# 问答

算法的定义：

算法是一个有穷的指令集，这些指令为解决某一特定任务规定了一个运算序列。

特性：

有穷性 算法应在执行有穷步后结束；

有效性 每一条运算都是可行的；

确定性 每步定义都是确切无歧义的；

输入 有0个或多个输入；

输出 有一个或多个输出(处理结果)；

什么是算法？它有哪些特点？

算法（ algorithm ）：有限指令序列, 某个问题能够得出正确答案的规则集合。有5个特点： 确定性，可终止性，可行性，输入，输出。

什么是最优算法？

算法时间复杂度的上界等于求解问题固有时间复杂度的下界。

请指出快速排序、Floyd、二叉树遍历等算法的基本运算各是什么？

快速排序：关键字比较，Floyd：长度比较，二叉树遍历：对顶点的访问。

递归算法三要素：

1）定义一个最小规模的问题，并给出其解。

2）把复杂的问题划分为同类型的若干规模较小的子问题，并分别解决子问题。

3）把各子问题的解组合起来，即得到原问题的解。

最坏情况下时间复杂度：

Dn 是问题的所有尺寸为n的输入集，  
I是 Dn 的元素，  
t(I)是算法对输入I所做工作量，  
用W(n)=max{t(I)| I属于Dn} 表示 最坏情况下时间复杂度。

平均时间复杂度：

设I是算法A的所有规模为n的输入集，Ik∈I，pk是该输入的概率，tk是输入Ik时，算法的执行时间，平均时间复杂度是：



时、空复杂度之间有关系吗？ 是什么关系？（

时间复杂度和空间复杂度往往是相互矛盾的，常常可用时间换取空间，也可以用空间换取时间。

解释Ο、Ω和Θ含义。

Ο是上界、Ω是下界，Θ是同数量级。

**求Θ**

**排序时间复杂度**

贪婪算法可解的问题有如下特性：

（1）优化问题，有一个候选对象的集合，如硬币，边，路径，顶点等。

（2）随着算法的进行，形成2个集合，一个是已经被选中的对象集合，另一个是被抛弃的对象集合

（3）有一个函数（solution），检查候选对象集是否是问题的解，不一定是最优的。

（4）还有一个函数，检查候选对象是否可加入到当前解的对象集合中，不一定是最优的。

（5）选择函数（selection function），指出哪个剩余的候选对象最有可能构成问题的解。

（6）目标函数(object function)，给出解的值。如硬币个数，路径长度，顶点个数等。

function greedy(C:set):set

{C是候选对象集合}

S=Φ {在集合S中构造解}

while C<>Φ and not solution(S) do

x=select(C)

C=C\{x}

if feasible(S∪{x}) then S=S∪{x}

if solution(S) then return S

else return “No solution”

什么是P类问题？什么是NP类问题？什么是NP-Complete类问题？

（1）P类问题：可以以多项式时间算法求解的一类判断问题；

（2）NP类问题：判定问题A, 如果给定一个解 C，有一个多项式界的算法A可以确定C是一个正确的解。

（3）NP-Complete问题：如果所有NP问题都可多项式时间归约为问题某NP问题，则该NP问题是NP-Complete问题。

P类、NP类和NP-Complete类问题之间的关系是什么？

NP包含了P类和NP-Complete类问题，所有NP类问题都能多项式规约为NP-Complete问题，NP是否等于P问题目前还无定论。

请列出3个NP-Complete问题：

邮递员问题、0-1背包问题、最小覆盖问题、图的着色问题、最大团问题

什么是NP类问题？ 什么是NP-Complete类问题？如何证明一个问题是NP-Complete的？

（3）首先证明问题A是NP问题，然后证明一个已知的NP-Complete问题B可以多项式归约为A；

在什么情况下NP=P？

当NP-Complete问题具有多项式算法时；

问题：平面上的n个点，是否有k个点在同一直线上。该问题是否是一个NP-Complete 问题，简单说明为什么。

该问题是一个 NP-Complete 问题。这是一个组合问题，需要对n个点中的任意k个点的子集进行检测是否在一条直线上，共有 C(k,n)个子集，取其中的任意两个点，建立直线方程，然后检测其他点是否在此直线上。时间复杂度是C(k,n)\*k，是指数级的。

平面上有n个点，是否有k个点在同一直线上。证明该问题是一个NP类问题。

证明：给定任意的k个点(x1,y1),(x2,y2),…,(xk,yk)，首先求出用(x1,y1)和(x2,y2)求出直线公式

然后分别验证其他各点是否在此直线上，可在O(k)次验证完。所以是NP类问题。

请列出3个P类的优化问题：

最短路径问题，最小生成树问题，关键路径问题

**证明TSP（货郎担）问题是NP类的**

一个动态规划算法，通常可以按以下几个步骤进行：

（1）找出最优解的性质，并刻画其结构特征。

（2）递归地定义最优值。

（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。（该步骤是求最优解的必须步骤）

动态规划算法中的最优性是什么？

解决一个问题的过程中，需要依次作出n个决策D1,D2,…Dn,若这个决策序列是最优的，对于任何一个整数k，1<k<n，不论前面k个决策是怎样的，以后的最优决策只取决于前面的决策，即前k个是最优的，则k+1也是最优的。

搜索算法的基本思想：

搜索法可以说是解决问题的万能法。组合优化问题在不同判定阶段都有不同的状态，在当前状态下，面临着许多个可选的后继状态，这样一层一层下去，构成一棵树。通常预先不知道选择哪个后继状态更好 ，一个直接的方法是对每个状态都试一下，以获得最优解。显然对状态树进行搜索，一定可以获得最优解。但是状态的数量可能是非常的大。

