Fakultät für Mathematik und Informatik

Projekt 14: Chomsky-Grammatik zur Wachstumsmodellierung auf Basis einer dreistufigen Fibonacci-Folge

Modul Molekulare Algorithmen - Sommersemester 2022

Maria Schreiber, Natalie Söllner

18. Juli 2022

1 Einleitung

DNA ist eines der robustesten und kompaktesten Speichermedien, die wir bis heute kennen, mit einer Halbwertszeit weit höher, als moderne Festplatten [1]. Deshalb weckt sie wohl auch das Interesse der Informatik, jedoch nicht nur als Datenspeicher, sondern auch als Computer. Mit diesem Konzept befasst sich das sogenannte DNA-Computing. Algorithmische Berechnungen sollen mithilfe verschiedener Operationen auf DNA realisiert werden. Es findet Anwendung bei kombinatorischen Problemen, numerischen Berechnungen und anderen Fragestellungen, die einen hohen Speicherbedarf haben (DNA: $1\frac{bit}{nm^3}$) oder enorme Parallelität (DNA: $1,2 * 10^{18}$ Operationen pro Sekunde) erfordern [1, 2, 3, 4]. Vorteilhaft ist auch, dass unter den richtigen Bedingungen DNA ein persistenter, redundanter, dezentraler und verlustreicherer Informationsspeicher *in-vivo* oder *in-vitro* darstellt [4].
Der Startpunkt für DNA-Computing ist die Lösung des Hamiltonpfadproblems durch DNA (1994) und findet bis heute Anwendung beispielsweise in Filtering-Modellen, Watson-Crick-Automaten *in-vitro*, Gene-Assembly- und Membrane-Computing-Modellen *in-vivo* verwendet [1, 3]. Viele dieser DNA-Computer lassen sich mit einer formal-abstrakten Beschreibung als eine sogenannte Chomsky-Grammatik implementieren [2].
Ziel dieses Projektes ist es, eine solche Chomsky-Grammatik zu entwickeln, die Wörter einer festen Länge erzeugt. Die Länge der akzeptierten Wörter wird durch eine modifizierte, dreigliedrige Fibonacci-Folge bestimmt (sieh Kapitel 2).
Benannt ist die Folge nach Leonardo Fibonacci (2), der damit Anfang des 13. Jahrhunderts versuchte, das Wachstum einer Kaninchenpopulation zu beschreiben [5, 6]. Auch heute noch weckt die Fibonacci-Zahl aufgrund ihrer Eigenschaften Interesse. Beispielsweise ergeben die Differenz oder die Summe zweier Fibonacci-Zahlen eine Fibonacci-Zahl. Auch bekannt ist die Zahlenfolge dafür, dass das Verhältnis aufeinander folgender Summenglieder gegen den Goldenen Schnitt konvergiert. Der größte gemeinsame Teiler zweier Fibonacci-Zahlen ist ebenfalls eine Fibonacci-Zahl [7].
Die Rekursionsformel der (unmodifizierten) Fibonacci Folge ist definiert:

$$f(n) = \begin{cases} 0 & n = 0 \\ 1 & n = 1,\ n = 2 \\ f(n-1) + f(n-2) & n \geq 3 \end{cases} \tag{1}$$

2 Aufgabenstellung und Grundlagen

2.1 Aufgabe

Gegeben sei eine modifizierte Fibonacci-Folge (2) mit folgenden Startwerten und Berechnungsvorschrift:

$$f(n) = \begin{cases} 0 & n = 0,\ n = 1 \\ 1 & n = 2 \\ f(n-1) + f(n-2) + f(n-3) & n \geq 3 \end{cases} \tag{2}$$

- Entwickeln Sie eine Chomsky-Grammatik G, mit der alle Glieder von $f(n)$ sukzessive als Wörter von $L(G)$ erzeugt werden.
- Eine Zahl $n \in \mathbb{N}$ soll durch das Wort $a \dots a$ (n - mal) kodiert werden.
- Erläutern Sie die algorithmische Idee und die Arbeitsweise Ihrer Grammatik.
- Geben Sie den Ableitungsbaum für die ersten 8 Glieder an.
- Wie viele Regelanwendungen sind nötig, um zu einem gegebenen n den Wert $f(n)$ zu berechnen (Komplexitätsabschätzung)?

2.2 Formale Sprachen und Grammatiken

Eine Chomsky-Grammatik ist ein 4-Tupel $G = (V, \Sigma, P, S)$, wobei:

- V ... endliche Menge von Symbolen - die Nicht-Terminale
- Σ ... endliche Menge von Symbolen - die Terminale mit $V \cap \Sigma = \emptyset$
- P ... endliche Menge von Produktionsregeln $P \subset ((V \cup \Sigma)^* \otimes V \otimes (V \cup \Sigma)^* \times (V \cup \Sigma))$ [1]
- S ... Startsymbol mit $S \in V$

Eine Chomsky-Grammatik erzeugt regelbasiert, fortlaufenden und startend mit dem Startsymbol S durch Teilwortersetzung jedes Wort einer festen Länge der formalen Sprache $L(G) = \{x \in \Sigma^* \mid S \vdash_G^* x\}$ mit $\vdash_G$ als ein Ableitungsschritt [2]. [2, 4].

3 Lösungsansatz

3.1 Vorüberlegungen und Ideen

Die Schwierigkeit dieser Aufgabe bestand vor allem darin, eine rekursive Vorschrift zu finden, die zur Wortverlängerung dient, ohne dabei die Komplexität zu stark anwachsen zu lassen. Tabelle 1 listet die ersten 19 Glieder der Folge. Das Ergebnis $f(n)$ ist die Summe aus den drei vorherigen Ergebnissen ab $n \geq 3$ neben der Initialisierung für $n = 0, 1, 2$ (siehe Abbildung 2, Formel (2)).
Das $f(n)$ korrespondiert zu Wortlänge, also die Anzahl der a's, der formalen Sprache. Damit ist $L(G) = \{0, a, aa, aaaa, aaaaaaa, ...\}$.

Im Folgenden werden einige Lösungsansätze diskutiert, die später aufgrund hoher Anzahl an Regeln und somit einer hohen Komplexität verworfen wurden.

n	$f(n)$	n	$f(n)$	n	$f(n)$	n	$f(n)$
0	0	5	4	10	81	15	1705
1	0	6	7	11	149	16	3136
2	1	7	13	12	274	17	5768
3	1	8	24	13	504	18	10609
4	2	9	44	14	927	19	19513

Abbildung 1: Die ersten 19 Glieder der Folge.

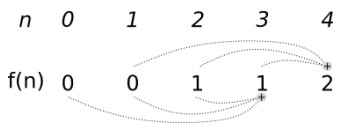

Abbildung 2: Darstellung der rekursiven Arbeitsweise.

Ein Ansatz war das Trennen und Schieben der einzelnen Teilworte (im Folgenden w_1, w_2, w_3, getrennt mit \$). Grundidee bestand darin, rekursiv die Buchstaben (a's) aus w_1, w_2 und w_3 ganz an das Wortende zu kopieren, welches das nächste Summenglied ergeben hätte. Im Anschluss würde das aktuelle w_1 gelöscht werden. Das neue Summenglied wird das neue w_3, das aktuelle w_3 wird das neue w_2 und das aktuelle w_2 wird das neue w_1. Das Schieben der einzelnen Buchstaben konnte jedoch nicht effizient umgesetzt werden und wurde verworfen.

Ein anderer Ansatz war der Versuch, eine Formel zu finden, die dieselben Zahlen berechnet, aber nur zwei Summenglieder hat, wie $f(n) = 2 \cdot f(n-1) - f(n-4)$. Der Ansatz wurde nicht weiter verfolgt, da die Grammatik sich $f(n-4)$ merken muss und dies rekursiv nicht so einfach realisierbar war.

[1] $(V \cup V)^* \otimes V \otimes (V \cup \Sigma) =_{def} \{w_1 v w_2 \mid w_1, w_2 \in (V \cup \Sigma)^* \text{ und } v \in V\}$
[2] Relation $\vdash_G \subset ((V \cup \Sigma)^*) \times ((V \cup \Sigma)^*)$ mit $x \vdash_G y = \{(x, y) \mid x = x_1 u x_2 \wedge y = x_1 v x_2 \wedge \exists x_1, x_2 \in (V \cup \Sigma)^* \cdot ((u, v) \in P)\}$ und (u, v) als eine Grammatikregel

3.2 Lösung

3.2.1 Grammatik

Chomsky-Grammatik $G = (V, \Sigma, P, S)$ für modifizierte Fibonacci-Folge:

$$V = \{S, L, R, X, Y, Z, A, B, C\} \qquad \Sigma = \{a\} \qquad S = S$$

$$\begin{aligned}
P = \{ & \\
& S \to \epsilon \mid a \mid aa \mid LABCXR && (i) && \textit{Initialisierung} \\
& CX \to XB && && \\
& BX \to XA && && \\
& AX \to XABC && (ii) && \textit{X-Phase} \\
& LX \to LY && (iii) && \textit{Reset} \\
& YA \to AY && && \\
& YB \to BY && && \\
& YC \to CY && (iv) && \textit{Y-Phase} \\
& YR \to XR \mid Z && (v) && \textit{Erweitern} \mid \textit{Beenden} \\
& AZ \to Za && && \\
& BZ \to Za && && \\
& CZ \to Z && && \\
& LZ \to \epsilon && (vi) && \textit{Z-Phase} \\
& \}
\end{aligned}$$

3.2.2 Funktionsweise

Der Lösungsansatz für die Grammatik beruht darauf, die modifizierte Fibonacci-Formel (2) Schritt für Schritt aufzulösen, wobei in jedem Schritt das erste Glied jeweils ersetzt wird durch seine Vorgänger, beispielsweise wird $f(n-1)$ ersetzt durch $f(n-2) + f(n-3) + f(n-4)$:

$$\begin{aligned}
& f(n) \\
&= 1 \cdot f(n-1) + 1 \cdot f(n-2) + 1 \cdot f(n-3) \\
&= 2 \cdot f(n-2) + 2 \cdot f(n-3) + 1 \cdot f(n-4) \\
&= 4 \cdot f(n-3) + 3 \cdot f(n-4) + 2 \cdot f(n-5) \\
&= 7 \cdot f(n-4) + 6 \cdot f(n-5) + 4 \cdot f(n-6) \\
&= 13 \cdot f(n-5) + 11 \cdot f(n-6) + 7 \cdot f(n-7) \\
&= \dots
\end{aligned} \tag{3}$$

Die drei neu eingesetzten Summenglieder werden zu den noch vorhandenen zwei Gliedern addiert, sodass sich neue Faktoren für alle drei Summenglieder ergeben und nun die Indices der Summenglieder um eins verringert wurden.
Dieser rekursive Abstieg kann von einem n bis zu den gewählten Startwerten

$$A = f(3) = 1 = a, \quad B = f(2) = 1 = a, \quad C = f(1) = 0 = \epsilon$$

fortgeführt werden, sodass entsprechende Faktoren vor den Startwerten die Anzahl des jeweiligen Startwertes für das aktuelle $f(n)$ darstellen. Dasselbe Prinzip wird für die Chomsky-Grammatik für die modifizierte Fibonacci-Folge verwendet, nur dass statt eines schrittweisen Abstieges ein Aufbau stattfindet.

Für die ersten Glieder der Folge $f(0)$ bis $f(4)$ - also $0, 0, 1, 1, 2$ - ist der Regelblock (i) zuständig, der die Worte direkt erzeugt (Initialisierung). Für größere $f(n)$ mit $n \geq 5$ wird die Regel $S \to LABCXR$ aus (i) verwendet. Der Ausdruck $LABCXR$ bildet die Grundlage für die Verlängerung der Worte, wobei die Nicht-Terminale L und R als Begrenzer des Wortes dienen (linker und rechter Rand). Die Grammatik arbeitet in Zyklen, wobei jeder Zyklus das nächstgrößere $f(n)$ bildet.
Nun kommen die Nicht-Terminale X, Y, Z ins Spiel. Diese bestimmen die "Phase"der Grammatik. Der Aufbau erfolgt, wenn X im Wort vorhanden ist. Diese Phase wird X-Phase genannt. X durchläuft das Wort von rechts nach links mit den Ersetzungsregeln des Regelblocks (ii) für die Nicht-Terminale A, B, C. Jedes C im Wort, wird durch ein B ersetzt, jedes B im Wort durch ein A und jedes A durch ein ABC, ebenso wie im Beispiel der Aufspaltung $f(n-1)$ durch $f(n-2)+f(n-3)+f(n-4)$ ersetzt wurde, $f(n-2)$ durch $f(n-3)$ und $f(n-3)$ durch $f(n-4)$. Also ist A analog zum ersten, B analog zum zweiten und C zum dritten Summenglied.
Ist X nun am linken Rand des Wortes angelangt, so geht die Grammatik in den "Reset", indem X durch Y ersetzt wird (Regel (iii)). In der Y-Phase wird ausschließlich das Y wieder zum rechten Rand des Wortes geschoben mittels des Regelblocks (iv).
Ist Y nun am rechten Rand angelangt, kann mit Regel $YR \to XR$ $((v))$ wieder in die X-Phase und damit den weiteren Wortaufbau für das nächste Summenglied - $f(n+1)$ - übergegangen, oder mit $YR \to Z$ $((v))$ die Zyklen zur Wortverlängerung beendet und das Ausgabewort gebildet werden. Wird das Wort nicht mehr verlängert, wird das R entfernt.
In der Z-Phase läuft das Z vom rechten Rand des Wortes zum linken Rand und ersetzt dabei jedes A, B, C durch ihren zugeordneten Startwert mit Regelblock (vi). A's und B's werden jeweils durch ein a ersetzt und C's werden gelöscht, da sie den Wert 0 repräsentieren.
Ist das Z am linken Wortrand angelangt, wird es gemeinsam mit dem L entfernt, sodass nur noch Terminale im Wort vorhanden sind. Die Anzahl der a's im resultierenden Wort entspricht einem $f(n)$ der modifizierten Fibonacci-Folge. Je größer die Wörter, umso mehr Zyklen müssen durchlaufen werden. Für $f(5)$ bedarf es nur einen, für $f(6)$ zwei Zyklen, usw.

