Lineare Speicher

```
signature HEAP = sig
   exception Address
   exception OutOfMemory
   type address = int
   type index = int
   val new : int -> address
   val sub : address -> index -> int
   val update : address -> index -> int -> unit
   val release : address -> unit
   val show : unit -> (address * int) list
end
```

Darstellung von Listen

```
fun putList nil = ~1
  | putList (x::xr) = dump [x, putList xr]

fun getList a = if a = ~1 then nil
        else sub a 0 :: getList (sub a 1)
```

Beispiel: `putList [1,2,3]`

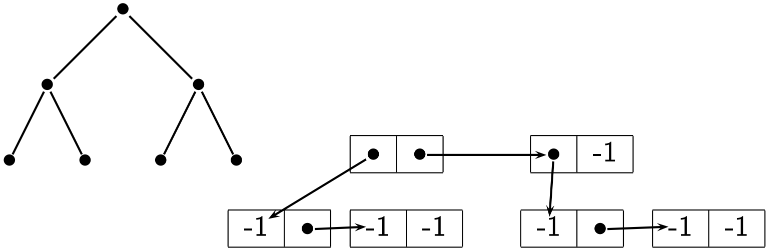
3	-1	2	0	1	2
0	1	2	3	4	5

Darstellung von Bäumen

```
fun putList nil = ~1
   | putList (x::xr) = dump [x, putList xr]
fun getList a = if a = ~1 then nil
        else sub a 0 :: getList (sub a 1)

datatype tree = T of tree list

fun putTree (T ts) = putList (map putTree ts)
fun getTree a = T (map getTree (getList a))
```



Speicherplatzbedarf

```
rev nil = nil
rev (x::xr) = rev xr @ [x]
```

- Sei sn die Anzahl der Zellen die `rev` für eine Liste der Länge n alloziert. Dann:
  $s0 = 0$
  $sn = s(n-1) + 2 + 2(n-1) = s(n-1) + 2n$ für $n > 0$
  $\Rightarrow sn = n(n+1)$. Also $O(s) = O(n^2)$.
- Für Reversion einer Liste der Länge 10 werden 110 Zellen alloziert
- Nur 20 für Ergebnis erforderlich
- Rest räumt der Speicherbereiniger (garbage collector) weg.