算法：有输入输出的、有限的、确定的、有效的过程

程序：实现算法的一种方式

递归分治，动态规划，贪心，回溯分支（数学推导）

算法设计：抽象为数学模型、写求解过程

　　随机算法：按照一定概率进行选择

　　描述：自然语言、流程图、程序语言、Pseudocode

算法评估：Empirical（实验结果->理论推导），Theoretical（算法效率：时间/空间）

　　　　W(n)：最坏情况时间复杂度

　　　　A(n)=∑P()*t：平均时间复杂度

　　　　基本运算的执行次数（加减、比较）

　　时间复杂度：时间的函数表示基本运算 （I：输入 N：问题规模）

　　复杂性的**渐近性态**：舍弃低阶项，不必考虑常数因子（**输入规模趋近于无穷**）

　　渐近上，下界记号：

　　　　　　　　O（存在 c 与 n0，使 n>=n0 都有 f(n)<=c*g(n),则 f(n)=O(g(n))（<=）

　　　　　　　　/=最坏情况下时间复杂度（输入 I 相关）

　　　　　　　　Ω（存在 c 与 n0，使 n>=n0 都有 f(n)>=c*g(n),则 f(n)= Ω(g(n))（>=）

　　紧渐近界限记号：**Θ**（c1，c2，n0 被两个相似函数夹住 **O 且 Ω**）（=）

　　非紧上，下界限：**o**（f(n)/g(n)->0），**ω**（f(n)/g(n)->∞）（<，>）

Determinism（确定性算法）：每一步有确定选择

Polynomial（P 类问题）：多项式时间内**求解**

Non- Determinism（非确定性算法）：穷举并用确定性算法验证

Non- Deterministic Polynomial（**NP**）：多项式时间内**可验证**

Non- Deterministic Polynomial Complete Problem（NPC NP 完全问题）：NP 问题可以归约至 NPC 问题

Non- Deterministic Polynomial Hard Problem NP 难问题：**不一定为 NP 问题**，NPC 问题可归约到 NP 难问题（不一定在多项式时间内**可验证**）

　　问题复杂度不会超过**解决其的算法**的复杂度

　　判定问题（decision problem）：证明易于求解

判定形式的 NP 完全问题的**最优化**为 NP 难问题

TSP：时间复杂度 n! 最优化：min{∑d（Ck,Ck+1）+d（Ck,C1）} （d（Ck,Ck+1）存在）

递归分治：

　　分治：分解问题，解决相同子问题 递归：直/间接调用自身（终止，通项条件）

　　递归：

　　　　双递归函数：Ackerman 函数

　　　　整数划分：q(n, m) m 为最大加数

1. q(n, 1)=1 （n>=1）
2. q(n, m)=q(n, n) （m>=n）
3. q(n, n)=1+q(n,n-1)
4. q(n, m)=q(n-m, m)+q(n,m-1) (n>m>1)

　　　　Hanoi 塔：Hanoi(n, A, B, C)（从 A 利用 B 移到 C）

1. n=1, move(A, C)
2. Hanoi(n-1, A, C, B)
3. Move(A, C)

4. Hanoi(n-1, B, A, C)

分治：

问题具有**最优子结构性质**（二分搜索）

分解问题，解决小问题，合并，**子问题相互独立**（非独立则使用动态规划）

分治：

大整数乘法：

X=a(n/2 位)b(n/2 位) Y=c(n/2 位)d(n/2 位)

XY=ac2^n+(ad + bc) *2^(n/2)+bd

快速傅里叶变换

Strassen 矩阵乘法：

将矩阵分为大小相等的子矩阵 ， 使用分治法降阶求出子矩阵

T(n)=8T(n/2)+O(n^2)

排序问题：

二分归并排序：划分，求解子问题，合并（时间复杂度 O(n log n)，稳定）

快速排序：选择轴值与右指针，从右开始扫描，遇小交换，从左扫描（不稳定 平均情况 **O(n log n) 差消法**）

最近点对问题：分为左右两个点集，再考虑跨越边界的点对

递推方程：T（n）=2T（n-1）+1（终止条件：T(1)=1）

迭代法求解（找出 T(n)通项公式，**数学归纳法验证**正确性）

换元迭代法：将 n 转换为变元 K 的递推，

T(n)=4T(n/2)+O(n)

T(n)=2T(n/2)+n-1，令 n=2^k

T(n)=a*T(n/b)+f(n)

公式：$\mathbf{T(n)=n^{\log_b a} * \sum a^i f(\frac{n}{b^i})}$

递归树验证换元迭代法：

T(N)等于树上所有节点的值（非函数项）

W(m)=（函数项）W(m1)+…+W(m k)+（节点值 非函数项）f(m)+…+g(m)

主定理 Master：T(n)=a*T(n/b)+f(n)

