Exercise 1 (Minimum Cost Overflow, ★). Consider the following problem: Given \( n \) items of non-negative size \( s_1, \ldots, s_n \) and non-negative cost \( c_1, \ldots, c_n \), and a real \( W \), find a subset \( S \subseteq \{1, \ldots, n\} \) of minimum cost such that \( \sum_{i \in S} s_i \geq W \), where cost\((S) = \sum_{i \in S} c_i \).

**Question 1.1** Give a polynomial-time reduction from Partition to Minimum Cost Overflow, showing that it is NP-complete. (Recall that Partition consists, given \( n \) non-negative integers \( x_1, \ldots, x_n \), in deciding if there is a subset \( S \subseteq \{1, \ldots, n\} \) such that \( \sum_{i \in S} x_i = \sum_{i \not\in S} x_i \).)

We first study the following greedy algorithm: 1) sort and renumber the items so that: \( c_1/s_1 \leq \cdots \leq c_n/s_n \); then 2) consider the smallest index \( i \) such that \( \sum_{j \leq i} s_j \geq W \), and output \( S = \{1, \ldots, i\} \).

**Question 1.2** Show that this algorithm can produce solution whose cost is arbitrarily large with respect to the optimum.

**Hint.** Two items are enough.

**Question 1.3** Assume that the \( c_i \) are integers and let \( C = \max c_i \). Give a poly(\( C, n \))-time algorithm that solves the Minimum Cost Overflow problem. Analyze its asymptotic time and memory complexities precisely in \( C \) and \( n \).

**Question 1.4** Design and analyze a fully polynomial time approximation scheme for the Minimum Cost Overflow Problem, i.e. a \((1 + \varepsilon)\)-approximation that runs in polynomial time in \( n \) and \( 1/\varepsilon \).

**Hint.** First assume that \( C \leq \text{OPT} \) and round the input properly to reduce the number of different costs; and then extend your algorithm to get rid of the assumption \( C \leq \text{OPT} \).