Ludwig-Maximilians-Universität München
Institut für Informatik
Prof. Dr. Thomas Seidl
Anna Beer, Florian Richter

München, 03.05.2018

Algorithmen und Datenstrukturen

SS 2018

Übungsblatt Global 3: Komplexität II

Aufgabe Global 3-1 *Knobelei: Josephus Problem*

Der jüdische Historiker Flavius Josephus hielt sich 67 n.Chr. beim Kampf um die galiläische Stadt Jotapata mit 40 weiteren Männern in einer Höhle vor den Römern versteckt. Als das Versteck verraten wurde, beschlossen diese, lieber zu sterben, als den Römern in die Hände zu fallen. Da Josephus lieber das ihm angebotene freie Geleit der Römer nutzen wollte, falls er die Höhle verlassen würde, schlug er einen kollektiven Suizid vor: Alle sollten sich im Kreis aufstellen und jeder seinen linken Nachbarn töten. Das sollte wiederholt werden, bis nur noch einer übrig bleibt.

An welche Stelle muss sich Josephus stellen, um zu überleben? Stellen Sie eine Funktion $f(n)$ auf, die für eine gegebene Anzahl n von Menschen in dem Kreis die Position des Überlebenden ausgibt.

Lösungsvorschlag:

In der ersten Runde werden die Elemente 2, 4, 6, .. entfernt. Bei gerader Anfangszahl n war das Element aus der $t+1$-ten Runde x in der t-ten Runde an der Stelle $2x-1$. $f(n)$ beschreibt die Position, die bei n Elementen am Ende übrig bleibt. Für gerade n lässt sich n als $2*j$ darstellen, für ungerade n kann man $n = 2*j+1$ schreiben. Es gilt also für die Positionen

$$f(2j) = 2f(j) - 1$$

bei geradem n. Bei ungeradem n ergibt sich

$$f(2j+1) = 2f(j) + 1$$

Betrachtet man die Werte die sich so rekursiv ergeben mit $f(1) = 1$ (gibt es nur einen, bleibt der natürlich als letzter übrig):

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
$f(n)$	1	1	3	1	3	5	7	1	3	5	7	9	11	13	15	1

Wir stellen fest, dass $f(n)$ bei Potenzen von 2 den Wert 1 annimmt und dann immer um zwei hochzählt. Wählt man die Darstellung $n = 2^m + l$ die n in eine Zweierpotenz 2^m und den Rest l zerlegt, kann man f schreiben als $f(n) = 2*l+1$, da $f(n)$ offensichtlich nur von dem Rest l abhängt. Der Beweis per Induktion, dass somit die beiden oben aufgestellten Rekursionsgleichungen erfüllt sind, verbleibt als Übungsaufgabe für besonders interessierte Studierende.

Bei den oben genannten 41 Leuten muss sich Josephus also an die Stelle $f(41) = f(2^5+9) = 2*9+1 = 19$ stellen, um zu überleben.

Implementierung der Funktion in $O(1)$:

```
int josephus(int n){
        m=floor(log n);
        l=n-2^m;
        j=2*l+1;
        return j;
        }
```

Implementierung der rekursiven Gleichungen in $O(log(n))$:

```
int josephus(int n){
        if(n==1) return 1;
        if((n%2)==0) return 2*josephus(n/2)-1;
        if((n%2)==1) return 2*josephus((n-1)/2)+1;
        }
```

Aufgabe Global 3-2 *Multivariate Komplexität*

Univariate Definition (siehe Vorlesung):

$$O(f) = \{g : \mathbb{R} \to \mathbb{R} | \exists c > 0\ \exists x_0 > 0\ \forall x \geq x_0 : |g(x)| \leq c \cdot |f(x)|\}$$

Eine mögliche mutlivariate Definition : Seien f und g Funktionen definiert auf einer Teilmenge von $\mathbb{R}^n$

$$O(f) = \{g : \mathbb{R}^n \to \mathbb{R} | \exists c > 0\ \exists \vec{x_0}\ mit\ \forall i \leq n\ x_i > 0\ so\ dass\ \forall \vec{x} : \forall i \leq n\ x_i \geq x_{0,i} \Rightarrow |g(\vec{x})| \leq c|f(\vec{x})|\}$$

Welche der folgenden Aussagen ist korrekt? Begründen Sie Ihre Antworten.

