p];
25     }
26 };

```