NP-Completeness and Approximation Algorithms Practice

Problem 1. This is a variant of the Clique problem. You are given an undirected graph $G = (V, E)$. We say that a subset $V' \subseteq V$ is a pseudo-clique if for any two vertices $u, v \in V'$ the distance from u to v in G is at most two (that is, either there is an edge between them or they share a common neighbor).
The *Pseudo-Clique Problem* (PC) is, given an undirected graph $G = (V, E)$ and an integer k, does G have a pseudo-clique of size k.
Prove that PC is NP-complete. (Hint: Reduction from Clique.)

Problem 2. Recall that the TSP (optimization) problem is: Given a complete undirected graph $G = (V, E)$ with weighted edges, find the cycle of minimum total cost that visits every node exactly once. Recall that this problem is NP-complete, with the reduction from the Hamiltonian Cycle problem (given an undirected graph $G = (V, E)$, does there exist a simple cycle that visits all the nodes). In class we showed that if the edge weights satisfy the triangle inequality, then there exists a factor-2 approximation.

The goal of this problem is to explore the approximability of TSP if the graph's edge weights *do not* satisfy the triangle inequality.

(a) Give an example of a complete undirected weighted graph (whose edge weights *do not* satisfy the triangle inequality) such that the TSP approximation given in class results in a tour whose cost is *strictly more* than twice the cost of the optimum TSP tour. Show each of the steps of the approximation algorithm on your graph. (Hint: This can be done with as few as four vertices.)

(b) Suppose that for some constant $c > 1$, there existed a polynomial time, factor-c approximation algorithm for TSP for graphs with arbitrary edge weights (that is, not satisfying the triangle inequality). Show that this is very unlikely, by proving that this procedure could be applied to solve the Hamiltonian Cycle problem in polynomial time.

Problem 3. A sequence of positive integers $A =< a_1, ..., a_n >$ is given along with a positive integer B. A subset $S \subseteq A$ is said to be *feasible* if the sum of numbers in S does not exceed B, that is

$$\sum_{a_i \in S} a_i \leq B.$$

This sum is called the *total value* of S. Our objective is to find the feasible subset S of A of maximum total value. For example, given $A =< 5, 9, 13, 15, 19, 27 >$ and $B = 35$, the subset $S = \{5, 13, 15\}$ has a total value of 33. (This is the best solution I found, but there may be a better one.)

Consider the following heuristic for producing a feasible solution. Initialize $S = \emptyset$ and $t = 0$. For i running from 1 to n, if $t + a_i \leq B$, then set $t = t + a_i$ and add a_i to S. For example, on the above example, the output would be $S = \{5, 9, 13\} = 27$. After this, the addition of any of the remaining elements would result in a sum larger than 35. (Note that the elements of A are considered in exactly the order in which they are given.)

For any input (A, B), let t^* denote the total value of an optimal solution, and let t be the total value produced by the above heuristic. The *performance ratio* is $\rho = t^*/t$. The ratio bound is the largest achievable ratio over all possible inputs.

(a) Show that this heuristic could generate arbitrarily bad solutions, in the sense that for any $\rho \geq 1$, there exists an input for which the heuristic has a performance ratio at least as large as ρ.

(b) Show how to modify the above heuristic (or devise a new heuristic of your own) that achieves a performance ratio bound of at most 2. Your heuristic should run in $O(n \log n)$ time. Present a proof of your heuristic's ratio bound.