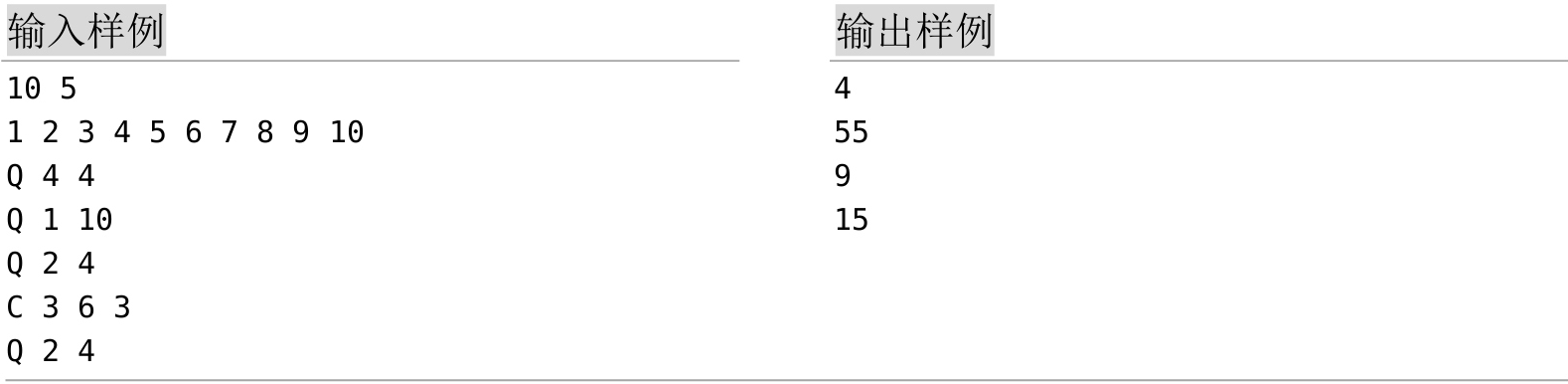
**POJ3648**

**题目描述（POJ3648）：**有N个整数A1, A2, …, AN，需要对其进行两种操作，一种操作是对给定区间中的每个数都添加一个给定的数，另一种操作是查询给定区间中数的总和。

**输入：**第1行包含两个数N和Q（1≤N，Q≤105）；第2行包含N个数，为A1, A2, …, AN的初始值（-109≤Ai≤109）；接下来的Q行，每行都表示一种操作，“C a b c”表示将Aa, Aa+1, …, Ab中的每一个数都加c（-104≤c≤104），“Q a b”表示查询Aa, Aa+1, …, Ab的总和。

**输出：**对每个查询，都单行输出区间和的值。



**提示：**总和可能超过32位整数的范围。

**题解：**本题有两种操作：区间更新和区间查询，可采用用分块算法解决。

**算法设计如下：**

（1）分块预处理。将序列分块，然后对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；标记每个元素所属的块，累加每一块的和值。

（2）区间更新。首先取l和r所属的块，p=pos[l]，q=pos[r]；若属于同一块，则对该区间的所有元素都进行暴力修改，同时更新该块的和值。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；对首尾两端的元素暴力修改即可。

（3）区间查询。首先取l和r所属的块，p=pos[l]，q=pos[r]；若属于同一块，则对该区间的所有元素都进行暴力累加，然后加上懒标记上的值。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，然后对首尾两端的元素暴力累加元素值及懒标记值。

**POJ1019**

**题目描述（POJ1019）：**给出单个正整数i，编写程序以找到位于数字组S1, S2, …, Sk序列中第i位上的数字。每个组Sk都由一系列正整数组成，范围为1～k，一个接一个地写入。序列的前80位数字如下：



**输入：**第1行包含一个整数t（1≤t≤10），表示测试用例的数量。每个测试用例后都跟一行，包含单个整数i（1≤i≤2, 147, 483, 647）。

**输出：**对每个测试用例，都单行输出第i位上的数字。

**题解：**在测试用例中，序列的第8位和第3位都是2：

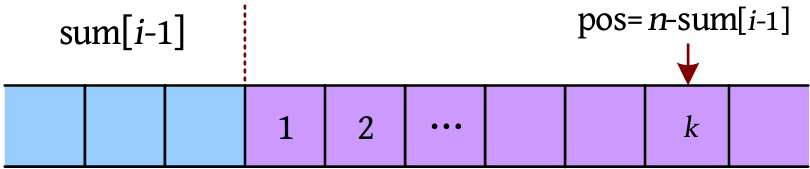


将每个组都看作一个分块，每个组（分块）的长度都为a[i]：当组内的每个数都由一位数字组成时，当前组的长度等于前一组的长度+1；当组内出现两位数10～99时，当前组的长度等于前一组的长度+2，以此类推。

