

GYMNASIUM OTTOBRUNN

Oberstufenjahrgang 2017/19

Seminarfach Softwareentwicklung

Seminararbeit

## **36. Bundeswettbewerb Informatik**

### **Runde 2**

### **Aufgabe 1 und 3**

Verfasser: Jonas Fritsch

Seminarleiter: StD Peter Brichzin

Bewertung: ..... Punkte

Unterschrift des Seminarleiters: .....

# INHALTSVERZEICHNIS

<b>1</b>	<b>EINLEITUNG</b>	<b>3</b>
<b>2</b>	<b>AUFGABE 1 - "Die Kunst der Fuge"</b>	<b>4</b>
2.1	Aufgabenstellung . . . . .	4
2.2	Auslegung der Aufgabe . . . . .	4
2.3	Lösungsidee . . . . .	5
2.3.1	Verstehen des Problems . . . . .	5
2.3.2	Der Algorithmus . . . . .	10
2.4	Implementierung . . . . .	11
2.5	Laufzeitanalyse . . . . .	11
2.6	Optimierungsmöglichkeiten . . . . .	11
2.7	Beispiele . . . . .	11
2.8	Quellcode . . . . .	11
<b>3</b>	<b>AUFGABE 3 - "Quo vadis, Quax?"</b>	<b>12</b>
3.1	Aufgabenstellung . . . . .	12
3.2	Lösungsidee . . . . .	12
3.3	Teilaufgabe (a) . . . . .	12
3.4	Teilaufgabe (b) . . . . .	12
3.5	Teilaufgabe (c) . . . . .	12
3.5.1	Implementierung . . . . .	12
3.5.2	Laufzeitanalyse . . . . .	12
3.5.3	Optimierungsmöglichkeiten . . . . .	12
3.5.4	Beispiele . . . . .	12
3.6	Teilaufgabe (d) . . . . .	12
3.7	Quellcode . . . . .	12
<b>4</b>	<b>FAZIT</b>	<b>13</b>
<b>5</b>	<b>ABBILDUNGSVERZEICHNIS</b>	<b>14</b>
<b>6</b>	<b>LITERATURVERZEICHNIS</b>	<b>15</b>
<b>7</b>	<b>ERKLÄRUNG DES VERFASSERS</b>	<b>16</b>

# 1 EINLEITUNG

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed tempus consectetur lorem, imperdiet dignissim est auctor a. Vivamus convallis, leo et iaculis egestas, nunc massa porttitor tellus, id faucibus urna justo eget massa. Praesent quis feugiat odio. Nullam quis mattis enim. Fusce volutpat odio in enim sodales venenatis. Mauris consequat.

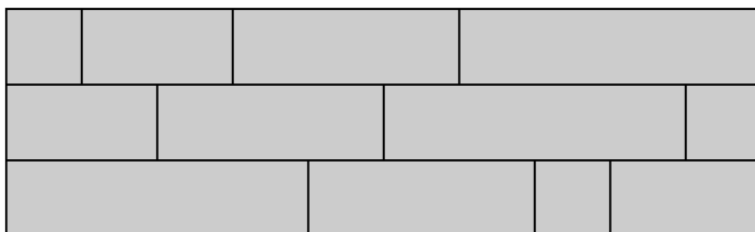
## 2 AUFGABE 1 - "Die Kunst der Fuge"

### 2.1 Aufgabenstellung

Ilona besitzt einen riesigen Haufen Holzklötzchen: Diese haben alle dieselbe Höhe und Tiefe, aber verschiedene Längen.

Ilona möchte eine Mauer bauen. Jede Reihe der Mauer soll aus  $n$  Klötzchen bestehen, die die Längen 1 bis  $n$  haben und lückenlos aneinander liegen. Die Stellen zwischen den Klötzchen heißen Fugen. Ilona möchte, dass in der fertigen Mauer niemals zwei Fugen übereinander liegen, selbst wenn sich mehrere Reihen dazwischen befinden. Außerdem soll ihre Mauer möglichst hoch sein.

