Friedrich Schiller Universität Jena
Fakultät für Mathematik und Informatik

Eine Projektabgabe im Modul

Molekulare Algorithmen 2020

Projekt 14: Chomsky-Grammatik zur Wachstumsmodellierung auf Basis einer Fibonacci-Folge mit verstärkter Reproduktion

von Kirsten Göbel und Laura Ölsner

Betreuer
Dr.-Ing. habil. Thomas Hinze

7. Juli 2020

1 Einleitung

Das Future Computing ist eine Sammlung unkonventioneller Konzepte, das aufgrund möglicher Minimierung des Ressourcenverbrauchs[1] immer mehr an Wichtigkeit gewinnt. Mechanismen der Natur, die aufgrund evolutionärer Optimierung meist sehr effizient sind, werden gesucht, um sie als mögliche Rechenoperationen zu nutzen[1]. Dabei scheint das Molecular Computing (DNA-, RNA-, und Protein-Computing) das größte Potenzial zu haben, denn z.B. DNA weist eine massive Datenparallelilität und Speicherkapazität auf. Für das DNA-Computing wird die DNA als Vorbild eines Datenträgers betrachtet und molekularbiologische Techniken werden in Rechenoperationen umgewandelt[1].
Eine mathematische Grundlage im Bereich des DNA-Computing sind die formalen Sprachen. Sie sind ein wichtiges Mittel zur Beschreibung von Berechnungsergebnissen und spielen ein Rolle bei der Klassifikation von Berechnungsstärken[1]. Formale Sprachen können mittels Grammatiken beschrieben werden, z.B. Chomsky-Grammatiken.
Durch Chomsky-Grammatiken kann man verschiedene Sprachen generieren, unter anderem lässt sich die Fibonacci-Folge darstellen.
Die Fibonacci-Zahlen sind nach einem Mathematiker des Mittelalters benannt: Leonardo da Pisa. Dieser beschrieb mittels der bekannten Rekursionsformel das Wachstum einer Kaninchenpopulation. Die Fibonacci-Zahlen stehen in enger Verbindung zum goldenen Schnitt, dem euklidischen Algorithmus und dem exponentiellem Wachstum[2].

Diese Projektarbeit beschäftigt sich mit der Entwicklung einer Chomsky-Grammatik für eine abgeänderte Fibonacci-Folge, die durch verstärkte Reproduktion gekennzeichnet ist:

$$f(n) = \begin{cases} 0 & \text{für } n = 0 \\ 1 & \text{für } n = 1 \\ f(n-1) + 2 \cdot f(n-2) & \text{für } n \geq 2 \end{cases}$$

Die entwickelte Chomsky-Grammatik G, die die Sprache $L(G) = \{a^x \mid \exists n\ f(n) = x\}$ erzeugt, wird im Rahmen dieser Projektarbeit bewiesen und es wird eine Komplexitätsabschätzung vorgenommen.

[1] Hinze, Thomas, and Monika Sturm. Rechnen mit DNA: eine Einführung in Theorie und Praxis. Walter de Gruyter, 2009.

[2] Seite „Leonardo Fibonacci“. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 1. Juli 2020, 10:33 UTC. URL: `https://de.wikipedia.org/w/index.php?title=Leonardo_Fibonacci&oldid=201468930` (Abgerufen: 2. Juli 2020, 11:50 UTC)

2 Grammatiken

Grammatiken sind mathematische Modelle zur Beschreibung formaler Sprachen. Ausgehend von einer Startvariable wird durch das Anwenden von Regeln – also Ableitungsschritten – ein Wort der Sprache generiert. Es gibt die sogenannten Chomsky-Grammatiken, diese dienen zur Beschreibung von rekursiv aufzählbaren Sprachen, der allgemeinsten Klasse formaler Sprachen.
Eine Chomsky-Grammatik G ist definiert als ein 4er-Tupel $G = (V,\ \Sigma,\ R,\ S)$ mit

- V, der Menge and Nichtterminalen, auch Variablenmenge genannt,
- Σ, dem Alphabet der Terminalsymbole,
- R, der nichtleeren Menge an Regeln und
- S $(\in V)$, dem Startsymbol.