Erklärung der Funktionsweise an einem Beispiel
Um die Funktionsweise der Grammatik zu verdeutlichen, erklären wir sie am Beispiel $f(5) = 4$ (Abbildung 3). Wir starten mit der Initialisierung in (i) mit der Regel $S \to LABCXR$. Dadurch wird in die X-Phase gewechselt ((ii) für Wortverlängerung). Nacheinander werden folgende Regeln angewendet:

$CX \to XB$, sodass $LABC\,\mathrm{X}\,R \to LAB\,\mathrm{X}\,BR$
$BX \to XA$, sodass $LAB\,\mathrm{X}\,BR \to LA\,\mathrm{X}\,ABR$
$AX \to XABC$, sodass $LA\,\mathrm{X}\,ABR \to L\,\mathrm{X}\,ABCABR$

Die Wortverlängerung ist mit diesem Schritt beendet und X befindet sich am linken Wortrand. Die Regel $LX \to LY$ (Regel (iii)), sodass $LXABCABR \to LYABCABR$, setzt die Grammatik in die Y-Phase, in der das Nicht-Terminal Y an den rechten Wortrand geschoben wird, mit folgenden Regelanwendungen aus Block (iv):

$YA \to AY$, sodass $L\,\mathrm{Y}\,ABCABR \to LA\,\mathrm{Y}\,BCABR$
$YB \to BY$, sodass $LA\,\mathrm{Y}\,BCABR \to LAB\,\mathrm{Y}\,CABR$
$YC \to CY$, sodass $LAB\,\mathrm{Y}\,CABR \to LABC\,\mathrm{Y}\,ABR$
$YA \to AY$, sodass $LABC\,\mathrm{Y}\,ABR \to LABCA\,\mathrm{Y}\,BR$
$YB \to BY$, sodass $LABCA\,\mathrm{Y}\,BR \to LABCAB\,\mathrm{Y}\,R$

Damit ist die Y-Phase abgeschlossen, nun kann mit Regel (v) das Wort entweder erneut in die X-Phase gesetzt werden (Regel $YR \to XR$) oder in die Z-Phase, welche die Nicht-Terminale durch Terminale ersetzt und damit die Wortbildung abschließt. Für unser Beispiel ist die Wortlänge erreicht und wir gehen in die Z-Phase über mit $YR \to Z$, sodass $LABCABYR \to LABCABZ$. Folgende Regelanwendungen sind notwendig, um das Ausgabewort zu erzeugen:

$BZ \to Za$, sodass $LABCAB\,\mathrm{Z} \to LABCA\,\mathrm{Z}\,a$
$AZ \to Za$, sodass $LABCA\,\mathrm{Z}\,a \to LABC\,\mathrm{Z}\,aa$
$CZ \to Z$, sodass $LABC\,\mathrm{Z}\,aa \to LAB\,\mathrm{Z}\,aa$
$BZ \to ZA$, sodass $LAB\,\mathrm{Z}\,aa \to LA\,\mathrm{Z}\,aaa$
$AZ \to Za$, sodass $LA\,\mathrm{Z}\,aaa \to L\,\mathrm{Z}\,aaaa$

und Regel (vi): $LZ \to \epsilon$, sodass $LZaaaa \to$ aaaa = f(5) .

