

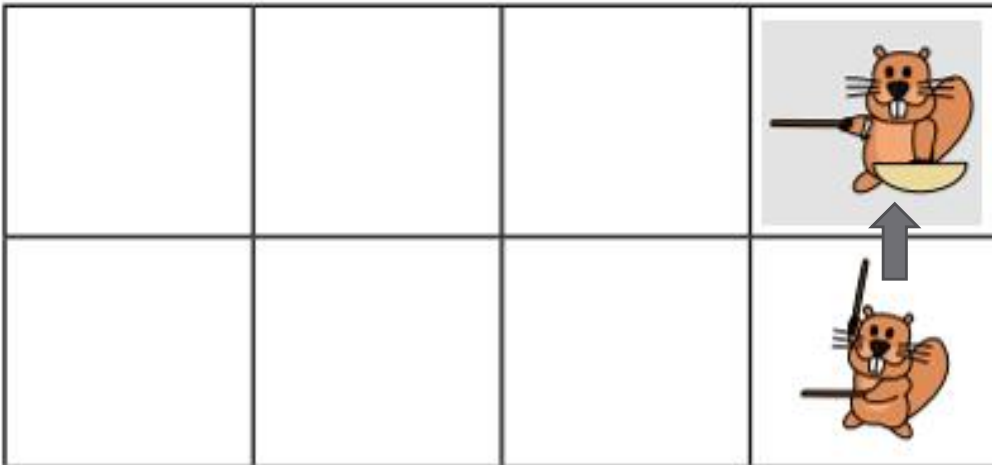
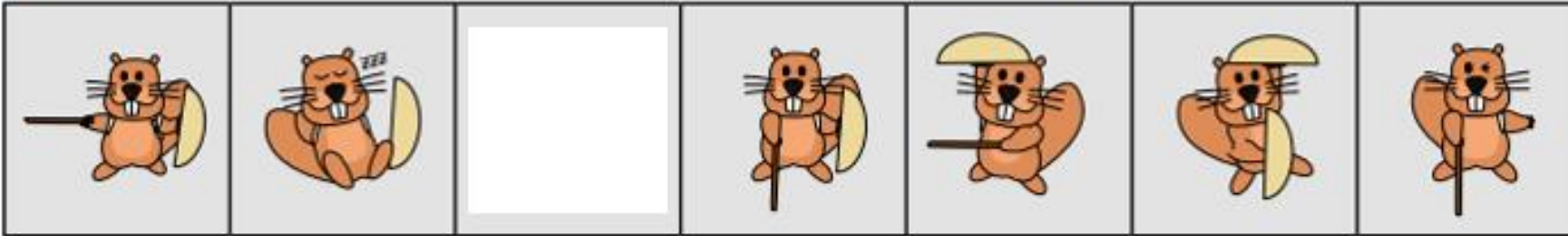
Bundeswettbewerb  
Informatik



**ioi!**

# Stock und Schild

Lucia und ihre Freunde sind Anhänger eines japanischen Spiels mit Stock und Schild. Für ein Foto möchten sie sich auf dem Schulhof so aufstellen, dass jeder Stock auf ein Schild zeigt. Dafür wurden Felder auf den Schulhof gezeichnet. Lucia hat sich bereits in Pose gestellt. Die Bilder darunter zeigen die Freunde in ihren Lieblingsposen.



6! unterschiedliche Möglichkeiten  
 $6 * 5 * 4 * 3 * 2 * 1 = \mathbf{720}$

7 Stöcke  
7 Schilde

Schiebe die Bilder der Freunde in die Felder auf dem Schulhof.

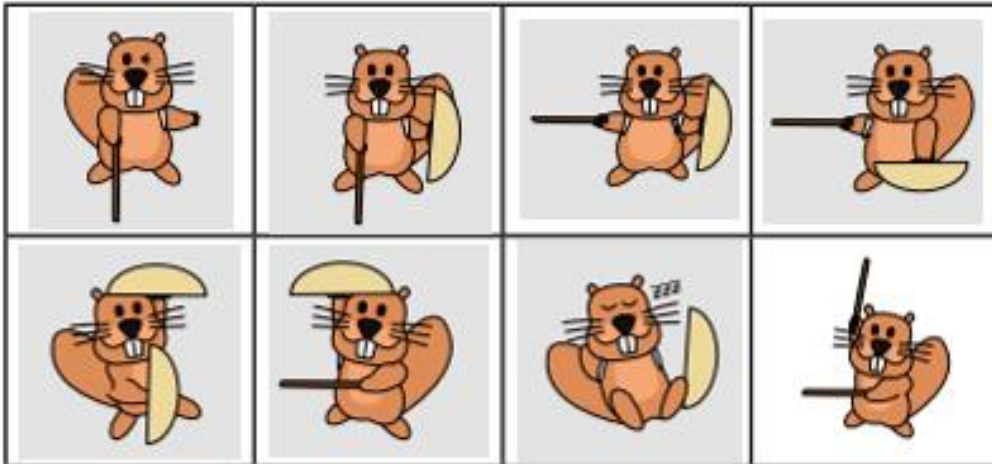
Am Ende muss jeder Stock auf ein Schild zeigen. und jeder Schild auf einen Stock.

# Stock und Schild

Lucia und ihre Freunde sind Anhänger eines japanischen Spiels mit Stock und Schild. Für ein Foto möchten sie sich auf dem Schulhof so aufstellen, dass jeder Stock auf ein Schild zeigt. Dafür wurden Felder auf den Schulhof gezeichnet. Lucia hat sich bereits in Pose gestellt. Die Bilder darunter zeigen die Freunde in ihren Lieblingsposen.



