Weihnachtsaufgabe

Gegeben sei folgender Algorithmus. Er findet (für eine Zweierpotenz n) in einer n-elementigen Menge von gleichwertigen Elementen ein einzelnes mit höherem Wert.

Die Werte seinen in einem Array $A[1...n]$ gegeben für ein $n = 2^p$. Für die Eingabe sei garantiert, dass es ein k gibt mit $A[i] < A[k]$ für alle $i \neq k$ und $A[i] = A[j]$ für alle $i, j \neq k$.

Die Methode `weight`$(A[i, j])$ gibt dabei in $\mathcal{O}(1)$ die Summe aller Werte zwischen den Stellen i und j aus.

Algorithm 1: SockSearch

```
Aufruf: sockSearch(A, 1, n)
sockSearch(A, l, r) begin
    while l ≠ r do
        m ← (l+r−1)/2;
        if weight(A[l, m]) < weight(A[m+1, r]) then
            l ← m + 1;
        else
            r ← m;
    return "gefunden an Stelle l";
```

1. Geben Sie seine Laufzeit in $\mathcal{O}$-Notation an. Begründen Sie!
   - Bei jedem Schleifendurchlauf wird betrachteter Bereich genau halbiert (dadurch $\mathcal{O}(\log n)$ Schleifendurchläufe).
   - In der Schleife werden nur Operationen mit konstanter Laufzeit durchgeführt,
   - das gilt auch für den Aufruf der Methode `weight`.
   - Laufzeit ist also in $\mathcal{O}(\log n)$.
2. Beweisen Sie die Korrektheit des Algorithmus mit Hilfe der Verifikation nach Floyd.
   - INV: $l \leq k \leq r$ vor jedem Schleifendurchlauf
   - In die Schleife hinein: Keine Operationen $\Rightarrow$ k ist im Suchbereich (ganzes Array)
   - (IndAnfang) $i = 1$: Suchbereich ist ganzes Array, k also zwischen l und r
   - (IndSchritt) $i - 1 \Rightarrow i$: In Schleifendurchlauf $i - 1$ wird der Suchbereich auf die schwerere Hälfte begrenzt, damit gilt immer noch $l \leq k \leq r \Rightarrow$ INV gilt nach Schleifendurchlauf $i - 1$ und damit auch vor i
   - Im letzten Schleifendurchlauf (INV gilt bis VOR letzten SL) besteht der Suchbereich nur noch aus 2 Elementen. l bzw. r wird auf das schwerere Element gesetzt $\Rightarrow$ Ausgabe $l = r = k$
   - Da der Suchbereich immer (um eine natÃ$\frac{1}{4}$rliche Zahl) kleiner wird $\Rightarrow$ terminiert.