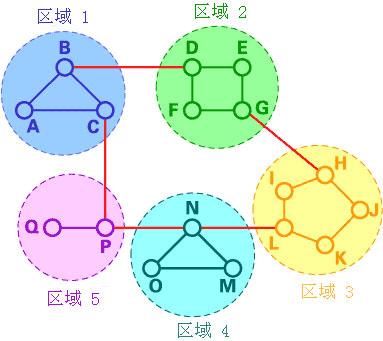
**优秀程序员必须知道的32个算法，提高你的开发效率**

2013-07-09 15:26 佚名 eoe Android开发者社区 字号：[**T**](javascript:setfont(12);) | [**T**](javascript:setfont(16);)

奥地利符号计算研究所（Research Institute for Symbolic Computation，简称RISC）的Christoph Koutschan博士在自己的页面上发布了一篇文章，提到他做了一个调查，参与者大多数是计算机科学家，他请这些科学家投票选出最重要的算法，以下是这次调查的结果，按照英文名称字母顺序排序。

AD：[干货来了,不要等！WOT2015 北京站演讲PPT开放下载！](http://wot.51cto.com/2015operation/)

[](http://s1.51cto.com/wyfs01/M01/10/1D/wKioOVHbvJmz3D1cAABORaFp_d8861.jpg)

