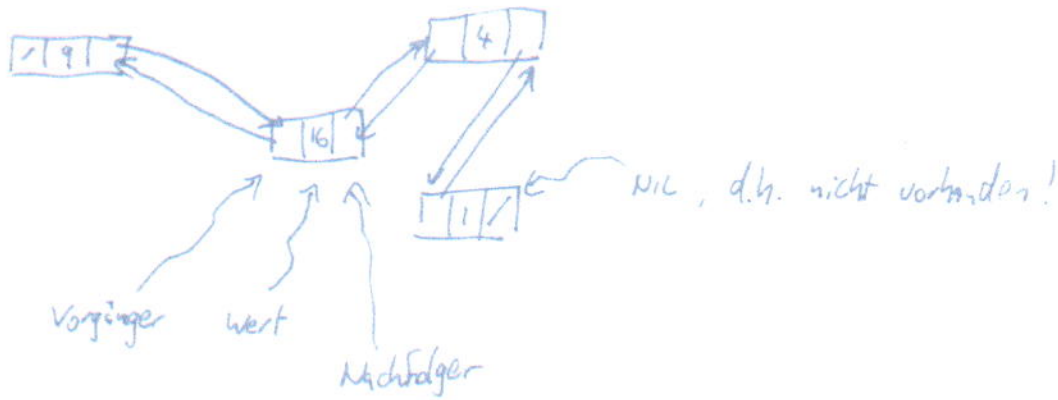
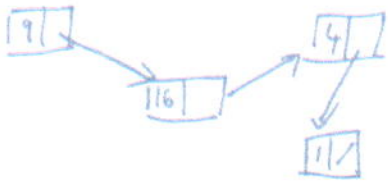
20.12.2011

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)

```
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 $O(1)$

4.4 Binäre Suche und binäre Bäume

Grundfrage: Wie kann man effizienter suchen?

Beispiel: Gesucht wird eine Zahl zwischen 1 und 100

Erlaubte Fragen: „Ist die Zahl z?"

Antworten: (0) Ja
(-) kleiner
(+) Größer

Variante 1: Fragefolge

1? +
2? +
3? +
⋮ ⋮

↪ 100 Fragen!

Variante 2: Fragefolge

50? -
25? +
37? +
43? -
40? +
41? +
42? 0 → 7 Fragen!

Also Idee: Nutze Sortierung der Zahlen aus, halbiere jeweils noch verbleibende Kandidatenmenge!

Frage 1: Wie lässt sich das Algorithmisch beschreiben?
Frage 2: Wie lässt sich das als Datenstruktur beschreiben?

Algorithmus 4.1 (Binäre Suche)

Eingabe: Sortierter Array mit Einträgen $S[i]$,
linke Randposition LINKS,
rechte Randposition RECHTS,
Suchwert WERT

Ausgabe: Position von WERT zwischen Arrayposition LINKS und RECHTS, falls existent

```
BINAERESUCHE (S, WERT, LINKS, RECHTS)
