P，NP，NP Complete，NP Hard

目录

算法评估

算法的效率主要由以下两个复杂度来评估:

时间复杂度: 评估执行程序所需的时间，可以估算出程序对处理器的使用程度。

空间复杂度: 评估执行程序所需的存储空间，可以估算出程序对计算机内存的使用程度。

Recall： Fibonacci Sequence

1，1，2，3，5，8，13…..

```
def f(n):
    if n==1 or n==2: return 1
    else:  return f(n-1) + f(n-2)
```

Time complexity：？ $O(2^n)$

Space complexity：？O(n)

时间复杂度

时间复杂度：随着问题规模的增大，算法执行时间增长的快慢。
它可以用来表示一个算法运行的时间效率。

注意：时间复杂度不是计算机执行的真实运行时间，而是关注数量级增大后的执行时间趋势。

算法复杂度可从最理想情况、平均情况和最坏情况三个角度评估，由于平均情况大多和最坏情况持平，一般情况下直接估算最坏情况的复杂度。

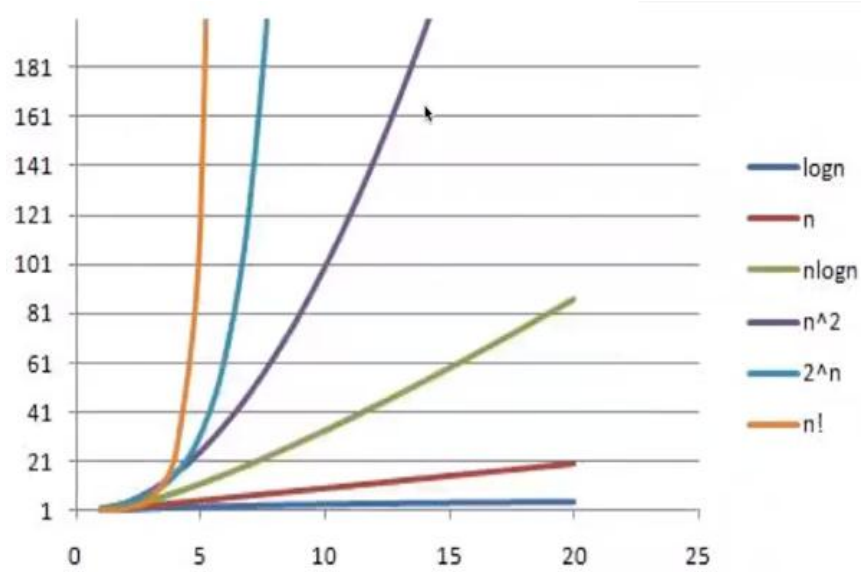
复杂度使用大O表示方法，如多项式时间复杂度表示为$O(n^k)$
多项式：$f(x)=ax^k+bx^{k-1}+\ldots+c$称为x的k阶多项式

时间复杂度

常见函数的时间复杂度

f(n)	阶	函数名	阶名
1	O(1)	常数函数	常数阶
2n+1	O(n)	线性函数	线性阶
$2n^2+2n+2$	$O(n^2)$	二次函数	平方阶
2log2n+2	O(logn)	对数函数	对数阶
2nlog2n+2n+2	O(nlogn)	nlogn函数	nlogn阶
$2n^3+2n+2$	$O(n^2)$	三次函数	立方阶
2^n	$O(2^n)$	指数函数	指数阶
n!	O(n!)	阶乘函数	阶乘阶

时间复杂度



n	logn	√n	nlogn	n^2	2^n	n!
5	2	2	10	25	32	120
10	3	3	30	100	1024	3628800
50	5	7	250	2500	约10^15	约3.0*10^64
100	6	10	600	10000	约10^30	约9.3*10^157
1000	9	31	9000	1000 000	约10^300	约4.0*10^2567

时间复杂度

举例：对数组($x_1,x_2,...,x_n$)进行冒泡排序

最坏情况是数组完全反序，需要进行n-1趟排序，每趟排序要进行n-i次相邻数字的比较($1 \le i \le n-1$)。

$$C_{max} = \frac{n(n-1)}{2} = O(n^2)$$

时间复杂度

一个问题有多种求解算法，不同算法的时间复杂度不同
问题的固有复杂度：所有算法中时间复杂度最低的那个复杂度

以排序方法为例：
冒泡排序：把小的元素往前调或者把大的元素往后调
选择排序：先找最小元素放起始位，从余下数据找最小元素，依次放入序列
插入排序：对未排序数据，在已排序序列中从后向前扫描，找到相应位置并插入
希尔排序：插入排序的一种改进版本，按照不同步长对元素进行插入排序
快速排序：序列分为两个子序列，小于基准的元素放前面，大的放后面，子序列重复该过程
归并排序：将数据拆分为两份子数据分别排序, 再将两份子数据合并
堆排序：　利用完全二叉树堆结构设计，节点值大于等于左右孩子节点值
基数排序：将整数按位数切割成不同的数字，然后按每个位数分别比较

