# Anmerkungen zu zwei Aufgaben aus dem diesjährigen Bolyai-Wettbewerb - Rechnen, bis der Arzt kommt?

Der Bolyai-Mathematikteamwettbewerb[1] wurde 2014 das erste Mal in Deutschland durchgeführt und hat seitdem an vielen Schulen zahlreiche Interessierte gefunden, wie man auf der Homepage nachlesen kann.
Der Teamwettbewerb wird in einer vorgegebenen Zeit i.d.R. an der Schule geschrieben und als Hilfsmittel dürfen bis zur Klassenstufe 8 nur Zeichengeräte genutzt werden, insbesondere Taschenrechner und Smartphones sind untersagt. In den Wettbewerbsstufen 9 bis 12 ist ein nicht programmierbarer Taschenrechner zugelassen.

Im Wettbewerb 2021 der Klassenstufe 12 gab es Aufgaben, die wir hier aufgreifen wollen, um zu zeigen, wie man diese Aufgaben nutzen kann, um Schülerinnen und Schüler mit verschiedenen mathematisch inhaltlichen sowie rechentechnischen Problemen vertraut zu machen. Dabei wird der Rahmen des Wettbewerbs verlassen und auch die Zielgruppe Klassenstufe 12 muss dann nicht von Bedeutung sein.

## Zur Aufgabe 10 Klassenstufe 12

**10.** Die Folge $(a_n)$ wird folgendermaßen definiert:

$$a_1 = k \text{ (positive ganze Zahl)}, \quad a_{n+1} = \begin{cases} \frac{a_n}{2}, \text{ wenn } a_n \text{ gerade} \\ a_n + 5, \text{ wenn } a_n \text{ ungerade.} \end{cases}$$

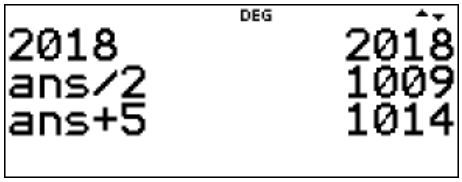
Bei welchen der vorgegebenen Werten für $k$ kommt dann unter den Folgengliedern die Zahl 1 vor?

**(A)** *2018* **(B)** *2019* **(C)** *2020* **(D)** *2021* **(E)** *2022*

(Die richtigen Lösungen sind anzukreuzen. Es kann auch mehr als eine Antwort richtig sein.)

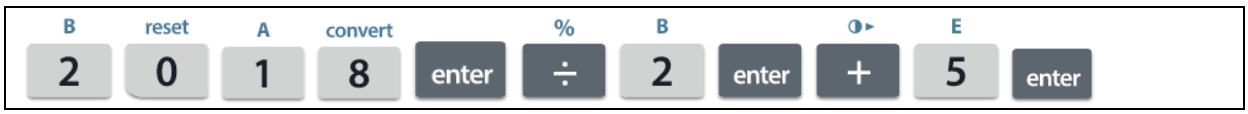
Die einfache „Probierlösung“ mit einem wissenschaftlichen Taschenrechner (hier mit dem TI-30X Plus MathPrint) wird nun erläutert:

Man nutzt die Möglichkeit, mit der „letzten Antwort“ zu arbeiten. Der Start erfolgt mit der Eingabe der gegebenen Zahl. Die Eingabe wird mit ENTER bestätigt. Wenn die angezeigte Zahl gerade ist, wird sie durch 2 dividiert, ist sie ungerade, werden 5 addiert. Die Entscheidung darüber und die Ausführung gehen stets unmittelbar vom Nutzer aus.
Gibt man diese Rechenoperation ein, so wird sie automatisch auf die „letzte Antwort“, also die darüberstehende Zahl angewendet. Im Folgenden wird die Verfahrensweise für die Startzahl 2018 in den ersten Schritten durch nebenstehenden Bildschirmabdruck und die zugehörige Tastenfolge dokumentiert.



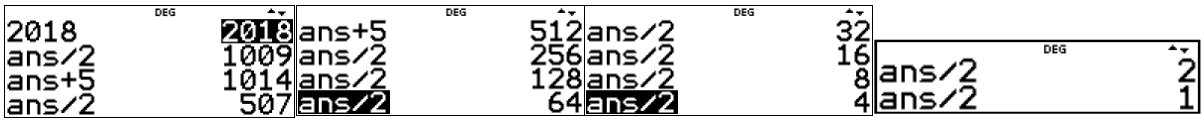

[1] https://www.bolyaiteam.de/

Tastenfolge:




Das Verfahren wird solange fortgesetzt, bis man auf die 1 oder die 5 als Ergebnis kommt.
Würde man es weiter fortsetzen, ergäbe sich eine zyklische Wiederholung der Zahlenfolgenglieder:
1 → 6 → 3 → 8 → 4 → 2 → 1 bzw. 5 → 10 → 5
Diese zyklischen Wiederholungen bewirken, dass die Zahlenfolgen unendlich viele Glieder haben und man deshalb rechnen könnte, „bis der Arzt kommt."

Hier wird das Verfahren für die Startzahl 2018 vollständig dokumentiert, bis zum ersten Mal die Zahl 1 erscheint:




Wendet man dieses Verfahren auf alle Zahlen an, die in der Aufgabenstellung gegeben sind, so ist zu erkennen, dass die Zahl 1 lediglich bei der Startzahl 2020 nicht in der Zahlenfolge vorkommt. Anzukreuzen sind also die Antwortmöglichkeiten A, B, D und E.

Lösung des Wettbewerbsausrichters[2]:

**Lösung:** Ist $k$ durch 5 teibar, dann sind auch die Folgenglieder durch 5 teilbar, vergleiche hierzu das Bildungsgesetz. Also kann 1 nicht Glied der Folge sein. Des Weiteren setzen wir voraus, dass $k$ nicht durch 5 teilbar ist und damit auch kein Glied der Folge. In diesem Fall gibt es unter den Folgengliedern (positive ganze Zahlen) ein Glied, das am kleinsten ist. Dieses Glied wird mit $m$ bezeichnet. Wir wissen, dass $m$ nicht durch 5 teilbar ist, es muss aber ungerade sein, denn sonst wäre $\frac{m}{2}$ als Nachfolger kleiner als $m$, was einen Widerspruch bedeuten würde. Also kann das nächste Folgenglied gerade und $m+5$ heißen, danach folgt das ungerade Glied $\frac{m+5}{2}$. Der Vergleich $m \le \frac{m+5}{2}$ zeigt, dass $m \le 5$ ist, woraus direkt folgt, dass für $m$ nur 1 und 3 infrage kommen. 3 kann nicht akzeptiert werden, denn für $a_n = 3$ ist $a_{n+3} = 3$, also kommt später ein kleineres Glied vor, was der Voraussetzung widerspricht. So bleibt die Zahl 1, die genau dann Folgenglied ist, wenn $k$ nicht durch 5 teilbar ist.