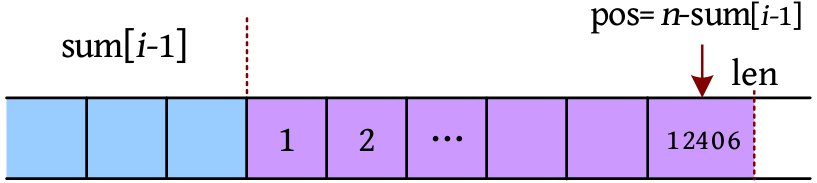
• 1 12 123 1234 12345 123456 1234567 12345678 123456789　前一组的长度+1

• 12345678910 1234567891011 123456789101112……　前一组的长度+2a[i]为第i块的长度，sum[i]为前i（包括i）块的总长度。

例如，查询第n位上的数字，首先定位到第i块，然后在当前块内查找具体的数k。



k可能是多位数，例如k=12406，如下图所示。



第pos位的数字应为k/10len-pos=124，124%10=4。

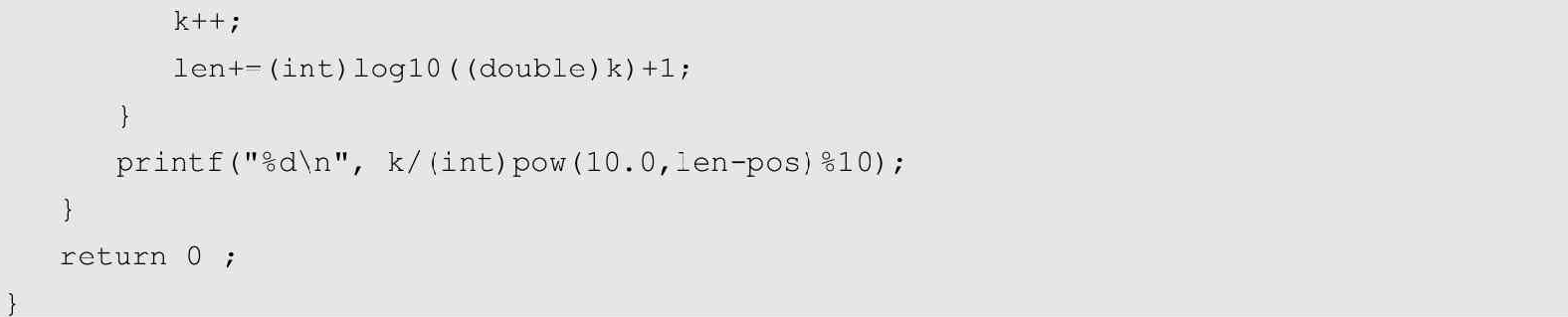
**1. 算法设计**

（1）计算每一块的长度a[i]及前i块的总长度sum[i]。

（2）定位到第i块，在块内查找第pos位所在的数k。

（3）数k有可能是多位数，第pos位为k/(int)pow(10.0, len-pos)%10。

