Vorlesung 5 (Donnerstag 22.2.2018)

5.2 Binäre Suchbäume

Suchoperation für Listen: $O(N)$

Besser: binäre Suchbäume:

binäre Bäume:
Jedes Element (genannt Knoten) hat bis zu zwei Nachfolger.
Element ohne Vorgänger = Wurzel des Baumes
Jeder Knoten = Wurzel des Unterbaums, der Knoten + alle seine direkten + indirekten Nachfolger enthält.

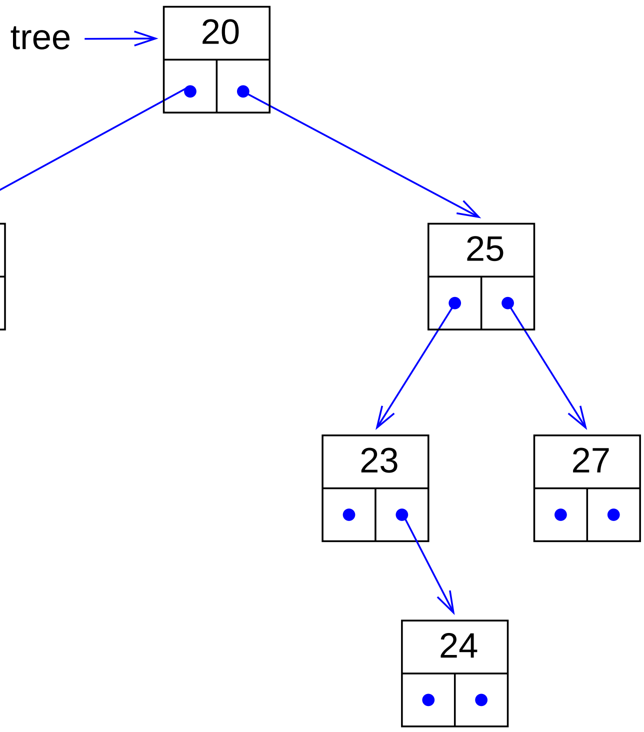


Figure 9: Binärer Suchbaum.

Suchbaum:
Linker Unterbaum: Elemente sind "kleiner" als an Wurzel

Rechter Unterbaum: Elemente sind "größer" als an Wurzel
gilt auch für jeden Unterbaum
→ Suche funktioniert in $O(\log N)$ (typischerweise).
Baum: repräsentiert durch Zeiger auf Wurzel
Knoten ohne Nachfolger = Blatt

Grundlegende Datenstruktur

```
/* data structures for tree elements */
struct node_struct
{
    int                 info;                 /* holds ``information'' */
    struct node_struct  *left;  /* points to left subtree (NULL if none) */
    struct node_struct *right; /* points to right subtree (NULL if none) */
};

typedef struct node_struct node_t;          /* define new type for nodes */
```

Erzeugen und Löschen von einzelnen Knoten: analog zu Listen.

[Selbsttest]

Wie könnte ein Algorithmus zum einsetzen eines Elements aussehen. Überlegen Sie erst 3 Minuten selber, dann diskutieren Sie mit Ihrem Nachbarn.

ACHTUNG: Lesen Sie den Rest des Abschnitts NICHT, bevor Sie sich etwas überlegt haben

---

Einsetzen eines Knotens:
Such nach Wert
Falls gefunden: Ende
Falls nicht: Setze Werte als Blatt dort ein, wo Suche endete

```
/******************** insert_node() *****************/
/** Inserts 'node' into the 'tree' such that the    **/
/** increasing order is preserved                   **/
/** if node exists already, nothing happens         **/
/** PARAMETERS: (*)= return-paramter                **/
/**         tree: pointer to root of tree           **/
/**         node: pointer to node                   **/
/** RETURNS:                                        **/
/**   (new) pointer to root of tree                 **/
```