1. 若 f(n)**阶小**（$<O\left(n^{\log_b a}-e\right)$），则$T（n）=\Theta(n^{\log b^a})$
2. 若与 f(n)**同阶（Θ）**，则$T(n)=\Theta(n^{\log_b a} * \log^n)$
3. 若 f(n)**阶大**($>\Omega(n^{\log_b a}+e)$，且存在 c<1 使$af\left(\frac{n}{b}\right)=cf(n)$，则$T(n)=\Theta(f(n))$

主定理三的两个条件需同时满足才能使用

动态规划：

子问题及其间依赖关系确定（填表+追踪解）

设计备忘录

最优化问题：解空间中满足约束条件的可行解中可使目标函数取得极值的解

最优化原理：**多段决策过程**，子问题为原始问题的一个子集

最优子结构+重叠子问题性质：最大子问题的最优解包含其他**子问题最优解**

Dynamic **P**rogramming：重叠子问题，最优子结构性质,通过表记录已知结果，避免重复

计算 **选择向量**记录选择结果

1. 建模，寻找**目标函数与约束条件**
2. 分段，将问题**分解为子问题**，确定子问题**边界**
3. 分析，原/子问题间**依赖关系**
4. 判断，是否满足**最优子结构性质**
5. 确定最小子问题初值，**自底向上**，利用已知结果简化计算**选择向量**

最短路径问题：

子问题界定：后边界不变，前边界前移

$f(k_i) = min\{k_i k_{i-1} + f(k_{i-1})\}$

设计要素：目标函数，约束条件，边界，递推方程，最优化原则，最小子问题界定

矩阵连乘：加括号划分至只有两两连乘（简化矩阵连乘运算）

A<i,j>，B<j,k> AB 相乘共需 ijk 次运算

迭代：时间复杂度低，空间消耗多

保存子问题的解结果并**标记决策**（m 子问题最少次数，s 划分方式 k 的值）

1. 确定链长 r 1~n
2. 左右边界 i，j
3. 遍历确定划分位置 k，

$$\mathbf{m[i,j] = min\{m[i,k] + m[k+1,j]\} + (\textit{子矩阵相乘})} \boldsymbol{p_{i-1} p_k p_j}$$

4. 更新解 m[i, j]

时间复杂度：$T(n) = O(n^3)$（三次 O(n)循环，n^2个备忘录一个 O(n)）

动态规划时间复杂度：备忘录计算+各项合并时间

最长公共子序列 LCS：

子序列：从给定序列中按序任意选出一段

Longest **C**ommon **S**ubsequence：序列 X，Y 的最长的相同子序列

穷举：2^m个子序列，$O(n*2^m)$

DP：

构造解 O(m*n)，追踪解 O(m+n)

空间复杂度 O(m*n)

1. **i=0/j=0，C[i, j]=0(设置初值** C[i, j]=0**)**
2. 若 Xn=Yn， Zk 为 Xn-1 与 Yn-1 的 LCS+1
3. 若 Xn≠Yn，Zk=max{ C[i, j-1]，C[i-1, j] }

标记函数 B[i, j]，左上 ↖（LCS+1）/左 ←（C[i, j-1])/上↑(C[i-1, j])

背包问题 KP：

填表 O(n*b)（**伪多项式时间**算法 b 为常数：输入规模 $\boldsymbol{n, logb}$，$\mathbf{O}(\boldsymbol{n * 2^{logb}})$）

约束条件∑wi*xi≤b（最大重量）

物品(vi, wi) 使∑vi*xi 最大

线性规划：目标函数与约束条件为线性函数

$F_k(y) = max\{F_{k-1}(y), F_k(y - w_k) + v_k\}$

K=1 时，Fk(y)初值为y/w_k向下取整

贪心算法：

最优子结构，**贪心选择**性质

Greedy Algorithm：每步选择均为**局部最优选择**（最优解的近似解）

选择性质：整体最优可通过局部最优得到，**自顶向下**

背包问题：

一般背包（物品数可为小数）：每次均装入性价比最大的

0-1 背包（物品为整数）：贪心无法保证背包装满

活动选择问题：

N 个活动集合，s_i/f_i 开始/结束时间

1. 截止时间排序，选取第一个
2. 遍历比较，若$f_1 \le s_i$选取n_i

证明：

K=1，假设存在 j≠1 的活动为 A 的第一个活动，用 1 替换必定满足

K=n→K=n+1，假设对前 K=n 为真，证明 K+1 为真

最优装载问题：

装载能力 C，无体积限制（0-1 背包子问题）

策略：轻者优先

正确性证明：

数学归纳法：

第一数归：归纳基础（P(1)为真），归纳步骤（假设 P(n)为真，证明 P(n+1)真）$p(n) \to p(n+1)$

第二数归：归纳基础（P(1)为真），归纳步骤（假设 P(1,2⋯n)为真，证明 P(n+1)真）$P(1) \cap P(2) \cap \cdots P(n) \to P(n+1)$

