Angewandte Mathematik

Abzählbarkeit

Andre Wolfram, Christoph Miesel, Robert Seilbeck

Gliederung

- Einleitung
- Cantors
  - 264 Count on Cantors
  - 880 Cantor Fractions
- Spirals
  - 493 Rational Spirals
  - 10920 Spiral Tap
  - 903 Spirals of Numbers

Gelöste Probleme

- Mengen:
  - 264, 484, 880, 11172
- Teilbarkeit:
  - 113, 160, 294, 382, 406, 543, 686, 10139, 10104, 10235
- Zahlen:
  - 493, 903, 10622, 10920, 11461
- Zeichenketten:
  - 401, 10062, 10789
- Formeln:
  - 10070, 11479

Schönstes Problem

1) Ecological Bin Packing
   - Lebensnahe Problemstellung
2) Easter Eggs
   - Nette Beschreibung
3) Rational Spiral
   - Lerneffekt

Schönste Lösung

- Word Scramble 483 – Mohr, Schirm, Mathauser
  - Übersichtlich
- Smith Numbers 10042 – Sachse, Seidl
  - Gut in Methoden zerlegt
- Password Search 902 - Wilfert,Studt
  - In Java schwierig

Georg Cantor (1845 - 1918)

- Deutscher Mathematiker
- Begründer der Mengenlehre
- Bewies das rationale Zahlen abzählbar sind

Einleitung

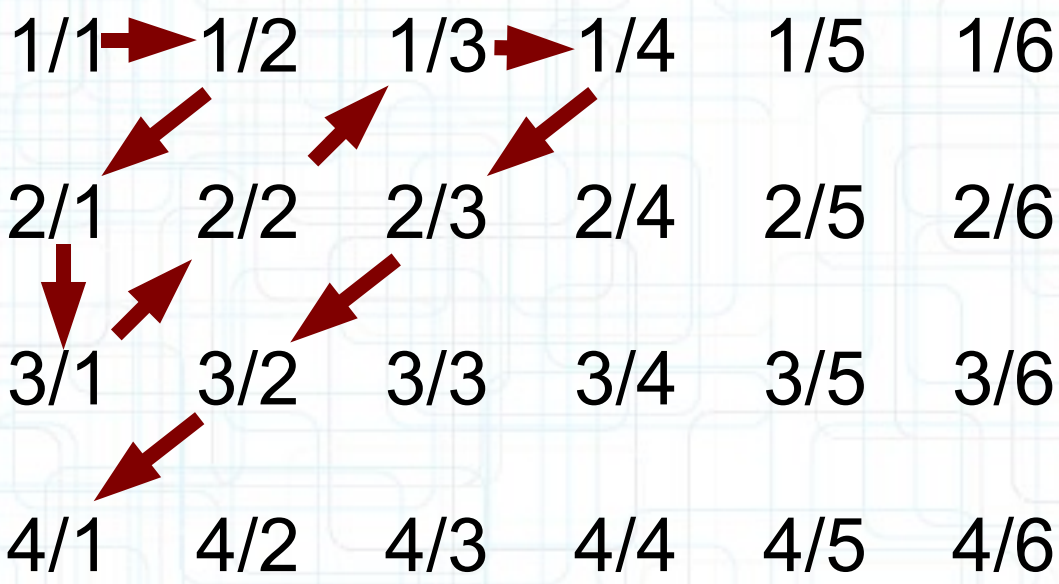
- Allgemeine Thematik: Abzählbarkeit
- Abzählbarkeit der rationalen Zahlen
  - Count on Cantor
  - Cantor Fractions
  - Rational Spirals
- Erweiterte Problemstellungen
  - Spiral Tap
  - Spiral of Numbers

264 Count on Cantors

- Aufgabenstellung
- Erkenntnisse
- Algorithmus
- Implementierung

Aufgabenstellung

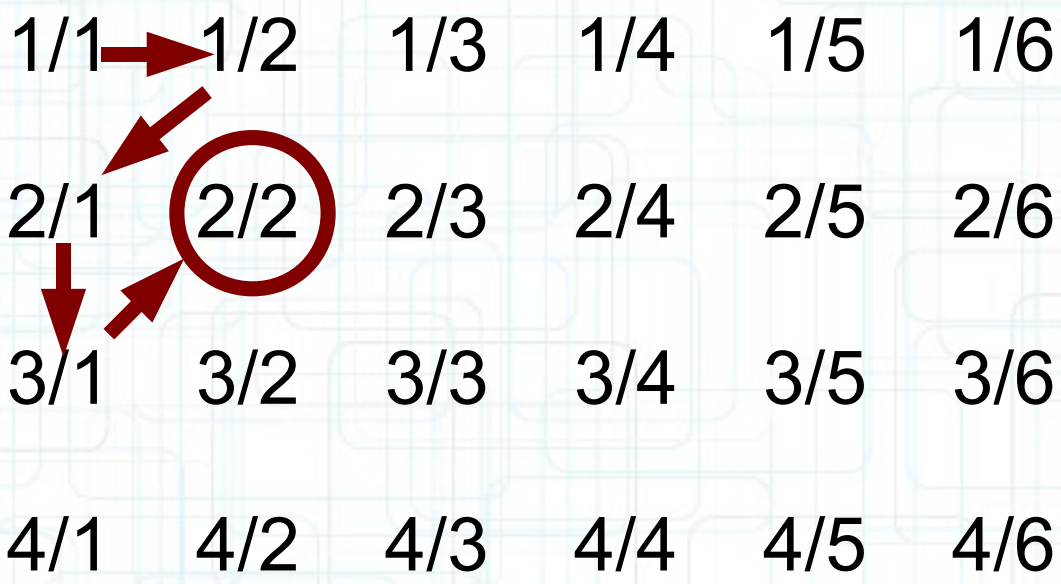
- Abbildung der Rationalen Zahlen



- Eingabe besteht aus Indizes

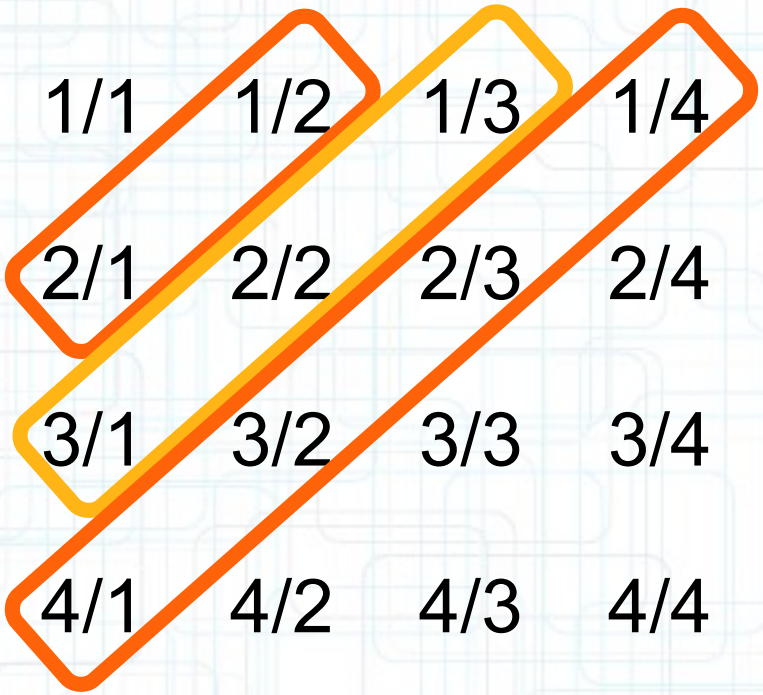
Aufgabenstellung

- Beispiel Index 5



- Größter Index ist: 10^7

Erkenntnisse

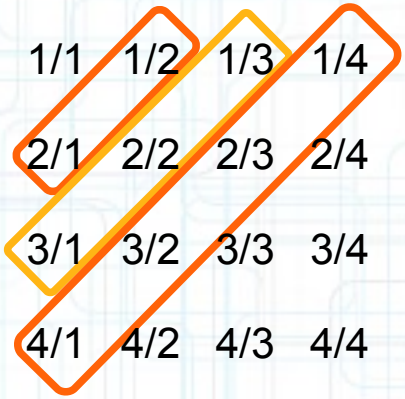


	1/1	1/2	1/3	1/4	1/5	1/6
2	2/1	2/2	2/3	2/4	2/5	2/6
3	3/1	3/2	3/3	3/4	3/5	3/6
4	4/1	4/2	4/3	4/4	4/5	4/6

- Gerade Anzahl in Linie:
  - Zähler aufsteigend
  - Nenner absteigend
- Ungerade Anzahl in Linie:
  - Zähler absteigend
  - Nenner aufsteigend

Algorithmus

Zeile des Elements bestimmen



- Summe der Elemente:
  - 1+2+3+4+...+n
- Addieren bis Summe größer/gleich als Index
  - Anzahl Additionen = Linie

Algorithmus

Element aus Linie bestimmen

- Elementnr.:
  - Index – Summe der Elemente aus vorangegangenen Linien
- Nochmaliges Abzählen zu langsam!

Beispiel:

Index: 9

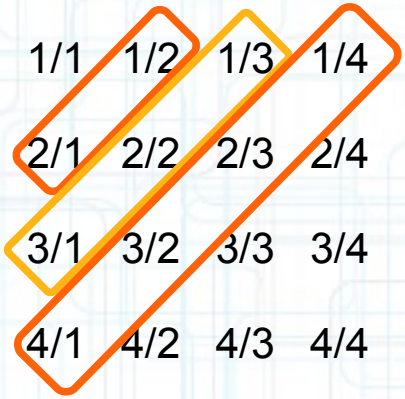
Linie: 4

Elemente in vorange. Linien: 6

Element in Linie: 9 -6 =3

Algorithmus

Lösung: Geschlossene Form



Statt 1+2+3+..+n

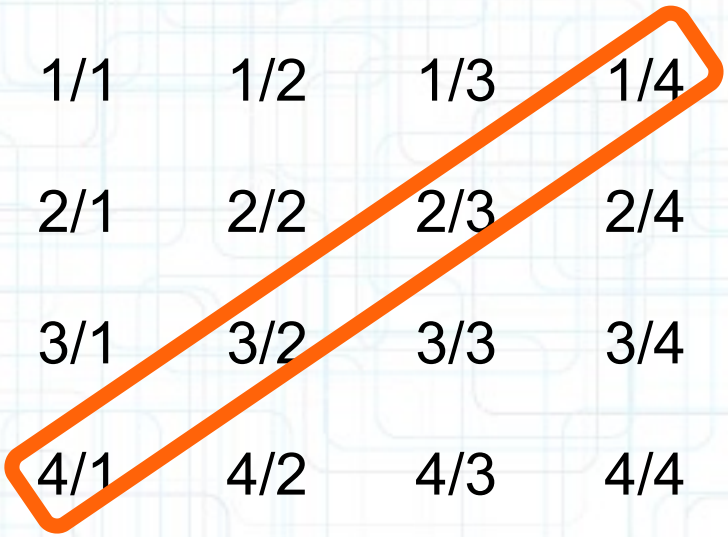
der kleine Gauß

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

Algorithmus

Index: 9

- Anzahl Additionen: 4



1/1	1/2	1/3	1/4	1/5	1/6
2/1	2/2	2/3	2/4	2/5	2/6
3/1	3/2	3/3	3/4	3/5	3/6
4/1	4/2	4/3	4/4	4/5	4/6

Element aus Linie: Index – kleiner Gauß von 3

9 – 6 = 3

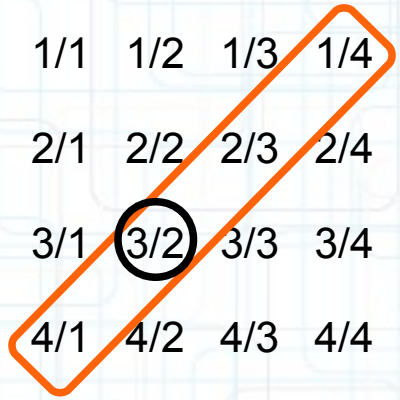
Algorithmus

Linie 4 (Gerade Anzahl)