Für  $n = 4$  gelingt es ihr recht schnell, eine Mauer mit drei Reihen zu bauen:



### Aufgabe

Hilf Ilona, indem du ein Programm schreibst, das nach Eingabe von  $n$  eine nach ihren Vorgaben konstruierte, möglichst hohe Mauer ausgibt. Für  $n = 10$  sollte dein Programm eine Mauer der Höhe 6 ausgeben können. Wie hoch werden die Mauern deines Programms für größere  $n$ ?

### 2.2 Auslegung der Aufgabe

Die grundlegende Aufgabe ist, eine Mauer mit möglichst vielen Reihen, die aus verschieden länglichen Klötzchen bestehen, zu bauen.

Die Längen der Klötzchen in einer Reihe sind durch eine Variable  $n$  definiert. Die Variable  $n$  ( $n \in \mathbb{N} \setminus \{0\}$ ) definiert dabei die Längen und auch die Gesamtanzahl der Klötzchen pro Reihe. Gesamtanzahl heißt, dass bei zum Beispiel  $n = 5$  jede Reihe aus 5 einzelnen Klötzchen besteht. Jede Reihe setzt sich dann aus einem 1er, einem 2er, einem 3er, einem 4er und einem 5er Klotz zusammen.

Der Trick bei der Aufgabe ist nun, dass niemals zwei Fugen **übereinanderliegen**

**dürfen.** Eine Fuge ist die Lücke zwischen zwei Klötzchen (siehe Abbildung 1).

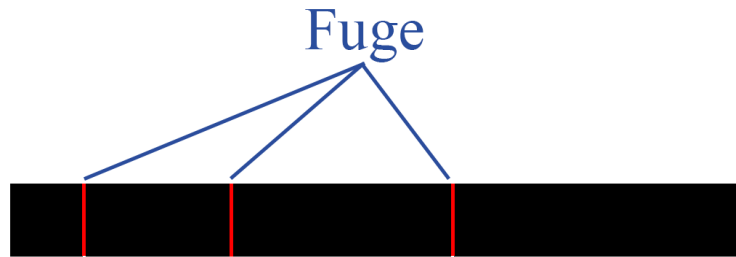


Abbildung 1: Darstellung der 3 Fugen einer Reihe mit 4 Klötzchen

Als Hilfestellung wurde außerdem erwähnt, dass eine Mauer für  $n = 10$  eine maximale Höhe von 6 hat.

## 2.3 Lösungsidee

### 2.3.1 Verstehen des Problems

Um die Problemstellung der Aufgabe genauer zu verstehen, bietet es sich erstmal an, Mauern für kleinere  $n$  auf Papier zu zeichnen. Beginnt man mit  $n = 1$ , fällt auf, dass ganz gleich wie viele Reihen übereinanderliegen, es nie zu einer Fugenüberlappung kommen kann, da eine Reihe mit einem einzelnen Klötzchen keine Fuge besitzt. Für  $n = 1$  kann also theoretisch eine **unendlich** hohe Mauer gebaut werden.

Für  $n = 2$  ist es auch noch recht einfach, eine Mauer zu bauen. Denn es gibt pro Reihe nur 2 mögliche Reihenfolgen von Klötzchen. Entweder kommt zuerst der 1er oder der 2er Klotz. Eine der zwei möglichen Lösungen ist in Abbildung 2 zu sehen. Es gibt keine Möglichkeit, noch ein Klötzchen mehr zu platzieren, ohne eine Fuge doppelt zu besetzen.