1. A搜索算法——图形搜索算法，从给定起点到给定终点计算出路径。其中使用了一种启发式的估算，为每个节点估算通过该节点的最佳路径，并以之为各个地点排定次序。算法以得到的次序访问这些节点。因此，A\*搜索算法是最佳优先搜索的范例。
2. 集束搜索（又名定向搜索，Beam Search）——最佳优先搜索算法的优化。使用启发式函数评估它检查的每个节点的能力。不过，集束搜索只能在每个深度中发现最前面的m个最符合条件的节点，m是固定数字——集束的宽度。
3. 二分查找（Binary Search）——在线性数组中找特定值的算法，每个步骤去掉一半不符合要求的数据。
4. 分支界定算法（Branch and Bound）——在多种最优化问题中寻找特定最优化解决方案的算法，特别是针对离散、组合的最优化。
5. Buchberger算法——一种数学算法，可将其视为针对单变量最大公约数求解的欧几里得算法和线性系统中高斯消元法的泛化。
6. 数据压缩——采取特定编码方案，使用更少的字节数（或是其他信息承载单元）对信息编码的过程，又叫来源编码。
7. Diffie-Hellman密钥交换算法——一种加密协议，允许双方在事先不了解对方的情况下，在不安全的通信信道中，共同建立共享密钥。该密钥以后可与一个对称密码一起，加密后续通讯。
8. Dijkstra算法——针对没有负值权重边的有向图，计算其中的单一起点最短算法。
9. 离散微分算法（Discrete differentiation）
10. 动态规划算法（Dynamic Programming）——展示互相覆盖的子问题和最优子架构算法
11. 欧几里得算法（Euclidean algorithm）——计算两个整数的最大公约数。最古老的算法之一，出现在公元前300前欧几里得的《几何原本》。
12. 期望-最大算法（Expectation-maximization algorithm，又名EM-Training）——在统计计算中，期望-最大算法在概率模型中寻找可能性最大的参数估算值，其中模型依赖于未发现的潜在变量。EM在两个步骤中交替计算，第一步是计算期望，利用对隐藏变量的现有估计值，计算其最大可能估计值；第二步是最大化，最大化在第一步上求得的最大可能值来计算参数的值。
13. 快速傅里叶变换（Fast Fourier transform，FFT）——计算离散的傅里叶变换（DFT）及其反转。该算法应用范围很广，从数字信号处理到解决偏微分方程，到快速计算大整数乘积。
14. 梯度下降（Gradient descent）——一种数学上的最优化算法。
15. 哈希算法（Hashing）
16. 堆排序（Heaps）
17. Karatsuba乘法——需要完成上千位整数的乘法的系统中使用，比如计算机代数系统和大数程序库，如果使用长乘法，速度太慢。该算法发现于1962年。
18. LLL算法（Lenstra-Lenstra-Lovasz  lattice reduction）——以格规约（lattice）基数为输入，输出短正交向量基数。LLL算法在以下公共密钥加密方法中有大量使用：背包加密系统（knapsack）、有特定设置的RSA加密等等。
19. 最大流量算法（Maximum flow）——该算法试图从一个流量网络中找到最大的流。它优势被定义为找到这样一个流的值。最大流问题可以看作更复杂的网络流问题的特定情况。最大流与网络中的界面有关，这就是最大流-最小截定理（Max-flow min-cut theorem）。Ford-Fulkerson 能找到一个流网络中的最大流。
20. 合并排序（Merge Sort）
21. 牛顿法（Newton's method）——求非线性方程（组）零点的一种重要的迭代法。
22. Q-learning学习算法——这是一种通过学习动作值函数（action-value function）完成的强化学习算法，函数采取在给定状态的给定动作，并计算出期望的效用价值，在此后遵循固定的策略。Q-leanring的优势是，在不需要环境模型的情况下，可以对比可采纳行动的期望效用。
23. 两次筛法（Quadratic Sieve）——现代整数因子分解算法，在实践中，是目前已知第二快的此类算法（仅次于数域筛法Number Field Sieve）。对于110位以下的十位整数，它仍是最快的，而且都认为它比数域筛法更简单。
24. RANSAC——是“RANdom SAmple Consensus”的缩写。该算法根据一系列观察得到的数据，数据中包含异常值，估算一个数学模型的参数值。其基本假设是：数据包含非异化值，也就是能够通过某些模型参数解释的值，异化值就是那些不符合模型的数据点。
25. RSA——公钥加密算法。首个适用于以签名作为加密的算法。RSA在电商行业中仍大规模使用，大家也相信它有足够安全长度的公钥。
26. Schönhage-Strassen算法——在数学中，Schönhage-Strassen算法是用来完成大整数的乘法的快速渐近算法。其算法复杂度为：O(N log(N) log(log(N)))，该算法使用了傅里叶变换。
27. 单纯型算法（Simplex Algorithm）——在数学的优化理论中，单纯型算法是常用的技术，用来找到线性规划问题的数值解。线性规划问题包括在一组实变量上的一系列线性不等式组，以及一个等待最大化（或最小化）的固定线性函数。
28. 奇异值分解（Singular value decomposition，简称SVD）——在线性代数中，SVD是重要的实数或复数矩阵的分解方法，在信号处理和统计中有多种应用，比如计算矩阵的伪逆矩阵（以求解最小二乘法问题）、解决超定线性系统（overdetermined linear systems）、矩阵逼近、数值天气预报等等。
29. 求解线性方程组（Solving a system of linear equations）——线性方程组是数学中最古老的问题，它们有很多应用，比如在数字信号处理、线性规划中的估算和预测、数值分析中的非线性问题逼近等等。求解线性方程组，可以使用高斯—约当消去法（Gauss-Jordan elimination），或是柯列斯基分解（ Cholesky decomposition）。
30. Strukturtensor算法——应用于模式识别领域，为所有像素找出一种计算方法，看看该像素是否处于同质区域（ homogenous region），看看它是否属于边缘，还是是一个顶点。
31. 合并查找算法（Union-find）——给定一组元素，该算法常常用来把这些元素分为多个分离的、彼此不重合的组。不相交集（disjoint-set）的数据结构可以跟踪这样的切分方法。合并查找算法可以在此种数据结构上完成两个有用的操作：
    * 查找：判断某特定元素属于哪个组。
    * 合并：联合或合并两个组为一个组。
32. 维特比算法（Viterbi algorithm）——寻找隐藏状态最有可能序列的动态规划算法，这种序列被称为维特比路径，其结果是一系列可以观察到的事件，特别是在隐藏的Markov模型中。

以上就是Christoph博士对于最重要的算法的调查结果，InfoQ的读者们？你们熟悉哪些算法？又有哪些算法是你们经常使用的？

# [程序员必须知道的10大基础实用算法及其讲解](http://www.oschina.net/question/1397765_159365)

## 算法一：快速排序算法

**快速排序**是由东尼·霍尔所发展的一种排序算法。在平均状况下，排序 *n* 个项目要**Ο**(*n* log *n*)次比较。在最坏状况下则需要**Ο**(*n*2)次比较，但这种状况并不常见。事实上，快速排序通常明显比其他**Ο**(*n* log *n*) 算法更快，因为它的内部循环（inner loop）可以在大部分的架构上很有效率地被实现出来。