Sei $G = (V,\ \Sigma,\ R,\ S)$ eine Chomsky-Grammatik. $x, y \in (V \cup \Sigma)^*$. Ein Ableitungsschritt von x nach y ist eine Relation, definiert durch:
$x \vdash_G y = (x, y) | x = x_1 u x_2 \wedge y = x_1 v x_2 \wedge \exists x_1, x_2 \in (V \cup \Sigma)^*.((u, v) \in P)$.

2.1 Die Grammatik für die modifizierte Fibonacci-Folge

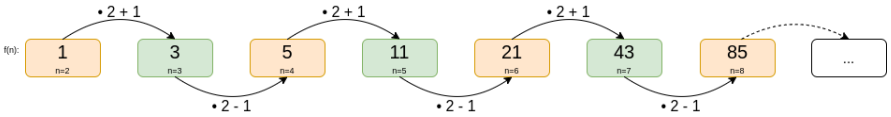

Abbildung 1: Regelmäßigkeit der Ergebnisse von $f(n)$. Das Ergebnis von $f(n)$ entspricht dem Doppelten des Ergebnisses für $f(n-1) \pm 1$, je nachdem, ob n gerade oder ungerade ist.

Bei der Untersuchung der vorgegebenen Rekursion ist eine Regelmäßigkeit der Ergebnisse aufgefallen. Es ist zu beobachten, dass das Ergebnis von $f(n)$ für ein n zwei mal das Ergebnis von $f(n-1)$ $+1$ beziehungsweise -1 ist, abhängig davon, ob n gerade oder ungerade ist. Dies ist auch in der Abbildung 1 zu sehen.
Basierend auf dieser Regelmäßigkeit lässt sich die gegebene Fibonacci-Folge ebenfalls mit der Rekursionsvorschrift

$$g(n) = \begin{cases} 0 & \text{für } n = 0 \\ 1 & \text{für } n = 1, 2 \\ g(n) = 2 \cdot f(n-1) - 1 & \text{für } n = 2m,\ n \geq 3 \\ g(n) = 2 \cdot f(n-1) + 1 & \text{für } n = 2m+1,\ n \geq 3 \end{cases}$$

beschreiben. Basierend auf dieser Rekursionsformel ist folgende Chomsky-Grammatik G mit $G = (V,\ \Sigma,\ P,\ S)$ entstanden:

N = {S, A, X, Y, G, U},
T = {a},
S = S und

$$
\begin{aligned}
\text{R} = \quad \{ & \\
& S \longrightarrow \epsilon \mid a \mid LXAAU \\
& AXU \longrightarrow aa \mid AAYG \\
& AAXG \longrightarrow a \mid AYU \\
& XA \longrightarrow AX \\
& AY \longrightarrow YAA \\
& LY \longrightarrow LX \\
& La \longrightarrow a \\
& Aa \longrightarrow aa \\
\} &
\end{aligned}
$$

Für $n = 0, 1, 2$ müssen keine weiteren Regeln angewendet werden, diese Wörter können direkt aus dem Startsymbol durch einen Ableitungsschritt erzeugt werden. Wenn komplexere Wörter generiert werden sollen, werden die Buchstaben L und G/U eingeführt, um den linken bzw. rechten Rand zu begrenzen. Diese Begrenzungen dienen also als Stoppzeichen für die Laufvariablen X und Y.
G und U dienen nicht nur als rechter Rand, sie dienen auch als Markierung für das ± 1, das je nach n nötig ist. Dabei symbolisiert G den Abzug eines As bei geraden n und U das Hinzufügen eines As bei ungeraden n.
Die bereits angesprochene Laufvariable Y dient dazu, die Verdopplung der As vorzunehmen, um von bspw. $f(n-1)$ auf $f(n)$ zu kommen. Das X hingegen ermöglicht den Abschluss des Wortes, oder aber den Wechsel von bspw. $f(n-1)$ zu $f(n)$.
Die As sind endscheidend für die Wortlänge. Sie werden zum Schluss in die terminalen as übersetzt, alle anderen nicht-terminalen Variablen werden entfernt, ohne Einfluss auf die Länge des Wortes zu haben.