**Richtige Antwort(en): A, B, D, E**

[2] https://www.bolyaiteam.de/, Stand 03.03.2021, 16:15 Uhr

Wir wollen die Grundidee, die in der Aufgabe steckt, nun ein wenig ausbauen:

## 1. Kopfrechnen

Die Aufgabenstellung lässt sich z. B. dadurch variieren, dass die Folge $(a_n)$ folgendermaßen definiert wird:

$$a_1 = k \text{ (positive ganze Zahl)}, \; a_{n+1} = \begin{cases} \frac{a_n}{2}, \text{ wenn } a_n \text{ gerade} \\ a_n + c, \text{ wenn } a_n \text{ ungerade} \end{cases}$$

Dabei sei c eine ungerade Zahl größer als 1.

Vielleicht ab Klassenstufe 6, wenn die Schüler Kenntnisse über die Teilbarkeit erworben haben, können Untersuchungen dazu angestellt werden, ohne dass Rechenhilfsmittel Verwendung finden. Für zweistellige Startzahlen und z. B. c = 3 könnte die Aufgabe etwa so gestellt werden.
(Die Schüler rechnen im Kopf und notieren nacheinander die Ergebnisse.)

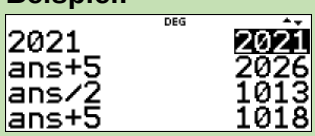
*Notiere eine zweistellige Startzahl. Führe nun folgende Rechnungen aus.*
*(1) Wenn die Startzahl ungerade ist, addiere 3.*
*Wenn die Startzahl gerade ist, dividiere sie durch 2.*
*(2) Notiere die entstandene Zahl und berechne mit ihr eine neue Zahl:*
*Wenn sie ungerade ist, addiere 3; wenn sie gerade ist, dividiere sie durch 2.*
*(3) Gehe zu Schritt 2.*
*(4) Beende deine Rechnung, wenn du den Eindruck hast, dass sich Ergebnisse wiederholen.*
*(5) Wähle andere gerade oder ungerade zweistellige Startzahlen und wiederhole mit jeder von ihnen dieses Verfahren.*
*(6) Betrachte die Zahlenfolgen, die durch deine Rechnungen entstanden sind. Versuche, verschiedene Fälle zu unterscheiden.*
*(7) Beschreibe deine Beobachtungen.*
*(8) Kannst du deine Beobachtungen erklären?*

**Beispiel:**
Startzahl 38
38 : 2 = 19
19 + 3 = 22
22 : 2 = 11
11 + 3 = 14
…

## 2. Simulation mit einem WTR

Wenn die Schüler als Rechenhilfsmittel einen wissenschaftlichen Taschenrechner (WTR) verwenden dürfen, kann man die Startzahlen auch größer wählen und sie mit der auf Seite 1 beschriebenen Methode vertraut machen.
Für dreistellige Startzahlen und z. B. c = 5 könnte die Aufgabe so gestellt werden, wie oben beschrieben.
(Die Schüler rechnen mit dem WTR oder im Kopf und notieren nacheinander die Ergebnisse.)

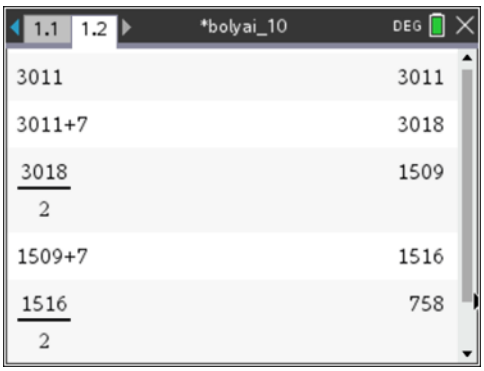
**Beispiel:**




…

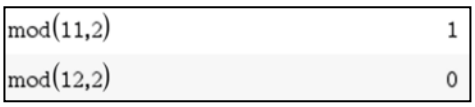
Dasselbe Verfahren wie oben beschrieben, lässt sich auch mit dem CAS-Rechner TI-Nspire durchführen. (Hier im Beispiel für c = 7.)

In der Anwendung *Calculator* wird die Startzahl eingegeben und mit enter bestätigt.
Bei einer geraden Zahl wird eingegeben ÷ 2, bei einer ungeraden Zahl + 7, diese Rechnungen werden auf die jeweils „letzte Antwort“ automatisch angewendet.

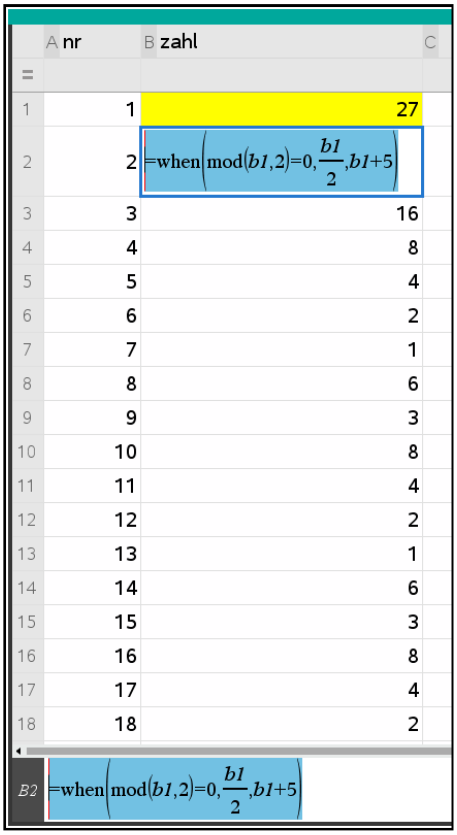



## 3. Tabellenkalkulation des TI-Nspire verwenden

Komfortabler wird das Verfahren, wenn die Tabellenkalkulation verwendet wird. Man kann außerdem dem CAS-Rechner „abverlangen“, dass er selbst entscheidet, ob eine Zahl gerade oder ungerade ist und damit die notwendige Rechnung ausführt. Für die Entscheidung „gerade Zahl“ oder „ungerade Zahl“ kann die Anweisung „mod“ für die Modulo-Rechnung, also die Rechnung mit Zahlenkongruenzen verwendet werden. Dies bedeutet, kurz gesagt:
Bezüglich der Teilbarkeit durch 2 lassen sich die ganzen Zahlen in die Restklassen 0 und 1 unterteilen, je nachdem, ob die ganze Zahl bei Division durch 2 den Rest 0 oder den Rest 1 lässt.
Da z. B. die Zahl 11 bei Division durch 2 den Rest 1 lässt, gibt die Anweisung *mod(11, 2)* den Wert 1, und die gerade Zahl 12 mit *mod(12, 2)* den Wert 0 zurück, weil 12 durch 2 teilbar ist.