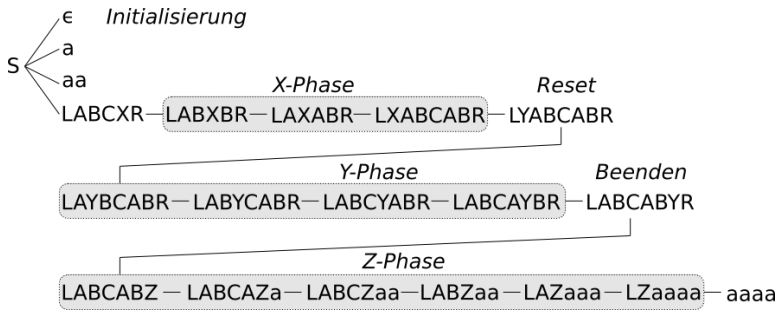

Abbildung 3: Ableitungsbaum bis zum 5. Glied.

Ein weiteres Beispiel der Anwendung der Regeln ist in Abbildung 4 für $n = 8$ zu sehen.

Abbildung 4: Ableitungsbaum bis zum 8. Glied

3.2.3 Korrektheit

Um die Korrektheit einer Grammatik zu zeigen, müssen wir zeigen, dass 1. nur Wörter der Sprache $L(G) = \{\epsilon, a, aa, aaaa, aaaaaaa, ...\}$ erzeugt werden können und 2. alle Wörter der Sprache erzeugt werden können.

Beweis.
Nur Wörter der Sprache werden erzeugt
Für $f(0)$ bis $f(4)$, also $0, 0, 1, 1, 2$ erzeugt die Grammatik entsprechende Worte mit $S \to \epsilon \mid a \mid aa$: das leere Wort für $f(0) = f(1) = 0$, a für $f(2) = f(3) = 1$ und aa für $f(4) = 2$.

Für alle $n \geq 5$ gelten folgende Aussagen:
Mit der Regel $S \to LABCXR$ geht die Grammatik in die **X-Phase** über. Diese Phase dient der Verlängerung des Wortes unter Benutzung des Regelblocks (ii). Genauer, die Anzahl der A's, B's und C's wird so erhöht, dass von $f(n)$ auf $f(n+1)$ inkrementiert wird. Sehen wir uns dafür erneut die Faktoren aus 3 an, dieses Mal aufsteigend:

$$\begin{array}{rlll} & \#A & \#B & \#C \\ f(4) = & 1 \cdot f(3) + & 1 \cdot f(2) + & 1 \cdot f(1) \\ f(5) = & 2 \cdot f(3) + & 2 \cdot f(2) + & 1 \cdot f(1) \\ f(6) = & 4 \cdot f(3) + & 3 \cdot f(2) + & 2 \cdot f(1) \\ f(7) = & 7 \cdot f(3) + & 6 \cdot f(2) + & 4 \cdot f(1) \\ f(8) = & 13 \cdot f(3) + & 11 \cdot f(2) + & 7 \cdot f(1) \end{array}$$

Da $C = 0$, gilt für $f(n+3)$:

$$\#A_n \widehat{=} f(n+2) \tag{4}$$
$$\#B_n \widehat{=} f(n+1) + f(n) \tag{5}$$

also $\#A_n + \#B_n = f(n+3)$. Verbinden wir das mit den Regeln aus (ii):

$$\#A_{n+1} = \#A_n + \#B_n \qquad (A \to ABC, B \to A) \tag{6}$$
$$\#B_{n+1} = \#A_n + \#C_n \qquad (A \to ABC, C \to B) \tag{7}$$
$$\#C_{n+1} = \#A_n \qquad (A \to ABC) \tag{8}$$

Wir beweisen über vollständige Induktion:

Beweis.
Induktionsanfang:

$n = 1$: $\#A_1 = 1 = f(3)$
$n = 2$: $\#A_2 = 2 = f(4)$
$n = 2$: $\#A_3 = 4 = f(5)$

Induktionsbehauptung: $\#A_{n+1} = f(n+3)$
Induktionsvoraussetzung: $\#A_n = f(n+2)$
Induktionsbeweis:

$$\begin{array}{lll} \#A_{n+1} = \#A_n + \#B_n & & (6) \\ \quad = f(n+2) + \#A_{n-1} + \#C_{n-1} & & (4, 7) \\ \quad = f(n+2) + f(n+1) + \#A_{n-2} & & (4, 8) \\ \quad = f(n+2) + f(n+1) + f(n) & & (4) \\ \quad = f(n+3) & & \end{array}$$