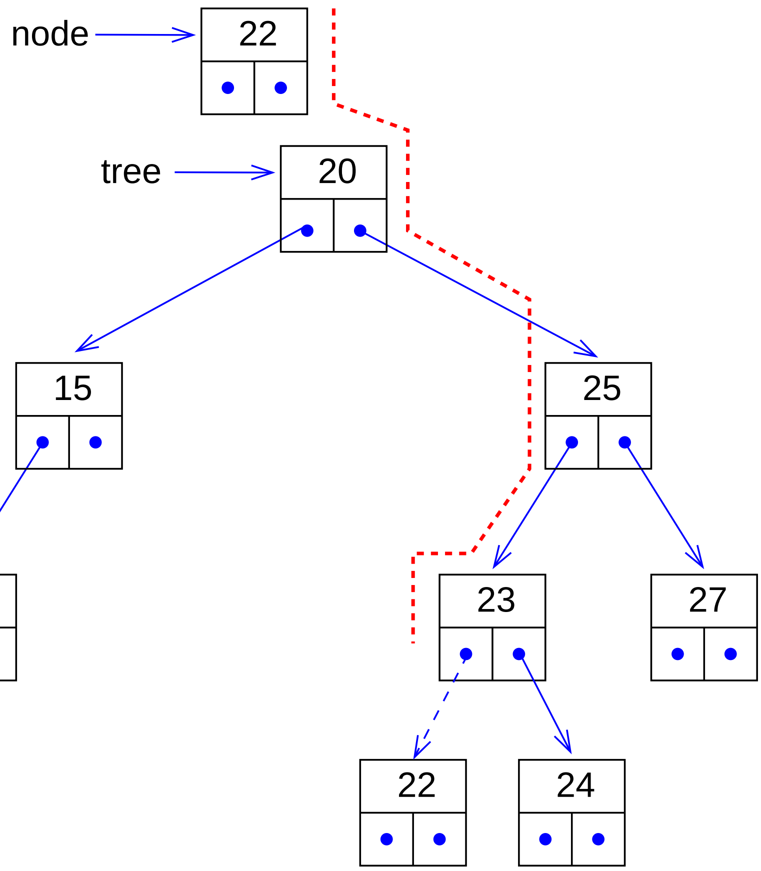


Figure 10: Einsetzen eines neuen Elementes in den Suchbaum

```
/*******************************************************/
node_t *insert_node(node_t *tree, node_t *node)
{
  node_t *current;

  if(tree==NULL)
    return(node);
  current = tree;
  while( current != NULL)                  /* run through tree */
  {
    if(current->info==node->info)  /* node already contained ? */
      return(tree);
    if( node->info < current->info)            /* left subtree */
    {
      if(current->left == NULL)
      {
```

```
        current->left = node;                       /* add node */
        return(tree);
      }
      else
        current = current->left;          /* continue searching */
    }
    else                                        /* right subtree */
    {
      if(current->right == NULL)
      {
        current->right = node;                      /* add node */
        return(tree);
      }
      else
        current = current->right;          /* continue searching */
    }
  }
}
```

[Selbsttest]

Wie kann man einen Baum ausgeben? Überlegen Sie drei Minuten und diskutieren dann drei Minuten zu zweit/dritt.

ACHTUNG: Lesen Sie den Rest des Abschnitts NICHT, bevor Sie sich etwas überlegt haben

