Algo-Tutorium 3

05.11.2021

Check-In

Tipps

- LaTeX: Symbole finden

Detexify LaTeX handwritten symbol recognition (kirelabs.org)

- Bitte genau arbeiten ☺
- Viel Feedback -> Hilfestellung zur Verbesserung
- 3b) war auf meinem Blatt nicht aktualisiert. Sollte aber bei allen nachkorrigiert sein.

Array

- Länge n
- Anordnung von n Objekten des selben Typs in fortlaufenden Adressen im Speicher

- Nachteil: Größe ist fest
- Vorteil: schneller Zugriff in O(1)

Liste – einfach verkettet

- Können ihre Größe (= Länge) ändern
- Zugriff auf bestimmtes Element kann dauern (Finden in O(n))

- Elemente besteht aus :
  - Schlüssel
  - Zeiger next auf das nächste Element
- Zeiger auf das Startelement der Liste

- Braucht wenig Speicherplatz

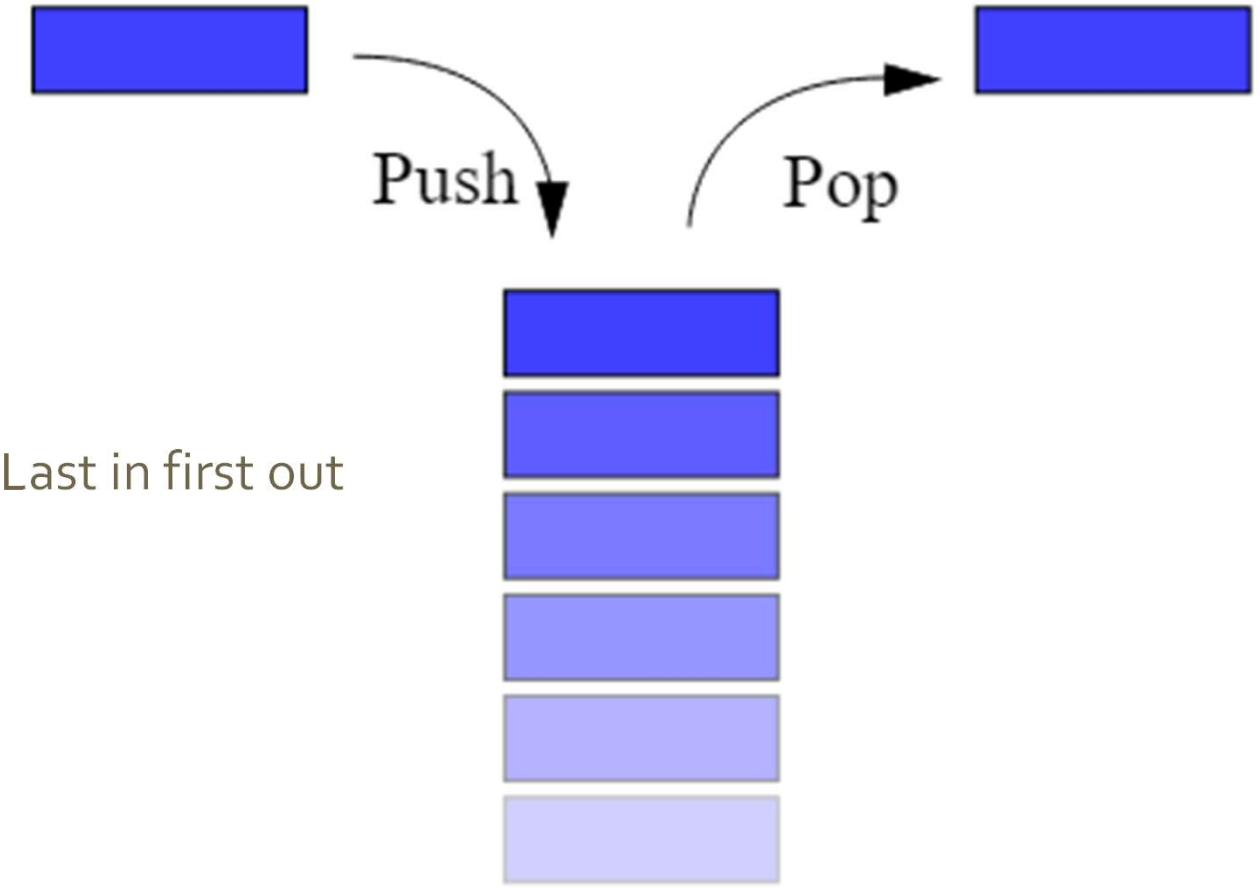
Liste - doppelt verkettet

- Elemente besteht aus :
  - Schlüssel
  - Zeiger next auf das nächste Element
  - Zeiger prev auf das Vorgängerelement

