Übungsblatt 10 für "Diskrete und geometrische Algorithmen"

1.) p Familien mit a_i, $1 \leq i \leq p$, Familienmitgliedern besuchen gemeinsam ein Restaurant, q Tische mit b_j, $1 \leq j \leq q$, Plätzen seien für das Essen reserviert. Im Sinne einer optimalen Durchmischung der einzelnen Familienmitglieder (oder weil sich die jeweiligen Familienmitglieder aus dem Weg gehen wollen), sollen niemals zwei Familienmitglieder der selben Familie am selben Tisch sitzen. Formulieren Sie ein Maximales-Fluss-Problem, welches dieses Problem löst.

2.) Eine aktivste Kante in einem Netzwerk G ist eine Kante, deren Entfernung die größte Abnahme des maximalen Flusses von der Quelle s zur Senke t bewirkt. Sei f nun ein maximaler Fluss von s nach t in G. Zeigen oder widerlegen (durch passende Gegenbeispiele) Sie die folgenden Behauptungen:

(a) Eine aktivste Kante ist eine Kante e mit maximalem Fluss $f(e)$.

(b) Eine aktivste Kante ist eine Kante e mit maximaler Kapazität.

(c) Eine aktivste Kante ist eine Kante e mit maximalem Fluss $f(e)$ unter jenen Kanten, welche zu einem minimalen Schnitt (S, T) gehören.

(d) Eine Kante, welche zu keinem minimalen Schnitt gehört, kann keine aktivste Kante sein.

(e) Ein Netzwerk kann mehrere aktivste Kanten enthalten.

3.) Wie wir wissen, gilt für die Fibonacci-Zahlen F_n die explizite Formel

$$F_n = \frac{\phi^n - \hat{\phi}^n}{\sqrt{5}}, \quad \text{mit} \quad \phi = \frac{1+\sqrt{5}}{2} \quad \text{und} \quad \hat{\phi} = \frac{1-\sqrt{5}}{2}.$$

Man zeige nun, daß der Aufruf $\textsc{Euclid}(a, b)$ für $a > b \geq 1$ höchstens $1 + \log_\phi(b)$ rekursive Aufrufe durchführt. Man zeige weiters, daß man diese Schranke sogar auf $1 + \log_\phi(b/\mathrm{ggT}(a, b))$ verbessern kann.

4.) Was gibt die Prozedur $\textsc{Extended-Euclid}(F_{k+1}, F_k)$ zurück? Beweisen Sie, daß Ihre Antwort korrekt ist.

5.) Erklären Sie die Funktionsweise des FFT-Algorithmus an Hand der Berechnung des Produkts der Polynome $A(x) = 4 - 4x$ und $B(x) = 6 + 2x$.

6.) Eine Toeplitz-Matrix ist eine $n \times n$-Matrix $A = (a_{k,j})_{1 \leq k,j \leq n}$, für die $a_{k,j} = a_{k-1,j-1}$ für $2 \leq k, j, \leq n$ gilt.

(a) Ist die Summe zweier Toeplitz-Matrizen ebenfalls eine Toeplitz-Matrix? Ist das Produkt zweier Toeplitz-Matrizen ebenfalls eine Toeplitz-Matrix?

(b) Beschreiben Sie, wie wir eine Toeplitz-Matrix so darstellen können, daß wir zwei $(n \times n)$-Toeplitz-Matrizen in der Zeit $\mathcal{O}(n)$ addieren können.

(c) Geben Sie einen Algorithmus zur Multiplikation einer $(n \times n)$-Toeplitz-Matrix mit einem Vektor der Länge n an, der eine Laufzeit von $\mathcal{O}(n \log n)$ hat.
Hinweis: Darstellung aus Teil (b) verwenden, die Toeplitz-Matrix in eine obere und untere Dreiecksmatrix aufspalten und die Multiplikationen als Faltungsprodukte anschreiben.

7.) Wir betrachten nun allgemeiner das bereits in der letzten Übung vorgestellte "Wechsel-Geld-Problem" (WGK). In einer Währung existieren (unlimitiert viele) Münzen im Wert von $1 = d_1 < d_2 < \cdots < d_k$ Cent. Ziel des WGK ist es, mit möglichst wenigen Münzen einen gegebenen Betrag von n Cent zu wechseln.

(a) Bestätigen Sie, daß das WGK 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 $\sum_i c_i$ ist minimal, auch eine optimale Lösung für 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 in der Vorlesung besprochenen Stabzerlegungsproblem) eine Rekursion für $A(n)$.

(c) Erklären Sie die Funktionsweise des folgenden Algorithmus Wechsel(d, k, n), welcher für den Wert n und für ein gegebenes 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)
seien C[0..n] und S[0..n] neue Felder
C[0] := 0
FOR j = 1 TO n DO
  C[j] := ∞
  FOR i = 1 TO k DO
    IF d_i ≤ j AND 1 + C[j − d_i] < C[j] THEN
      C[j] := 1 + C[j − d_i]
      S[j] := d_i
    END IF
  END DO
END DO
RETURN C und S
```

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

8.) Das Rucksack-Problem. Sie haben schon wieder gewonnen! Diesmal dürfen Sie sich unter n Gegenständen (die Sie aber nicht zerteilen dürfen), wobei der i-te Gegenstand v_i Euro wert ist und w_i Kilo wiegt, so viele aussuchen und in Ihren Rucksack hineinpacken, so viel Sie tragen können; wir nehmen an, daß Sie höchstens W Kilo tragen können. Wir nehmen dabei weiters immer an, daß v_i, w_i und W positive ganze Zahlen sind. Die Aufgabe beim "Rucksack-Problem" besteht nun darin, anzugeben, welche Gegenstände Sie mitnehmen sollen, sodaß der

Gesamtwert der eingepackten Gegenstände möglichst groß ist.
Etwas genauer: Gesucht ist $f(n, W)$, wenn

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

für $m \in \{1, 2, \ldots, n\}$ und $W' \in \{0, \ldots, W\}$.

(a) Überlegen Sie sich, daß das Rucksack-Problem die optimale Teilstruktur-Eigenschaft (wie lautet diese hier?) besitzt.

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

(c) Man gebe einen Algorithmus an, der unter Zuhilfenahme von dynamischer Programmierung das Rucksack-Problem löst.