Induktion als Forschungsinstrument und eine Methode zum Entwurf von effizienten Algorithmen

Ziele

Lernziel

Ein Algorithmus kann man ansehen als eine eindeutig interpretierbare Beschreibung einer Tätigkeit. In einfacheren Darstellungen kann man ein Algorithmus als eine Folge von einfachen Basisaktionen (einzelne Schritte) ansehen. Das Interessante dabei ist, ob der Algorithmus korrekt ist und ob er effizient ist. D. h., löst mein Algorithmus das Problem? Wie schnell kommt der Algorithmus zu einer Lösung? Wie viele Rechenoperationen sind nötig, um das Problem zu lösen? In dieser Unterrichtssequenz werden Antworten zu diesen Fragen gesucht.

Das Verfahren «Beweisen durch vollständige Induktion» ist eine Methode, die beim Entwerfen von Algorithmen nützlich ist. Damit kann man beweisen, dass ein Algorithmus tatsächlich funktioniert und korrekt ist. Durch geschickten Einsatz der Induktion beim Entwurf eines Algorithmus kann die Effizienz berechnet und analysiert werden: Wie schnell kann der Algorithmus das Problem lösen? Gibt es eine schnellere Möglichkeit? Geht es noch schneller?

Die Unterrichtssequenz versucht mit geeigneten Aufgaben und Beispielen beide Aspekte, die Korrektheit und die Effizienz eines Algorithmus, zu erklären.

Voraussetzung

Bekannte Konzepte und Begriffe

Algorithmus – Induktion – Induktionsschritt – Induktionsanfang – Effizienz – Korrektheit – Logarithmus – Exponentielles Wachstum – Fakultät

Die Beweismethode der vollständigen Induktion ist den Lernenden bekannt. Das Thema wurde bereits in der Mathematik eingeführt. Die Lernenden kennen einige Beispiele:

- n Elemente kann man auf n! verschiedene Arten anordnen
- Eine Gerade zerlegt eine Ebene in 2 Gebiete. In wie viele Gebiete kann eine Ebene durch n Geraden höchstens zerlegt werden?
- $2^n > n^2$ für n >= 5
  $n! > 2^n$ für n >= 4
  …….

Die Lernenden kennen den Begriff Algorithmus aus der Programmierung. Sie haben bereits einfache Algorithmen selbst programmiert. Das modulare Programmieren ist der Klasse bekannt.

Einstiegsrätsel

Rätsel

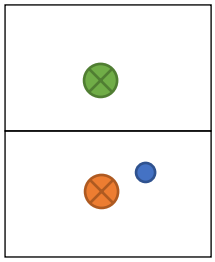
Wir spielen Völkerball, ein Spiel aus dem Sportunterricht!
Völkerball ist ein beliebtes Spiel, welches aus Tradition und aufgrund der einfachen Organisation oft gespielt wird. Die Spielidee ist, die Gegenspieler abzuschiessen. Keines der grossen Sportspiele hat eine ähnliche Spielidee. Trotzdem oder gerade deshalb lebt das Spiel von einer sehr grossen Spannung.

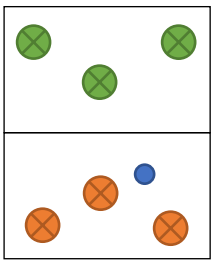
Völkerball Grundregeln (vereinfacht): Gespielt wird auf gleich grossen, durch die Mittellinie getrennt, Spielfeldern. Jeder Spieler/ jede Spielerin versucht die GegnerInnen zu treffen. Getroffene Spieler*innen sind nicht mehr im Spiel. Das Spiel ist fertig, sobald eine Mannschaft keine Spieler*innen mehr hat. Es wird nur mit einem Ball gespielt.

Variante Zahlenvölkerball: Jedes Team nummeriert seine SpielerInnen durch, ohne dass die Gegner wissen, wer welche Nummer hat. Die gegnerische Mannschaft muss zuerst die Nr. 1, anschliessend Nr. 2 usw. treffen. Wenn man den Spieler/die Spielerin mit der kleinsten Nummer im Spiel trifft, muss er/sie seine/ihre Nummer zeigen und das Spiel verlassen. Spieler mit grösseren Nummern, auch wenn getroffen, behalten ihr Geheimnis - welche Nummer sie tragen und spielen weiter. Man speichert aber, dass man sie schon mal getroffen hat und trifft sie nicht wiederholend in dieser Runde. Somit muss man im schlechtesten Fall alle SpielerInnen genau einmal treffen, um am Ende die kleinste Nummer zu treffen.

