Tagesprogramm

Aufwand für wiederholte Berechnungen

Array als Zwischenspeicher

Pascalsches Dreieck

$$\binom{0}{0}$$

$$\binom{1}{0} \quad \binom{1}{1}$$

$$\binom{2}{0} \quad \binom{2}{1} \quad \binom{2}{2}$$

$$\binom{3}{0} \quad \binom{3}{1} \quad \binom{3}{2} \quad \binom{3}{3}$$

$$\binom{4}{0} \quad \binom{4}{1} \quad \binom{4}{2} \quad \binom{4}{3} \quad \binom{4}{4}$$

				1				
			1		1			
		1		2		1		
	1		3		3		1	
1		4		6		4		1

$$\binom{0}{0} = 1$$

$$\binom{1}{0} = 1$$

$$\binom{n}{n} = 1$$

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k} \quad \text{für} \quad n > k \quad \text{und} \quad k > 0$$

			1		
		1	1		
	1	2	1		
	1	3	3	1	
1	4	6	4	1	

Berechnung des Pascalschen Dreiecks

```
public class PascalschesDreieck {
    public static void main(String[] args) {
        long zeilen = 6;
        for (long n = 0; n < zeilen; n++) {
            for (long k = 0; k <= n; k++) {
                System.out.print(binom(n,k) + " ");
            }
            System.out.println();
        }
    }
    private static long binom(long n, long k) {
        if (k == 0 || k == n) {  // n == 0 impliziert k == 0
            return 1;
        }
        return binom(n-1, k-1) + binom(n-1, k);
    }
}
```

Aufgabe: Berechnungsdauer

Warum dauert die Berechnung der Werte das Pascalschen Dreiecks so lange?

A: Weil die Berechnung über Rekursion nicht so effizient ist wie über Schleifen.

B: Weil der Rechner langsamer wird wenn der Speicher voll ist.

C: Weil viele Werte nicht nur einmal sondern wiederholt berechnet werden.

D: Weil größere Zahlen länger zur Berechnung brauchen.

Aufgabe: Anzahl der Aufrufe

Schätzen Sie, wie viele Aufrufe von `binom` als Ergebnis 1 liefern:

A:	bei 6 Zeilen:	7	bei 30 Zeilen:	1023
B:	bei 6 Zeilen:	15	bei 30 Zeilen:	32.767
C:	bei 6 Zeilen:	31	bei 30 Zeilen:	33.554.431
D:	bei 6 Zeilen:	63	bei 30 Zeilen:	1.073.741.823

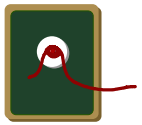
Berechnung neuer Zeile als Array

```
private static long[] nextLine(long[] upper) {

    int n = upper.length;
    long[] lower = new long[n + 1];

    lower[0] = 1;
    lower[n] = 1;

    for (int k=1; k<n; k++) {
        lower[k] = upper[k-1] + upper[k];
    }

    return lower;
}
```

Arrays in Variablen und Parametern

```
long[] upper
```


```
long[] lower;
```



```
lower = new long[n + 1];
```


```
return lower;
```

Aufgabe: Anzahl der Arrayzugriffe

Schätzen Sie, wie oft bei der Berechnung von `lower` auf jeden Wert in `upper` zugegriffen wird (für 30 Zeilen):

A: nur einmal

B: bis zu zweimal

C: bis zu 511 Mal

D: bis zu 32.767 Mal

Aufgabe: Negative Zahlen

Warum ergibt die Berechnung des Pascalschen Dreiecks negative Zahlen?

A: Weil das mathematisch so definiert ist.

B: Aufgrund eines Überlaufs.

C: Bei vielen Berechnungen kann schon einmal eine falsch sein.

D: Weil die Ausgabe sehr großer Zahlen Fehler liefert.