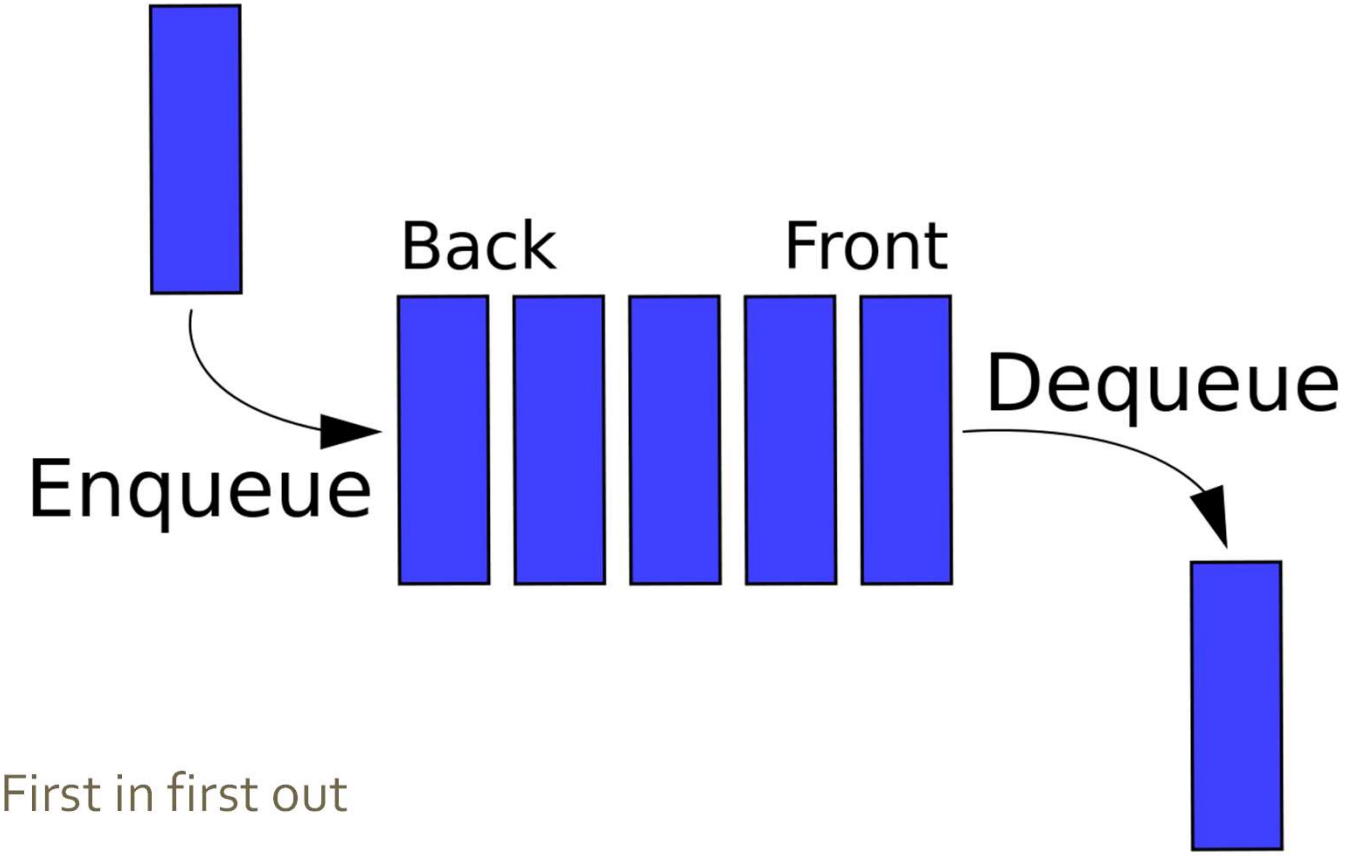
- Braucht mehr Speicher
- Löschen von Elementen einfacher