**2. 算法实现**

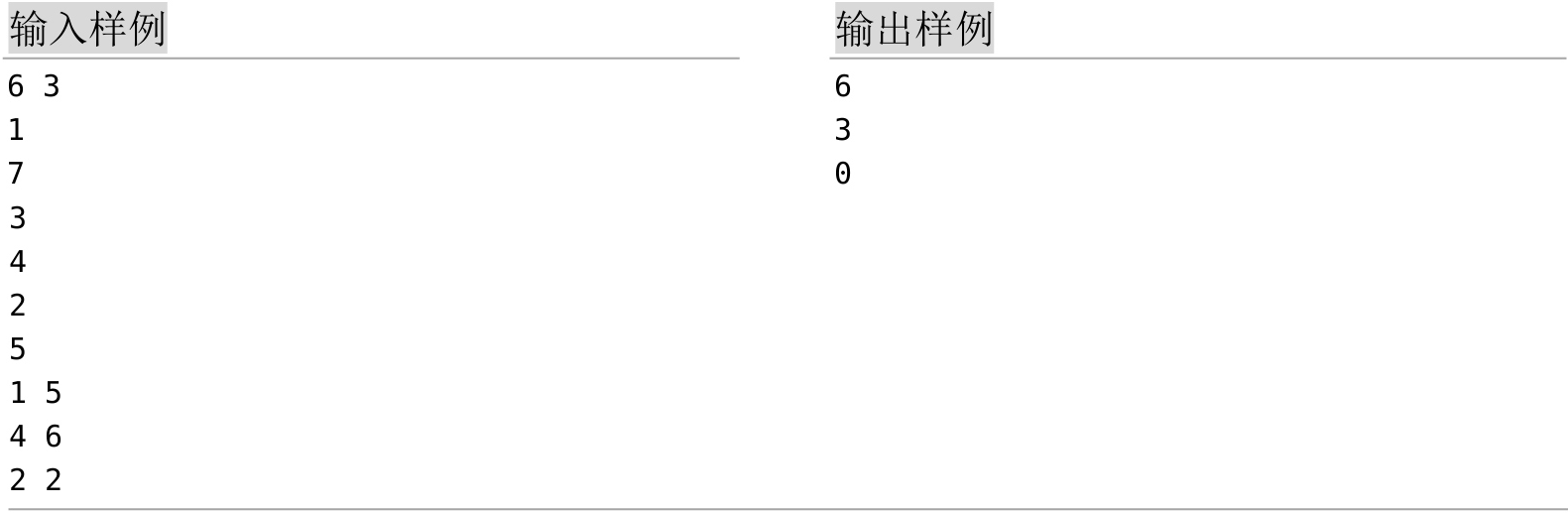


**POJ3264**

**题目描述（POJ3264）：**每天挤奶时，约翰的N头奶牛（1≤N≤50,000）都以相同的顺序排队。他挑选一系列连续的奶牛来玩游戏。为了让所有奶牛都玩得开心，它们的高度差异不应太大。约翰列出了Q组（1≤Q≤200,000）奶牛和它们的高度（1≤height≤1,000,000）。他希望确定每个小组中最高和最矮的奶牛之间的高度差异。

**输入：**第1行包含两个整数N和Q。接下来N行，每行都包含一个整数，表示奶牛的高度。最后Q行，每行都包含两个整数A和B（1≤A≤B≤N），代表从A到B的奶牛范围。

**输出：**输出Q行，每行都包含一个整数，表示该范围内最高和最矮奶牛的高度差。



**题解：**本题是典型的区间最值查询问题，可采用线段树、ST或分块解决。

**1. 算法设计**

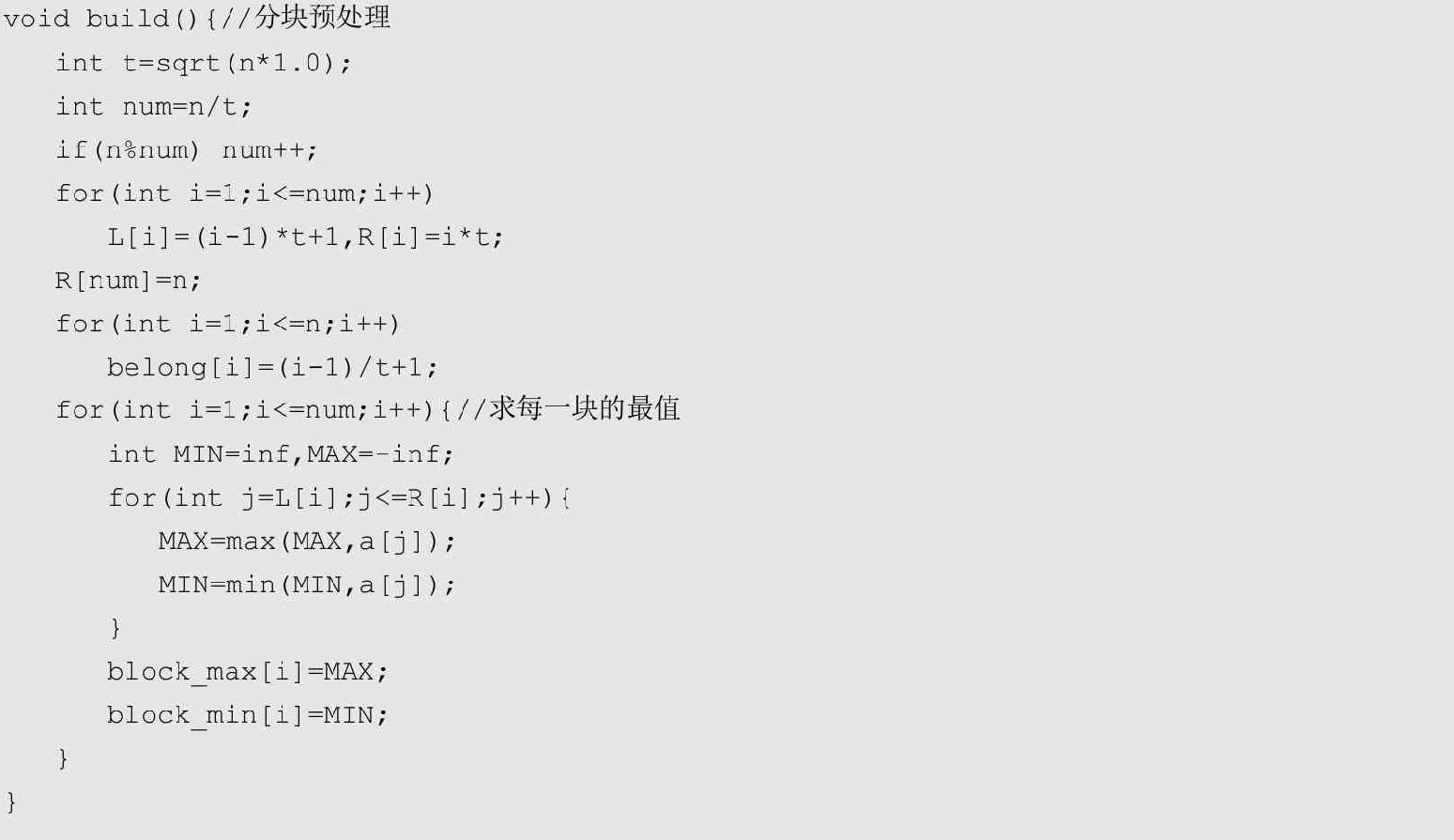
（1）分块。划分块，记录每个元素所属的块，以及每一块的左右端点下标、最大值和最小值。