In der Tabellenkalkulation kann das z. B. folgendermaßen realisiert werden:
In der Spalte A werden die Schrittfolgen gezählt.
In der Spalte B wird in B1 ein Startwert eingegeben. In B2 sorgt der Befehl
**=when(mod(b1,2)=0,b1/2,b1+5)** dafür, dass die nächste Zahl mittels Division durch 2 erzeugt wird, falls die Zahl in B1 gerade war, sonst aber (Zahl in B1 ungerade) durch Addition von 5 entsteht.



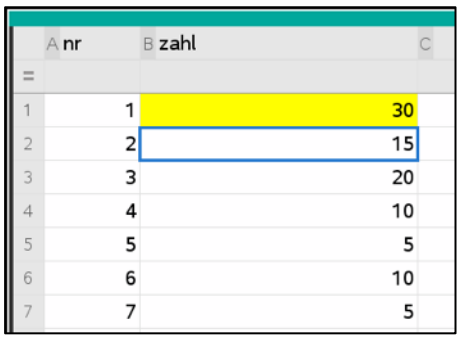

In der Anwendung *Lists&Spreadsheet* werden folgende Eingaben getätigt und mit enter bestätigt:

Zelle A1: 1
Zelle B1: 27 (Startzahl)
Zelle A2: **= a1 + 1**
Zelle B2: **=when(mod(b1,2)=0,b1/2,b1+5)**

Die Zellen A2 und B2 werden dann mit ⇧shift markiert und die Befehle mit menu *Daten – Füllen* bis in die Zeile 30 (oder mehr) als relative Zellbezüge kopiert.

Für die Startzahl 27 ist gut zu erkennen, dass im 7. Schritt die Zahl 1 erstmalig erreicht wird, und dann der Zyklus 1-6-3-8-4-2 einsetzt.

Es genügt, die Startzahl zu ändern und mit enter zu bestätigen, um die Untersuchungen für diese neue Startzahl durchzuführen. Wenn die Startzahl sehr groß ist, müssen ggf. die Befehle aus A2 und B2 noch weiter nach unten kopiert werden.



| | A nr | B zahl | C |
|---|---|---|---|
| = | | | |
| 1 | 1 | 30 | |
| 2 | 2 | 15 | |
| 3 | 3 | 20 | |
| 4 | 4 | 10 | |
| 5 | 5 | 5 | |
| 6 | 6 | 10 | |
| 7 | 7 | 5 | |

Ist die Startzahl durch 5 teilbar, so ergibt sich ein anderer Zyklus und die Zahl 1 taucht nicht in der Zahlenfolge auf.

## 4. Programmierung mit TI-Basic

Analog zum Vorgehen in der Tabellenkalkulation lässt sich der Test für die Zahlen auch durch Programme realisieren. Für den TI-Nspire kommen dafür die Programmiersprachen TI-Basic und Python in Frage.

Mit dem Programmnamen **test** werden die zu testende **zahl** und die Anzahl **n** der geplanten Durchläufe eingegeben.

Lokale Variable festlegen

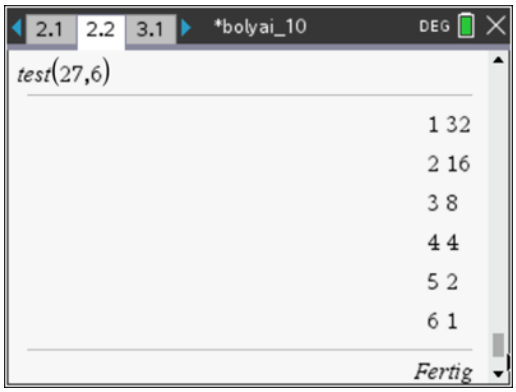
In n Schritten wird geprüft, ob die aktuelle Zahl durch 2 teilbar ist und es wird festgelegt, wie die neue Zahl berechnet wird.

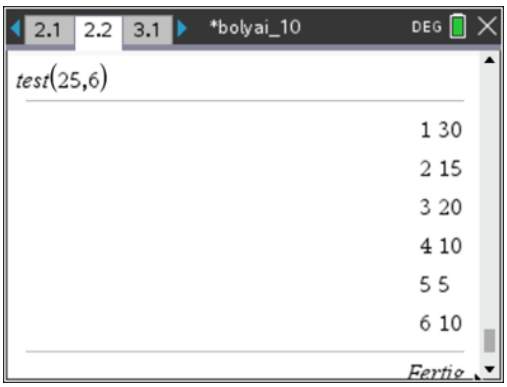
Nach jedem Durchlauf wird die Nummer i des Durchlaufs und die aktuelle Zahl angezeigt.

Der Nutzer muss entscheiden, ob in der Zahlenfolge eine 1 vorkommt oder nicht.

```
"test" erfolg. gespeichert
Define test(zahl,n)=
Prgm
Local i,n,zahl
For i,1,n
  If mod(zahl,2)=0 Then
    zahl:=zahl/2
  Else
    zahl:=zahl+5
  EndIf
  Disp i,zahl
 EndFor
EndPrgm
```

Die untenstehenden Screenshots zeigen Realsierungen für zwei einfache Beispiele: Zahl = 27 bzw. Zahl = 25 und jeweils sechs Durchläufe




Etwas problematisch ist bei dem obigen Programm, dass man i. A. vorher die Anzahl n der Durchläufe nicht kennt, die man für eine Entscheidung im Sinne der Aufgabenstellung braucht. Man kann im Falle, dass noch keine zyklische Wiederholung von Ergebnissen vorliegt, sich damit behelfen, dass man einen neuen Programmdurchlauf mit einem höheren Wert für n startet.

