Einsendeaufgabe 2

Stefan Berger

1.

Schreiben Sie einen rekursiven Algorithmus für die binäre Suche Seite 43 im Skript. Siehe auch Übung 3 Übungsaufgaben Datei AlgOnl-Aufg-WS17-18-oL.pdf im Kursmaterial.

Eingabe: $A = (a_1, ..., a_n)$ mit $a_i, i, n \in \mathbb{N}, 1 \leq i \leq n, x \in \mathbb{N}, a_1 < a_2 < ... < a_n$
Ausgabe: $i \in 0, 1, ..., n$ mit $1 \leq i \leq n$ gefunden, $i = 0$ nicht gefunden

```
Binares Suchen(A, x, 1, Länge von A + 1)
function Binares Suchen(A, x, l, r)
    if l ≥ r then
        return 0
    i ← ⌊(l + r)/2⌋
    if A[i] = x then
        return i
    if A[i] < x then
        l ← i + 1
    else
        r ← i
    return Binares Suchen(A, x, l, r)
end function
```

3.

Erzeugen Sie ein int Array mit den Zahlen 0 bis 999 und suchen Sie 12. Wie viele Aufrufe sind notwendig? Gleiche Frage mit Zahlen von 0 bis 99999.

Der Algorithmus benötigt 10 rekursive Aufrufe, um die Zahl 12 in dem Array mit den Zahlen von 0 bis 999 zu finden. Für das Array mit den Zahlen 0 bis 99999 benötigt er 16 rekursive Aufrufe. Der Algorithmus hat eine logarithmische Komplexität ($T(n) = \lfloor \log n + 1 \rfloor$).

4.

Stellen Sie die Rekursionsgleichung zur Bestimmung der Zeitkomplexität Ihres Algorithmus im schlechtesten Fall in Abhängigkeit von der Eingabe n auf.

$$T(n) = \begin{cases} konstant, & falls\ n = 1 \\ T(n/2) + \Theta(1), & falls\ n \geq 2 \end{cases}$$

5.

Lösen Sie die Rekursionsgleichung mit Hilfe des Master-Theorems und geben sie die Zeit- Komplexität in asymptotischer Notation. Entspricht das Ergebnis Ihrer Antworten der Frage 3?

$b = 1, c = 2, k = 0$

$b(1/c)^k = (1/2)^0 = 1$

$T(n) \in \Theta(\log n)$