Wie viele Treffer braucht eine Mannschaft, um zu gewinnen? (Variante Zahlenvölkerball)

Wir suchen die Anzahl Treffer, die es **höchstens** (worst case) braucht, um ein Spiel zu gewinnen, und zwar aus der Sicht von einer angreifenden Mannschaft. Wir spielen mit einem Ball.





1 Gegner, 1 Ball → Der Angreifer braucht 1 Treffer T(1) = 1, um das Spiel zu gewinnen.
3 Gegner, 1 Ball → Wie viele Treffer braucht die angreifende Mannschaft, um das Spiel zu gewinnen?

Wie sieht es aus, wenn 2, 3, 4, oder 5 Spieler*innen spielen?

Vervollständige die folgende Tabelle:

Anzahl Spieler	1	2	3	4	5
Treffer t_n[1]	1				

[1]Es wird immer nach den Treffern im schlechtesten Fall gefragt. Wenn man zufällig, immer die anstehende Zahl trifft, dann ist das Spiel schneller fertig.

- Welche Strategie hast Du gewählt, um die Anzahl Treffer zu finden?
- Beim Lösen des Rätsels hast Du eine Folge von Zahlen bekommen. Gibt es einen erkennbaren Zusammenhang oder eine erkennbare Regelmässigkeit zwischen den Zahlen?
- Wie sind die Zahlen zustande gekommen?
- Kannst Du eine Formel für die Zahlenfolge definieren?
- Angenommen, die angreifende Mannschaft trifft **zufällig** immer die richtige Zahl zuerst. Wie viele Treffer braucht es dann, um das Spiel zu gewinnen, bei einer Mannschaftsgrösse n?

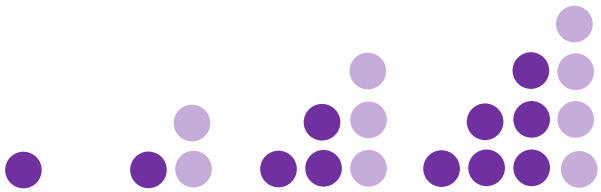
Begründung: Dreieckszahlen[2]

Eine Dreieckszahl ist eine Zahl, die der Summe aller Zahlen von 1 bis zu einer Obergrenze n entspricht.
Beispielsweise ist die 10 eine Dreieckszahl, da 1 + 2 + 3 + 4 = 10 ist.
Die ersten Dreieckszahlen sind: 1, 3, 6, 10, 15, 21, 28, 36, 45, 55, …
Die Bezeichnung Dreieckszahl leitet sich von der geometrischen Figur des Dreiecks her.

Beispiel Zahlenvölkerball

Spieler	1	2	3	4	5	n
Treffer t_n	1	3	6	10	15	?
	1	1 + 2	1+2+3	1+2+3+4	1+2+3+4+5	n(n+1) /2

Behauptung



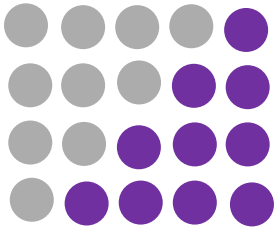
Die Anzahl der Treffer, die man zum Gewinnen benötigt, kann man als gleichseitiges Dreieck anschauen und entspricht immer einer Dreieckszahl. Aus zehn Treffern lässt sich beispielsweise ein Dreieck legen, bei dem jede Seite von vier Treffer (Anzahl Spieler) gebildet wird.

Aufgrund dieser Verwandtschaft mit einer geometrischen Figur zählen die Dreieckszahlen zu den figurierten Zahlen, zu denen auch die Quadratzahlen und Kubikzahlen gehören. Schon Pythagoras hat sich mit Dreieckszahlen beschäftigt.
Die n-te Dreieckszahl ist die Summe der Zahlen von 1 bis n.

Anstatt die einzelnen Zahlen zu addieren, können Dreieckszahlen auch durch die **gausssche Summenformel (kleiner Gauss)** berechnet werden: $\Delta_n = n\ (n + 1) / 2$.

