4. Übung

1. Doch etwas Wahrscheinlichkeitstheorie:

   (a) Drei Klassiker der WT (so Sie sie nicht kennen, fragen Sie Ihr Lieblings–WT–Buch bzw. Prof. Google) – wiederholen Sie die vorkommenden Begriffe.

      i. Formulieren und beweisen Sie die Ungleichung von Markow.

      ii. Formulieren und beweisen Sie die Tschebycheff'sche Ungleichung (mit der Ungleichung von Markow).

      iii. Formulieren und beweisen Sie das schwache Gesetz der großen Zahlen (mit der Ungleichung von Tschebycheff).

   (b) Berechnen Sie mit der Ungleichung von Tschebyscheff die Wahrscheinlichkeit, dass man bei 1000–maligem Wurf einer fairen Münze mehr als 550 mal Kopf wirft.

2. In einer Währung existieren (unlimitiert viele) Münzen im Wert von $1 = d_1 \leq d_2 \leq \ldots \leq d_k$ Cent. Ziel des W(echsel)–G(eld)–P(roblems) ist es, mit möglichst wenigen Münzen einen gegebenen Betrag von n Cent zu wechseln.
   Z.B.: Existieren 1, 2 und 5 Cent Münzen, so kann man 17 Cent offenbar minimal durch 4 Münzen – $2, 5, 5, 5$ – darstellen.

   (a) Bestätigen Sie, dass das WGP die optimale Teilstruktur–Eigenschaft erfüllt, d.h., zeigen Sie, dass eine optimale Lösung für n Cent, d.h., $n = \sum_i c_i d_i$ und $\sum c_i$ ist minimal, auch eine optimale Lösung b bzw. $n-b$ Cent darstellt, wenn b ein Anteil von n ist, welcher durch die in der Darstellung von n verwendeten Münzen zustandekommt, d.h., $b = \sum_i c'_i d_i$ und $c'_i \leq c_i$.

   (b) Sei $A(n)$ die Anzahl der Münzen, die zum Wechseln des Betrags n mindestens notwendig sind. Überlegen Sie sich–analog zum rod cutting problem–eine Rekursion für $A(n)$.

   (c) Erklären Sie die Funktionsweise des folgenden Algorithmus $Wechsel(d, k, n)$, welcher für den Wert n und für gegebene Münzen(–Array) $d = (d_1, \ldots, d_k)$ die optimale Wechsellösung liefert am Beispiel $(d_1, d_2, d_3) = (1, 2, 5)$ und $n = 8$.
   $Wechsel(d, k, n)$:

   (1) $C(0) = 0$
   (2) for $j = 1$ to n
   (3) $\quad C(j) = \infty$
   (4) $\quad$ for $i = 1$ to k
   (5) $\quad\quad$ if $d_i \leq j$ and $1 + C(j - d_i) < C(j)$ then
   (6) $\quad\quad\quad C(j) = 1 + C(j - d_i)$
   (7) $\quad\quad\quad S(j) = d_i$
   (8) Return C and S

   (d) Wie groß ist (asymptotisch) die Komplexität von $Wechsel(n, d, k)$.

3. (Fortsetzung Beispiel (2)):

   (a) Naheliegend wäre eine Lösung des WGP mittels greedy–Algorithmus. Wie würde ein solcher Algorithmus beim WGP aussehen?

   (b) Angenommen die Menge an zur Verfügung stehenden Münzen wäre $(d_1, d_2, d_3) = (1, 4, 9)$. Finden Sie einen Betrag n, bei welchem der greedy–Algorithmus versagen würde.

   (c) Begründen Sie: Falls $(d_1, d_2, d_3) = (1, 5, 10)$, so liefert ein greedy–Algorithmus eine optimale Lösung.

4. Das Rucksack–Problem: Gegeben sind n–Elemente und zugehörig zwei Arrays $p = (p_1, \ldots, p_n)$ und $w = (w_1, \ldots, w_n)$ (üblicherweise) positiver natürlicher Zahlen, sowie ein $C \in \mathbb{R}$. Typischerweise interpretiert man p_i als den Preis und w_i das Gewicht eines i–ten Gegenstandes.
   Angenommen ein Dieb kann nun maximal C Kilogramm tragen, so ergibt sich die Frage: Welche der n Güter

soll er mitnehmen, um maximale Beute zu machen?
Etwas präzieser: Gesucht ist $f(n, C)$, wenn

$$f(m, C') = \max \left\{ \sum_{i=1}^{m} x_i p_i \;\middle|\; \sum_{i=1}^{m} x_i w_i \leq C' \text{ und } x_i \in \{0, 1\} \right\}$$

für $m \in \{1, \ldots n$ und $C' \in \{0, \ldots, C\}$.

(a) Überlegen Sie sich, dass das Rucksackproblem die optimale Teilproblem Eigenschaft (wie lautet diese hier?) erfüllt.

(b) Geben Sie eine Rekursion für $f(m, C')$, d.h. für optimale Teillösungen des ursprünglichen Problems, an.
Anmerkung: Diese Rekursion hängt nun–anders als beim vorigen Beispiel, von beiden Parametern m und C' ab.

(c) Wie lautet ein Algorithmus (dynamische Programmierung, vgl. oben), welcher das Rucksackproblem löst?

(d) Wie müsste man $p = (p_1, \ldots, p_n)$ und $w = (w_1, \ldots, w_n)$ wählen, damit der Rucksackproblem–Algorithmus die Lösung des WSP aus Beispiel 2 liefert?

5. Gegeben sei ein Array $A[1, \ldots, n]$ von n reellen (nicht notwendig positiven!) Zahlen, gesucht ist

$$M = \max \left\{ \sum_{i=h}^{l} A[i],\ 0 \leq h \leq l \leq n \right\}.$$

(a) Was ist also gesucht?

(b) Geben Sie einen offensichtlichen brute force Algorithmus an, welcher M in $O(n^2)$ Schritten findet.

(c) Dieses Problem lässt sich sogar in $O(n)$ Schritten lösen – betrachten Sie dazu für, $0 \leq k \leq n$,

$$M_k = \max \left\{ \sum_{i=h}^{k} A[i],\ 0 \leq h \leq k \right\},$$

sowie die Rekursion $M_k = \max(M_{k-1} + A[k], A[k])$. Erklären Sie diese Rekursion am Beispiel $A = (1, -2, 3, 4, -1, 2)$ und geben Sie einen Algorithmus an, welcher (auf dieser Rekusion beruhend) M in $O(n)$ Schritten findet.