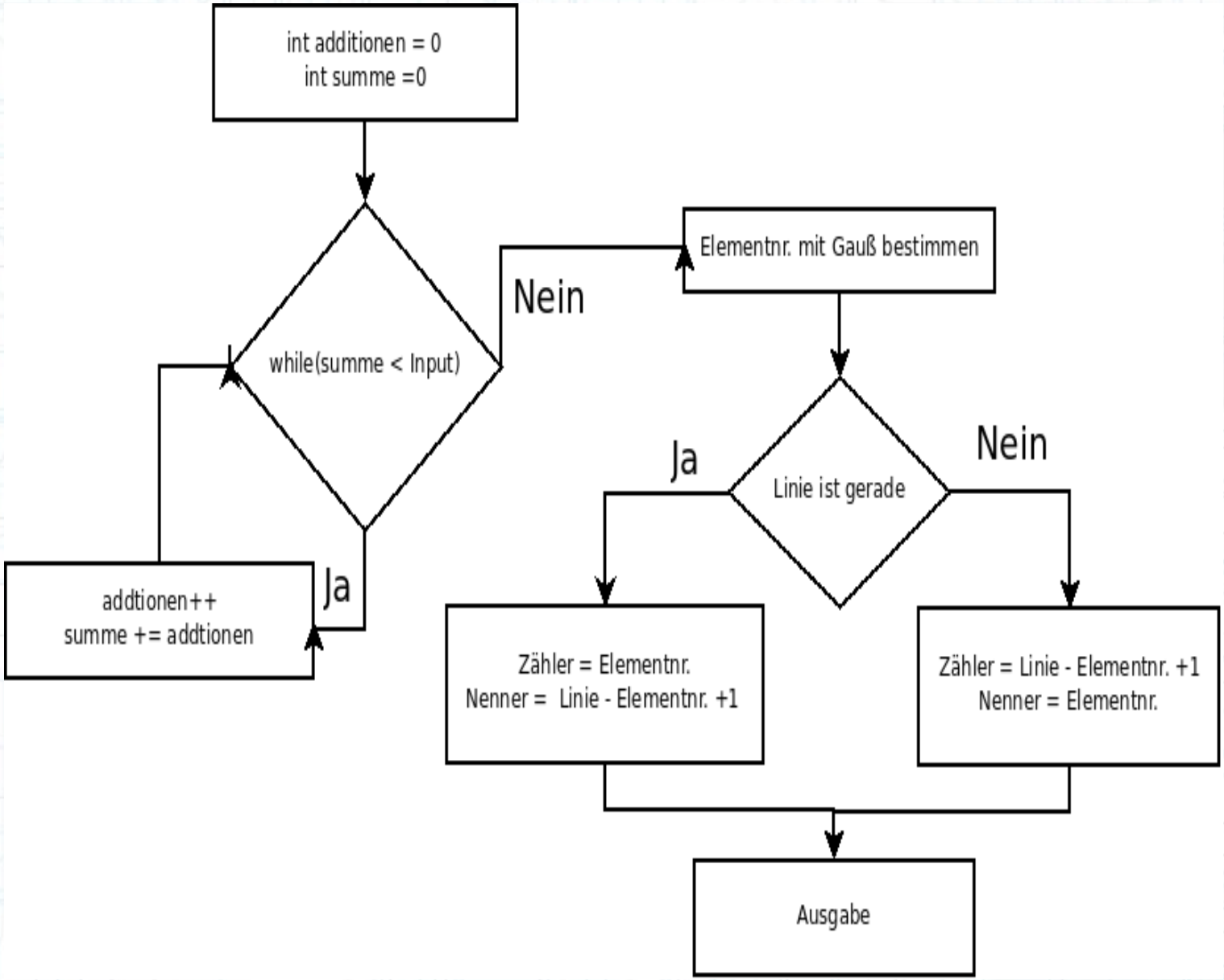
- Zähler aufsteigend
- Nenner absteigend

- Element in Linie 3

Element	Zähler	Nenner
1	1	4
2	2	3
3	3	2
4	4	1



Implementierung

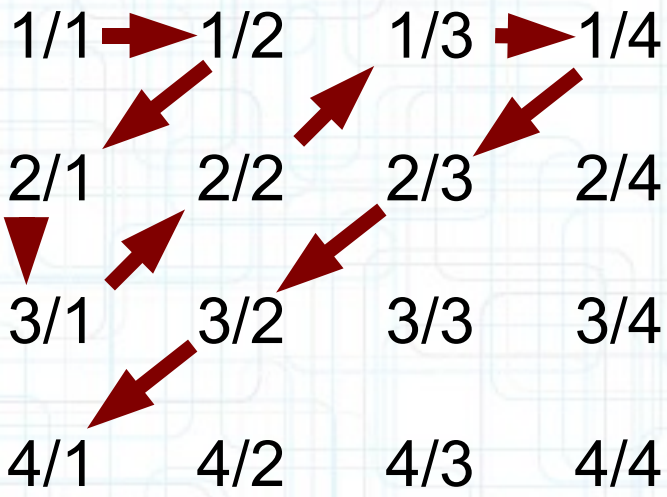


880 Cantor Fractions

- Aufgabenstellung:
  - Wie in vorherigen Problem
  - Andere Zählweise
- Eingabe:
  - Wieder Indizes
  - Aber deutlich mehr und höhere

Vergleich zu Count on Cantor

- Count on Cantor



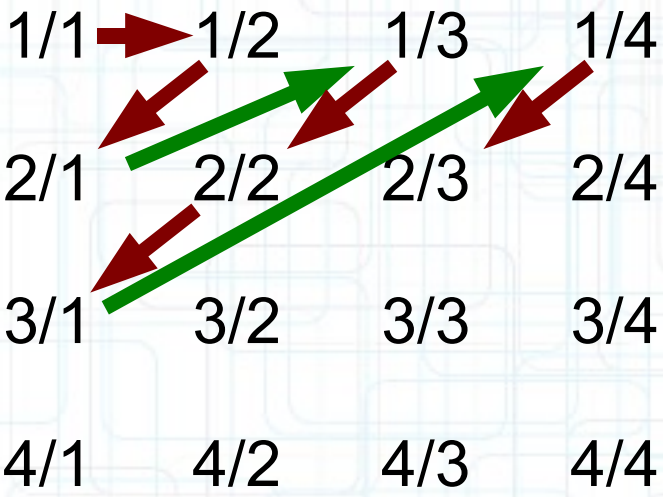
- Zähler

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

- Nenner

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

- Cantor Fractions



- Zähler

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

- Nenner

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

Algorithmus

Summe der Elemente bis Zeile n

1+2+3+4+...+n

Der kleine Gauß

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

Gesuchte Linie kann direkt ermittelt werden

$$k = \frac{-1+\sqrt{1+4*2*l}}{2}$$

Algorithmus

Index: 9

- Ergebnis kleiner Gauß: 3,772001873

Aufgerundet = 4

1/1	1/2	1/3	1/4	1/5	1/6
2/1	2/2	2/3	2/4	2/5	2/6
3/1	3/2	3/3	3/4	3/5	3/6
4/1	4/2	4/3	4/4	4/5	4/6

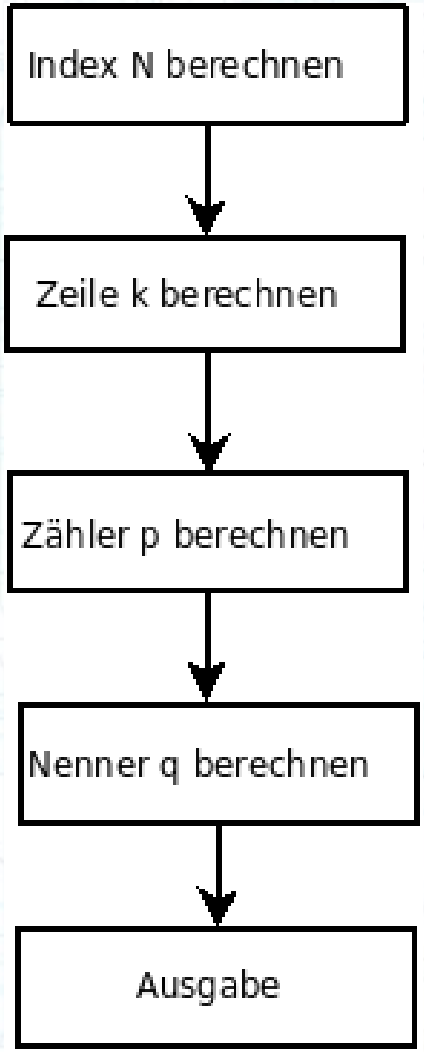
Element aus Linie: Index – kleiner Gauß von 3

9 – 6 = 3

Zusammenfassung

- Berechnung aktuelle Zeile k: $k = \frac{\sqrt{8l+1}-1}{2}$
- Berechnung Zähler IMMER!: $p = l - \frac{(k-1)k}{2}$
- Berechnung Nenner IMMER!: $q = k + 1 - p$