--	--	--	--	--	--	--



6! unterschiedliche Möglichkeiten

$$6 * 5 * 4 * 3 * 2 * 1 = \mathbf{720}$$

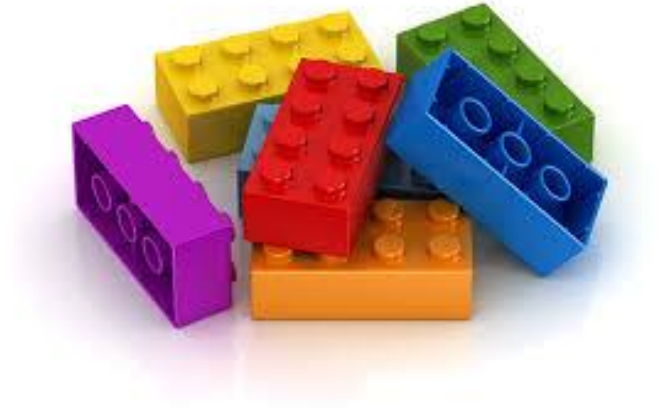
7 Stöcke

7 Schilde

Schiebe die Bilder der Freunde in die Felder auf dem Schulhof.  
Am Ende muss jeder Stock auf ein Schild zeigen.

# 1. Aufgabe der 2. Runde des BwInfs 2017

## Aufgabe 1: Die Kunst der Fuge



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.

$$n = 4$$

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

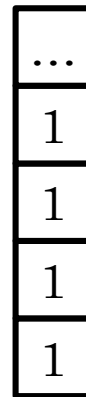
Eine Variable:  **$n$**

## 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  $n$ ?

Kontrolle:  $n = 10 = \text{Mauer der Höhe } 6$

$$n = 1$$



„Die Stellen **zwischen** den Klötzchen“,  
dürfen niemals übereinanderliegen



Für  $n = 1$  **unendlich** hohe Mauer



$$n = 2$$



2	1
1	2

Eine Reihe hat  
einen 1er-Klotz und  
einen 2er-Klotz



$$n = 3$$



2		3	1
1	2		3

$$n = 4$$

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.

$$n = 4$$

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

### Aufgabe:

Bilde eine möglichst hohe Mauer für  $n = 4$

$n = 4$

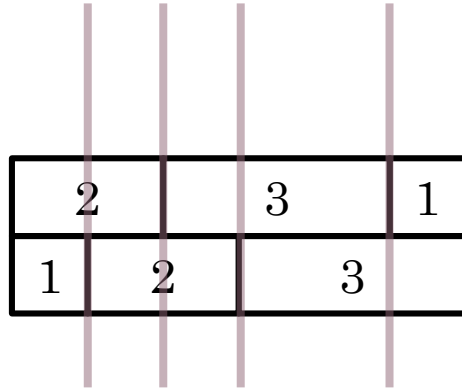


	4			3		1	2
2		3			4		1
1	2		3			4	

1. Keine Fugen überlappen
2. Maximale Höhe (Hier 3)

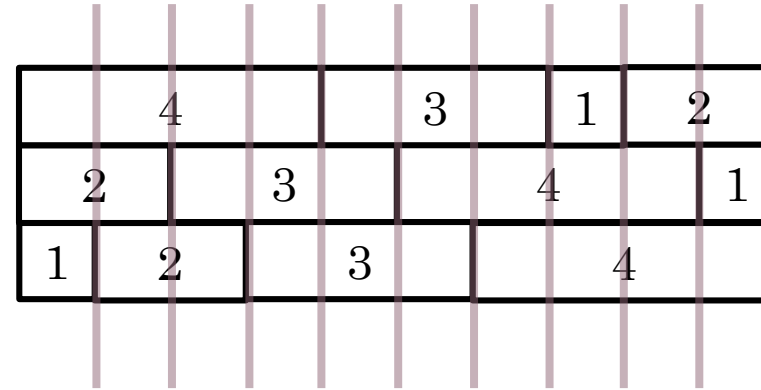
Wann ist die maximale  
Höhe erreicht?

Die maximale Mauerhöhe ist dann erreicht, wenn...



A diagram showing a 2x3 grid of cells. There are four vertical lines: one at the left edge, one between the first and second columns, one between the second and third columns, and one at the right edge. The cells contain the following numbers:

2		3	1
1	2		3



A diagram showing a 3x5 grid of cells. There are nine vertical lines: one at the left edge, and then eight more lines that divide the grid into five columns of width 1. The cells contain the following numbers:

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

...nicht mehr genug Fugenstellen frei sind, um eine weitere Reihe zu bauen.

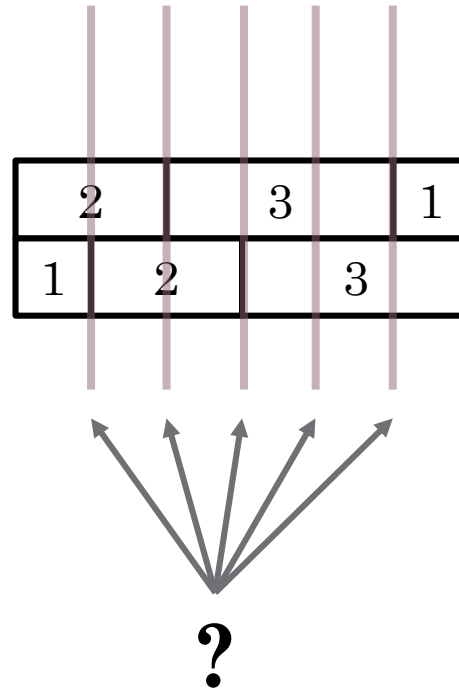
$$\text{Maximale Mauerhöhe} = \frac{\text{Fugenstellen gesamt}}{\text{Fugenstellen pro Reihe}}$$

