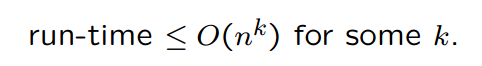
我们现在讨论的问题都是Polynomial Run-Time Complexity

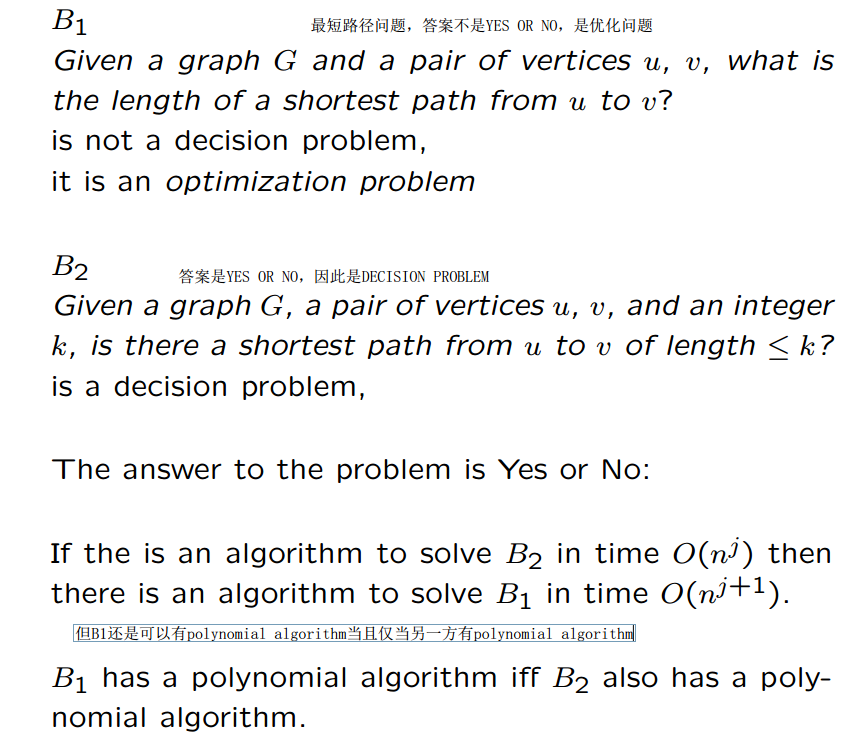
polynomial多项式的



接下来我们会讨论一些no polynomial algorithms

这个问题会被简化如果我们引入Decision problem的概念，（Decision并不一定是No polynomial,也有可能是Polynomial）

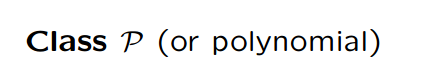
一个问题可以被看作decision problem,当他的答案要么是Yes要么是no



