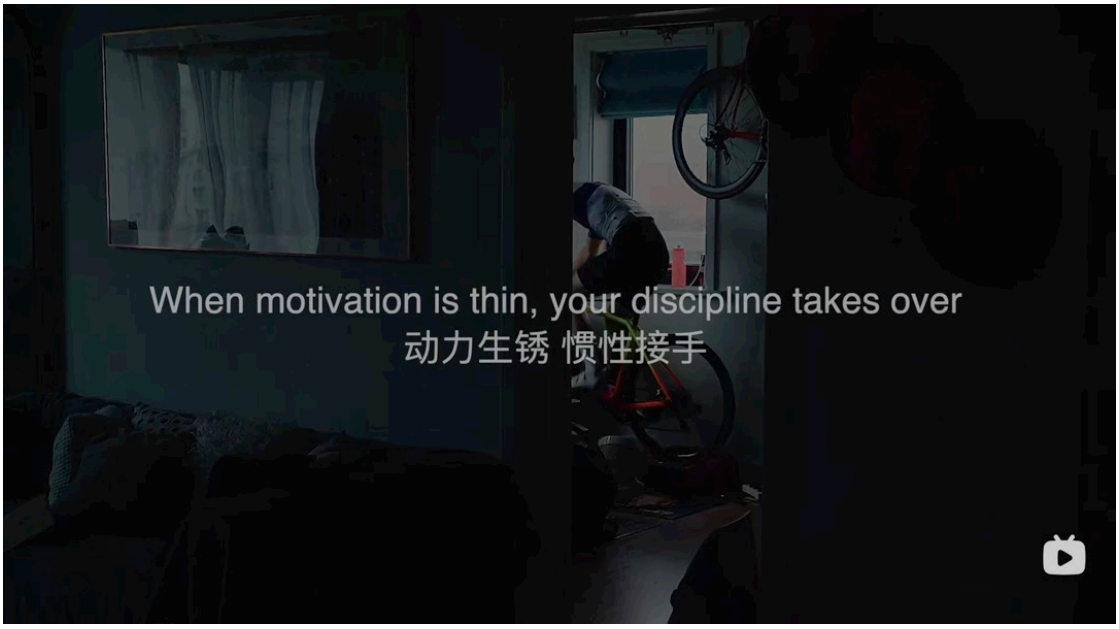
CS211FZ Note 6 | Algorithms & Data Structures | DSA2

The words from the bottom of Lance's heart

首先，感谢并恭喜你看到了「CS211FZ Note 6」，这是CS211笔记的最后一个部分。谢谢你信任我的笔记，并认真复习到了这里。下面的引言和视频是我这几年来最喜欢的，作为礼物送给你，希望对你有所帮助。最后，祝你考试好运，顺利取得理想的成绩。大二学年马上也要结束了，也祝你在这个夏天一切顺利。

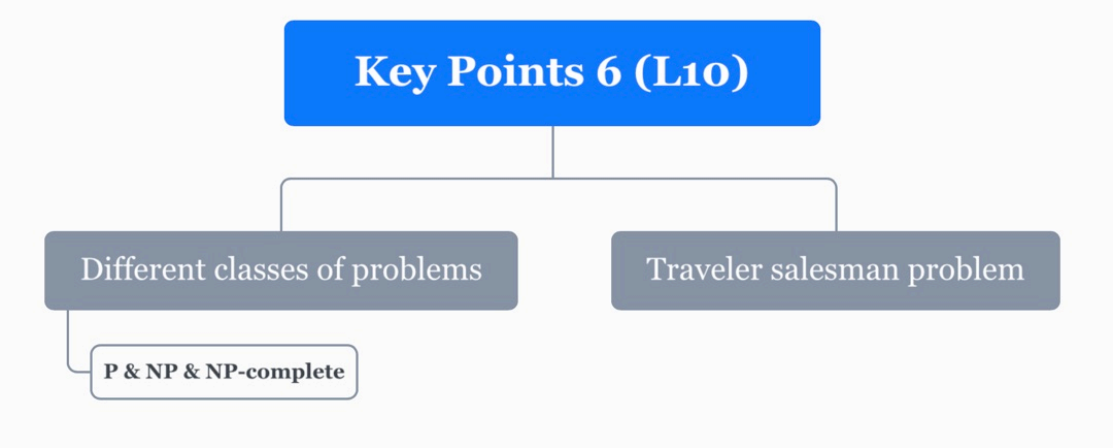
引言：When motivation is thin, your despline takes over