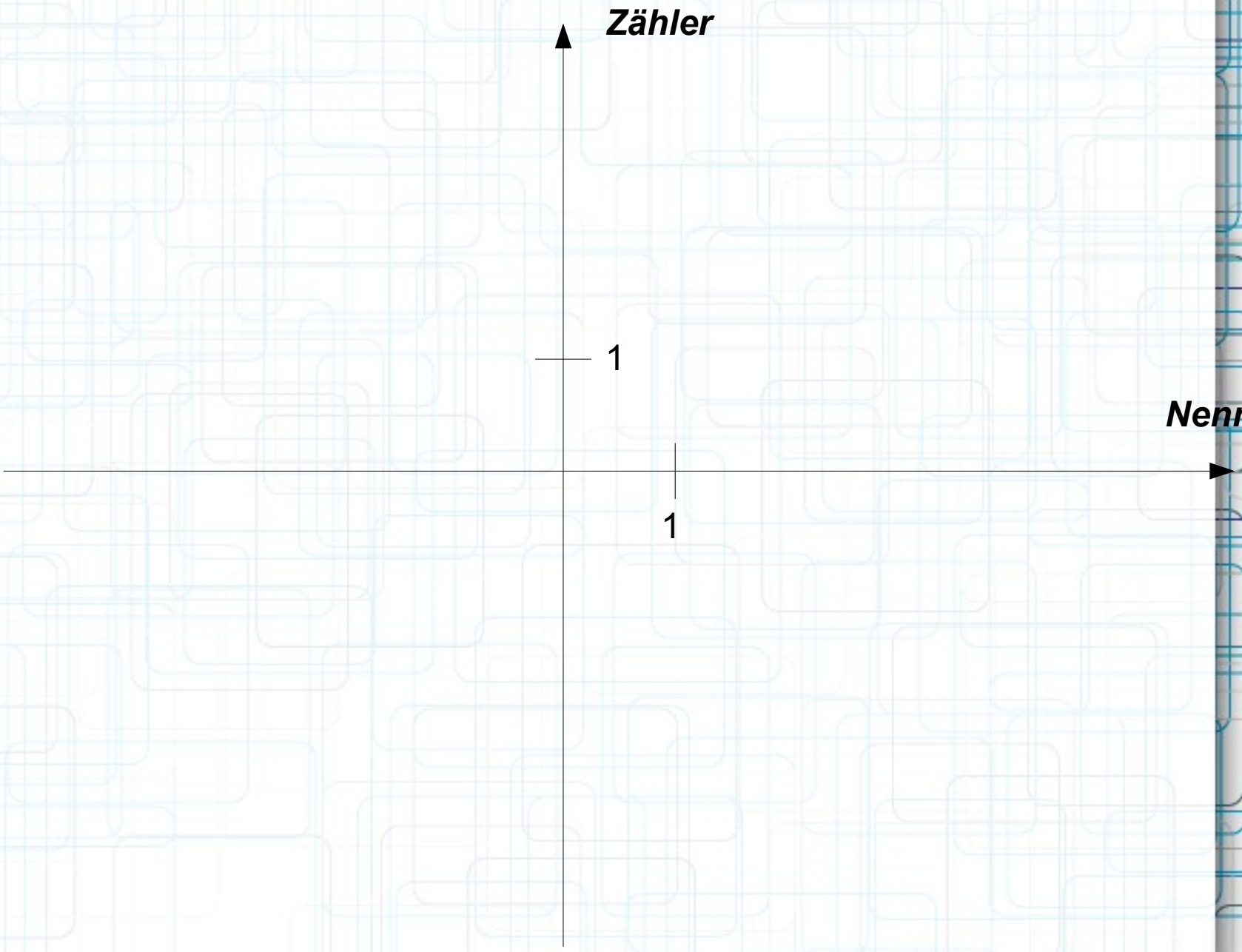
Implementierung



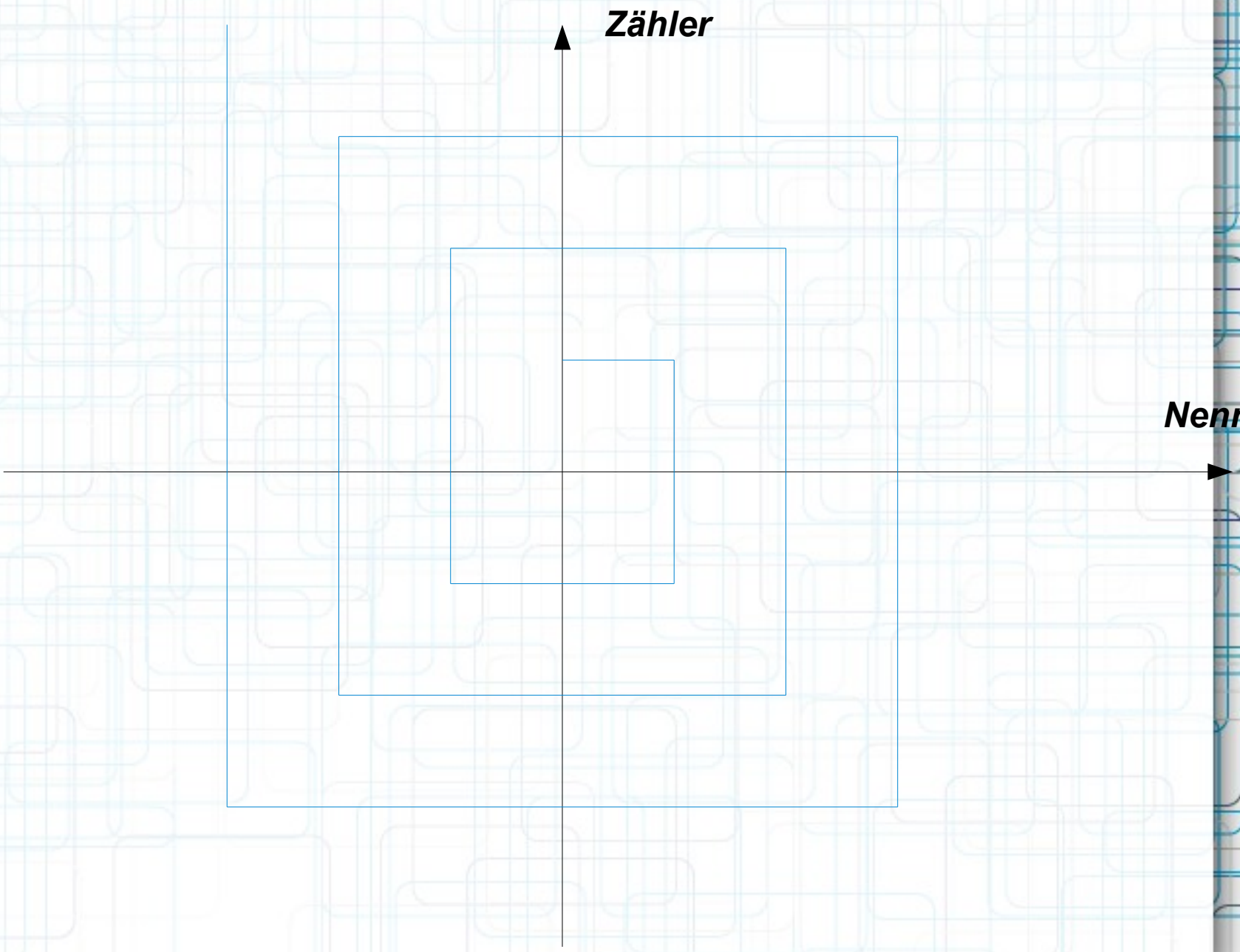
493 Rational Spiral

- Problemstellung
- Beispiel
- Grundlagen für die Lösung
- Algorithmen
- Implementierung

