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  ENDWHILE
10 RETURN "WERT nicht gefunden!"
```

Satz 4.2

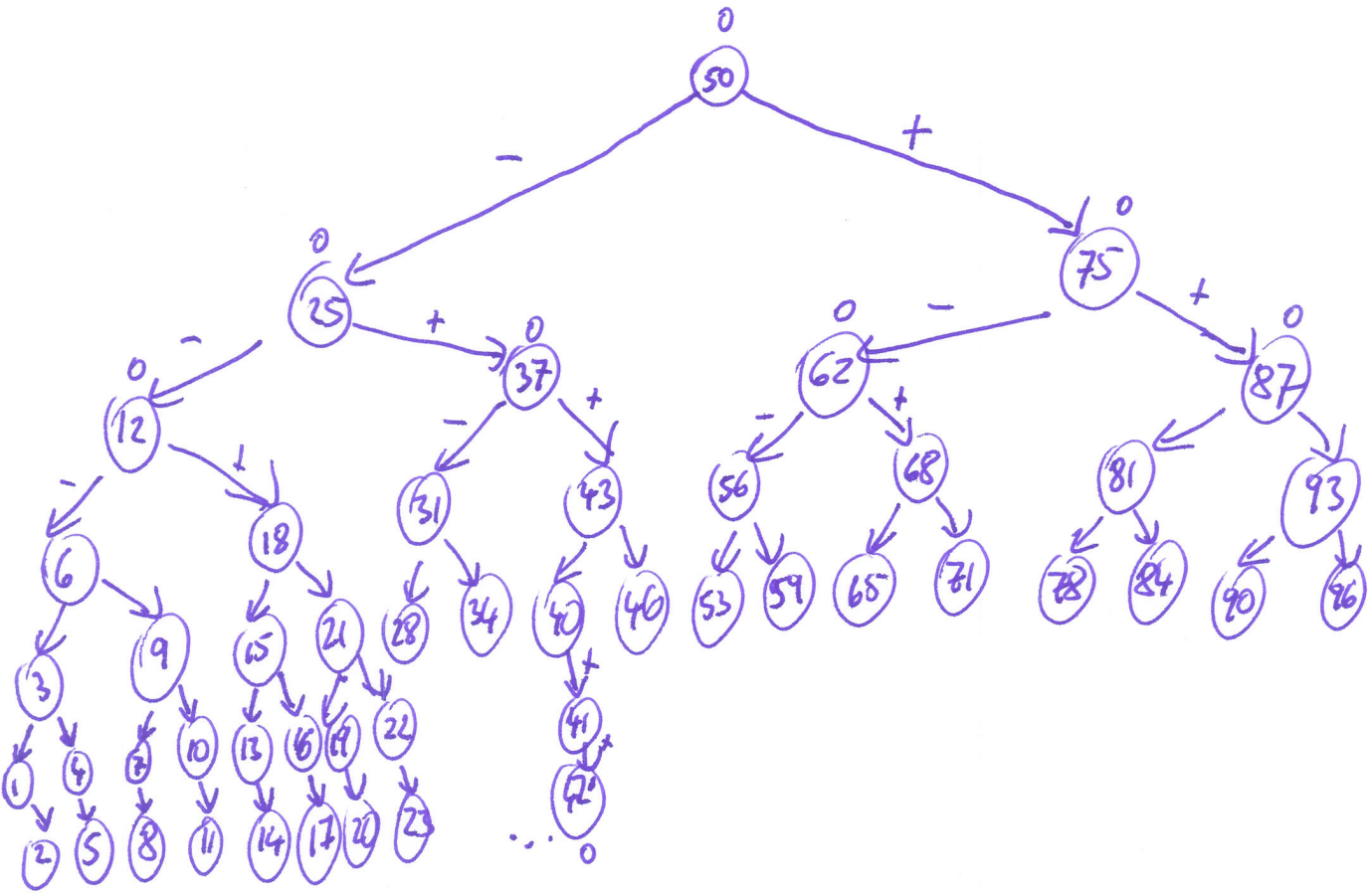
Die binäre Suche terminiert in $O(\log_2(\text{RECHTS} - \text{LINKS}))$ Schritten (für RECHTS > LINKS).

Beweis:

Selbst!

Hier interessiert uns die Datenstruktur!

