Exercise 2.1 (10 Points) Consider the following greedy algorithm for the knapsack problem. We initially sort all the items in order of non-increasing ratio of value to size so that \( \frac{v_1}{s_1} \geq \frac{v_2}{s_2} \geq \cdots \geq \frac{v_n}{s_n} \). Let \( i^* \) be the index of the item of maximum value. The greedy algorithm puts items in the knapsack in index order until the next item no longer fits; that is, it finds \( k \in \mathbb{N} \) such that \( \sum_{i=1}^{k} s_i \leq B \) but \( \sum_{i=1}^{k+1} s_i > B \). The algorithm returns either \( \{1, \ldots, k\} \) or \( \{i^*\} \), whichever has greater value.

a) Show that the value of an optimal solution is less than \( \sum_{i=1}^{k+1} v_i \).

b) Show that the above greedy algorithm is a \((1/2)\)-approximation algorithm for the knapsack problem.

Exercise 2.2 (10 Points) One key element in the construction of the fully polynomial-time approximation scheme for the knapsack problem was to provide the lower and upper bounds for the optimal value that are within a factor of \( n \) of each other. Use the result of the previous exercise to design a refined approximation scheme that eliminates one factor of \( n \) in the running time of the algorithm.

Exercise 2.3 (10 Points) Consider the two-dimensional knapsack problem as follows: we are given a path \( P \) where every edge \( e \in P \) has the same capacity \( B \in \mathbb{N} \). We are also given \( n \) items where the \( i^{th} \) item has value \( v_i \in \mathbb{N} \) and size \( s_i \in \mathbb{N} \), and in addition, it uses only edges from some subpath \( P_i \subseteq P \). We assume that there is some constant \( \delta \in (0, 1) \) such that the item size \( s_i \) is at least \( \delta \cdot B \) for every \( i \). We want to select a subset of items of maximum total value, such that on each edge \( e \), the total size of the selected items using \( e \) is at most \( B \). Design a dynamic program that computes an optimal solution to this problem in polynomial time. (Hint: Every edge \( e \in P \) can only be used by at most \( 1/\delta \) items in any feasible solution.)

Exercise 2.4 (10 Points) In the uncapacitated facility location problem, we have a set of clients \( D \) and a set of facilities \( F \). For each client \( j \in D \) and facility \( i \in F \), there is a cost \( c_{i,j} \) of assigning client \( j \) to facility \( i \). Furthermore, there is a cost \( f_i \) associated with each facility \( i \in F \). The goal of the problem is to choose a subset of facilities \( F' \subseteq F \) so as to minimize the total cost of the facilities in \( F' \) and the cost of assigning each client \( j \in D \) to the nearest facility in \( F' \). In other words, we wish to find \( F' \) so as to minimize \( \sum_{i \in F'} f_i + \sum_{j \in D} \min_{i \in F'} c_{i,j} \).

a) Give an \( O(\ln |D|) \)-approximation algorithm for this problem.

b) Show that there exists some \( c > 0 \) such that there is no \( (c \ln n) \)-approximation algorithm for this problem unless \( P = NP \).