Sinnvoller ist deshalb eine Programmierung, die es dem Nutzer gestattet, das Programm z. B. durch Drücken der Taste d manuell zu beenden, sobald er das wünscht, z. B. dann, wenn er eine zyklische Wiederholung von Ergebnissen beobachtet hat. Dazu kann das folgende Programm dienen.
Wichtig ist, dass ein manueller Programmabbruch mittels *getkey()*-Befehl im Programm eingebaut wird, ansonsten ist hier eine *while*-Schleife der Kern des Algorithmus, da die Anzahl der Durchläufe für beliebige k nicht bekannt ist.

```
zfolge
Define zfolge(k)=
Prgm
Local key,ct,k,i
key:=" "
ct:=1
While k≠1 and key≠"esc"
 key:=getKey()
 ct:=ct+1
 If mod(k,2)=0 Then
  k:=intDiv(k,2)
 Else
  k:=k+5
 EndIf
Disp k
EndWhile
Disp k
Disp ct
EndPrgm
```

## 5. Programmierung mit Python

Der TI-Nspire-CAS verfügt über zwei Programmiersprachen (TI-Basic und ab der Version 5.2 zusätzlich Python), welche schnell erlernbar und auch im Mathematikunterricht einsetzbar sind.
Auf der Webseite

https://education.ti.com/en/activities/ti-codes/python/TI-Nspire

findet man einen einfachen Einstieg in Python, es werden kleinere und auch weiterführende Python-Programme vorgestellt, die die Verwendung der allgemeinen Grundstrukturen Sequenz, Verzweigung und Zyklus veranschaulichen und zum Weiterprobieren anregen sollen.

Die Wettbewerbsaufgabe kann man z. B. nun auch mittels des folgenden kleinen Pythonprogramms (Autor Veit Berger) bearbeiten.

zahlenfolge1.py

```
from random import *
from ti_system import *

def zahlenfolge(k):
    key = ""
    count = 1
    while k != 1 and key != "esc":
        key = get_key(0)
        count += 1
        if k % 2 == 0:
            k = k // 2
        else:
            k = k + 5
        print(k)
    return [k, count]
```

```
>>>zahlenfolge(2018)
1009
1014
507
512
256
128
64
32
16
8
4
2
1
[1, 14]
>>>
```

Die Eingabe von *zahlenfolge(2020)* läuft in eine Endlosschleife, die man nur mittels „esc“ abbrechen kann, falls die Folge nicht terminiert.
Auch die verallgemeinerte Fragestellung ist auf diese Art lösbar. Eine weitere Programmiervariante finden Sie in der Datei zahlenfolge2.tns (Autor Sebastian Rauh).

## 6. Bemerkungen zur Verwendung im Mathematikunterricht

Je nachdem, welche Rechenhilfsmittel zugelassen bzw. verfügbar sind und in welcher Klassenstufe man das Problem untersucht, können solche Simulationen dazu dienen, eine Vermutung über die beobachteten Zahlenfolgen zu gewinnen.

Schon nach wenigen Beispielen werden Schüler erkennen, dass man in Abhängigkeit von den Primfaktoren der Startzahl vorhersagen kann, in welchem Zyklus die Zahlenfolge enden wird.

Setzt man im Unterricht den CAS-Rechner ein, so kann man vielleicht ab der Klassenstufe 8 auch eine Programmiersprache verwenden, um die Simulationen zu realisieren, ggf. auch nur für interessierte Schüler in differenzierter Arbeit.

## Zur Aufgabe 9 Klassenstufe 12

Wechselspiel zwischen Simulation und Theorie

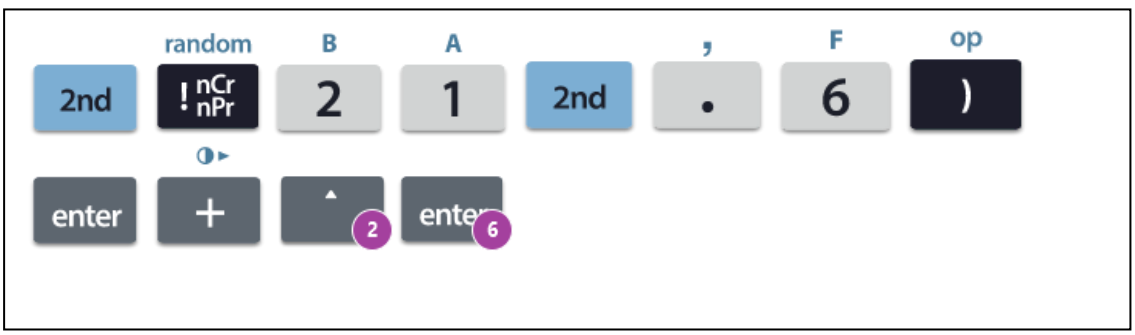
> **9.** Wir würfeln mit einem nicht gefälschten Würfel (Laplace-Würfel) so lange, bis die Summe $S$ der gewürfelten Zahlen 100 übersteigt. Welcher Wert von den gegebenen ist die wahrscheinlichste Summe?
>
> **(A)** *101* **(B)** *102* **(C)** *103* **(D)** *104* **(E)** *105*

Spannend an der Aufgabe war zunächst, dass viele Schüler als auch Lehrer versuchten, mit dem Erwartungswert (3,5) zu argumentieren und sowohl 101, 102 und 105 als wahrscheinlichste Lösung angaben.
Andere wiederum argumentierten damit, dass alle Zahlen gleich wahrscheinlich wären, da es sich ja um einen Laplace-Würfel handelt, bei dem die Wahrscheinlichkeiten für die Würfelzahlen 1bis 6 gleich sind und damit auch diejenigen für die kumulierten Summen 101 bis 106.
Hier lag es also nahe, Simulationen zu probieren. Ein erster Zugang ohne Programmierung ist z. B. auch hier wieder mit dem WTR oder einer Tabellenkalkulation denkbar.

### 1. Simulationen mit WTR-Zufallszahlen


Mit randint(1, 6) wird mit einem WTR eine „Würfelzahl“ aus dem Intervall [1; 6] erzeugt. Dann wird mit „der letzten Antwort“ weitergemacht. Man drückt die Taste „+“ und kopiert dann den Befehl randint(1,6) aus der Zeile 1 in die Zeile 2. (oder man gibt ihn den Befehl nochmal ein.) Dann braucht man nur noch wiederholt die Taste enter zu drücken und erzeugt so die kumulierte Summe der Würfelzahlen, bis die Summe 100 erstmals überschritten wird.