1  WHILE (LINKS ≤ RECHTS) DO
2      MITTE := ⌊(LINKS + RECHTS) / 2⌋      (Mittelwert, abgerundet)
3      IF (S[MITTE] = WERT) THEN
4          RETURN MITTE
5      IF (S[MITTE] > WERT) THEN
6          RECHTS := MITTE - 1
7      IF (S[MITTE] < WERT) THEN
8          LINKS := MITTE + 1
9  ENWHILE
10 RETURN "WERT nicht gefunden!"
```

Satz 4.2

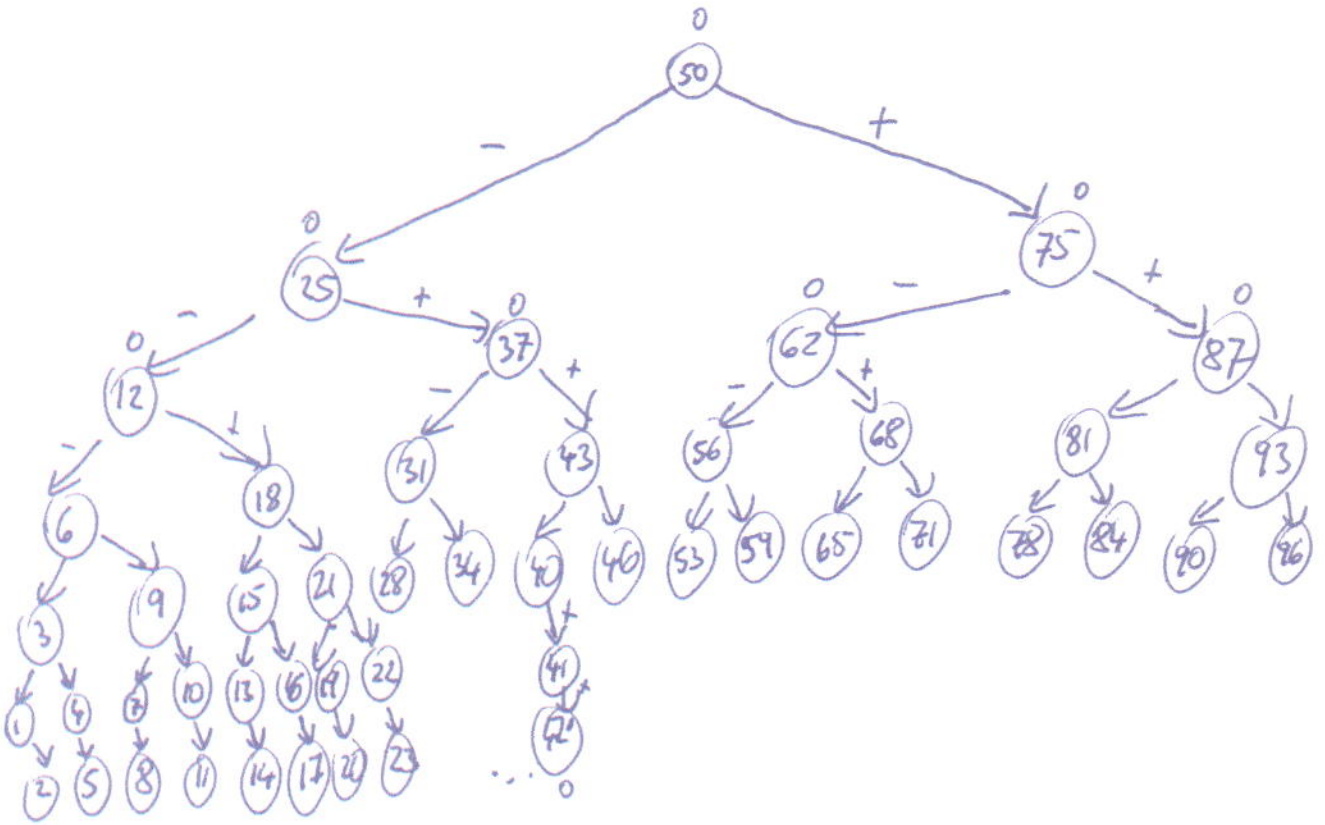
Die binäre Suche terminiert in $O(\log_2(\text{RECHTS} - \text{LINKS}))$ Schritten (für RECHTS > LINKS).

Beweis:

Selbst!

Hier interessiert uns die Datenstruktur!

Dafür stellen wir die möglichen Ereignisse in einem Ereignisbaum dar:



Beobachtungen:

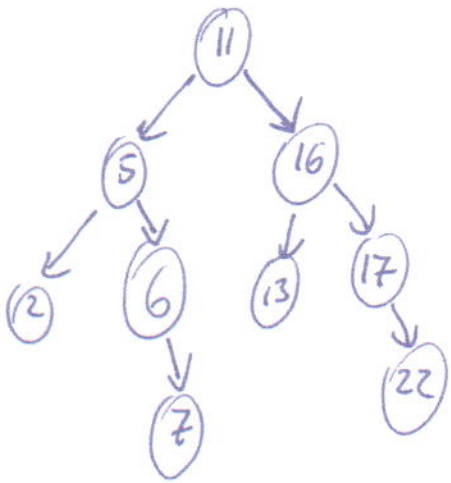
(i) Jeder Knoten im Baum entspricht einer Zahl

(ii) Jede Kante im Baum entspricht einem der möglichen Ereignisse „KLEINER" oder „GRÖSSER"

(iii) Jeder Knoten im Baum hat maximal zwei Nachfolgeknoten

(iv) Eine derartige Struktur taucht auch in anderen Fällen auf, z.B. für

$$S = (2, 5, 6, 7, 11, 13, 16, 17, 22):$$



Etwas sorgfältiger: