1.

Solution:

We will add free variables to a clause that has fewer than 3 literals in order to transform it.

When a clause has more than three literals, it will be divided into two or more parts, with each piece having three literals while also having one or more free variables added. The original clause and the new clauses must both be equivalently satisfied.

Additionally, if a clause has fewer than three literals, we will add a free variable to increase the number of literals in the clause, as well as create another clause such that the satisfiability of the new clause's product is only dependent on the satisfiability of the original clause.

As a result, the first clause of the given CNF formula B has one literal, so we add two free variables to make it y_1 and y_2 to make it

$(x_1 \vee y_1 \vee y_2) \wedge (x_1 \vee y_1 \vee y'_2) \wedge (x_1 \vee y'_1 \vee y_2) \wedge (x_1 \vee y'_1 \vee y'_2)$. So if $x_1$is satisfiable then $(x_1 \vee y_1 \vee y_2) \wedge(x_1 \vee y_1 \vee y'_2) \wedge (x_1 \vee y'_1 \vee y_2) \wedge (x_1 \vee y'_1 \vee y'_2)$ is satisfiable and vice-versa.

$x_2' + x_3 + x_5 + x_6'$ will become $(x_2' \vee x_3 \vee y_3) \wedge (y_3' \vee x_5 \vee x_6')$

$x_1 + x_4$ will be transformed to $(x_1 \vee x_4 \vee y_4) \wedge (x_1 \vee x_4 \vee y_4')$

$x_3 + x_5$ will become $(x_3 \vee x_5' \vee y_5) \wedge (x_3 \vee x_5' \vee y_5')$

Hence the 3SAT formula equivalent to the above is obtained by combining all clauses using AND operator will be

$(x_1 \vee y_1 \vee y_2) \wedge (x_1 \vee y_1 \vee y'_2) \wedge (x_1 \vee y'_1 \vee y_2) \wedge (x_1 \vee y'_1 \vee y'_2) \wedge (x_2' \vee x_3 \vee y_3) \wedge (y_3' \vee x_5 \vee x_6') \wedge (x_1 \vee x_4 \vee y_4) \wedge (x_1 \vee x_4 \vee y_4') \wedge (x_3 \vee x_5' \vee y_5) \wedge (x_3 \vee x_5' \vee y_5')$

2.

Solution:

No

Determining whether a graph has a clique of size K is not NP-complete and hence it does not prove that P = NP.

3.

Solution:

To begin, keep in mind that HYPER-COMMUNITY is in NP since a nondeterministic machine might just estimate k websites and check that they are all connected to one another.

Next, we reduce from INDEPENDENT-SET to prove that HYPER-COMMUNITY is NP-hard. Assume we have an n-vertex graph G and wish to identify an independent collection of size k. On the same n vertices, we build G′, where (v,w) is an edge in G′ if and only if it is not an edge in G. Because we only have to iterate over all pairs of vertices, this reduction obviously takes polynomial time.

Now, if G′ contains a collection of k mutually linked vertices, they must constitute an independent set in G. In contrast, if G contains an independent collection of size k vertices, then those k vertices must all be linked in G′. HYPER-COMMUNITY must be NP-complete because INDEPENDENT-SET reduces it.

4.

Solution:

The collection of decision problems that can be solved in polynomial time on a non-deterministic Turing computer and whose answers can be verified in polynomial time are known as NP-complete questions. A decision problem C is NP-complete if and only if the following conditions are met:

C exists in NP, and any NP issue can be reduced to C in polynomial time by proving that a candidate solution to C can be verified in polynomial time, C may be demonstrated to be in NP.

First, we must demonstrate that it is an NP problem. We need a polynomial-time verifier to show it is in NP.

We can easily determine whether a company is non-competitive and does not belong to a pair of competing firms from the previous year in polynomial time. So it is proven in NP.

Evidence that it is NP-Hard

We take a problem that has already been demonstrated to be NP-Hard. We must demonstrate that any NP-hard problem can be reduced to a DR drama problem. Inviting all firms is a difficult challenge, and not accepting non-competitive companies is an example of inviting all companies (if we include restrictions in our algorithm). hence it is np hard

5.

Solution:

In the Euclidean traveling salesman problem, cities are points in the plane, and the distance between two cities is the Euclidean distance between the points for these cities, i.e. the length of the straight line connecting these points. Prove that a simple polygon, that is, a linked sequence of line segments that never cross, is an optimum solution to the Euclidean TSP.

Because a straight line is the shortest path between two places, a series of straight lines cannot intersect.

Hence, an optimal solution to the Euclidean TSP is a simple polygon, that is, a connected sequence of line segments such that no two ever cross.

6.

Solution:

Think of K as the maximum weight that the vehicle is permitted to transport.

Take into account that there are n boxes, labeled b1, b2,..., bn, and that wi is the integer weight of the box b.

A greedy algorithm is an example.

Every box bi of weight wi is loaded into the first truck via the greedy method.

The greedy method loads the box onto the following truck if the first truck's weight exceeds K when it is being loaded.

The greedy algorithm puts the box onto the next vehicle if the weight of the second truck likewise exceeds K when it is being loaded.

So on and so forth, until all the boxes are loaded, the procedure above is carried out.

The ratio of the Greedy solution is approximate:Assume that n boxes are loaded onto n vehicles.

If one truck *j* is half full(i.e., *Wj*>*K*/2), then the greedy algorithm loads an item of weight *wj*>*K*/2 onto a new truck.

Thus, if *N* trucks are used to load all *n* boxes, *N*-1 trucks are more than half full.

$$\sum_{i=1}^{n} w_j > (N-1) * k/2$$

i.e. $2 * \sum_{i=1}^{n} w_j > (N-1) * k$

i.e. $2 * \sum_{i=1}^{n} w_j / k > (N-1)$

Hence $\sum_{i=1}^{n} w_j / k = N'$ gives us the total number of trucks to load n boxes. Therefore,

$2N' > (N-1)$

Therefore we prove that our algorithm uses a number of trucks that is within a factor of 2 of the optimal number of trucks.