Stack

- Last in first out

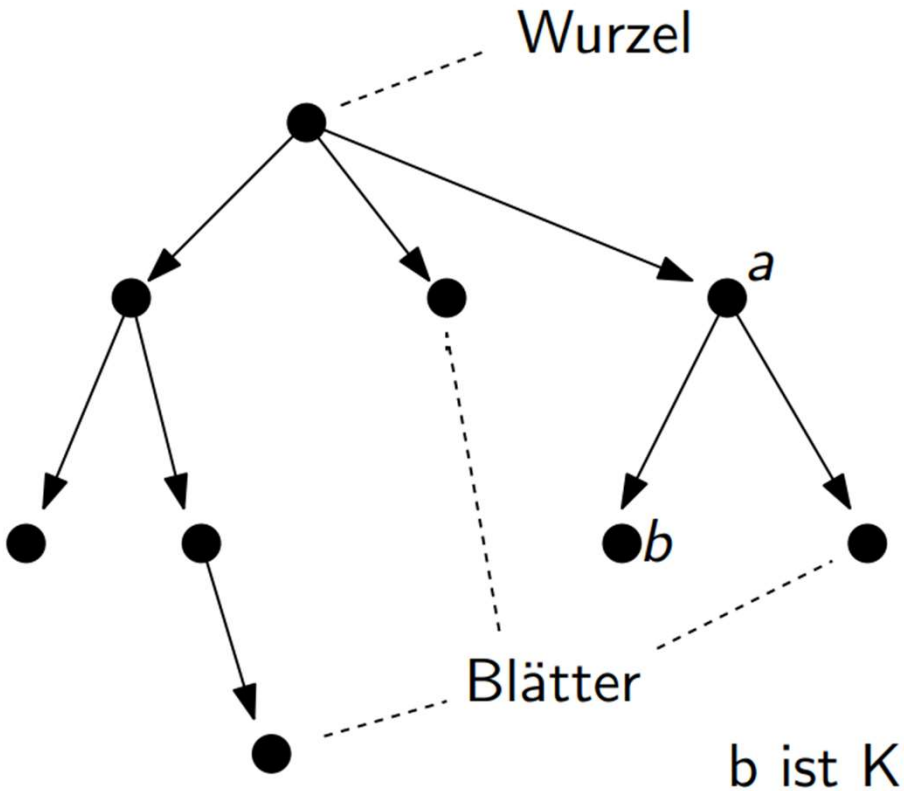


Queue



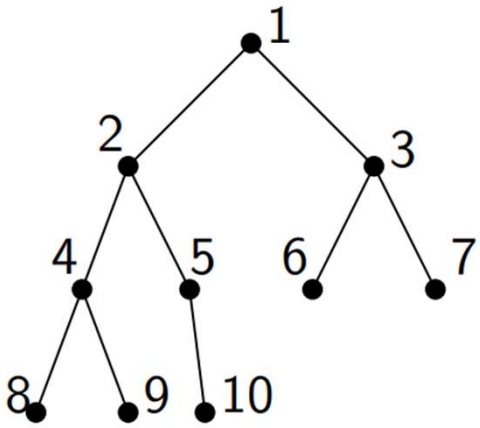
- First in first out

Bäume



b ist Kind von a.
a ist Vater/parent von b.

