Matrix67: The Aha Moments

这是一篇旧文，点击此处以旧主题模式浏览。

什么是P问题、NP问题和NPC问题

这或许是众多Oier最大的误区之一。

你会经常看到网上出现“这怎么做，这不是NP问题吗”、“这个只有搜了，这已经被证明是NP问题了”之类的话。你要知道，大多数人此时所说的NP问题其实都是指的NPC问题。他们没有搞清楚NP问题和NPC问题的概念。NP问题并不是那种“只有搜才行”的问题，NPC问题才是。好，行了，基本上这个误解已经被澄清了。下面的内容都是在讲什么是P问题，什么是NP问题，什么是NPC问题，你如果不是很感兴趣就可以不看了。接下来你可以看到，把NP问题当成是 NPC问题是一个多大的错误。

还是先用几句话简单说明一下时间复杂度。时间复杂度并不是表示一个程序解决问题需要花多少时间，而是当问题规模扩大后，程序需要的时间长度增长得有多快。也就是说，对于高速处理数据的计算机来说，处理某一个特定数据的效率不能衡量一个程序的好坏，而应该看当这个数据的规模变大到数百倍后，程序运行时间是否还是一样，或者也跟着慢了数百倍，或者变慢了数万倍。不管数据有多大，程序处理花的时间始终是那么多的，我们就说这个程序很好，具有O(1)的时间复杂度，也称常数级复杂度；数据规模变得有多大，花的时间也跟着变得有多长，这个程序的时间复杂度就是O(n)，比如找n个数中的最大值；而像冒泡排序、插入排序等，数据扩大2倍，时间变慢4倍的，属于O(n^2)的复杂度。还有一些穷举类的算法，所需时间长度成几何阶数上涨，这就是O(a^n)的指数级复杂度，甚至O(n!)的阶乘级复杂度。不会存在O(2*n^2)的复杂度，因为前面的那个“2”是系数，根本不会影响到整个程序的时间增长。同样地，O (n^3+n^2)的复杂度也就是O(n^3)的复杂度。因此，我们会说，一个O(0.01*n^3)的程序的效率比O(100*n^2)的效率低，尽管在n很小的时候，前者优于后者，但后者时间随数据规模增长得慢，最终O(n^3)的复杂度将远远超过O(n^2)。我们也说，O(n^100)的复杂度小于O(1.01^n)的复杂度。

容易看出，前面的几类复杂度被分为两种级别，其中后者的复杂度无论如何都远远大于前者：一种是O(1),O(log(n)),O(n^a)等，我们把它叫做多项式级的复杂度，因为它的规模n出现在底数的位置；另一种是O(a^n)和O(n!)型复杂度，它是非多项式级的，其复杂度计算机往往不能承受。当我们在解决一个问题时，我们选择的算法通常都需要是多项式级的复杂度，非多项式级的复杂度需要的时间太多，往往会超时，除非是数据规模非常小。

自然地，人们会想到一个问题：会不会所有的问题都可以找到复杂度为多项式级的算法呢？很遗憾，答案是否定的。有些问题甚至根本不可能找到一个正确的算法来，这称之为“不可解问题”(Undecidable Decision Problem)。The Halting Problem就是一个著名的不可解问题，在我的Blog上有过专门的介绍和证明。再比如，输出从1到n这n个数的全排列。不管你用什么方

法，你的复杂度都是阶乘级，因为你总得用阶乘级的时间打印出结果来。有人说，这样的“问题”不是一个“正规”的问题，正规的问题是让程序解决一个问题，输出一个“YES”或“NO”（这被称为判定性问题），或者一个什么什么的最优值（这被称为最优化问题）。那么，根据这个定义，我也能举出一个不大可能会有多项式级算法的问题来：Hamilton回路。问题是这样的：给你一个图，问你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（满足这个条件的路径叫做Hamilton回路）。这个问题现在还没有找到多项式级的算法。事实上，这个问题就是我们后面要说的NPC问题。

下面引入P类问题的概念：如果一个问题可以找到一个能在多项式的时间里解决它的算法，那么这个问题就属于P问题。P是英文单词多项式的第一个字母。哪些问题是P类问题呢？通常NO和NOIP不会出不属于P类问题的题目。我们常见到的一些信息奥赛的题目都是P问题。道理很简单，一个用穷举换来的非多项式级时间的超时程序不会涵盖任何有价值的算法。

接下来引入NP问题的概念。这个就有点难理解了，或者说容易理解错误。在这里强调（回到我竭力想澄清的误区上），NP问题不是非P类问题。NP问题是指可以在多项式的时间里验证一个解的问题。NP问题的另一个定义是，可以在多项式的时间里猜出一个解的问题。比方说，我RP很好，在程序中需要枚举时，我可以一猜一个准。现在某人拿到了一个求最短路径的问题，问从起点到终点是否有一条小于100个单位长度的路线。它根据数据画好了图，但怎么也算不出来，于是来问我：你看怎么选条路走得最少？我说，我RP很好，肯定能随便给你指条很短的路出来。然后我就胡乱画了几条线，说就这条吧。那人按我指的这条把权值加起来一看，嘿，神了，路径长度98，比100小。于是答案出来了，存在比100小的路径。别人会问他这题怎么做出来的，他就可以说，因为我找到了一个比100 小的解。在这个题中，找一个解很困难，但验证一个解很容易。验证一个解只需要O(n)的时间复杂度，也就是说我可以花O(n)的时间把我猜的路径的长度加出来。那么，只要我RP好，猜得准，我一定能在多项式的时间里解决这个问题。我猜到的方案总是最优的，不满足题意的方案也不会来骗我去选它。这就是NP问题。当然有不是NP问题的问题，即你猜到了解但是没用，因为你不能在多项式的时间里去验证它。下面我要举的例子是一个经典的例子，它指出了一个目前还没有办法在多项式的时间里验证一个解的问题。很显然，前面所说的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非常容易。但我要把问题换成这样：试问一个图中是否不存在Hamilton回路。这样问题就没法在多项式的时间里进行验证了，因为除非你试过所有的路，否则你不敢断定它“没有Hamilton回路”。

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。相信读者很快明白，信息学中的号称最困难的问题——“NP问题”，实际上是在探讨NP问题与P类问题的关系。

很显然，所有的P类问题都是NP问题。也就是说，能多项式地解决一个问题，必然能多项式地验证一个问题的解——既然正解都出来了，验证任意给定的解也只需要比较一下就可以了。关键是，人们想知道，是否所有的NP问题都是P类问题。我们可以再用集合的观点来说明。如果把所有P类问题归为一个集合P中，把所有 NP问题划进另一个集合NP中，那么，显然有P属于NP。现在，所有对NP问题的研究都集中在一个问题上，即究竟是否有P=NP？通常所谓的“NP问题”，其实就一句话：证明或推翻P=NP。

NP问题一直都是信息学的巅峰。巅峰，意即很引人注目但难以解决。在信息学研究中，这

是一个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歌德巴赫猜想等。

目前为止这个问题还“啃不动”。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，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，也即所谓的 NPC问题。C是英文单词“完全”的第一个字母。正是NPC问题的存在，使人们相信P≠NP。下文将花大量篇幅介绍NPC问题，你从中可以体会到NPC问题使P=NP变得多么不可思议。

为了说明NPC问题，我们先引入一个概念——约化(Reducibility，有的资料上叫“归约”)。

简单地说，一个问题A可以约化为问题B的含义即是，可以用问题B的解法解决问题A，或者说，问题A可以“变成”问题B。《算法导论》上举了这么一个例子。比如说，现在有两个问题：求解一个一元一次方程和求解一个一元二次方程。那么我们说，前者可以约化为后者，意即知道如何解一个一元二次方程那么一定能解出一元一次方程。我们可以写出两个程序分别对应两个问题，那么我们能找到一个“规则”，按照这个规则把解一元一次方程程序的输入数据变一下，用在解一元二次方程的程序上，两个程序总能得到一样的结果。这个规则即是：两个方程的对应项系数不变，一元二次方程的二次项系数为0。按照这个规则把前一个问题转换成后一个问题，两个问题就等价了。同样地，我们可以说，Hamilton回路可以约化为TSP问题(Travelling Salesman Problem，旅行商问题)：在Hamilton回路问题中，两点相连即这两点距离为0，两点不直接相连则令其距离为1，于是问题转化为在TSP问题中，是否存在一条长为0的路径。Hamilton回路存在当且仅当TSP问题中存在长为0的回路。