🔗视频链接：`https://www.bilibili.com/video/BV1aR4y1J7sB?spm_id_from=333.999.0.0`

Key Points 6 (L10)

- Different classes of problems
    - **P & NP & NP-complete**
- Traveler salesman problem



10.1 P & NP & NP-complete

P :如果一个问题可以找到一个能在多项式的时间里解决该问题的算法，这个问题就属于P问题;

NP:可以在多项式的时间里验证一个解的问题。

所有的P问题都是NP问题，即能在多项式时间内解决一个问题，必然能在多项式时间里验证一个问题的解。

NPC问题：首先必须是一个NP问题，然后所有的NP问题都可以约化到该问题。

约化：问题A可以约化为问题B，即可以用问题B的解法解决问题A，解决问题B的时间复杂度大于等于解决A的时间复杂度。约化具有传递性，问题A可以约化为问题B，问题B可以约化为问题C，则问题A一定可以约化为问题C。

约化的过程需要在多项式时间内完成。目前被证明是NPC问题的有很多，任何一个找到了多项式时间复杂度的算法，所有的NP问题都可以被完美解决。

10.1.1 P 问题

P = polynomial time

> 《算法导论》给出的定义：在多项式时间内可解的问题为 P 问题（Polynomial Problem，多项式问题）

2.P 问题

《算法导论》给出的定义：在多项式时间内可解的问题为 P 问题（Polynomial Problem，多项式问题）。

更为具体的是：P 问题指可以在多项式时间内求解的问题，例如：时间复杂度为 O(nlog(n)) 的快速排序和堆排序，$O(n^2)$ 的冒泡排序和直接选择排序算法都是P问题，也就是多项式时间算法。相反，时间复杂度为 $O(n^{logn})$、$O(2^n)$ 的算法是指数时间算法。

- Problems are assigned to classes depending on the complexity of the algorithm required to solve them.
- A problem is assigned to the P (polynomial time) class if at least one algorithm exists to solve that problem with a complexity of $O(n^x)$, where x is some number.
  - $n^2 + 3n$
  - $n^5 + n^3 + \log n$
  - $n^{10000} + 16$
- These are considered 'easy' problems to solve.

10.1.2 NP 问题

NP = Non-Deterministic polynomial time.

> **定义：**NP问题（(Non-deterministic Polynomial Problem，非确定性多项式问题），指问题只能通过验证给定的猜测是否正确来求解。
>
> 1、所谓**多项式**指的是验证猜测可在多项式时间内完成；
>
> 2、所谓**非确定性**指的是问题只能通过验证猜测来解，而不能直接求解。

NP-problems

NP is not the same as non-polynomial complexity/running time. NP does not stand for not polynomial.

- NP = Non-Deterministic polynomial time.
- NP means verifiable in polynomial time.

Verifiable?

- If we are somehow given a 'certificate' of a solution we can verify the legitimacy in polynomial time.

A problem is assigned to the complexity class NP if a non-deterministic polynomial machine can compute a solution, and the solution can be recognized/verified in P time by an ordinary Turing machine.

如**Hamilton回路**是NP问题，因为验证一条路是否恰好经过了每一个顶点可在多项式时间内完成，但是找出一个Hamilton回路却要穷举所有可能性，不能直接求解。

又如**大合数的质因数分解**，没有给定的公式可直接求出一个合数的两个质因数是什么，但是验证两个数是否是质因数却可在多项式时间完成，所以它也是非确定性多项式问题，即NP问题。

之所以要定义NP问题，是因为通常只有NP问题才可能找到多项式的算法。我们不会指望一个连多项式地验证一个解都不行的问题存在一个解决它的多项式级的算法。

简单的说，**存在多项式时间的算法的一类问题，称之为P类问题；而像梵塔问题，推销员旅行问题等问题。至今没有找到多项式时间算法解的一类问题，称之为NP问题。**

同时，P类问题是NP问题的一个子集。也就是说，能多项式时间地解决一个问题，必然能多项式时间地验证一个问题的解。

3.1 NP 与 P 的关系

目前，**人类还未解决的问题是**：是否所有的NP问题都是P类问题，即P=NP?。这就是注明的世界七大数学难题之首。虽然这个问题尚未解决，但是，一个总的趋势和大方向是人们普遍认为，P=NP不成立，也就是说，多数人相信，存在至少一个不可能有多项式级复杂度的算法的NP问题。