---

Geordnete Ausgabe des Baums: rekurisv

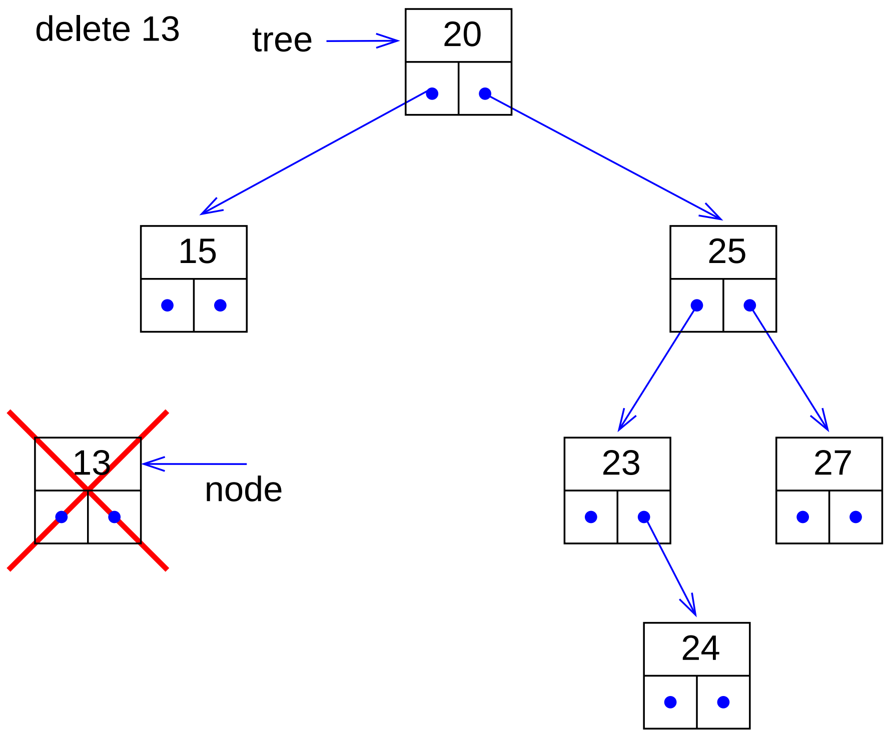
```
/********************* print_tree() ******************/
/** Prints tree in ascending order recursively.      **/
/** PARAMETERS: (*)= return-paramter                 **/
/**         tree: pointer to root of tree            **/
/** RETURNS:                                         **/
/**   nothing                                        **/
/*****************************************************/
void print_tree(node_t *tree)
{
  if(tree != NULL)
  {
    print_tree(tree->left);
    printf("%d ", tree->info);
    print_tree(tree->right);
```

```
  }
}
```

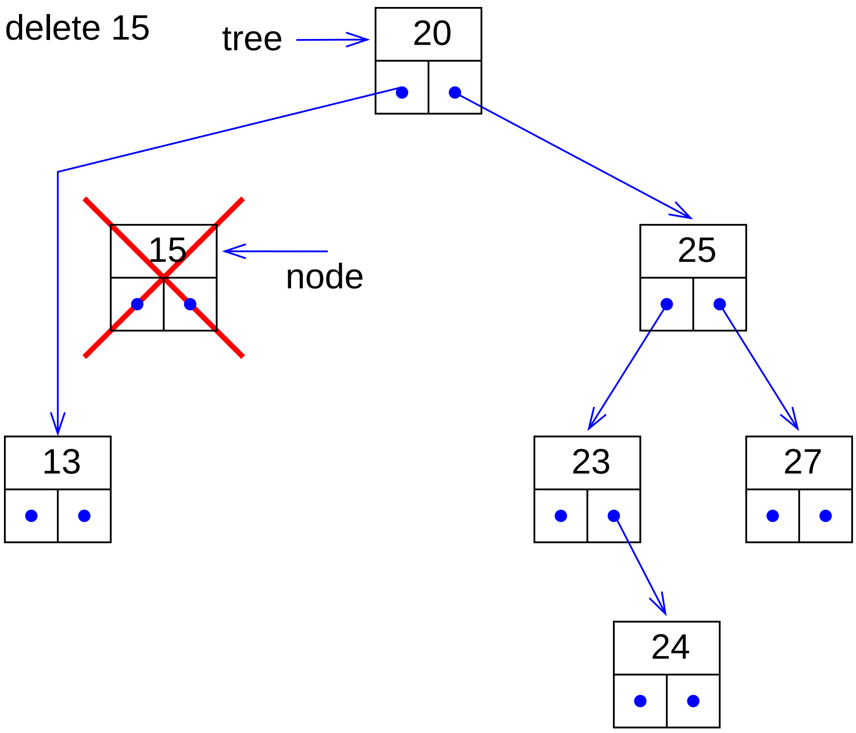
Hinweis: Auch preorder (Erst Knoten, dann linken, dann rechten Unterbaum ausgeben) und postorder möglich.