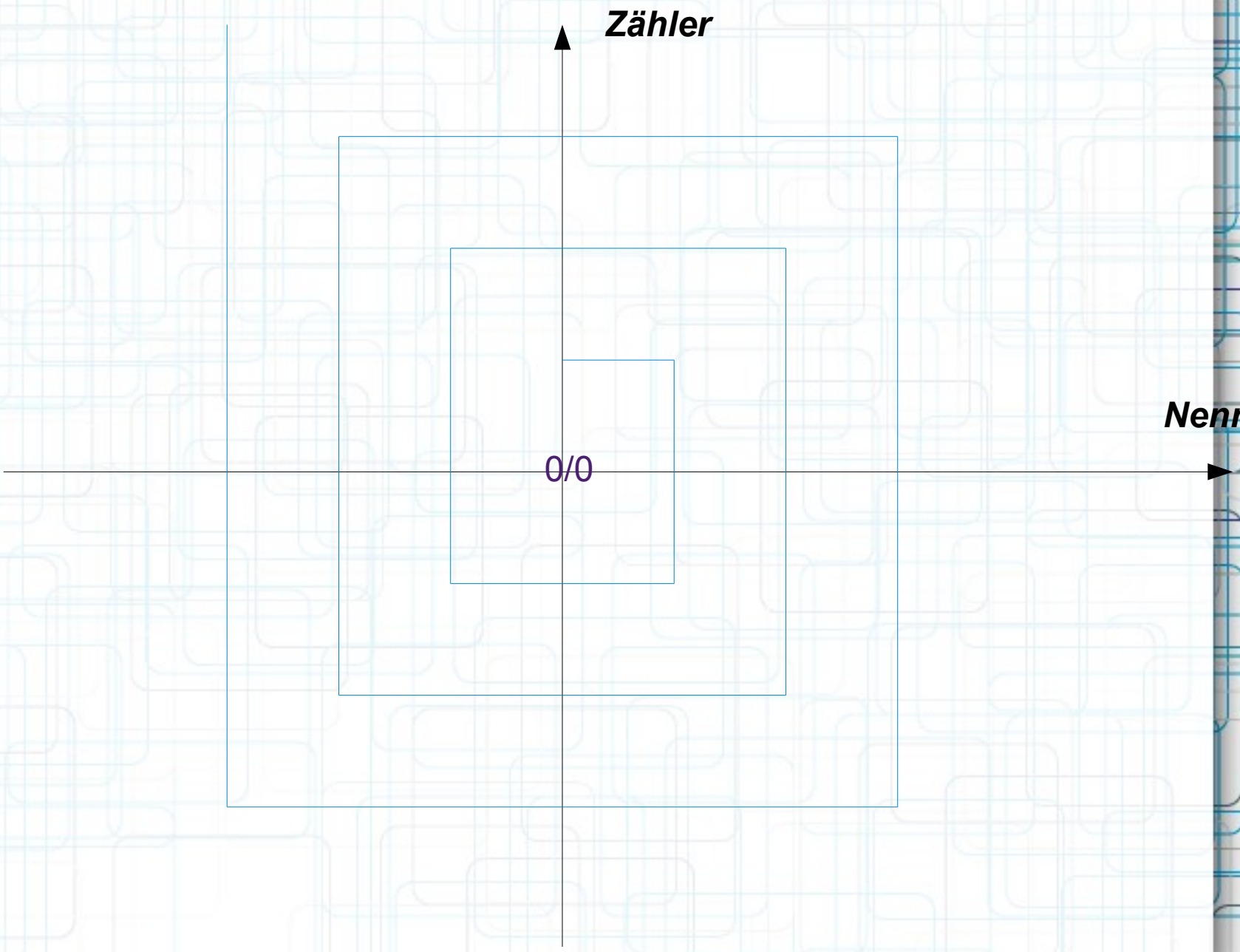
Problemstellung



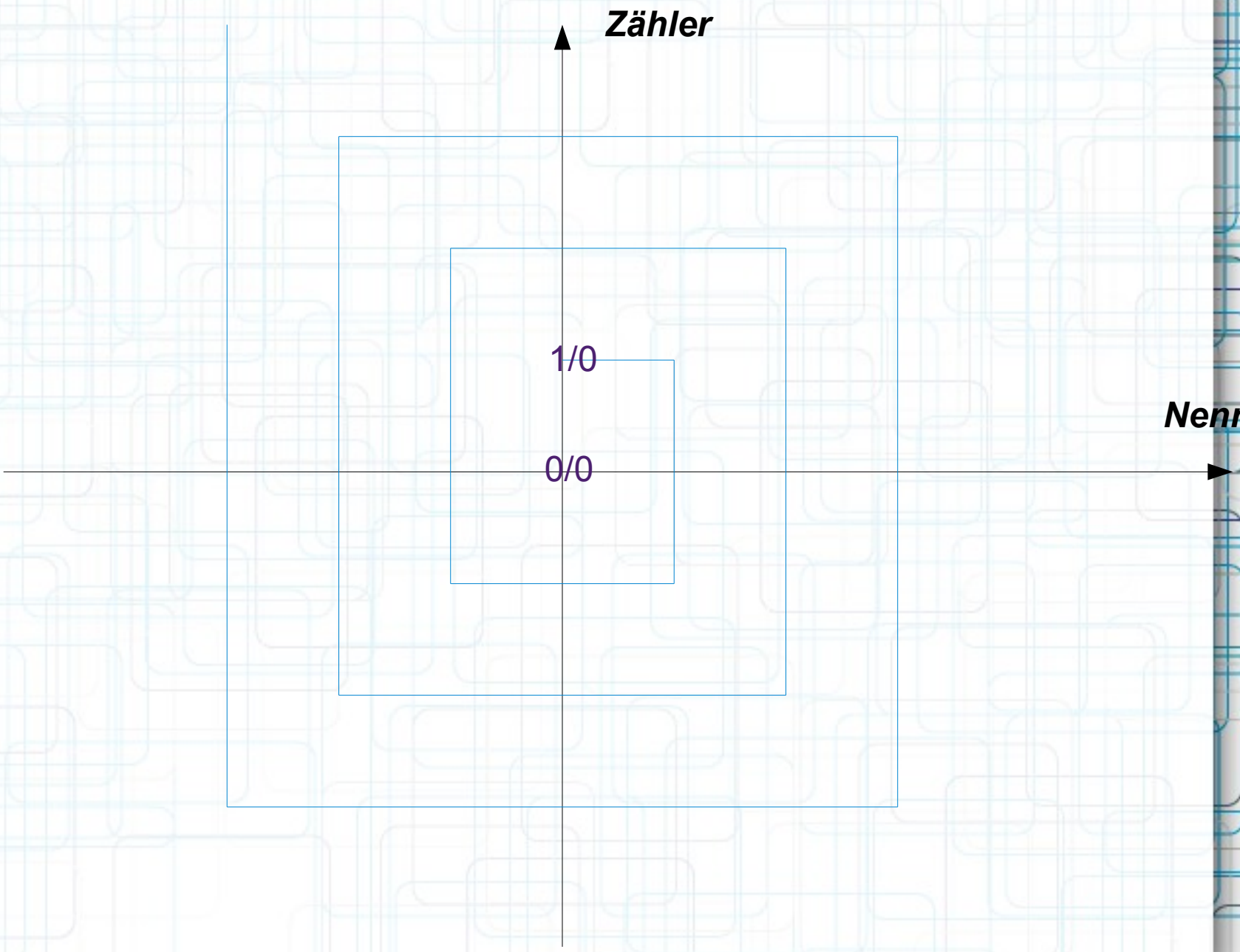
Problemstellung



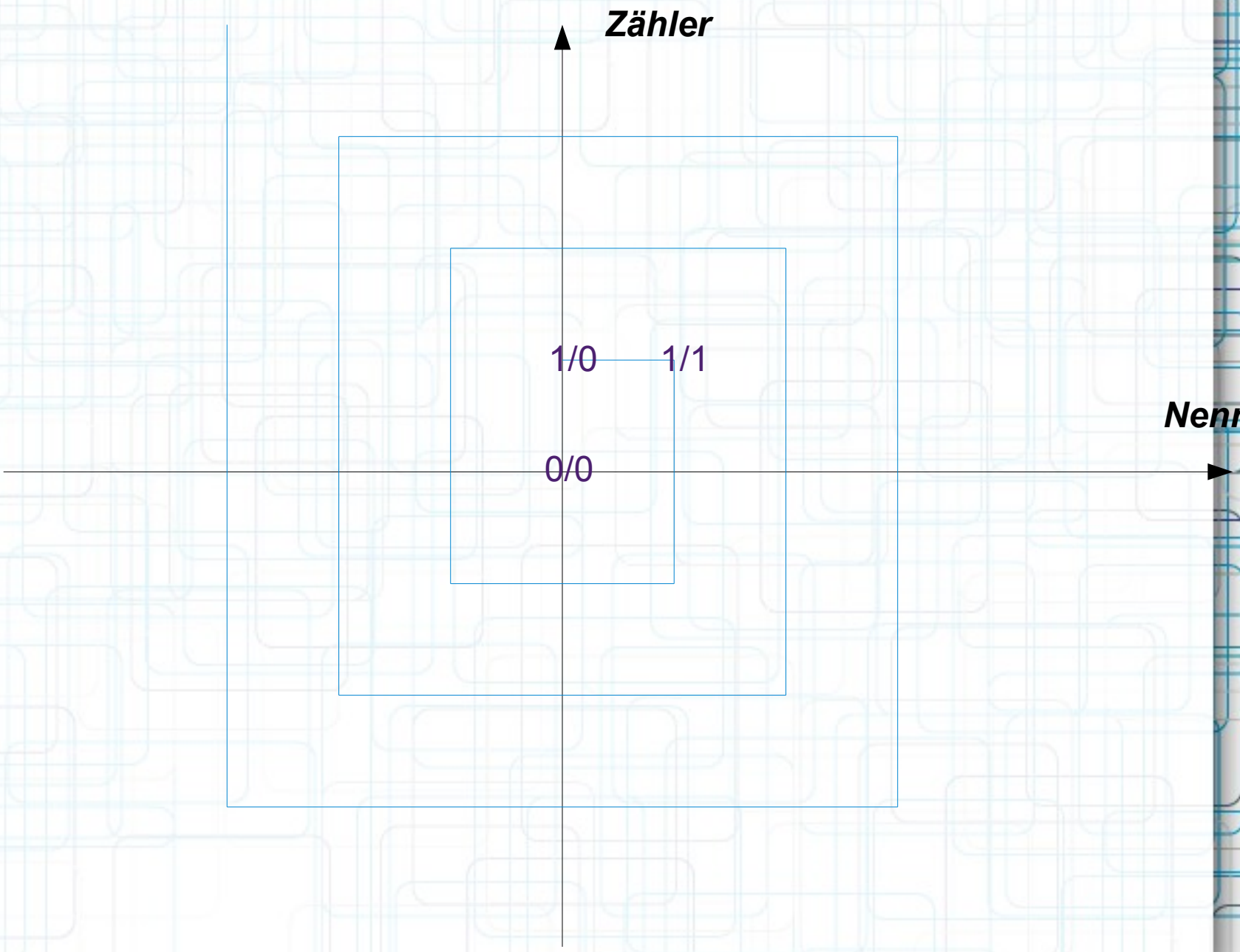
Problemstellung



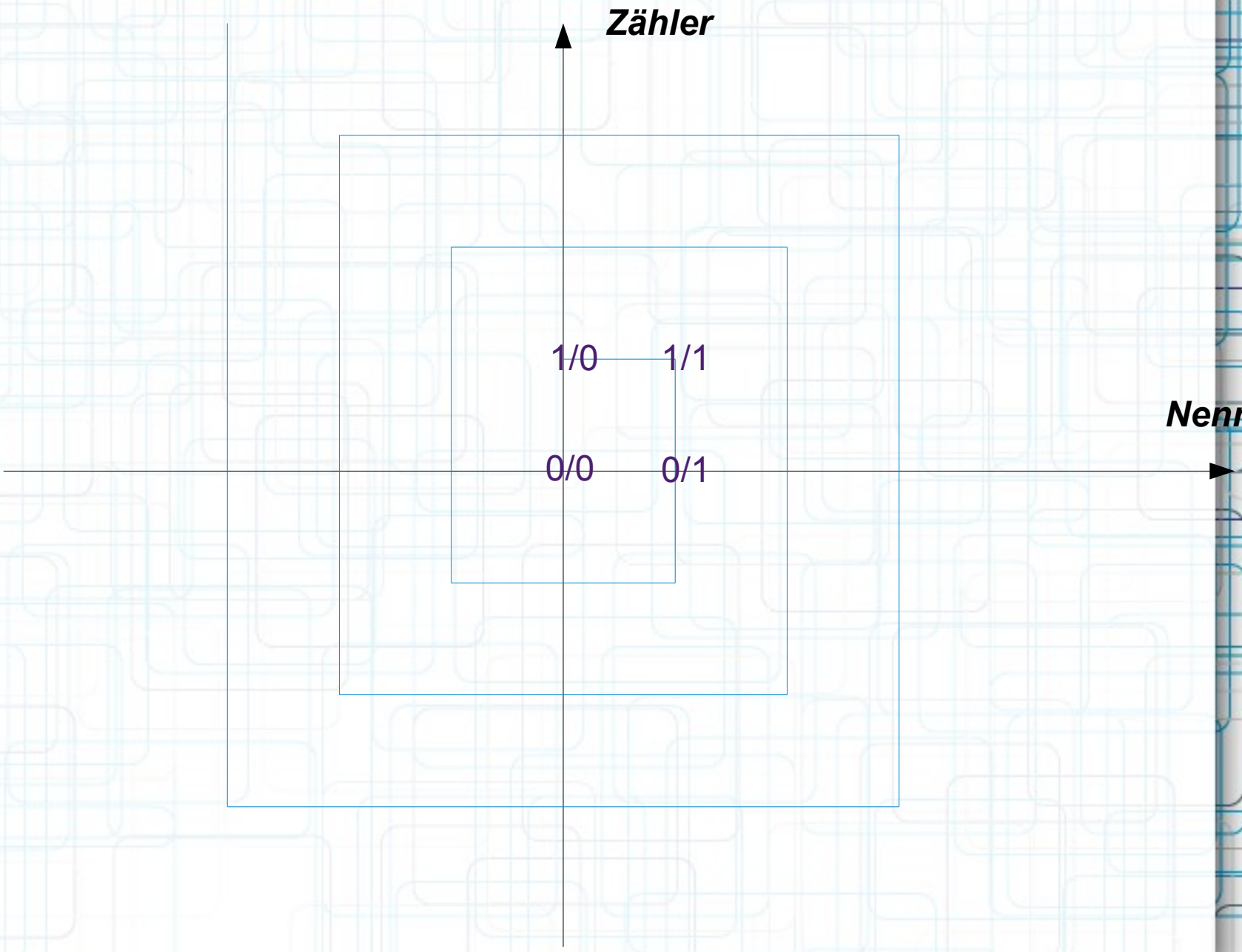
Problemstellung



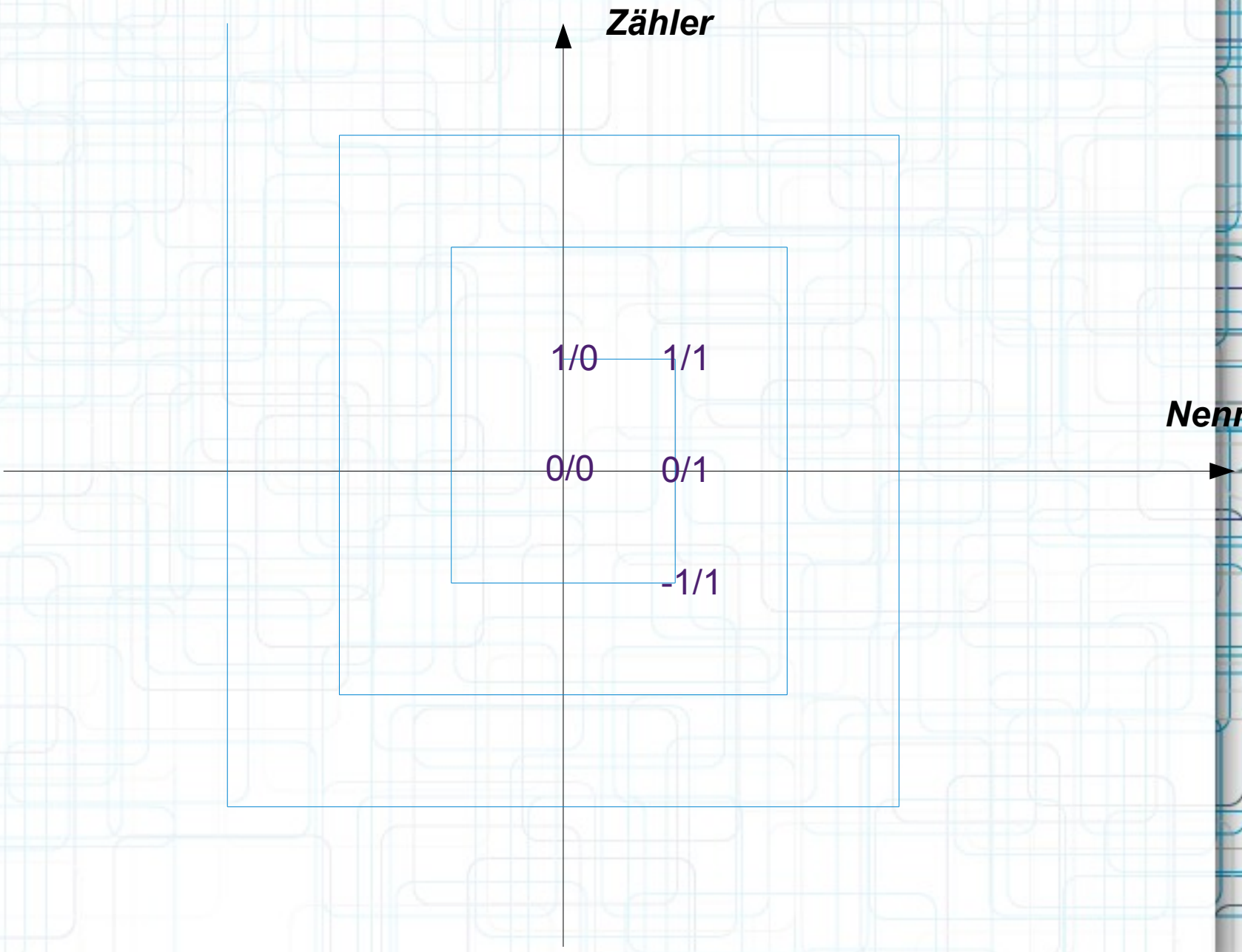
Problemstellung



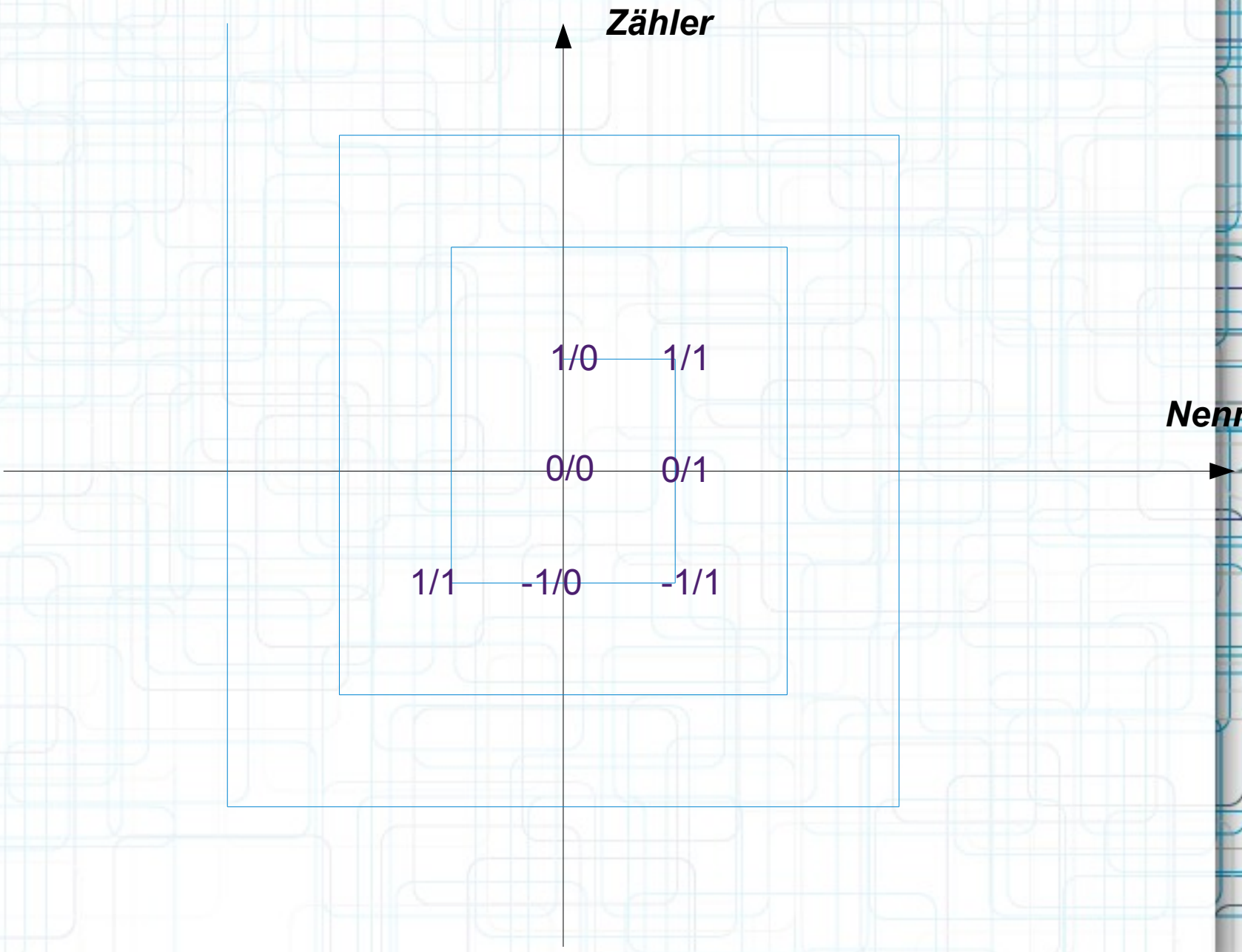
Problemstellung



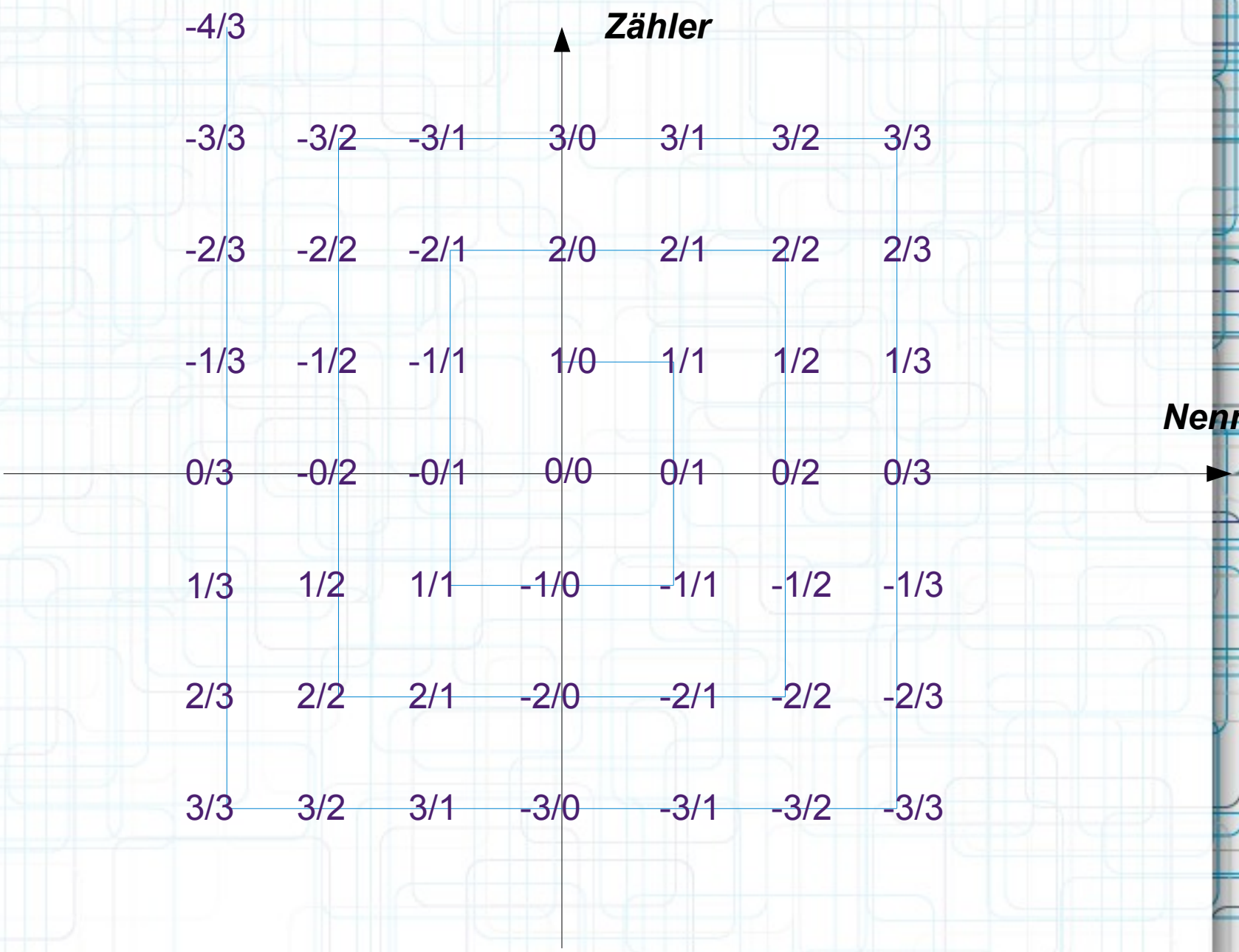
Problemstellung



