核心算法分析

王陆萱

1．**快速排序**—用于课程和活动关键字段排序、课程表排序、活动时间和名称排序

我们用非递归的意义就在于，当数据量过大的时候，我们使用递归会导致栈溢出(Stack Overflow)的问题，这是由于递归太深，栈空间不足出现的。所以我们要使用非递归，在堆区去开辟空间从而实现快速排序。而实现非递归的快速排序的核心就在于如何去模拟实现递归。这里我们需要借助（数据结构中的）栈，借助栈的后进先出的性质，这一性质和我们递归调用函数的时候非常相似，所以我们才可以用栈区模拟实现递归。

- 时间复杂度分析：

快速排序每次确定一个数，每次交换为等差数列，(n，n-1，n-2，...，2,1)，一共 n 个数，所以在最坏的情况下是 O(N*N)。一般情况下复杂度为（ $O(N*\log_2(N))$ ）：

- 空间复杂度分析：借助栈，为 O（N）

2．**kmp 算法**—用于关键字段匹配

- 时间复杂度分析：假设 m 为模式串 str1 长度，n 为待匹配的字符串 str2 长度。

O(m+n)=O([m,2m]+ [n,2n]) = 计算 next 数组的时间复杂度+遍历比较的复杂度。

- 空间复杂度

空间复杂度很容易分析，KMP 算法只需要一个额外的 next 数组，数组的大小跟模式串相同。 所以空间复杂度是 O (m)，m 表示模式串的长度。

3．**贪心算法**—用于地图路径查询

本组地图路径查询的算法选用的是贪心算法手动入栈，而不是 Dijkstra 算法。贪心算法总是作出在当前看来最好的选择，即贪心算法不从整体的角度来考虑，其所作的选择某种意义上的局部最优情况，不一定能够达到全局最优。我们的的目的是寻求点到点的路径并不追求全局最优，贪心算法相比于 Dijkstra 算法能够出更简单的算法设计和更低的算法复杂度。

- 时间复杂度分析：

  贪心算法：时间复杂度为 $O(n^2)$ （n 为起点到终点的点位数）

  迪杰斯特拉算法：时间复杂度为 $O(N^2)$（N 为所有节点数）

- 空间复杂度分析

  贪心算法：时间复杂度为 $O(n^2)$ （n 为起点到终点的点位数）

  迪杰斯特拉算法：时间复杂度为 $O(N^2)$（N 为所有节点数）

4．**冲突检测算法**—活动冲突检测、课程冲突检测

- 时间复杂度分析：

  活动冲突检测：使用滑动窗口和课程比较冲突，时间复杂度：O(n)

  使用时间和已有活动比较冲突，时间复杂度：O(n)

  课程冲突检测：使用节数星期数比较，时间复杂度：O(n)

5．**压缩算法**

使用哈夫曼编码，构建哈夫曼树

- 时间复杂度 $O(n(\log n)^2)$d