Romil V. Shah

20008692

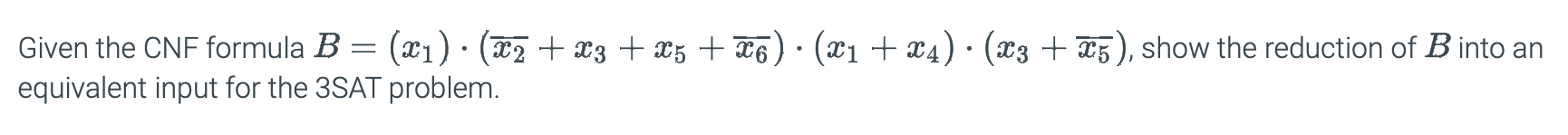
CS / CPE 600

Prof. Reza Peyrovian

Homework Assignment 9

Submission Date: 11/20/2022

Q1. No. 17.8.8.



Sol.

B = (x1) · (x2 + x3 + x5 + x6) · (x1 + x4) · (x3 + x5)

Let us break B and apply local replacements for each Bi in B

B1 = (x1) = (x1 + ~x7 + ~x8) · (x1 + ~x7 + x8) · (x1 + x7 + ~x8) · (x1 + x7 + x8)

B2 = (x2 + x3 + x5 + x6) = (x2 + x3 + x9) · (~x9 + x5 + x10) · (~x10 + x6 + x11)

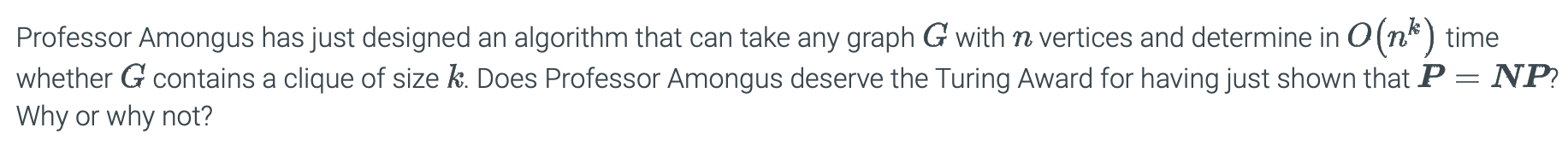
B3 = (x1 + x4) = (x1 + x4 + x11) · (x1 + x4 + ~x11)

B4 = (x3 + x5) = (x3 + x5 + x12) · (x3 + x5 + ~x12)

Therefore

B = (x1 + ~x7 + ~x8) · (x1 + ~x7 + x8) · (x1 + x7 + ~x8) · (x1 + x7 + x8) · (x2 + x3 + x9) · (~x9 + x5 + x10) · (~x10 + x6 + x11) · (x1 + x4 + x11) · (x1 + x4 + ~x11) · (x3 + x5 + x12) · (x3 + x5 + ~x12)

Q2. No. 17.8.12



Sol.

The clique problem is NP complete. Reduction from vertex-cover.

Input: G = (V, E) and integer k > 0

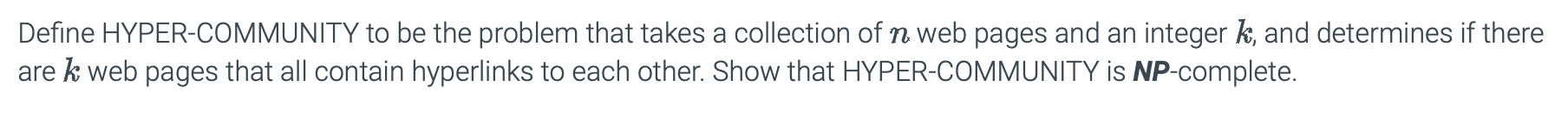
1. Non-deterministically select a subset C of nodes of G.
2. Test whether all nodes in c are connected and whether G contains all edges connecting nodes in C.
3. If yes, accept
4. Else reject

To show P = NP, there must be every problem that belongs to NP can be solved in polynomial time. For any value of k, it must be solvable in polynomial time. The value for k is not defined.

No, determining whether a graph has a clique of size K is not NP complete and hence it does not prove P = NP problem.

No, The Turing Award is given for major contributions to computer science. While Professor Amongus algorithm may be a major contribution, it has not been proven to be correct. Therefore, it does not deserve the Turing Award.

Q3. No. 17.8.28



Sol.

