Technische Universität München, Zentrum Mathematik
Lehrstuhl für Angewandte Geometrie und Diskrete Mathematik

Combinatorial Optimization (MA 4502)

Prof. Dr. A. Taraz | Dr. M. Ritter

exercise sheet 2

Exercise 2.1 (Dynamic Programming for the Knapsack Problem)
Consider the 0-1-knapsack problem on 4 items given by values $v = (1, 3, 2, 5)^T$, weights $w = (2, 4, 3, 4)^T$ and bound $b = 6$. Compute a solution to the problem by applying the dynamic programming algorithm presented in the lecture. Give a detailed presentation of each step.

Exercise 2.2 (A Different Dynamic Programming Recursion for the Knapsack Problem)
Let $v, w \in \mathbb{N}^n$ and $b \in \mathbb{N}$ be an instance of the 0-1-knapsack problem as defined in the lecture. For $j \in \{1, \dots, n\}$ and $w \in \mathbb{N}_0$, define $V_j(w)$ as the maximum value that can be attained using only items in the set $\{1, \dots, j\}$ subject to the constraint that the total accumulated weight is at most equal to w.

a) Devise a recursion formula for $V_j(w)$.

b) Using your recursion, devise a dynamic programming algorithm for the knapsack problem in analogy to the algorithm discussed in the lecture.

c) Which complexity does your algorithm have? How does it compare to the algorithm from the lecture? Is your algorithm's running time polynomial in the input size of the knapsack problem or not?

d) Try your algorithm for the example given in exercise 2.1.

e) Bonus exercise: Implement your algorithm using your favourite programming language. (If you do not yet have a favourite programming language, we recommend Matlab or Python for this exercise.)

Exercise 2.3 (A Dynamic Programming Algorithm for Shortest Path)
Let $G = (V, A)$ be a directed graph on n nodes and m arcs and let $w : A \to \mathbb{N}$ be a positive weight function on the arcs. The Shortest Path Problem asks for the length (with respect to w) of a shortest path from node 1 to node n. For $k \in \{0, \dots, m\}$ and $j \in V$, define $D_k(j)$ to be the length of a shortest path from node 1 to node j that uses at most k arcs.

a) Show that the following recursion holds:

$$D_{k+1}(j) = \min \left\{ D_k(j), \min_{j' \in V} \{ D_k(j') + w(j', j) : (j', j) \in A \} \right\}$$

b) Design a dynamic programming algorithm based on that recursion that solves Shortest Path. What is the complexity of your algorithm? Is it polynomial?

Please turn over.

Exercise 2.4 (An Approximation Algorithm for the Knapsack Problem)
Consider an instance of the knapsack problem on $n \in \mathbb{N}$ items given by values $v \in \mathbb{N}^n$, weights $w \in \mathbb{N}^n$ and a bound $b \in \mathbb{N}$, where $w_i \leq b$ for all $i \in \{1, \ldots, n\}$. Let z^* denote the value of a maximum knapsack and z_{greedy} the value of a knapsack obtained through the following algorithm:

Input: An integer n, a vector $v \in \mathbb{N}^n$ of values, a vector $w \in \mathbb{N}^n$ of weights and a bound $b \in \mathbb{N}$ such that $w_i \leq b$ for all $i \in \{1, \ldots, n\}$.
Output: A feasible knapsack $I \subset \{1, \ldots, n\}$.
Sort the items such that $\frac{v_1}{w_1} \geq \cdots \geq \frac{v_n}{w_n}$.
Determine $k' := \max\left\{k \in \{1, \ldots, n\} : \sum_{i=1}^{k} w_i \leq b\right\}$.
if $\sum_{i=1}^{k'} v_i > v_{k'+1}$
then
  return $I := \{1, \ldots, k'\}$
else
  return $I := \{k' + 1\}$
end

a) Show that the algorithm is a 2-approximation for the knapsack problem, i. e. $z^* \leq 2 \cdot z_{\text{greedy}}$.

b) Prove that the constant 2 is best possible for this algorithm, i. e. there is no contant $r < 2$ such that $z^* \leq r \cdot z_{\text{greedy}}$ for all possible instances of the knapsack problem.

c) Show that the last step of the algorithm is essential, i. e. prove that by always returning the knapsack $I := \{1, \ldots, k'\}$ the ratio $\frac{z_{\text{greedy}}}{z^*}$ can become arbitrarily small.

This exercise sheet will be discussed in the tutorials on Nov 07 and Nov 08.