问题：如何选择排序算法？

参考：经典排序算法（含动画演示）
https://zhuanlan.zhihu.com/p/41923298

时间复杂度

排序方式	时间复杂度			空间复杂度	稳定性	复杂性
	平均情况	最坏情况	最好情况			
插入排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	稳定	简单
希尔排序	$O(n^{1.3})$			$O(1)$	不稳定	较复杂
冒泡排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	稳定	简单
快速排序	$O(n\log_2 n)$	$O(n^2)$	$O(n\log_2 n)$	$O(\log_2 n)$	不稳定	较复杂
选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定	简单
堆排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(1)$	不稳定	较复杂
归并排序	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n\log_2 n)$	$O(n)$	稳定	较复杂
基数排序	$O(d(n+r))$	$O(d(n+r))$	$O(d(n+r))$	$O(r)$	稳定	较复杂

排序问题的固有复杂度：$O(n\log n)$

排序算法时间复杂度、空间复杂度、稳定性比较
https://blog.csdn.net/pange1991/article/details/85460755#t23

时间复杂度

当计算机处理数据达到100万个，对于时间复杂度分别为$O(n^2)$ 和$O(2^n)$ 的两个算法，运行时间天壤之别，所以需要研究问题是否具有多项式时间算法。

某个问题是否可在多项式时间复杂度内解决（或验证）？
==>需要了解P问题和NP问题

对于不存在多项式时间复杂度解法的问题，如何区分问题的难度？
==>需要了解NP完全问题和NP难问题

时间复杂度、P、NP、NP完全、NP难也是算法面试中可能考察的问题

目录

P问题和NP问题

P问题：能在**多项式时间内求解**的问题，即存在多项式时间算法的问题。
(P：Polynominal，多项式，如Binominal 二项式，Multinomial 多项式)
Key: Easy to Solve

P问题包括: 复杂度为O(1), O(logn), O(n), O(nlogn), $O(n^2)$, $O(n^k)$的问题
比如：找数组最大值问题为O(n)，冒泡排序问题为$O(n^2)$

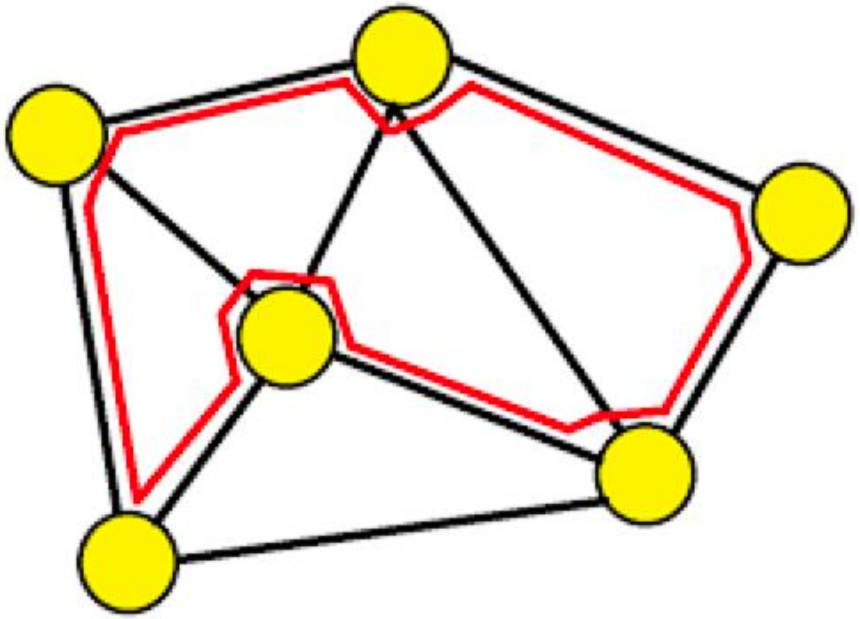
P问题和NP问题

NP问题：能在**多项式时间内验证**给定的某个解是否为正确解的问题。
(NP: Non-deterministic Polynominal，非确定性多项式)
Key: Easy to Verify

如果可以在多项式级的时间内验证一个问题解的正确性，那么我们说该问题是NP问题，直接找NP问题的一个解可能很慢，但验证NP问题的解却很快。

对比：
P问题：**多项式时间内求解**（Easy to Solve）
NP问题：**多项式时间内验证**（Easy to Verify）

从P问题到NP问题



Hamilton回路：
能否找到一条经过每个顶点一次且恰好一次（不遗漏也不重复）最后又走回来的路

寻找汉密尔顿回路

暴力搜索：O(n!)

