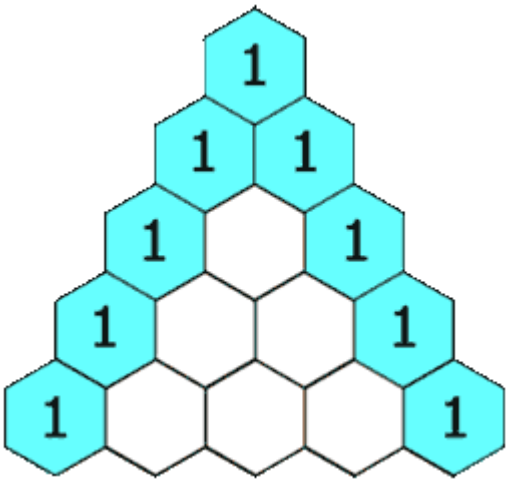
Übungen 3: Aufrufbäume

(A1) Pascalsches Dreieck

1)



Beim Pascalschen Dreieck werden die jeweils oberhalb liegenden Felder addiert und ergeben das darunter liegende Feld, wie in der Animation rechts zu sehen ist.

Man kann eine rekursive Funktion implementieren, die den Wert des Feldes in Zeile z und Spalte s des Pascalschen Dreiecks berechnet: `pascal(int z, int s): int`.

(A) Implementiere `pascal(int z, int s): int`.

Tipp 1

Ein mögliches Codegerüst könnte so aussehen:

```
public int pascal(int z, int s)
    {
       if (  ) {
           return 1;
       } else {
           return pascal( , );
       }
    }
```

Tipp 2

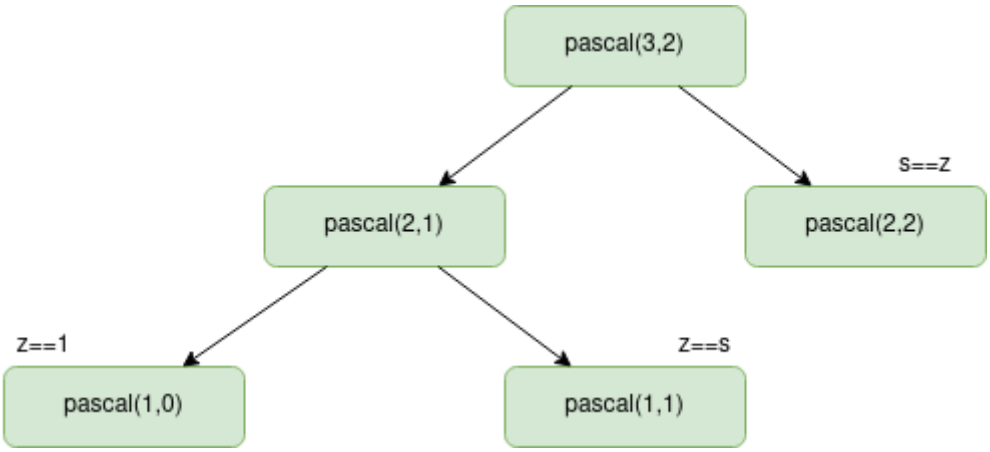
- Der Basisfall tritt immer dann ein, wenn das Element in Zeile 0 oder 1 oder in Spalte 0 oder z ist. Dann wird 1 zurückgegeben.
- Im Rekursionsfalll wird `pascal(z,s)` zwei mal aufgerufen, nämlich so, das die beiden oberhalb liegenden Felder addiert werden: `return pascal( , ) + pascal ( , )` - was muss da als

Argument in den Aufrufen stehen?

Lösungsvorschlag

```
public int pascal(int z, int s)
    {
        if ( z==0 || z==1 || s==0 || s==z ) {
            return 1;
        } else {
            return pascal(z-1, s-1) + pascal(z-1,s);
        }
    }
```

Die rekursiven Aufrufe können in einem **Aufruf-Baum** veranschaulicht werden, hier am Beispiel `pascal(3,2)` (Zählung von Zeile und Spalte beginnen in der Implementatin des Lösungsvorschlags bei 0, d.h. das Element an der Spitze des Dreiecks ist `pascal(0,0)=1`):



(A2) Ficonacci mit Baum

Die Fibonacci-Funktion (auch als Fibonacci-Folge bezeichnet) ist definiert als:

$$f(n)=\begin{cases} 0 & \text{wenn} \quad n=0 \\ 1 & \text{wenn} \quad n=1 \\ f(n-1)+f(n-2) & \text{sonst} \end{cases}$$

Die ersten Werte der Fibonacci-Funktion sind 0, 1, 1, 2, 3, 5, 8, 13, … Die Definition kann einfach in eine rekursive Methode übersetzt werden:

```
fib(n: int): int
  wenn n ≤ 1:
    gib n zurück
  sonst:
    gib fib(n-1) + fib(n-2) zurück
```

(A) Gib die vier Werte der Fibonacci-Folge an, die auf das Folgenglied 13 folgen.

(B) Stelle die ausgeführten Methodenaufrufe bei der Ausführung von `fib(4)` als Baum dar.

(C) Begründe, warum die Anzahl der Methodenaufrufe für `fib(n)` weniger als $2^{n+1>}$ beträgt.

(D) Implementiere eine Methode `fibIterativ(n: int): int`, die dasselbe Ergebnis wie `fib` berechnet, die aber ohne Rekursion arbeitet.

1) https://commons.wikimedia.org/wiki/File:PascalTriangleAnimated2.gif

From:
https://www.info-bw.de/ -

Permanent link:
https://www.info-bw.de/faecher:informatik:oberstufe:algorithmen:rekursion:uebungen03:start?rev=1675617919

Last update: **05.02.2023 17:25**