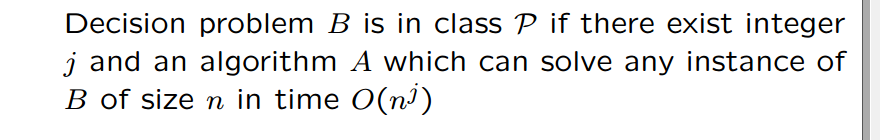
Decision Problem分为两种

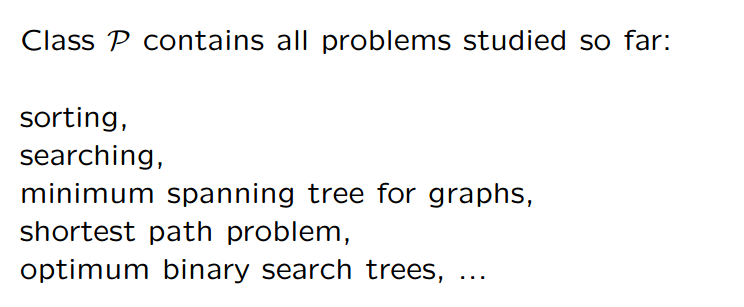
一种是Polynomial

一种是Nondeterministic Polynomial



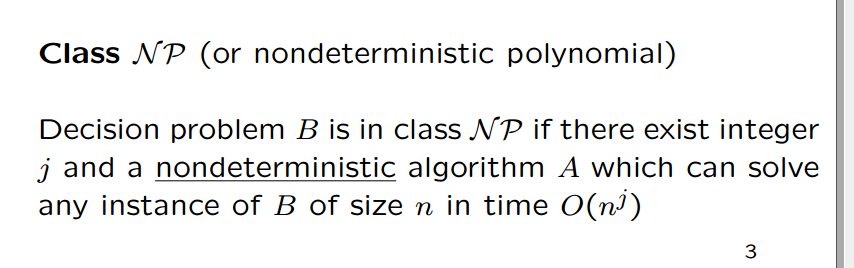
Polynomial decision problem可以看做我们之前学的那些东西的另一种问法，是否存在一个整数j,让算法A在时间以下解决任意size n的问题//问你time complexity变成了是否存在，本质没区别







还是n^j，但解决问题的算法是nondeterministic algorithm



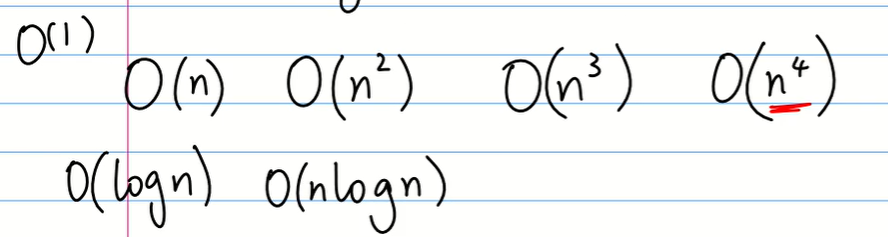
nondeterministic代表算法允许猜测正确答案

我们可以说任意被提议的解可以在polynomial time之内被验证

这是老师版本，太复杂了

P类问题：所有能用多项式时间算法计算得到结果的问题，称为多项式问题，也就是P(polynomial)。

多项式时间举例：



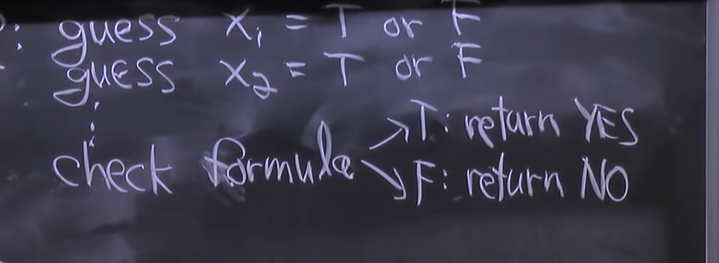
NP类问题(Non-Deterministic Polynomial Problems)：NP问题是指存在多项式算法能够验证的非决定性问题

**NP概念的奥妙在于，它躲开了求解到底需要多少时间这样的问题，而仅仅只是强调验证任意实例需要多少时间**

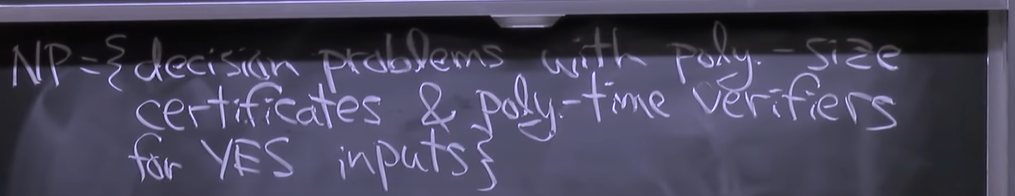
显然，P肯定是NP，因为你既然能用多项式求解，就肯定能用多项式验证（难不成我再算一遍就好了嘛！）

**所以，一个问如果是P类问题，那么必然是NP类问题**（因为P问题就是NP问题的子集啊）

还有一种定义方式，MIT的，



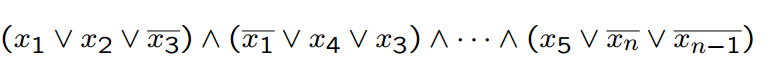
对一个polynomial size的，你可以进行一系列Guess(certificate)，然后poly time verify (重点还是input)

