哈尔滨工业大学(深圳)

# 《数据结构》实验报告

实验五

排序、查找及其应用


学　院: 计算机科学与技术

姓　名: 宗晴

学　号: 200110513

专　业: 计算机科学与技术


日　期: 2021-05-21

# 一、问题分析

题目 1 是：找出“中位数”母牛的产奶量，算法的平均时间复杂度不得大于 $O(n\log_2 n)$。转换成计算机要解决的问题就是，如何给一个序列排序并查找中间位置的元素。可利用数组存储并用快速排序的方法排序，并直接查找出中位数。

题目 2：找出长度为 n 的未排序数组中最大的 k 个元素，并按升序输出，算法的平均时间复杂度不得大于 $O(n\log_2 n)$。可利用堆排序，找到 k 个最大元素即可停止（当 k 值较小时，大大减少比较次数），最终将它们升序输出。

题目 3：已知学生人数以及每个人的空闲时间段的起始时间和终止时间，要求统计空闲人数最多的时间段，算法的平均时间复杂度不得大于 $O(M\log_2 M)$。可利用两个数组分别存储所有起始时间和终止时间。利用归并排序，将两个数组按升序排序，然后计算每个时间段的人数，输出人数最多的时间段的起始和终止时间对。

# 二、详细设计

## 2.1 设计思想

题目 1：该程序采取用数组存储，并采用快速排序的方法进行排序。由于数组支持随机查找，因此对于中间位置的数可直接进行查找。

将待排序序列的第一个数据定为枢轴元素（即每轮固定位置的元素），利用两个指针分别从头和尾向中间查找，左指针不断后移，直到指向的数值大于枢轴元素，右指针不断前移，直到指向的数值大于枢轴元素，交换两指针指向的数值。继续重复上述操作直到两指针相遇，此时的位置即为枢轴的位置，返回枢轴的位置。这样的一轮循环称为一次划分。

快排时，将待排序序列从头至尾进行一次划分，确定枢轴的位置。若枢轴恰好是中间位置，则排序结束，输出中间位置的值（该判断能适当地减少排序次数）。否则，则对枢轴两侧的序列进行递归排序。

最终返回数组中间位置的值，即为“中位数”母牛的产奶量。

题目 2：可利用大根堆存储待查找序列，并利用堆排序的方法查找 k 个最大的数据。

首先，定义调整某个结点的函数。此时，以此结点为根结点的大根堆，只有该结点不满足条件。沿其较大的孩子结点向下筛选，将较大的且比它大的孩子结点上移，继续向下筛选，直到两个孩子结点均比它小，将它插入该位置。调整结束。

在主算法中，首先建立大根堆，即从最后一个非叶结点开始，从后往前依次调整每个结点。然后将堆顶元素与堆的最后一个元素交换，将大根堆的范围减一。调整此时的堆顶元素，重复上述交换与调整的过程，直到找到 k 个最大元素（该判断能适当地减少排序次数）。最后将此时数组中最后 k 个元素复制到 res 数组中返回，即为所求的升序排列的 k 个最大元素。

题目 3：利用数组存储起始时间和终止时间数组，利用归并排序将两个数组进行升序排序。

首先定义一个用来将两个有序序列归并的函数 Merge，用两个指针分别指向两个序列，将当前元素相比较，较小者放入新数组中，该指针后移，直到有一个指针移到表尾，将另一个表中剩下的元素复制到新数组中。归并完成。

然后定义一个对数组进行一趟归并排序的函数 Mpass，len 为每段待归并段的长度，该段内的元素都是有序排列的。调用 Merge 函数将每两个长度为 len 的数组归并。若最后剩余的元素个数大于 len，也将这两段归并，否则直接将剩余元素复制到辅助数组中。一趟归并排序结束。

然后定义归并排序函数 MergeSort。申请一个辅助数组用来归并排序。len 置为 1，对数组进行归并排序，归并后的数组存入辅助数组中。而后，len 的值每次增加一倍，不断进行归并排序直到整个数组按升序排列。原数组和辅助数组交替存储归并后的序列，最终有序序列存储在原数组中。归并完成。