2种概论算法：

p-正确的蒙特卡罗算法：设0<p<1，如果它对任何实例都以至少为p 的概率返回正确的解。p依赖于实例的大小，但不依赖于实例本身。有时会给出错误的解，概率很低，并且不知道解是错误的

Las Vegas算法进行概率性选择，帮助和指导我们更快地得到正确的解。不会给出错误答案，而给出一个提示。 Las Vegas 算法有2种：

(1) 总能给出正确答案，只是选择不合适时，花的时间长。

(2) 选择不恰当，可能导致算法陷入绝境，它会报告找不到解。

A\*算法的搜索策略是什么，是如何搜索的？

A\*算法是A算法的改进,每个状态都有一个评估函数f(x)=g(x)+h\*(x),

g(x):是有根状态到达当前状态x的以用代价；到达x后，该值是已知的。

h\*(x):是从当前状态x到终止状态要花费代价的实际值；是未知的。在算法中只能估计一个值h(x).

h(x):是从当前状态x到终止状态要花费的代价的估计值；要求： h(x) <= h\*(x)。

在田字格中放A、B（可重复出现），分析其状态空间，有多少个状态？ 初始状态是什么？终止状态是什么？

初态：空格；共有81个，终态是所有格放满，有16个。

搜索算法有哪几种搜索法？各有什么特点？

搜索算法主要有深度优先搜索和广度优先搜索法

遗传算法：

遗传算法的基本运算过程如下：

a)初始化：设置进化代数计数器t=0，设置最大进化代数T，随机生成M个个体作为初始群体P(0)。

b)个体评价：计算群体P(t)中各个个体的适应度。

c)选择运算:将选择算子作用于群体。选择的目的是把优化的个体直接遗传到下一代或通过配对交叉产生新的个体再遗传到下一代。选择操作是建立在群体中个体的适应度评估基础上的。

d)交叉运算：将交叉算子作用于群体。所谓交叉是指把两个父代个体的部分结构加以替换重组而生成新个体的操作。遗传算法中起核心作用的就是交叉算子。

e)变异运算：将变异算子作用于群体。即是对群体中的个体串的某些基因座上的基因值作变动。

　　群体P(t)经过选择、交叉、变异运算之后得到下一代群体P(t 1)。

f)终止条件判断:若t=T,则以进化过程中所得到的具有最大适应度个体作为最优解输出，终止计算。

遗传算法特点:

　　遗传算法是解决搜索问题的一种通用算法，对于各种通用问题都可以使用。[搜索算法](http://baike.baidu.com/view/3688332.htm)的共同特征为：

　① 首先组成一组候选解

　② 依据某些适应性条件测算这些候选解的适应度

　③ 根据适应度保留某些候选解，放弃其他候选解

　④ 对保留的候选解进行某些操作，生成新的候选解。

模拟退火算法：

模拟退火算法采用类似于物理退火的过程，

先在一个高温状态下（相当于算法随机搜索），然后逐渐退火，

在每个温度下（相当于算法的每一次状态转移）徐徐冷却

（相当于算法局部搜索），最终达到物理基态

（相当于算法找到最优解）。

模拟退火算法的优点

质量高；

初值鲁棒性强；

简单、通用、易实现。

模拟退火算法的缺点

由于要求较高的初始温度、较慢的降温速率、较低的终止温度，以及各温度下足够多次的抽样，因此优化过程较长

递归算法三要素：

1）定义一个最小规模的问题，并给出其解。

2）把复杂的问题划分为同类型的若干规模较小的子问题，并分别解决子问题。

3）把各子问题的解组合起来，即得到原问题的解。

贪婪算法可解的问题有如下特性：

（1）优化问题，有一个候选对象的集合，如硬币，边，路径，顶点等。

（2）随着算法的进行，形成2个集合，一个是已经被选中的对象集合，另一个是被抛弃的对象集合

（3）有一个函数（solution），检查候选对象集是否是问题的解，不一定是最优的。

（4）还有一个函数，检查候选对象是否可加入到当前解的对象集合中，不一定是最优的。

（5）选择函数（selection function），指出哪个剩余的候选对象最有可能构成问题的解。

（6）目标函数(object function)，给出解的值。如硬币个数，路径长度，顶点个数等。

function greedy(C:set):set

{C是候选对象集合}

S=Φ {在集合S中构造解}

while C<>Φ and not solution(S) do

x=select(C)

C=C\{x}

if feasible(S∪{x}) then S=S∪{x}

if solution(S) then return S

else return “No solution”

一个动态规划算法，通常可以按以下几个步骤进行：

（1）找出最优解的性质，并刻画其结构特征。

（2）递归地定义最优值。

（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。（该步骤是求最优解的必须步骤）

搜索法可以说是解决问题的万能法。组合优化问题在不同判定阶段都有不同的状态，在当前状态下，面临着许多个可选的后继状态，这样一层一层下去，构成一棵树。通常预先不知道选择哪个后继状态更好 ，一个直接的方法是对每个状态都试一下，以获得最优解。显然对状态树进行搜索，一定可以获得最优解。但是状态的数量可能是非常的大。

2种概论算法：

p-正确的蒙特卡罗算法：设0<p<1，如果它对任何实例都以至少为p 的概率返回正确的解。p依赖于实例的大小，但不依赖于实例本身。

Las Vegas算法进行概率性选择，帮助和指导我们更快地得到正确的解。不会给出错误答案，而给出一个提示。

Las Vegas 算法有2种：

(1) 总能给出正确答案，只是选择不合适时，花的时间长。

(2) 选择不恰当，可能导致算法陷入绝境，它会报告找不到解。