**分块**

树状数组和线段树虽然非常方便，但维护的信息必须满足信息合并特性（如区间可加、可减），若不满足此特性，则不可以使用树状数组和线段树。**分块算法**可以维护一些线段树维护不了的内容，它其实就是**优化过后的暴力算法**。

分块可以解决几乎所有区间更新和区间查询问题，但效率相对于线段树等数据结构要差一些。

分块算法是将所有数据都分为若干块，维护块内信息，使得块内查询为O(1)时间，而总询问可被看作若干块询问的的总和。

分块算法将长度为n的序列分成若干块，每一块都有k个元素，最后一块可能少于k个元素。为了使时间复杂度均摊，通常将块的大小设为k=，用pos[i]表示第i个位置所属的块，对每个块都进行信息维护。

分块可以解决以下问题：

**• 单点更新：**一般先将对应块的懒标记下传，再暴力更新块的状态，时间复杂度为O()。

**• 区间更新：**若区间更新横跨若干块，则只需对完全覆盖的块打上懒标记，最多需要修改两端的两个块，对两端剩余的部分暴力更新块的状态。每次更新都最多遍历个块，遍历每个块的时间复杂度都是O(1)，两端的两个块暴力更新次，总的时间复杂度是O()。

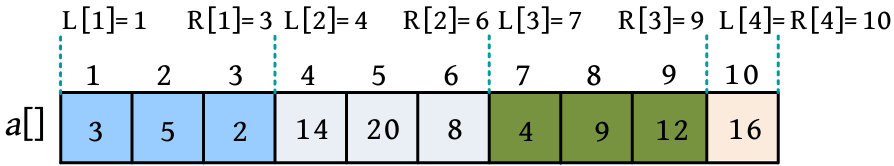
**• 区间查询：**和区间更新类似，对中间跨过的整个块直接利用块存储的信息统计答案，对两端剩余的部分可以暴力扫描统计。时间复杂度和区间修改一样，也是O()。

将整个段分成多个块后进行修改或查询时，对完全覆盖的块直接进行修改，像线段树一样标记或累加；对两端剩余的部分进行暴力修改。

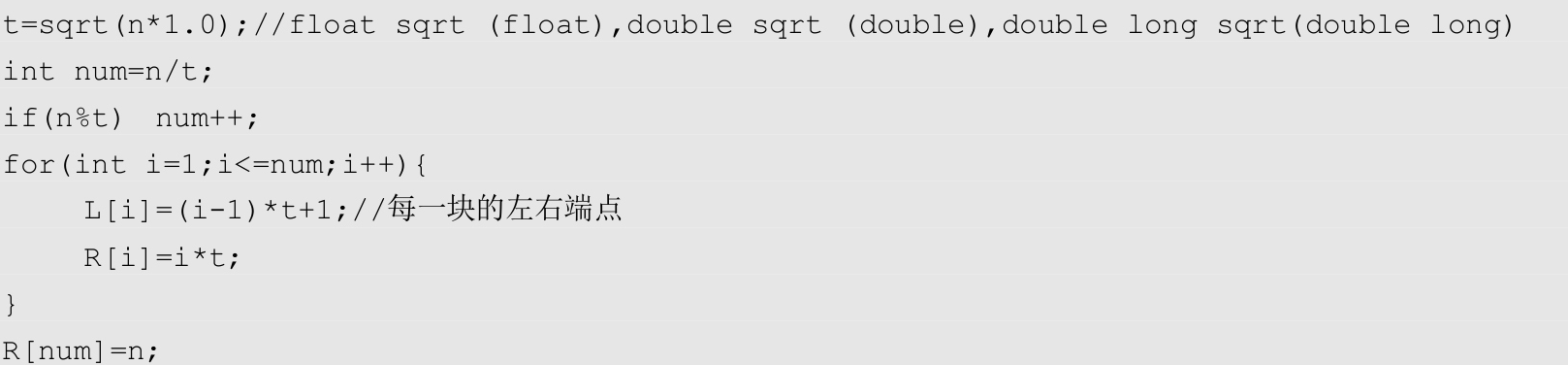
分块算法遵循“**大段维护、局部朴素**”的原则。

**1. 预处理**

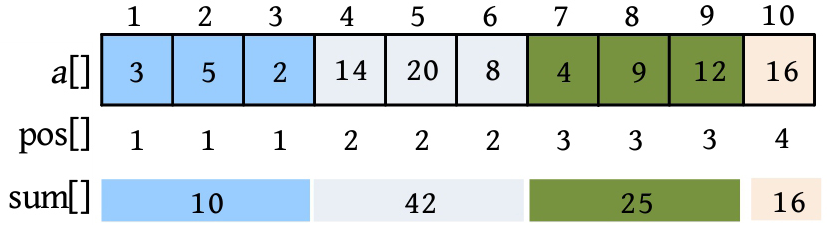
（1）将序列分块，然后将每个块都标记左右端点L[i]和R[i]，对最后一块需要特别处理。n=10，t==3，每3个元素为一块，一共分为4块，最后一块只有一个元素。



