Exercise 1 (12 Points) In the weighted interval scheduling problem we are given a set $J$ of $n$ jobs, where each job $j$ comes with a start time $s_j$, a finishing time $f_j$, and some value $v_j$ (you may assume that all these values are integers). A feasible solution to the problem is a subset $S \subseteq J$ of the jobs, so that no two jobs in $S$ overlap, in other words, for any $i, j \in S$, $[s_i, f_i] \cap [s_j, f_j] = \emptyset$. Goal is to find such a feasible set $S$ of jobs of maximum total value $\sum_{j \in S} v_j$.

(i) Show that the following two greedy algorithms can produce solutions that are arbitrarily worse than the optimal one.

- **starttime-based greedy**: Go through the jobs in order of ascending start times, while adding them to $S$ if and only if they do not overlap with any other job already in $S$.
- **weight-based-greedy**: Go through the jobs in order of decreasing values, while adding them to $S$ if and only if they do not overlap with any other job already in $S$.

(5 Points)

(ii) Give an optimal, polynomial-time algorithm for the problem using dynamic programming.

(7 Points) *(In case you have difficulties solving this exercise, a pseudopolynomial-time dynamic programming algorithm gives part of the points.)*

Exercise 2 (8 Points) Consider the following greedy algorithm for the knapsack problem. Assume that the items are indexed in order of non-increasing ratio of value to size, i.e., $v_1/s_1 \geq v_2/s_2 \geq \cdots \geq v_n/s_n$, and let $i^*$ be the index of an item of maximum value, i.e., $v_{i^*} = \max_{i \in I} v_i$. The algorithm identifies the largest $k$ so that $\sum_{i=1}^k s_i \leq B$. It then outputs either $\{1, 2, \ldots, k\}$ or $\{i^*\}$, whatever has greater value. Prove that this algorithm is a 2-approximation algorithm for the knapsack problem.

Exercise 3 (10 Points) Suppose we are given a directed acyclic graph with a specified source vertex $s$ and a sink vertex $t$, and each edge $e$ has an associated cost $c_e$ and length $l_e$. You
may assume that $c_e$ and $\ell_e$ are positive integers. Give a fully polynomial-time approximation scheme for the problem of finding a minimum-cost path from $s$ to $t$ of total length at most $L$.

**Exercise 4 (10 Points)** Consider some minimization problem $\Pi$ such that:

- any feasible solution has a non-negative, integer objective function value, and
- there is some polynomial $p$, such that if it takes $n$ bits to encode the input instance $I$ in unary, $OPT(I) < p(n)$.

Prove that if there is a fully polynomial-time approximation scheme for $\Pi$, then there is a pseudopolynomial algorithm for $\Pi$.

*Note: Since there is no pseudopolynomial-time algorithm for a strongly NP-hard problem unless P=NP, you proved that unless P=NP there cannot be any such problem $\Pi$ that is strongly NP-hard.*