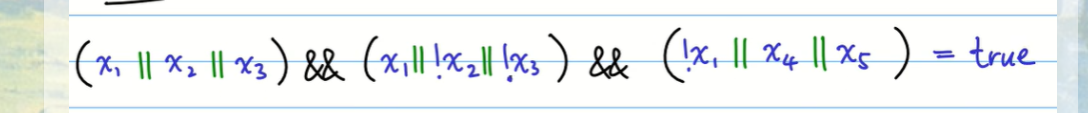


3.NP Complete问题，也叫做NPC问题：一个属于NP类的问题，**但目前为止没办法在P时间内解决**（也就是说目前只能在P时间内判断解的正确性）’

例如经典的3SAT问题，就是标准的NP COMPLETE 问题·

一个括号叫一个clause





对于这么一个判断式，是否存在某一组x1,x2.x3…xn，让判断式=true

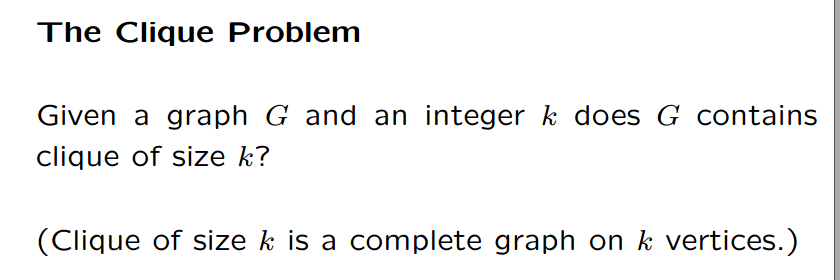
我们不知道怎么通过一个polynomial time算出他

但是，对于给定的一组x1,x2…xn，我们可以在Polynomial time判断是否是True

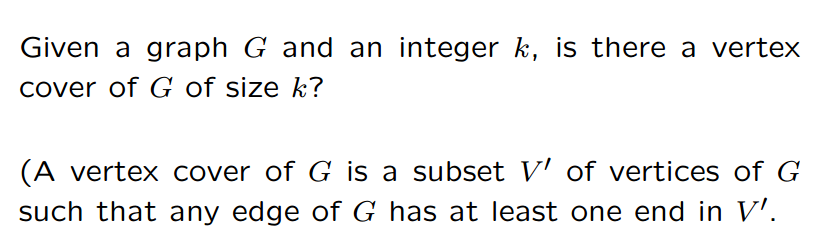
3SAT问题的意义，任意NP问题都可以被表示成是否一个实例能够满足一个Boolean formula

以下问题都是NP complete

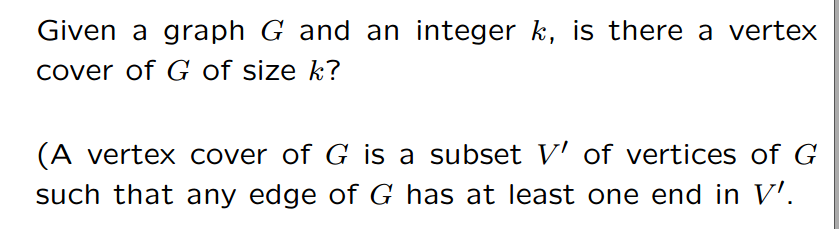
Clique Problem



The Vertex Cover Problem



Vertex Cover Problem



Hamiltonian -cycle problem

Traveling salesman Problem

….

NP-complete

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。

NP-hard

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,sh）。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**