很难找到多项式复杂度解法(hard to solve)

但是给定一个路径时：

很容易在多项式时间复杂度O(n)内验证它是否是我们想要的回路(but easy to verify)

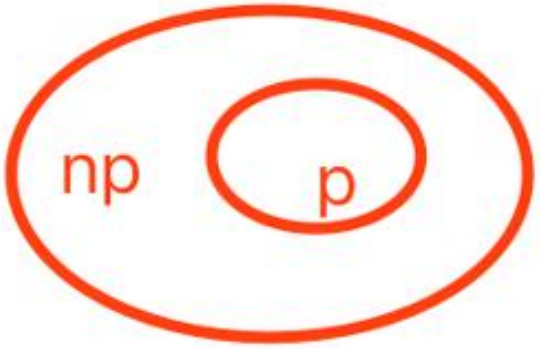
P问题和NP问题

问题：寻找一个数组的中位数，给定某个解：10

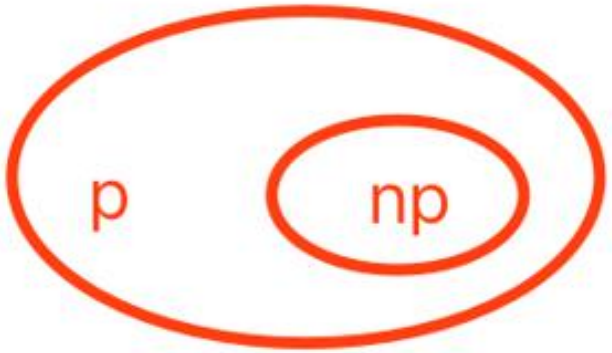
问题求解：先排序[O(nlogn)]，再取中间值[O(1)]，求解复杂度O(nlogn)

问题验证：数组元素逐一和10比对，比较大于10的数和小于10的数个数是否相等，验证复杂度O(n)

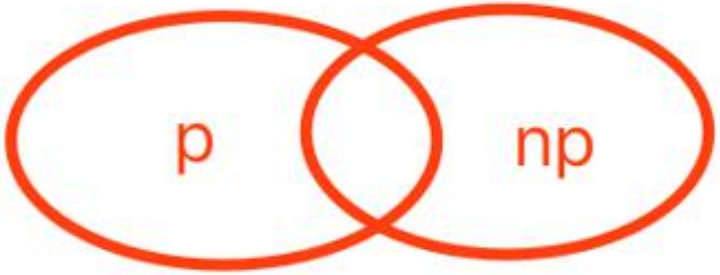
P问题和NP问题的关系



A



B



C

D

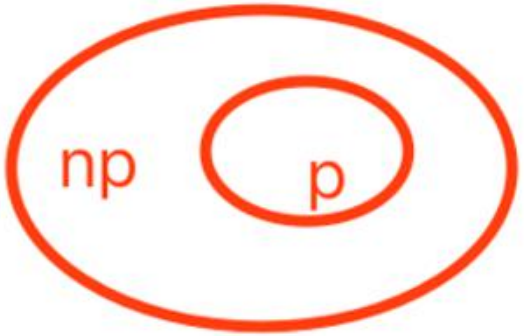
Which One?

P问题和NP问题

关系：P类问题是NP类问题的子集
即存在多项式时间算法的问题，总能在多项式时间内验证它

NP问题不是“非P问题”，而是不确定性多项式（non-deterministic polynomial）问题。NP中的N并非针对P，而是针对deterministic，即不确定性。

NP问题不确定是否存在多项式时间求解算法，但确定存在多项式时间验证算法。



NP ≠ 非P

目录

NP完全问题

在介绍NP完全问题前，先引入规约的概念

规约：假设有两个问题A和B，对问题A的输入a经过某种规则转换为对问题B的输入b，而A(a)和B(b)的结果相同，可将求解A问题转换为求解B问题，或者说可以用解决问题B的方法来解决问题A，那么称A可以归约到B。