快速排序使用分治法（Divide and conquer）策略来把一个串行（list）分为两个子串行（sub-lists）。

**算法步骤：**

1 从数列中挑出一个元素，称为 “基准”（pivot），

2 重新排序数列，所有元素比基准值小的摆放在基准前面，所有元素比基准值大的摆在基准的后面（相同的数可以到任一边）。在这个分区退出之后，该基准就处于数列的中间位置。这个称为**分区（partition）**操作。

3 递归地（recursive）把小于基准值元素的子数列和大于基准值元素的子数列排序。

递归的最底部情形，是数列的大小是零或一，也就是永远都已经被排序好了。虽然一直递归下去，但是这个算法总会退出，因为在每次的迭代（iteration）中，它至少会把一个元素摆到它最后的位置去。

[](http://zezhi.qiniudn.com/wp-content/uploads/2014/06/Sorting_quicksort_anim.gif)

**详细介绍：**[**快速排序**](http://cricode.com/970.html)

## 算法二：堆排序算法

**堆排序**（Heapsort）是指利用堆这种数据结构所设计的一种排序算法。堆积是一个近似完全二叉树的结构，并同时满足*堆积的性质*：即子结点的键值或索引总是小于（或者大于）它的父节点。

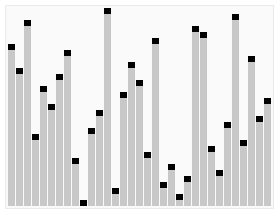
堆排序的平均时间复杂度为**Ο**(*n*log*n*) 。

**算法步骤：**

1. 创建一个堆H[0..n-1]
2. 把堆首（最大值）和堆尾互换

3. 把堆的尺寸缩小1，并调用shift\_down(0),目的是把新的数组顶端数据调整到相应位置

4. 重复步骤2，直到堆的尺寸为1

[](http://zezhi.qiniudn.com/wp-content/uploads/2014/06/Sorting_heapsort_anim.gif)

**详细介绍：**[**堆排序**](http://cricode.com/977.html)

## 算法三：归并排序

**归并排序（Merge sort，台湾译作：合并排序）**是建立在归并操作上的一种有效的排序算法。该算法是采用分治法（Divide and Conquer）的一个非常典型的应用。

**算法步骤：**

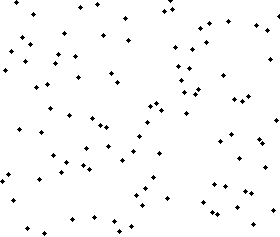
1. 申请空间，使其大小为两个已经排序序列之和，该空间用来存放合并后的序列

2. 设定两个指针，最初位置分别为两个已经排序序列的起始位置

3. 比较两个指针所指向的元素，选择相对小的元素放入到合并空间，并移动指针到下一位置

4. 重复步骤3直到某一指针达到序列尾

5. 将另一序列剩下的所有元素直接复制到合并序列尾

[](http://zezhi.qiniudn.com/wp-content/uploads/2014/06/Merge_sort_animation2.gif)

**详细介绍：**[**归并排序**](http://cricode.com/1333.html)

## 算法四：二分查找算法

**二分查找算法**是一种在有序数组中查找某一特定元素的搜索算法。搜素过程从数组的中间元素开始，如果中间元素正好是要查找的元素，则搜 素过程结束；如果某一特定元素大于或者小于中间元素，则在数组大于或小于中间元素的那一半中查找，而且跟开始一样从中间元素开始比较。如果在某一步骤数组 为空，则代表找不到。这种搜索算法每一次比较都使搜索范围缩小一半。折半搜索每次把搜索区域减少一半，时间复杂度为**Ο**(log*n*) 。

**详细介绍：**[**二分查找算法**](http://taop.marchtea.com/04.01.html)

## 算法五：BFPRT(线性查找算法)

**BFPRT算法**解决的问题十分经典，即从某n个元素的序列中选出第k大（第k小）的元素，通过巧妙的分 析，BFPRT可以保证在最坏情况下仍为线性时间复杂度。该算法的思想与快速排序思想相似，当然，为使得算法在最坏情况下，依然能达到o(n)的时间复杂 度，五位算法作者做了精妙的处理。

**算法步骤：**

1. 将n个元素每5个一组，分成n/5(上界)组。

2. 取出每一组的中位数，任意排序方法，比如插入排序。

3. 递归的调用selection算法查找上一步中所有中位数的中位数，设为x，偶数个中位数的情况下设定为选取中间小的一个。

4. 用x来分割数组，设小于等于x的个数为k，大于x的个数即为n-k。

5. 若i==k，返回x；若i<k，在小于x的元素中递归查找第i小的元素；若i>k，在大于x的元素中递归查找第i-k小的元素。

终止条件：n=1时，返回的即是i小元素。

**详细介绍：**

[寻找最小(最大)的k个数](http://cricode.com/968.html)