Zur Verdeutlichung wird die Arbeitsweise im folgenden an einem Beispiel ($f(4)$) gezeigt. Eine Auflistung der angewendeten Regeln ist auf der nächsten Seite rechts zu sehen.
Zuerst wird die Regel $S \rightarrow LXAAU$ angewendet. Anschließend ist es nur möglich, die Regeln $XA \rightarrow AX$ zwei mal anzuwenden, sodass das Wort $LAAXU$ entsteht. Da die endgültige Länge für $f(4) = 5$ noch nicht erreicht ist, wird nun $AXU \rightarrow AAYG$ für das Fortsetzen, bzw. den Wechsel, angewendet. Nun wird für jedes A in $LAAAYG$ die Regel $AY \rightarrow YAA$ angewendet, sodass die Anzahl der As verdoppelt wird. Es ergibt sich also

das Wort $LYAAAAAAG$. Durch eine Anwendung von $LY \rightarrow LX$ und sechs Anwendungen von $XA \rightarrow AX$ erhält man $LAAAAAAXG$.

Wie zu Anfang angesprochen, ist das $n = 4$ gerade, womit von den vorhandenen As nun eins abgezogen werden muss, dies geschieht durch die Anwendung von $AAXG \rightarrow a$. Anschließend wird $Aa \rightarrow aa$ angewendet und schlussendlich wird das Wort durch $La \rightarrow aa$ beendet, das Ergebnis lautet $aaaaa$.

$f(4) = 5$

$$
\begin{array}{lll}
S & \longrightarrow & LXAAU \\
LXAAU & \xrightarrow{2 \cdot XA \rightarrow AX} & LAAXU \\
LAAXU & \xrightarrow{AXU \rightarrow AAYG} & LAAAYG \\
LAAAYG & \xrightarrow{3 \cdot AY \rightarrow YAA} & LYAAAAAAG \\
LYAAAAAAG & \xrightarrow{LX \rightarrow LY} & LXAAAAAAG \\
LXAAAAAAG & \xrightarrow{6 \cdot XA \rightarrow AX} & LAAAAAAAXG \\
LAAAAAAAXG & \xrightarrow{AAXG \rightarrow a} & LAAAAa \\
LAAAAa & \xrightarrow{4 \cdot Aa \rightarrow aa} & Laaaaa \\
Laaaaa & \xrightarrow{La \rightarrow a} & aaaaa
\end{array}
$$

Ein expliziter Beweis mit spezifischerer Erklärung folgt in Abschnitt 3.

2.2 Ableitungsbaum

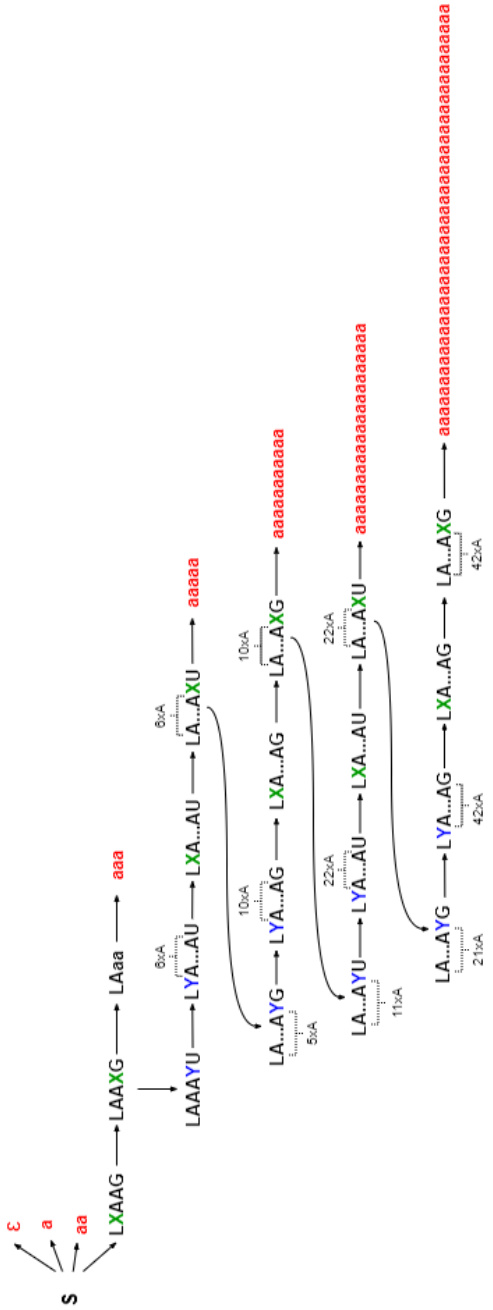

Abbildung 2: Dieses Diagramm zeigt den Ableitungsbaum für die ersten acht Glieder. Jede Zeile steht für ein $f(n)$, bis in der letzten Zeile $f(n) = f(8) = 43$ erreicht wird. Repetitive Regelanwendungen sind zusammengefasst, sodass eine kompaktere Darstellung möglich ist.