规约将一个特殊问题一般化，即将原问题推广为一个更一般的、更有概括性、更难的、计算复杂度可能更高的问题。

归约具有传递性，如果A归约到B，B归约到C，那么A可以归约到C。

NP完全问题

问题A：求解一元一次方程$b_1x+c_1=0$的解

问题B：求解一元二次方程$a_2x^2+b_2x+c_2=0$的解

A的输入：b_1、c_1； B的输入：a_2、b_2、c_2
转换规则：令$a_2=0$, $b_2=b_1$, $c_2=c_1$

求解一元二次方程解的方法可用来求解一元一次方程的解
结论：问题A可归约到问题B

A归约B: 问题A不会难于问题B，即A要归约到更难的问题

NP完全问题

NP完全问题：如果所有NP问题可在多项式时间内归约成某个NP问题，则该NP问题称为NP完全问题(NP Complete)，也记为NPC问题。

NPC问题Q满足以下两条性质：
(1). Q是NP问题
(2). 任一NP问题都可在多项式时间内归约到问题Q

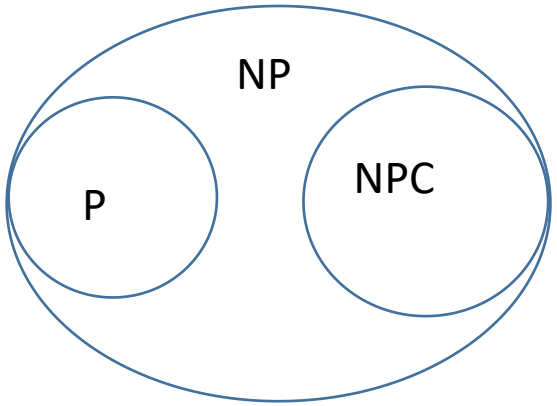
NP完全问题

NPC和NP关系：
NPC问题是NP中最难的问题（Hardest in NP）
NPC问题是暂时不能在多项式时间内求解的NP问题

注意：NPC问题不只一个，它有很多个，它是一类问题
只要解决一个NPC问题，所有的NP问题都能在多项式时间内解决，从而NP=P。

给NPC问题找多项式时间复杂度的算法太难，人们普遍相信P≠NP。
NPC问题目前没有多项式的有效算法，只能用指数级甚至阶乘级复杂度的搜索方法

P、NP和NP完全的关系



P问题： Easy to Solve，多项式时间内求解

NP问题： Easy to Verify，多项式时间内验证

NPC问题： Hardest in NP，所有NP问题均可归约的NP问题

目录

NP Hard问题

NPC问题：如果一个问题Q满足以下两条性质则为NPC问题：
(1). Q是NP问题
(2). 任一NP问题都可在多项式时间内归约到问题Q

NP Hard问题：
满足NPC问题定义的第二条但不一定要满足第一条，即任意NP问题都可以在多项式时间内归约为该问题，但是不限定该问题是NP问题。

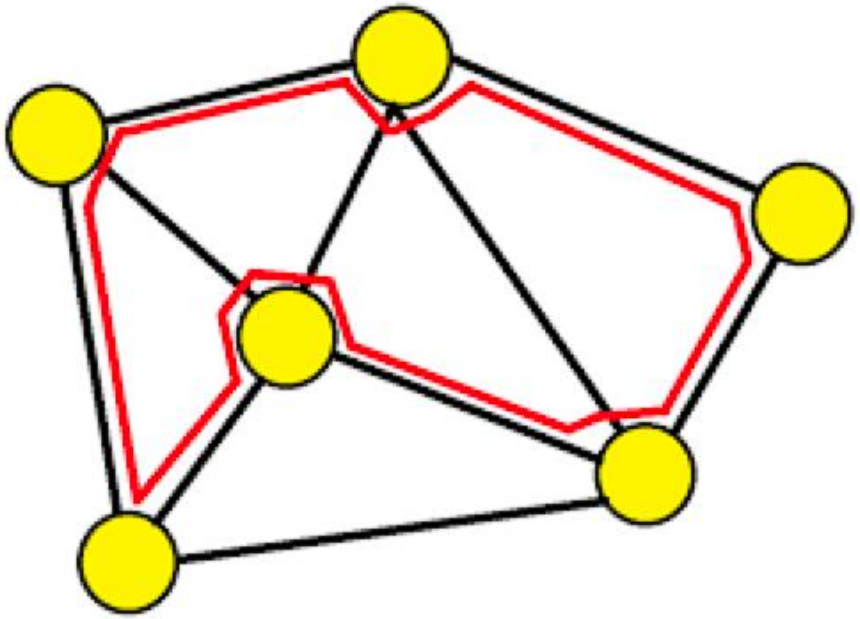
NP Hard问题

虽然NP-Hard问题包括NPC问题，但要比 NPC问题的范围更广。
所有的NP问题都能规约到它，但是它不一定是一个NP问题。

NP-Hard问题难以找到多项式求解算法，但它不列入我们的研究范围，因为它不一定是NP问题，不一定可在多项式时间内验证。

即使NPC问题发现了多项式级的算法，NP-Hard问题有可能仍然无法得到多项式级的算法。由于NP-Hard放宽了限定条件，它将有可能比所有的NPC问题的时间复杂度更高从而更难以解决。

