**算法设计与分析实验二**

**—k1到k2小问题：**

# 1 实验内容

问题概述：

从包含n个整数的无序列表中输出第k1小到第k2小之间的所有整数，其中k1<=k2。分析时间复杂度。

实验要求：

必须用分治法求解，不能简单地重复使用求第k小元素的减治法。

禁止使用排序算法求解

给出复杂度分析过程

# 2 算法思想

分治法查找K1-K2小的算法思想：

本次实验要求输出无序数列中的第k1到k2小的值。同时不能使用完全的排序算法。

故此次实验参考排序算法的思想。具体思路如下：

首先，定义一个结构，用来保存数据的值，以及赋予这个值一个flag，用来决定最后是否要输出他。

算法设想过程:

基于快速排序的原理，将数排在中轴两边，中轴左边的小于中轴，中轴右边的大于中轴。这一思想，想设计两个中轴，最终的结果舍弃两个中轴两边的数据，中轴之间的数据即k1到k2的数据。

基于这一思路，通过一个while循环来确定哪些数据是需要输出的。While中嵌套一个for循环，for循环用来确定这一轮中哪些数据是需要被抛弃的。For循环确定的数据将有后面的语句将他们的flag值修改为0（不输出）.最终输出只需输出所有flag为1的值。

# 3 实验过程

代码部分：

#include<iostream>

using namespace std;

#define  swap(x, y) {int a = x; x = y; y = a;}//定义交换操作，将下x，y值交换

#define max  100 //定义一个静态数组的大小

typedef struct Data

{

    int data;

    bool flag;

}Data, Min[max];//使用一个数据结构，此结构意在保存每一个数据，并给他们一个标志值

Min a;

void Binary\_search(Min &a, int n, int k1, int k2)

{

    //查找无序列表中第k1到第k2小的整数

    int x = 0;

    int y = n - 1;

    //初始化x，y值

    while (x < k1 - 1 || y > k2 - 1)

    {

        int temp;

        //存储最小和最大数的下标

        int f1, f2;

        f1 = x;

        f2 = y;

        //此处for循环，即是将每一次进来的最小值筛选出来

        for (int i = x; i <= y; i++)

        {

            if (a[f1].data > a[i].data)

                f1 = i;

        }

        //相当于把不在k1,k2范围内的数组元素标志值改为0，到后面输出操作时不输出。

        if (x < k1 - 1)

        {

            temp = a[x].data;

            a[x].data = a[f1].data;

            a[f1].data = temp;

            a[x].flag = 0;

            x++;

        }

        //此处for循环，即是将每一次进来的最大值筛选出来

        for (int i = x; i <= y; i++)

        {

            if (a[f2].data < a[i].data)

                f2 = i; //缩小空间

        }

        //相当于把不在k1,k2范围内的数组元素标志值改为0，到后面输出操作时不输出。

        if (y > k2 - 1)

        {

            temp = a[y].data;

            a[y].data = a[f2].data;

            a[f2].data = temp;

            a[y].flag = 0;

            y--;

        }

    }

    cout << "第" << k1 << "小到" << k2 << "小之间的所有整数有:";

    for (int i = 0; i < n; i++)

    {

        if (a[i].flag)

            cout << a[i].data << "  ";

    }

    cout << endl;

}

int main()

{

        int n;

        int k1, k2;

        cout << "请输入无序列表大小::";

        cin >> n;

        cout << "请输入无序列表中元素:";

        for (int i = 0; i < n; i++)

        {

            a[i].flag = 1;

            cin >> a[i].data;

        }

        cout << "请输入k1-k2的值:";

        cin >> k1 >> k2;

        if (k1 > k2)

            swap(k1, k2);

        if (k1<0 || k2>n || n < 0)

        {

            cout << "输入不合法！" << endl;

        }

        Binary\_search(a, n, k1, k2);

        system("pause");

        return 0;

}

# 4 实验结论

实验输入：

一个长度为n的无序数组a，例：

11, 3, 5, 4, 1, 10, 8, 2, 6, 9

整数k1和k2(1<=k1<k2<=n)，例：

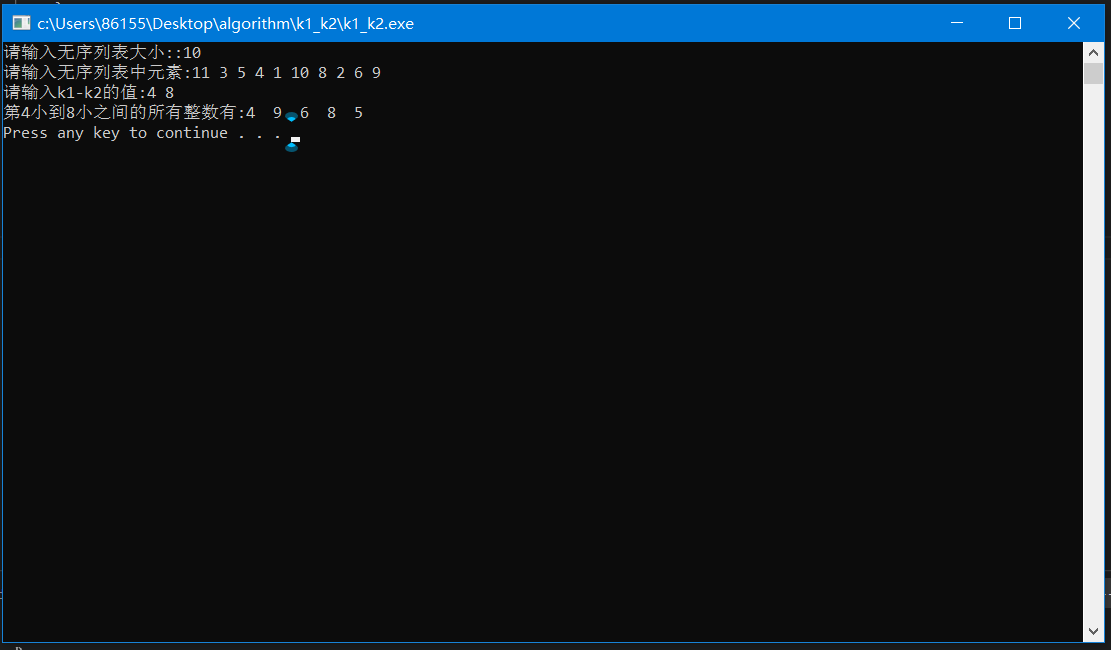
4和8

实验输出：

数组a中第k1大到第k2大的数（不一定要有序）

4，5，9，6，8

实例1



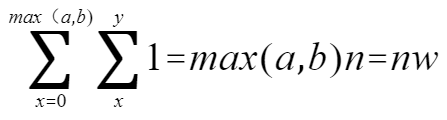
# 5、算法复杂度分析

1. 算法的输入规模：n（即输入的数据个数）
2. 基本操作：

if (a[f1].data > a[i].data)

1. 该算法是非完全排序。每次执行必执行比较操作，因此无做好最坏之分。
2. 时间复杂度公式：

公式



Max(a,b)表示k1,k2到整个数据列边界的距离。最后用w表示此值。