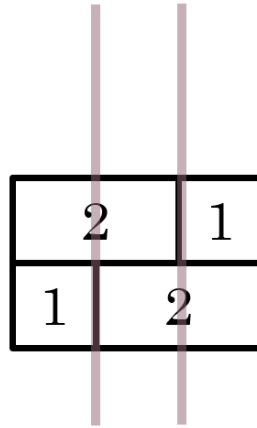


Abbildung 2: Eine der zwei möglichen Lösungen für  $n = 2$ . Die roten Striche zeigen an, dass die jeweilige Fuge bereits besetzt ist.

Auch für  $n = 3$  gibt es eine Mauer mit einer maximalen Höhe von 2 (siehe Abbildung 3). Hier wird es jedoch schon komplexer, da es nun für eine Reihe nicht mehr  $2! = 2 \times 1 = 2$ , sondern  $3! = 3 \times 2 \times 1 = 6$  mögliche Klötzchen-Reihenfolgen gibt. Außerdem tritt bei  $N = 3$  das erste Mal auf, dass eine **mögliche Fugenstelle nicht besetzt wird**.

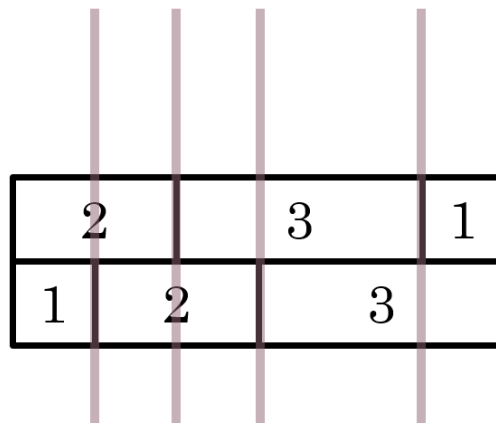


Abbildung 3: Eine mögliche Lösung für  $n = 3$ . Zu beachten ist, dass in der Mauer eine **mögliche Fugenstelle**, dort wo es eine größere Lücke zwischen den zwei letzten besetzten Fugenstellen gibt, **unbesetzt bleibt**.

Geht man noch weiter und bildet eine Mauer für  $n = 4$  ( $4! = 24$  mögliche Klötzchen Anordnungen pro Reihe), so kann man erstmals eine Mauer der Höhe 3 bilden (siehe Abbildung 4). Hier ist auch wieder jede einzelne Fugenstelle besetzt.

	4		3	1	2
2		3		4	1
1	2		3		4

Abbildung 4: Eine mögliche Lösung für  $n = 4$

Die Komplexität der Aufgabe steigt mit höheren  $n$  stark an. Während man bei  $n = 2$  2 Möglichkeiten pro Reihe und 2 Reihen hatte ( $2! \times 2 = 4$  unterschiedliche Lösungen), gibt es bei  $n = 4$  bereits  $4! \times 3 = 72$  verschiedene Lösungen. Allgemein gibt es also für eine Mauer  $n! \times h$  Lösungen, wobei  $h$  die Anzahl der übereinanderliegenden Reihen ist.

Es ist logisch, dass zwischen der Anzahl der maximal möglichen Reihen  $h$  und der Anzahl der Klötzchen pro Reihe  $n$  eine Abhängigkeit existiert.

Allgemein kann man sagen, dass die **maximale Mauerhöhe** dann erreicht ist, wenn **nicht** mehr **genug Fugenstellen in der Mauer frei sind**, um eine **weitere Reihe zu bilden**.

Somit lässt sich die Formel

$$h = \frac{f_{Mauer}}{f_{Reihe}}$$

ableiten, wobei  $f_{Mauer}$  die maximale Anzahl an Fugen in der Mauer und  $f_{Reihe}$  die Anzahl der Fugen, die eine Reihe besetzt, beschreibt.