**Fugenstellen gesamt** = Länge einer Reihe/Mauer – 1

Länge einer Reihe/Mauer =  $1 + 2 + 3 + \dots + (n - 2) + (n - 1) + n$

= Gaußsche Summenformel

$$= \frac{n * (n + 1)}{2}$$



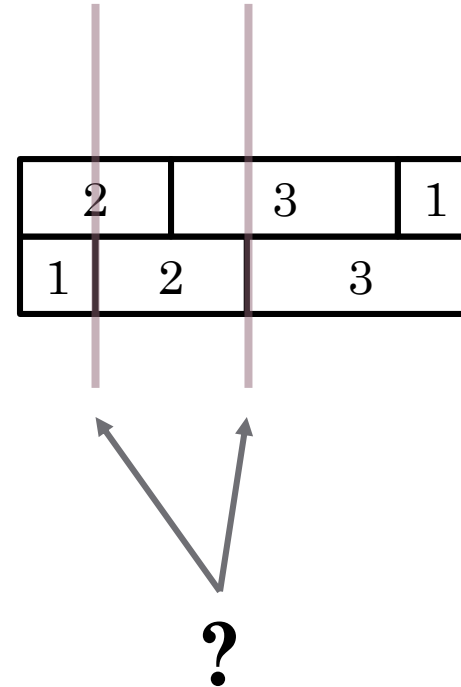
**Fugenstellen gesamt** = Länge einer Reihe/Mauer – 1

Länge einer Reihe/Mauer =  $1 + 2 + 3 + \dots + (n - 2) + (n - 1) + n$

= Gaußsche Summenformel

$$= \frac{n * (n + 1)}{2}$$

**Fugenstellen pro Reihe** =  $n - 1$





$$\text{Fugenstellen gesamt} = \text{Länge einer Reihe/Mauer} - 1$$

$$\text{Länge einer Reihe/Mauer} = 1 + 2 + 3 + \dots + (n - 2) + (n - 1) + n$$

$$= \text{Gaußsche Summenformel}$$

$$= \frac{n * (n + 1)}{2}$$

$$\text{Fugenstellen pro Reihe} = n - 1$$



$$\text{Maximale Mauerhöhe} = \frac{\text{Fugenstellen gesamt}}{\text{Fugenstellen pro Reihe}} = \frac{\frac{n * (n + 1)}{2} - 1}{n - 1}$$

$$= \frac{n}{2} + 1$$

# Kontrolle: $n = 10 =$ Mauer der Höhe 6

GeoGebra Grafikrechner



$$f(n) = \left\lfloor \frac{n}{2} + 1 \right\rfloor$$



$$a = f(10)$$



→ 6

$$b = f(3)$$

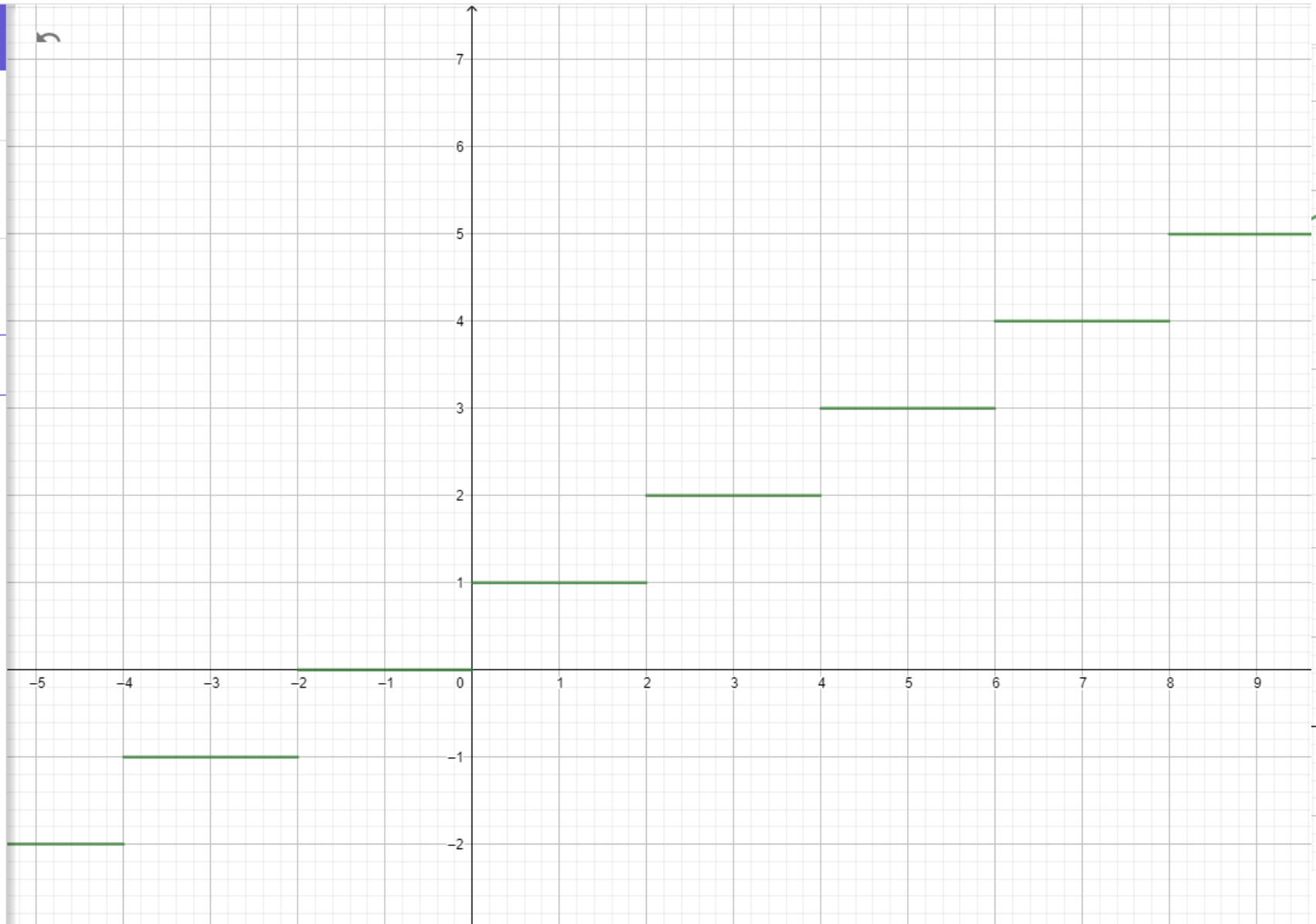


→ 2



Eingabe...