Dafür stellen wir die möglichen Ereignisse in einem Ereignisbaum dar:



Beobachtungen:

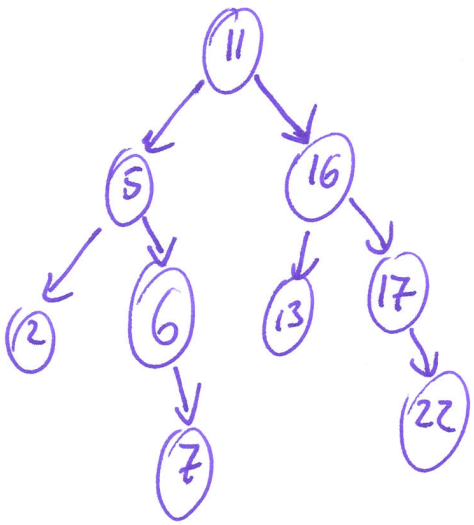
(i) Jeder Knoten im Baum entspricht einer Zahl

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

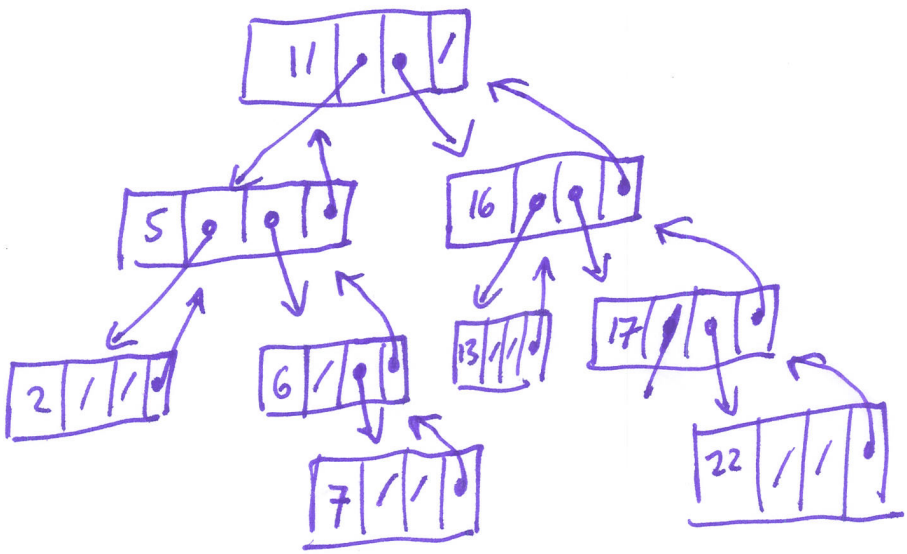
(iii) Jeder Knoten im Baum hat maximal zwei Nachfolgeknoten

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

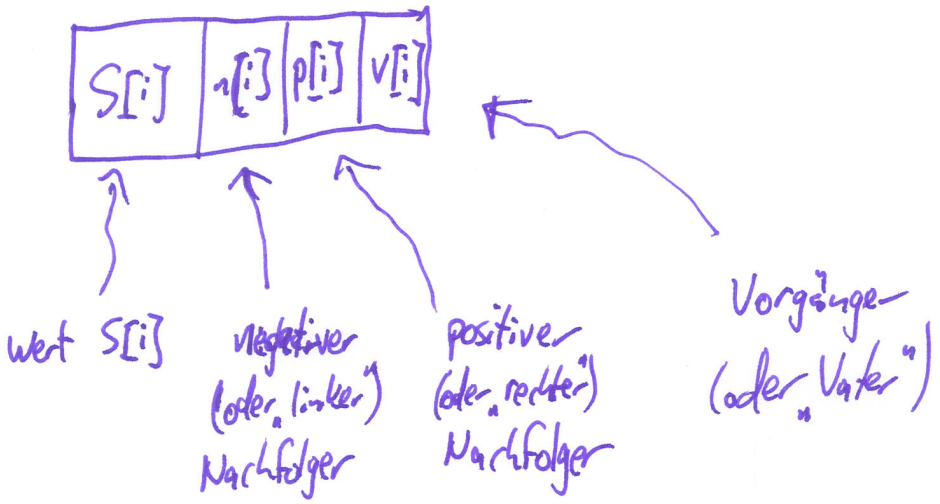
$$S = (2, 5, 6, 7, 11, 13, 16, 17, 22):$$



Etwas sorgfältiger:



Dabei sieht jedes Knotenobjekt so aus:



(Wie vorher schon steht „/" für „NIL", also kein Nachfolger oder Vorgänger.)

Definition 4.3

(1) Ein gerichteter Graph $D=(V,A)$ besteht aus einer endlichen Menge von Knoten V und einer endlichen Menge von gerichteten Kanten $a=(v,w)$. 

(v ist Vorgänger von w.)

(2) Ein gerichteter Baum $B=(V,T)$ hat folgende Eigenschaften:

(i) Es gibt einen eindeutigen Knoten $w \in V$ ohne Vorgänger (d.h. ohne pr. Knoten, die auf ihn zeigt).

(ii) Jeder Knoten $v \in V \setminus \{w\}$ ist durch einen eindeutigen Weg von w aus erreichbar; das heißt insbesondere, dass v einen eindeutigen Vorgänger („Vaterknoten") hat.

(3) Ein binärer Baum ist ein gerichteter Baum, in dem jeder Knoten höchstens zwei Nachfolger hatt. („Kindknoten") hat.

(3) Die Höhe h eines gerichteten Baumes ist die maximale Länge eines gerichteten Weges von der Wurzel.

(4) Ein binärer Baum ist ein gerichteter Baum, in dem jeder Knoten höchstens zwei Nachfolger („Kindknoten") hat. Wir nennen einen davon den linken l(i), den anderen den rechten r(i).

(5) Ein binärer Baum heißt voll, wenn jeder Knoten zwei oder keinen Kindknoten hat.

(6) Ein Knoten ohne Kindknoten heißt Blatt.

(7) Der Teilbaum eines Knotens i ist durch die Menge der von i erreichbaren Knoten und der dabei verwendeten Kanten definiert. Der linke Teilbaum ist der Teilbaum von l[i].

(8) In einem binären Suchbaum hat jeder Knoten i einen Schlüsselwert S[i], und es gilt:
- Wenn j ein Knoten im linken Teilbaum ist, gilt $S[j] \leq S[i]$.
- Wenn j ein Knoten im rechten Teilbaum ist, gilt $S[i] \leq S[j]$.

Binäre Suchbäume als Datenstruktur?!
- Suchen
- Minimum/Maximum bestimmen
- Nachfolger/Vorgänger bestimmen
- Einfügen
- Löschen

Iterative Baumsuche (i, k)

```
1  WHILE (i ≠ NIL und k ≠ S[i]) DO
2      IF k < S[i]
3          THEN i := l[i]
4          ELSE i := r[i]
5  RETURN i
```