

NP完全性理论

NP完全性理论

- 
- **P** 问题
 - **NP** 问题
 - **NP完全** 问题 NPC(complete)
 - **NP难** 问题 NP-hard

NP完全性理论

多项式与非多项式时间复杂性

$2n, 2^n, 3n^2+4n, 5n+n^{10}, 2^{0.01n}, n\log n, n!$

- $2n, 3n^2+4n, 5n+n^{10}, n\log n$
- $2^{0.01n}, 2^n, n!$

多项式时间算法一定更快吗?

NP完全性理论

- 易解的：可在多项式时间内求解
- 难解的：不能在多项式时间内求解

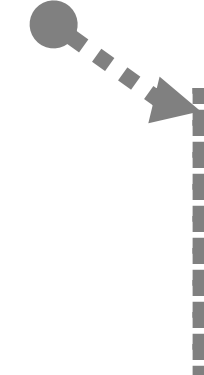
TSP问题

20个城市：	蛮力：	100年以上
	动态规划：	45秒
40个城市：	动态规划：	6.7年

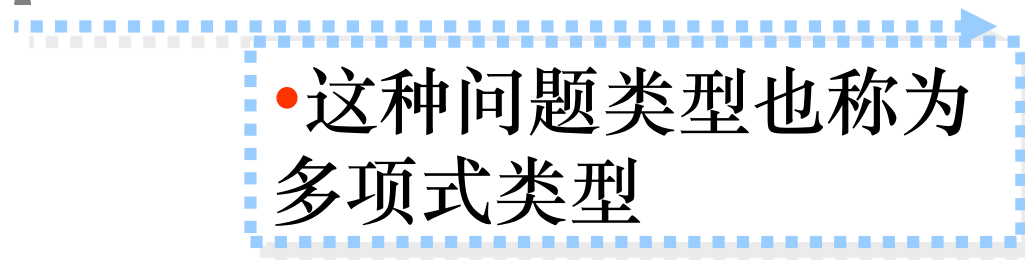
若解决在世界范围内铺设光缆呢？

NP完全性理论

P问题



- 是一类能够用(确定的)算法在多项式时间内求解的可判定问题



- 这种问题类型也称为多项式类型

NP完全性理论

停机问题

- 阿兰·图灵1936年提出的著名问题：
给定一段计算机程序和它的一个输入，
判定该程序对于该输入是会终止，还是
会无限运行下去。

- 这是个不可判定问题

NP完全性理论

图灵停机问题的悖论


```
bool God_algo(char * program, char * input)
{  if(<program> halts on input)
    return true;
    else return false;
}
```

类似有人站在门口，让你猜他
是要进门，还是要出门？ 你
可能永远猜不对！


```
bool Satan_algo(char * program)
{
    if(God_algo(program, program))
    {
        while(true); // loop forever!
        return false; // can never get here!
    }
    else
        return true;
}
```

NP完全性理论

NP问题



- 是一类能够用不确定算法在多项式时间内求解的可判定问题



- 在确定性计算模型下多项式时间可验证的可判定问题

故: $P \subseteq NP$

NP完全性理论

NPC问题(NP完全问题)



一个可判定问题**D**，满足：

1. 其属于**NP**问题
2. **NP**中的任何问题都能够多项式时间内归约为**D**

则**D**为**NP**完全问题

NP完全性理论

NPC问题的意义：

有一个NPC问题找到多项式时间的解法，
则全部解决

- 不要浪费时间去寻找有效算法
- 找近似算法或特例算法
- 注意一些看似简单的问题其实也是**NPC**.

NP完全性理论

NP完全与NP难的区别

NP完全定义：问题L是NP完全的当且仅当

(1) $L \in \text{NP}$;

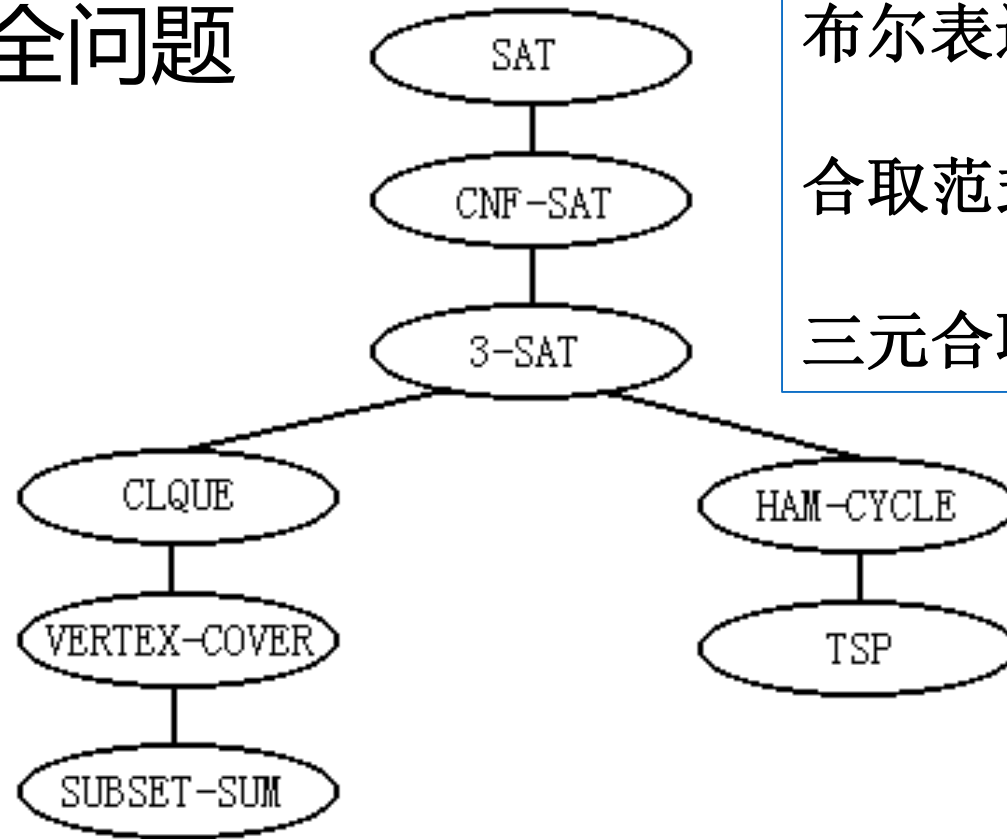
(2) 对于所有 $L' \in \text{NP}$ 有 $L' \leq_p L$ 。

NP难定义：问题L满足上述性质(2)，但不一定满足性质(1)。

根据定义可得： $\text{NPC} = \text{NP} \cap \text{NP难}$

NP完全性理论

一些典型的NP完全问题



布尔表达式的可满足性问题

合取范式的可满足性问题

三元合取范式的可满足性问题

部分NP完全问题树

迄今为止，发现1000多个npc问题

NP完全性理论



P, NP, NPC、NP难问题之间的关系

- $P \subseteq NP$ (Sure)
- $NPC \subseteq NP$ (sure)
- $P = NP$ (or $P \subset NP$, or $P \neq NP$) ???
- $NPC = NP$ (or $NPC \subset NP$, or $NPC \neq NP$) ???
- NP难问题，不确定是NP问题。

121. Best Time to Buy and Sell Stock

Easy

👍 15618

💬 526

❤ Add to List

📄 Share

You are given an array `prices` where `prices[i]` is the price of a given stock on the i^{th} day.

You want to maximize your profit by choosing a **single day** to buy one stock and choosing a **different day in the future** to sell that stock.

Return *the maximum profit you can achieve from this transaction*. If you cannot achieve any profit, return `0`.

Example 1:

Input: `prices = [7,1,5,3,6,4]`

Output: 5

Explanation: Buy on day 2 (price = 1) and sell on day 5 (price = 6), profit = 6-1 = 5.

Note that buying on day 2 and selling on day 1 is not allowed because you must buy before you sell.

算法训练

习题：给定一个字符串s，如果该串最多删除一个字符就是一个回文， 返回1， 否则返回0

Input: aba

Output: 1

Input: abca

Output: 1

Input: abc

Output: 0