Um nun auf  $f_{Mauer}$  schließen zu können, braucht man nur die Länge der Mauer. Da jede Reihe der Mauer gleich lang ist, ist die Länge der Mauer gleich der Länge einer Reihe. Die Länge einer Reihe definiert sich wiederum durch die Menge und Breite ihrer Klötzchen. Das heißt also, die Länge einer Reihe/der Mauer ist nach der **Gaußschen Summenformel**

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

Um nun aber von der Länge der Mauer auf  $f_{Mauer}$  zu kommen, muss man noch 1 von der Länge der Mauer abziehen. Dies wird offensichtlich, wenn man zu einer Mauer eine Reihe aus lauter 1er Klötzchen bildet (siehe Abbildung 5). Da das Ende des letzten Klötzchens nicht als Fuge gilt, wird von der Länge der Reihe noch 1 abgezogen.

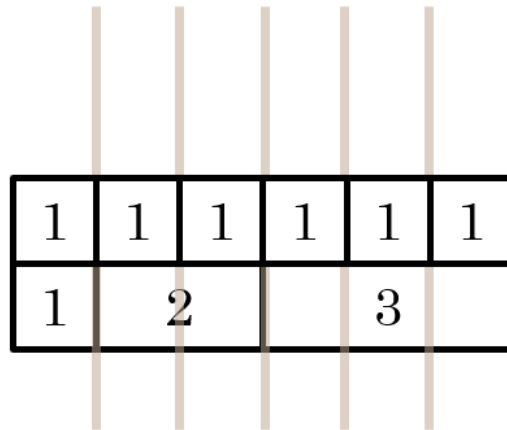


Abbildung 5: Eine Mauer der Länge 6 besitzt 5 mögliche Fugenstellen.

Ähnlich lässt sich auch  $f_{Reihe}$  berechnen. Um wissen zu wollen, wie viele freie Fugen eine Reihe mit  $n$  Klötzchen besetzen wird, zieht man einfach wieder von der Anzahl der Klötzchen pro Reihe ( $n$ ) 1 ab (siehe Abbildung 6).

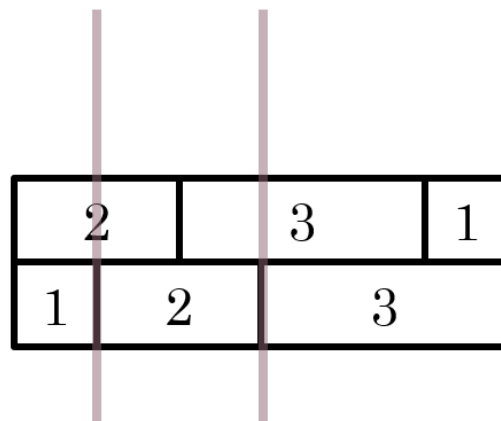


Abbildung 6: Eine Reihe bei  $n = 3$  besetzt  $n - 1 = 2$  freie Fugen.

Daraus ergibt sich die fertige Formel für die Anzahl der Reihen  $h$  in einer maximal hohen Mauer in Abhängigkeit von  $n$ .



$$h = \frac{\frac{n \times (n+1)}{2} - 1}{n - 1} = \frac{n}{2} + 1$$

Die naivste Lösung für die Aufgabe wäre per Brute-Force so lange Klötzchen aneinanderzureihen und Reihen aufeinanderzustapeln, bis man eine fertige Mauer der maximalen Höhe erreicht hat. Jedoch ist die Laufzeit der Brute-Force Methode mit

$$\mathcal{O}(n!^{\lfloor \frac{n}{2} + 1 \rfloor})$$

schlechter als exponentiell.

Für die in der Aufgabenstellung festgelegte Mindestanforderung  $n = 10$ , gäbe es bereits

2.283.380.023.591.730.815.784.976.384.000.000.000.000

**unterschiedliche Mauerkombinationen** für eine Mauerhöhe von 6.

Nun ist hierbei allerdings zu beachten, dass es durchaus mehrere mögliche unterschiedliche Lösungen geben kann und das bei der Gesamtmenge unterschiedlicher Mauern auch auf die Reihenfolge von Reihen geachtet wird (siehe Abbildung ———), welche für die Aufgabenstellung keinen Unterschied macht.