人们如此坚信P≠NP是有原因的，就是在研究NP问题的过程中找出了一类非常特殊的NP问题叫做NP-完全问题（Non-deterministic Polynomial Complete Problem），也即所谓的 NPC问题。正是NPC问题的存在，使人们相信P≠NP。

10.1.3 NPH问题（NP-Hard）

NPH问题（NP难问题，英文NP-hard问题），其满足NPC问题定义的第二条但不一定要满足第一条（就是说，NP-Hard问题要比 NPC问题的范围广，但不一定是NP问题）。

所有的NP/NPC问题都能在多项式时间复杂度内约化到的问题，但该问题不一定是NP问题。

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。事实上，*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。

4.最具代表性的NP-Hard问题：TSP

售货员旅行问题 (traveling salesman problem)，是最具有代表性的NP问题之一。假设一个推销员需要从香港出发，经过广州，北京，上海，…，等 n 个城市， 最后返回香港。 任意两个城市之间都有飞机直达，但票价不等。现在假设公司只给报销 C 块钱，问是否存在一个行程安排，使得他能遍历所有城市，而且总的路费小于 C?
推销员旅行问题显然是 NP 的。因为如果你任意给出一个行程安排，可以很容易算出旅行总开销。但是，要想知道一条总路费小于 C 的行程是否存在，在最坏情况下，必须检查所有可能的旅行安排! 这将是个天文数字。
这个天文数字到底有多大？目前的方法接近一个一个的排着试，还没有找到更好可以寻得最短路径的方法。对七个城而言，共有 6!=720 个排法，还比较简单；，但若有 20 个城，则排法就有 19! 种。因故在排列组合里 n! 写起来轻松。但 $1.21 * 10^{17}$ 是一个大得不得了的数字。若每秒钟排一次，要排 $3.84 * 10^{9}$ 年（一年约为 $3.15 * 10^{7}$ 秒），即使使用计算器，每秒排一百万次（不容易做到）也得重做三千年才能找到答案。「生也有涯，知也无涯」，想不到区区二十个城，要三十个世纪才能找到答案。

说到这里为止，童鞋们应该对NP-Hard有个大致的了解了吧！

10.1.4 NPC问题（NP-Complete）

1. Hamilton回路
2. Traveling salesman problem

4.2 NPC 问题

NPC 问题是指满足下面两个条件的问题：
（1）它是一个 NP 问题；
（2）所有的 NP 问题都可以用多项式时间约化到它。

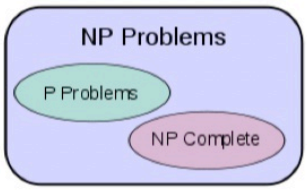
所以显然NP完全问题具有如下性质：它可以在多项式时间内求解，当且仅当所有的其他的NP完全问题也可以在多项式时间内求解。这样一来,只要我们找到一个NPC问题的多项式解，所有的NP问题都可以多项式时间内约化成这个NPC问题，再用多项式时间解决，这样NP就等于P了。

目前，NPC 问题还没有找到一个多项式时间算法，因此我们就此可直观地理解，NPC 问题目前没有多项式时间复杂度的有效算法，只能用指数级甚至阶乘级复杂度的搜索。

大多数人的观点对于 P类问题，NP类问题，NPC之间的关系可用下图表示（此图正确性有待验证）：

NP-complete problems

- NP-complete problems are the hardest possible NP problems.
- If any NP-complete problem can ever be solved by a polynomial algorithm, then P must equal NP because it would mean that even the hardest NP problems would be easy to solve
- All NP-complete problems are really the same problem – they can be reduced to each other in polynomial time.
- Nobody has ever found a polynomial-time algorithm to solve any of them.

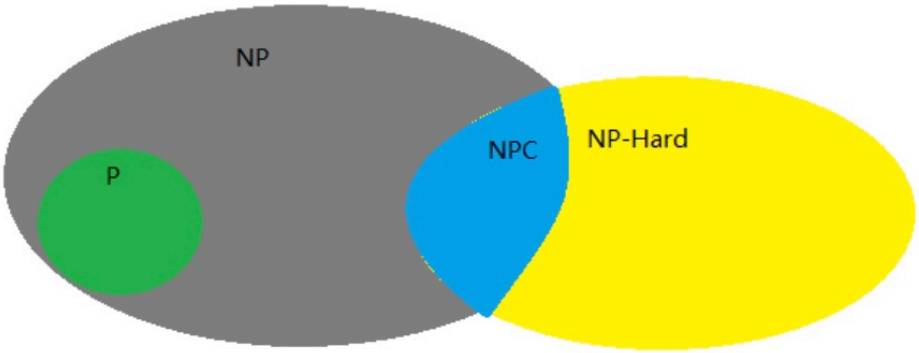


Ex.