- $2^{(2^n)} \in O(n^{(2^n)})$

Lösungsvorschlag:
Wahr, da $2^{(2^n)} = 2 * 2 * 2 * ... * 2 \leq n * n * n * ... * n = n^{(2^n)}$

- Wenn $f(n) \in O(s(n))$ und $g(n) \in O(r(n))$ gilt, dann gilt auch $f(n) - g(n) \in O(s(n) - r(n))$

Lösungsvorschlag:
Falsch. Wähle $f(n) = 4n, g(n) = 3n, r(n) = n, s(n) = n$. Dann gilt: $f(n) - g(n) = 4n - 3n = n = O(n)$, aber $O(s(n) - r(n)) = O(n - n) = O(0)$

- $m^3n^2 \in O(m^2n^3)$

Lösungsvorschlag:
Falsch. Angenommen es wäre wahr. Dann würde nach Definition gelten:$\exists c > 0\ \exists \vec{x_0}\ mit\ \forall i \leq n\ x_i > 0\ so\ dass\ \forall \vec{x} : \forall i \leq n\ x_i \geq x_{0,i} \Rightarrow |g(\vec{x})| \leq c|f(\vec{x})|$
Wähle z.B. $\vec{x} = \begin{pmatrix} m_0 * n_0 * c \\ n_0 \end{pmatrix}$. Für dieses gilt zwar $\forall \vec{x} : \forall i \leq n\ x_i \geq x_{0,i}$, allerdings
$|g(\vec{x})| = |(m_0 * n_0 * c)^3 * n_0^2| = |m_0^3 * n_0^5 * c^3| > c * |m_0^2 * n_0^5 * c^2| = c * |f(\vec{x})|$, also $|g(\vec{x})| \not\leq c * |f(\vec{x})|$
Daraus folgt $m^3n^2 \notin O(m^2n^3)$

- $log(m) * n \in O(m * n)$

Lösungsvorschlag:
Wahr. Für $\log_b(m)$ wähle $\vec{x_0} = \begin{pmatrix} b \\ 0 \end{pmatrix}$ und $c = 1$.

- $log(log(m)^n) \in O(n * log(m))$

Lösungsvorschlag:
Wahr. $log(log(m)^n) = n * log(log(m)) \in O(n * log(m))$

- $4mn \in O(n^3)$

Lösungsvorschlag:
Falsch. Wähle z.B. $\vec{x} = \begin{pmatrix} m_0 * n_0^2 * c \\ n_0 \end{pmatrix}$, restliche Begründun analog zu c).

Aufgabe Global 3-3 *Multivariate Komplexität Anwendung*

- Welche Komplexität hat ein naiver Algorithmus, der zwei Listen unterschiedlicher Länge auf gleiche Elemente überprüft?

Lösungsvorschlag:
Angenommen die Listen haben Längen m bzw n. Naiv kann man einfach jedes Element der einen Liste mit jedem Element der anderen vergleichen. Also $O(n * m)$

- Welche Komplexität kann man durch einen effizienteren Algorithmus erreichen?

Lösungsvorschlag:
Man sortiert zunächst beide Listen separat voneinander beispielsweise mit Merge Sort (siehe nächste VL), das geht in $O(n * \log n) + O(m * \log m)$. Danach reicht es die Listen "abwechselnd" durchzugehen, also $O(n + m)$. Gesamt ergibt sich so $O(n * \log n) + O(m * \log m) + O(n + m) \subsetneq O(n * m)$. Alternativ falls $l = max(m, n)$: $O(l * \log l)$.