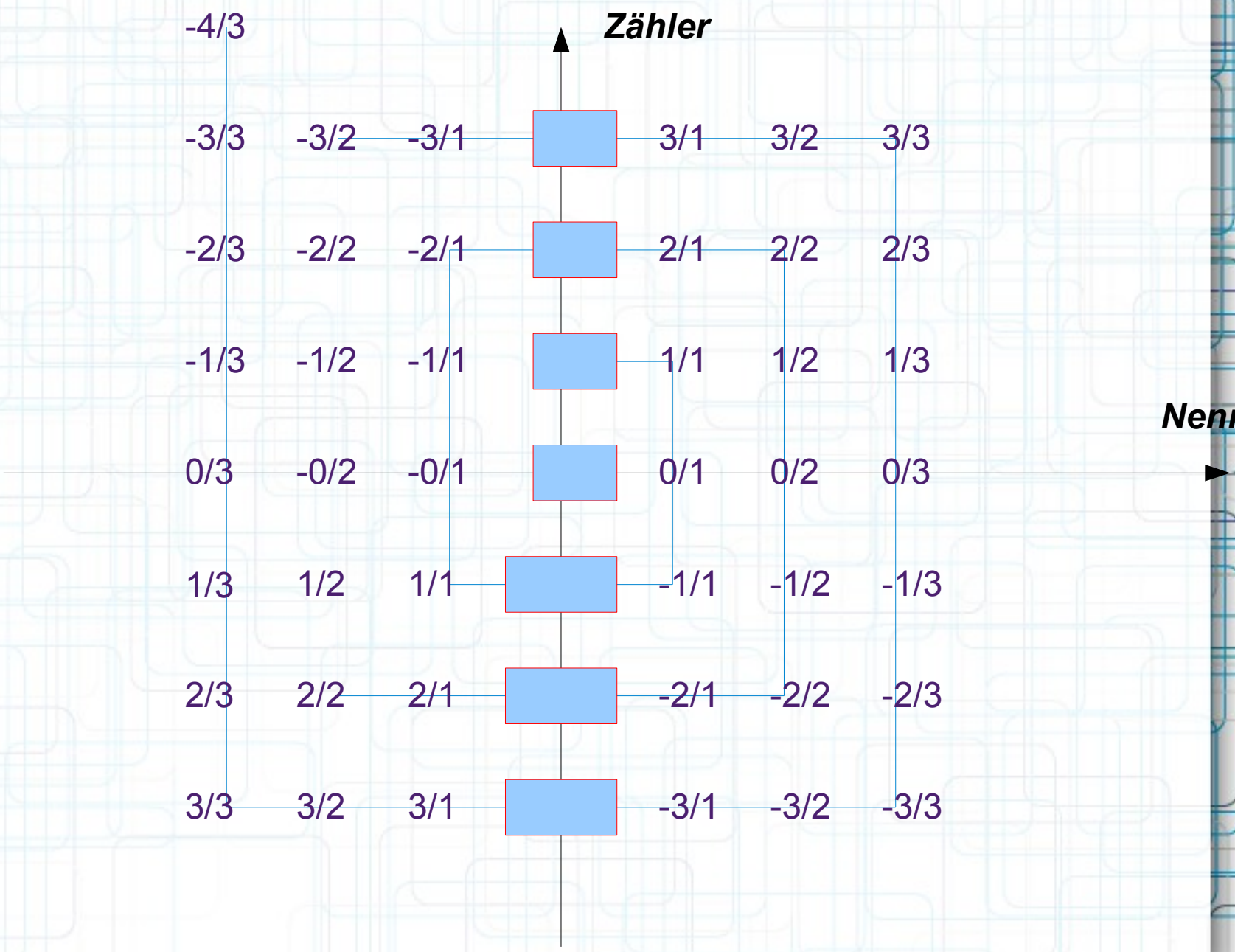
Problemstellung



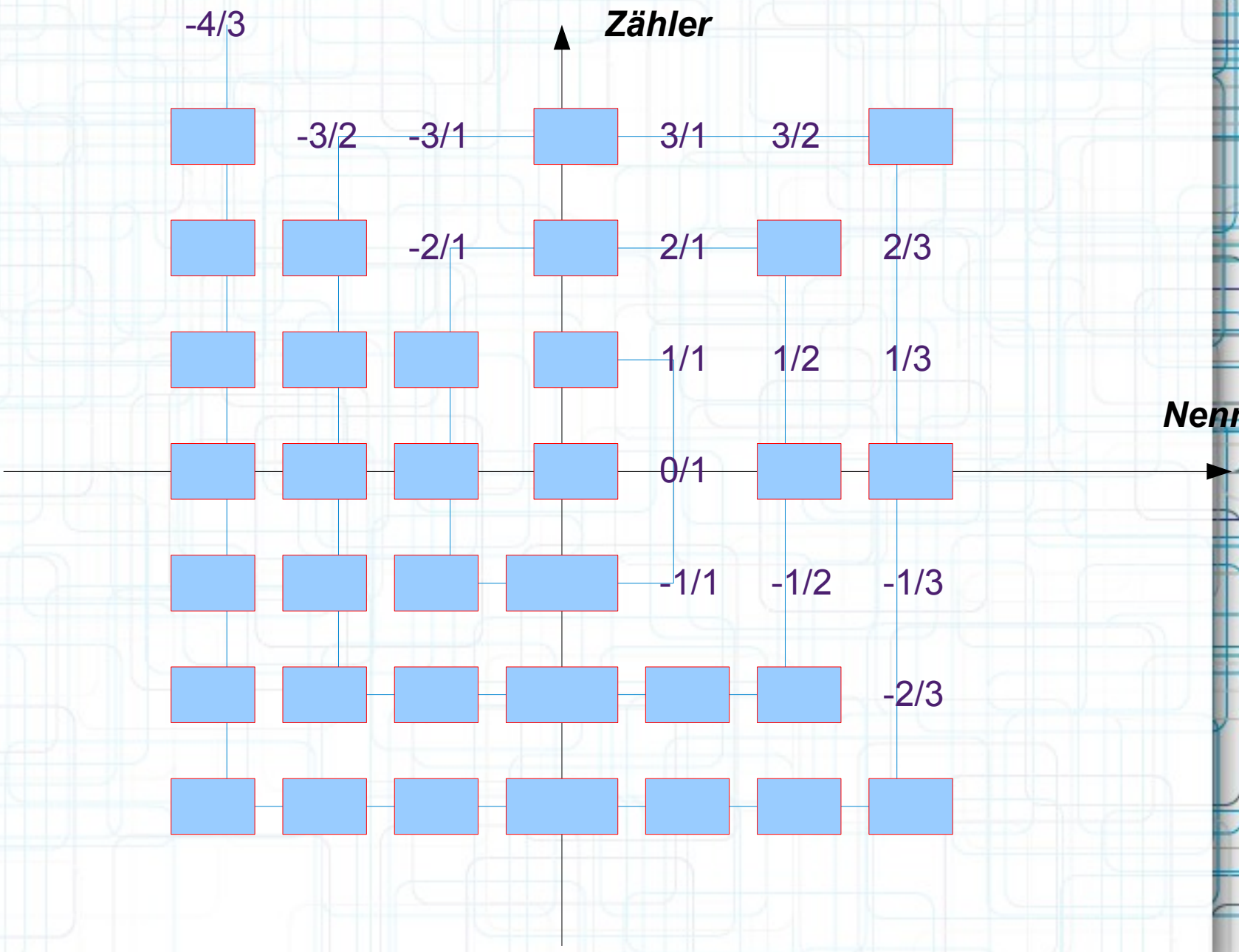
Problemstellung



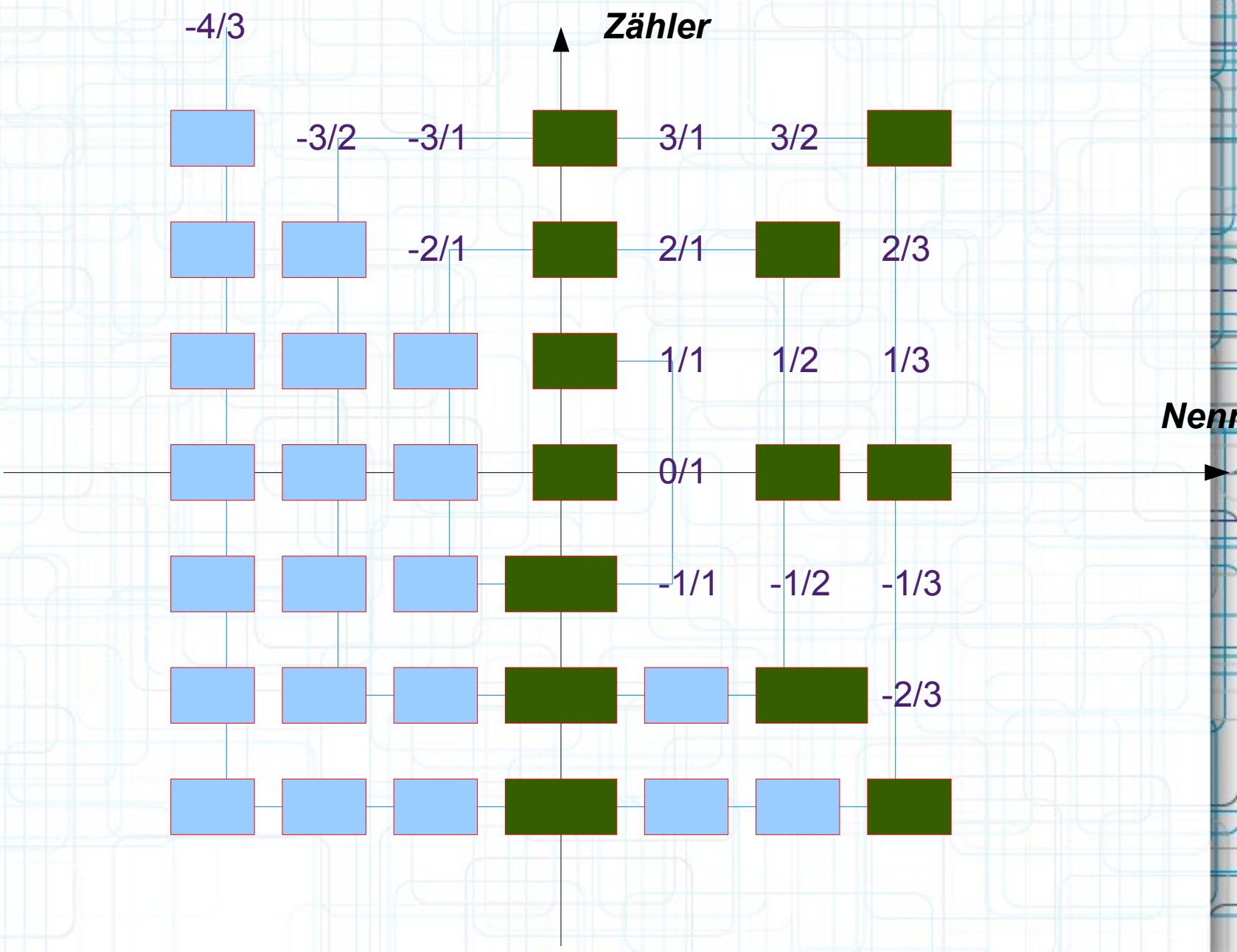
Problemstellung



Problemstellung

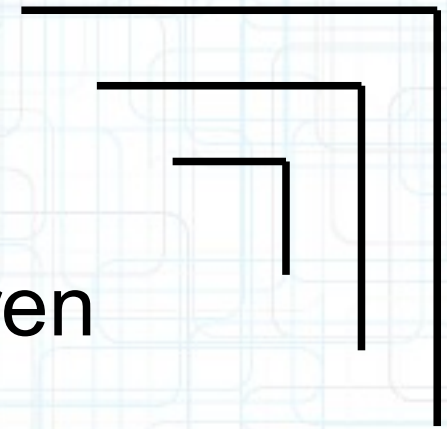


Grundlagen für die Lösung



Grundlagen für die Lösung

- Sonderfälle: Index 0 => 1/1
  1 => 0/1
  2 => -1/1

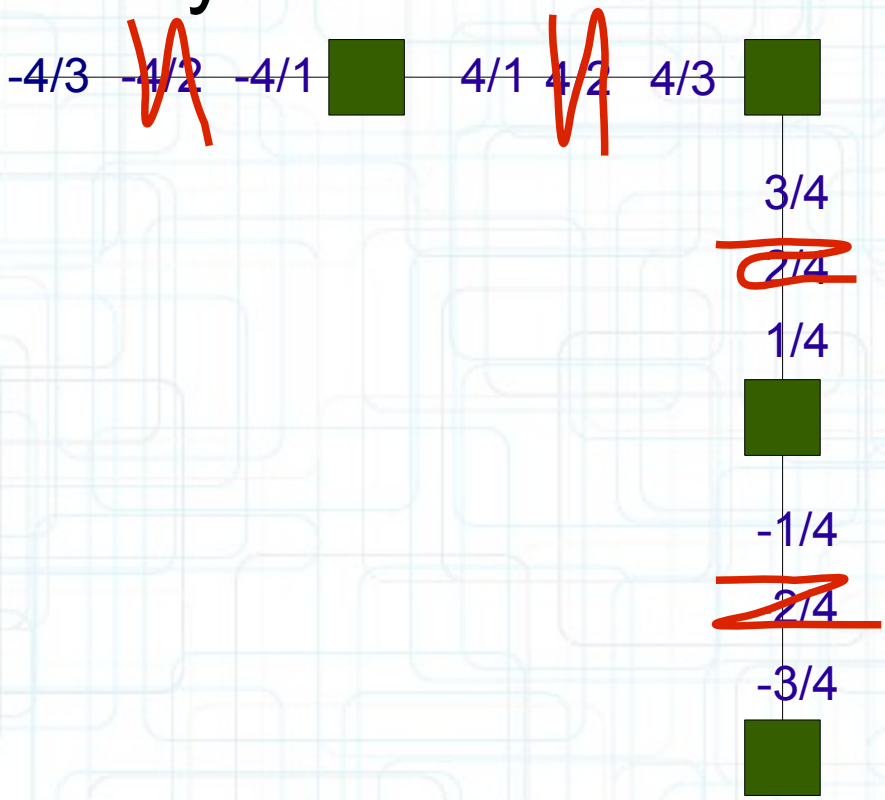


- Einteilung der Spirale in Sektoren

