

Theory of Computing - Assignment 6

Consider the Knapsack problem.

Input: A set U of n types of items. A weight w_i and value v_i for each item type i , $1 \leq i \leq n$. A capacity B . The w_i , v_i and B are all positive integers.

Goal: Find a collection of items that maximizes the sum of the values with the total price at most B . You can use any item type as many times as needed.

1. Use dynamic programming to solve knapsack in time polynomial in m where $m = B + w_1 + \dots + w_n$.
2. The problem of determining for some k whether there is a collection of value at least k is NP-complete (you don't have to prove this). Why doesn't the NP-completeness of Knapsack contradict part 1?
3. The greedy algorithm does the following: Start with an empty knapsack. Pick an item i that will fit in the knapsack and maximizes v_i/w_i and add item i to the knapsack. Repeat until you can add no more.
Show the greedy algorithm always gets within a factor 2 of the optimal solution.
4. Give an example where the greedy algorithm does no better than a factor 1.99 of the optimal solution.
5. For every k give a polynomial-time algorithm that gives a solution with $(1 + 1/k)$ of the optimal. The polynomial of the running time can depend on k .

Hint: Try each subset of k or fewer items as the initial knapsack, adding as many items as possible to each of these using the greedy procedure. Take the best of the resulting sets as the approximate solution.