Theoretische Informatik I **Blatt 7** **Tutor: Martin Klossek**

Sebastian Steinhorst

Aufgabe 7.1

a) Betrachten wir eine zu sortierende Folge mit der in der Aufgabe beschriebenen Eigenschaft der Elemente, einen maximalen Abstand von d zu ihrer endgültigen Position zu haben. Wie viele Ordnungstypen, also mögliche Anordnungen der Elemente, kann sie Besitzen? Beginnen wir mit dem ersten (linken) Element. Da es keine links zu belegenden Plätze erreichen kann, kann es nur einen der d Plätze rechts neben ihm erreichen, oder es besitzt schon den richtigen Platz. Das macht also $d+1$ mögliche Anordnungen für das erste Element. Das zweite Element kann nun entweder den Platz des ersten Elements, seinen bisherigen oder wiederrum einen der d rechten Plätze belegen, wobei eine Möglichkeit durch den bereits in dieser Umgebung vom ersten Element belegten Platz entfällt. Somit haben wir wieder $d+1$ mögliche Anordnungen für das zweite Element. Dies setzt sich fort, da jeweils für ein folgendes Element bereits d Plätze in seiner erreichbaren Umgebung schon belegt sind. Also haben die ersten $n-d$ Elemente jeweils $d+1$ mögliche Anordnungen. Für die letzten d Elemente kommt als Einschränkung hinzu, dass nicht nur schon d Plätze in der Umgebung belegt sind, sondern auch der rechte Rand der Folge die möglichen Anordnungen weiter einschränkt. Für das d-letzte sind es dann noch d Möglichkeiten, für das $d-1$-letzte noch $d-1$, was beim letzten Element zu nur noch einer möglichen Position führt. Somit gilt für die d letzten, dass es $d!$ Anordnungen gibt. Für die übrigen $n-d$ Elemente gibt es insgesamt $(d+1)^{n-d}$ Anordnungen. Zusammen ist die Anzahl der Ordnungstypen des Entscheidungsbaums:

$$d! \cdot (d+1)^{n-d}$$

Somit ist dessen Tiefe mindestens:

$$log_2\,(d! \cdot (d+1)^{n-d}) = log_2\, d! + (n-d) \cdot log_2(d+1)$$

Die Worst-Case Laufzeit eines vergleichsorientierten Sortieralgorithmus auf einer Folge mit maximalem Abstand d der Elemente von ihrer Endposition ist also:

$$\Omega(log_2\, d! + (n-d) \cdot log_2(d+1))$$

□

Aufgabe 7.2

a) Wir befassen uns zunächst mit der unsortierten Teilfolge der Länge $\frac{n}{\log n}$. Diese sortieren wir mit einem Algorithmus, der für n Elemente die Laufzeit $O(n \cdot \log(n))$ bietet. (z.B. Mergesort)

$\Rightarrow$ Die Teilfolge wird in $O(\frac{n}{\log n} \cdot \log \frac{n}{\log n})$ sortiert.

Nun haben wir zwei sortierte Arrays, die wir mit der Funktion `merge`, die, wie aus der Vorlesung bekannt, in $O(n)$ arbeitet, zu einem sortierten Array verschmelzen können.

$\Rightarrow$ Gesamtlaufzeit: $O(\frac{n}{\log n} \cdot \log \frac{n}{\log n}) + O(n)$

Nun ist noch zu zeigen, dass $O(\frac{n}{\log n} \cdot \log \frac{n}{\log n}) + O(n)$ einer Gesamtlaufzeit von $O(n)$ entspricht:

$$\begin{aligned} \frac{n}{\log n} \cdot \log \frac{n}{\log n} &= \frac{n}{\log n} \cdot (\log n - \log \log n) \\ &= n - n \cdot \frac{\log \log n}{\log n} \\ &= n \cdot \Big[1 - \frac{\log \log n}{\log n}\Big] \end{aligned}$$

Da $1 - \frac{\log \log n}{\log n}$ für alle $n > 1$ einen Wert ≤ 1 annimmt, folgt $\frac{n}{\log n} \cdot \log \frac{n}{\log n} = O(n)$. Damit erhalten wir als Gesamtlaufzeit:

$$O\Big(\frac{n}{\log n} \cdot \log \frac{n}{\log n}\Big) + O(n) = O(n) + O(n) = O(n)$$

□

b) **Gegeben:**

n Verkehrssünder mit Information über ihr Herkunftsbundesland.

Gesucht:

effiziente Möglichkeit, für jedes Bundesland eine Liste mit seinen Verkehrssündern zu erstellen.

Verfahren:

Wir nehmen an, wir hätten folgende Datenstruktur in der Datenbank:

```
Record Suender:
    String Name
    Integer Bundesland
    ...

Array Datenbank[1..n] of Suender
```

wobei es eine eine Zuordnung gibt, die jedem Bundesland nach der Reihe einen bestimmten Wert zwischen 1 und 16 zugewiesen hat. Nun brauchen wir nur noch ein Array `Bundeslandliste[1..16]`, das für jedes der 16 Bundesländer einen Zeiger auf eine Zeigerliste enthält, die den Befehl `insertfront` unterstützt (am Anfang einfügen).

Somit können wir durch das Array gehen und für jeden Datensatz mittels des Bundesland - Index, dessen Zielliste Ansprechen und ihn dort einfügen.

Pseudocode:

```
for i = 1 to n:
    Bundeslandliste[Datenbank[i].Bundesland].insertfront(Datenbank[i])
```

Da wir linear durch die Datenbank gehen und jeder Datensatz einmal addressiert und bewegt wird, haben wir eine Laufzeit von $O(n)$.

□