

Tutorium

Grundlagen: Algorithmen und Datenstrukturen

Übungsblatt Woche 3

Aufgabe 3.1

Gegeben sei eine Zahl $a \in \{0, 1\}$ sowie eine Zahlenfolge (x_1, x_2, \dots, x_k) , wobei $x_i \in \{0, 1\}$ für alle $i \in \{1, \dots, k\}$ gilt, wobei k die Länge der Zahlenfolge ist. Gefragt ist, ob a in der Zahlenfolge vorkommt.

Algorithmus 1 löst diese Aufgabe. Berechnen Sie die erwartete Anzahl der Vergleiche $x_i = a$, die dieser Algorithmus durchführt, wenn jede Eingabe mit den oben beschriebenen Eigenschaften mit derselben Wahrscheinlichkeit auftritt. Bestimmen Sie zudem die asymptotisch erwartete Laufzeit.

```
Input: int a, int[]  $(x_1, x_2, \dots, x_k)$   
1 int  $i = 1$   
2 while  $i \leq k$  do  
3   | if  $x_i = a$  then  
4   |   | return Ja  
5   | end  
6   |  $i = i + 1$   
7 end  
8 return Nein
```

Aufgabe 3.1

Zu bestimmen:

- Erwartete Anzahl an Vergleichen: $x_i = a$
- Erwartete asymptotische Laufzeit

Hinweis: Verwenden Sie, wie in der Vorlesung demonstriert, binäre Zufallsvariablen (sogenannte Indikatorvariablen). Verwenden Sie zudem, dass für eine binäre Zufallsvariable Y mit $\mathbb{P}[Y = 1] = c$ und $\mathbb{P}[Y = 0] = 1 - c$ für ein $c \in [0, 1]$, der Erwartungswert von Y genau c ist, d.h. $\mathbb{E}[Y] = \sum_{y \in \{0,1\}} y \cdot \mathbb{P}[Y = y] = 1 \cdot \mathbb{P}[Y = 1] + 0 \cdot \mathbb{P}[Y = 0] = \mathbb{P}[Y = 1] = c$. Sie können weiter von der Gleichung $\sum_{i=0}^n c^i = \frac{c^{n+1}-1}{c-1}$ für $c \neq 1$ Gebrauch machen.

Aufgabe 3.2

Gegeben sei eine Zahlenfolge $A[0], \dots, A[n-1]$, wobei die Länge n ist, und eine Zahl x aus dieser Folge. Gehen Sie im Folgenden davon aus, dass die Wahrscheinlichkeit von Duplikaten in A vernachlässigbar gering ist.

Betrachten wir den folgenden Algorithmus `count(x, A)`:

```
1  int c = 0;
2  for (int i = 0; i < n; i++)
3      if (A[i] == x) c++;
4  return c;
```

Aufgabe 3.2

- a) Bestimmen Sie die Komplexität von `count` im worst-case, best-case und average-case.
- b) Überlegen Sie sich einen alternativen Algorithmus, der auf sortierten Folgen arbeitet, und bestimmen Sie dessen Komplexität im worst-case, best-case und average-case. Vergleichen Sie diese mit den Ergebnissen aus Aufgabenteil a).
- c) Lässt sich die Komplexität verbessern, indem man die Folge zunächst sortiert?

Aufgabe 3.2

a) Bestimmen Sie die Komplexität von `count` im worst-case, best-case und average-case.

```
1  int c = 0;
2  for (int i = 0; i < n; i++)
3      if (A[i] == x) c++;
4  return c;
```

Aufgabe 3.2

- b) Überlegen Sie sich einen alternativen Algorithmus, der auf sortierten Folgen arbeitet, und bestimmen Sie dessen Komplexität im worst-case, best-case und average-case. Vergleichen Sie diese mit den Ergebnissen aus Aufgabenteil a).

Aufgabe 3.2

Variante 1 (Linear Search):

```
1  int c = 0;
2  for (int i = 0; i < n; i++) {
3      if (A[i] == x) { c++; }
4      else { if (A[i] > x) return c; }
5  }
6  return c;
```


Aufgabe 3.2

Variante 2 (Binary Search):

```
1  int c = 0, l = 0, r = n - 1, m;  
2  while (l <= r) {  
3      m = (l + r) / 2;  
4      if (A[m] == x) {  
5          while (m != -1 && A[m] == x) {  
6              m--; c++;  
7          }  
8          m = (l + r) / 2 + 1;  
9          while (m != A.length && A[m] == x) {  
10             m++; c++;  
11         }  
12         return c;  
13     }  
14     if (A[m] < x) l = m + 1;  
15     else r = m - 1;  
16 }  
17 return c;
```

Aufgabe 3.3

Seien $f, g, h : \mathbb{N}_0 \rightarrow \mathbb{R}$ Funktionen mit $\exists n_0 : \forall n > n_0 : f(n), g(n), h(n) > 0$. Zeigen Sie die folgenden „Transitivitätsregeln“.

$$(a) \quad f(n) \in o(g(n)) \quad \Rightarrow \quad f(n) \in \mathcal{O}(g(n))$$

$$(b) \quad f(n) \in \mathcal{O}(g(n)) \quad \text{und} \quad g(n) \in \mathcal{O}(h(n)) \quad \Rightarrow \quad f(n) \in \mathcal{O}(h(n))$$

$$(c) \quad f(n) \in o(g(n)) \quad \text{und} \quad g(n) \in \mathcal{O}(h(n)) \quad \Rightarrow \quad f(n) \in o(h(n))$$

Aufgabe 3.3

$$(a) \quad f(n) \in o(g(n)) \quad \Rightarrow \quad f(n) \in \mathcal{O}(g(n))$$

Aufgabe 3.3

$$(b) \quad f(n) \in \mathcal{O}(g(n)) \quad \text{und} \quad g(n) \in \mathcal{O}(h(n)) \quad \Rightarrow \quad f(n) \in \mathcal{O}(h(n))$$

Aufgabe 3.3

$$(c) \quad f(n) \in o(g(n)) \quad \text{und} \quad g(n) \in \mathcal{O}(h(n)) \quad \Rightarrow \quad f(n) \in o(h(n))$$