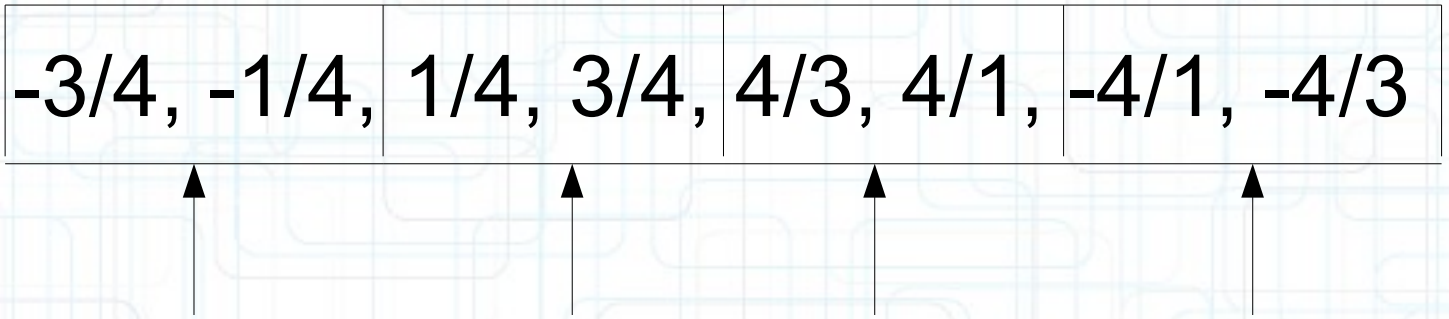
- Passende Identifikationsnummern für die Sektoren

Grundlagen für die Lösung

- Ausnutzung der Symmetrie:
- ID = 4



Grundlagen für die Lösung

- Idee: Lotung auf den IMMER ENTSTEHENDEN Bruch „rechts unten“ im jeweiligen Sektor
- Rückvollzug mit for-Schleifen:
- Schema:
- 

Grundlagen für die Lösung

- Ein Sektor besteht aus vier Quadranten
- Maximal entstehbare Brüche pro Quadrant:

