HGEN 48800 (2024 spring)
Problem Set #2

Problem 1 (20 points) Consider the knapsack problem: *A thief robbing a store finds n items. The i-th item is worth v_i dollars and weighs w_i pounds. The thief wants to take as valuable a load as possible, but he can carry at most W pounds in his knapsack. Which items should he take?*

(a) (10 points) When the thief can take fractions of items, prove by contradiction that a greedy algorithm gives you the optimal solution. Provide the pseudo code for your algorithm.

(b) (10 points) When the thief has to make a binary choice for each item, we've seen that the greedy algorithm does not guarantee an optimal solution. Instead, a dynamic programming approach can solve the problem. Please give a dynamic programming solution that runs in $O(nW)$ time, and provide the corresponding pseudo code.

Problem 2 (20 points) The *swap sorting* of permutation π is a transformation of π into the identity permutation by exchanges of adjacent elements. For example, $3142 \to 1342 \to 1324 \to 1234$ is a three-step swap sorting of permutation 3124. We would like to use minimum number of swaps to sort a permutation.

(a) (8 points) For any array A, we define inversion as a pair of indices $i < j$ such that $A_i > A_j$. For example, the array $(3, 1, 2)$ has two inversions (31) and (32). Show that the minimum number of swaps needed to sort a permutation π is equal to the number of inversions in π.

(b) (12 points) Use the results above to design an algorithm to find the minimum number of swaps needed to sort a permutation. Also estimate the running time of the algorithm. Try to be as efficient as you can. Implement the algorithm and show that it works with some example input. Hint: think about divide-and-conquer.