[2]https://de.wikipedia.org/wiki/Dreieckszahl

Grafische Begründung der gausssche Summenformel



Die Formel $\Delta_n = n\,(n + 1) / 2$ lässt sich auch durch Auslegen der Dreieckszahlen veranschaulichen. Die Dreieckszahl Δ_4 lässt sich als Dreieck oder Treppe auslegen. Das Doppelte einer Dreieckszahl entspricht zwei gleichen Treppen, die sich zu einem Rechteck zusammenfügen lassen. Das Rechteck hat eine Höhe von 4 und eine Breite von 5 → $\Delta_4 = 4\,(4 + 1) / 2 = 10$. Allgemein hat das Rechteck die Grösse $n(n + 1)$.

Aus der Geschichte[3]

Vom bedeutenden Mathematiker Karl Friedrich Gauss (1777-1855) erzählt man sich die folgende Geschichte:
Er sollte als Schüler in der Schule die Zahlen von 1 bis 100 zusammenzählen. Der Lehrer nahm an, dass er damit eine Weile beschäftigt war. Schon nach kurzer Zeit fand er die Summe 5050.

Mathematische Begründung:
Statt stur die Zahlen von 1 bis 100 der Reihe nach zu addieren, bildete er Zahlenpaare mit denselben Summenwerten und konnte multiplizieren:
Für die Summe der ersten n natürlichen Zahlen gilt:

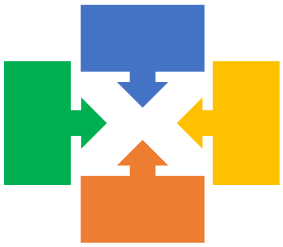
$$\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

$$\begin{aligned}
2 \cdot \sum_{k=1}^{n} k &= \sum_{k=1}^{n} k + \sum_{k=1}^{n} k \\
&= 1 + 2 + \ldots + (n-1) + n + 1 + 2 + \ldots + (n-1) + n \\
&\quad \downarrow \text{Umsortieren} \\
&= \underbrace{(1+n)}_{=n+1} + \underbrace{(2+n-1)}_{=n+1} + \ldots + \underbrace{(n-1+2)}_{=n+1} + \underbrace{(n+1)}_{=n+1} \\
&= \underbrace{(n+1) + (n+1) + \ldots + (n+1) + (n+1)}_{n\text{-mal } n+1} \\
&= n \cdot (n+1)
\end{aligned}$$

Die Formel erhält man durch eine Division der beiden Seiten mit 2. $2 \cdot \sum_{k=1}^{n} k = n \cdot (n+1)$

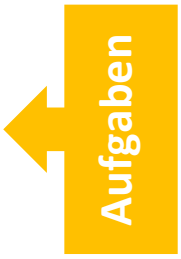
[3]Quelle: Gausssche Summenformel – Serlo „Mathe für Nicht-Freaks“ – Wikibooks, Sammlung freier Lehr-, Sach- und Fachbücher

Beweis durch vollständige Induktion



Die Methode der vollständigen Induktion ist mit dem Dominoeffekt vergleichbar: Wenn der erste Dominostein fällt und durch jeden fallenden Dominostein der nächste umgestossen wird, wird schliesslich jeder Dominostein der unendlich lang gedachten Kette irgendwann umfallen.
Die Allgemeingültigkeit einer Aussageform **A(n)** ist für n=1,2,3, … bewiesen, wenn **A(1)** gültig ist (der erste Stein fällt um) und wenn zusätzlich gilt aus A(n) folgt A(n+1) für n=1,2,3, … (jeder Stein reisst beim Umfallen den nächsten Stein mit).

Um die Korrektheit der Behauptung bzw. unseres Algorithmus zu beweisen, sollte nun das Prinzip der vollständigen Induktion verwendet werden.



1. Aufgabe
Zeige, mit vollständiger Induktion, dass $t_n = n\ (n + 1) / 2$ für alle natürliche Zahlen gilt.
Für den Induktionsschritt wird die Funktionsgleichung aus dem Anfangsrätsel benutzt: $t_{n+1} = t_n + (n + 1)$.