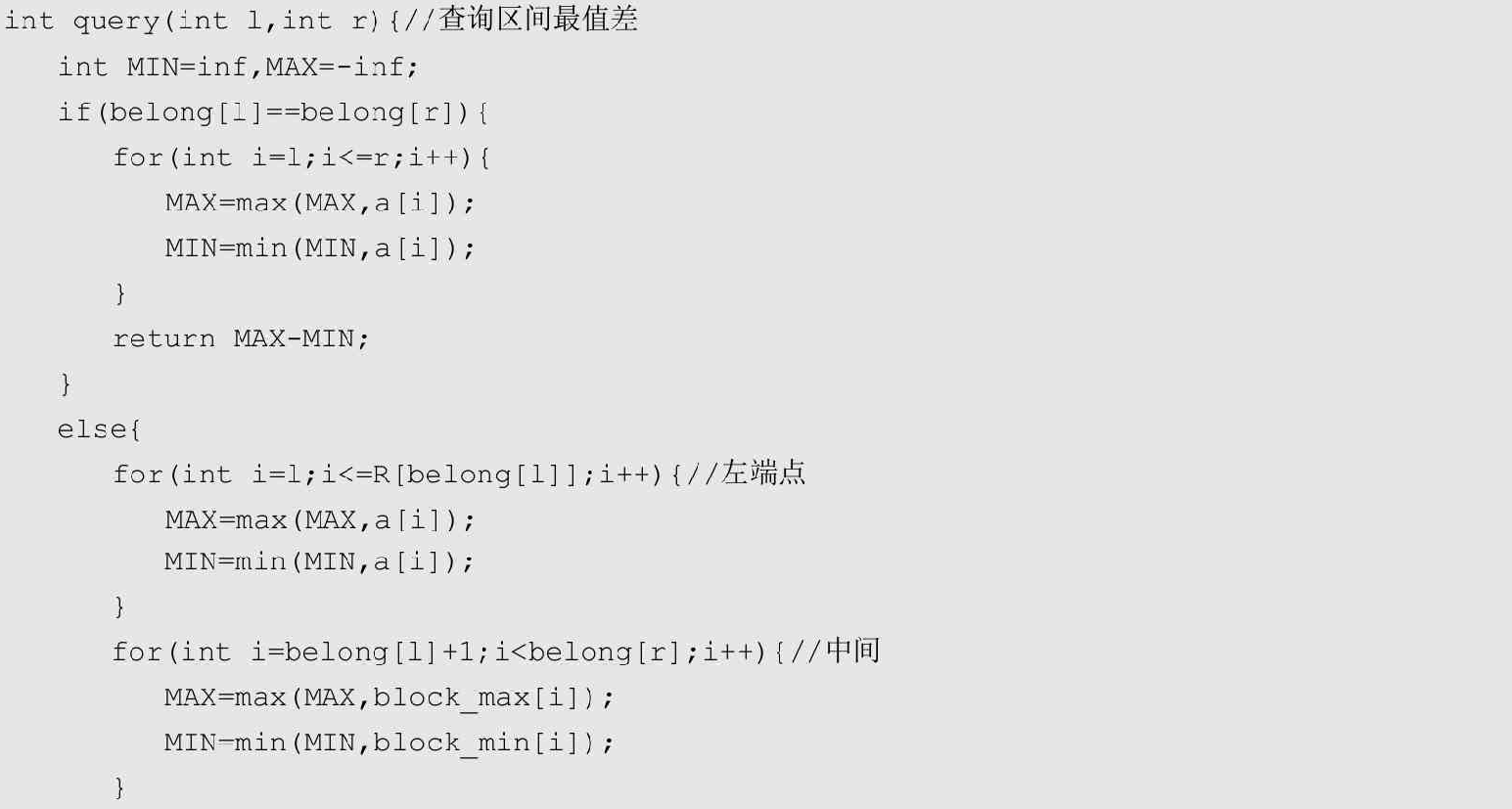
（2）查询。查询[l, r]区间最大值和最小值的差值。

• 若该区间属于同一块，则暴力统计最大值和最小值，返回两者的差值。

• 若该区间包含多个块，则统计中间每个块的最大值和最小值，然后暴力统计左端点和右端点的最大值和最小值，返回两者的差值。

**2. 算法实现**



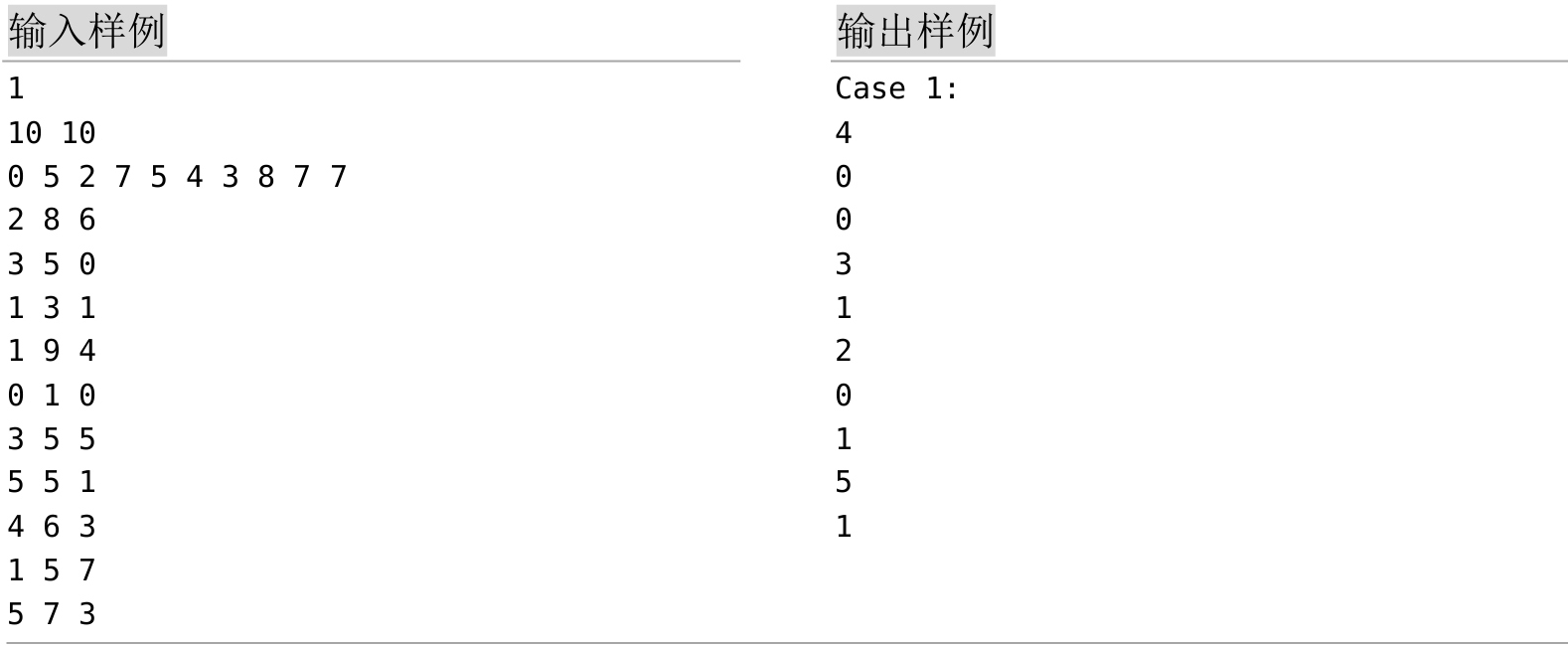


**HDU4417**

**题目描述（HDU4417）：**可怜的公主陷入困境，马里奥需要拯救他的情人。把通往城堡的道路视为一条线（长度为n），在每个整数点i上都有一块高度为hi的砖，马里奥可以跳的最大高度是H，求他在[L, R]区间可以跳过多少砖块。

**输入：**第1行是整数T，表示测试用例的数量。每个测试用例的第1行都包含两个整数n、m（1≤n，m≤105），n是道路的长度，m是查询的数量。下一行包含n个整数，表示每个砖的高度（范围是[0, 109]）。接下来的m行，每行都包含三个整数L、R、H（0≤L≤R<n，0≤H≤109）。

