Problem Set 5

Problem 1
Give a (possibly randomized) algorithm that solves the fractional knapsack problem in expected linear time in the worst case.

Problem 2
Let \( X \subseteq \mathbb{R}^d \) denote a set of \( N \) vectors. You may assume that \( x_i \neq y_i \) for every \( i \in [d] \) and all vectors \( x, y \in X \) with \( x \neq y \). Then a vector \( x \in X \) is Pareto-optimal if and only if there does not exist a vector \( y \in X \) with \( y_i < x_i \) for all \( i \in [n] \). Give an algorithm that computes the set \( P \subseteq X \) of Pareto-optimal vectors in time \( O(N \log N) \) for \( d = 2 \) and \( d = 3 \).

Hint: For \( d = 3 \), first sort the vectors in increasing order according to their first coordinate. Then insert the vectors one after another and compute the sets \( S_1, \ldots, S_N \), where \( S_i \subseteq X \) denotes the subset of the first \( i \) vectors from \( X \) that are Pareto-optimal with respect to the second and third coordinate. How do these sets help you to compute the set of Pareto-optimal vectors?

Problem 3
Give an implementation of the Expanding Core algorithm in Java or C++. Use your implementation to solve instances in which all profits and weights are chosen uniformly at random from \([0, 1]\) for \( n = 100, 200, 300, \ldots\). How does the number of items in the core and the running time depend on \( n \) in your experiments.