  ID - 1

- Tatsächliche Anzahl entstehender Brüche:

  ID – 1 – (kürzbare Brüche)

Algorithmen

- Neue Brüche pro Sektor:

  4*(ID-1-Kills)

- Klassischer euklidischer Algorithmus zur Bestimmung der Kills

  ggT(m,n) = ggT(n mod m, m)

Implementierung

```
int input = scanner.nextInt();
int index = 2;
int quaterBack = 0;
int ID = 1;
```

Implementierung

```
while(index < input)
                quaterBack = 0;
                ID++;
                for(int i = 1; i < ID; i++)
                                if(ggT(i,ID) == 1) quaterBack++;
                index += 4*quaterBack;

int difference = index - input;
```

Implementierung

1. for-Schleife:

```
for(int i = ID – 1; i >= 1; i—)
        if(ggT(i, ID) == 1)
                if(difference == 0)
                        System.out.println(“-“+i+“/“+ID);
                difference -= 1;
                break;
                difference -= 1;
```

10920 – Spiral Tap

- Problemstellung
- Beispiel
- Grundlagen für die Lösung
- Algorithmen
- Implementierung

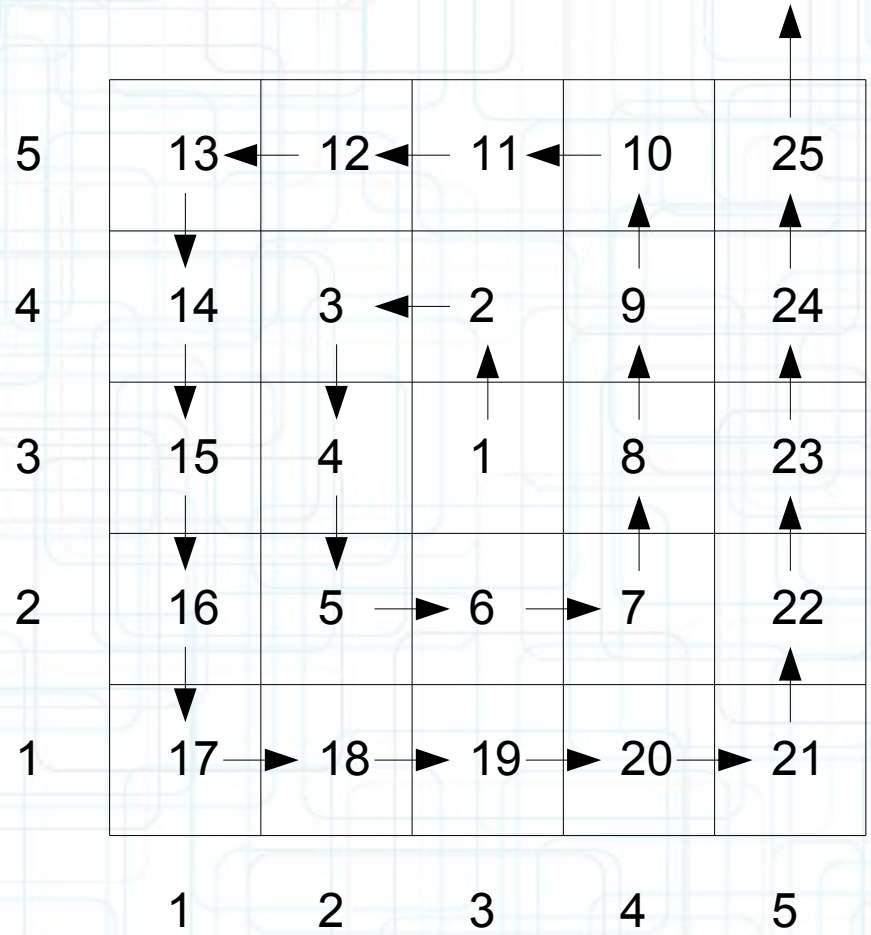
Problemstellung

- Input:
  - Größe des Feldes (ungerade Zahl)
  - Index
- Bedingung:
  - Startpunkt in der Mitte des Feldes
  - Spirale wird im Gegenuhrzeigersinn durchlaufen
- Output:
  - Spalte & Zeile des Indexes

Problemstellung

13	12	11	10	25
14	3	2	9	24
15	4	1	8	23
16	5	6	7	22
17	18	19	20	21

Problemstellung



Beispiel

ROW: 4 →

5	13	12	11	10	25
4	14	3	2	9	24
3	15	4	1	8	23
2	16	5	6	7	22
1	17	18	19	20	21
	1	2	3	4	5

↑
COLUMN 1

Grundlagen für die Lösung

1) Columns/Rows auf den Startpunkt legen:

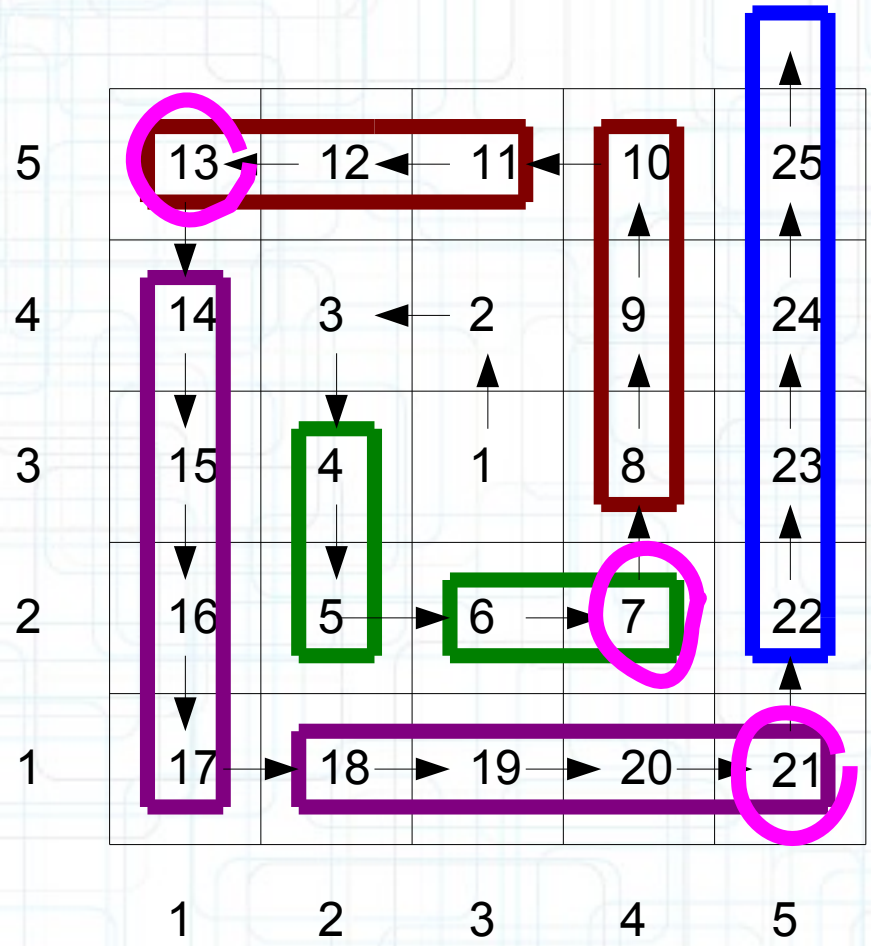
$$\frac{\text{Größe des Feldes}}{2} + 1$$

2) Ziel: Direkte Ermittlung der „Columns“ und „Rows“ über charakteristische Größen, die sich pro „Schleifendurchgang“ ändern

Grundlagen für die Lösung

- Charakteristische Größen:
  - Summand
  - Distanz zum Mittelpunkt
  - Horizontal oder Vertikal
  - Momentaner Index
  - Differenz
- Idee: Lotung auf einen Eckpunkt, und Rückvollzug auf die Reihen und Spalten über die charakteristischen Größen

Grundlagen für die Größen



Algorithmen/Implementierung

```
int distanceFromMid = 1;
int workingIndex = 3;
int adder = 1;
boolean bottomRight = false;
while(          )
{




}
```

Algorithmen/Implementierung

```
int distanceFromMid = 1;
int workingIndex = 3;
int adder = 1;
boolean bottomRight = false;
while(workingIndex < index)
{
    adder++;
    workingIndex += 2*adder;
    bottomRight = !bottomRight;
    if(!bottomRight) distanceFromMid++;
}
```

Algorithmen/Implementierung

```
int difference = workingIndex - index;
if(bottomRight)
{




}
else
```

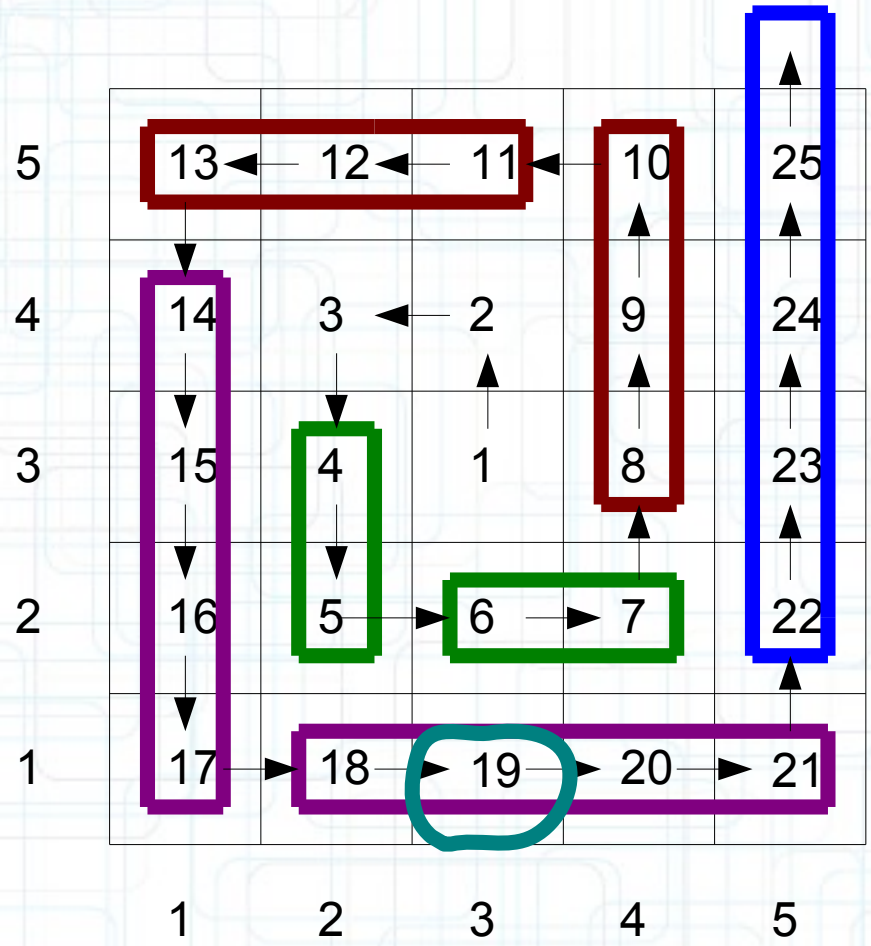
Algorithmen/Implementierung

```
int difference = workingIndex - index;
if(bottomRight)
{
if(difference <= adder)
{

}
if(difference > adder)
{

}
}
else
```

Algorithmen/Implementierung

```
if(difference <= adder)
{
	column = column - difference;
	column = column + distanceFromMid;
	row = row - distanceFromMid;
}
if(difference > adder)
{









}
```