在主函数中，首先对 start 和 end 两个数组进行归并排序。由于空闲人数最多的时间段的始末位置只可能是在 start 和 end 数组中出现的时间点，因此只需用 i 和 j 两个指针分别遍历 start 和 end 数组，且对每一个终止时间，起始时间指针不用回溯，因为起始时间指针越靠后，人数越多。对每一段时间，计算空闲人数：起始时间（相同则取最后一个）的下标与结束时间（相同则取第一个）的下标之差再加一。若该时间段人数和当前最大人数相等，计数器加一，将起始和终止时间点存入数组的下一个位置；若该时间段人数更大，将最大人数更新为该

人数，计数器置为初始值，并将起始和终止时间点存入数组的第一个位置。最终输出空闲人数最多的时间段。

## 2.2 存储结构及操作

(1) 存储结构（一般为自定义的数据类型，比如单链表，栈等。）

题目 1：静态存储的顺序表——数组

```
#define N 10000
int a[N]; // 用于存储奶牛产奶量
```

题目 2：用数组存储的大根堆

```
#define MAX 1000
int arr[MAX + 1];
```

题目 3：静态存储的顺序表——数组

```
int start[1002];
int end[1002];
```

(2) 涉及的操作 (一般为自定义函数，可不写过程，但要注明该函数的含义。)

题目 1：

- int solve1(int *a, int n);

参数：存储各母牛产奶量的数组、奶牛数量

功能：查找“中位数”母牛的产奶量

返回值：返回“中位数”母牛的产奶量

- void my_QSort(int *a, int low, int high);

参数：待排序数组、待排序数组的起始位置、待排序数组的终止位置

功能：对数组进行快速排序

返回值：无返回值

- int Partition(int *a, int low, int high);

参数：待排序数组、待排序数组的起始位置、待排序数组的终止位置

功能：对待排序数组进行一次划分

返回值：返回枢轴位置

题目 2：

- int *solve2(int *arr, int n, int k);

参数：待查找数组、数组中元素个数、查找最大的 k 个元素的 k 值

功能：查找数组中最大的 k 个元素

返回值：返回按从小到大的顺序存储的 k 个最大元素的数组

- void HeapSort(int *arr, int n, int k);

参数：待排序数组、数组中的元素个数

功能：对数组进行堆排序（大根堆）

返回值：无返回值

- void HeapAdjust(int *arr, int s, int m);

参数：待排序数组,从位置 s 到 m 只有 s 需要调整（s 为此子大根堆的堆顶元

素）、待调整的大根堆的根结点位置、待调整的大根堆的终止位置

功能：调整某个结点的位置

返回值：返回枢轴位置

题目 3：

- void sort(int *start, int *end, int M);

参数：起始时间数组、终止时间数组、学生人数

功能：对起始时间与终止时间数组进行排序

返回值：无返回值

- void MergeSort(int *a, int n);

参数：待排序数组、数组元素个数

功能：归并排序

返回值：无返回值

- void Mpass(int *a, int *b, int n, int len);

参数：待排序数组、辅助数组（存储排序后的数组）、每段待归并段的长度

功能：对数组进行一趟归并排序

返回值：无返回值

- void Merge(int *a, int *b, int min, int m, int max);

参数：待归并数组、辅助数组（存储归并后的数组）、第一段序列的起始位置、第一段序列的终止位置、第二段序列的终止位置

功能：将两个有序序列归并

返回值：无返回值

- void findPeriod(int *start, int *end, int M);