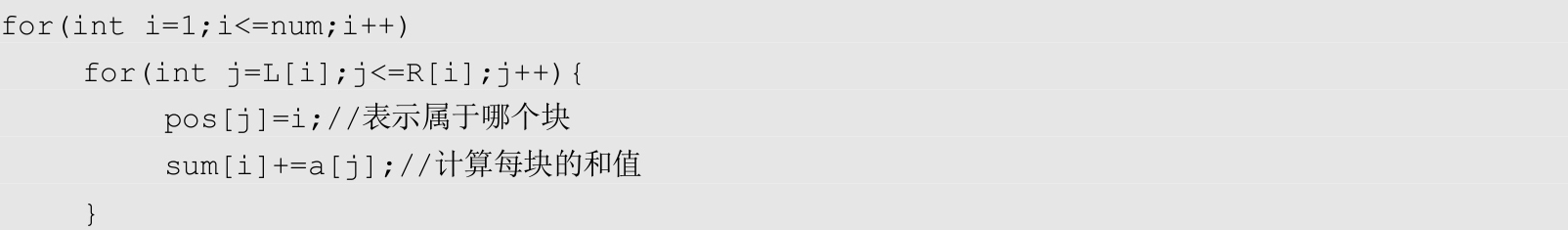
**算法代码：**



1. 用pos[]标记每个元素所属的块，用sum[]累加每一块的和值。



**算法代码：**



**2. 区间更新**

区间更新，例如将[l, r]区间的元素都加上d。

（1）求l和r所属的块，p=pos[l]，q=pos[r]。

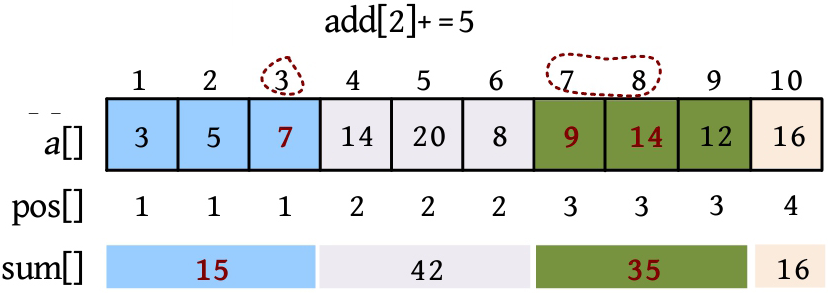
（2）若属于同一块（p=q），则对该区间的元素进行暴力修改，同时更新该块的和值。