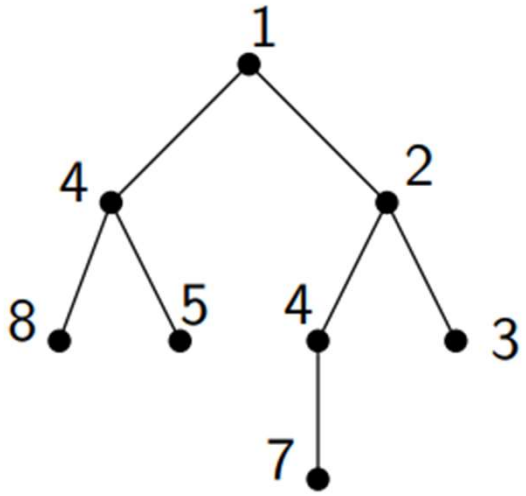
(vollständiger) Binärbaum



- Binärbaum:
  - Jeder Knoten hat höchstens 2 Kinder
  - Tiefe/Höhe O(log(n))
- Vollständig:
  - Alle Level außer dem untersten gefüllt. Dort stehen die Knoten möglichst weit links.
- Voll:
  - Wie vollständig, nur: das unterste Level ist komplett voll.
- Indizierung bei vollständigen und vollen Binärbäume
  - Knoten = i
  - Elternknoten = $\left\lfloor \frac{i}{2} \right\rfloor$
  - Kindknoten = $2i,\ 2i+1$

Heap

- Zeichnen wir als Baum
- Implementieren wir als Array
- Sinn: Verwaltung einer Menge mit Ordnungsrelation.
- Kompromiss zwischen ungeordnetem und geordnetem Array
- Höhe ist O(log n), da vollständiger Baum
- **Heapeigenschaft**: Für Knoten u und v mit Vater(v) = u gilt:
  S[u] ≤ S[v]



$$S = \{1, 2, 3, 4, 5, 4, 7, 8\}$$

Heap – Element einfügen

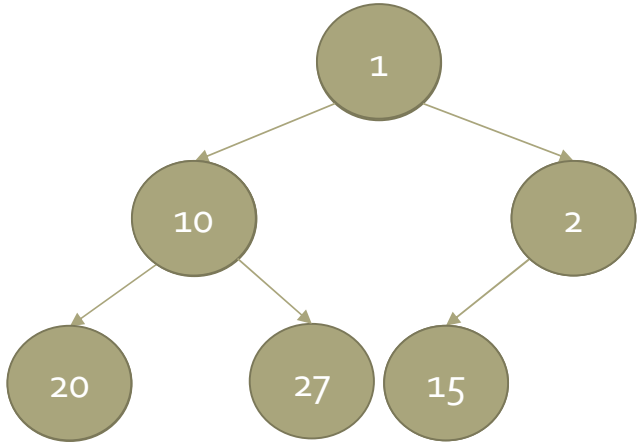
```
Insert(a, H):
  sei n die Größe von H;
  n ← n + 1;
  H(n) ← a;
  i ← j ← n;
  while (i > 1) {
     j ← ⌊j/2⌋;
     if (H[j] > H[i]) {tausche H[i] und H[j]; i ← j};
     else i ← 1;
     };
```

- Ablauf:
  - Füge neues Element als **„rechtestes" Blatt** ein
  - Falls Heapeigenschaft verletzt ist:
    Vergleiche und tausche neues Element so lange mit Elternknoten bis der Elternknoten kleiner als der eingefügte Knoten ist
- Laufzeit: O(log(n))

Heap – Element einfügen

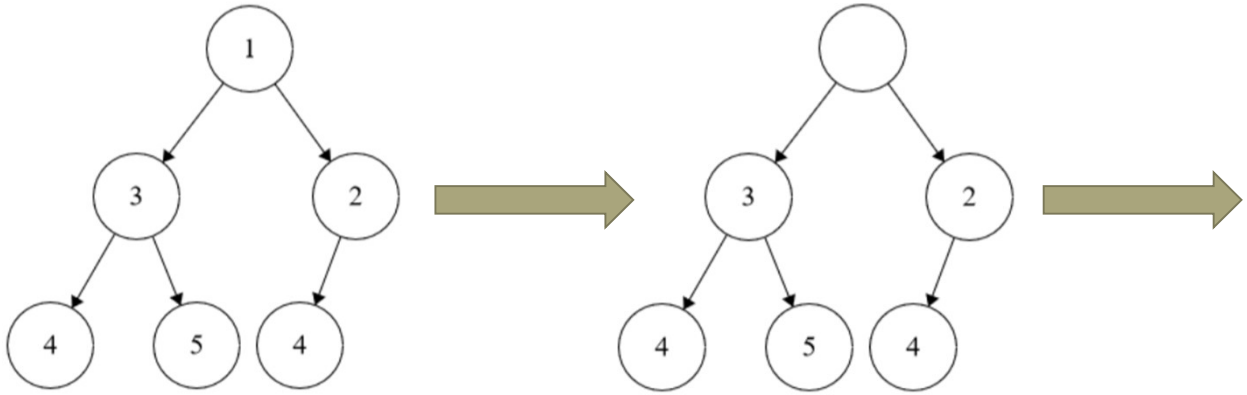
- Ablauf:
  - Füge neues Element als **„rechtestes" Blatt** ein
  - Falls Heapeigenschaft verletzt ist:
    Vergleiche und tausche neues Element so lange mit Elternknoten bis der Elternknoten kleiner als der eingefügte Knoten ist

