**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

**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）（=）

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)+1(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)

**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）

分治：

问题具有**最优子结构性质**

分解问题，解决小问题，合并（**平衡子问题**）