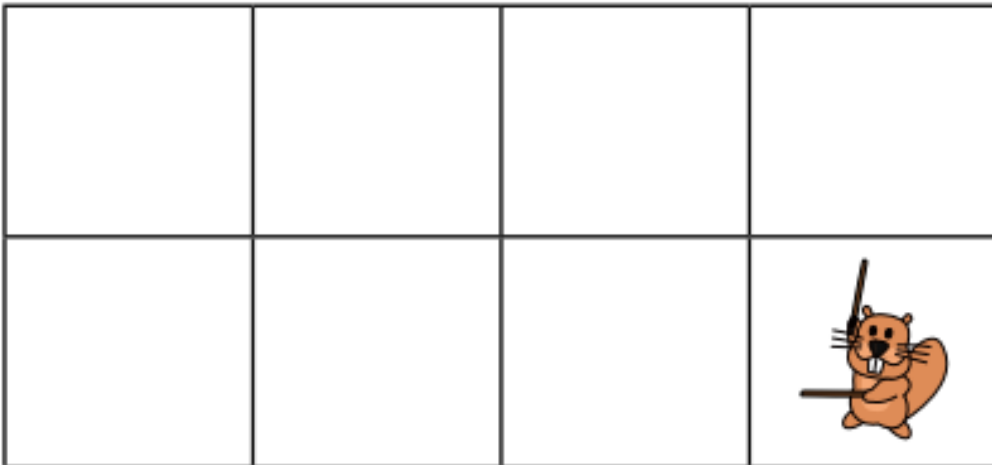
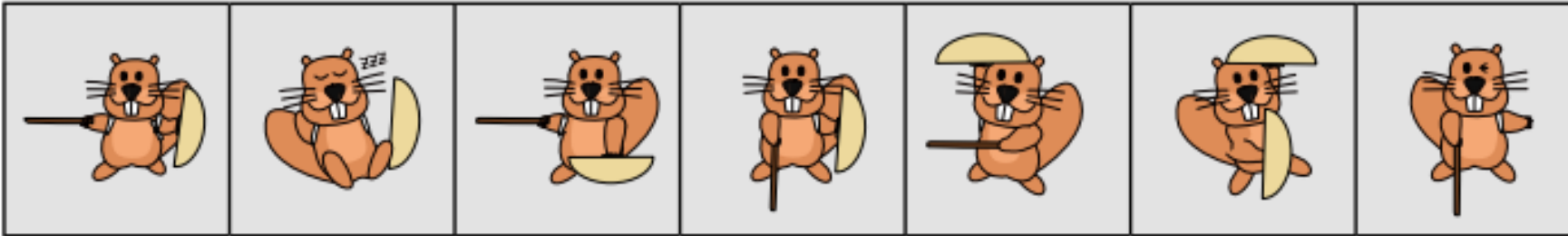
$f(n)$			$n$	
2	3	1		
1	2	3		



Wie baut man jetzt so eine  
Mauer?

# Stock und Schild

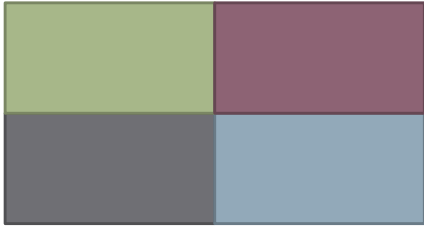
Lucia und ihre Freunde sind Anhänger eines japanischen Spiels mit Stock und Schild. Für ein Foto möchten sie sich auf dem Schulhof so aufstellen, dass jeder Stock auf ein Schild zeigt. Dafür wurden Felder auf den Schulhof gezeichnet. Lucia hat sich bereits in Pose gestellt. Die Bilder darunter zeigen die Freunde in ihren Lieblingsposen.



