**同济大学计算机系**

**算法分析与设计期中大作业实验报告**

****

**学 号 2050283**

**姓 名 陆明奇**

**专 业 计算机科学与技术**

**授课老师 程大伟 刘钦源**

1. **选择问题**
2. **题目**

给定线性序集中n个元素和一个整数k，1kn，要求找出这n个元素中第k小的元素（这里给定的线性集是无序的）。

1. **算法思想**

（1）基于堆的选择

不需要对全部n个元素进行排序，只需要维护k个元素的最大堆，即容量为k的最大堆存储最小的k个数。首先创建一个新数组，用于存放一个大根堆，将原数组的前k个数搞成大根堆的形式存放到该新数组中。接着从第k个数开始遍历到原数组最后，只要存在元素比堆顶元素小则将堆顶用该元素替换，并维护大根堆。最后取出大根堆的堆顶元素就是第k小的数了。

（2）随机划分线性选择

这个算法运用了分治的思想，是模仿快速排序算法设计出来的。基本思想是对输入数组进行递归划分。不同于快速排序算法，它只对划分出来的字数组之一进行递归处理。该算法用到了在随机快速排序算法中用到过的随机划分函数，因此划分是随机产生的。由此可见，该算法也是一个随机化算法。要找到数组arr[0:n-1]中的第k小的元素，只需要调用random\_select(arr,0,n-1,k)即可。

（3）利用中位数的线性时间选择

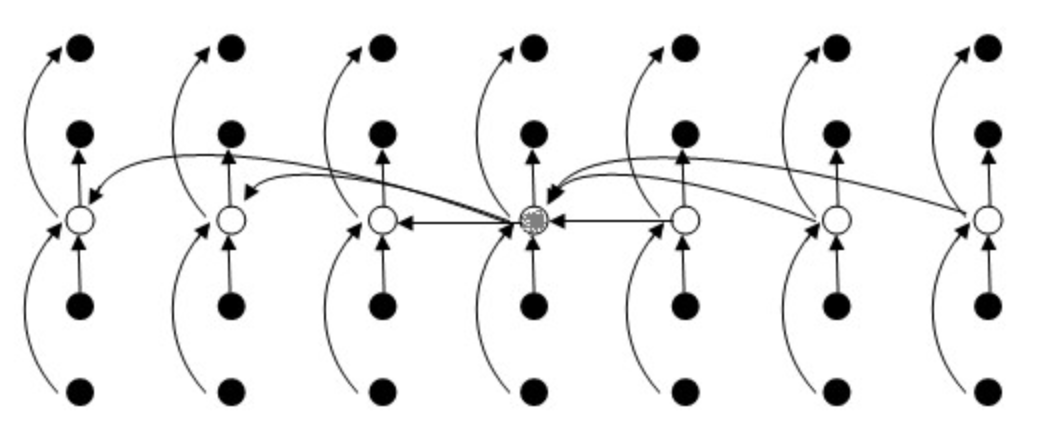
如果能在线性时间内找到一个划分基准，使得按这个基准划分出的两个子数组的长度都至少为原数组长度的倍（0<<1是某个正常数），那么在最坏情况下用O（n）时间就能完成选择任务。可以按以下步骤找到满足要求的划分基准：

