数据结构

```
/* 单点修改区间查询 */
class BIT{
  public: int d[CN];
    void add(int p, int x) {while(p <= n) d[p] += x, p += p & (-p);}
    int qu(int p) {int r = 0; while(p) r += d[p], p -= p & (-p); return r;}
    int qu(int l, int r) {return qu(r) - qu(l - 1);}
} ;

/* 区间修改区间查询 */
class SGT{
  public: int d[CN << 2], tag[CN << 2];
    #define lc k << 1
    #define rc k << 1 | 1
    void pd(int l, int r, int m, int k){
        d[lc] += (m - l + 1) * tag[k], d[rc] += (r - m) * tag[k];
        tag[lc] += tag[k], tag[rc] += tag[k], tag[k] = 0;
    }
    void pu(int k){
        d[k] = d[lc] + d[rc];
    }
    void md(int l, int r, int k, int s, int t, int x){
        if(s <= l && r <= t){
            d[k] += (r - l + 1) * x, tag[k] += x;
            return;
        }
        int m = (l + r) >> 1; if(tag[k]) pd(l, r, m, k);
        if(s <= m) md(l, m, lc, s, t, x);
        if(m < t) md(m + 1, r, rc, s, t, x);
        pu(k);
    }
    int qu(int l, int r, int k, int s, int t){
        if(s <= l && r <= t) return d[k];
        int m = (l + r) >> 1, res = 0; if(tag[k]) pd(l, r, m, k);
        if(s <= m) res += qu(l, m, lc, s, t);
        if(m < t) res += qu(m + 1, r, rc, s, t);
        return res;
    }
} ;

/* 线段覆盖 */
int d[CN * 20], ct[CN * 20], toty[CN], ty;
void upd(int l, int r, int u){
    int ls = u << 1, rs = u << 1 | 1;
    if(ct[u]) d[u] = toty[r] - toty[l - 1]; else d[u] = d[ls] + d[rs];
}
void mod(int l, int r, int u, int x, int y, int k){
    if(x <= l && r <= y){
        ct[u] += k; upd(l, r, u);
        return;
    }
    int mid = (l + r) >> 1;
```

```
    if(x <= mid) mod(l, mid, u << 1, x, y, k);
    if(mid < y) mod(mid + 1, r, u << 1 | 1, x, y, k);
    upd(l, r, u);
}
int qu(int l, int r, int u, int x, int y){
    if(ct[u]) return toty[y] - toty[x - 1];
    if(x <= l && r <= y) return d[u];
    int mid = (l + r) >> 1, ans = 0;
    if(x <= mid) ans = qu(l, mid, u << 1, x, min(y, mid));
    if(mid < y) ans += qu(mid + 1, r, u << 1 | 1, max(x, mid + 1), y);
    return ans;
}

/* 静态主席树 */
int d[CN * 50], ch[CN * 50][2], rt[CN], idx;
void mod(int l, int r, int p, int &u, int x){
    if(!u) u = ++idx;
    if(l == r) {d[u] = d[p] + 1; return;}
    int mid = (l + r) >> 1;
    if(x <= mid) ch[u][1] = ch[p][1], mod(l, mid, ch[p][0], ch[u][0], x);
    else ch[u][0] = ch[p][0], mod(mid + 1, r, ch[p][1], ch[u][1], x);
    d[u] = d[ch[u][0]] + d[ch[u][1]];
}
int qu(int l, int r, int p, int u, int k){
    if(d[u] - d[p] < k) return 2e9;
    if(l == r) return val[l];
    int mid = (l + r) >> 1, t = d[ch[u][0]] - d[ch[p][0]];
    if(t >= k) return qu(l, mid, ch[p][0], ch[u][0], k);
    return qu(mid + 1, r, ch[p][1], ch[u][1], k - t);
}
```