3 Korrektheit

Wie in Unterabschnitt 2.1 beschrieben, ist die Grundlage der Grammatik G die Rekursionsvorschrift

$$g(n) = \begin{cases} 0 & \text{für } n = 0 \\ 1 & \text{für } n = 1, 2 \\ g(n) = 2 \cdot f(n-1) - 1 & \text{für } n = 2m,\ n \geq 3 \\ g(n) = 2 \cdot f(n-1) + 1 & \text{für } n = 2m+1,\ n \geq 3 \end{cases}$$

.
Es gilt nun zu zeigen (Induktionsbehauptung), dass

$$f(m) = f(m-1) + 2 \cdot f(m-2) \overset{!}{=} g(m) = \begin{cases} 2 \cdot g(m-1) - 1 & m = 2n \\ 2 \cdot g(m-1) + 1 & m = 2n+1 \end{cases}$$

Wir zeigen die Korrkeitheit dieser Aussage für $m = 2n$ mittels vollständiger Induktion, der Beweis für $m = 2n + 1$ ist analog.
Für den Induktionsanfang wird $m = 2$ gewählt, so ergibt sich für $f(m)$

$$f(2) = f(1) + 2 \cdot f(0) = 1$$

und für $g(m)$ ergibt sich folglich

$$g(2) = 2 \cdot g(1) - 1 = 1.$$

Damit ist gezeigt, dass die Behauptung für $m = 2$ gilt.
Die Induktionsvoraussetzung ist daher $\forall m : m = 2n$:

$$\begin{aligned} & f(m) = g(m) \\ \Leftrightarrow\ & f(m-1) + 2 \cdot f(m-2) = 2 \cdot g(m-1) - 1 \end{aligned}$$

Im folgenden Induktionsschritt werden, aufgrund der zu zeigenden Gleichheit, die Funktionen $f(m)$ und $g(m)$ synonym und austauschbar verwendet.
Für den Induktionsschritt ist nun zu zeigen, dass die Induktionsbehauptung für beliebige m

gilt, also $m \rightsquigarrow m+2$:

$$\begin{aligned} f(m+2) &= f(m+2-1) + 2 \cdot f(m+2-2) \\ &= f(m+1) + 2 \cdot f(m) \\ &= f(m+1) + 2 \cdot (f(m-1) + 2 \cdot f(m-2)) \\ &\overset{\text{IV}}{=} f(m+1) + 2 \cdot (2 \cdot g(m-1) - 1) \\ &= f(m+1) + 4 \cdot g(m-1) - 2 \end{aligned}$$

Um zu zeigen, dass $g(m+2) = f(m+2)$, muss nun gezeigt werden, dass

$$\begin{aligned} g(m+2) &\overset{!}{=} f(m+1) + 4 \cdot g(m-1) - 2 \\ \Leftrightarrow 2 \cdot g(m+1) - 1 &= f(m+1) + 4 \cdot g(m-1) - 2 \end{aligned}$$

Auf der rechten Seite der Gleichung wird nun $g(m-1)$ durch $f(m-1)$ ersetzt, daraus folgt:

$$\Leftrightarrow g(m+1) + 1 = 4 \cdot f(m-1)$$

Hierfür wird nun die Induktionsbehauptung genutzt. Der Term $g(m+1)$ soll mithilfe der Formel $f(m+1) = g(m+1) = f(m) + 2 \cdot f(m-1)$ in $4 \cdot f(m-1)$ überführt werden. Die beiden Terme werden also gleichgesetzt, um die Induktionsbehauptung zu beweisen.

$$\begin{aligned} \Leftrightarrow g(m+1) + 1 &\overset{!}{=} \overbrace{f(m) + 2 \cdot f(m-1) + 1}^{\overset{!}{=} 4 \cdot f(m-1)} \\ \Rightarrow 4 \cdot f(m-1) &= f(m) + 2 \cdot f(m-1) + 1 \\ \Leftrightarrow 2 \cdot f(m-1) &= f(m) + 1 \\ \Rightarrow 2 \cdot f(m-1) - 1 &= f(m) \\ \overset{\text{IV}}{\Leftrightarrow} 2 \cdot g(m-1) - 1 &= g(m) \end{aligned}$$

q.e.d.