a.将输入的n个数划分成 ⌈n/5⌉个组，当然最后一组的数目可能是小于5的

b.用任意的排序方法对他们进行排序，并取出一共 ⌈n/5⌉个中位数。

c.找出该 ⌈n/5⌉个中位数中的中位数。（如果 ⌈n/5⌉是偶数则取相对大的那个数）

d.将全部的数划分为两个部分，小于基准的在左边，大于等于基准的放右边。



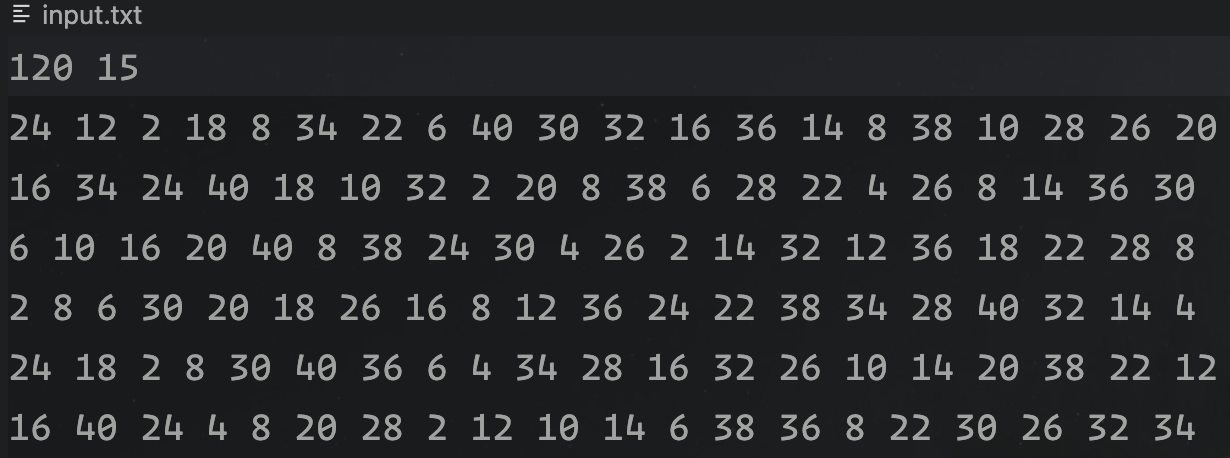
1. **编程语言及环境**

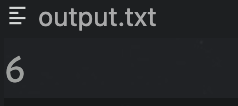
操作系统：Windows10

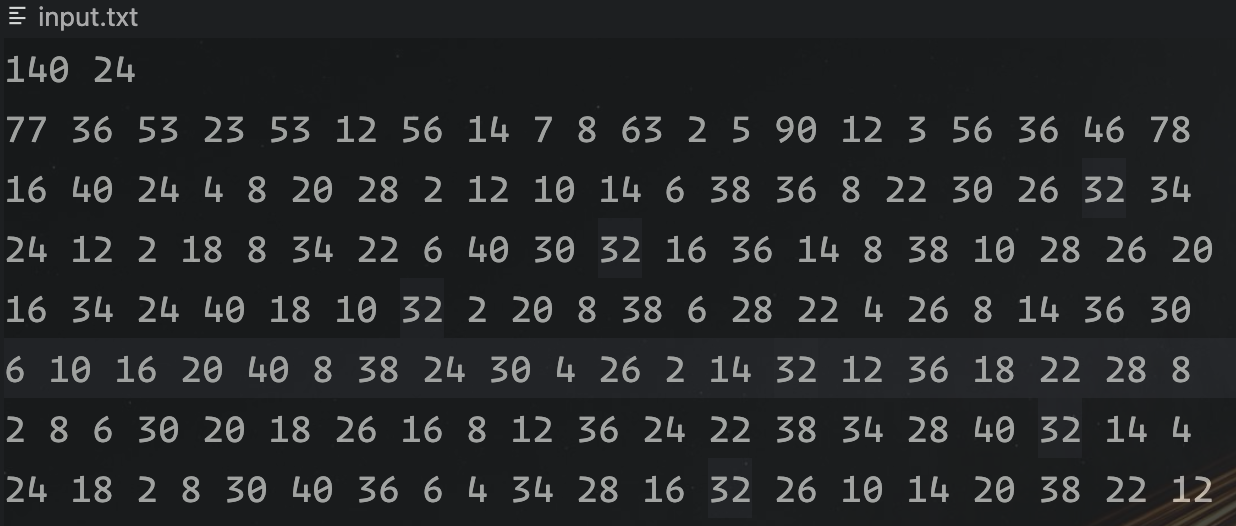
开发语言：C++

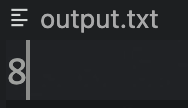
开发环境及具体版本：Visual Studio Code

1. **运行结果**

****

****

****

****

1. **算法分析**

（1）基于堆的选择

利用堆排序，特别适用于海量数据中寻找最大或者最小的k个数字。即构建一个大堆容器，初始化大小为k，选择数组的前k个元素，填充堆，然后调整为最大堆。调整完之后，继续从初始数组中拿出一个元素，如果该元素比大堆的堆顶小，则替换堆顶，继续调整为最大堆，如果大于等于堆顶则直接丢弃，不作调整。 这种算法的优点是优点：不用修改原始数组，适合海量数据。算法复杂度为O（k+（n-k）logk）

