Übungsblatt 5 für "Diskrete und geometrische Algorithmen"

33.) Zum Lösen von Rekursionen.

(a) Gegeben sei ein Feld aus $n \times 2$ quadratischen Kacheln. P_n sei die Anzahl der Möglichkeiten, das Feld mit Dominosteinen zu pflastern. Hierbei können Dominosteine waagrecht (2×1) oder senkrecht (1×2) auf das Feld gelegt werden. Bestimmen Sie eine Rekursion für P_n.

(b) Eine wichtige Methode zum Lösen von Rekursionen sind sogenannte erzeugende Funktionen. Dabei ordnet man der Folge $(P_n)_{n \geq 0}$ die Potenzreihe

$$P(z) := \sum_{n \geq 0} P_n z^n$$

zu und obige Rekursion für P_n läßt sich in eine einfache Funktionalgleichung für $P(z)$ "übersetzen", woraus sich die Funktion $P(z)$ leicht bestimmen läßt. Wie lautet $P(z)$? (Um P_n zu bestimmen, müsste man von $P(z)$ die Koeffizienten ablesen, was nach Partialbruchzerlegung leicht gelingt; das ist hier aber nicht mehr verlangt.)

34.) In der Vorlesung wurde der Algorithmus RANDOMIZE-IN-PLACE zum zufälligen Permutieren eines Feldes $A[1..n]$ vorgestellt. Betrachten wir nun den "leicht" abgewandelten Algorithmus, wo das Element $A[i]$ jeweils mit einem aus dem gesamten Feld zufällig gewählten Element vertauscht wird:

```
PERMUTE-WITH-ALL(A)
n := A.länge
FOR i = 1 TO n DO
  vertausche A[i] mit A[RANDOM(1, n)]
END DO
```

Erzeugt dieser Algorithmus ebenfalls eine zufällige Permutation von A? Weshalb oder weshalb nicht?

35.) Noch eine "leichte" Abwandlung des Algorithmus RANDOMIZE-IN-PLACE. Diesmal wird das Element $A[i]$ jeweils mit einem zufällig gewählten Element aus dem Teilfeld $A[i+1..n]$ vertauscht:

```
PERMUTE-WITHOUT-IDENTITY(A)
n := A.länge
FOR i = 1 TO n − 1 DO
  vertausche A[i] mit A[RANDOM(i + 1, n)]
END DO
```

Man könnte zunächst meinen, daß dadurch alle von der Identität verschiedene Permutationen zufällig erzeugt werden. Überlegen Sie sich, daß dies aber nicht der Fall ist. Können Sie herausfinden, welche Klasse von Permutationen mit diesem Algorithmus tatsächlich zufällig erzeugt wird?

36.) Nehmen Sie an, wir wollten eine zufällige Stichprobe der Größe m aus $\{1, 2, \ldots, n\}$ entnehmen, das heißt eine m-elementige Teilmenge S mit $0 \leq m \leq n$, sodaß jede m-elementige Teilmenge mit der gleichen Wahrscheinlichkeit erzeugt wird. Zeigen Sie, daß die folgende Prozedur mit nur m Aufrufen von RANDOM eine zufällige m-elementige Teilmenge S von $\{1, 2, \ldots, n\}$ berechnet, also jede m-elementige Teilmenge mit der gleichen Wahrscheinlichkeit durch die Prozedur erzeugt wird:

```
RANDOM-SAMPLE(m, n)
IF m = 0 THEN
   RETURN ∅
ELSE   S := RANDOM-SAMPLE(m − 1, n − 1)
   i := RANDOM(1, n)
   IF i ∈ S THEN
      S := S ∪ {n}
   ELSE
      S := S ∪ {i}
   END IF
   RETURN S
END IF
```

37.) Das Coupon-Sammel-Problem: Eine Person versucht eine vollständige Sammlung von n verschiedene Coupons zu erhalten. Bei jedem Kauf erhält er zufällig (also jedes Coupon mit Wahrscheinlichkeit $1/n$ und unabhängig von allen anderen Käufen) eines von n Coupons. Man berechne die erwartete Anzahl an Coupons, die diese Person erwerben muß, um eine vollständige Sammlung zu erhalten.
Hinweis: Man teile das Problem in n "Etappen" auf, d.h., man betrachte $\mathbb{E}(X_j)$, für $1 \leq j \leq n$, wobei die Zufallsvariable X_j die Anzahl an Coupons angibt, die man benötigt, um ausgehend von einer Sammlung von $j - 1$ verschiedenen Coupons ein weiteres von diesen verschiedenes Coupon zu erhalten.

38.) In der Vorlesung haben wir die Datenstruktur der (Max-)Heaps kennengelernt, welche z.B. für Prioritätswarteschlangen wichtig sind. Eine Operation, die wir bislang nicht betrachtet haben, ist HEAP-DELETE(A, i), welche das Element in Knoten i aus dem Heap A löscht. Geben Sie eine Implementierung von HEAP-DELETE an, die für einen Heap mit n Elementen die Zeit $O(\log n)$ benötigt.

39.) Erzeugen eines Heaps mittels Einfügens: Wir können einen Heap bauen, indem wir wiederholt die in der Vorlesung kennengelernte Prozedur MAX-HEAP-INSERT aufrufen, um ein Element in den Heap einzufügen. Betrachten Sie die wie folgt geänderte BUILD-MAX-HEAP-Prozedur:

```
BUILD-MAX-HEAP'(A)
A.heap-size := 1
FOR i = 2 TO A.länge DO
  MAX-HEAP-INSERT(A, A[i])
END DO
```

(a) Erzeugen die Prozeduren BUILD-MAX-HEAP und BUILD-MAX-HEAP' immer den selben Heap, wenn sie auf das gleiche Eingabefeld angewendet werden? Beweisen Sie, daß sie dies tun, oder geben Sie ein Gegenbeispiel an.

(b) Zeigen Sie, dass BUILD-MAX-HEAP' im schlechtesten Fall die Laufzeit $\Theta(n \log n)$ benötigt, um einen Heap mit n Elementen zu erzeugen.

40.) Die Höhe eines Knotens v in einem Heap ist gegeben durch die Anzahl der Kanten eines längsten Pfades von v zu einem Blatt. Weiters ist die Höhe des Heaps definiert als die Höhe des Wurzelknotens.

Überlegen Sie sich, dass ein n-elementiger Heap die Höhe $\lfloor \log_2 n \rfloor$ hat und dass es höchstens $\lceil \frac{n}{2^{h+1}} \rceil$ Knoten der Höhe h gibt.