Tastenfolge:



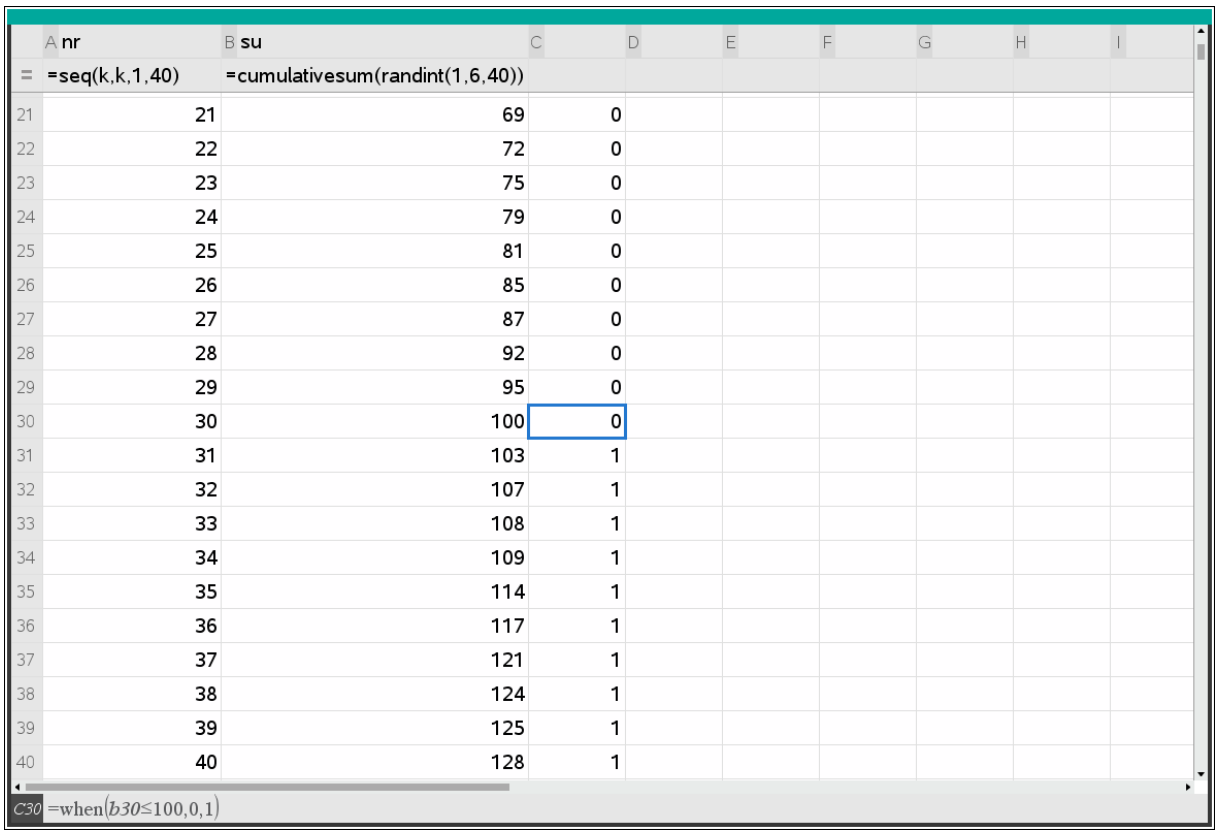
Mögliche Aufgabenstellungen:

Notiert die Zwischenergebnisse in der Tabelle (Partnerarbeit):

| Serie 1 | Serie 2 | Serie 3 | Serie 4 | Serie 5 | Serie 6 | Serie 7 | Serie 8 | Serie 9 | Serie 10 |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |
| | | | | | | | | | |
| *Hinweis: Man sollte ein Arbeitsblatt mit ca. 35 Zeilen vorbereiten*. | | | | | | | | | |

a) Ermittelt in Partnerarbeit die relativen Häufigkeiten für die Summen 101 bis 105.
b) Fasst die Ergebnisse der Klasse zusammen. Ergibt sich daraus eine Vermutung?
c) Welche Zahlen kommen als vorletzte Summe in Frage, bevor man die 100 überschreitet?
d) Wenn z. B. die letzte Zahl vor der Überschreitung der 100 die 95 ist, wie groß ist dann die Wahrscheinlichkeit, als Summe die 101, 102, 103, 104, 105 mit dem nächsten Wurf zu erreichen?

## 2. Simulation mit der Tabellenkalkulation des CAS-Rechners



| | A nr | B su | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| = | =seq(k,k,1,40) | =cumulativesum(randint(1,6,40)) | | | | | | | |
| 21 | 21 | 69 | 0 | | | | | | |
| 22 | 22 | 72 | 0 | | | | | | |
| 23 | 23 | 75 | 0 | | | | | | |
| 24 | 24 | 79 | 0 | | | | | | |
| 25 | 25 | 81 | 0 | | | | | | |
| 26 | 26 | 85 | 0 | | | | | | |
| 27 | 27 | 87 | 0 | | | | | | |
| 28 | 28 | 92 | 0 | | | | | | |
| 29 | 29 | 95 | 0 | | | | | | |
| 30 | 30 | 100 | 0 | | | | | | |
| 31 | 31 | 103 | 1 | | | | | | |
| 32 | 32 | 107 | 1 | | | | | | |
| 33 | 33 | 108 | 1 | | | | | | |
| 34 | 34 | 109 | 1 | | | | | | |
| 35 | 35 | 114 | 1 | | | | | | |
| 36 | 36 | 117 | 1 | | | | | | |
| 37 | 37 | 121 | 1 | | | | | | |
| 38 | 38 | 124 | 1 | | | | | | |
| 39 | 39 | 125 | 1 | | | | | | |
| 40 | 40 | 128 | 1 | | | | | | |

C30 =when(b30≤100,0,1)

Aufgaben:

a) Beschreibe die auf den Screenshots dargestellte Simulation im Kontext mit dem zu untersuchenden Sachverhalt.
b) Erstelle nach dem Beispiel auf den Screenshots eine Datei zum Ermitteln der kumulierten Summen der Würfelzahlen.
c) Ermittelt in Partnerarbeit, welche der Summen 101 bis 105 als erste nach dem Überschreiten der Zwischensumme 100 erreicht wird. Realisiert dazu mit ctrl R mindestens 30 Wiederholungen dieser Simulation, notiert die Ergebnisse und erstellt eine Häufigkeitsverteilung.

| Summe | 101 | 102 | 103 | 104 | 105 |
|---|---|---|---|---|---|
| Anzahl | | | | | |