[**线性查找相关算法**](http://taop.marchtea.com/02.01.html)

## 算法六：DFS（深度优先搜索）

**深度优先搜索算法**（Depth-First-Search），是搜索算法的一种。它沿着树的深度遍历树的节点，尽可能深的搜索树的分 支。当节点v的所有边都己被探寻过，搜索将回溯到发现节点v的那条边的起始节点。这一过程一直进行到已发现从源节点可达的所有节点为止。如果还存在未被发 现的节点，则选择其中一个作为源节点并重复以上过程，整个进程反复进行直到所有节点都被访问为止。DFS属于盲目搜索。

深度优先搜索是图论中的经典算法，利用深度优先搜索算法可以产生目标图的相应拓扑排序表，利用拓扑排序表可以方便的解决很多相关的图论问题，如最大路径问题等等。一般用堆数据结构来辅助实现DFS算法。

**深度优先遍历图算法步骤：**

1. 访问顶点v；

2. 依次从v的未被访问的邻接点出发，对图进行深度优先遍历；直至图中和v有路径相通的顶点都被访问；

3. 若此时图中尚有顶点未被访问，则从一个未被访问的顶点出发，重新进行深度优先遍历，直到图中所有顶点均被访问过为止。

上述描述可能比较抽象，举个实例：

DFS 在访问图中某一起始顶点 v 后，由 v 出发，访问它的任一邻接顶点 w1；再从 w1 出发，访问与 w1邻 接但还没有访问过的顶点 w2；然后再从 w2 出发，进行类似的访问，… 如此进行下去，直至到达所有的邻接顶点都被访问过的顶点 u 为止。

接着，退回一步，退到前一次刚访问过的顶点，看是否还有其它没有被访问的邻接顶点。如果有，则访问此顶点，之后再从此顶点出发，进行与前述类似的访问；如果没有，就再退回一步进行搜索。重复上述过程，直到连通图中所有顶点都被访问过为止。

**详细介绍：**[**深度优先搜索**](http://www.cnblogs.com/skywang12345/p/3711483.html)

## ****算法七：BFS(广度优先搜索)****

**广度优先搜索算法**（Breadth-First-Search），是一种图形搜索算法。简单的说，BFS是从根节点开始，沿着树(图)的宽度遍历树(图)的节点。如果所有节点均被访问，则算法中止。BFS同样属于盲目搜索。一般用队列数据结构来辅助实现BFS算法。

算法步骤：

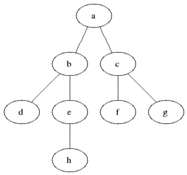
1. 首先将根节点放入队列中。

2. 从队列中取出第一个节点，并检验它是否为目标。

* 如果找到目标，则结束搜寻并回传结果。
* 否则将它所有尚未检验过的直接子节点加入队列中。

3. 若队列为空，表示整张图都检查过了——亦即图中没有欲搜寻的目标。结束搜寻并回传“找不到目标”。

4. 重复步骤2。

[](http://zezhi.qiniudn.com/wp-content/uploads/2014/06/Animated_BFS.gif)

**详细介绍：**[**广度优先搜索**](http://www.cnblogs.com/skywang12345/p/3711483.html)

## 算法八：Dijkstra算法

**戴克斯特拉算法**（Dijkstra’s algorithm）是由荷兰计算机科学家艾兹赫尔·戴克斯特拉提出。迪科斯彻算法使用了广度优先搜索解决非负权有向图的单源最短路径问题，算法最终得到一个最短路径树。该算法常用于路由算法或者作为其他图算法的一个子模块。

该算法的输入包含了一个有权重的有向图 *G*，以及G中的一个来源顶点 *S*。我们以 *V* 表示 *G* 中所有顶点的集合。每一个图中的边，都是两个顶点所形成的有序元素对。(*u*, *v*) 表示从顶点 *u* 到 *v* 有路径相连。我们以 *E* 表示*G*中所有边的集合，而边的权重则由权重函数 *w*: *E* → [0, ∞] 定义。因此，*w*(*u*, *v*) 就是从顶点 *u* 到顶点 *v* 的非负权重（weight）。边的权重可以想像成两个顶点之间的距离。任两点间路径的权重，就是该路径上所有边的权重总和。已知有 *V* 中有顶点 *s* 及 *t*，Dijkstra 算法可以找到 *s* 到 *t*的最低权重路径(例如，最短路径)。这个算法也可以在一个图中，找到从一个顶点 *s* 到任何其他顶点的最短路径。对于不含负权的有向图，Dijkstra算法是目前已知的最快的单源最短路径算法。

