**1.分治法**

关键词：递归（递归式）、大问题分解成子问题（子问题相互独立，且与原问题相同）、合并（子问题的解合并成原问题的解）

　　步骤：

　　1.分解。将原问题分解成一系列子问题。

　　2.求解。递归地求解各子问题。若子问题足够小，则直接求解。（递归式）

　　3.合并。将子问题的解合并成原问题的解

　　示例：归并排序；最大子段和问题；循环赛日程安排

**2.动态规划法**

关键词：递归（递归式）、表记录（已解决的子问题的答案）、根据子问题求解原问题的解（子问题不独立）、最优解（可选项）

　　步骤：

　　1.找出最优解的性质，刻画其结构特征；

　　2.递归地定义最优解；

　　3.以自底向上的方式计算出最优值；

　　4.根据计算最优值时得到的信息，构造一个最优解

　　只需求出最优值，步骤4可以省略；若需求出问题的一个最优解，则必须执行步骤4。

　　适用环境：

　　1.最优子结构。一个问题的最优解包含了其子问题的最优解。

　　2.重叠子问题。原问题的**递归算法**可以反复地解同样的子问题，而不是总是产生新的子问题

　　示例：0-1背包问题；矩阵链乘问题；最长公共子序列（LCS）；

**3.贪心法**

关键词：局部最优（较好的近似最优解，贪心）、简单、根据当前信息最选择，且不改变、

　　使用环境：

　　1.最优子结构。一个问题的最优解包含了其子问题的最优解。

　　2.贪心选择性质。问题的整体最优解可以通过一系列局部最优的选择（贪心选择）来得到

　　示例：活动选择问题、背包问题、多机调度问题

**4.回溯法**

关键词：通用的解题法、解空间树（深度优先遍历）、界限函数、所有解（找出满足条件的所有解）

　　步骤：

　　1.针对所给问题，定义问题的解空间。问题的解空间应至少包含问题的一个（最优）解

　　2.确定易于搜索的解空间结构。通常将解空间表示为树、图；解空间树的第i层到第i+1层边上的标号给出了变量的值；从树根到叶子的任一路径表示解空间的一个元素。

　　3.以深度优先的方式搜索整个解空间。如果当前宽展节点处为死节点，则回溯至最近的一个活节点处。（以此方式递归搜索）

　　算法框架：非递归、递归

　　界限函数：回溯法的核心。尽可能多、尽可能早地“杀掉”不可能产生最优解的活节点。好的界限函数可以大大减少问题的搜索空间，大大提高算法的效率。

　　示例：0-1背包、N皇后问题

**5.分支界限法**

关键字：解空间（广度优先、最小耗费优先）、界限函数（队列式、优先队列式）

　　步骤：

　　1.针对所给问题，定义问题的解空间。问题的解空间应至少包含问题的一个（最优）解

　　2.确定易于搜索的解空间结构。通常将解空间表示为树、图；解空间树的第i层到第i+1层边上的标号给出了变量的值；从树根到叶子的任一路径表示解空间的一个元素。

　　3.以广度优先或最小耗费优先的方式搜索整个解空间。每个活节点只有一次机会成为扩展节点，活节点一旦成为扩展节点，其余儿子节点被加入活节点表中。（以此方式递归搜索）

　　界限函数：分支界限法的核心。尽可能多、尽可能早地“杀掉”不可能产生最优解的活节点。好的界限函数可以大大减少问题的搜索空间，大大提高算法的效率。

　　1.队列式（FIFO）分支界限法。先进先出

　　2.优先队列式分支界限法。组织一个优先队列，按优先级选取。通常用最大堆来实现最大优先队列，最小堆来实现最小优先队列。

**6.概率算法**

关键词：随机性选择、小概率错误（运行时间大幅减少）、不同解（对同一问题求解两次，可能得到完全不同的解，且所需时间、结果可能会有相当大的差别）

　　基本特征：

　　1.输入包括两部分。一，原问题的输入；二，供算法进行随机选择的随机数序列

　　2.运行过程中，包括一处或多处随机选择，根据**随机值**来决定算法的运行

　　3.结果不能保证一定是正确的，但可以限制出错率。

　　4.不同的运行过程中，对于相同的输入实例可以有不同的结果，执行时间也可能不同。

　　分类：

　　1.数值概率算法。常用于数值问题的求解。近似解，近似解的精度随计算时间的增加不断提高。

　　2.蒙特卡罗（Monte Carlo）算法。精确解，解未必是正确的，正确的概率依赖于算法所用的时间。一般情况下，无法有效地判定所得到的解是否肯定正确。

　　3.拉斯维加斯（LasVegas）算法。一旦找到解，一定是正确解。找到的概率随计算时间的增加而提高。对实例求解足够多次，使求解失效的概率任意小。

　　4.舍伍德（Sherwood）算法。总能求得问题的一个解，且所求得的解总是正确的。设法消除最坏情形与特定实例之间的关联性。多用于最快情况下的计算复杂度与其在平均情况下的计算复杂度差别较大。

**7.近似算法**

　　关键词：近似解、解的容错界限（近似最优解与最优解之间相差的程度）、不存在多项式时间算法

　　基本思想：放弃求最优解，用近似最优解替代最优解。使算法简化，时间复杂度降低

　　衡量性能的标准：

　　1.算法的时间复杂度。时间复杂度必须是多项式阶的

　　2.解的近似程度。与近似算法本身、问题规模、不同的输入实例有关。

　　示例：NP问题、定点覆盖问题、TSP、子集和数问题、