

$\rightarrow t$

1W 2W 3W 4W 5W 6W 7W

1 1 2 3 5 8 13

1 1

• Bezug auf "sich selbst"

Rekursionsanker

```
int anzahl_Flaeschen (int wocke) {
```

```
    int w_vv = 1;
```

```
    int w_v = 1;
```

```
    if (woche <= 2)
```

```
        return 1;
```

```
    for (int w = 3; w <= wocke; w++) {
```

```
        int t = wvv;
```

```
        w_vv = w_v + w_vv;
```

```
        w_v = t
```

```
}
```

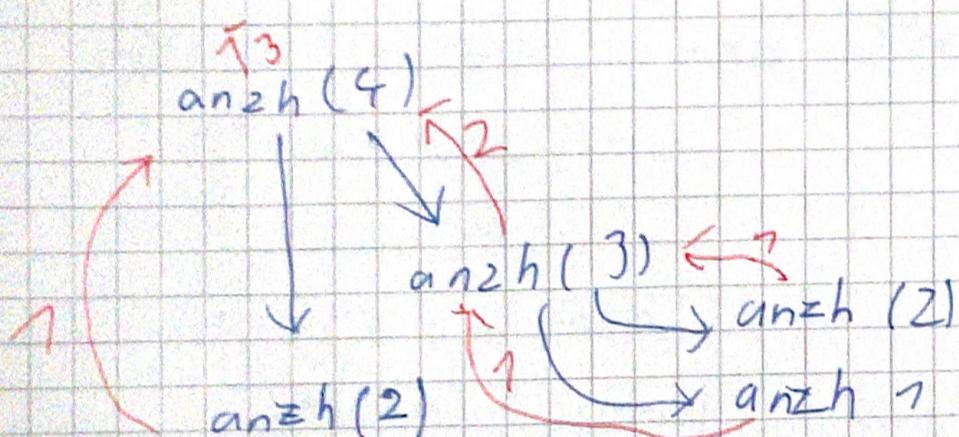
```
return w_v;
```

```
}
```

SEN

```
int anzahl_haeschen (int woch) {  
    if (woche <= 2) {  
        return 1;  
    }
```

```
    return anzahl_haeschen (woche - 1) + anzahl_haeschen  
(woche - 2);  
}
```



$$n! = \underbrace{1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdots (n-1)}_{(n-1)!} \cdot n$$

$$n! = (n-1)! \cdot n$$

$$1! = 1$$

```
int fact (int n) {
```

```
    if (n == 1)
```

```
        return 1;
```

```
    return fact (n - 1) * n;
```

```
}
```

fact(4)
 4 * fact(3)
 3 * fact(2)
 2 * fact(1)
 1

SE 1

Binäre Suche

0	1	2	3	4	5	6	7
2	3	5	7	11	13	17	19

↑
n

Suche: 17

2	3	5	7	11	13	17	19
---	---	---	---	----	----	----	----

↑
n

2	3	5	7	11	13	17	19
---	---	---	---	----	----	----	----

↑

int search (int element, int *a, int low; int high) {

if (low > high)

return -1;

int m = (low + high) / 2;

if (*a + m) == element)

return m;

if (elements * (a + m) <

return search (element, a, low, m - 1);

} else {

return search (element, a, m + 1, high);

}

3