2	3	1
1	2	3

1	2	3
2	3	1

Abbildung 7: Obwohl beide Mauern aus den selben zwei Reihen bestehen, werden sie als unterschiedliche Mauern angesehen.

Weiterhin ist zu beachten, dass es für Mauern mit ungeradem  $n$  mehr unterschiedliche Lösungen gibt, als für die wieder darauffolgende Mauer mit geradem  $n$ .

Dies liegt eben daran, dass bei ungeraden  $n$  immer mindestens eine Fugenstelle frei bleibt, was zu mehr richtigen Mauervariationen führt.


Trotzdem würde das Finden einer Lösung für  $n = 10$ , selbst wenn ein Compu-

ter nur **1 Millisekunde** für die Permutation einer **gesamten Mauer** benötigen würde, die maximal erwartete Algorithmuslaufzeit einer BwInf-Lösung bei weitem überschreiten.

Damit ist eine naive Brute-Force Methode als Lösung ausgeschlossen und es muss nach einem effizienteren Algorithmus gesucht werden.

### 2.3.2 Der Algorithmus

Grundlegend gibt es zwei unterschiedliche Wege eine Mauer, wie in der Aufgabenstellung zu bauen.

Der offensichtlichere ist die Bauart '**von unten nach oben**'. Bei dieser Bauart setzt man zuerst die Klötzchen zu einer Reihe zusammen. Hat man eine Reihe fertig, beginnt man mit der zweiten Reihe und so weiter (siehe Abbildung ). Man setzt Reihe auf Reihe (von unten nach oben) bis man die Maximalhöhe erreicht hat.