□

Es bleibt noch zu zeigen, dass $\#B_n = f(n+1) + f(n)$:

Beweis.

$$\begin{aligned} \#B_{n+1} &= \#A_n + \#C_n && (7) \\ &= f(n+2) + \#A_{n-1} && (4, 8) \\ &= f(n+2) + f(n+1) && (4) \end{aligned}$$

□

Der Übergang zur Y-Phase kann ausschließlich dann stattfinden, wenn X am linken Begrenzer L angekommen ist ($LX \to LY$) und somit alle A's, B's und C's abgearbeitet hat. Es existiert keine Regel, die es möglich macht, von der X-Phase in die Z-Phase überzugehen.
In der **Y-Phase** kann das Wort nicht verlängert werden. Es kann ausschließlich das Y vom linken zum rechten Wortrand geschoben werden. Es gibt keine Regel, die das Y vorzeitig in ein X oder ein Z umwandeln kann, sodass keine Verlängerung oder Terminalersetzung mitten im Wort stattfinden kann. Die Y-Phase hat also keinen Einfluss auf die Wortlänge und muss beendet sein, um in eine andere Phase übergehen zu können.
Der Übergang zur Z-Phase kann ausschließlich dann stattfinden, wenn Y am rechten Wortrand angekommen ist ($YR \to Z$). Die Regel sorgt außerdem dafür, dass das Nicht-Terminal R entfernt wird.
Die **Z-Phase** beendet die Wortbildung. Sie ersetzt ausschließlich von rechts nach links Nicht-Terminale durch Terminale und kann nicht die Richtung wechseln. Diese Phase hat damit keinen Einfluss auf die Anzahl der A's, B's und C's. Wir wissen, dass in der X-Phase genauso viele A's, B's und C' erzeugt werden, dass sie den Faktoren für $f(1)$ (für C), $f(2)$ (für B) und $f(3)$ (für A) entsprechen, die das jeweilige $f(n)$ erzeugen. In der Z-Phase wird immer genau ein A, B oder C durch genau ein entsprechendes a, a oder ϵ ersetzt. Zum Schluss werden durch $LZ \to \epsilon$ die letzten beiden Nicht-Terminale aus dem Wort entfernt, sodass es nur noch aus a's besteht. Das heißt, am Ende hat ein Wort eine Anzahl an a's, die exakt einem $f(n)$ nach der modifizierten Fibonacci Folge (2) entspricht.

Alle Wörter der Sprache werden erzeugt
Die Werte $f(0)$ bis $f(4)$ werden durch die Regeln $S \to \epsilon \mid a \mid aa$ aus (i) initialisiert.
Wir haben schon gezeigt, dass nur Wörter der Sprache erzeugt werden können. Wir wissen außerdem, dass mit jeder X-Phase $f(n)$ auf $f(n+1)$ inkrementiert wird. Das heißt, dass alle Wörter der Sprache $L(G)$ erzeugt werden können, da die Regel $YR \to XR$ aus (v) beliebig oft angewendet werden kann, also beliebig oft die X-Phase zur Wortverlängerung für $n = 5, \ldots \infty$ stattfinden kann.

□

4 Komplexitätsanalyse

Für $f(0)$ bis $f(4)$ benötigt die Grammatik einen Ableitungsschritt, da die Worte ϵ, a, aa direkt durch Regelblock (i) erzeugt werden können. Daher gilt, dass diese Worte in $O(1)$ erzeugt werden.
Für $f(n)$ mit $n \geq 5$ muss in jedem Zyklus zuerst in der X-Phase das Wort durchlaufen werden. Dabei wird es verlängert, deshalb muss in der darauf folgenden Y-Phase ein wiederum längeres Wort durchlaufen werden, mit jeweils einer Regelanwendung pro A, B, C in beiden Phasen. Die Wortlänge in der X-Phase beträgt $f(n) - f(n-2)$ für das aktuelle n. In der Y-Phase beträgt die Wortlänge insgesamt $f(n) + f(n-2)$ für das aktuelle n. Zwischendurch muss jeweils eine Regel angewendet werden, die den Phasenwechsel angibt, also von X zu Y und Y zu X ((iii) und (v)).
Für die X- und Y-Phase gilt also folgende Formel:

$$
\begin{aligned}
&\sum_{h=5}^{n} (f(h) - f(h-2) + 1) + (f(h) + f(h-2) + 1) \\
=&\sum_{h=5}^{n} f(h) + f(h) + f(h-2) - f(h-2) + 2 \\
=&\sum_{h=5}^{n} 2 + \sum_{h=5}^{n} 2 \cdot f(h) \\
=&2 \cdot (n-4) + 2 \cdot \sum_{h=5}^{n} f(h)
\end{aligned}
$$