Algorithmen/Implementierung

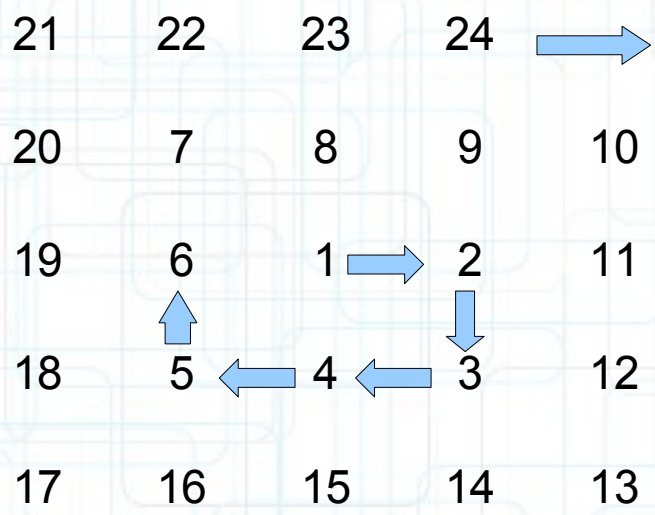


Algorithmen/Implementierung

```
if(difference <= adder)
{
	column = column - difference;
	column = column + distanceFromMid;
	row = row - distanceFromMid;
}
if(difference > adder)
{
	column = column - adder;
	difference = difference - adder;
	row = row + difference;
	column = column + distanceFromMid;
	row = row - distanceFromMid;
}
```

903 Spiral of Numbers

- Abbildung der natürlichen Zahlen auf eine Spirale



- Besonderheit:
  - Häufig Primzahlen auf den Diagonalen

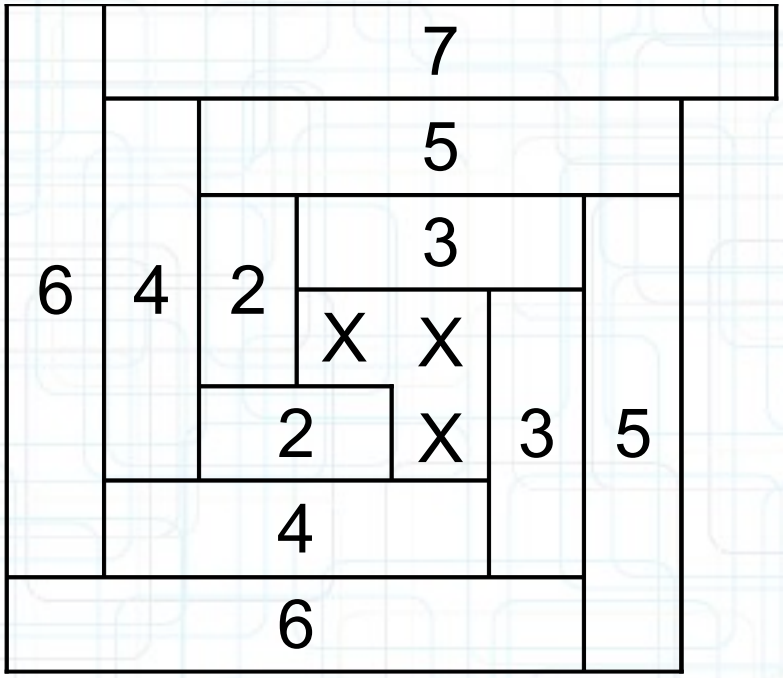
Aufgabenstellung

- Eingabe:
  - Index
- Ausgabe:
  - Alle Nachbarn des Index
- Beispiel: 7

21;22;23;
20;7;8;
19;18;17;

Erkenntnisse

- Folgendes Muster

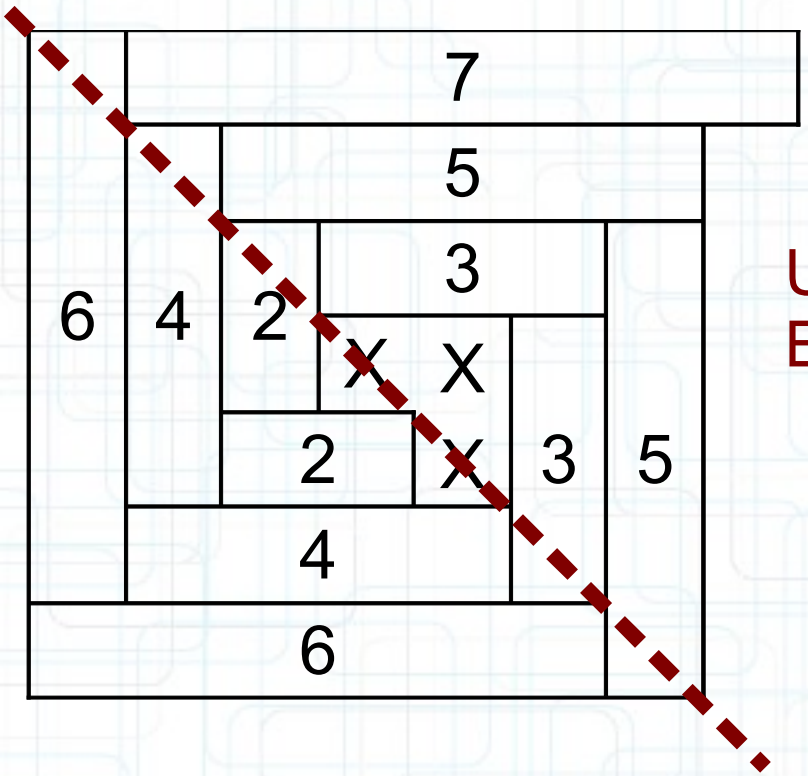


- Ermöglicht Bestimmung der Position

Erkenntnisse

- Folgendes Muster

Gerade Elementanzahl

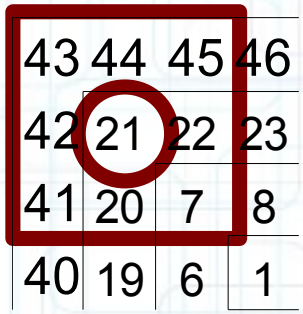


Ungerade Elementanzahl

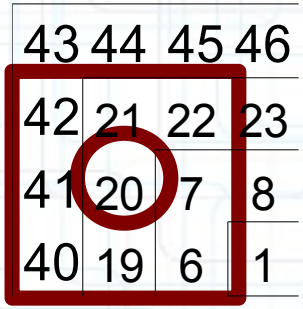
- Ermöglicht Bestimmung der Position

Erkenntnisse

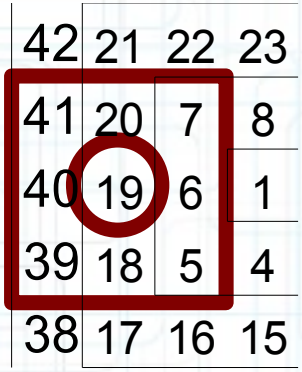
- Folgende Fälle bei Ausgabe:
  - Ecke



  - Unterhalb Ecke

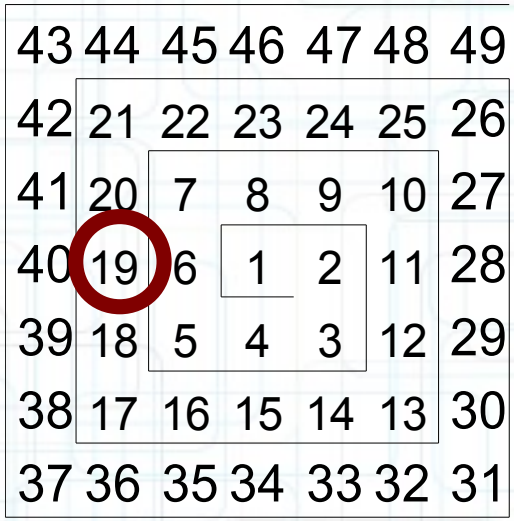


  - Ohne Ecke



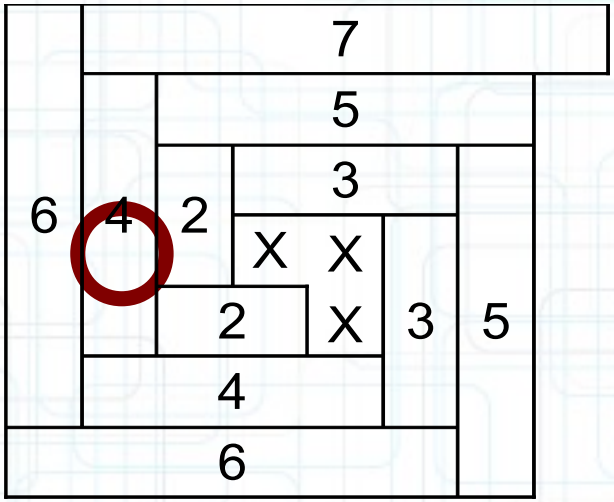
Beispiel

- Index: 19



Beispiel

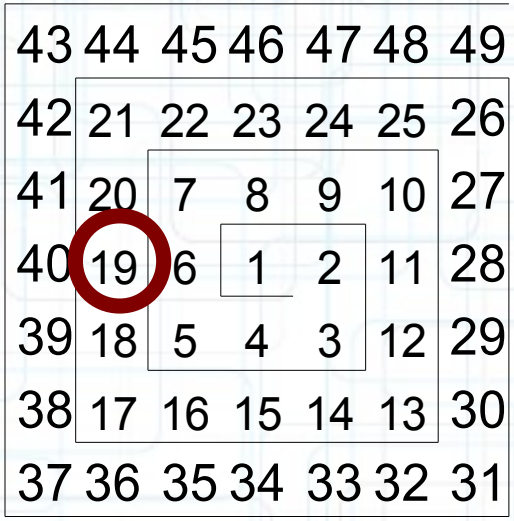
- Index: 19



- Block: 4
- Summe der Elemente: 21
- Differenz von Index und Summe: 2

Algorithmus

- Bestimmung der Nachbarelemente
  - Für jeden Ausgabefall getrennt
  - Insgesamt 16 Fälle
  - Bestimmung erfolgt über Blocknummer



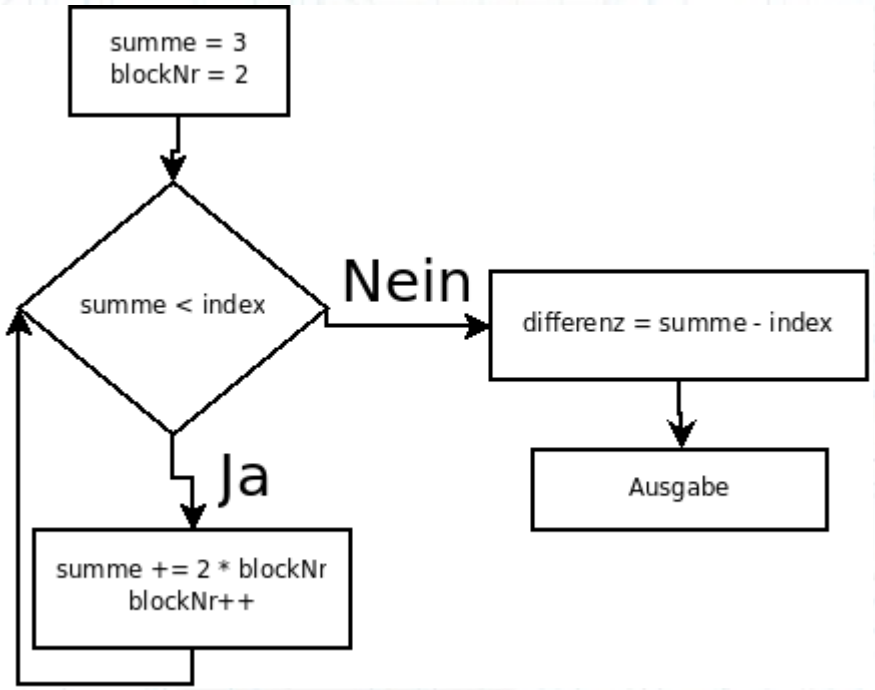
Bestimmung des rechten Nachbars

Index – 4 *(Blocknummer -1)+3

19 - 4 * (4 – 1)+3 = 6

Implementierung

- Berechnung der Blocknr. und Summe der Elemente



Vielen Dank für eure Aufmerksamkeit!