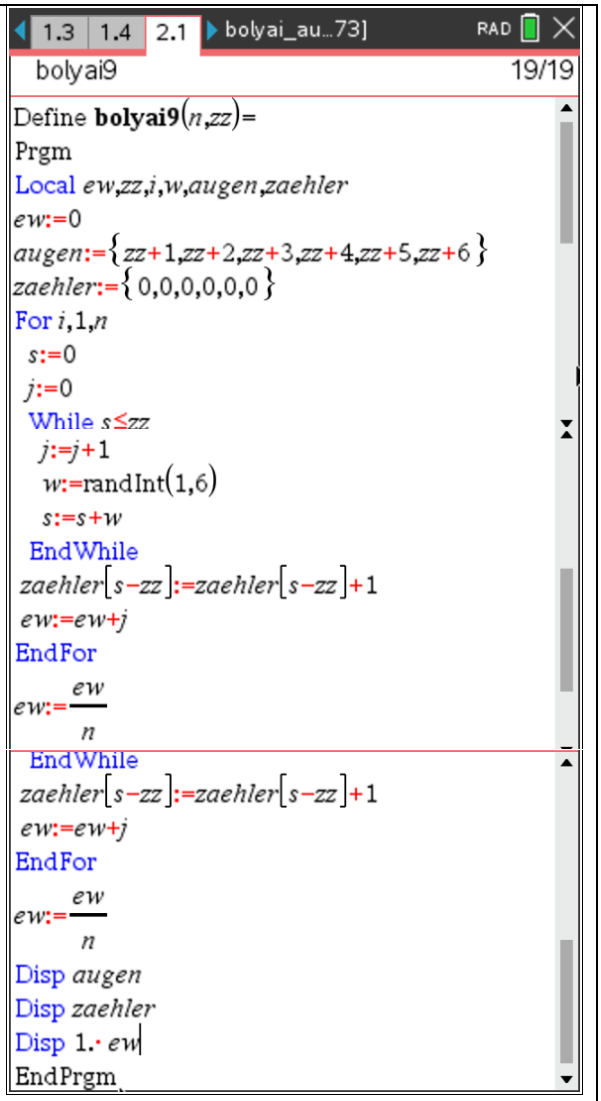
d) Fasst die Ergebnisse der Klasse zusammen.

Die Simulationsverfahren mit WTR oder die oben beschriebene Verfahrensweise mit der Tabellenkalkulation haben den Vorteil, dass der zu beobachtende Prozess „greifbarer“ wird. Allerdings sind einigermaßen große Versuchsumfänge nur näherungsweise zu erreichen, indem man die Ergebnisse von in Einzelversuchen bei individueller Schülerarbeit oder Partnerarbeit ermittelten Resultaten zusammenfasst.

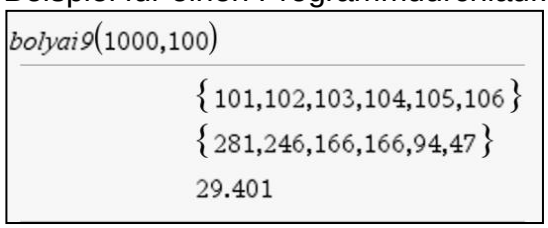
### 3. Programmierung mit TI-Basic (vgl. auch bolyai_aufgabe9.tns)

Der Parameter n legt die Anzahl der Simulationen fest.
Mit dem Parameter zz kann man eine beliebige Schranke festlegen, die überschritten werden muss.
(In der hier betrachteten Aufgabe wäre zz = 100).
Die Liste „augen“ enthält die Zielzahlen, für zz = 100 gilt:
augen = {101,102,103,104,105,106}
Vor jedem der n Durchläufe werden die Variablen s und j auf null gesetzt.
Die Variable s wird solange um eine Würfelzahl vermehrt, bis erstmals die beim Programmstart eingegebene Schranke (hier zz = 100) überschritten wird. Die while-Schleife bricht ab, wenn die Schranke überschritten wird. Der Zähler j zählt die Anzahl der Schritte, die beim einmaligen Durchlauf der while-Schleife benötigt werden, um zu einer der Zielzahlen zu gelangen.
In der Liste „zaehler“ wird gezählt, welche Zielzahl beim Durchlauf einer while-Schleife erreicht wurde.
Zaehler[s-zz] steuert das Listenelement mit der Nummer s-zz an.
Nach n while-Schleifen gibt die Liste „zaehler“ an, wie oft dabei jede Zielzahl erreicht wurde.
Mittels der beiden Listen „augen“ und „zaehler“ wird die Ausgabe ermöglicht.

Die Variable ew gibt die mittlere Wurfzahl aus.




Beispiel für einen Programmdurchlauf:



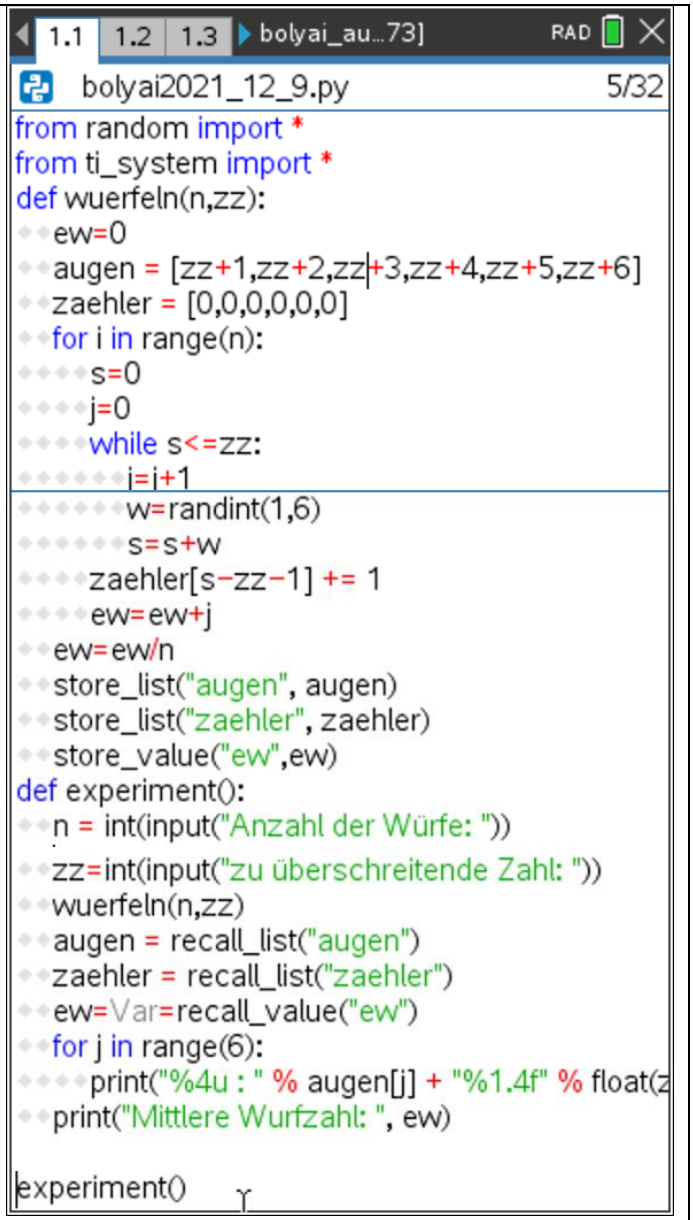

Das TI-Basic-Programm hat den Vorteil, dass sich viel schneller als mit den bisher beschriebenen Verfahren genügend viele Simulationen durchführen lassen, um zu besser vertretbaren Vermutungen zu gelangen.
Das TI-Basic-Programm hat aber den Nachteil, dass man die Versuchszahl nicht über n = 10 000 einstellen sollte.
Schon mit n = 10 000 benötigt das Programm am PC mehr als 17 Sekunden gegenüber weniger als einer Sekunde beim nachfolgend vorgestellten Python-Programm.