//可以由NPC转化而来，但是不能

小结：

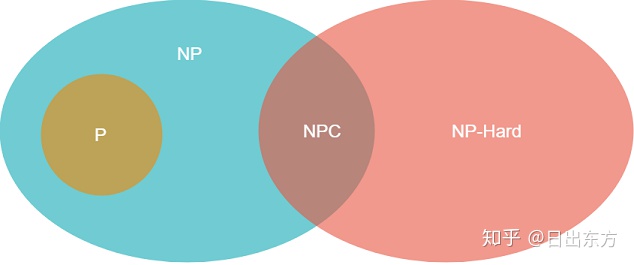
P问题(Polynomial Problem)：可以在多项式时间内解决的问题。

NP问题(Non-Deterministic Polynomial Problem)：可以在多项式时间内验证一个解的问题。

NPC问题(NP Complete Problem)：所有NP问题都可以在多项式时间内约化(Reducibility)到它并且它本身就是一个NP问题的问题。

NP-Hard问题(NP Hard Problem)：所有NP问题都可以在多项式时间内约化(Reducibility)到它的问题。

它们的关系如下：



我们怎么证明一个问题C是NP-complete的

分两步走：

1.首先对于任意一个实例解，存在一个polynomial algorithm能够**验证**它

2.存在一个已知的NP compete problem B, B的任意实例可以通过polynomial time的转换公式，转换成C的实例

这种方式叫做polynomial-time reducibility of B into C



约化

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。