**输出：**对每种情况都输出“Case X：”（X是从1开始的案例编号），后跟m行，每行都包含一个整数。第i个整数是第i个查询中马里奥跳过的砖块数。



**题解：**本题为区间查询问题，查询[l, r]区间小于或等于h的元素个数，可以采用分块的方法解决。

**1. 算法设计**

（1）分块。划分块并对每一块进行非递减排序。在辅助数组temp[ ]上排序，原数组不变。

（2）查询。查询[l, r]区间小于或等于h的元素个数。

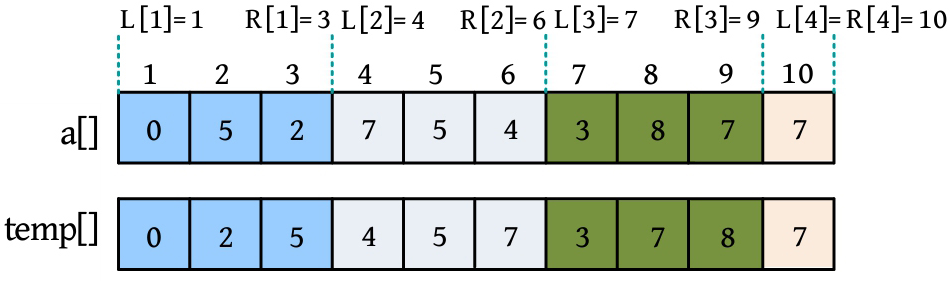
• 若该区间属于同一块，则暴力累加块内小于或等于h的元素个数。

• 若该区间包含多个块，则累加中间每一块小于或等于h的元素个数，此时可以用upper\_bound()函数统计，然后暴力累加左端和右端小于或等于h的元素个数。

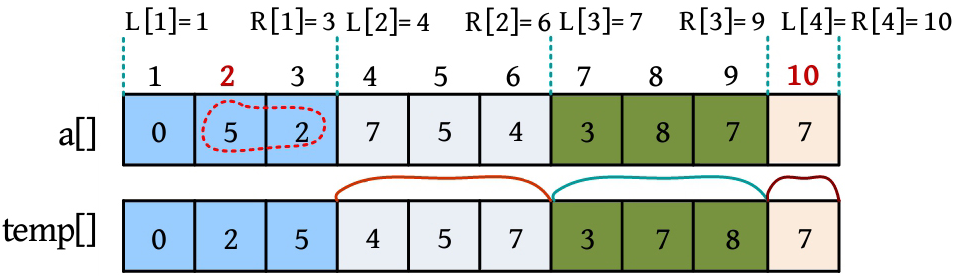
**2. 图解**

根据测试用例的输入数据，分块算法的求解过程如下。

（1）分块。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。原数组a[ ]和每一块排序后的辅助数组temp[ ]如下图所示。