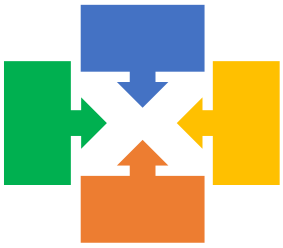
Induktionsanfang
Zeige: A(1) → $t_1 = 1\ (1 + 1) / 2 = 1$

Induktionsschritt
Zeige: aus A(k) folgt A(k + 1)
$t_k = k\ (k + 1) / 2$ → zu zeigen $t_{k+1} = (k + 1)\ (k + 2) / 2$

2. Aufgabe – Teilbarkeit (Repetition aus der Mathematik)
$\forall\ \boldsymbol{n} \in \mathbb{N} = \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \ldots\}$
Zeige mit Induktion, dass $\boldsymbol{n^2 + n}$ durch 2 teilbar ist (gerade).

3. Aufgabe – Teilbarkeit (Repetition aus der Mathematik)
$\forall\ \boldsymbol{n} \in \mathbb{N} = \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \ldots\}$
Zeige, dass $\boldsymbol{n^3 - n}$ durch 6 teilbar ist.

Effizienz eines Algorithmus



Was wissen wir schon?
Ein Algorithmus ist eine Abfolge von Schritten, die ein Problem löst. Das Interessante dabei ist, ob der Algorithmus korrekt und ob er effizient ist. D. h., wie schnell löst der Algorithmus das Problem bzw. wie viele Rechenoperationen sind nötig, um das Problem zu lösen. Wir haben für das Anfangsrätsel einen Algorithmus gefunden, welcher mit wenigen Operationen die Anzahl Treffer berechnet, ohne dass man die n Additionen ausführen muss, um die gesuchte Zahl zu berechnen. Bei einer kleinen Zahl n könnte man die Anzahl Additionen in Kauf nehmen und das Problem wäre einfach gelöst. Je grösser die Zahl n ist, desto umständlicher wird es, die einzelnen Additionen durchzuführen. Unser Algorithmus löst die Aufgabe sehr schnell.

Faltenproblem

Wir falten quadratische Papierblätter!

Die Aufgabe besteht darin, ein Gitter durch Falten eines Papierblattes zu erhalten, das aus 16 Kästchen besteht. Jede Faltung entspricht einer oder mehreren Linien. Wenn man das gefaltete Papier wieder öffnet, müssen die Falten genau der Linien des Gitters entsprechen.

Dabei sollten 3 verschiedene Möglichkeiten/Algorithmen beschrieben werden, welche die Aufgabe lösen.

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

1. **Finde 3 Möglichkeiten (Algorithmen), um das Problem zu lösen!**
2. **Wie viele Faltoperationen braucht es für jeden Algorithmus?**

4. Aufgabe – Gitter mit n x n Kästchen

a. Wie bereits im Rätsel gesehen, wird ein Gitter mit 16 Kästchen gefaltet. Nun möchtest Du ein n x n Gitter falten, wobei $n = 2^m$ und m ist eine natürliche Zahl ≥ 0. Benutze die bereits im Rätsel gefundenen Algorithmen und versuche die Anzahl Faltungen bezüglich des Parameters n zu berechnen. Verwende für die Formulierung das Prinzip der vollständigen Induktion.

a. Vergleiche die Komplexität der Algorithmen. Wie gross ist der Unterschied für n = 1024?

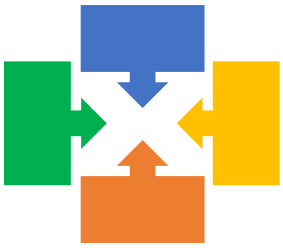
5. Aufgabe – Gitter mit 16 Kästchen zeichnen (anspruchsvoll)

Schreibe ein Programm, welches das Gitter aus dem obigen Beispiel zeichnet. Verwende dabei, Schleifen und den modularen Entwurf.

6. Aufgabe – Komplexität

Wie viele Vorwärts-Bewegungen braucht der entworfene Algorithmus, um das Gitter zu zeichnen?

Zusammenfassung /wichtige Begriffe



Was haben wir gelernt?

Der Entwurf von Algorithmen ist ein kreativer Prozess, der nicht automatisiert werden kann. Dieser Prozess muss auf systematischer Weise durchgeführt werden. Methoden zum systematischen Entwurf und zur Analyse von Algorithmen (Komplexität und Korrektheit) sind hilfreich. Eine davon ist das Domino-Prinzip der mathematischen Induktion: Es wird zuerst eine Lösung eines Basisproblems (Induktionsanfang) gesucht. In einem zweiten Schritt sucht man die Problemlösung aus Lösungen kleinerer Probleme (Induktionsschritt).