Für die Z-Phase werden noch zusätzlich für die Umwandlung aller Nicht-Terminale in Terminale sowie die einmalige Regelanwendung $LZ \to \epsilon$ insgesamt $n + 1$ Schritte gebraucht.

Da aber eine Summenformel, deren Summenglied selbst eine rekursive Folge darstellt, schwer aufzulösen ist, nutzen wir einen anderen Ansatz zur Ermittlung der Laufzeit. Die unmodifizierte Fibonacci-Folge (1) wächst exponentiell [6]. Ebenso tut es auch die modifizierte Folge (2), die wir betrachtet haben. Schauen wir uns dafür die Anzahl der Schritte für die ersten Folgenglieder ab $f(5)$ und deren Verhältnis an:

$$
\begin{array}{rll}
8 & \cdot 1.75 & = 14 \\
14 & \cdot 1.\overline{857142} & = 26 \\
26 & \cdot 1.\overline{846153} & = 48 \\
48 & \cdot 1.8\overline{3} & = 88 \\
88 & \cdot 1.84\overline{09} & = 162 \\
162 & \cdot 1.\overline{839506172} & = 298
\end{array}
$$

Wir sehen, dass die Werte der Faktoren um etwa 1.84 liegen. Für die Fibonacci-Zahlen (1) wurde eine Anzahl an Rekursionsaufrufen von etwa $\frac{1+\sqrt{5}}{2}^n \approx 1.62$, was dem Wert des Goldenen Schnittes entspricht, bewiesen [6].

Wir suchen also eine Konstante x, um die Anzahl der Rekursionsaufrufe für unsere modifizierte Fibonacci-Folge abzuschätzen, statt die Summenformel aufzulösen.
Wir sehen uns deshalb folgende Gleichung an:

$$x^n = x^{n-1} + x^{n-2} + x^{n-3}$$

Stellen wir sie nun um:

$$
\begin{aligned}
x^n &= x^{n-1} + x^{n-2} + x^{n-3} && \mid \div x^{n-3} \\
x^3 &= x^2 + x + 1 && \mid -x^2 - x - 1 \\
x^3 - x^2 - x - 1 &= 0
\end{aligned}
$$

Diese Gleichung wollen wir nun nach x auflösen, um unsere Konstante zu finden. Bei Gleichungssystemen dritten Grades kann man die Cardanische Formel zur Ermittlung der Nullstellen benutzen, da das Nullstellen-Raten hier nicht funktioniert und somit auch keine Polynomdivision möglich ist.
Um die Cardanische Formel anwenden zu können, müssen wir die allgemeine Gleichung dritten Grades reduzieren.

$$Ax^3 + Bx^2 + Cx + D = 0 \tag{9}$$

mit den Vorfaktoren $A = 1$ und $B = C = D = -1$ soll in die Form

$$x^3 + ax^2 + bx + c = 0 \tag{10}$$

gebracht werden, wobei:

$$\begin{aligned} a &= \frac{B}{A} = \frac{-1}{1} = -1 \\ b &= \frac{C}{A} = \frac{-1}{1} = -1 \\ c &= \frac{D}{A} = \frac{-1}{1} = -1 \end{aligned}$$

Wir sehen $a = B$, $b = C$ und $c = D$. Deshalb verändert sich an unserer ursprünglichen Gleichung nichts. Anschließend wird das quadratische Glied durch die Substitution $x = z - \frac{a}{3} = z + \frac{1}{3}$ beseitigt und wir erhalten die reduzierte Form:

$$z^3 + pz + q = 0$$

mit

$$p = b - \frac{a^2}{3} = -1 - \frac{-1^2}{3} = -1 - \frac{1}{3} = -\frac{4}{3}$$

und

$$q = \frac{2a^3}{27} - \frac{ab}{3} + c = \frac{2(-1)^3}{27} - \frac{(-1)(-1)}{3} - 1 = \frac{-2}{27} - \frac{1}{3} - 1 = \frac{-2}{27} - \frac{9}{27} - \frac{27}{27} = -\frac{38}{27}$$