NP-complete example (subset-sum)

- The subset-sum problem has been proven to be NP-complete.
- Does any subset of these numbers sum to 53?
  - (15, 8, 14, 26, 32, 16, 9, 22)
- Solutions:
  - 15 + 22 + 16
  - 22 + 14 + 9 + 8
  - These solutions are hard to find yet easy to verify!

10.1 Overall



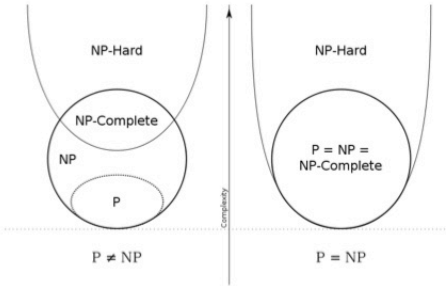
10.2 Traveler salesman problem

Traveling Salesman Problem

- The Traveling Salesman Problem (TSP) is another famous NP-complete problem that Hamilton defined.
- You're given a list of cities on a map, and you want to visit each one covering the least distance possible.
- Unfortunately, the list of possible routes is very large – it's the factorial of the number of towns.
  - 6! = 6 x 5 x 4 x 3 x 2 x 1 = 720
- The problem becomes impractical to solve for even 40 cities.

TSP decision is NP-complete

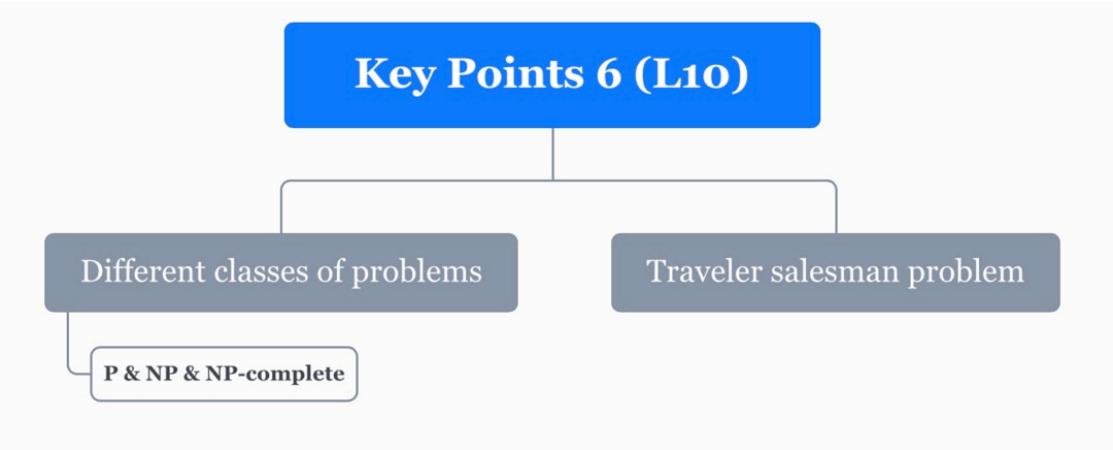
- Say somebody makes a claim – *"there is a TSP path for this problem that is under 1600km".*
- It seems difficult to verify if this is correct or not.
- However, if they show you the path, it is easy to verify if they are telling the truth.
- This is why the TSP decision problem is NP-Complete.
- The TSP optimization problem is NP-Hard, which means it is at least as hard to solve as the hardest NP problems. Still, it may not be in NP (solution is as hard as possible to compute but not necessarily easy to recognize).
- If any NP-Hard problem can be solved in P time, then so can all NP, and thus P=NP.
- For $1,000,000, can you find a quick solution for finding paths?



DFS with Branch-and-Bound

- Branch-and-bound is a heuristic we can use to speed up the depth-first search by not bothering to consider paths that couldn't possibly end up being better than the best solution found so far.
- If a partial tour is already longer than the best solution found so far that there is no need to continue searching down that path

CS211 Note 6 Review



CS211 Note-6

by Lance Cai

2022/07/06