Entfernen eines Wertes x. 3 Fälle:

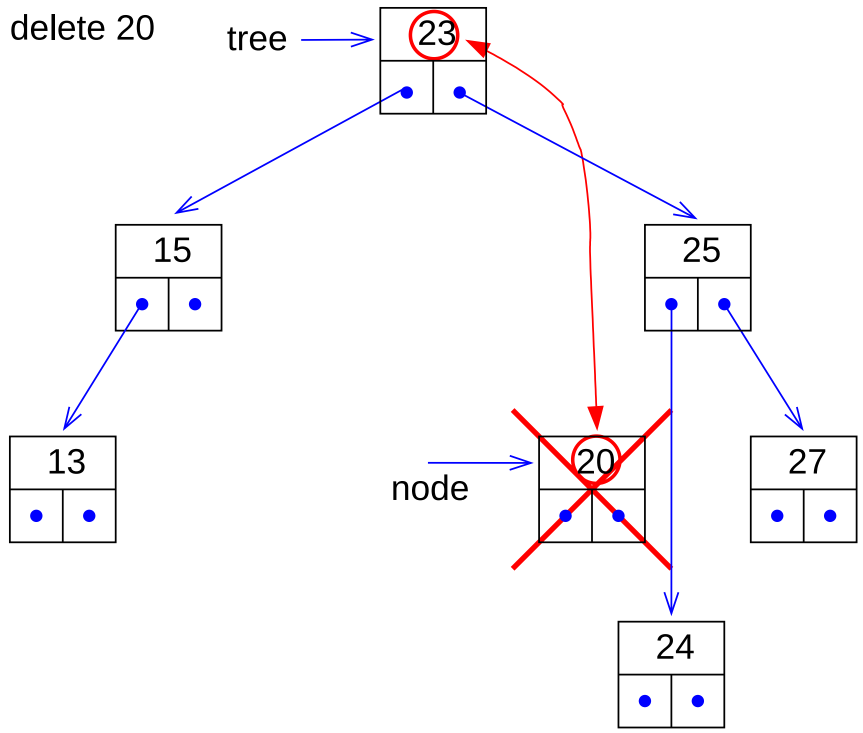
- Wert ist in Blatt (keine Nachfolger): Einfach entfernen.



- Knoten mit x hat einen Nachfolger: Knoten durch Nachfolger ersetzen



- Knoten mit x hat zwei Nachfolger:
  Such nach Knoten n_2, der den kleinsten Wert y im rechten Unterbaum hat (d.h. n_2 hat keinen linken Nachfolger).
  Vertausche Werte x und y. Entferne n_2 wie in Fällen eins oder zwei.



Hinweis: Binäre Bäume können <u>unbalanciert</u> (oder <u>entartet</u>) sein
Bsp: Iterierte Eingabe zu `insert_node` ist geordnet → Baum wird zur Liste, also Suche dauert $O(N)$ (worst case).
Lösung: <u>Balancierte Bäume</u> (z.B. "rot-schwarz Bäume"): Falls Baum unbalanciert ist, wird er ausgeglichen (z.b. durch "Rotationen") → Alle Operationen dauern nur $O(\log N)$.

6 Datenanalyse und Zufallszahlen

Statistische Datenanalyse: für deterministische Simulation (Molekulardynamik, DGLs) und für stochastische Simulationen. Für letzteres:
Anwendung von Zufallszahlen bei Computersimulationen:

- Systeme mit zufälligen Wechselwirkungen ("Spingläser")
- Simulationen bei endlichen Temperaturen mit Monte-Carlo Verfahren
- Randomisierte Algorithmen (aus deterministischen Algorithmen entstanden)

Zufallszahlen im Computer möglich (z.B. Schwankungen der Spannungen an einem Transistor durch thermisches Rauschen). Vorteil: Zufällig. Nachteil:

Statistische Eigenschaften unbekannt und nicht kontrollierbar.
Daher: Pseudozufallszahlen = nicht zufällig, aber *möglichst* gleiche statische Eigenschaften (Verteilung, Korrelationen).

6.1 Zufallsvariablen

Zufallsvariable (ZV) X (unscharf): Zufallsexperiment mit $\Omega = \mathbb{R}$

Definition: Verteilungsfunktion (VF) einer ZV X ist eine Funktion $F_X : \mathbb{R} \longrightarrow [0, 1]$ definiert über

$$F_X(x) = P(X \leq x) \tag{7}$$

Bsp: Münze:

$$F(x) = \begin{cases} 0 & x < 0 \\ 0.5 & 0 \leq x < 1 \\ 1 & x \geq 1 \end{cases} . \tag{8}$$

x-Position eines Gasteilchens im Container $[0, L_x]$

$$F(x) = \begin{cases} 0 & x < 0 \\ x/L_x & 0 \leq x < L_x \\ 1 & x \geq L_x \end{cases} . \tag{9}$$