**“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。

**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。**

**存在这样一个NP问题，所有的NP问题都可以约化成它。**

**那就是NPC问题**

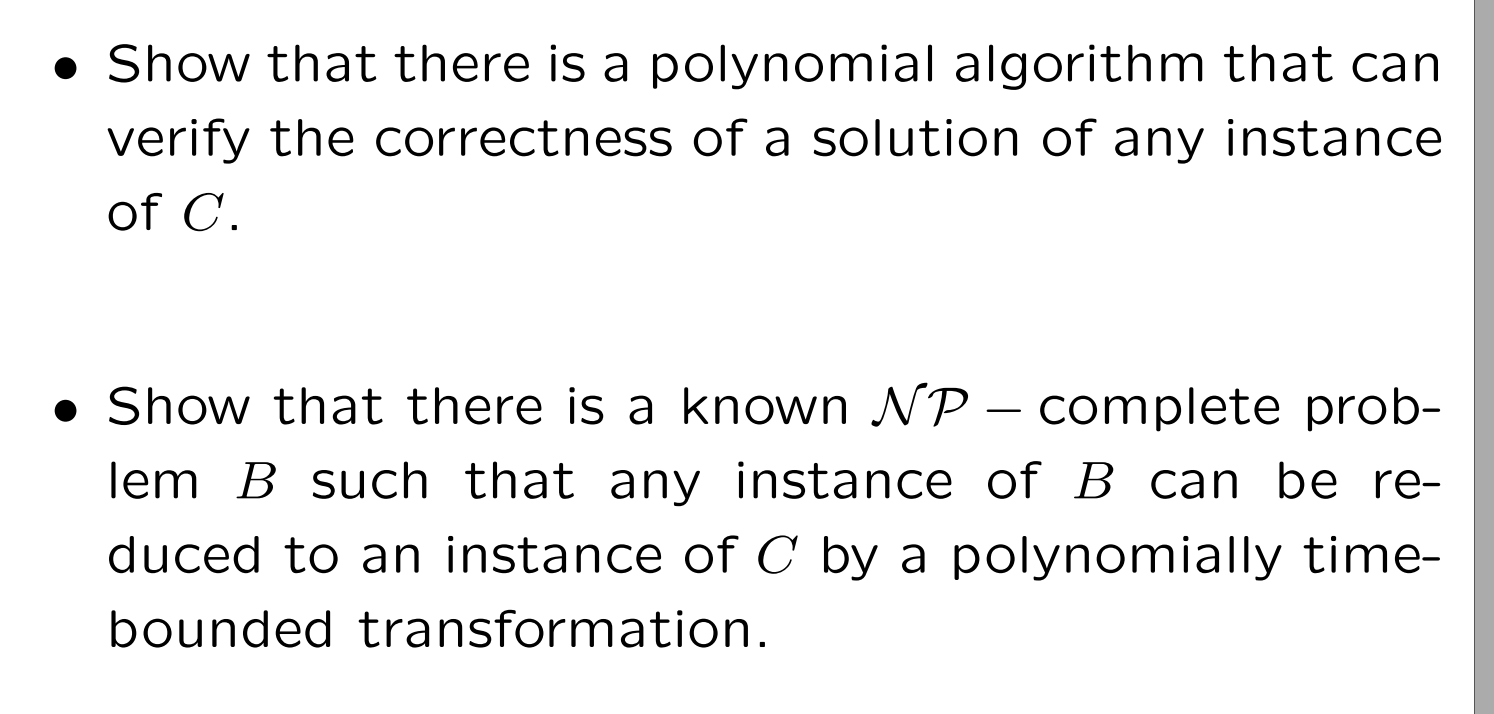
Reduction from problem A to problem B

如果我们能把简单的A转换成复杂的B（NP-complete必然能被转换），且知道怎么解决问题B

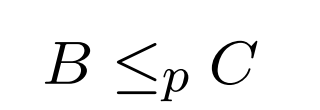
那么我们就把A的input转化成B的input

代入我们能够解的problem B,得到B的解，同时我们也间接得到了A的解因为解是一样的

证明NP complete



STEP2解释：

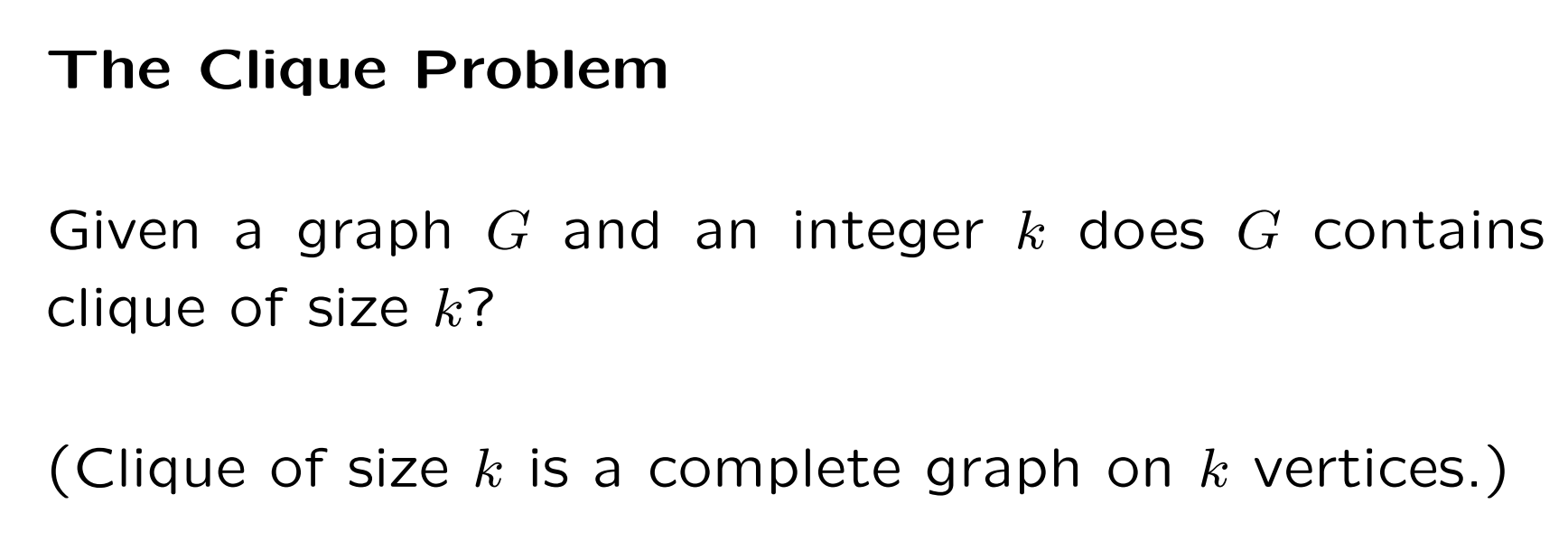
 //C比B复杂，我们能把已知的NP-complete转化成C，说明C要么是NP-HARD要么是NP-COMPLETE,又因为我们能验证（STEP1），所以不是NP-HARD

B代表已知complete, C代表我们关心的

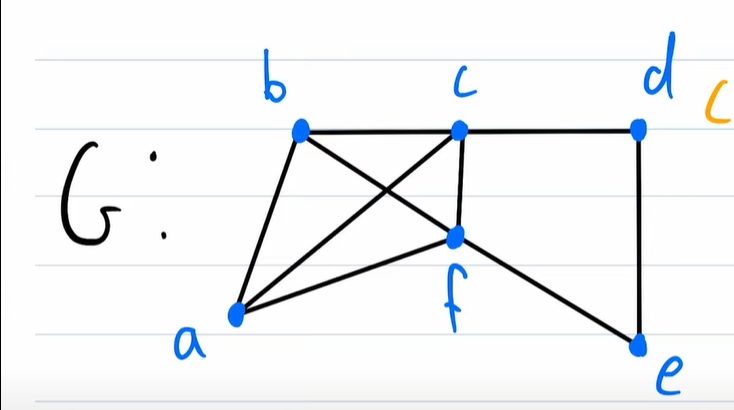
我们有的是C的input，转化成B的input，代入B，得到结果

”

