Homework 9 - Solution

R-17.8

$$B = (x_1 + a + b) * (x_1 + \bar{a} + b) * \left(x_1 + a + \bar{b}\right) * \left(x_1 + \bar{a} + \bar{b}\right) * (\overline{x_2} + x_3 + c) * (\bar{c} + x_5 + \overline{x_6}) * (x_1 + x_4 + d) * \left(x_1 + x_4 + \bar{d}\right) * (x_3 + \overline{x_5} + e) * (x_3 + \overline{x_5} + \bar{e})$$

R-17.12

This algorithm does not run in polynomial time since k is a number in the input, not the size of input. If we decide on a value of k, then the algorithm for a fixed k runs in polynomial time. But in other cases it might be exponential, suppose if the value of k is in order of the input size. Therefore, it cannot be proved that P = NP, thus he shouldn't receive the Turing award.

C-17.11

In order to show that HYPER-COMMUNITY is NP-complete, we need to prove that HYPER-COMMUNITY is NP-hard and it is in NP.

First, HYPER-COMMUNITY is an instance of a CLIQUE problem and is in NP because we know that a non-deterministic machine could simply guess k web pages and check that all contain hyperlinks to each and other.

Next, we can show that HYPER-COMMUNITY is NP-hard in two ways:

(1) By reducing this problem from VETEX-COVER problem. using the local-replacement technique. Let (G, k) be an instance of the VERTEX-COVER problem. We construct the complement graph G`, which has the same vertex set as G that consists of edges not in G. We define the integer parameter for HYPER-COMMUNITY as n−k, where k is the integer parameter for VERTEX-COVER. This construction runs in polynomial time and serves as a reduction, for G` has a clique of size at least n – k if and only if G has a vertex cover of size at most k. Thus, HYPER-COMMUNITY is NP-complete.

Or

(2) By reducing this problem from INDEPENDENT-SET (C-17.10 defines INDEPENDENT-SET and we can show that it is NP-complete by reducing from VERTEX-COVER). Now, 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
For NP-complete we must show that it is in NP & NP-Hard.

We construct a graph G with n vertices (representing n companies) and there is an edge (u, v) in G if and only if company u and company v are competitors.

Now we Prove A-17.1 is in NP:
we must show that there exists a polynomial time algorithm that takes an instance of the problem and a certificate as parameters and verifies that the certificate is a yes instance of the input. Thus, our instance is a graph G, and our certificate is a set of vertices of size k. Our algorithm performs 2 steps:

1. We use repeated calls that choose a method to form a collection C of k vertices from the graph G.
2. Then, check if there are no edges between any two vertices in the collection C. If so, we output *yes* otherwise *no*. Such a computation runs in Polynomial time.

Now we prove A-17.1 is in NP-Hard
This problem is in NP-hard because it can be reduced from VERTEX-COVER:
Assume that the vertices set of G is n and the VERTEX-COVER for the Graph G is set k.
If k is a vertex cover of G, then n – k companies are non-competing because at least one vertex of every edge is contained in S. Therefore, there exists a vertex cover of size k if and only if n-k companies are noncompeting. This construction runs in polynomial time and serves as a reduction. So, A-17.1 is NP-Hard.

Since it is both NP & NP-Hard It is an NP-Complete problem.

Alternative Solution: Model the problem using a graph, where companies are vertices and (u, v) is an edge if and only if u and v are competitors. Then we need to show that the decision version of this problem is an instance of the CLIQUE problem as opposed to VERTEX COVER.

C-18.7
In order to prove that an optimal solution to the Euclidean TSP is a simple polygon, we use contradiction method. We assume that the optimal solution is a graph with cross, that is, a connected sequence of line segments such that two of those, say (a, b) and (c, d) crosses at point e. So, we have $(a,b) + (c,d) = (a,e) + (e,b) + (c,e) + (e,d)$

Now, by triangle inequality,
$(a,e) + (e,d) \geq (a,d)$ and also $(c,e) + (e,b) \geq (c,b)$

Using above two equations, we can say that,
$(a,b) + (c,d) \geq (a,d) + (c,b)$

From the above equation, we conclude that our assumption is wrong as the distance of path with cross is greater than the distance of path with no cross. Thus, by contradiction, it proves that an optimal solution to the Euclidean TSP is a simple polygon.

A-18.4

Consider a greedy algorithm that fills up the trucks one at a time. To see that this is a 2-approximation algorithm, consider boxes to be "big" if they weigh more than M/2 pounds and "small" otherwise. Note that every big box will go in a different truck, in both the greedy solution and the optimal solution. Let G_B be the number of trucks in the greedy solution containing big boxes and let H_B be the number of trucks in the optimal solution containing big boxes. Then $G_B = H_B$.

Thus, let us consider the number of trucks containing only small boxes. In the greedy solution, any truck filled with only small boxes must be filled to have weight at least M/2. Otherwise, we could have put one more small box in that truck. Let G_S be the number of trucks in the greedy solution containing only small boxes and let H_S be the number of trucks in the optimal solution containing only small boxes. Note that the optimal solution might fit some small boxes into trucks that also contain a big box, whereas the greedy solution might not. Still, the most that the optimal solution can fill a truck with small boxes is to weight M and the most it can pack into a truck already containing a big box is M/2. Thus,

$G_S \leq 2H_S + H_B$.

In other words, the number of trucks used in the greedy solution, $G_S + G_B$, satisfies the following:

$G_S + G_B \leq 2H_S + 2H_B = 2(H_S + H_B)$.

Therefore, this greedy solution is a 2-approximation algorithm.