Nicht nur die Korrektheit, sondern auch die Effizienz eines Algorithmus ist ein wichtiges Entwurfskriterium. Es gibt mehrere Möglichkeiten alle Zahlen zwischen 1 und 100 aufzusummieren: (1+100) + (2+99) +… (50+51) = 50*101 = 5050 oder 1+2+3+4…+99+100 = 5050. Oft gibt es verschiedene Algorithmen zur Lösung eines Problems. Es stellt sich dann die Frage, welcher dieser Algorithmen der günstigste ist und in der Praxis eingesetzt werden sollte. Besonders interessant sind die Algorithmen, die mit möglichst wenig Rechenzeit auskommen.

Schlussrätsel

Unser Anfangsbeispiel: Zahlenvölkerball

Nachdem wir Einiges über Entwurfsmethoden von Algorithmen gelernt haben, stellt sich folgende Frage:

Mit welchen Regel-/Spieländerungen würde man die Komplexität des Zahlenvölkerballs verringern?

Lösungen

Einstiegsrätsel:

Wir suchen die Anzahl Treffer, die es **höchstens** (worst case) braucht, um ein Spiel zu gewinnen, und zwar aus der Sicht von einer angreifenden Mannschaft. Wir spielen mit einem Ball. Zuerst versucht man alle Spieler*innen zu treffen (n Treffer), um die Nummer 1 zu finden. Danach spielt man weiter mit (n – 1) Spieler*innen weiter.

Anzahl Spieler	1	2	3	4	5
Treffer t_n^*	1	1 + 2	1 + 2 + 3	t_3 + 4	t_4 + 5

$t_{n+1} = t_n + (n+1)$ oder **$t_n = t_{n-1} + n$**

Aufgabe 1

Zu zeigen die Behauptung A(n): $t_n = n(n+1)/2 = 1 + 2 + \ldots + (n-1) + n$ für alle natürliche Zahlen, n >0

Induktionsanfang

Zeige A(1) → $t_1 = 1$ und für n = 1 gilt $n(n+1)/2 = 1(1+1)/2 = 1$ ✓

Induktionsvoraussetzung

A(n) ➔ $t_n = n(n+1)/2$

Induktionsschritt

Zeige aus A(k) impliziert A(k + 1)

$t_k = k(k+1)/2$ zu zeigen $t_{k+1} = (k+1)(k+2)/2$

Dafür benötigen wir die Funktionsgleichung $t_{n+1} = t_n + (n+1)$

$$t_{k+1} = t_k + (k+1)$$

$$t_{k+1} = \frac{k(k+1)}{2} + (k+1)$$ [nach A(k) können wir t_k durch k (k + 1) / 2 ersetzen]

$$t_{k+1} = \frac{k(k+1)+2(k+1)}{2}$$ [Distributivgesetz]

$$t_{k+1} = \frac{(k+1)(k+2)}{2}$$ ✓

Aufgabe 2

$\forall \boldsymbol{n} \in \mathbb{N} = \{\mathbf{1}, \mathbf{2}, \mathbf{3}, \ldots\}$
Zu zeigen mit Induktion, dass $\boldsymbol{n^2 + n}$ durch 2 teilbar ist (gerade).

Induktionsanfang
n = 1: $n^2 + n = 1^2 + 1 = 2$ ist eine gerade Zahl.

Induktionsvoraussetzung
Es gelte die Induktionsvoraussetzung: $n^2 + n$ ist eine gerade Zahl.

Induktionsschritt
Die Behauptung gilt auch für (n + 1), zu zeigen ist, dass $(n + 1)^2 + (n + 1)$ gerade ist.

$$\begin{aligned}(n + 1)^2 + (n + 1) &= n^2 + 2n + 1 + n + 1\\ &= (n^2 + n) + (2n + 2)\\ &= (n^2 + n) + 2(n + 1)\ \checkmark\end{aligned}$$

Ist eine gerade Zahl, denn der erste Summand gerade nach Induktionsvoraussetzung ist und der zweite Summand ein ganzzahliges Vielfaches von 2 ist. Die Summe von zwei geraden Zahlen ist eine gerade Zahl.

Aufgabe 3

$\forall \boldsymbol{n} \in \mathbb{N} \geq \mathbf{0}$
Zu zeigen mit Induktion, dass $\boldsymbol{n^3 - n}$ durch 6 teilbar ist.

