一维数组程序的综合分析——轴枢分区

2022 年 8 月 16 日

1. 阅读程序，回答相应问题。

```
#include <stdio.h>

int main()
/*本程序请求用户指定数组中的一个有效位置，并在从个位置中取出元素，
然后把这个元素为轴枢值，将数组分成两部分，
要求前面部分的元素都不超过轴枢值，后面部分的元素都不小于轴枢值*/
{
   int array[] = {-1, 25, 33, 19, 27, 16, 47, 12, 58, 46, 92, 41}; //
       0号单元空闲，暂不存储有用数据
   int len = sizeof(array) / sizeof(int) - 1; // 获取数组中元素的个数

   printf("The original array:\n");
   for(int i = 1; i <= len; i++)
   {
      printf("%d\t", array[i]);
   }
   printf("\n");

   int location_for_pivot; //
       以后将把location_for_pivot作为下标来获取数组中的一个元素，
                       // 并把该元素的值作为轴枢值（pivot_key）
   printf("Input a location for choosing values to be a pivot (>=1 and <= %d):\n",
       len);
   scanf("%d", &location_for_pivot);

   int low = 1, high = len; // 确定好需要进行分区的区间，包含全部有用数据

```

```
25    // 交换，以下3行直接写出或者画出结果
26    int temp = array[low];
27    array[low] = array[location_for_pivot];
28    array[location_for_pivot] = temp;
29
30    //
31    array[0] = array[low]; //
          把low号单元的元素存在空闲单元，然后low号单元可以被腾出来做其他事
32
33    int pivot_key = array[0]; // 以0号单元的值作为轴枢值，以方便后面使用
34
35    while(low < high) // 当区间的两个端点未相遇
36    {
37       // 以下while循环直接写出或者画出结果
38       while(low < high && array[high] >= pivot_key) // 后面元素符合要求
39       {
40          high--; // 继续往前检查
41       }
42
43       // 此时array[high]不符合之前的约定，搬到空闲单元
44       array[low] = array[high]; // 然后high号单元变为空闲单元
45
46       // 注意到搬动之后，array[low]符合下方a[low] <= pivot_key的条件
47
48       // 以下while循环直接写出或者画出结果
49       while(low < high && array[low] <= pivot_key) // 前面元素符合要求
50       {
51          low++; // 继续往后检查
52       }
53
54       // 此时array[low]不符合之前的约定，搬到空闲单元
55       array[high] = array[low]; // 然后low号单元变为空闲单元
56
57       // 注意到搬动之后，array[high]符合array[high] >= pivot_key的条件
58    }
59
60    array[low] = array[0];
61
62    printf("The partitioned array by %d:\n", array[0]);
```

```
63    // 按顺序输出array中的元素
64    for(int i = 1; i <= len; i++)
65    {
66       printf("%d\t", array[i]);
67    }
68
69    return 0;
70 }
```

(a) 分别就以下两种情况，分析以上程序的运行，写出输出结果。

i. 假设用户输入 2；

ii. 假设用户输入 6。

(b) 假设用户输入 2。

i. 计算第 35 行的条件 `low < high` 被判断的次数。

ii. 计算第 38 行的条件被判断的次数以及判断后发现它不成立的次数。用 $\surd$ 表示该行被判断并且发现成立，用 $\times$ 表示该行被判断并且发现不成立。那么，请用 $\surd$ 和 $\times$ 构成的序列来表示该行先后被判断得到的结果。例如，第一、三、四次判断得到不成立，第二、五次判断得到成立，用 $\times\surd\times\times\surd$ 表示。

iii. 计算第 49 行的条件被判断的次数以及判断后发现它不成立的次数。用 $\surd$ 表示该行被判断并且发现成立，用 $\times$ 表示该行被判断并且发现不成立。那么，请用 $\surd$ 和 $\times$ 构成的序列来表示该行先后被判断得到的结果。

iv. 当第 35 行的条件被判断后发现不成立时，分别给出 `high` 和 `low` 的值，并且分别给出 `array[high]` 和 `array[low]` 的值。请说明它们与数组前后元素的等量和不等关系。

(c) 假设用户输入 6。

i. 计算第 35 行的条件 `low < high` 被判断的次数。

ii. 计算第 38 行的条件被判断的次数以及判断后发现它不成立的次数。用 $\surd$ 表示该行被判断并且发现成立，用 $\times$ 表示该行被判断并且发现不成立。那么，请用 $\surd$ 和 $\times$ 构成的序列来表示该行先后被判断得到的结果。

iii. 计算第 49 行的条件被判断的次数以及判断后发现它不成立的次数。用 $\surd$ 表示该行被判断并且发现成立，用 $\times$ 表示该行被判断并且发现不成立。那么，请用 $\surd$ 和 $\times$ 构成的序列来表示该行先后被判断得到的结果。

iv. 当第 35 行的条件被判断后发现不成立时，分别给出 `high` 和 `low` 的值，并且分别给出 `array[high]` 和 `array[low]` 的值。请说明它们与数组前后元素的等量和不等关系。