7! unterschiedliche Möglichkeiten  
 $7 * 6 * 5 * 4 * 3 * 2 * 1 = 5040$

**Schiebe die Bilder der Freunde in die Felder auf dem Schulhof.**  
Am Ende muss jeder Stock auf ein Schild zeigen.

# Anzahl Mauer Variationen



$2! = 2$  Variationen

$2!^2 = 4$  Variationen

*Variationen = (Anzahl Elemente in Reihe)!<sup>Anzahl der Reihen</sup>*

**n = 6**

$6!^4 = 268.738.560.000$

**n = 10**

$10!^6 = 2.283.380.023.591.730.815.784.976.384.000.000.000.000$



**TAKTIK!**



# Der Algorithmus

# Grundlegende Mauerbauarten

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



von links nach rechts

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



von unten nach oben

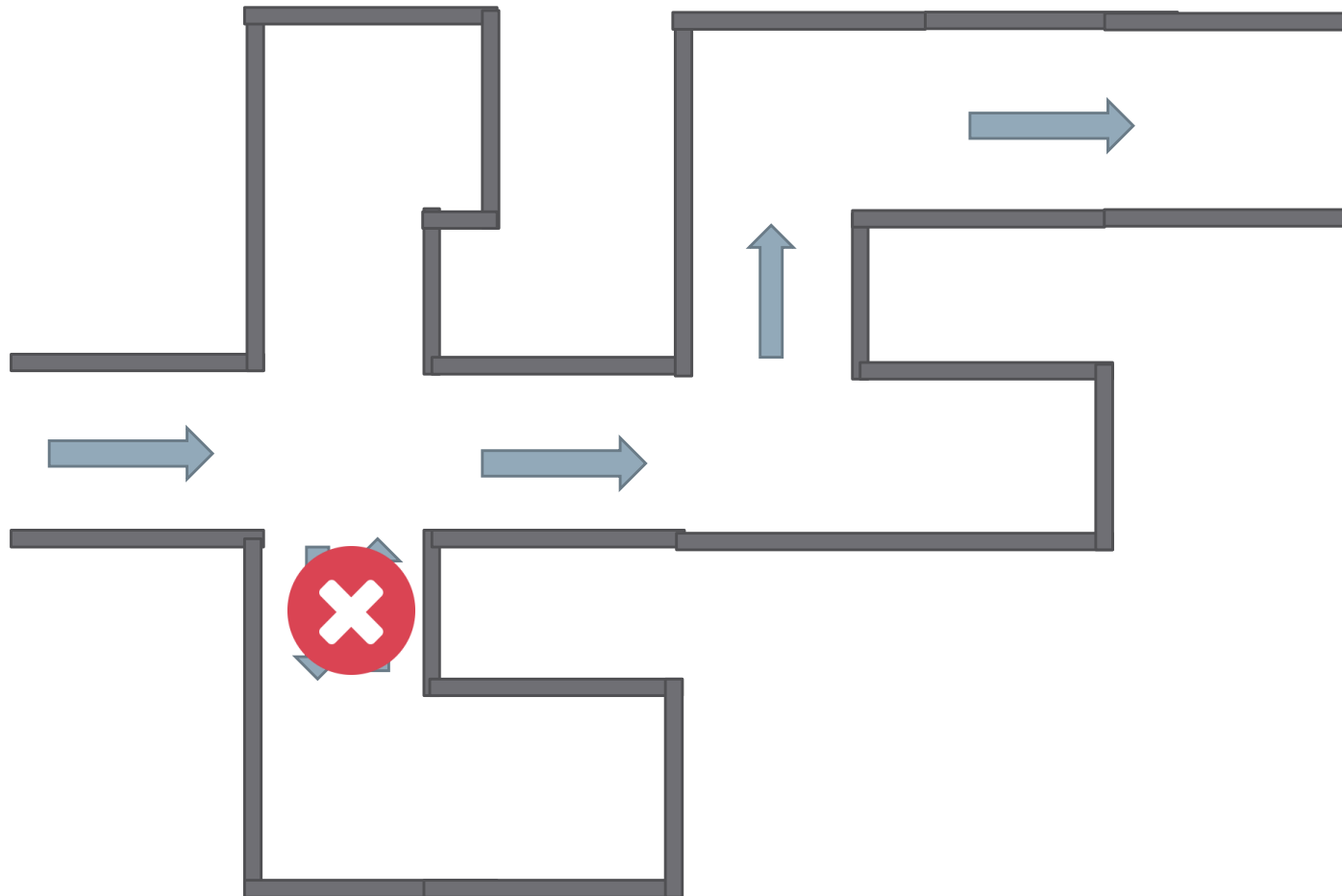


$n = 8$

5	6	3	7	2	1	4	8	
4	5	6	7	2	1	,3	8	
3	5	4	6	7		1,2	8	
2	5	6	4	3	7		1	8
1	5	4	6	3	7		2	8

