1 Suchverfahren auf Listen

1.1 Die Lineare Suche

Zu implementieren ist der Suchalgorithmus auf einer unsortierten Liste mit n Elementen.

Algorithmische Idee: Das zu suchende Element wird von vorn beginnend jeweils mit dem Kopf der Liste verglichen und bei Ungleichheit rekursiv fortgesetzt mit der Restliste.

Listing 1: Sortieren durch Einfügen in Haskell

```
1  liste = [4,6,7,8,3,4,12,45,34,32,1,16,87]
2
3
4  linSearch :: Eq a => a -> [a] -> Bool
5  linSearch a [] = False
6  linSearch a (x:xs) | a == x = True
7                     | otherwise = linSearch a xs
```

Aufgaben: Analysiere den Quellcode! Probiere alle Varianten aus.
Erweitere das Programm so, dass die Anzahl des Auftretens des gesuchten Elements ermittelt wird.

1.2 Binäre Suche

Voraussetzung ist nun eine sortierte Liste. Zu implementieren ist die Binäre Suche.

Algorithmische Idee: Es wird jeweils das zu suchende Element s mit einem in der Mitte der Liste liegendem Element mid verglichen. Dabei sind drei Fälle möglich: Ist der Vergleich negativ, so wird rekursiv entweder mit der linken Teilliste weitergesucht (für den Fall $s < mid$) oder mit der rechten Teilliste (für den Fall $s > mid$).

Listing 2: Sortieren durch Auswählen in Haskell

```
1  liste = [4,6,7,8,3,4,12,45,34,32,1,16,87]
2  liste1 = [4,6,7,8,13,14,23,45,55,67,89]
3
4  binSearch:: Ord a => a -> [a] -> Bool
5  binSearch b [] = False
6  binSearch b x | (b == mid) = True
7                | (b < mid) = binSearch b left
8                | (b > mid) = binSearch b right
9                where
10               half = (length x) `div` 2
11               left = take half x
12               right = drop half x
13               mid = head right
```