**算法步骤：**

1. 初始时令 S={V0},T={其余顶点}，T中顶点对应的距离值

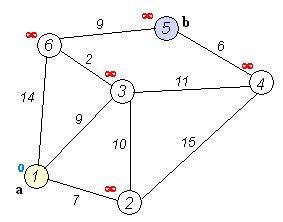
若存在<V0,Vi>，d(V0,Vi)为<V0,Vi>弧上的权值

若不存在<V0,Vi>，d(V0,Vi)为∞

2. 从T中选取一个其距离值为最小的顶点W且不在S中，加入S

3. 对其余T中顶点的距离值进行修改：若加进W作中间顶点，从V0到Vi的距离值缩短，则修改此距离值

重复上述步骤2、3，直到S中包含所有顶点，即W=Vi为止

[](http://zezhi.qiniudn.com/wp-content/uploads/2014/06/Dijkstra_Animation.gif)

**详细：**[**Dijkstra算法**](http://www.cnblogs.com/skywang12345/p/3711512.html)

## 算法九：动态规划算法

**动态规划**（Dynamic programming）是一种在数学、计算机科学和经济学中使用的，通过把原问题分解为相对简单的子问题的方式求解复杂问题的方法。 动态规划常常适用于有重叠子问题和最优子结构性质的问题，动态规划方法所耗时间往往远少于朴素解法。

动态规划背后的基本思想非常简单。大致上，若要解一个给定问题，我们需要解其不同部分（即子问题），再合并子问题的解以得出原问题的解。 通常许多 子问题非常相似，为此动态规划法试图仅仅解决每个子问题一次，从而减少计算量： 一旦某个给定子问题的解已经算出，则将其记忆化存储，以便下次需要同一个 子问题解之时直接查表。 这种做法在重复子问题的数目关于输入的规模呈指数增长时特别有用。

关于动态规划最经典的问题当属背包问题。

**算法步骤：**

1. 最优子结构性质。如果问题的最优解所包含的子问题的解也是最优的，我们就称该问题具有最优子结构性质（即满足最优化原理）。最优子结构性质为动态规划算法解决问题提供了重要线索。

2. 子问题重叠性质。子问题重叠性质是指在用递归算法自顶向下对问题进行求解时，每次产生的子问题并不总是新问题，有些子问题会被重复计算多次。 动态规划算法正是利用了这种子问题的重叠性质，对每一个子问题只计算一次，然后将其计算结果保存在一个表格中，当再次需要计算已经计算过的子问题时，只是 在表格中简单地查看一下结果，从而获得较高的效率。

**详细参考：**

[从全球导航到输入法：谈谈动态规划](http://cricode.com/1096.html)

[**动态规划**](http://hawstein.com/posts/dp-novice-to-advanced.html)

## 算法十：朴素贝叶斯分类算法

朴素贝叶斯分类算法是一种基于贝叶斯定理的简单概率分类算法。贝叶斯分类的基础是概率推理，就是在各种条件的存在不确定，仅知其出现概率的情况下， 如何完成推理和决策任务。概率推理是与确定性推理相对应的。而朴素贝叶斯分类器是基于独立假设的，即假设样本每个特征与其他特征都不相关。

朴素贝叶斯分类器依靠精确的自然概率模型，在有监督学习的样本集中能获取得非常好的分类效果。在许多实际应用中，朴素贝叶斯模型参数估计使用最大似然估计方法，换言之朴素贝叶斯模型能工作并没有用到贝叶斯概率或者任何贝叶斯模型。

尽管是带着这些朴素思想和过于简单化的假设，但朴素贝叶斯分类器在很多复杂的现实情形中仍能够取得相当好的效果。

**详细参考：**

[贝叶斯网络](http://cricode.com/1078.html)

[**朴素贝叶斯分类算法**](http://cricode.com/1098.html)

本文链接：[程序员必须知道的10大基础实用算法及其讲解](http://cricode.com/2001.html)

出处：[快课](http://cricode.com/)