# 十大经典算法

参考 ： https://blog.csdn.net/zuochao_2013/article/details/79413213

**第十名：Huffman coding（霍夫曼编码）**

霍夫曼编码(Huffman Coding)是一种编码方式，是一种用于无损数据压缩的熵编码（权编码）算法。1952年，David A. Huffman在麻省理工攻读博士时所发明的，并发表于《一种构建极小多余编码的方法》（A Method for the Construction of Minimum-Redundancy Codes）一文。

**第九名：Binary Search （二分查找）**

在一个有序的集合中查找元素，可以使用二分查找算法，也叫二分搜索。二分查找算法先比较位于集合中间位置的元素与键的大小，有三种情况（假设集合是从小到大排列的）：

1.键小于中间位置的元素，则匹配元素必在左边（如果有的话），于是对左边的区域应用二分搜索。

2.键等于中间位置的元素，所以元素找到。

3.键大于中间位置的元素，则匹配元素必在右边（如果有的话），于是对右边的区域应用二分搜索。

另外，当集合为空，则代表找不到。

**第八名：Miller-Rabin作的类似的试验测试**

这个想法是利用素数的性质(如使用费马大定理)的小概率寻找见证不数素数。如果没有证据是足够的随机检验后发现,这一数字为素数。

**第七名：Depth First Search、Breadth First Search（深度、广度优先搜索）**

它们是许多其他算法的基础。关于深度、广度优先搜索算法的具体介绍，请参考此文：**教你通透彻底理解：BFS和DFS优先搜索算法**。

**第六名：Gentry's Fully Homomorphic Encryption Scheme（绅士完全同态加密机制）算法。**

此算法很漂亮，它允许第三方执行任意加密数据运算得不到私钥。

**第五名：Floyd-Warshall all-pairs最短路径算法**

关于此算法的介绍，可参考我写的此文：几个最短路径算法比较（http://blog.csdn.net/v_JULY_v/archive/2011/02/12/6181485.aspx）。

d[]: 二维数组. d[i,j]最小花费、或最短路径的邻边。

```
for k from 1 to n:
  for i from 1 to n:
    for j from 1 to n:
      d[i,j] = min(d[i,j], d[i,k] + d[k,j])
```

**第四名：Quicksort（快速排序）**

快速排序算法几乎涵盖了所有经典算法的所有榜单。它曾获选二十世纪最伟大的十大算法（参考这：细数二十世纪最伟大的10大算法）。关于快速排序算法的具体介绍，请参考我写的这篇文章：**一之续、快速排序算法的深入分析**，及**十二、一之再续：快速排序算法之所有版本的c/c++实现**。

**第三名：BFPRT 算法**

1973 年，Blum、Floyd、Pratt、Rivest、Tarjan一起发布了一篇名为 “Time bounds for selection” 的论文，给出了一种在数组中选出第k大元素平均复杂度为O（N）的算法，俗称"中位数之中位数算法"。这个算法依靠一种精心设计的 pivot 选取方法，即选取中位数的中位数作为枢纽元，从而保证了在最情况下的也能做到线性时间的复杂度，打败了平均O（N*logN）、最坏 O(n^2) 复杂度的快速排序算法。

事实上，这个所谓的BFPRT，就是本blog中阐述过的快速选择SELECT算法，详情请参考下列博文：**第三章、寻找最小的k个数**、**十四、快速选择SELECT算法的深入分析与实现**。在我的这两篇文章中，给出了此快速选择SELECT算法，借助选取数组中中位数的中位数作为枢纽元，能做到最坏情况下运行时间为O（N）的复杂度的证明。

我在这里简单介绍下在数组中选出第k大元素的时间复杂度为O（N）的算法：

类似快排中的分割算法：

每次分割后都能返回枢纽点在数组中的位置s,然后比较s与k的大小

若大的话，则再次递归划分array[s..n]，

小的话，就递归array[left...s-1] //s为中间枢纽点元素。

否则返回array[s]，就是partition中返回的值。 //就是要找到这个s。

找到符合要求的s值后，再遍历输出比s小的那一边的元素。

各位还可参考在：算法导论上，第九章中，以期望线性时间做选择，有寻找数组中第k小的元素的，平均时间复杂度为O（N）的证明。原程序随机选取数组中某一元素作为枢纽元，最后可证得程序的期望运行时间为O(n)，且假定元素是不同的。

**第二名：Knuth-Morris-Pratt字符串匹配算法（KMP）**

关于此算法的介绍，请参考此文：六、教你从头到尾彻底理解KMP算法。KMP算法曾经落选于二十世纪最伟大的十大算法，但人们显然不能接受，如此漂亮、高效的KMP算法竟然会落选。所以，此次最终投票产出生，KMP算法排到了第二名。

**第一名：Union-find**

并查集是一种树型的数据结构，用于处理一些不相交集合（Disjoint Sets）的合并及查询问题。常常在使用中以森林来表示。集就是让每个元素构成一个单元素的集合，并就是按一定顺序将属于同一组的元素所在的集合合并。并行查找，最终占据了此份榜单的第一名。