从NP问题到NP Hard问题

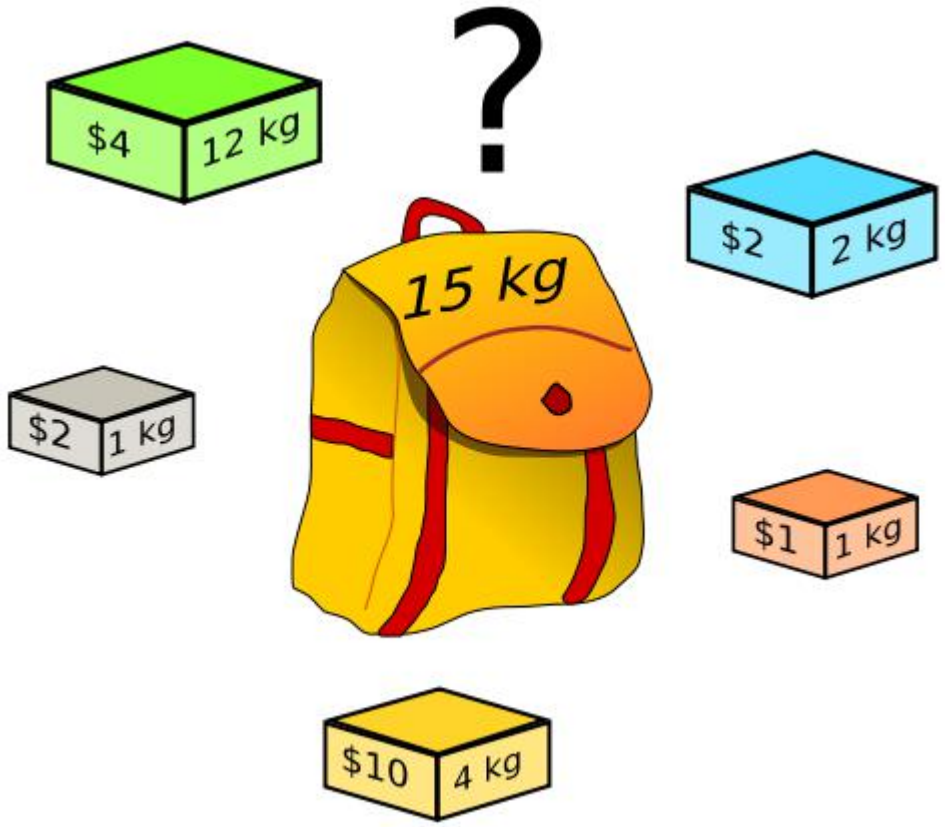


汉密尔顿回路问题：NP问题
验证一条路是否恰好经过了每一个顶点容易

但是把问题换成这样：
试问一个图中是否不存在Hamilton回路？

没法在多项式时间里进行验证，因为除非你试过所有的路，否则你不敢断定没有回路

0/1背包问题



给定一个背包和n件物品，每件物品重量为w_i，价格为v_i，背包最大承重为C，问选择哪些物品放入背包，使得物品总价最高。

$$\max \quad V = \sum_{i=1}^{n} x_i v_i$$

$$\begin{cases} \sum_{i=1}^{n} x_i w_i \le C \\ x_i \in \{0,1\}, 1 \le x_i \le n \end{cases}$$

思考：

上述0-1背包上述问题是NPC问题还是NP Hard问题？

0/1背包判定问题

(1) 证明 0/1背包判定问题是NPC问题

问题描述:

例：给定一个有限集合 X ，对每一个 $x \in X$ ，对应一个值 $w(x) \in Z^+$ ，和相应的值 $p(x) \in Z^+$ 。另外，还有一个容量约束 $M \in Z^+$ 和一个价值目标 $K \in Z^+$ 。

问：是否存在X的一个子集 X' ，使得 $\sum_{x \in X'} w(x) \leq M$ 而且 $\sum_{x \in X'} p(x) \geq K$ 。

证明如下： (限制法)

第一步：给定X的一个子集 X' ，要判断其是否满足 $\sum_{x \in X'} w(x) \leq M$ 和 $\sum_{x \in X'} p(x) \geq K$ ，只需要做 $2 * (|X'| - 1)$ 次加法（其中 $|X'|$ 表示 X' 里元素的个数），时间复杂度为 $O(2 * (|X'| - 1))$ ，即能在多项式时间内完成判断，所以0/1背包判定问题是NP问题。

第二步：考虑特殊的0/1背包问题：对所有的 $x \in X$ 有 $w(x) = p(x)$ ，且取 $M = K = \frac{1}{2} \sum_{x \in X} w(x)$ 。这个0/1背包判定问题回答为“是”，当且仅当集合X的划分问题回答为“是”。可见，划分问题恰是0/1 背包问题的特例。从而由划分问题是NPC问题推得0/1背包判定问题是NPC问题。

证毕。

结论:
“0/1背包判定问题是NPC问题”
“0/1背包问题是NPH问题但不是NPC问题”

https://zhuanlan.zhihu.com/p/102362515

背包问题

(2) 证明0/1背包问题是NPH问题

问题描述：

有n个物品，它们有各自的体积 w_i 和价值 p_i ，现有给定容量M的背包，如何让背包里装入的物品具有最大的价值总和?