Let us consider that HYPER-COMMUNITY is in NP, since a non-deterministic machine could simply guess ‘k’ web pages and check that they are all connected to one another.

Now reducing HYPER-COMMUNITY from Independent-Set. Let’s say a graph G with n vertices, find independent set f size k. Now consider another graph G’ having same n vertices as G. Graph G’ contain an edge (u, v) if and only if this is not in G. So, iterating over all the pair of vertices, whole process will take polynomial time to run.

If there is an independent set of k size in G, then all k vertices are connected in G’ and, we can say that if set of k mutually connected vertices in G’, then k vertices form an independent set in G.

As Independent-Set is reducible to HYPER-COMMUNITY. So, it is NP-complete.

Q4. No. 17.8.35

Text

Description automatically generated

Sol.

The above problem is vertex cover problem. An instance of vertex cover problem is a graph G(V, E) and a positive integer k. Now check it is NP-complete or not.

To prove it is in NP we must have a polynomial-time verifier. We can easily find out in polynomial time whether the companies belong to non-competing and also not belong to pair of competing companies of last year. So, it is in NP proved.

For a problem to be NP-complete, it must satisfy the two conditions:

1. Proof that the problem is in NP.
2. Proof that the problem is NP-Hard.

Proof that the problem is in NP.

1. Non-deterministically choose a subset W of size k.
2. Test that for each vertex v in W remove all edges adjacent to v from set X. If the edges removed is equal to k and set X gets empty.
3. If yes, accept
4. Else reject

As above algorithm can be done in polynomial time thus it is in NP.

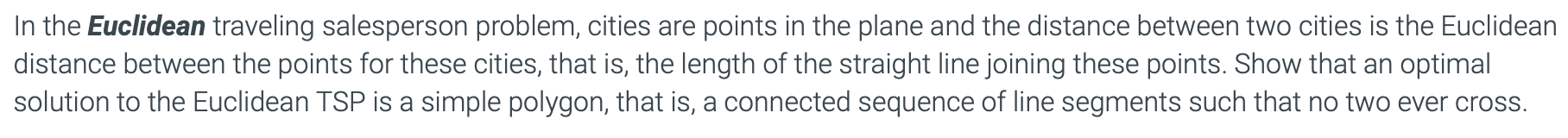
Proof that the problem is NP-Hard

1. For NP-Hard reduce 3SAT to Vertex-Cover.
2. Let S be the Boolean formula in CNF with each clause having 3 literals
3. Now construct a graph G and an integer k such that G has a vertex cover if and only if S is satisfiable.
4. According to “truth setting” component and “Satisfaction setting component” vertex cover must include one of the two vertices.

According to theorem 17.5.1 given in the ZyBook

Vertex-Cover is NP complete.

Q5. No. 18.6.19



Sol.

Assume that two edges cross (x, y) (u, v) each other like shown in the figure. Now considering the path P followed from y to x then path x to v and then v to u is optimal.

Considering another path P’ from y to v (straight line) then v to x then x to u (straight line). By proving that the optimal solution for Euclidean TSP will be achieved from path P’.

Let there is a point d which is the intersection point of the two edges (x, y) (u, v). Let d(w, z) be the distance between w and z.

Now according to the Euclidean formula

d(u, v) + d(x, y) = d(y, d) + d(d, x) + d(u, d) + d(d, v)

By triangle inequality

d(y, d) + d(d, v) >= d(y, v) and

d(u, d) + d(d, x) >= d(u, x)

Now, above equality gives

d(x, y) + d(u, v) >= d(y, v) + d(u, x)

Path P’ incurs a smaller total distance than the original path P. Our assumption contradicts this above situation which means there is no optimal path can have crossing edges.

So, P’ is 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.

Q6. No. 18.6.26

Text

Description automatically generated

Sol.

Algorithm GreedyMinTruck(W, M, n):

Input: Set W of boxes, such that each box i ∈ W has a positive weight wi ; positive maximum total weight M, that is the limit of each truck can carry; and number of boxes

Output: Minimum number of trucks t such that no truck carries more than M pounds

sum ← 0

for each box i ∈ W do

sum ← sum + wi

return sum / M + 1

Now, finding the optimal solution for the problem the minimum trucks required will be less than 2M. A better strategy can be sorting the weights from largest to smallest and insert them in such an order that can best fit into the truck. Scanning all the current filled trucks and checking any weight can be inserted into it, if not then take a new truck for this weight.

The running time of the above algorithm will be O(n2) where n is the number of boxes.