1. (15 Punkte) Es sei S eine Menge von n Stücken, wobei für jedes $s \in S$ eine Zahl w_s gegeben ist. Für $R \subset S$ sei $W_R = \sum_{r \in R} w_r$.

   Finden Sie einen Algorithmus, der bei gegebener Zahl A eine Teilmenge $R \subset S$ mit möglichst wenigen Elementen findet mit $W_R \geq A$.

   Idealerweise sollte Ihr Algorithmus lineare Laufzeit haben, also $\mathcal{O}(n)$ mit $n = |S|$.

2. (15 Punkte)

   (a) Betrachten Sie die in der Vorlesung behandelten Sortiermethoden InsertionSort, SelectionSort und HeapSort.
   Welche dieser Methoden lassen sich leicht als stabile Sortiermethoden realiseren? Skizzieren Sie die jeweilige Realisierung oder erklären Sie, warum die offensichtlichen Realisierungen keine stabile Sortiermethode darstellen.

   (b) Erklären Sie, warum die offensichtliche Realisierung von Quicksort keine stabile Sortiermethode darstellt.

   (c) Zeigen Sie, wie sich ein stabiles Quicksort implementieren lässt, wenn man zusätzlich einen Teil eines zweiten Arrays B gleicher Größe zur Verfügung hat.

3. (15 Punkte) In der Vorlesung haben wir RadixSort kennengelernt. Mit dieser Methode kann man eine Menge A von n Stücken, deren Schlüssel d-Tupel aus $\{0..K-1\}^d$ sind, in Zeit $\mathcal{O}(d(n+K))$ sortieren. Voraussetzung dafür ist, dass man eine *stabile* Sortiermethode zur Verfügung hat, die diese Menge bezüglich einer Schlüsselkomponente (also einer Zahl aus $\{0..K-1\}$ in Zeit $\mathcal{O}(n+K)$ sortieren kann. (CountingSort wäre so eine Methode.)

   Nehmen wir einmal an, stabiles CountingSort stünde nicht zur Verfügung (wegen Patentierung durch die Firma EvilSort, oder aber auch, weil der zusätzlich notwendige Speicherplatz nicht zur Verfügung steht). Sie haben nur Zugriff auf ein Programm UnstableCountingSort, das Mengen der Größe n mit Schlüsseln aus $\{0..K-1\}$ in Zeit $\mathcal{O}(n+K)$ *instabil* sortieren kann.

   Wie können Sie mit Hilfe von UnstableCountingSort eine Menge A mit Schlüsslen aus $\{0..K-1\}^d$ korrekt sortieren?

   Welche Laufzeit (als Funktion von n, K und d) können Sie erreichen?

4. (10 Punkte)

   (a) Beweisen Sie, dass in jedem binären Baum mit N Knoten es ein Blatt mit Tiefe mindestens $\log_2(N+1)-1$ geben muss. Dabei ist die Tiefe eines Blattes die Anzahl der Kanten auf dem Weg zur Wurzel.

   (b) Beweisen Sie, dass die durchschnittliche Tiefe eines Knotens in einem binären Baum mit N Knoten bei mindestens $\log_2(N+1)-c$ liegt, wobei c eine für alle binären Bäume gültige positive Konstante sein soll.
   Hinweis: Betrachten Sie zunächst die Summe aller Knotentiefen und verwenden Sie das Ergebnis des ersten Teils dieser Aufgabe.