Faltung

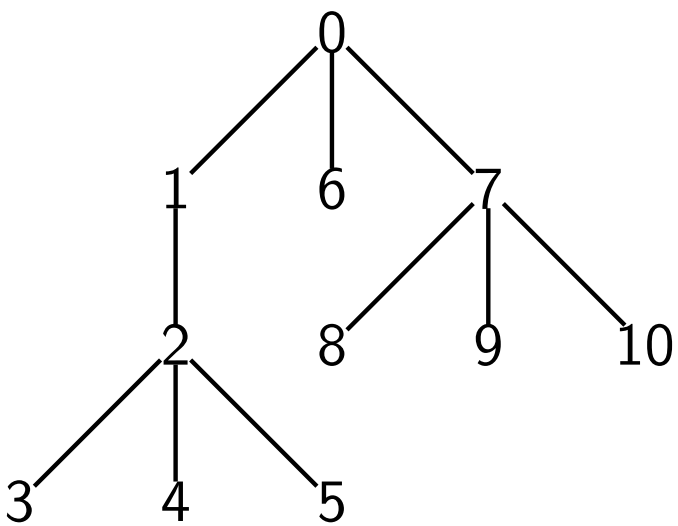
- ```
  fun fold f (T ts) = f (map (fold f) ts)
  fold:  ∀ a .  (a list -> a) -> tree -> a
  ```

- ```
  val size = fold (foldl op+ 1)
  val size:  tree -> int
  ```

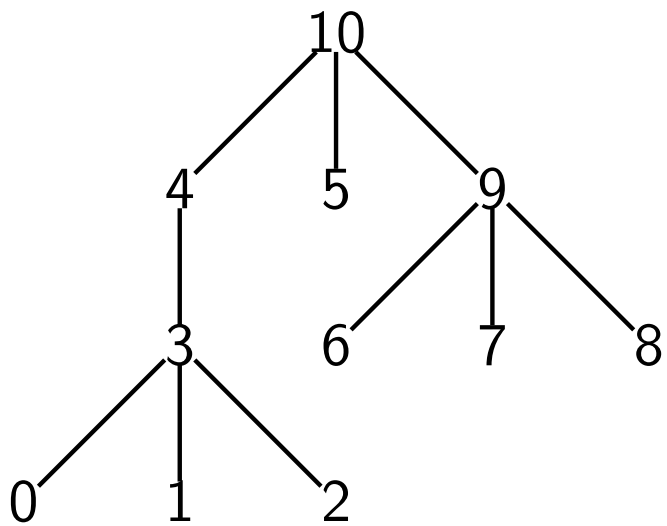
- ```
  val depth = fold (fn xs => 1 + foldl Int.max ~1 xs)
  val depth:  tree -> int
  ```
  ```

Präordnung und Postordnung

Pränummerierung
Nummeriere Knoten bei
erstem Besuch
im Laufe der Standardtour



Postnummerierung
Nummeriere Knoten bei
letztem Besuch
im Laufe der Standardtour



Linearisierungen

- **Prälinearisierung:**

```
T[] -> [0]
T[T[]] -> [1,0]
T[T[],T[]] -> [2,0,0]
T[T[], T[T[],T[]], T[T[]]] -> [3,0,2,0,0,1,0]
```

- **Postlinearisierung:**

```
T[] -> [0]
T[T[]] -> [0,1]
T[T[],T[]] -> [0,0,2]
T[T[], T[T[],T[]], T[T[]]] -> [0,0,0,2,0,1,3]
```

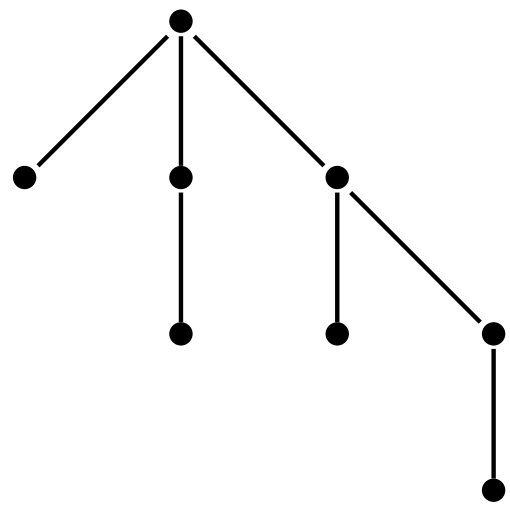
Ein Baum läßt sich eindeutig aus seiner Linearisierung rekonstruieren.

Finitäre Mengen

- Eine Menge heißt **rein**, wenn jedes Ihrer Elemente eine reine Menge ist.
- Eine Menge heißt **finitär**, wenn sie endlich ist und jedes Ihrer Elemente eine finitäre Menge ist.

- Jede finitäre Menge kann durch einen reinen Baum dargestellt werden.

$\{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}$



```
fun eqset x y = subset x y andalso subset y x
and subset (T xs) y = List.all (fn x => member x y) xs
and member x (T ys) = List.exists (eqset x) ys
```