“问题A可约化为问题B”有一个重要的直观意义：B的时间复杂度高于或者等于A的时间复杂度。也就是说，问题A不比问题B难。这很容易理解。既然问题A能用问题B来解决，倘若B的时间复杂度比A的时间复杂度还低了，那A的算法就可以改进为B的算法，两者的时间复杂度还是相同。正如解一元二次方程比解一元一次方程难，因为解决前者的方法可以用来解决后者。

很显然，约化具有一项重要的性质：约化具有传递性。如果问题A可约化为问题B，问题B可约化为问题C，则问题A一定可约化为问题C。这个道理非常简单，就不必阐述了。

现在再来说一下约化的标准概念就不难理解了：如果能找到这样一个变化法则，对任意一个程序A的输入，都能按这个法则变换成程序B的输入，使两程序的输出相同，那么我们说，问题A可约化为问题B。

当然，我们所说的“可约化”是指的可“多项式地”约化(Polynomial-time Reducible)，即变换输入的方法是能在多项式的时间里完成的。约化的过程只有用多项式的时间完成才有意义。

好了，从约化的定义中我们看到，一个问题约化为另一个问题，时间复杂度增加了，问题的应用范围也增大了。通过对某些问题的不断约化，我们能够不断寻找复杂度更高，但应用范围更广的算法来代替复杂度虽然低，但只能用于很小的一类问题的算法。再回想前面讲的P和NP问题，联想起约化的传递性，自然地，我们会想问，如果不断地约化上去，不断找到能“通吃”若干小NP问题的一个稍复杂的大NP问题，那么最后是否有可能找到一个时间复杂度最高，并且能“通吃”所有的 NP问题的这样一个超级NP问题？答案居然是肯定的。也就是说，存在这样一个NP问题，所有的NP问题都可以约化成它。换句话说，只要解决了这个问题，那么所有的NP问题都解决了。这种问题的存在难以置信，并且更加不可思议的是，这种问题不只一

个，它有很多个，它是一类问题。这一类问题就是传说中的NPC 问题，也就是NP-完全问题。NPC问题的出现使整个NP问题的研究得到了飞跃式的发展。我们有理由相信，NPC问题是最复杂的问题。再次回到全文开头，我们可以看到，人们想表达一个问题不存在多项式的高效算法时应该说它“属于NPC问题”。此时，我的目的终于达到了，我已经把NP问题和NPC问题区别开了。到此为止，本文已经写了近5000字了，我佩服你还能看到这里来，同时也佩服一下自己能写到这里来。

NPC问题的定义非常简单。同时满足下面两个条件的问题就是NPC问题。首先，它得是一个NP问题；然后，所有的NP问题都可以约化到它。证明一个问题是 NPC问题也很简单。先证明它至少是一个NP问题，再证明其中一个已知的NPC问题能约化到它（由约化的传递性，则NPC问题定义的第二条也得以满足；至于第一个NPC问题是怎么来的，下文将介绍），这样就可以说它是NPC问题了。

既然所有的NP问题都能约化成NPC问题，那么只要任意一个NPC问题找到了一个多项式的算法，那么所有的NP问题都能用这个算法解决了，NP也就等于P 了。因此，给NPC找一个多项式算法太不可思议了。因此，前文才说，“正是NPC问题的存在，使人们相信P≠NP”。我们可以就此直观地理解，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。