描述**与自然数 n 有关命题**，证明第一个命题成立然后利用数归证明贪心正确

二元前缀码：

0-1 代码表示字符，字符代码不能为其它前缀

Huffman 哈夫曼树：选取频率最小字符并形成子根，构建向上构建树

最小生成树：n 阶联通图 G 的生成树 T 无环且含 n-1 条边

Prim 算法：选取结点 S{1}，$\mathrm{while}(V! = S)$选取到集合 S 距离最小的顶点 j 加入 S（更新维护堆$\mathrm{O(logn)}$）（运行 n-1 次，每次执行$O(n), T(n) = O(nlogn)$）（点少适用）

Kruskal 算法：排序并选择最小边，判断不形成回路则加入（并查集：更新父结点，若两结点根节点相同，则不能将它们连接）$T(n) = O(mlogm)$（边少适用）

单源最短路径：带权有向图，选取原点到其它所有点的最短路径

Dijkstra 算法：S=s{u}，S=V 时结束，维护节点到 S 最短路径$\mathrm{dist[i]}$，选取dist中最小的 j 加入 S 并修改dist（用堆维护dist，$T(n) = O(m \log n)$）

多机调度问题 NPC：使用贪心**近似**求解

近似比：$\mathbf{Ap} = \frac{\boldsymbol{A()}}{\boldsymbol{Ope()}}$（贪心策略/最优策略，取及差下限）

回溯 Backtracking：

组织解空间，跳跃性穷举（找出所有解，比较得出最优解）

条件：多米诺性质 $\boldsymbol{(\neg p(x_1 x_2 \cdots x_k) \to \neg p(x_1 x_2 \cdots x_{k+1}))}$ 进行减枝

深度/宽度优先搜索，满足（扩张解向量，继续搜索），不满足（回溯至父节点）

显约束（解空间），隐约束（约束条件）

4 皇后问题：解向量$< x_1, x_2, x_3, x_4 >$（使用 **n 叉树**表示解空间）$O(n^n)$

0-1 背包问题：bool $< x_1, x_2, \dots x_i >$ 表示是否选择物品，**子集树**含$O(2^n)$片树叶

旅行商问题：<k1,k2⋯,ki>城市排列，O(n!)个节点的**排列树**

$$\min\{\sum_{i=1}^{n} d(k_{i-1},k_i)\}+d(k_i,k_1)$$

递归：

1. 确定 Xi 初始取值
2. 计算满足约束的分支 Sk
3. 遍历解空间 Sk，取出最大节点
4. If(k==n) $< x_1,x_2,\dots x_k >$为解，否则 goto 2

迭代：

1. If(k>n) $< x_1,x_2,\dots x_k >$为解
2. 遍历 Sk，计算 Sk+1 并调用 ReBack(k+1)（跳跃性：递归调用，保存现场）

ReBack（沿右分支遍历向左，遇右遍历右分支，遇根停止）

轮船载重问题：装入第一艘船，使其剩余空间尽量少 ($O(2^n)$)

1. 遍历 wi，若满足 x[i]置 1，否则置 0
2. 记录当前 best
3. 回溯（**沿着右分支向上回溯，直到发现为左分支再向右回溯**（至根停止））

图着色问题：解空间（m 叉树，一个节点有 m 种可能着色方案）($O(n*m^n)$)

每个节点选取可能颜色，若不满足则回溯

分支限界

以**广度/最小耗费优先**方式搜索解空间（根据**代价函数**调整搜索，细粒度的回溯）

队列式，栈式优先节点

代价函数：子树区域内所有可行解的上界（极大化 初值为 0）（极小化相反）

界：已知最大**可行解**

背包问题：代价函数（已装入价值+Δ（剩余背包*可行最大性价比））

0-1 背包：代价函数高方向搜索（广度优先 使最先找出的界最优）

非对称 TSP：优先访问当前最低+未选城市的最低出发票价之和（最小化问题）

分布式计算不同分支

最大团问题：

最大团：无向图$G=< V,E >$顶点最多的完全子图

点独立集：U 是 G 的最大团当且仅当 U 是$\bar{G}$（G 的补图）的最大点独立集

选择树$< x_1x_2\dots x_n >$，代价函数$\mathrm{F}=C_k+n-k$

约束条件：与当前被选择所有节点均有边

随机化算法

函数难以化简，多次随机产生值代入

随机数值算法：算法输出近似解，精确度与运行时间正比（随机投点，计算比例）

Sherwood 算法：$\bar{t}_A(n)=\sum_{x\epsilon X_n}\frac{t_A}{|x_n|}$，将输入实例排序，**消除最坏行为影响**

Las Vegas 算法：随机产生解并带入验证（随机+回溯 设置合适的随机数）

Monte **C**arlo 算法：获得精确解的概率与算法执行时间正比（反复运行）

主元素：数组中一半以上均为 x，寻找是否有主元素

线性规划与网络流：

线性规划：目标函数与约束条件均为线性

单纯形法：持续寻找基本可行解（约束条件内）直到找到最优解

网络流：网络 G=(V,E)，有源点 s 与汇点 t，边 e 有最大容量 cap(e)

兔子匹配 **FF**:

1. 找出一条 s->t
2. 构建残余网络，反向路径（边容量=流大小）
3. 于残余网络中重复寻找直至无法找到

从汇点返回到源点提取流