**北京邮电大学软件学院**

**2020-2021学年第一学期实验报告**

**课程名称：** 算法分析与设计

**项目名称： 实验四：分治法**

**项目完成人：**

**姓名：\_\_王衔飞\_\_学号：\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**姓名：\_\_\_\_\_\_\_\_学号：\_\_\_\_\_\_\_\_**

**指导教师：**\_\_\_\_\_\_\_\_\_\_\_李朝晖\_\_\_\_\_\_\_\_\_\_\_

**日 期： 2020年 11 月 15 日**

1. **实验目的**

1、 深刻理解并掌握分治法的设计思想；

2、 提高应用分治法设计算法的技能；

1. **实验内容**

运用分治法编程解决基本题，学有余力者继续解决选做题。

1. **实验环境**

Xcode 12.2 + macOS 11.0.1 + gcc9

语言：C++ 17

1. **实验结果**

**元素选择：**给定序列中 n 个元素和一个整数 k，1≤k≤n，输出这 n 个元素中第 k 小 元素的值及其位置。编写并调试程序（不排序）。

实验结果：完成实验任务。所用算法的时间复杂度最好情况（待排序列接近无序）时间复杂度为，最坏情况（待排序列接近有序）时间复杂度为O(n2)。

测试数据1：a[] ={2,5,3,6,7,9,57}; k =3; 输出结果：5,位置是2

图形用户界面, 文本, 应用程序

描述已自动生成

测试数据2：a[] ={54,62,12,49,81,27,33}; k =4; 输出结果：49,位置是4

图形用户界面, 文本, 应用程序

描述已自动生成

求逆序对问题：-如果存在 l<r ，并且 X(l)>X(r)，则称为一个逆序对。 求一个给定 序列的逆序对数目并列出所有的逆序对.

实验结果：完成实验任务。所用算法的时间复杂度时间复杂度为O(nlogn)。

测试数据1：{1,5,2,3,6,4} 输出：共有4对

分别是：<5,2> <6,4> <5,3> <5,4>

图形用户界面, 文本, 应用程序

描述已自动生成

测试数据2：{9,8,2,3,6,1,7} 输出：共有14对

分别是：<9,8> <8,2> <9,2> <8,3> <9,3> <6,1> <2,1> <3,1> <8,1> <9,1> <8,6> <9,6> <8,7> <9,7>

图形用户界面, 文本, 应用程序

描述已自动生成

1. **附录**

（附上实验文档，尽量有：问题分析、设计方案、算法原理、流程、设计图、程序、仿真结果、运行结果、心得等，具体内容根据实验要求来定。源代码请附在这里。源代码排版请特别注意，用5号字体，行间距为单倍行距。注意节省空间，不要浪费纸张。）

**元素选择：**

先选择一个中枢（一般我们选第一个），然后遍历后面的元素，最终会把数组分为两部分，前面部分比中枢值小，后面部分大于或等于中枢值，如果交换之后中枢值所在的位置就是从后面数第k个，我们直接返回中枢值即可，如果从后面数大于第k个，我们只需按照同样的方式从后面部分开始找即可。如果从后面数小于第k个，我们同样从前面部分开始查找。

代码如下：

#include <iostream>

#include <algorithm>

#include <array>

**template**<std::size\_t size>

**int** findKthMin(std::array<**int**,size> nums,**int** k) {

k = size - k;

**int** lo = 0, hi = size - 1;

**while** (lo <= hi) {

**int** i = lo;

//这里把数组以A[lo]的大小分为两部分，

//一部分是小于A[lo]的，一部分是大于A[lo]的

// [lo,i]<A[lo]，[i+1,j)>=A[lo]

**for** (**int** j = lo + 1; j <= hi; j++)

**if** (nums[j] > nums[lo]) std::swap(nums[j], nums[++i]);

std::swap(nums[lo],nums[i]);

**if** (k == i) **return** nums[i];

**else** **if** (k < i) hi = i - 1;

**else** lo = i + 1;

}

**return** -1;

}

**int** main() {

// 测试代码

std::array a{54,62,12,49,81,27,33};

**int** k =4;

**int** result = findKthMin(a, k);

std::cout << "第"<<k<<"小的数字是：" << result << std::endl;

std::cout << "位置是：" << std::find(a.begin(),a.end(),result)-a.begin()+1 << std::endl;

**return** 0;

}

**找逆序对：**

本题中分治法的细节如下：对于一个序列，将其分解成两个等大的子序列，称其为左序列和右序列，那么逆序对存在于如下三处：1.左序列中的逆序对；2.右序列中的逆序对；3.由左序列中的元素和右序列中的元素构成的逆序对。将三种情况找到的逆序对加起来即为最终答案。

代码如下：

#include <iostream>

#include <vector>

#include <utility>

**int** temp[100]; //临时数组用于合并步骤，不在递归函数中开启数组，防止递归过程内存中同时存在大量数组

std::vector<std::pair<**int**,**int**>> pairs;

**int** merge(**int** arr[], **int** l, **int** m, **int** r) {

**int** count = 0, i, j;

**for**(i = l; i <= r; i++)

temp[i] = arr[i];

**int** k = l;

**for**(i = l, j = m + 1; i <= m && j <= r;k++) {

**if**(temp[i] > temp[j]) {

count += m - i + 1;//每次发现左序列元素比右序列元素大，逆序对数量增加，m-i+1即为该元素到左序列尾的元素数量

//其余代码为归并排序的步骤

**for** (**int** ii=0; ii<m - i + 1; ii++) {

pairs.push\_back(std::pair<**int**,**int**>{temp[i+ii],temp[j]});

}

arr[k] = temp[j];

j++;

}

**else** {

arr[k] = temp[i];

i++;

}

}

**if**(i > m) {

**while**(j <= r) {

arr[k] = temp[j];

j++;

k++;

}

}

**else** {

**while**(i <= m) {

arr[k] = temp[i];

i++;

k++;

}

}

**return** count;

}

**int** findReversePair(**int** arr[], **int** l, **int** r) {

**if**(l == r)

**return** 0;

**int** m = (l + r) / 2;

**int** ln = findReversePair(arr, l, m);//左序列逆序对数量

**int** rn = findReversePair(arr, m + 1, r);//右序列逆序对数量

**int** mn = merge(arr, l, m, r);//左序列与右序列元素构成的逆序对数量

**return** ln + rn + mn;

}

**int** main(**void**) {

**int** test[]={9,8,2,3,6,1,7};

**int** i = **sizeof**(test)/**sizeof**(test[0]);

findReversePair(test, 0, i - 1);

std::cout << "共有"<< pairs.size() << "对\n分别是：";

**for**(**auto** [a,b]:pairs) std::cout<< '<' << a << ',' << b << '>' << ' ';

std::cout << std::endl;

**return** 0;

}