（注释：要注意“0/1背包问题”和“0/1背包判定问题”问题描述的区别）

证明如下：

要得到 $max\sum_{i=1}^{n}x_ip_i$ （s.t $\sum_{i=1}^{n}x_iw_i \le M$ ），需比较所有的 $X=(x_1,x_2,\ldots,x_n)$ ，$x_i \in \{0,1\}$ ，这有 2^n 个可能，以比较为基础的求最大问题复杂度下界 $O(n)$ ，故本问题 $O(2^n)$ ，不存在多项式时间算法，所以0/1背包问题不是NP问题。

但0/1背包判定问题是NPC问题，0/1背包问题不比其更容易，所以，它是NPH问题。

背包问题

分析：

对于n个物品，每个物品可能会选，也可能不选，总共有2^n种组合选择方式，对于每种选择方式，计算总价格需要n次乘加操作，复杂度为$O(n2^n)$。

思考：

0-1背包问题是NP难问题，是不是意味着没办法求解，应该如何高效地求解呢？

背包问题的动态规划求解

$$\max \quad V=\sum_{i=1}^{n} x_i v_i$$
$$\begin{cases} \sum_{i=1}^{n} x_i w_i \le C \\ x_i \in \{0,1\}, 1 \le x_i \le n \end{cases}$$

动态规划求解背包问题

设V(i, j)表示给定前 i个物品和容量为j的背包，可得的最大价值

假如已经求得V(i-1, 0:j)，现在求V(i, j)，存在两种选择：
（1）不放第i个物品，总价值为 V(i−1 , j)
（2）放入第i个物品，总价值为 v_i + V(i−1, j−w_i)

两种选择取最大值，可得递推关系式：
V(i, j) = max { V(i−1, j), v_i + V(i−1, j−w_i) }

利用上述递推关系求V(n,C)