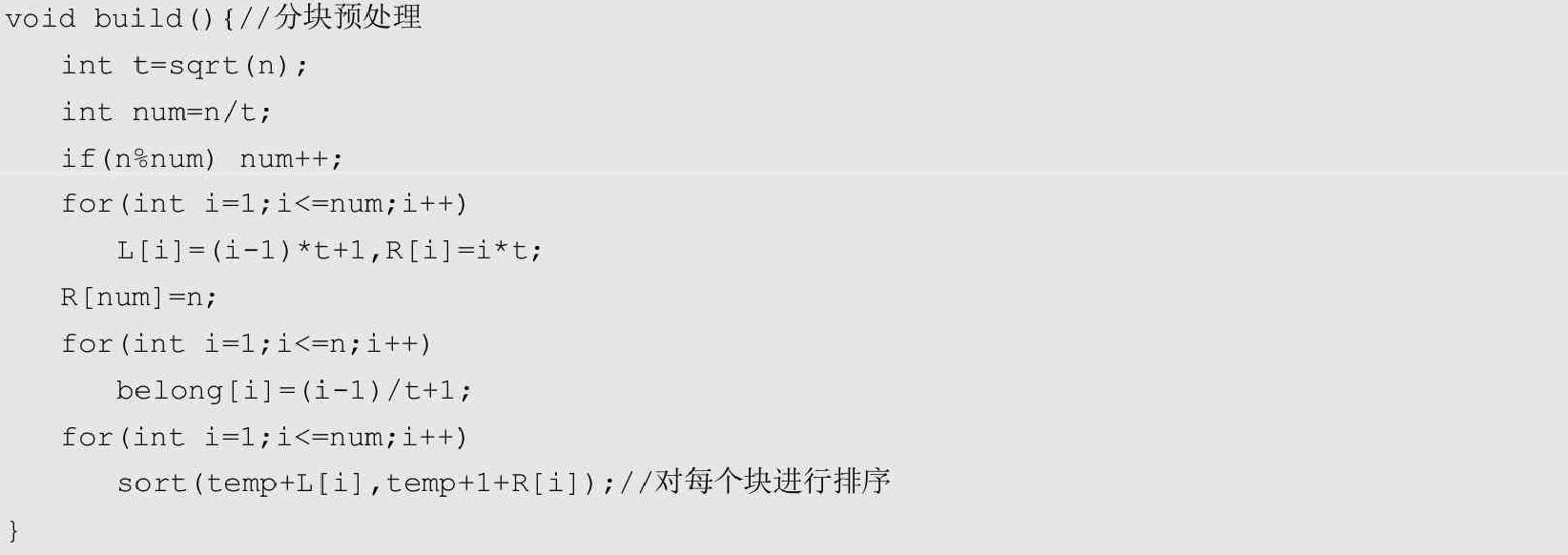


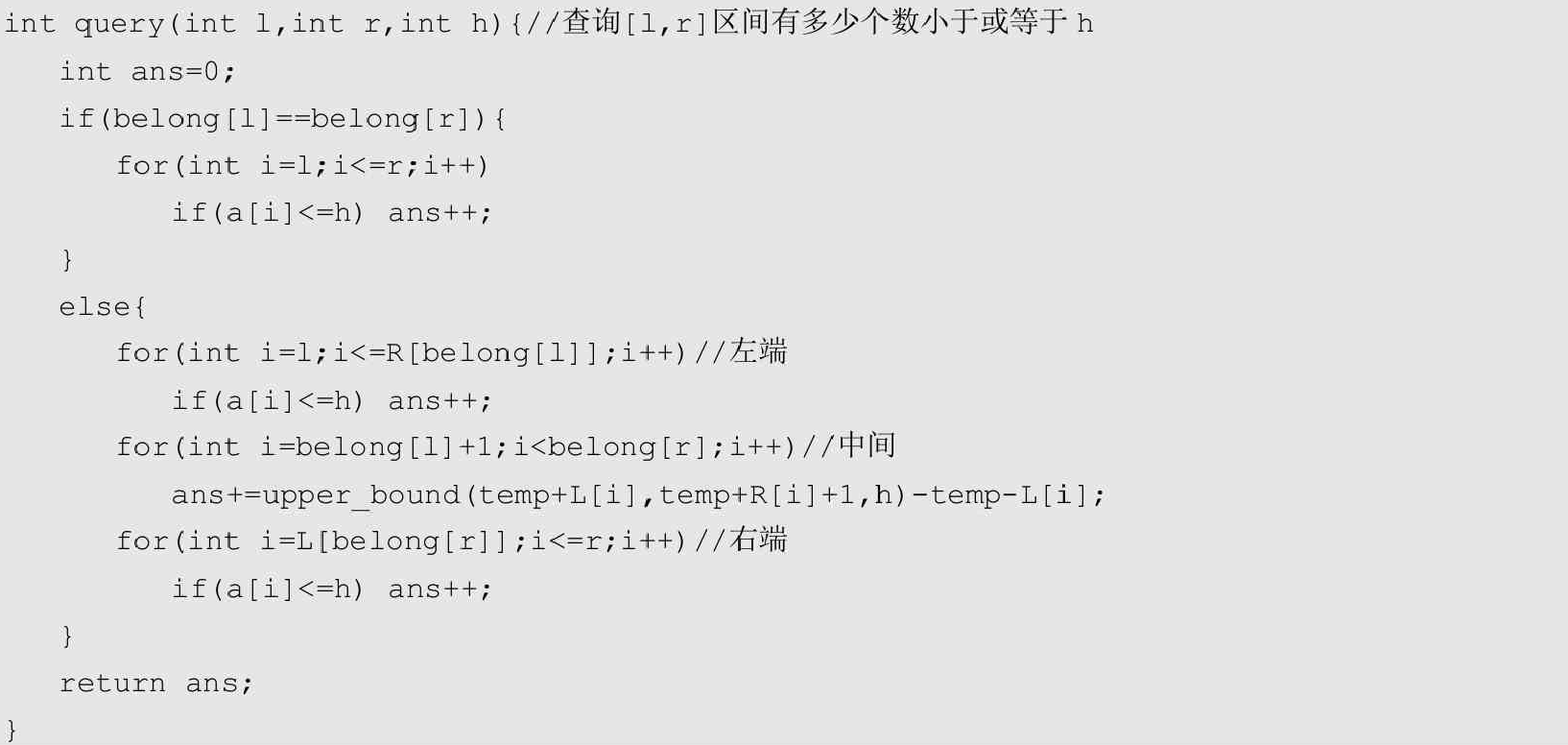
1. 查询。1 9 4：因为题目中的下标从0开始，上图中的下标从1开始，所以实际上是查询[2, 10]区间高度小于或等于4的元素个数。[2, 10]区间跨4个块，左端第1个块没有完全包含，需要暴力统计a[2]、a[3]小于或等于4的元素。后面3个块是完整的块，对完整的块可以直接用upper\_bound()函数在temp数组中统计，该函数利用有序性进行二分查找，效率较高。



**2. 算法实现**

upper\_bound( begin, end, num)：从数组的begin位置到end-1位置二分查找第1个大于num的数字，若找到，则返回该数字的地址，否则返回end。将返回的地址减去起始地址begin，即可得到小于或等于num的元素个数。