（2）随机划分线性选择

该算法在执行random\_partition函数后，数组arr[p:r]被划分为了两个子数组arr[p:i]和arr[i+1:r]，使得arr[p:i]中的每个元素都不大于arr[i+1:r]中的每一个元素。接着计算子数组arr[p:i]中的元素个数j。如果kj，则arr[p:r]中第k个元素落在arr[p:i]中。如果k>j,则要找的第k小元素落在子数组arr[i+1:r]中。由于已经知道子数组arr[p:i]中元素均小于要找的第k小元素，因此要找的arr[p:r]中第k小元素是arr[i+1:r]中的第k-j小元素。随机划分线性选择算法在最坏的情况下，时间复杂度为Ω（n2）。例如，在找最小元素时，总是在最大元素处划分。但该算法平均性能很好，它可以在O（n）平均时间内找出n个输入元素中的第k小元素。

（3）利用中位数的线性时间选择

当需要对n个元素的数组调用select函数时，即时间为T（n）。找中位数的中位数x至多用T（n/5）时间。而按照算法所选的基准x进行划分所得到的两个子数组分别至多有3n/4个元素。所以无论对哪个子数组调用select都至多用T（3n/4）。算法将每组的大小定为了5，并选取75作为是否进行递归调用的分界点。这两点保证了T（n）的递归式中两个自变量之和（）。这是使T（n）=O（n）的关键。然而，该算法实际上运行缓慢，因为n前面的常数比较大。

Input1-6为随机生成的不同规模的数据（10，100，1K，10K，100K，1M）

**

1. **问题及解决方案**

利用堆进行排序时，一开始使用了小顶堆，结果得到的答案是数组中第k大的数字。通过对堆排序算法的再理解，真正明白了堆排序算法的原理之后，改用了大顶堆，最终得出了正确的答案。在随机划分线性选择算法中，编写partition函数时，一开始使用了数组arr[p:r]的右边界arr[r]作为基准，若arr[r]是arr[p:r]中最大的元素，则会使函数陷入死循环。当改用数组左边界arr[p]作为基准时，问题得到解决。

1. **心得体会**

利用堆进行排序时选择大堆还是小堆很重要。不是寻找最小的k个元素就要选择小堆，而且恰恰相反，寻找最小的k个数，其实就是寻找第k个大的元素，即寻找k个数中最大的。不断调整堆，堆的元素个数是k，堆顶是最大值，遍历完初始数组后，堆中存在的元素即使我们所要寻找的k个最小元素。

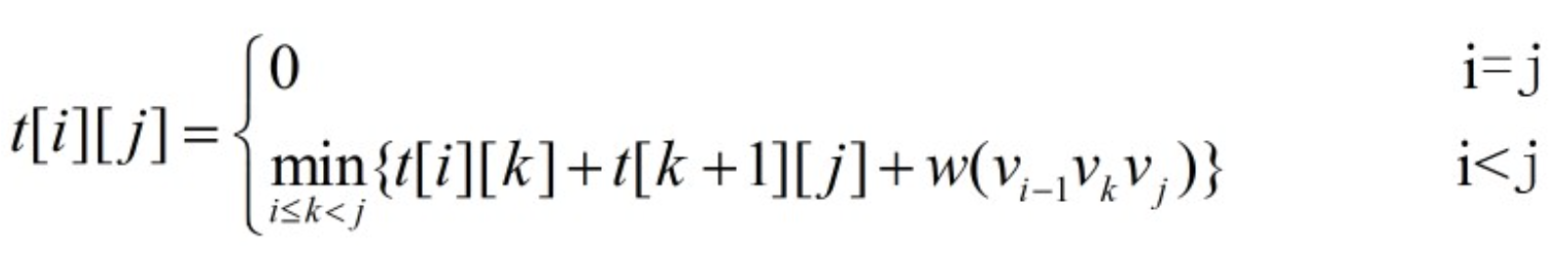
本题使用了三种不同的排序方法来寻找数组中第k小的数，使我对排序问题有了更深入地了解，体会到了不同算法的时间复杂性与空间复杂性的差异。随机划分线性选择算法利用到了分治思想，在代码形式上更为整洁美观。而堆排序则没有用到分治思想，两者一对比，使我对分治思想有了更好地了解。