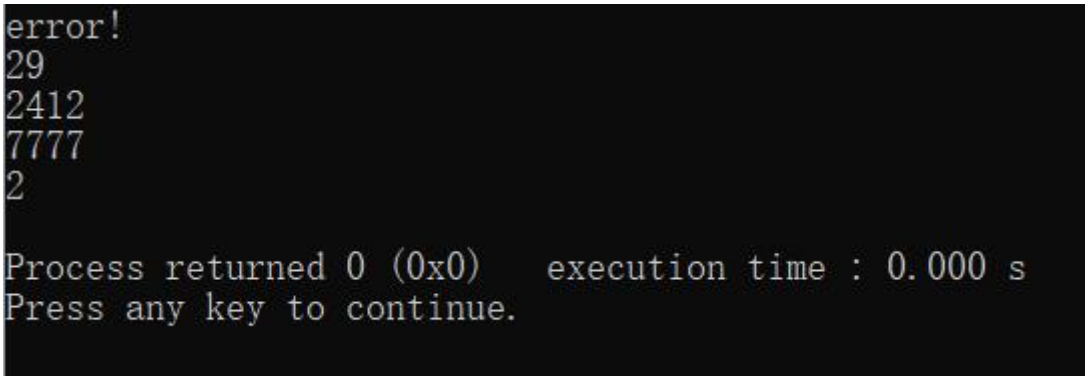
参数：起始时间数组、终止时间数组、学生人数

功能：查找空闲人数最多的时间段的起始时间和终止时间

返回值：无返回值

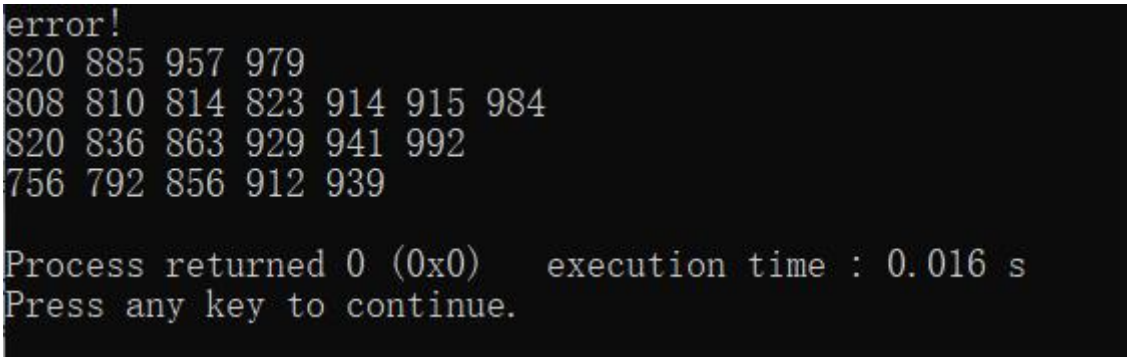
# 三、用户手册

题目 1：首先，用户应在第一行输入测试数据的组数，为一个正整数。接下去每两行代表一组测试数据。每组测试数据中，第一行输入一个奇数，表示奶牛数量 N；第二行输入 N 个整数，用空格隔开，代表 N 头奶牛的产奶量。（其中 1 ≤N<10000，产奶量∈ [0，1000000]）（健壮性）增加了对奶牛数奇偶性以及范围的判断，若输入数据不和要求，程序会输出提示信息“error!”,然后继续处理下一组数据。（如下图为第一组数据错误的输出情况）



程序会按照测试组数读入每组数据，按照奶牛数量读入产奶量。最终在每一行输出一个数值，为求得的每组“中位数”母牛的产奶量。每组数据之间换行输出。

题目 2：首先，用户应在第一行输入测试数据的组数，为一个正整数。接着换行输入每一组的数据。在每一组数据中，第一行输入一个正整数，表示最大 k 个数的 k 值。在下一行输入一个大于等于 k 的整数，表示数组的长度 n。然后在接下来的 n 行中，每行输入一个数据，为数组元素。（其中 1≤k≤n≤1000）（健壮性）此处增加了对于 k 范围的判断，若输入数据不和要求，程序会输出提示信息“error!”,然后继续处理下一组数据。（如下图为第一组数据错误的输出情况）




