Mathematische Grundlagen der Informatik

Graphentheorie

W. Gansterer, K. Schindlerova

28. Mai 2020

Graphentheorie

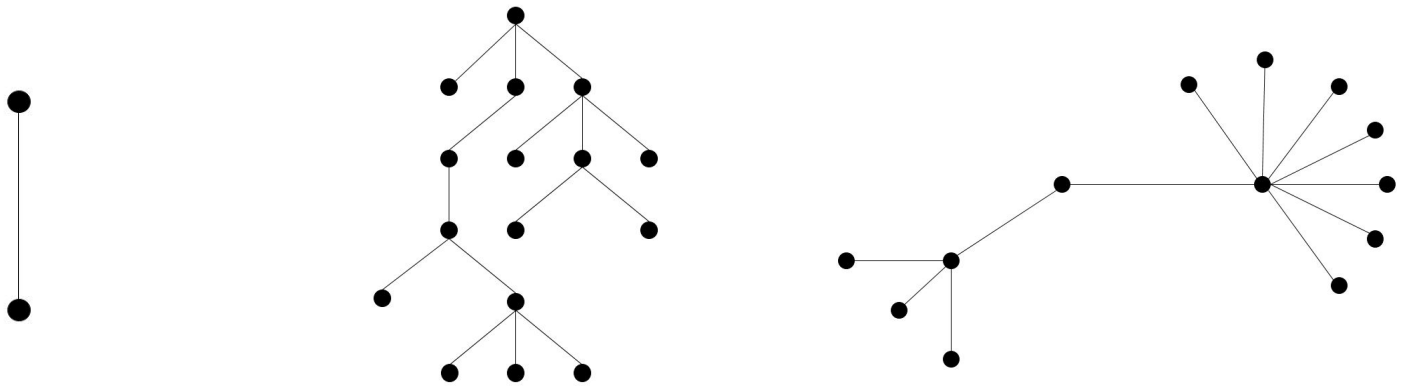
Inhaltsverzeichnis I

Bäume

WH: Bäume

Definition

Ein Graph, in dem je zwei Knoten durch genau einen Weg verbunden sind, heißt **Baum**. Ein Baum ist also ein zusammenhängender Graph ohne Kreise.



WH: Charakterisierung Baum

Für einen Graphen G mit n Knoten und m Kanten sind die folgenden Aussagen äquivalent:

1. G ist ein Baum (je zwei Knoten durch genau einen Weg verbunden)
2. G ist ein zusammenhängender Graph ohne Kreise
3. G ist zusammenhängend, aber entfernt man eine beliebige Kante, so zerfällt G in zwei Zusammenhangskomponenten
4. G ist zusammenhängend und $n = m + 1$
   (G hat einen Knoten mehr als Kanten)

Charakterisierung Baum

Beweis.

a) $\Rightarrow$ b), b) $\Rightarrow$ c), c) $\Rightarrow$ a) $\quad \rightarrow \quad$ siehe letzte VO!

Charakterisierung Baum

Beweis.

- Skizze für d) $\Leftrightarrow$ a), b), c):
  - Vorbereitung: Nach Wegnahme einer Kante aus einem Baum bilden die entstehenden Zusammenhangskomponenten wieder Bäume: Sind z. B. y und z Knoten in einer solchen Komponente, so kann es nicht mehrere verschiedene Wege von y nach z geben, da diese auch im ursprünglichen Graphen G Wege wären.

Charakterisierung Baum

Beweis.

- Skizze für d) $\Leftrightarrow$ a), b), c):
  - Vorbereitung: Nach Wegnahme einer Kante aus einem Baum bilden die entstehenden Zusammenhangskomponenten wieder Bäume: Sind z. B. y und z Knoten in einer solchen Komponente, so kann es nicht mehrere verschiedene Wege von y nach z geben, da diese auch im ursprünglichen Graphen G Wege wären.
  - Zeige a) $\Rightarrow$ d) mit vollständiger Induktion nach der Anzahl der Knoten n.