Also:

$$z^3 - \frac{4}{3}z - \frac{38}{27} = 0$$

Um z zu ermitteln, müssen wir eine Fallunterscheidung nach D und gegebenenfalls p machen. In unserem Fall ist $D = -1$, also $D < 0$, somit müssen wir p nicht weiter betrachten. Für diesen Fall ergeben sich folgende Lösungen für z:

$$\begin{aligned} z_1 &= \sqrt{-\frac{4p}{3}} \cdot \cos\left[\frac{1}{3} \cdot \arccos\left(-\frac{q}{2} \cdot \sqrt{-\frac{27}{p^3}}\right)\right] \\ z_2 &= -\sqrt{-\frac{4p}{3}} \cdot \cos\left[\frac{1}{3} \cdot \arccos\left(-\frac{q}{2} \cdot \sqrt{-\frac{27}{p^3}}\right) + \frac{\pi}{3}\right] \\ z_3 &= -\sqrt{-\frac{4p}{3}} \cdot \cos\left[\frac{1}{3} \cdot \arccos\left(-\frac{q}{2} \cdot \sqrt{-\frac{27}{p^3}}\right) - \frac{\pi}{3}\right] \end{aligned}$$

Wenn wir p und q eingesetzt haben, ergibt das

$$\begin{aligned} z_1 &= \sqrt{\frac{16}{9}} \cdot \cos\left[\frac{1}{3} \cdot \arccos\left(\frac{19}{27} \cdot \sqrt{\frac{729}{64}}\right)\right] && \approx -1.506 \\ z_2 &= -\sqrt{\frac{16}{9}} \cdot \cos\left[\frac{1}{3} \cdot \arccos\left(\frac{19}{27} \cdot \sqrt{\frac{729}{64}}\right) + \frac{\pi}{3}\right] && \approx -0.753 + 0.606i \\ z_3 &= -\sqrt{\frac{16}{9}} \cdot \cos\left[\frac{1}{3} \cdot \arccos\left(\frac{19}{27} \cdot \sqrt{\frac{729}{64}}\right) - \frac{\pi}{3}\right] && \approx -0.753 - 0.606i \end{aligned}$$

Die Rechenwege sparen wir uns hier. Jetzt müssen wir $z_{1,2,3}$ verwenden, um auf unsere $x_{1,2,3}$ zu kommen, mit folgender Formel:

$$x_i = z_i - \frac{a}{3}$$

Wir können aber x_2 und x_3 vernachlässigen, da wir nach einer reellen Lösung suchen und sowohl z_2, als auch z_3 komplexe Zahlen sind. Wir berechnen also $x_1 = z_1 - \frac{a}{3}$:

$$x_1 = \sqrt{\frac{16}{9}} \cdot \cos\left[\frac{1}{3} \cdot \arccos\left(\frac{19}{27} \cdot \sqrt{\frac{729}{64}}\right)\right] - \frac{-1}{3} \approx 1.8393$$

Dieser Wert entspricht etwa den Faktoren, die wir oben beispielhaft von $f(5)$ bis $f(10)$ berechnet haben. Der geschätzte Wert war 1.84. Somit haben wir also eine Laufzeit von $O(1.8393^n)$ für alle $n \in \mathbb{N}$.

Literatur

[1] Ezziane. DNA computing: applications and challenges. *Nanotechnology*, 17(2):R27, 2005.

[2] Hinze und Sturm. Eine DNA-Arithmetik auf der Basis von Chomsky-Grammatiken. 14:71–75, 2004.

[3] Gaßner und Zerjatke. DNA Computing. Ernst-Moritz-Arndt-Universität, Institut für Mathematik und Informatik, WS 2009 / 2010.

[4] Hinze und Sturm. Rechnen mit DNA - Eine Einfürung in Theorie und Praxis. 2009. Kapitel 4.1.4 Eigenschaften von DNA-Strängen.

[5] Vorobiev. *Fibonacci numbers*. Springer Science & Business Media, 2002.

[6] Sinha. The Fibonacci numbers and its amazing applications. *International Journal of Engineering Science Invention*, 6(9):7–14, 2017.

[7] Kalman and Mena. The Fibonacci numbers—exposed. *Mathematics magazine*, 76(3):167–181, 2003.