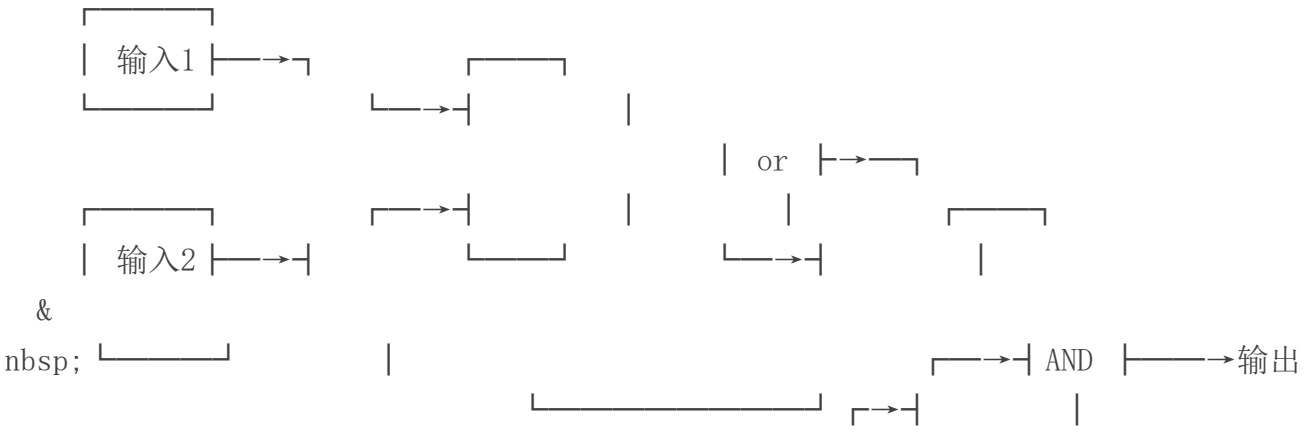
顺便讲一下NP-Hard问题。NP-Hard问题是这样一种问题，它满足NPC问题定义的第二条但不一定要满足第一条（就是说，NP-Hard问题要比 NPC问题的范围广）。NP-Hard问题同样难以找到多项式的算法，但它不列入我们的研究范围，因为它不一定是NP问题。即使NPC问题发现了多项式级的算法，NP-Hard问题有可能仍然无法得到多项式级的算法。事实上，由于NP-Hard放宽了限定条件，它将有可能比所有的NPC问题的时间复杂度更高从而更难以解决。

不要以为NPC问题是一纸空谈。NPC问题是存在的。确实有这么一个非常具体的问题属于NPC问题。下文即将介绍它。

下文即将介绍逻辑电路问题。这是第一个NPC问题。其它的NPC问题都是由这个问题约化而来的。因此，逻辑电路问题是NPC类问题的“鼻祖”。

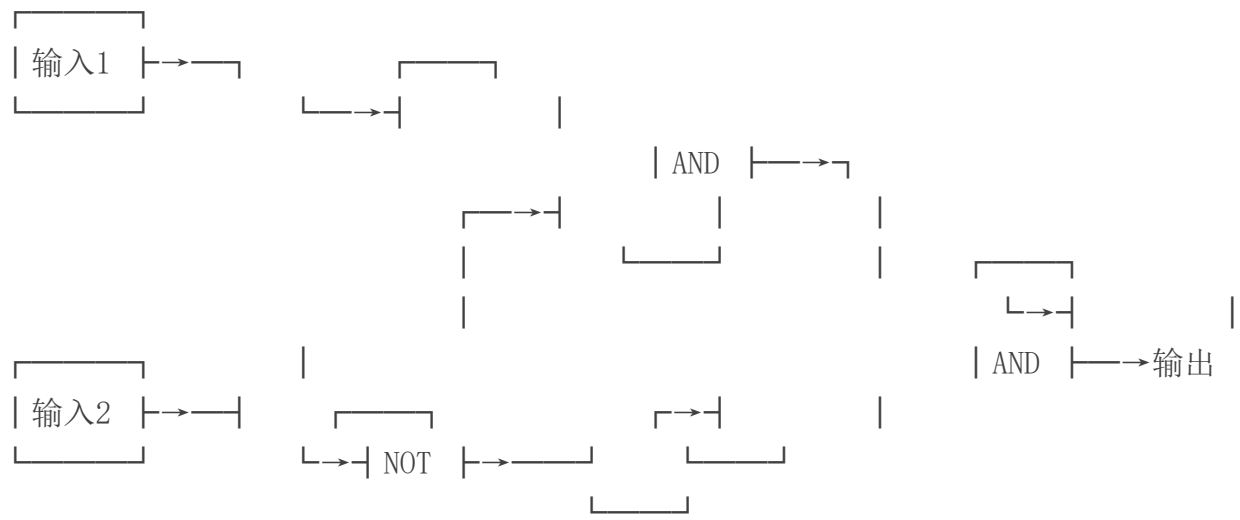
逻辑电路问题是指的这样一个问题：给定一个逻辑电路，问是否存在一种输入使输出为True。