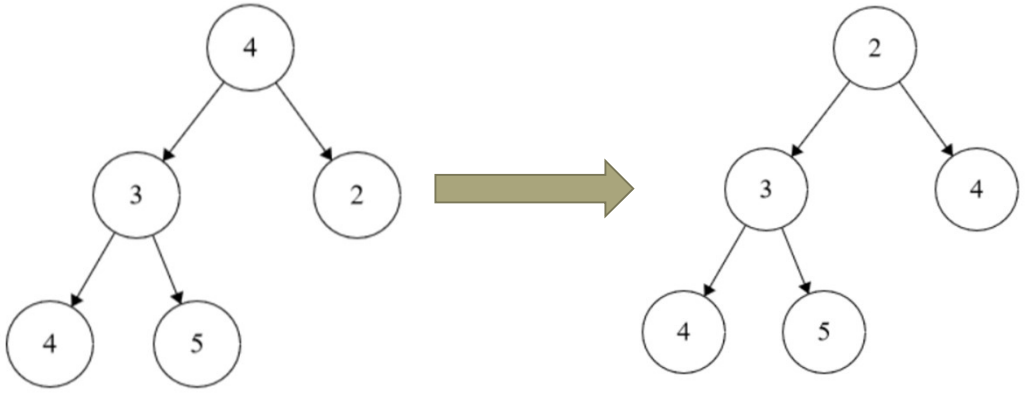
- Beispiel:
  - Füge folgende Zahlen ein: 1, 27, 2



Heap – Minimum extrahieren

1. Suche Minimum → Wurzel $O(1)$
2. Entferne Minimum aus Heap. Repariere Heap:
   - Nimm Blatt und füge es in Wurzel ein.
   - Lass es nach unten sinken (Vergleich mit kleinstem Kind)





Laufzeit: O(log(n))

PB

Aufgabe 1: Türme von Hanoi

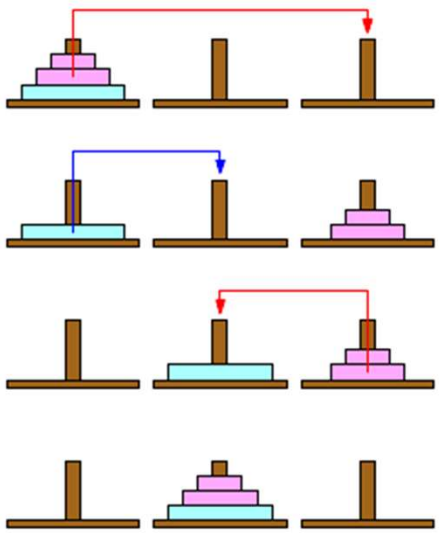
★ ★ ★ ☆ ☆

— *Vorbereitung auf Aufgabe 2 des Übungsblattes* —

Die Türme von Hanoi ist ein bekanntes Puzzle, bei dem 3 Säulen `links`, `mitte` und `rechts` gegeben sind, wobei auf `links` n Scheiben mit von oben nach unten größer werdenden Radien liegen. Die Aufgabe ist es nun alle n Scheiben von `links` zu `mitte` zu bewegen. In jedem Zug darf eine Scheibe oben von einer der 3 Säulen entfernt und auf eine andere Säule gelegt werden. Dabei darf niemals eine größere auf eine kleinere Scheibe gelegt werden.

Da Scheiben immer oben auf Säulen gelegt und auch von oben von Säulen entfernt werden, entsprechen `links`, `mitte` und `rechts` jeweils einer Stack-Datenstruktur, und wir nennen Säulen, die so funktionieren *Stack-Säulen*. Insofern ist ein Zug die Aktion `x.push(y.pop())` für `x,y` $\in$ `{links,mitte, rechts}`.