程序会按照测试组数读入每组数据，按照数组的长度读入数组元素。最终在每一行输出一个升序序列，为所求的最大的 k 个数据。

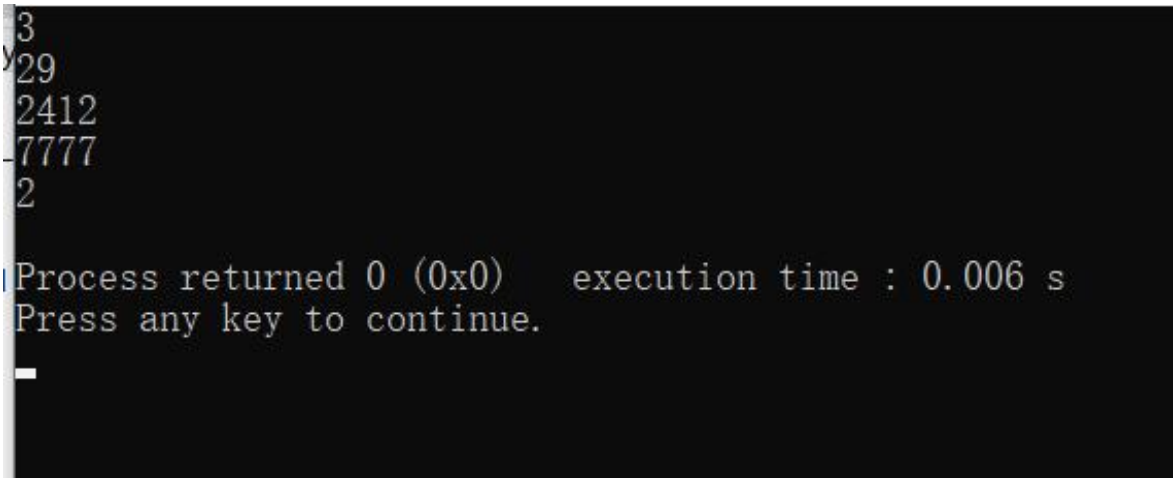
题目 3：每组数据换行输入。对于每组数据：首先，用户应在第一行输入两个整数 N，M，表示 N 个空闲时间段，M 个学生，用空格隔开，接着输入 M 行，

每行两个数字，用空格隔开，表示第 i 名同学空闲时间的开始时间段与终止时间段。其中 1≤N≤100000000 ， 1≤M≤1000。

对于每组数组，程序会在第一行输出“==== Case i ====”其中 i 表示第 i 组测试数据。然后在下一行输出表示空闲人数最多时间段的起始时间和终止时间。如若有多对，则以递增形式在一行中输出。不同对之间用英文逗号“,”分隔，对内元素用空格隔开。每组数据换行输出。

# 四、结果

题目 1:




题目 2:

```
738 764 782 813 919
820 885 957 979
808 810 814 823 914 915 984
820 836 863 929 941 992
756 792 856 912 939

Process returned 0 (0x0)   execution time : 0.000 s
Press any key to continue.
```

题目 3:

```
==== Case 1 ====
3 3
==== Case 2 ====
2 2,3 3
==== Case 3 ====
4 4,5 5

Process returned 0 (0x0)   execution time : 0.000 s
Press any key to continue.
```

# 五、总结

该实验涉及到的数据结构和算法，以及遇到的问题和收获。

该实验涉及到的数据结构有静态存储的顺序表——数组、大根堆等。涉及的算法有快速排序、堆排序、归并排序等。同时利用这些算法解决实际的问题，例如查找“中位数”母牛的产奶量、查找数组中最大的 k 个元素、查找空闲人数最多的时间段等。

在这次实验中，我亲身实现了三种排序算法，对它们的适用场合有了更深入的了解。题目 3 花费了我很多时间，起初我想了一种很复杂的做法，导致代码非

常冗长，后来才改成了现在这种更加简便的算法，这期间，我对这道题目的理解也大大加深了。