R-13.1

Professor Amongus has shown that a decision problem L is polynomial-time reducible to an NP-complete problem M. Moreover, after 80 pages of dense mathematics, he has also just proven that L can be solved in polynomial time. Has he just proven that $\mathrm{P} = \mathrm{NP}$? Why or why not?

Professor Amongus has reduced this problem in P to a problem in NP, and as we know, every problem in P is in NP, which means that we can reduce this problem in NP to a problem in P. As a result, $\mathrm{P} = \mathrm{NP}$.

R-13.7

Show that the CLIQUE problem is in NP.

In this problem, we can guess k vertices in the graph and check if each two vertices are adjacent. As a result, this problem is in NP.

R-13.12

Professor Amongus has just designed an algorithm that can take any graph G with n vertices and determine in $O(n^k)$ time whether or not G contains a clique of size k. Does Professor Amongus deserve the Turing Award for having just shown that $\mathrm{P} = \mathrm{NP}$? Why or why not?

No. In this problem, k is just a number, not the input size, and we don’t know what the value of k is, so we are not sure that this algorithm is a polynomial-time algorithm, which means that this algorithm is not in P. As a result, this problem cannot prove that $\mathrm{P} = \mathrm{NP}$.

C-13.7

Define SUBGRAPH-ISOMORPHISM as the problem that takes a graph G and another graph H and determines if H is a subgraph of G. That is, there is a mapping from each vertex v in H to a vertex f(v) in G such that, if (v, w) is an edge in H, then (f(v), f(w)) is an edge in G. Show that SUBGRAPH-ISOMORPHISM is NP-complete.

To prove this problem is NP-complete problem, we need to base on the reduction of CLIQUE problem, which is NP-complete problem. In this problem, we can assume that the number of vertices in graph H is k. We can guess k vertices in graph G and the edges between these vertices. We can check that each edge in Graph H is equal to the edge in Graph G in polynomial time.

C-13.13

Show that KNAPSACK problem is solvable in polynomial time if the input is given in a unary encoding. That is, show that KNAPSACK is not strongly NP-hard. What is the running time of your algorithm?

Algorithm:

If the profits of projects are small numbers, they will be bounded by a polynomial and 1/ε where ε is a bound on the correctness of the solution, which means we can find a solution in polynomial time.

We can assume that:

$P = max_{a \in S}\ \text{profit}(a)$

$K = \dfrac{\varepsilon P}{n}$ (the error parameter $\varepsilon < 0$)

For each object, $\text{profit}'(i) = \left\lfloor \frac{\text{profit}(i)}{K} \right\rfloor$

With these profits of objects, use the dynamic programming algorithm to find the most profitable set S and output S.

$$\text{profit}(\mathrm{S}) = \sum_{i \in S} \text{profit}(i)$$

$$\geq \sum_{i \in S} \left(K \left\lfloor \frac{\text{profit}(i)}{K} \right\rfloor \right) = \sum_{i \in S} \left(K \times \text{profit}'(i) \right)$$

$$\geq \sum_{i \in OPT} \left(K \times \text{profit}'(i) \right) = \sum_{i \in OPT} \left(K \left\lfloor \frac{\text{profit}(i)}{K} \right\rfloor \right)$$

$$\geq \sum_{i \in OPT} (\text{profit}(i) - K) = \sum_{i \in OPT} \text{profit}(i) - \sum_{i \in OPT} \frac{\varepsilon P}{n}$$

$$\geq \text{profit}(OPT) - \varepsilon \times \text{profit}(OPT) = (1 - \varepsilon) \times \text{profit}(OPT)$$

Running time:

For a general KNAPSACK problem, the running time is $O(NV)$ which N is the number of the object and V is the capacity of the bag. For this algorithm, N is n and V is $n \times \mathrm{v}_{max}/K$. As a result, the running time is $O(NV) = O(n \times n \times \mathrm{v}_{max}/K) = O(n \times n \times n/\varepsilon) = O(n^3/\varepsilon)$, which is polynomial in n and 1/ ε. As a result, KNAPSACK is not strongly NP-hard.