分块

Idea：将长度为 n 的序列分为长度为 $\sqrt{n}$ 的 $\sqrt{n}$ 块，针对每一块维护一些信息。修改或询问时，暴力修改/询问块外的数，利用维护的信息修改/询问块内的数。

Complexity：$O(n\sqrt{n}+q\sqrt{n})$

Code： `L[i],R[i]` 分别表示块 i 的左右端点位置；

`belong[i]` 表示第 i 个数所属块的编号；

`sq` 为一个块的大小，一共有 `belong[n]` 个块。

```
int belong[N], sq;
int L[SQRN], R[SQRN];
inline void init(){
    sq = sqrt(n);
    for(int i = 1; i <= n; i++)
        belong[i] = (i - 1) / sq + 1;
    for(int i = 1; i <= belong[n]; i++){
        L[i] = (i - 1) * sq + 1;
        R[i] = i * sq;
    } R[belong[n]] = n;
}
inline void maintain(int x){
    // maintain info of block x after bf
}
inline void modify(int l, int r, int val){
    for(int i = l; i <= min(R[belong[l]], r); i++){
        // modify left-side part with bf
    }
    maintain(belong[l]);
    if(belong[l] != belong[r]){
        for(int i = L[belong[r]]; i <= r; i++){
            // modify right-side part with bf
        }
        maintain(belong[r]);
    }
    for(int i = belong[l] + 1; i < belong[r]; i++){
        // modify the whole block
    }
}
inline int query(int l, int r, int x){
    int res = 0;
    for(int i = l; i <= min(R[belong[l]], r); i++){
        // query left-side part with bf
    }
    if(belong[l] != belong[r]){
        for(int i = L[belong[r]]; i <= r; i++){
            // query right-side part with bf
        }
    }
    for(int i = belong[l] + 1; i < belong[r]; i++){
        // query the whole block
    }
    return res;
}
```