Induktionsanfang
n = 0: $n^3 - n = 0^3 - 0 = 0$ →ist durch 6 ohne Rest teilbar ✓

Induktionsvoraussetzung
Es gelte die Induktionsvoraussetzung: $n^3 - n$ ist durch 6 teilbar

Induktionsschritt
Die Behauptung gilt auch für (n + 1), zu zeigen ist, dass $(n + 1)^3 - (n + 1)$ durch 6 teilbar ist.

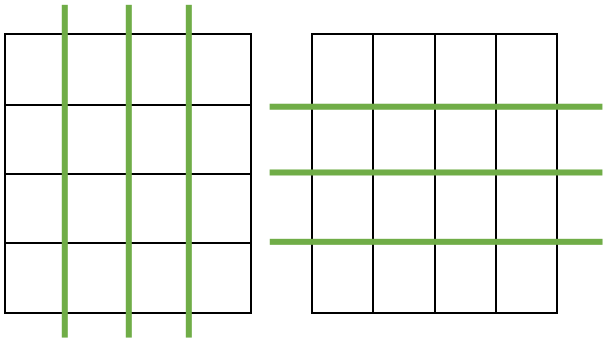
$$\begin{aligned}(n + 1)^3 - (n + 1) &= n^3 + 3n^2 + 3n + 1 - n - 1\\ &= n^3 - n + 3n^2 + 3n\\ &= (n^3 - n) + 3n(n + 1)\ \checkmark\end{aligned}$$

Der erste Summand $(n^3 - n)$ ist durch 6 teilbar nach Induktionsvoraussetzung. Der zweite Summand 3n(n + 1) ist durch 3 und auch durch 2 teilbar, da entweder n oder die darauf folgende Zahl (n + 1) eine gerade Zahl ist. Ist eine Zahl durch 3 und auch durch 2 teilbar, dann ist sie auch durch 6 teilbar. Da beide Summanden durch 6 teilbar sind, ist auch deren Summe durch 6 teilbar.

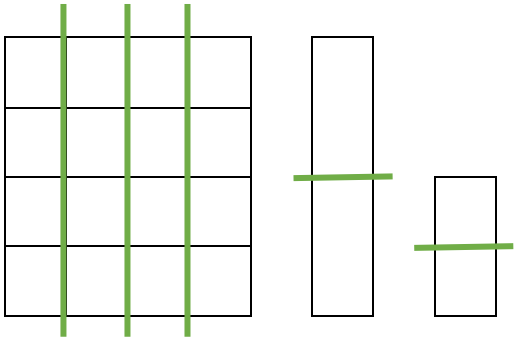
Faltenproblem:

Die Aufgabe besteht darin, ein 4 x 4 Gitter durch Falten eines Papierblattes zu erhalten, das aus 16 Kästchen besteht. Dabei sollten 3 verschiedene Möglichkeiten/Algorithmen beschrieben werden, welche die Aufgabe lösen.

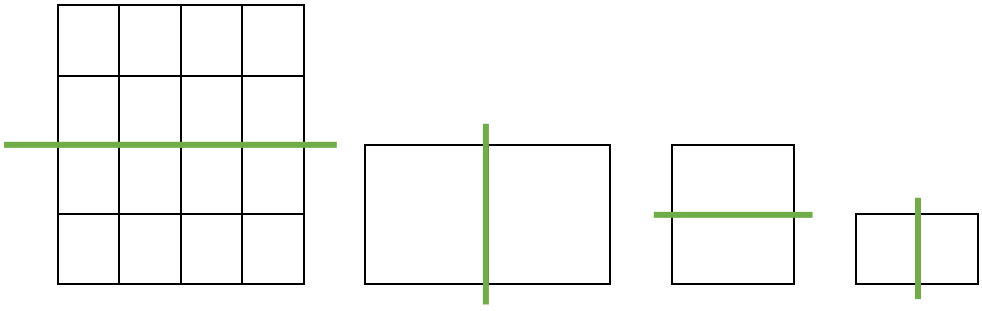
1. Algorithmus: Wir falten das Papier spaltenweise (3 Faltungen), danach öffnen wir es und falten das Papier zeilenweise (3 Faltungen). Mit 6 Faltoperationen lösen wir das Problem.