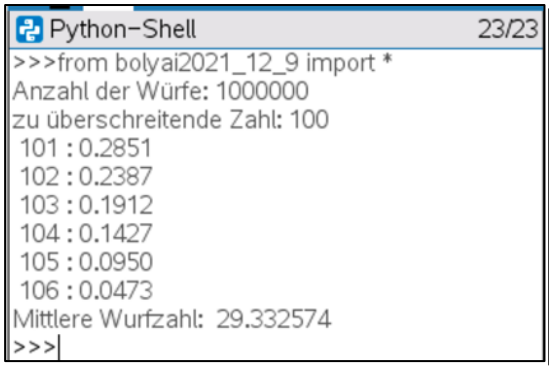
## 4. Programmierung mit Python (vgl. bolyai_aufgabe9.tns)

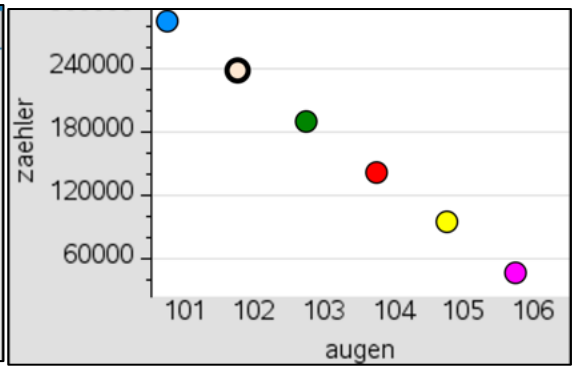
Der große Vorteil dieses kleinen Python-Programms gegenüber der Tabellenkalkulation und dem Basic-Programm ist, dass man problemlos auch 1 000 000 Versuche durchführen kann (Zeitdauer weniger als sechs Sekunden), ein Nachteil könnte natürlich eine Fehlerhaftigkeit des Algorithmus sein.

Der Parameter n in der Funktion wuerfeln() legt die Anzahl der Simulationen fest. Mit dem Parameter zz kann man beliebige Schranken festlegen, die überschritten werden müssen.
(In der eigentlichen Aufgabe wäre zz = 100)
Mittels der beiden Listen „augen“ und „zaehler“ wird die Ausgabe ermöglicht.
Die while-Schleife bricht ab, wenn die Schranke überschritten wird.
Die Variable ew gibt die mittlere Wurfzahl aus.

Das eigentliche Experiment wird dann über die Funktion „experiment() durchgeführt, in der die oben definierte Funktion wuerfeln() aufgerufen wird.
Die Ausgabe wird über die beiden Print-Befehle in der letzten For-Schleife realisiert.






Mit 1 Million Versuchen (Laufzeit des Programms am PC ca. 6 Sekunden) sind die Ergebnisse so stabil, wie oben angegeben, sprich in ca. 28,5% der Versuche ist 101 die „Endzahl“.

Wie ließe sich dies nun begründen?

Betrachtet man die Wahrscheinlichkeiten zu diesen Zahlen zu kommen, wenn man vorher auf einer der Zahlen 95, 96, ...,99, 100 gestanden hat, dann ist es relativ klar:

| | **101** | **102** | **103** | **104** | **105** | **106** |
|---|---|---|---|---|---|---|
| 95 | $\frac{1}{6}$ | | | | | |
| 96 | $\frac{1}{6}$ | $\frac{1}{6}$ | | | | |
| 97 | $\frac{1}{6}$ | $\frac{1}{6}$ | $\frac{1}{6}$ | | | |
| 98 | $\frac{1}{6}$ | $\frac{1}{6}$ | $\frac{1}{6}$ | $\frac{1}{6}$ | | |
| 99 | $\frac{1}{6}$ | $\frac{1}{6}$ | $\frac{1}{6}$ | $\frac{1}{6}$ | | |
| 100 | $\frac{1}{6}$ | $\frac{1}{6}$ | $\frac{1}{6}$ | $\frac{1}{6}$ | $\frac{1}{6}$ | $\frac{1}{6}$ |

In sechs der 21 Fälle (ca. 28,5%) gelangt man zu 101:

Von der 95 mit einer „6“, von der 96 mit einer „5“, von der 97 mit einer „4“, von der 98 mit einer „3“, von der 99 mit einer „2“ und von der 100 mit einer „1“.

Analog lassen sich die Fälle für das Erreichen der Zielzahlen 102 bis 106 zählen:
In 5 der 21 Fälle (ca. 23,8%) gelangt man zu 102 usw.

Damit scheint die Bolyai-Aufgabe gelöst.

Ein wesentlicher Einwand ist aber: Wenn wir es so machen, dann gehen wir ja von der „Gleichwahrscheinlichkeit“ aus, die Zahlen 95, 96, ..., 99, 100 zu erreichen.

Ist dies gerechtfertigt?

Eigentlich müsste man die Wahrscheinlichkeiten doch folgendermaßen berechnen, wie es hier am Beispiel für das Erreichen der Zielzahl 101 erläutert wird:

Zur Zwischensumme 95 gelangt man mit einer Wahrscheinlichkeit $p_{95}$ und von da aus mit $p = \frac{1}{6}$ zur Zielzahl 101. Insgesamt ergibt sich für die etwas weiter oben genannten sechs Fälle folgende Gesamtwahrscheinlichkeit:

$$p_{95} \cdot \frac{1}{6} + p_{96} \cdot \frac{1}{6} + p_{97} \cdot \frac{1}{6} + p_{98} \cdot \frac{1}{6} + p_{99} \cdot \frac{1}{6} + p_{100} \cdot \frac{1}{6}$$

Problem: Wie groß sind die Wahrscheinlichkeiten $p_{95}$ bis $p_{100}$?