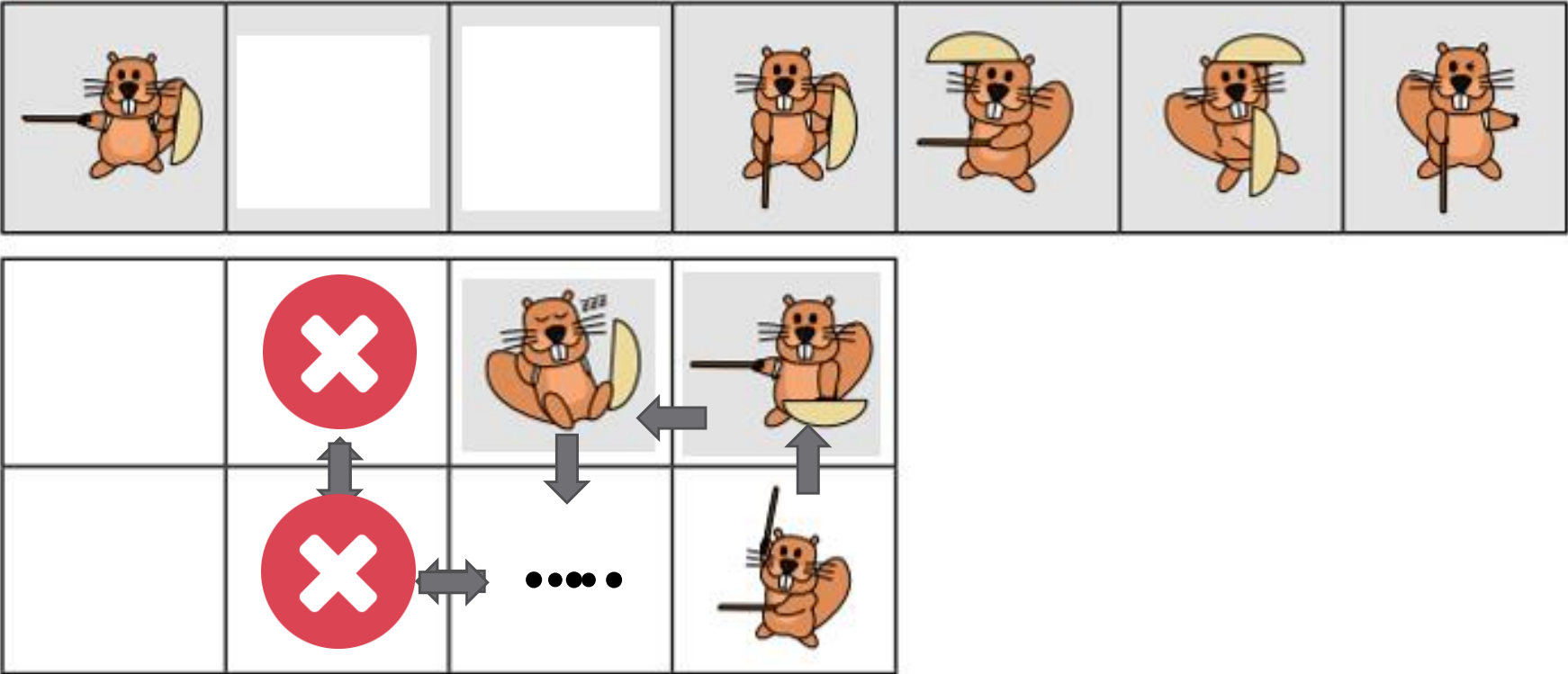


# Lösung: Backtracking



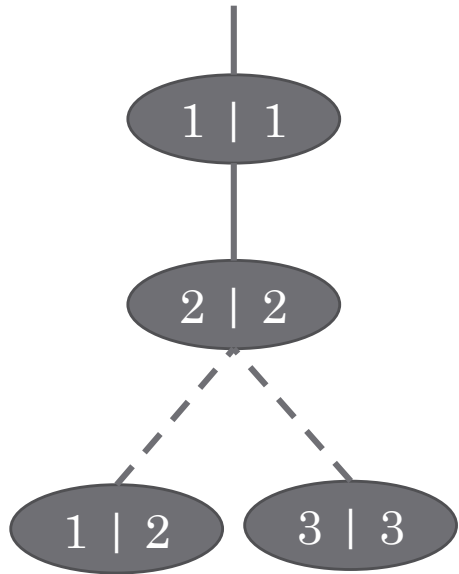
# Stock und Schild

Lucia und ihre Freunde sind Anhänger eines japanischen Spiels mit Stock und Schild. Für ein Foto möchten sie sich auf dem Schulhof so aufstellen, dass jeder Stock auf ein Schild zeigt. Dafür wurden Felder auf den Schulhof gezeichnet. Lucia hat sich bereits in Pose gestellt. Die Bilder darunter zeigen die Freunde in ihren Lieblingsposen.



Schiebe die Bilder der Freunde in die Felder auf dem Schulhof. Am Ende muss jeder Stock auf ein Schild zeigen.

