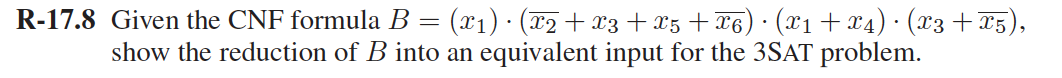
Chapter 17 Exercises: R-17.8,



To convert a clause containing less than 3 literals, we will add free variables so that the clause have 3 literals.

And the clause containing more than 3 literals, we will break the clause in two or more parts such that each clause should have 3 literals while adding one or more free variables and satisfiability of original clause and new clauses should be equivalent.

And the clause containing less than 3 literals, we will add free variable to make it clause of 3 literals and creating additional clause such that satisfiability of product of clause depend only on satisfiability of original clause.

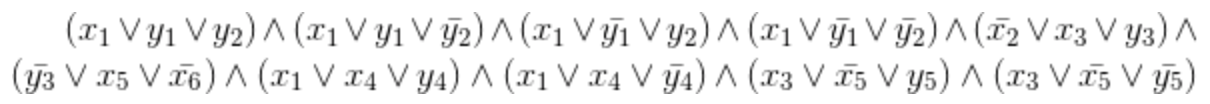
So in the given CNF formula B, the first clause is (x_1) having one literals, so we add two free variables y and y2 to make it  . So if (x_1) is satisfiable then   is satisfiable and vice-versa.

(\bar{x_2}+x_3+x_5+\bar{x_6}) will be break into (\bar{x_2}\vee x_3 \vee y_3)\wedge(\bar{y_3} \vee x_5 \vee \bar{x_6}) .

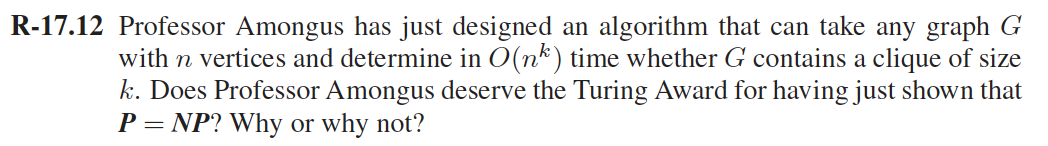
(x_1+x_4) will be transformed to (x_1 \vee x_4 \vee y_4)\wedge (x_1 \vee x_4 \vee \bar{y_4}) .

(x_3+\bar{x_5}) will be transformed to (x_3 \vee \bar{x_5}\vee y_5) \wedge (x_3 \vee \bar{x_5}\vee \bar{y_5}) .

Hence the 3SAT formula equivalent to above is obtaining by combining all clause using AND operator :-



R-17.12,



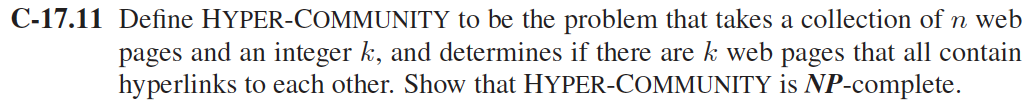
Answer：

YES ,he deserves that award if he has solved that P = NP problem because it's not a just thing.Many researchers are chasing for that and if he is the one who stands out by proving that solution then its a remarkable success for all of us.

A proof that **P** = **NP** could have stunning practical consequences if the proof leads to efficient methods for solving some of the important problems in **NP**. It is also possible that a proof would not lead directly to efficient methods, perhaps if the proof is non-constructive, or the size of the bounding polynomial is too big to be efficient in practice. The consequences, both positive and negative, arise since various **NP**-complete problems are fundamental in many fields

Cryptographic hashing as the problem of finding a pre-image that hashes to a given value must be difficult in order to be useful, and ideally should require exponential time. However, if **P**=**NP**, then finding a pre-image M can be done in polynomial time, through reduction to SAT

C-17.11,



First, note that HYPER-COMMUNITY is in NP, since a nondeterministic machine could simply guess k web pages, and check that they are all connected to one another.

