Wintersemester 2018/19
Professor Dr. S. Hougardy
Dr. U. Brenner

Forschungsinstitut für Diskrete Mathematik
Universität Bonn

Algorithmische Mathematik I
11. Übung

1. Modifizieren Sie den Algorithmus Mergesort so, dass die gegebene Menge S nicht mehr in zwei, sondern in b (mit $3 \leq b \leq n$) Teilmengen $S_1, \ldots, S_b$ aufgeteilt wird. Diese Teilmengen sollen natürlich möglichst ähnliche Größen haben, d.h. je zwei von ihnen sollen sich in ihren Größen höchstens um 1 unterscheiden. Die Mengen $S_1, \ldots, S_b$ werden dann rekursiv sortiert und die sortierten Teilmengen anschließend zu einer sortierten Gesamtmenge verschmolzen.

   (a) Zeigen Sie, dass das Verschmelzen der Teillösungen in Zeit $O(n \log b)$ möglich ist.

   (b) Führen Sie eine asymptotische Laufzeitanalyse des Verfahrens durch. (3+3 Punkte)

2. Wir wollen Binärstrings (d.h. Wörter über $\{0, 1\}$) lexikographisch sortieren. Wenn s und t zwei Binärstrings der Länge m sind, dann heißt s *lexikographisch kleiner als* t, wenn es einen Index $j \in \{1, \ldots, m\}$ gibt, so dass s und t an den ersten $j-1$ Stellen identisch sind, s an der Stelle j den Wert 0 hat und t an der Stelle j den Wert 1 hat. Zeigen Sie, dass man n Binärstrings der Länge m in Zeit $O(mn)$ (also in linearer Laufzeit) lexikographisch sortieren kann. (5 Punkte)

3. (a) Sei n eine Zweierpotenz. Zeigen Sie, dass Mergesort zum Sortieren von n Elementen maximal $n \log n - n + 1$ Vergleiche benötigt.

   (b) Zeigen Sie, dass es für jedes n, das eine Zweierpotenz ist, eine Instanz mit n Elementen gibt, für die Mergesort die unter (a) angegebene Anzahl Vergleiche benötigt. (2+2 Punkte)

4. Implementieren Sie mergesort, indem Sie die Funktion `sort1` im Programm `sort.cpp` durch eine Funktion `mergesort` ersetzen, die wie folgt aussehen soll:

```
template <typename T, class Compare>
void mergesort(std::vector<T> & v, size_t first, size_t last, const Compare & comp)
```

   Die Zeile 89 von `sort.cpp` ist dann zu ersetzen durch

```
mergesort(dates, 0, dates.size() - 1, comparison);
```

   (5 Punkte)

Weihnachtsaufgaben:

1. Zeigen Sie, dass jeder ungerichtete Graph mit n Knoten und mehr als $\frac{1}{2}n^{\frac{3}{2}}$ Kanten einen Kreis der Länge höchstens 4 besitzt. (5 Bonuspunkte)

2. Es sei S eine Menge mit n Elementen und $\mathcal{A} = \{A_1, \ldots, A_n\}$ eine Menge von paarweise verschiedenen Teilmengen von S. Zeigen Sie, dass es dann ein $x \in S$ geben muss, für das auch die Mengen $A_i \cup \{x\}$ $(i = 1 \ldots, n)$ paarweise verschieden sind. (5 Bonuspunkte)

3. Betrachten Sie folgendes Problem: Gegeben seien zwei Mengen $\{a_1, \ldots, a_n\}$ und $\{b_1, \ldots, b_n\}$ sowie eine durch Schlüssel induzierte partielle Ordnung „$\preceq$“ auf $S := \{a_1, \ldots, a_n\} \cup \{b_1, \ldots, b_n\}$ (die über ein Orakel gegeben ist). Beide Mengen seien in bezug auf „$\preceq$“ bereits sortiert, d.h. es gelte $a_i \preceq a_j$ und $b_i \preceq b_j$ für alle $i, j \in \{1, \ldots, n\}$ mit $i < j$. Gesucht ist eine Sortierung aller Elemente von S in bezug auf „$\preceq$“.

   (a) Wie viele Sortierungen der Menge S gibt es, bei denen für alle $i, j \in \{1, \ldots, n\}$ mit $i < j$ das Element a_i vor a_j und das Element b_i vor b_j steht? Geben Sie eine (von n abhängende) Formel an.

   (b) Zeigen Sie, dass es keinen Algorithmus gibt, der eine Sortierung von S in bezug auf „$\preceq$“ berechnet und stets weniger als $2n - 1$ Vergleiche benötigt. (2+3 Bonuspunkte)

Abgabe: Montag, den 7.1.2019, bis 10:12 Uhr.

Abgabe der Programmierübungen:
Per E-Mail an `alma_prog_gr_XX@dm.uni-bonn.de`, wobei `XX` durch die Nummer Ihrer Übungsgruppe zu ersetzen ist, also z.B. `alma_prog_gr_07@dm.uni-bonn.de`, wenn Sie in Gruppe 7 sind, oder `alma_prog_gr_12@dm.uni-bonn.de`, wenn Sie in Gruppe 12 sind. Wenn Sie Ihre Übungsgruppe nicht kennen, schreiben Sie an `alma_prog_gr_unbekannt@dm.uni-bonn.de`.

Öffnungszeiten des Help Desks:
Montags, 16 – 19 Uhr und freitags, 12 – 15 Uhr, jeweils in Raum N1.002, Endenicher Allee 60, Nebengebäude.
`www.mathematics.uni-bonn.de/files/bachelor/help-desk.pdf`

Zusätzlich gibt es ab sofort einen **Help Desk für Programmierfragen**, und zwar immer freitags, 8 – 10 Uhr und 12 – 16 Uhr, im PC-Pool in der Wegelerstraße 6, Raum E02 (Hochschulrechenzentrum).