# Mauerbau mit Backtracking

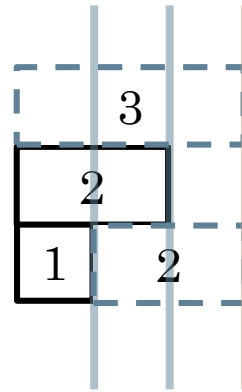


Fuge 1

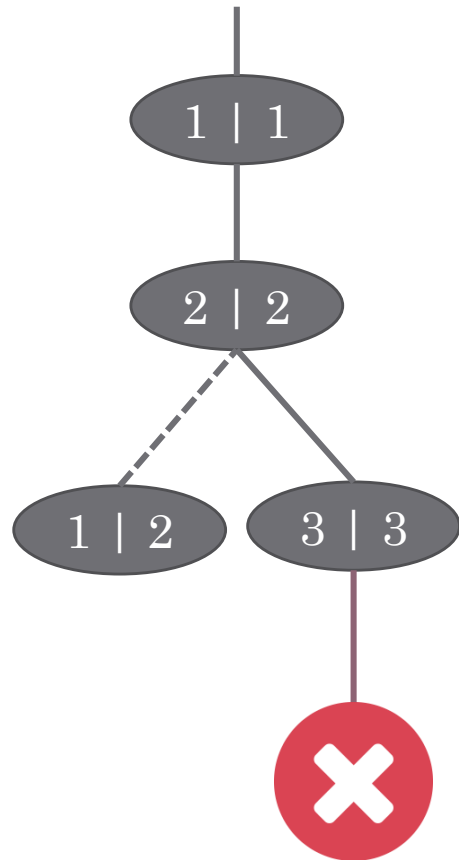
Fuge 2

Fuge 3

**n = 4**



# Mauerbau mit Backtracking



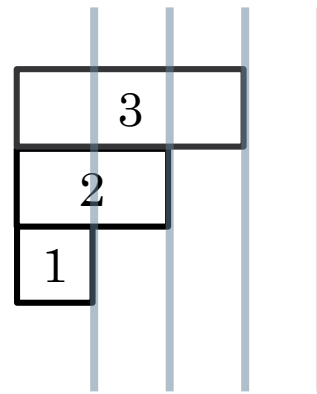
Fuge 1

Fuge 2

Fuge 3

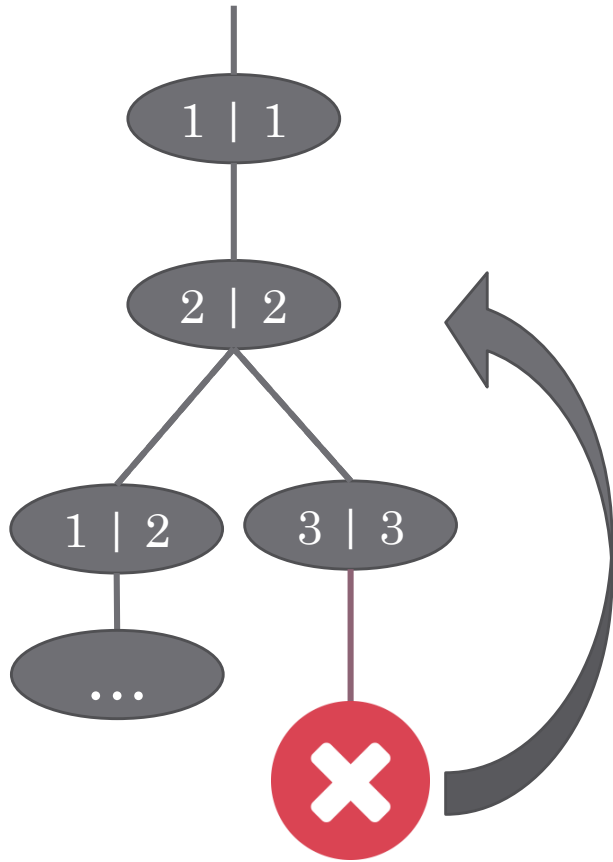
Fuge 4

**n = 4**



„Spalte kann nicht gefüllt werden“

# Mauerbau mit Backtracking



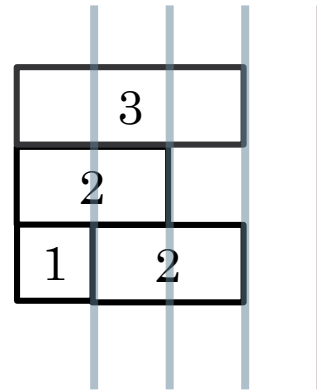
Fuge 1

Fuge 2

Fuge 3

Fuge 4

**n = 4**

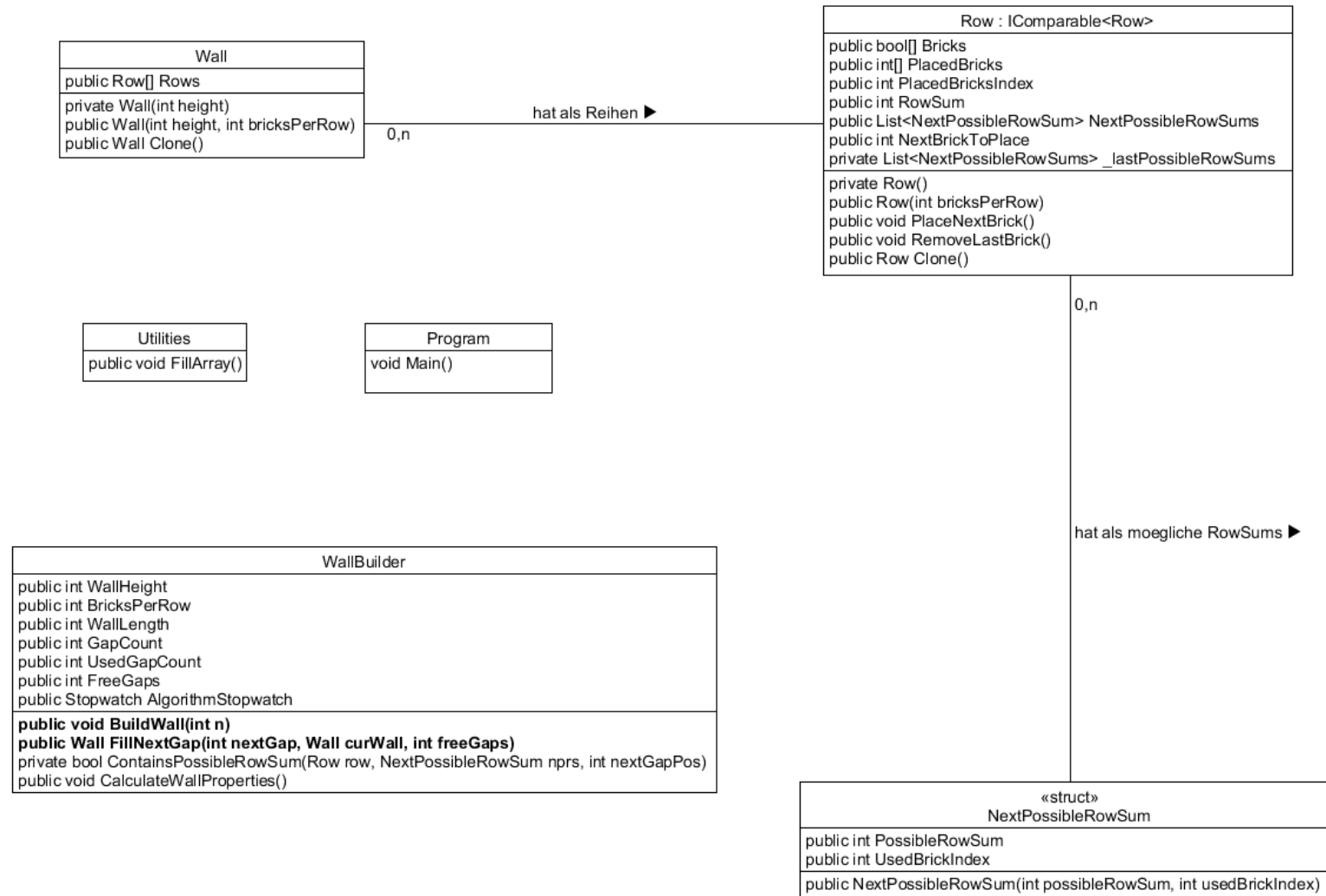


„Spalte kann nicht gefüllt werden“

# Implementierung



# Klassendiagramm



```

/// <summary>
/// Versucht die naechste Luecke in der Mauer zu fuellen
/// </summary>
/// <param name="nextGap">Die Luecke, die als naechstes gefuellt werden muss</param>
/// <param name="wall">Die aktuelle Mauer</param>
/// <returns>Das gueltige Mauer Objekt oder null</returns>
public Wall FillNextGap(int nextGap, Wall wall)
{
    // Ueberpruefe ob die Mauer fertig gebaut ist
    if (wall.Rows.All(r => r.RowSum == WallLength)) // ABBRUCHBEDINGUNG: Haben alle Mauerreihen die fertige Laenge
    {
        // GUELTIGE MAUER GEFUNDEN !!!
        // REKURSIONSENDE
        return wall;
    }

    // Sammle alle Reihen, die die naechste Spalte fuellen koennen
    Row[] possibleRows = wall.Rows.Where(r => r.NextPossibleRowSums.Any(nrs =>
        ContainsPossibleRowSum(r, nrs, nextGap))).ToArray();

    if (possibleRows.Length == 0) // ABBRUCHBEDINGUNG: Es gibt keine Reihe, die die naechste Spalte fuellen kann
    {
        // REKURSIONSENDE
        return null;
    }
}

```

```
// BACKTRACKING
for (int i = 0; i < possibleRows.Length; i++)
{
    // Fuelle die naechste Spalte
    possibleRows[i].PlaceNextBrick();

    // REKURSIVER AUFRUF: Versuche die WIEDER naechste Spalte zu fuellen
    Wall result = FillNextGap(nextGap, wall);

    if (result != null) // ABBRUCHBEDINGUNG: Eine gueltige Mauer wurde zurueckgegeben
    {
        // Die gueltige Mauer weitergeben
        // REKURSIONSENDE
        return result;
    }
    else
    {
        // Falsche Entscheidung! -> Backtracking
        // Den gesetzten Klotz wieder entfernen und...
        possibleRows[i].RemoveLastBrick();

        // ...die naechste Moeglichkeit ausprobieren
    }
}

// REKURSIONSENDE
return null;
```