Es sind $2^n - 1$ Züge nötig und es gibt Algorithmen, die diese untere Schranke erreichen. Einer dieser Algorithmen funktioniert wie folgt (vgl. auch mit der Abbildung rechts): Um n Scheiben von Säule `a` zu Säule `b` zu bewegen, werden zunächst $n - 1$ Scheiben von Säule `a` zu Säule `c` bewegt. Danach wird die n-te Scheibe von `a` auf `b` bewegt. An Ende werden nun die $n - 1$ Scheiben von `c` auf `b` bewegt. Dabei gilt, dass `a,b,c` $\in$ `{links, rechts, mitte}`.



a) Geben Sie den oben beschriebenen Algorithmus in Pseudocode an.

b) Überzeugen Sie sich davon, dass jeder Zug des oben beschriebenen Algorithmus ein gültiger Zug ist.

c) Stellen Sie die Anzahl der vom Algorithmus ausgeführten Züge als Rekursionsformel dar und zeigen Sie, dass dies $2^n - 1$ Züge sind.

Aufgabe 2: Rechnen mit Summen

★ ★ ★ ★ ☆

— Vorbereitung auf Aufgabe 3 des Übungsblattes —

Zeigen Sie, dass

$$\sum_{i=1}^{n} i \cdot k^i = \frac{k(nk^{n+1} - (n+1)k^n + 1)}{(k-1)^2}$$

für alle natürlichen Zahlen $n \geq 1$ für eine gegebene Konstante $k > 1$.
Tipp: Nutzen Sie vollständige Induktion über n.

Besprechung PB

Nr.1 a)

- Algorithm: **Bewege(n, von, zu, spare)**
- INPUT: Anzahl Scheiben n, Ausgangssäule von, Zielsäule zu, Extrasäule spare
- OUTPUT: none

```
1 If n=1 then
2     zu.push(von.pop());
3 else
4     Bewege(n-1, von, spare, zu);
5     zu.push(von.pop());
6     Bewege(n-1, spare, zu, von);
7 end
```

Die Türme von Hanoi - YouTube

Nr.1 b)

```
1 If n=1 then
2     zu.push(von.pop());
3 else
4     Bewege(n-1, von, spare, zu);
5     zu.push(von.pop());
6     Bewege(n-1, spare, zu, von);
7 end
```

- Invariante ist:

Nach Ausführen von **Bewege(n, von, zu, spare)** wurden die kleinsten Scheiben von **von** zu **zu** bewegt und dann dort ordentlich gestapelt.

- Beweis via Induktion:
- IA: n=1. Offensichtlich erfüllt.
- IV: Die Behauptung gelte für ein bel. aber festes $n - 1 \in \mathbb{N}$

Nr.1 b)

```
1 If n=1 then
2     zu.push(von.pop());
3 else
4     Bewege(n-1, von, spare, zu);
5     zu.push(von.pop());
6     Bewege(n-1, spare, zu, von);
7 end
```

- IS: $n-1 \rightarrow n$.
  - Nach IV: nach Z.4 sind die n-1 kleinsten Scheiben von **von** zu **spare** gestapelt worden.
  - Z.5: n-te Scheibe wird von **von** zu **zu** bewegt (Zielposition)
  - Die IV garantiert: nach Z.6 sind die n – 1 kleinsten Scheiben von **spare** zu **zu** gestapelt worden.
- Invariante erfüllt
- Beachte:

Züge in rekursiven Calls können immer ausgeführt werden, da nur Scheiben bewegt werden, die kleiner sind als die n-te.