Next, to show that HYPER-COMMUNITY is NP-hard, we reduce from INDEPENDENTSET. Suppose that we have a graph G with n vertices, and we want to find an independent set of size k. We construct G’ on the same n vertices, where (v, w) is an edge in G’ if and only if it is not an edge in G. This reduction obviously takes polynomial time, since we only have to iterate over all pairs of vertices.

Now if there is a set of k mutually connected vertices in G’ , then they must form an independent set in G. Conversely, if there is an independent set of size k in G, then those k vertices must all be connected in G’ .

Since INDEPENDENT-SET reduces to HYPER-COMMUNITY, HYPER-COMMUNITY must be NP-complete.

A-17.1

*NP* -complete problems defined as the set of decision problems that can be solved in polynomial time on a non-deterministic Turing machine and whose solutions can be verified in polynomial time.

A decision problem C is NP-complete if:

1. C is in NP, and
2. Every problem in NP is reducible to C in polynomial time.

C can be shown to be in NP by demonstrating that a candidate solution to C can be verified in polynomial time.

**first we need to prove it is NP problem** To prove it is in NP we must have a polynomial-time verifier.

we can easily find out in polynomial time whether the companies belongs to non competiting and also not belong to  pair of competing companies of last year. So it is in NP proved.

**Proof that it is NP Hard**

we take some problem which has already been proven to be NP Hard.  we need to show that any instance NP hard problem  can be reduced to an instance of DR drama problem. inviting all companies is np hard problem and not inviting the non competing companies is an instance of inviting all companies(if we put restriction in our algorithim) . so it is np hard.

Chapter 18 Exercises: C-18.7,

Give a pseudocode description of the branch-and-bound algorithm for TSP.

Let the nodes N = {1, 2, ... , n}. The following algorithm generates all possible solutions, and picks the shortest. S is an ordered set which includes a partial path (ordered list of k integers) and the sum of its edge weights w: S = (k, [i1, i2, ... ,ik], w)

Pseudocode Alogrithm:

w = w(1,2) + w(2,3) + w(3,4) + ... + w(n-1,n) + w(n,1)

Best\_S\_so\_far = ( n, [ 1, 2, 3, ... , n-1, n ], w )

S = ( 1, [ 1 ], 0 )

BnBSearch( S, Best\_S\_so\_far );

Procedure BnBSearch ( S, Best\_S\_so\_far )

let ( k, [ i1, i2, ... , ik ], w ) = S

let ( n, [ i1B, i2B, ... , inB ], wB ) = Best\_S\_so\_far

if k = n then

new\_w = w + w(ik,i1)

// if a partial tour is not longer than the best solution found so far, then continue searching that path.

if new\_w < wB then

Best\_S\_so\_far = ( k, [ i1, i2, ... , ik ], new\_w )

end if

else

for all j not in [ i1, i2, ... , ik ]

new\_w = w + w(ik,j)

//// if a partial tour is not longer than the best solution found so far, then continue searching that path.

if new\_w < wB then

New\_S = ( k+1, [ i1, i2, ... , ik, j ], new\_w)

BnBSearch ( New\_S, Best\_S\_so\_far )

end if

end for

endif

return

end

A-18.4

A greedy algorithm might use for this is the following.

Start with an empty truck, and begin piling boxes 1, 2, 3, . . . into it until you get to a box that would overflow the weight limit. Now declare this truck “loaded” and send it off; then continue the process with a fresh truck.

Let W= . Note that in any solution , each truck holds at most M units of weight, so W/M is a lower bound on the number of trucks needed. Suppose the number of trucks used by our greedy algorithm is an odd number m=2q+1.( the case when m is even is essentially the same). Dived the trucks used into consecutive groups of two, for a total of q+1 groups. In each group but the last ,the total weight of boxes must be strictly greater that M, otherwise the second truck in the group would not have been started then). Thus, W> qM, and so W/M >q. It follows by our argument above that the optimum solution uses at least q+1 trucks, which is within a factor of 2 of m =2q+1.