

PREDECESSOR (S, x) : „Finde das nächstkleinere Element“

Für ein in x stehendes Element in S, bestimme ein Element von nächstgrößeren Wert in S.

(Annahme: Totalordnung)

Ausgabe: Zeiger y auf Element

NIL, wenn x Minimum von S angibt.

Wie nimmt man das vor?

Wie lange dauert das? → kann unterschiedlich lange sein!

Beispiel : Sortierte Unterlagen → Suche Blatt (1) ~ vorne
 Suche Blatt (62) ~ hinten
 Suche Blatt (32) ~ Mitte
 Suche Blatt (8) → Mitte von vorderer Hälfte von vorderer Hälfte

Unsortierte Unterlagen → immer alles durchgehen!

Laufzeit : In Abhängigkeit der Kardinalität n der Menge S

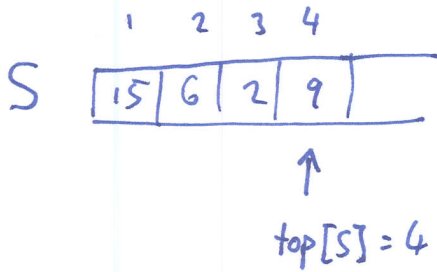
Also : $O(1)$: konstante Zeit → sehr schnell (z.B. Blatt 1 in sortiertem Manuskript)

$O(\log n)$: logarithmische Zeit → schnell (z.B. Blatt 8 in sortiertem Manuskript)

$O(n)$: lineare Zeit → langsam (Suche in unsortierten M)

4.2 Stapel und Warteschlange

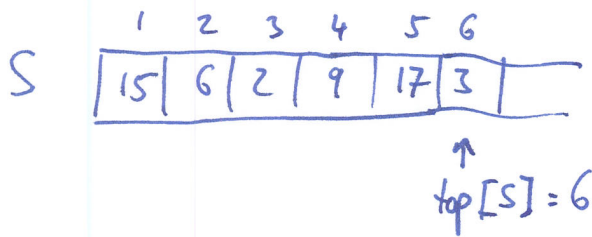
Stapel



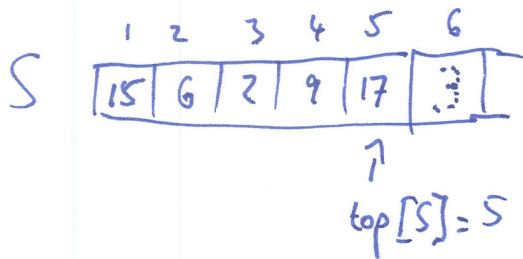
Stapel mit 4 Elementen

~~Werten~~ (S)

Einfügen von 17 und 3 liefert



Löschen?! Nur für letztes („oberstes“) Element möglich:



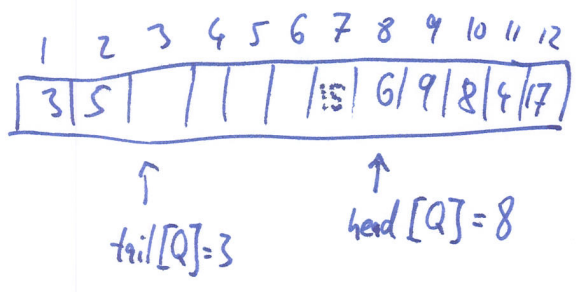
Spezielle Namen: PUSH
und POP!

In Pseudocode:

```
STACK-EMPTY (S)
1 if top[S] = 0
  then return WAHR
  else return FALSCH
```

} Absicherung gegen Zugriff auf leeren Stapel

Entfernen ^{nur von vorne!} liefert



↳ Pseudocode; hier ohne Fehlerkontrolle

Einfügen:

ENQUEUE (Q, x)

```

Q [tail[Q]] := x
if tail[Q] = länge[Q]
then tail[Q] := 1
else tail[Q] := tail[Q] + 1

```

Löschen

DEQUEUE (Q)

```

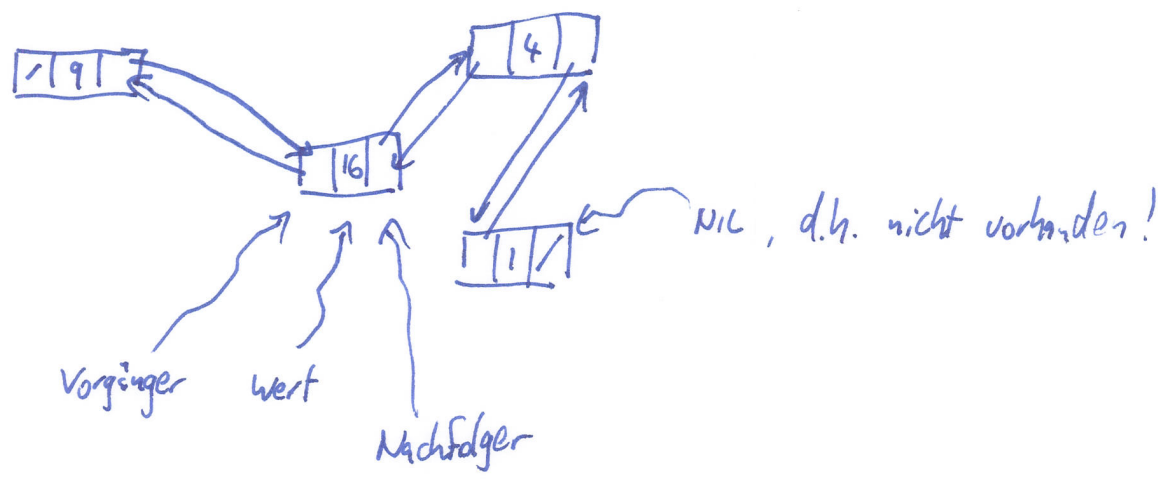
x := Wahrscheinlich Q [head[Q]]
if head[Q] = länge[Q]
then head[Q] := 1
else head[Q] := länge[Q] + 1

```

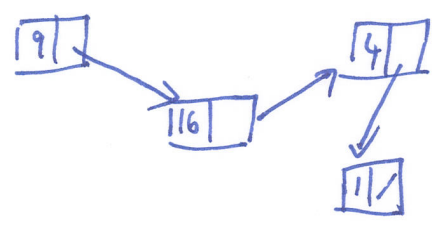
Aufwand wieder $O(1)$, aber Suchen etc. aufwändiger!

4.3 Verkettete Listen

Idee: Ordne Objekte nicht explizit in aufeinanderfolgenden Speicherzellen an, sondern gib jeweils Vorgänger und Nachfolger an:



Einfach verkettete Liste: Nur Nachfolger wird gemerkt



Doppelt verkettete Liste: Nachfolger und Vorgänger!

Alle Operationen werden ermöglicht:

LIST-SEARCH (L, k)

(Suche nach Objekt)

67

```
1 x := head [L]
2 while x ≠ NIL und Wert[x] ≠ k
3   do x := nachf [x]
4 return x
```

Laufzeit: $O(n)$

LIST-INSERT (L, x)

(Einfügen vorne)

```
1 nachf [x] := head [L]
2 if head [L] ≠ NIL
3   then vorg [head [L]] := x
4   head [L] := x
5 vorg [x] := NIL
```

Laufzeit: $O(1)$

LIST-DELETE (L, x)

(Entfernen irgendwo)

```
1 if vorg [x] ≠ NIL
2   then nachf [vorg [x]] := nachf [x]
3   else head [L] := nachf [x]
4 if nachf [x] ≠ NIL
   then vorg [nachf [x]] := vorg [x]
```

Laufzeit $O(1)$