背包问题

迭代：$V(i, j) = \max \{ V(i-1, j),\ v_i + V(i-1,\ j-w_i) \}$ $i<=n,\ j<=C$

背包问题可通过动态规划以O(nC)复杂度解决
O(nC)取决于物品数量n和背包最大承重C两个规模参数

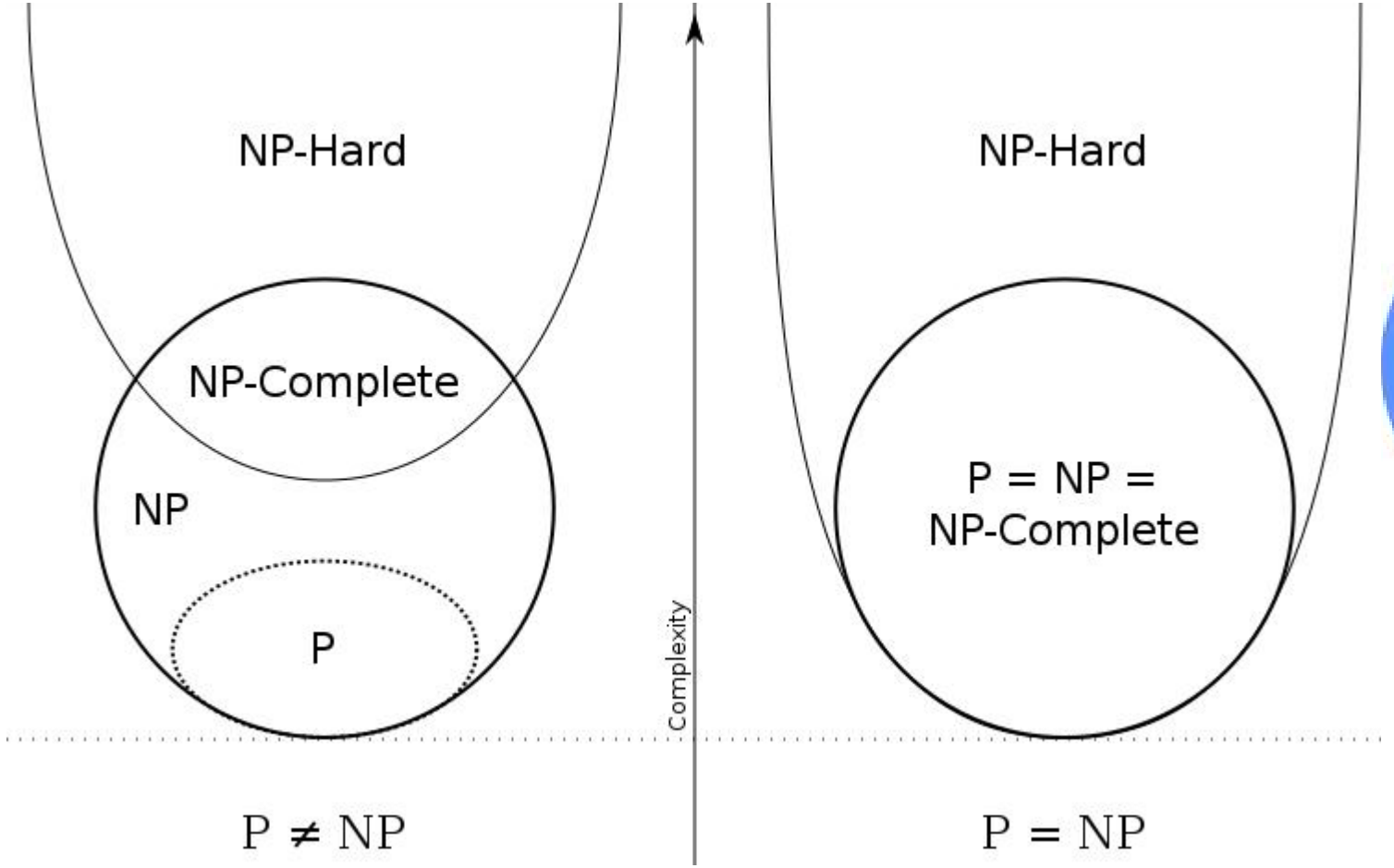
如果C较小，则O(nC)约等于O(n)，看起来像多项式时间，可称为伪多项式
如果$C>2^n$，$O(nC)=O(n2^n)$