Durch die Umformung zu $2 \cdot f(m-1) - 1 = f(m)$, die äquivalent zu $2 \cdot g(m-1) - 1 = g(m)$ ist, ist gezeigt, dass die Induktionsbehauptung auch für $m \rightsquigarrow m+2$ $(\forall m : m = 2n)$ gilt.

Da nun gezeigt ist, dass $f(n) = g(n)$, gilt es, die Korrektheit der Grammatik zu zeigen.

Um die Korrektheit einer Sprache zu beweisen, ist zu zeigen, dass eine Grammatik G jedes Wort der Sprache $L(G)$ generieren kann, aber auch nur die Wörter der Sprache $L(G)$.
Wie bereits erklärt, haben $U,\ G,\ X,\ Y,\ L$ keinen Einfluss auf die Länge des Wortes, diese wird ausschließlich durch die Anzahl der As definiert. Das bedeutet, dass auch die Regeln $XA \rightarrow AX$, $LY \rightarrow LX$, $La \rightarrow a$ und $Aa \rightarrow aa$ zu vernachlässigen sind.
Zur Sprache $L(G)$ gehören die Wörter a^1, a^3, a^5, a^{11}, a^{21}, ..., also die Wörter $a^{f(n)}$. Das Wort a ist durch den Ableitungsschritt $S \rightarrow a$ zur Sprache gehörend. Die folgenden Wörter werden wie folgt generiert:
Um ein Wort $a^{f(n)}$ zu generieren, muss ausgehend von $a^{f(3)}$ für $n-3$ Zwischenschritte die Regel $AY \rightarrow YAA$ mehrmals angewendet werden, entsprechend der Wortlängen für $f(4)$, $f(5)$, ... Da die Anzahl der Iterationen eine Rolle spielt (n gerade: von der Verdopplung muss ein A abgezogen werden; n ungerade: an die Verdopplung muss ein A angehängt werden), werden die Regeln $AAXG \rightarrow AYU$ (1) und $AXU \rightarrow AAYG$ (2) benötigt. Diese sorgen für die nötige Addition (2), beziehungsweise Subtraktion (1). Man kann die Regel (1) für ein n auch als $2(n-1)-1$ verstehen und durch das Weglassen der hier unwichtigen Variablen erhält man vereinfacht $AA \rightarrow A$. Die Addition ist analog. Es ist also gezeigt, dass die Grammatik G alle Wörter der Sprache $L(G)$ generieren kann.

Nun ist zu zeigen, dass G ausschließlich Wörter der Sprache $L(G)$ erzeugt. In diesem Fall sind die nicht-terminalen Symbole X und Y wichtig, die nicht gleichzeitig existieren können (durch (1) und (2) wird X in Y übersetzt, durch Anwendung von LY wird Y zu X) und nur ein mal vorliegen können. X dient als Laufvariable und wird nur dann erzeugt, wenn die Verdopplung der As durch das Y am linken Rand vollendet ist ($LY \rightarrow LX$). Da die Anzahl der As immer der Wortlänge $f(n-1)$ entspricht, kann eine Verdopplung nur die Wörter mit $2 \cdot f(n-1)$ As erzeugen, was durch eine Korrektur (± 1 durch Anwendung AXU bzw. $AAXG$) zum Wort mit $f(n)$ As führt.
Bei der Entstehung des terminalen Ergebnisses bleibt die Anzahl der As unverändert. Dies ist zum einen durch die Regel $Aa \rightarrow aa$ gewährleistet, zum anderen durch das simple Entfernen des nicht-terminalen Symbols L durch $La \rightarrow a$. Durch die Regeln $AAYG \rightarrow a$ und $AYU \rightarrow aa$ wird das Beenden des Wortes eingeleitet. Auch hier wird ein a addiert oder subtrahiert – entsprechend des ns. Damit ergibt sich ein Wort $a^{f(n)} \in L(G)$.