1. **博物馆警卫巡逻问题**
2. **题目**

博物馆是具有n个顶点的凸多边形的形状。博物馆由警卫队通过巡逻来确保馆内物品的安全。博物馆的安全保卫工作遵循以下规则，以尽可能时间经济的方式确保最大的安全：（1）警卫队中每个警卫巡逻都沿着一个三角形的路径，该三角形的每个顶点都必须是多边形的顶点。（2）警卫可以观察其巡逻路径三角形内的所有点，并且只能观察到这些点：我们说这些点由该警卫守护并覆盖。（3）博物馆内的每一处都必须由警卫人员守护。（4）任何两个警卫巡逻所在的三角形在其内部不会重叠，但他们可能有相同的边。在这些限制条件下，警卫的成本是警卫巡逻所沿路径的三角形的周长。我们的目标是找到一组警卫，以使警卫队的总成本（即各个警卫的成本之和）尽可能小。给定博物馆顶点的x坐标和y坐标以及这些顶点沿博物馆边界的顺序，设计一种算法求解该问题，并给出算法的时间复杂性。

1. **算法思想**

该问题是一个凸多边形的最优三角剖分问题，可以利用动态规划的思想来解决。设t[i][j],1i<jn为凸多边形{Vi-1,Vi……Vj}的最优三角剖分所对应的权值函数值，即其最优值。最优剖分包含三角形Vi-1VkVj的权，子多边形{Vi-1,Vi……Vk}的权，子多边形{Vk，Vk+1……Vj}的权之和。因此，可得递推关系式：



凸(n+1)边形P的最优权值为t[1][n]。

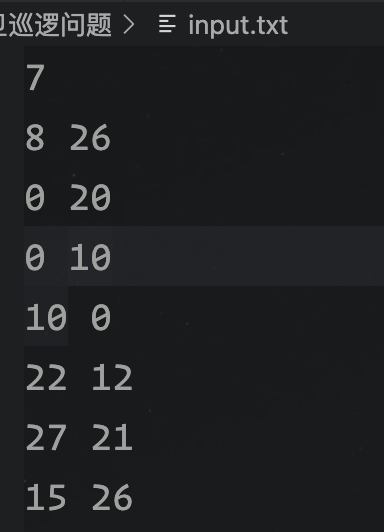
1. **编程语言及环境**

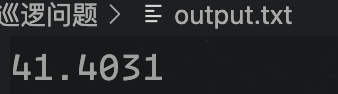
操作系统：Windows10

开发语言：C++

开发环境及具体版本：Visual Studio Code

1. **运行结果**

****

****

1. **算法分析**

凸多边形的最优三角剖分问题有最优子结构性质。 若凸(n+1)边形P={v0,v1,...,vn}的最优三角剖分T包含三角形v0vkvn，1≤k≤n-1，则T的权为3个部分权的和:三角形v0vkvn的权，子多边形{ v0, v1,..., vk}和{vk, vk+1,..., vn}的权之和。可以肯定，由T所确定的这2个子多边形的三角剖分也是最优的。因为若有{ v0, v1,..., vk}或 {vk, vk+1,..., vn}的更小权的三角剖分将导致T不是最优三角剖分的矛盾。定义t[i][j]，1≤i<j≤n为凸子多边形{ vi-1,vi,...,vj}的最优三角剖分所对应的权函数值，即其最优值。设退化的多边形{ vi, vi }具有权值0。据此定义，要计算的凸(n+1)边形P的最优权值为t[1][n]。t[i][j]的值可以利用最优子结构性质递归地计算。当j-i≥1时，凸子多边形至少有3个顶点。由最优子结构性质，t[i][j]的值应为 t[i][k]的值加上t[k+1][j]的值，再加上三角形vi-1vk vj的权值，其 中i≤k≤j-1。由于在计算时还不知道k的确切位置，而k的所有可能位置只有j-i个，因此可以在这j-i个位置中选出使t[i][j]值达到最小的位置。该算法的时间复杂度为O（n3），空间复杂度为O（n2）

1. **问题及解决方案**

我一开始没有将退化的三角形——边，的权值赋为0，使得它的初始值为一个随机数，导致最后的答案错误。后经过对动态规划算法的再理解，并用纸笔实际推导演算后，发现了问题，将边的权初始化为0后，问题得到解决。

1. **心得体会**

通过这题的练习，我对于动态规划算法有了更深入的理解。此题的权函数为三角形边的欧几里得距离之和，其实不论三角形的权函数是什么，这类问题都可以运用动态规划算法来解决，因为它具有最优子结构性质和重叠子问题性质。此外，动态规划算法每一层循环的起始及终止条件需要注意，否则会导致计算结果错误。