证明Clique Problem是NP complete

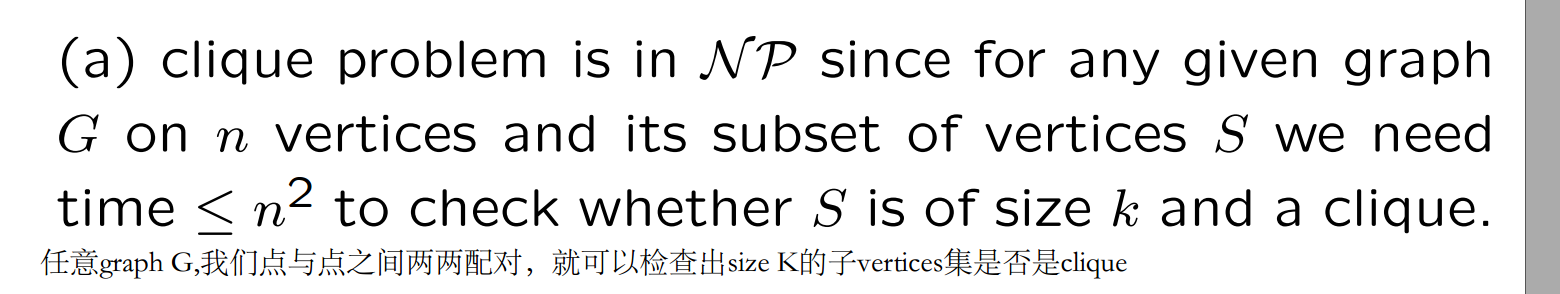


Clique就是任意两个点之间必有edge的complete graph



bcaf就是clique

STEP1:



STEP2:

看笔记

Vertex Cover: G中任意一个edge至少有一端在V'内

Now, we need to show that any instance (G, k) of the Clique problem can be reduced to an instance of the vertex cover problem. Consider the graph G’ which consists of all edges not in G, but in the complete graph using all vertices in G. Let us call this the complement of G. Now, the problem of finding whether a clique of size k exists in the graph G is the same as the problem of finding whether there is a vertex cover of size |V| – k in G’. We need to show that this is indeed the case.

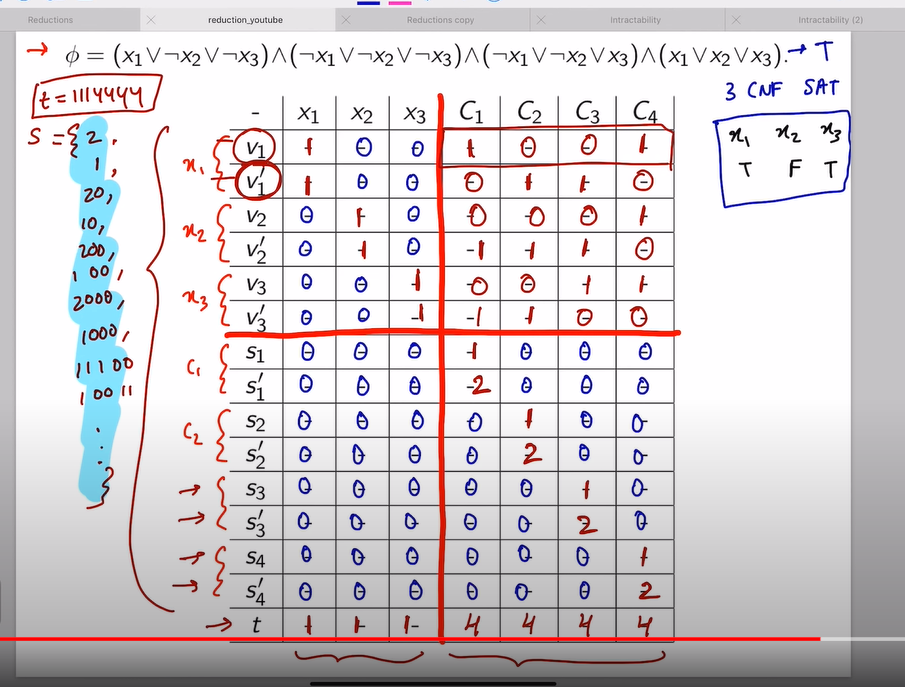
//将G里面存在一个clique转化成G'里存在vertex cover, Size=v-k

Assume that there is a clique of size k in G. Let the set of vertices in the clique be V’. This means |V’| = k. In the complement graph G’, let us pick any edge (u, v). Then at least one of u or v must be in the set V – V’. This is because, if both u and v were from the set V’, then the edge (u, v) would belong to V’, which, in turn would mean that the edge (u, v) is in G. This is not possible since (u, v) is not in G. Thus, all edges in G’ are covered by vertices in the set V – V’.