Unterschiedliche Mauern für  $n = 100$

$$100!^{55}$$

25242760984772180063952995825824081791377826384101299248754940860201854925699825609820627997751429861229708019421770600640447452367838180372066599660848689519937612787032  
39897929565557305658406188077828403838359872312429693269939694557155945243050684793216778179645676221077008168525120236107802923207687660746089015027191396233737615714942  
6650370047569687053488284522800781191095418526620550816690164222768494379362878616117783926559161666054416502736701969556655435213153751613568412473766131824911952984294  
99207713016326556415917205479959671877213986946014995334343787546549206527858009618292408175272213670492690493547533294187147625342031737433132293090946129364162307652423  
64943218211908500104944852488688710471186258327507751479151487183077394904623120426162646576543206032832441619470262430459902877929639995356672581317489001670967598672215  
54196023181498323992459337018698232017317596724026983252520499536712733152223409197989212080854086443341076420701643526637785492260935279513615297677689250716780472504762  
95302484914503834203236692789787273262089486177449839185618352688903227988479264754502261760973289587929500473015296537997027459932006033388484269373815968608975966924870  
70931592154236160247956728319880228096334517530379997370706322048950105895161872940952813257871716163764264420802136974930904488171889566722193353659683120669290431061310  
95124694989576112704482388302139009504251178758732490277963308146554461525741171147246315481655932250311613705857925423157300060765577265994069378839699055771886458086015  
5552191247239477016495485925202742841000315214129281830319424452443735151649802138448361583430957018452188739748788142636950243643624586821798864169241399800006442169032  
99942250754432025681892744617361286904268286699551036151988228654159671934191019496527118337082083329912847608411181190610225825811169604692280851327137578338059763836239  
992723547130027527996252926560934062277652031921885970197134348570283308151771671133782010757868927146154453904490742593521728220802232209509025501280708935750784935190343  
54563390117494866011062556956338620519020335515393770290229632416996359072412760684060588077947938061197004553890409658143168811502861300377004345316146311744206418488719  
09135480786110176736823332818602181844851879235359577645386740449752064198290318765219061384785117493285391530001777301331200586658239612348309402201333020812707455107543  
64007800003871169756553483481335832368813653935160069404127191882173420917801931668128577492392775366372132895442038352299805916326616785115962007242150000279841251608116  
1876814418160432732403676529449669785538385203026216821444512769706186898653346692525511463809854235639188983910134829433560938492901422376663989843910306642813077977200  
00898003690825168316563955088854626312854670591533216270650280582303975195077082263421847644342386935653887992398786205555326013414964945039032370932174372920490124042669  
13016190510668962013221494869137707618535096301471307891227926601466248009393355565117423172677980888706941319312021423510639975312278604638302517341019848329602970605051  
06331401104277109526410123864815859030845473011364556797908447058901864711788730900678827319019893628216160369673246806647528325111226821632713579566502978638612567168002  
017597233268437978587606883166730145176423696043902554161517616312177262403921110960548975426151471093934440483247873519456505810208242849853223672816863753037828833709  
425899083764783814683664999058503629133985698695475045928388841277769604982314679698949391282305044919018444899161888331519559359278844011161662809378966159106471317364  
9413488296465182998289660149651996907073490174512168696882183421573283724762678819803650447271803044373207951782842573292481900571964157172517176747782542240960  
9279479990839143800836658004971699569991349473188218884387762079787970221096328129055149506629913460277789264761027733443332038260780375046808550239387945092780285758252  
01656392250909762147900160822319790335092728381785492207872667036597624602339554552683360574917135765858160416708703758264733058380152148268361611837308116



# Bundeswettbewerb Informatik

[www.bwinf.de](http://www.bwinf.de)

[www.github.com/atalantus/BwInf36\\_Runde02](https://www.github.com/atalantus/BwInf36_Runde02)



# Quellen

- <https://bwinf.de/>
- <http://mathedia.com/analysis/vollstaendige-induktion/gaussssche-summenformel/>
- <https://www.geogebra.org/>
- <https://www.geeksforgeeks.org/backtracking-algorithms/>



# Bildquellen

- [https://static1.bmbfcluster.de/1/8/8/7\\_a71ec38177e2b9b/1887meg\\_75003e924b06111.jpg](https://static1.bmbfcluster.de/1/8/8/7_a71ec38177e2b9b/1887meg_75003e924b06111.jpg)
- [https://bwinf.de/fileadmin/user\\_upload/Informatik-Biber/2017/Biber-Aufgabenheft\\_2017/Biberheft\\_2017.pdf](https://bwinf.de/fileadmin/user_upload/Informatik-Biber/2017/Biber-Aufgabenheft_2017/Biberheft_2017.pdf)
- <http://www.ypool.de/wp-content/uploads/2014/09/Bundeswettbewerb-Informatik-Logo.png>
- [https://cdn3.iconfinder.com/data/icons/flat-actions-icons-9/792/Tick\\_Mark\\_Dark-512.png](https://cdn3.iconfinder.com/data/icons/flat-actions-icons-9/792/Tick_Mark_Dark-512.png)
- [https://upload.wikimedia.org/wikipedia/commons/thumb/8/8f/Flat\\_cross\\_icon.svg/2000px-Flat\\_cross\\_icon.svg.png](https://upload.wikimedia.org/wikipedia/commons/thumb/8/8f/Flat_cross_icon.svg/2000px-Flat_cross_icon.svg.png)
- [https://daserwachendervalkyrjar.files.wordpress.com/2017/09/shutterstock\\_462881602-696x465.jpg](https://daserwachendervalkyrjar.files.wordpress.com/2017/09/shutterstock_462881602-696x465.jpg)
- <https://pixabay.com/de/icon-isoliert-kunst-einkaufen-1641915/>
- [https://www.flaticon.com/free-icon/github-logo\\_25231](https://www.flaticon.com/free-icon/github-logo_25231)



Danke für Eure Aufmerksamkeit!