Nr.1 c) – Rekursionsformel Bewege(n, ·, ·, ·)

```
1 If n=1 then
2     zu.push(von.pop());
3 else
4     Bewege(n-1, von, spare, zu);
5     zu.push(von.pop());
6     Bewege(n-1, spare, zu, von);
7 end
```

- Für n=1:
  - Z.2 Aufwand 1
- Für n > 1:
  - Z.4 Bewege(n−1, ·, ·, ·) also Z(n-1)
  - Z.5 Aufwand 1
  - Z.6 Bewege(n−1, ·, ·, ·) also Z(n-1)
- Formel: $Z(n) = 2 \cdot Z(n-1) + 1$ und $Z(1) = 1$.
- Setzt man das wiederholt ein, erhält man

$$Z(n) = \cdots = 2^{n-1} \cdot Z\big(n-(n-1)\big) + \sum_{i=0}^{n-2} 2^i = \sum_{i=0}^{n-1} 2^i = 2^n - 1$$

Nr.1 c) – Rekursionsformel Bewege(n, ·, ·, ·)

- Rekursionsformel: $Z(n) = 2 \cdot Z(n-1) + 1$ und $Z(1) = 1$.
- Wiederholt einsetzen:

$$Z(n) = 2 \cdot Z(n-1) + 2^0$$

$$= 2^2 \cdot Z(n-2)) + 2^1 + 2^0$$

$$= \ldots$$

$$= 2^{n-1} \cdot Z\big(n - (n-1)\big) + \sum_{i=0}^{n-2} 2^i$$

$$= \sum_{i=0}^{n-1} 2^i = 2^n - 1$$

Nr.1 c)

- Rekursionsformel: $Z(n) = 2 \cdot Z(n-1) + 1$
- Behauptung: $Z(n) = 2^n - 1$

- Beweis per Induktion:

  - IA: n=1.
    $Z(1) = 2 - 1 = 1$

  - IV: Sei $Z(n-1) = 2^{n-1} - 1$ mit $n - 1 \in \mathbb{N}$ beliebig aber fest.

  - IS: $n - 1 \to n$
    $Z(n) = 2Z(n-1) + 1 =^{IV} 2 \cdot (2^{n-1} - 1) + 1 = 2^n - 1$

Nr.2 - IA

- n = 1.

$$\sum_{i=1}^{1} i \cdot k^i = 1 \cdot k^1 = k$$

$$= \frac{k \cdot (k-1)^2}{(k-1)^2}$$

$$= \frac{k(1 \cdot k^2 - 2k^1 + 1)}{(k-1)^2}$$

Nr.2 - IV

- Für ein $n \geq 1$ gelte:

$$\sum_{i=1}^{n} i \cdot k^i = \frac{k(nk^{n+1} - (n+1)k^n + 1)}{(k-1)^2}$$

Nr.2 - IS

- Betrachte n+1.

$$\sum_{i=1}^{n+1} i \cdot k^i = \sum_{i=1}^{n} i \cdot k^i + (n+1)k^{n+1}$$

$$\overset{IV}{=} \frac{k(nk^{n+1} - (n+1)k^n + 1)}{(k-1)^2} + \frac{(n+1)k^{n+1} \cdot (k-1)^2}{(k-1)^2}$$

$$= \text{(ausmultiplizieren, umformen)} \ldots$$

$$= \frac{k((n+1)k^{n+2} - (n+2)k^{n+1} + 1)}{(k-1)^2}$$

Nr.2 - IS

- Betrachte n+1.

$$\sum_{i=1}^{n+1} i \cdot k^i = \sum_{i=1}^{n} i \cdot k^i + (n+1)k^{n+1}$$

$$\overset{IV}{=} \frac{k(nk^{n+1} - (n+1)k^n + 1)}{(k-1)^2} + \frac{(n+1)k^{n+1} \cdot (k-1)^2}{(k-1)^2}$$

$$= \frac{(nk^{n+2} - (n+1)k^{n+1} + k) + (n+1)k^{n+1}(k^2 - 2k + 1)}{(k-1)^2}$$

$$= \frac{nk^{n+2} - (n+1)k^{n+1} + k + (n+1)k^{n+3} - 2(n+1)k^{n+2} + (n+1)k^{n+1}}{(k-1)^2}$$

$$= \frac{(n+1)k^{n+3} + nk^{n+2} - (2n+2)k^{n+2} - (n+1)k^{n+1} + (n+1)k^{n+1} + k}{(k-1)^2}$$

$$= \frac{(n+1)k^{n+3} - (n+2)k^{n+2}0k^{n+1} + k}{(k-1)^2}$$

$$= \frac{k((n+1)k^{n+2} - (n+2)k^{n+1} + 1)}{(k-1)^2}$$

Check-In nächste Woche