（3）若不属于同一块，则对中间完全覆盖的块打上懒标记，add[i]+=d，对首尾两端的元素进行暴力修改。

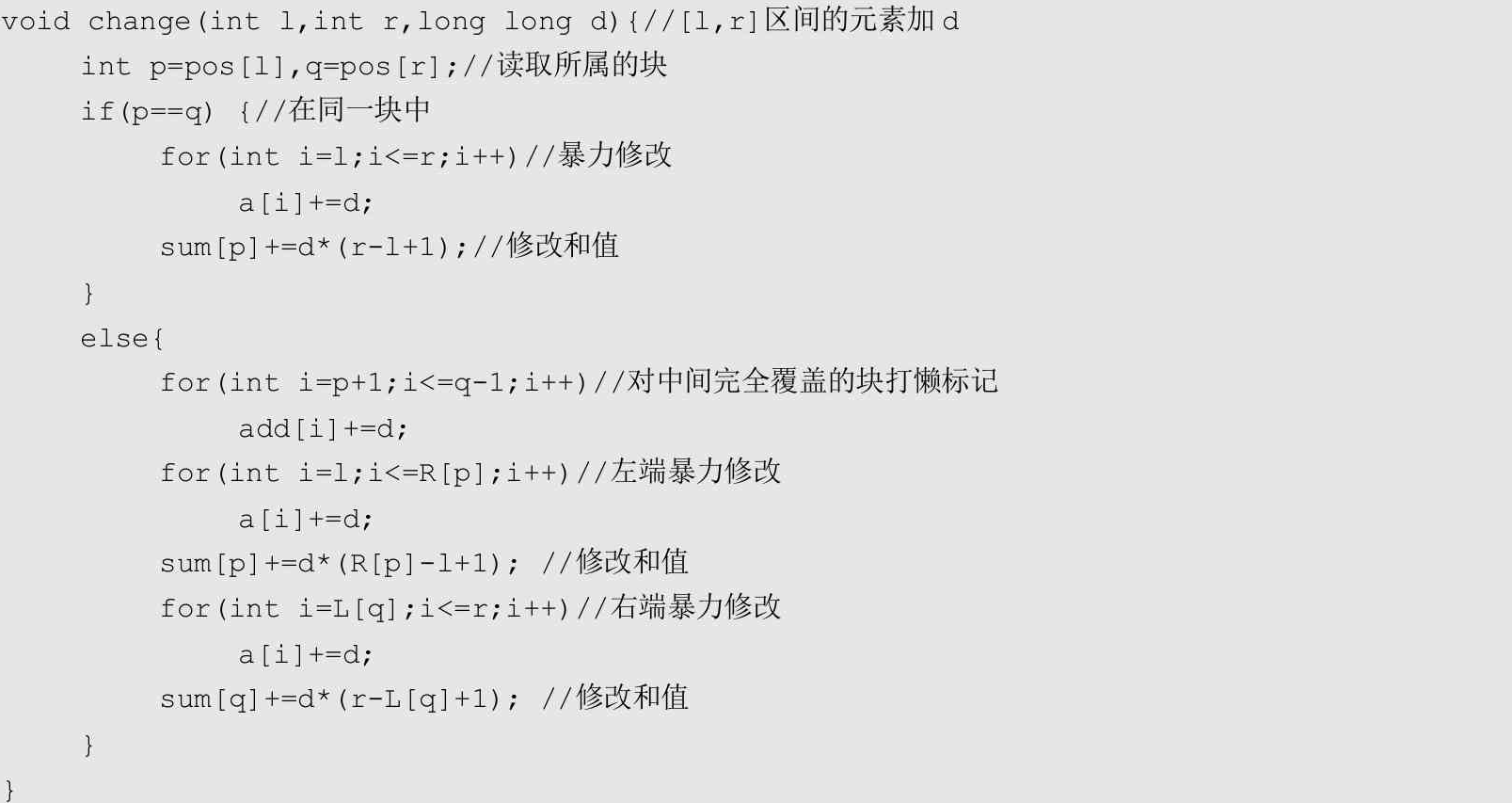
例如，将[3, 8]区间的元素都加上5，操作过程：

① 读取3和8所属的块p=pos[3]=1，q=pos[8]=3，不属于同一块，中间的完整块[p+1, q-1]为第2块，为该块打上懒标记add[2]+=5；

② 对首尾两端的元素（下标3、7、8）进行暴力修改，并修改和值。



**算法代码：**



**3. 区间查询**

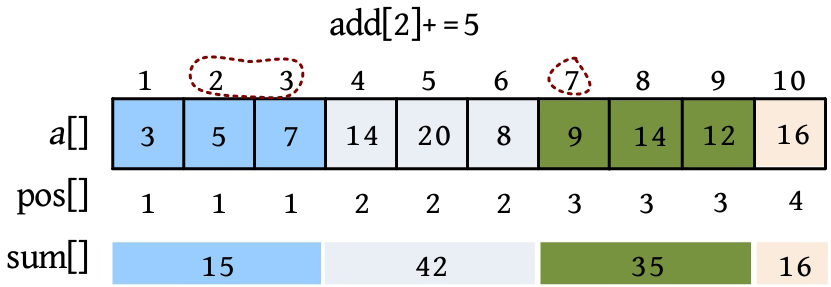
区间查询，例如查询[l, r]区间的元素和值。

（1）求l和r的所属块，p=pos[l]，q=pos[r]。

（2）若属于同一块（p=q），则对该区间的元素进行暴力累加，然后加上懒标记上的值。

（3）若不属于同一块，则对中间完全覆盖的块累加sum[]值和懒标记上的值，然后对首尾两端暴力累加元素值及懒标记值。

例如，查询[2, 7]区间的元素和值，操作过程：①读p=pos[2]=1，q=pos[7]=3，不属于同一块，则中间的完整块[p+1, q-1]为第2块，ans+=sum[2]+add[2]×(R[2]-L[2]+1)=42+5×3=57；②对首尾两端的元素暴力累加元素值及懒标记值。此时懒标记add[1]=add[3]=0，ans+=5+7+add[1]×(3-2+1)+9+add[3]×(7-7+1)=78。



**算法代码：**