2. Algorithmus: Wir falten das Papier spaltenweise (3 Operationen), dann halbieren wir jeweils die Höhe (2 Operationen). Im Ganzen sind 5 Faltoperationen nötig, um das Problem zu lösen.



3. Algorithmus: Wir halbieren jeweils mit einer Faltoperation das Blatt. Zuerst horizontal, dann vertikal, bis wir die gewünschte Anzahl Kästchen erhalten haben. Dafür brauchen wir 4 Faltoperationen.



Aufgabe 4a

1. Wir wollen die Behauptung A_1 (n): «Der 1. Algorithmus braucht für die Faltung des n x n Gitters 2(n - 1) Faltungen.» mit Induktion beweisen.
   Sei F_1 (n) die Anzahl der vom 1. Algorithmus nötigen Faltungen.

   Induktionsanfang
   Für n = 1 brauchen wir keine Faltungen, F_1 (0) = 0 und es gilt 2(n - 1) = 2(1 - 1) = 0 ✓

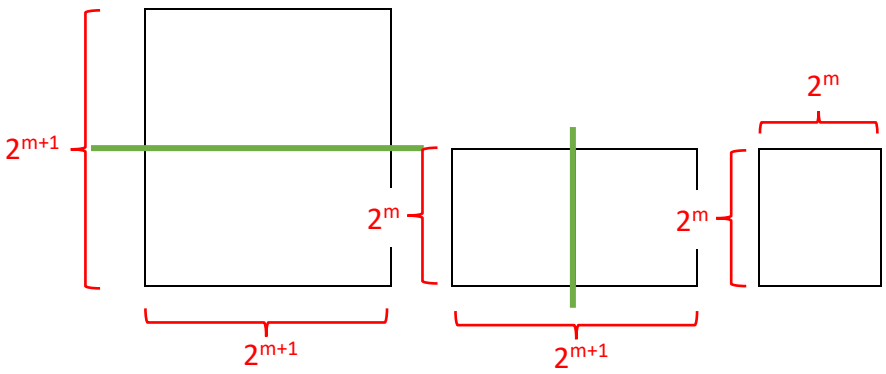
   Induktionsschritt
   Wenn das n x n Gitter zu einem (n + 1) x (n + 1) Gitter erweitert wird, braucht das 1. Algorithmus eine Spaltenfaltung und eine Zeilenfaltung mehr.
   Somit gilt F_1 (n + 1) = F_1 (n) + 2.
   Die Aufgabe ist zu zeigen, dass A_1 (n) die Behauptung A_1 (n + 1) impliziert.

   F_1 (n + 1) = F_1 (n) + 2
   = 2(n - 1) + 2 = 2n = 2((n + 1) -1)
   Somit ist der Induktionsschritt bewiesen. ✓

2. Wir wollen $\forall\, m \in \mathbb{N}$ die Behauptung A_3(m) beweisen: «Der 3. Algorithmus faltet das Gitter der Grösse 2^m x 2^m durch F_3(m) = 2m viele Faltungen.»

   Induktionsanfang
   Für m = 0 haben wir ein Gitter der Grösse 2^0 x 2^0 = 1 x 1, wir brauchen keine Faltung, also ist F_3 (m) = 0. Für m = 0 ist 2 x m = 2 x 0 = 0 ✓

   Induktionsschritt
   Die Aufgabe ist zu zeigen, dass A_3 (m) die Behauptung A_3 (m + 1) impliziert.
   Wir beobachten, dass der 3. Algorithmus zwei Faltungen braucht, um aus einem 2^{m+1} x 2^{m+1} Gitter zu einem 2^m x 2^m Gitter zu gelangen.



Somit gilt A_3 (m + 1) = A_3 (m) + 2.
A_3 (m + 1) = A_3 (m) + 2
= 2m + 2
= 2(m + 1) ✓

Aufgabe 4b

Vergleiche die Komplexität des 1. und des 3. Algorithmus A_1 und A_3 für $n = 2^m$. Wie gross ist der Unterschied für n = 1024?
Der 1. Algorithmus braucht $F_1 (n) = 2(n - 1) = 2(1024 -1) = 2046$ Faltungen.
Der 3. Algorithmus braucht $F_3(m) = 2m = 2\log_2 n = 2 \times 10 = 20$ Faltungen.
Der 3. Algorithmus ist wesentlich schneller.