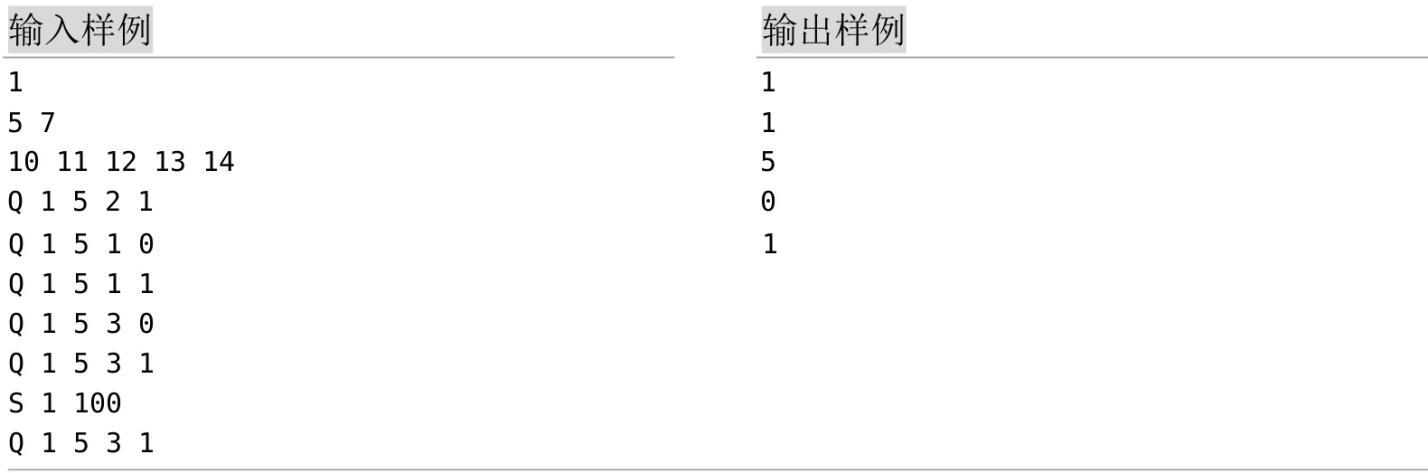


**HDU5057**

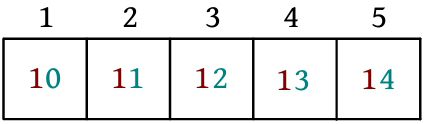
**题目描述（HDU5057）：**有由N个非负整数组成的序列：a[1], a[2], …, a[N]，对该序列进行M个操作，操作形式：①S X Y，将a[X]的值设置为Y（a[X]=Y）；②Q L R D P，求[L, R]区间第D位是P的元素个数，L和R是序列的索引。注意：第1位是最低有效位。

**输入：**第1行包含一个整数T，表示测试用例的数量。每个测试用例的第1行都包含两个整数N和M。第2行包含N个整数：a[1], a[2], …, a[N]。接下来的M行操作，若类型为S，则在该行中将包含两个整数X、Y；若类型为Q，则将包含4个整数L、R、D、P。其中：1≤T≤50，1≤N, M≤105，0≤a[i]≤231-1，1≤X≤N，0≤Y≤231-1，1≤L≤R≤N，1≤D≤10，0≤P≤9。

**输出：**对每个Q操作，都单行输出答案。



**题解：**根据测试用例的输入数据，序列如下图所示。



• Q 1 5 2 1：查询到[1, 5]区间第2位是1的元素有5个。

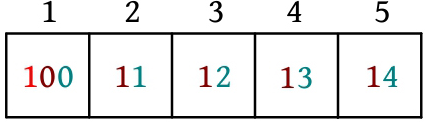
• Q 1 5 1 0：查询到[1, 5]区间第1位是0的元素有1个。

• Q 1 5 1 1：查询到[1, 5]区间第1位是1的元素有1个。

• Q 1 5 3 0：查询到[1, 5]区间第3位是0的元素有5个。

• Q 1 5 3 1：查询到[1, 5]区间第3位是1的元素有0个。

• S 1 100：将第1个元素修改为100。



• Q 1 5 3 1：查询到[1, 5]区间第3位是1的元素有1个。

本题包括点更新和区间查询，区间查询比较特殊，需要查询第D位是P的元素个数，可以采用分块的方法来解决。

**1. 算法设计**

（1）分块。划分块，统计每一块每一位上的元素个数。block[i][j][k]表示第i块中第j位是k的元素个数。

（2）查询。查询[l, r]区间第d位是p的元素个数。

• 若该区间属于同一块，则暴力累加块内第d位是p的元素个数。

• 若该区间包含多个块，则累加中间每一块i的block[i][d][p]，然后暴力累加左端和右端第d位是p的元素个数。

（3）更新。将a[x]的值更新为y。因为原来x所属的块已统计了a[x]每一位上的元素个数，所以此时需要减去，再将新的值y累加上即可。

**2. 算法实现**