Andreas Eichler und Frank Förster geben hierzu in dem sehr schönen Artikel „Ein Märchenspiel – Stochastische Modellbildung bei einem merkwürdigen Brettspiel“ (vgl. Istronheft Bd. 12, S. 107 ff.[3]) Erklärungen, die sie dort sowohl durch Simulation und mathematische Betrachtungen begründen. Sie zeigen, warum der „alternierende hypothetische“ Grenzwert für die Trefferwahrscheinlichkeit für eine beliebige Zahl der Kehrwert des Erwartungswerts des einfachen Würfelwurfs, nämlich 2/7 (ca. 28,57%), ist. Hier die durch rekursive Berechnung gefundenen Werte für die Zahlen von 1 bis 100:

| Feld | P(Z = m) | Feld | P(Z = m) | Feld | P(Z = m) | Feld | P(Z = m) | Feld | P(Z = m) |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 0,166667 | 21 | 0,285968 | 41 | 0,285714 | 61 | 0,285714 | 81 | 0,285714 |
| 2 | 0,194444 | 22 | 0,285944 | 42 | 0,285714 | 62 | 0,285714 | 82 | 0,285714 |
| 3 | 0,226852 | 23 | 0,285756 | 43 | 0,285715 | 63 | 0,285714 | 83 | 0,285714 |
| 4 | 0,264660 | 24 | 0,285598 | 44 | 0,285715 | 64 | 0,285714 | 84 | 0,285714 |
| 5 | 0,308771 | 25 | 0,285600 | 45 | 0,285714 | 65 | 0,285714 | 85 | 0,285714 |
| 6 | 0,360232 | 26 | 0,285748 | 46 | 0,285714 | 66 | 0,285714 | 86 | 0,285714 |
| 7 | 0,253604 | 27 | 0,285769 | 47 | 0,285714 | 67 | 0,285714 | 87 | 0,285714 |
| 8 | 0,268094 | 28 | 0,285736 | 48 | 0,285714 | 68 | 0,285714 | 88 | 0,285714 |
| 9 | 0,280369 | 29 | 0,285701 | 49 | 0,285714 | 69 | 0,285714 | 89 | 0,285714 |
| 10 | 0,289288 | 30 | 0,285692 | 50 | 0,285714 | 70 | 0,285714 | 90 | 0,285714 |
| 11 | 0,293393 | 31 | 0,285707 | 51 | 0,285714 | 71 | 0,285714 | 91 | 0,285714 |
| 12 | 0,290830 | 32 | 0,285725 | 52 | 0,285714 | 72 | 0,285714 | 92 | 0,285714 |
| 13 | 0,279263 | 33 | 0,285722 | 53 | 0,285714 | 73 | 0,285714 | 93 | 0,285714 |
| 14 | 0,283540 | 34 | 0,285714 | 54 | 0,285714 | 74 | 0,285714 | 94 | 0,285714 |
| 15 | 0,286114 | 35 | 0,285710 | 55 | 0,285714 | 75 | 0,285714 | 95 | 0,285714 |
| 16 | 0,287071 | 36 | 0,285712 | 56 | 0,285714 | 76 | 0,285714 | 96 | 0,285714 |
| 17 | 0,286702 | 37 | 0,285715 | 57 | 0,285714 | 77 | 0,285714 | 97 | 0,285714 |
| 18 | 0,285587 | 38 | 0,285716 | 58 | 0,285714 | 78 | 0,285714 | 98 | 0,285714 |
| 19 | 0,284713 | 39 | 0,285715 | 59 | 0,285714 | 79 | 0,285714 | 99 | 0,285714 |
| 20 | 0,285621 | 40 | 0,285714 | 60 | 0,285714 | 80 | 0,285714 | 100 | 0,285714 |

Tabelle 3: Exakte Werte der rekursiven Berechnung für das leere Spielfeld

112 Andreas Eichler & Frank Förster

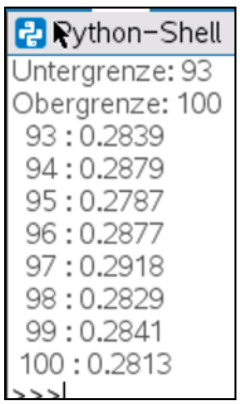
[3] https://www.istron.mathematik.uni-wuerzburg.de/istron/category/ausgaben/band-12/index.html

Erst hiermit rechtfertigt sich der Ansatz, von einer „Gleichwahrscheinlichkeit“ des Erreichens der Felder 95 bis 100 auszugehen.

Damit wird aus $p_{95} \cdot \frac{1}{6} + p_{96} \cdot \frac{1}{6} + p_{97} \cdot \frac{1}{6} + p_{98} \cdot \frac{1}{6} + p_{99} \cdot \frac{1}{6} + p_{100} \cdot \frac{1}{6}$ der Term $6 \cdot p \cdot \frac{1}{6} = p$, der den größten Wert der Terme $k \cdot p \cdot \frac{1}{6}$ mit $k \in \{6,5,4,3,2,1,\}$ hat.

Hinweis: Man kann mit dem beigefügten Python-Programm diese Werte $p_i$ für beliebige $i \in \mathbb{N}$ überprüfen – hier z. B. für die Zahl 7.


Möglich ist auch eine Ausgabe der gesamten Tabelle bzw. von Teilen davon.




**Beigefügte Programme**

zahlenfolge1.tns

zahlenfolge2.tns

Bolyai_aufgabe9.tns

Bolyai_aufgabe9_GES.tns

**Lösung des Wettbewerbsausrichters[4]:**

**Lösung:** Wir bezeichnen die Summe der gewürfelten Zahlen vor dem letzten Wurf mit $S_1$. Da $S_1 \leq 100$ und $S > 100$, muss $S_1 \geq 95$ gelten. Also kann $S_1$ sechs verschiedene Werte annehmen.

1. $S_1 = 100$: Mit der gleichen Wahrscheinlichkeit kann $S$ 101, 102, ... , 106 sein, je nach dem, ob der letzte Wurf 1, 2, 3, 4, 5, oder 6 war.
2. $S_1 = 99$: Der letzte Wurf konnte 2, 3, 4, 5 oder 6 sein, somit sind für $S$ die Werte 101, 102, 103, 104, 105 gleichwahrscheinlich.
3. $S_1 = 98$: Der letzte Wurf kann 3, 4, 5 oder 6 sein, somit gilt mit der gleichen Wahrscheinlichkeit für $S$ 101, 102, 103 oder 104.

Wir gehen ähnlich in den weiteren Fällen vor. Für $S_1 = 95$ kann der letzte Wurf nur 6 sein und der Wert von $S$ nur 101.

Wir können leicht sehen, dass für $S$ der Wert 101 am wahrscheinlichsten ist. Genauer gesagt: die Warscheinlichkeit für 101 ist eben mit der Warscheinlichkeit für $S_1 = 95$ mehr als die Wahrscheinlichkeit für 102 usw. Die Wahrscheinlichkeiten nehmen in der Reihenfolge 101, 102, 103, 104, 105, 106 ab.

**Richtige Antwort(en): A**

**Autoren:**
*Hubert Langlotz*
*Wilfried Zappe*

[4] https://www.bolyaiteam.de/, Stand 03.03.2021, 16:15 Uhr