//V;代表G内clique的端点数， 那么在G'内，任意一条edge的一端必然在V-V'也就是非clique内，因为如果都在V'，那么edge U'V 会变成clique的一员，与我们的假设G'的edge相反，因此G'所有edge都被v-v' cover

成功转换，这个V-V' 就是G'的vertex cover //别忘了证明反面if and only if

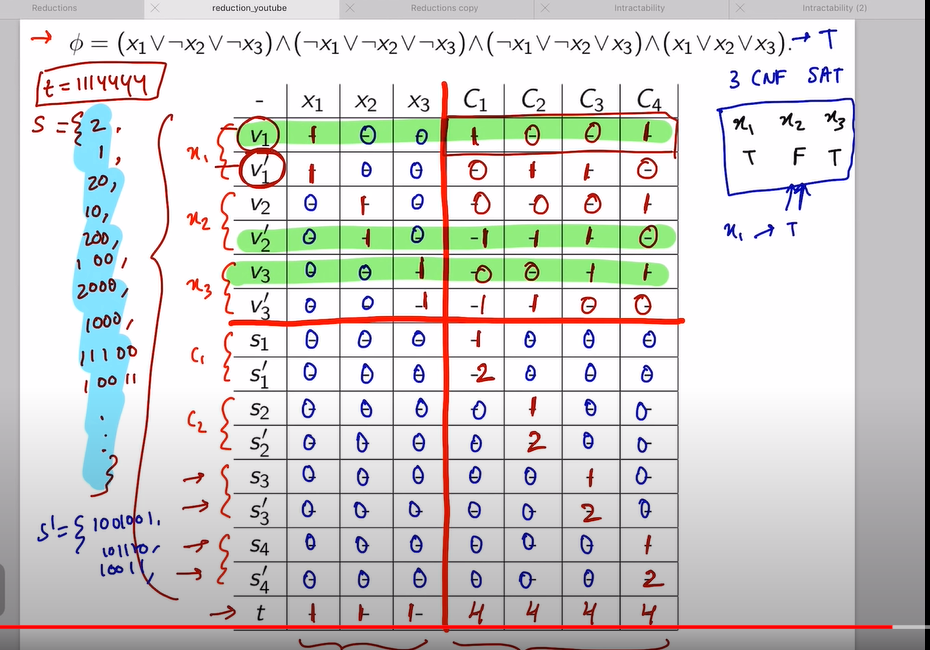
SUBSET的那个 https://www.youtube.com/watch?v=k8RkYp5KhhU



所有v1到s4'是我们的备选数

首先为了满足3SAT,我们至少每个clause要选一个，

例如X1=T,X2=F,X3=T



我们选了v1,v2',v3 //V才是我们根据X选的数，C只是单纯的辅助补足4的数

有的时候C是2，我们补足选2，有的时候是3，我们补足选1

有的时候是1，我们补足选2+1//只要是一组我们补足可以同时选两个

首先我们选好了X1,X2,X3,满足了3sat

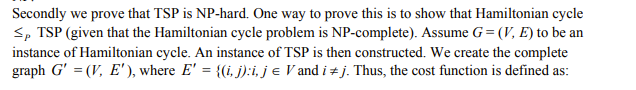
其次我们根据3SAT成功构建了S（全集）的范围以及TARGET，

并且我们成功地根据x1x2x3的值挑选出了target

# traveling salesman problem

Proof:

First, we have to prove that TSP belongs to NP. If we want to check a tour for credibility, we check that the tour contains each vertex once. Then we sum the total cost of the edges and finally we check if the cost is minimum. This can be completed in polynomial time thus TSP belongs to NP





如果是G的一部分，是0，不是1，

https://www.csd.uoc.gr/~hy583/papers/ch11.pdf