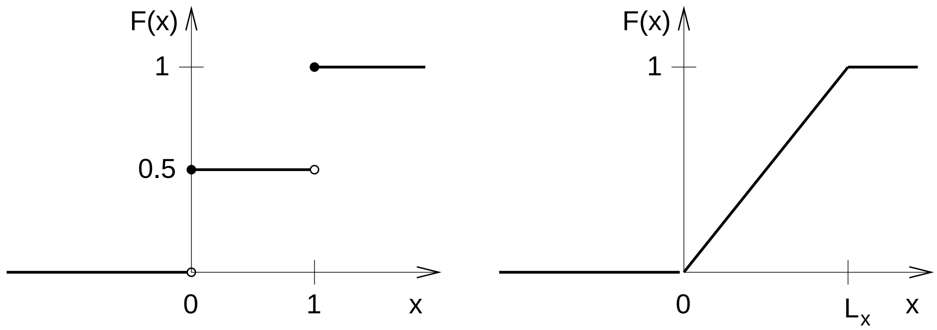


Figure 11: Verteilungsfunktionen für Münze und Gasteilchen.

Nach Zufallsexperiment gemäß X mit Ergebnis x, dann $y = g(x)$ berechnen: Transformation der ZV zu $Y = g(X)$, allgemein:

$$Y = \tilde{g}\left(X^{(1)}, X^{(2)}, \ldots, X^{(k)}\right) . \tag{10}$$

Verteilungsfunktion manchmal unhandlich → Wahrscheinlichkeits-/ Dichtefunktion:

6.1.1 Diskrete Zufallsvariablen

Definition: Die Wahrscheinlichkeitsfunktion (engl. probability mass function, pmf) $p_X : \mathbb{R} \to [0,1]$ ist gegeben durch

$$p_X(x) = P(X = x)\,. \tag{11}$$

Verteilung für n fachen Münzwurf (0/1):
Definition: Die *binomial Verteilung* mit Parametern $n \in \mathbb{N}$ und p ($0 < p \leq 1$) beschreibt eine ZV X mit WF

$$p_X(x) = \begin{cases} \binom{n}{x} p^x (1-p)^{n-x} & (0 \leq x \leq n, x \in \mathbb{N}) \\ 0 & \text{sonst} \end{cases} \tag{12}$$

Notation $X \sim B(n,p)$.
Charakterisierung von ZVen:

$\{\tilde{x}_i\}$ Menge der Werte für die $p_X(\tilde{x}) > 0$.
Definition:

- Erwartungswert
$$\mu \equiv \mathrm{E}[X] = \sum_i \tilde{x}_i P(X = \tilde{x}_i) = \sum_i \tilde{x}_i p_X(\tilde{x}_i) \tag{13}$$

- Varianz
$$\sigma^2 \equiv \mathrm{Var}[X] = \mathrm{E}[(X - \mathrm{E}[X])^2] = \sum_i (\tilde{x}_i - \mathrm{E}[X])^2 p_X(\tilde{x}_i) \tag{14}$$

- Standardabweichung
$$\sigma \equiv \sqrt{\mathrm{Var}[X]} \tag{15}$$

Eigenschaften:

$$\begin{aligned}
\mathrm{E}[\alpha_1 X^{(1)} + \alpha_2 X^{(2)}] &= \alpha_1 \mathrm{E}[X^{(1)}] + \alpha_2 \mathrm{E}[X^{(2)}] && (16) \\
\sigma^2 = \mathrm{Var}[X] &= \mathrm{E}[X^2] - \mathrm{E}[X]^2 && (17) \\
\Leftrightarrow \quad \mathrm{E}[X^2] &= \sigma^2 + \mu^2 && (18) \\
\mathrm{Var}[\alpha_1 X^{(1)} + \alpha_2 X^{(2)}] &= \alpha_1^2 \mathrm{Var}[X^{(1)}] + \alpha_2^2 \mathrm{Var}[X^{(2)}] && (19)
\end{aligned}$$

$\mathrm{E}[X^n]$: n-tes Moment
Es gilt für die Binomialverteilung:

$$\begin{aligned}
\mathrm{E}[X] &= np && (20) \\
\mathrm{Var}[X] &= np(1-p) && (21)
\end{aligned}$$