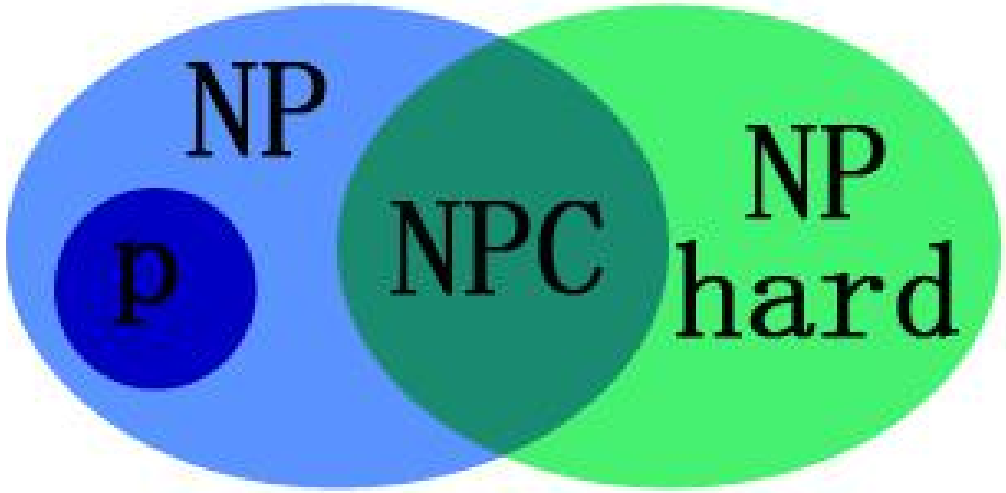
从最坏情况来看，动态规划的复杂度是$O(n2^n)$

结论：
背包问题虽然可通过动态规划求解，没有改变其属于NP难问题的本质

目录

P/NP/NPC/NPH对比



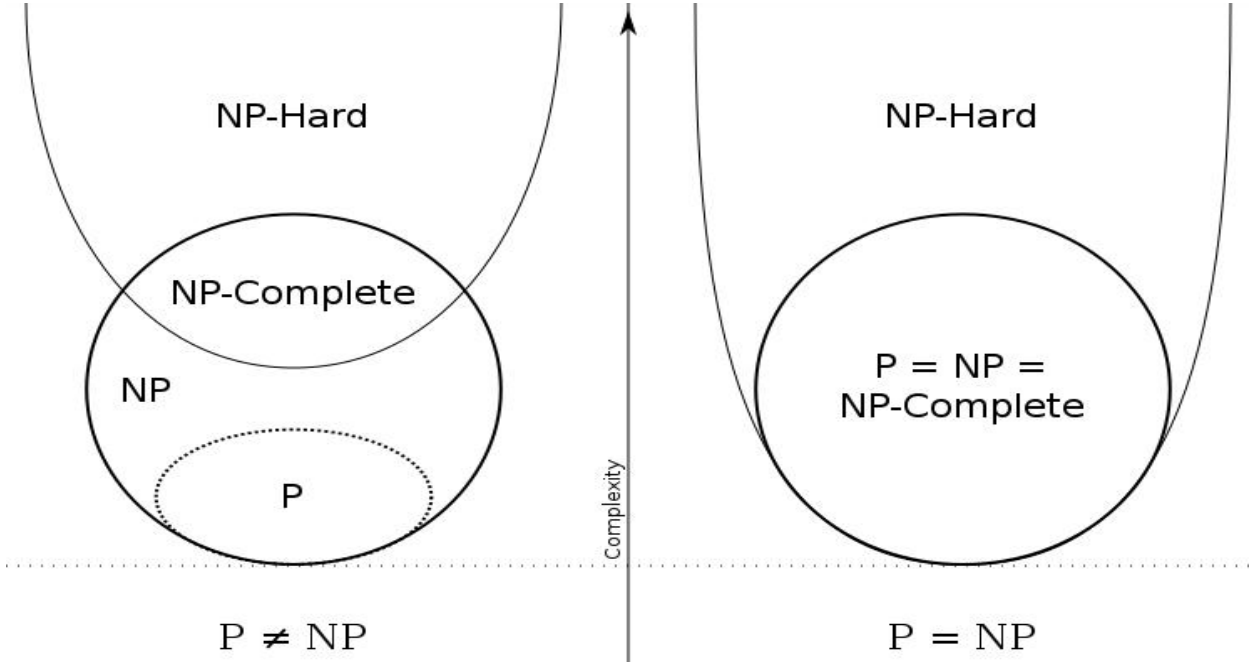


千禧年问题 P = NP ?　　证明或推翻P=NP

- 假设 NP=P 猜想不成立，那么计算复杂度的相对关系为（按照由低到高）：

P < NP < NPC < NP Hard

- 假设 NP=P 猜想成立，那么说明所有存在多项式时间验证算法的问题都存在多项式时间求解算法，而NPC本身属于NP问题，因此NPC也存在多项式时间求解算法，所以 P=NP=NPC，属于NP hard问题的一个子集。



千禧年问题 P = NP ?

该猜想尚未被证明（成立），也尚未被证伪（不成立）。

该问题一直都是信息学的巅峰，是耗费了很多时间和精力也没有解决的终极问题，好比物理学中的大统一和数学中的哥德巴赫猜想等。

但是，一个总的趋势、一个大方向是有的。人们普遍认为P=NP不成立，多数人相信，存在至少一个不可能有多项式级复杂度算法的NP问题。

人们在研究NP问题的过程中，找出一类非常特殊的NP问题叫做NP-完全问题，正是NPC问题的存在，使人们相信P≠NP。

NPC问题目前没有多项式的有效算法，只能用指数级甚至阶乘级复杂度的搜索。

小结

P问题： Easy to Solve，能在多项式时间内求解的问题。

NP问题： Easy to Verify，能在多项式时间内验证某个解是否为正确解的问题。

NPC问题： Hardest in NP 如果一个问题Q满足以下两条性质则为NPC问题：
(1). Q是NP问题；(2). 任一NP问题都可在多项式时间内归约到问题Q

NP Hard问题： Harder than NP 满足NPC问题定义的第二条但不一定要满足第一条，即任意NP问题都可以在多项式时间内归约为该问题，但是不限定该问题是NP问题。

参考资料

怎么理解 P 问题和 NP 问题？
https://www.zhihu.com/question/27039635

什么是P问题、NP问题和NPC问题？
http://www.matrix67.com/blog/archives/105

背包判定问题和背包问题
https://zhuanlan.zhihu.com/p/102362515

经典排序算法（含动画演示）
https://zhuanlan.zhihu.com/p/41923298

排序算法时间复杂度、空间复杂度、稳定性比较
https://blog.csdn.net/pange1991/article/details/85460755#t23

动态规划-0/1背包问题
https://yngzmiao.blog.csdn.net/article/details/81667885?utm_medium=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromBaidu-1.not_use_machine_learn_pai&depth_1-utm_source=distribute.pc_relevant_t0.none-task-blog-BlogCommendFromBaidu-1.not_use_machine_learn_pai

贪心科技 GREEDY TECHNOLOGY

贪心科技　让每个人享受个性化教育服务

THANKS

贪心学院讲师：占老师