什么叫做逻辑电路呢？一个逻辑电路由若干个输入，一个输出，若干“逻辑门”和密密麻麻的线组成。看下面一例，不需要解释你马上就明白了。



这是个较简单的逻辑电路，当输入1、输入2、输入3分别为True、True、False或False、True、False时，输出为True。

有输出无论如何都不可能为True的逻辑电路吗？有。下面就是一个简单的例子。



上面这个逻辑电路中，无论输入是什么，输出都是False。我们就说，这个逻辑电路不存在使输出为True的一组输入。

回到上文，给定一个逻辑电路，问是否存在一种输入使输出为True，这即逻辑电路问题。

逻辑电路问题属于NPC问题。这是有严格证明的。它显然属于NP问题，并且可以直接证明所有的NP问题都可以约化到它（不要以为NP问题有无穷多个将给证明造成不可逾越的困难）。证明过程相当复杂，其大概意思是说任意一个NP问题的输入和输出都可以转换成逻辑电路的输入和输出（想想计算机内部也不过是一些 0和1的运算），因此对于一个NP问题来说，问题转化为了求出满足结果为True的一个输入（即一个可行解）。

有了第一个NPC问题后，一大堆NPC问题就出现了，因为再证明一个新的NPC问题只需要将一个已知的NPC问题约化到它就行了。后来，Hamilton 回路成了NPC问题，TSP问题也成了NPC问题。现在被证明是NPC问题的有很多，任何一个找到了多项式算法的话所有的NP问题都可以完美解决了。因此说，正是因为NPC问题的存在，P=NP变得难以置信。P=NP问题还有许多有趣的东西，有待大家自己进一步的挖掘。攀登这个信息学的巅峰是我们这一代的终极目标。现在我们需要做的，至少是不要把概念弄混淆了。

Matrix67原创
转载请注明出处

2006年8月28日 / 复杂度

278 条评论

那么那么地

牛啊，博主不是一般的牛
能把比较复杂的东西表达的非常清晰非常条理，真不简单

2007年7月2日 17:46 / 回复

懒羊羊

感谢楼主,总算明白这三个东西了

2007年7月5日 11:56 / 回复

guest

很牛的楼主，可以将问题表述的如此有条理和清晰；坚持不懈，中国将出现另外一个大家，建议学习“自然语言处理”（AI范畴）：）

2007年12月8日 17:25 / 回复

AIP

Just 2 questions:
Does NP stand for Not-Polynomial?
What does OI mean?

http://isdox.com

回复：Non-deterministic Polynomial
Olympiad in Informatics

2007年12月20日 12:15 / 回复

老哥你是干啥的啊。。。真佩服你在这么浮躁的年代还能耐的住来研究这些。

2007年12月30日 19:57 / 回复

超神火星人 楼主是北大文科生，应用语言系

2018年10月7日 20:53

Alexandra

这句话:
再证明其中一个已知的NPC问题能约化到它.

应该是:
再证明其中一个已知的NP问题能约化到它吧?

回复：没写错，如果一个问题可以由某个npc问题归约得到，则所有的np问题都能归约到它

2007年12月31日 00:51 / 回复

dahe_1984