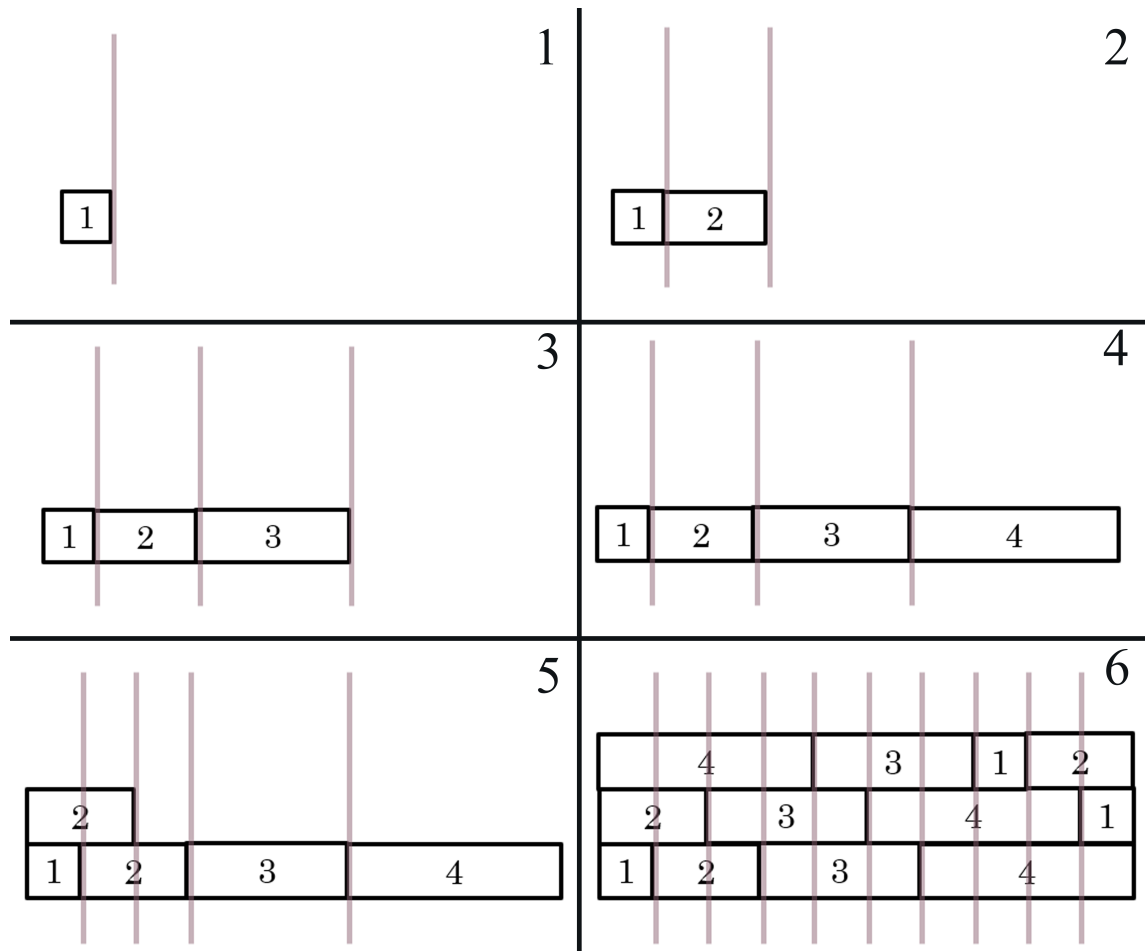


Abbildung 8: Der Bau einer Mauer nach der 'von unten nach oben' Methode. Es wird Reihe nach Reihe gebaut, bis die maximale Höhe erreicht wurde.

Eine andere Möglichkeit ist, wenn man die Anzahl der Reihen ( $h$ ) für eine Mauer der maximalen Höhe bereits weiß, die Mauer **'von links nach rechts'** aufzubauen.

## 2.4 Implementierung

## 2.5 Laufzeitanalyse

## 2.6 Optimierungsmöglichkeiten

## 2.7 Beispiele

## 2.8 Quellcode

### **3 AUFGABE 3 - "Quo vadis, Quax?"**

#### **3.1 Aufgabenstellung**

#### **3.2 Lösungsidee**

#### **3.3 Teilaufgabe (a)**

#### **3.4 Teilaufgabe (b)**

#### **3.5 Teilaufgabe (c)**

##### **3.5.1 Implementierung**

##### **3.5.2 Laufzeitanalyse**

##### **3.5.3 Optimierungsmöglichkeiten**

##### **3.5.4 Beispiele**

#### **3.6 Teilaufgabe (d)**

#### **3.7 Quellcode**

## 4 FAZIT

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed tempus consectetur lorem, imperdiet dignissim est auctor a. Vivamus convallis, leo et iaculis egestas, nunc massa porttitor tellus, id faucibus urna justo eget massa. Praesent quis feugiat odio. Nullam quis mattis enim. Fusce volutpat odio in enim sodales venenatis. Mauris consequat.

## 5 ABBILDUNGSVERZEICHNIS

Alle enthaltenen Abbildungen wurden mithilfe von folgenden Tools selbstständig erstellt:

- Microsoft PowerPoint
- Grafikrechner - GeoGebra
- Adobe Photoshop CC 2018

## **6 LITERATURVERZEICHNIS**

## 7 ERKLÄRUNG DES VERFASSERS

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed tempus consectetur lorem, imperdiet dignissim est auctor a. Vivamus convallis, leo et iaculis egestas, nunc massa porttitor tellus, id faucibus urna justo eget massa. Praesent quis feugiat odio. Nullam quis mattis enim. Fusce volutpat odio in enim sodales venenatis. Mauris consequat.

*Ich erkläre hiermit, dass ich die Seminararbeit ohne fremde Hilfe angefertigt und nur die im Literaturverzeichnis angeführten Quellen und Hilfsmittel benützt habe.*

....., den .....

Ort

Datum

.....

Unterschrift des Verfassers