4 Komplexitätsschätzung

Zur Berechnung der Komplexität der gefundenen Grammatik sind zwei Schritte zu verfolgen. Zuerst wird die Komplexität für einen Ableitungsschritt von $f(n-1)$ zu $f(n)$ berechnet. Anschließend wird die Komplexität basierend darauf für einen kompletten Ableitungsschritt von S zu $f(n)$ bestimmt.
Die Anwendung einer Regel entspricht einer Zeiteinheit (ZE). Die dabei zu betrachtenden Regelanwendungen sind:

1. XA $\rightarrow$ AX benötigt etwa $\begin{cases} f(n)+1 & n=2m \\ f(n)-1 & n=2m+1 \end{cases} \approx f(n)$ ZE
2. AXG $\rightarrow$ AAYU und AAXU $\rightarrow$ AYG benötigen je eine ZE
3. AY $\rightarrow$ YAA benötigt $f(n-1)$ Zeiteinheiten und
4. LY $\rightarrow$ LX benötigt eine ZE.

Die Regeln, die das Wort letztendlich in Terminale umwandeln (La $\rightarrow$ aa, Aa $\rightarrow$ aa), benötigen konstante Zeit und müssen daher nicht explizit betrachtet werden. Sie würden dazu führen, dass die Generierung jedes Wortes letztendlich $+f(n)$ Zeiteinheiten benötigt, was sich wiederum kürzen lässt.
Aus den betrachteten Regelanwendugnen ergibt sich nun bei der Ableitung von $f(n-1)$ zu $f(n)$ folgende Summe:

$$\overbrace{3}^{(2),\,(4)} + \overbrace{f(n-1)}^{(3)} + \overbrace{f(n)}^{(1)} \approx f(n+1)$$

Da Konstanten keinen/kaum Einfluss auf die Komplexität haben, fallen diese Weg. Das deutliche Abschätzen mit $f(n+1)$ für $f(n-1)+f(n)$ nach oben ermöglicht im folgenden Schritt eine geringere Grenze, als sie bei den originalen Fibonacci-Zahlen oft verwendet wird. Es ist nun also zu zeigen (Induktionsbehauptung), dass für die Ableitung eines Wortes $f(n)$ weniger als 1.75^n Zeiteinheiten benötigt werden. Die klassische Fibonacci-Rekursion ($f(n) = f(n-1) + f(n-2)$) kann mit $\mathcal{O}(2^n)$ abgeschätzt werden[3]. Allerdings ist es auch möglich durch geschickte Tail-Rekursion eine Zeitkomplexität von $\mathcal{O}(n)$ zu erreichen. Wir genügen uns hier damit, eine Zeitkomplexität von $\mathcal{O}(1.75^n)$ zu zeigen.
Auch dieser Beweis erfolgt mittels vollständiger Induktion über $f(n)$ (Achtung: Nicht $g(n)$, da Äquivalenz gezeigt).

[3] Wissen aus dem Modul "Berechenbarkeit und Komplexität, WS2017"

Für den Induktionsanfang wird $n = 3$ gewählt.

$$\begin{aligned} f(3) &= 3 \\ 1.75^3 &= 8 \\ f(3) &< 8 \end{aligned}$$

Die Induktionsvoraussetzung ist

$$\begin{aligned} f(n) &< 1.75^n \\ f(n-1) &< 1.75^{n-1} \end{aligned}$$

Im Induktionsschritt wird nun also gezeigt, dass die Induktionsbehauptung ($f(n) < 1.75^n$) auch für $n \rightsquigarrow n+1$ gilt.

$$\begin{aligned} f(n+1) &< 1.75^n + 2 \cdot 1.75^{n-1} \\ &\Leftrightarrow 1.75^{n-1} \cdot 1.75 + 2 \cdot 1.75^{n-1} \\ &= 1.75^{n-1} \cdot (1.75 + 2) \\ &< 2 \cdot 1.75^{n+1} \\ \Rightarrow 1.75^{n+1} &< 2 \cdot 1.75^{n+1} \\ &\text{q.e.d.} \end{aligned}$$

Damit wurde gezeigt, dass ein Wort der Länge $f(n)$ mit der Grammatik G in weniger als 1.75^n Zeiteinheiten abgeleitet werden kann, d.h. G hat eine exponentielle Zeitkomplexität: $time(G) \in \mathcal{O}(1.75^n)$.