Charakterisierung Baum

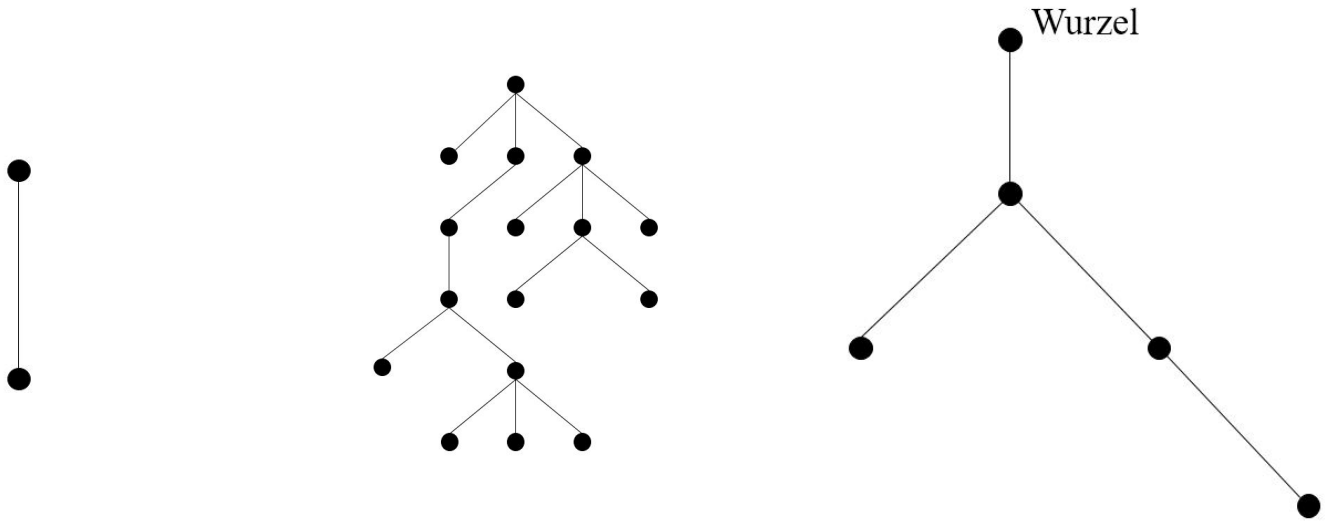
Beweis.

- Skizze für d) $\Leftrightarrow$ a), b), c):
  - Vorbereitung: Nach Wegnahme einer Kante aus einem Baum bilden die entstehenden Zusammenhangskomponenten wieder Bäume: Sind z. B. y und z Knoten in einer solchen Komponente, so kann es nicht mehrere verschiedene Wege von y nach z geben, da diese auch im ursprünglichen Graphen G Wege wären.
  - Zeige a) $\Rightarrow$ d) mit vollständiger Induktion nach der Anzahl der Knoten n.
  - Zeige d) $\Rightarrow$ a) mit vollständiger Induktion nach der Anzahl der Knoten n (unter Beachtung von $\sum_{x \in V} d(x) = 2 \cdot |E|$).

Wurzelbaum

Ein Wurzelbaum ist ein Baum, in dem ein Knoten ausgezeichnet wird: die *Wurzel*. Jeder Knoten eines Baumes kann Wurzel werden.

Die Baumwurzel wird i.d.R. oben gezeichnet, alle Kanten weisen nach unten.



Wurzelbaum – Definitionen

- Sind x und y duch eine Kante verbunden und x ist näher zur Wurzel als y, so ist x **Vater** von y und y **Sohn** von x.

Wurzelbaum – Definitionen

- Sind x und y duch eine Kante verbunden und x ist näher zur Wurzel als y, so ist x **Vater** von y und y **Sohn** von x.
- Existiert ein Weg von y zur Wurzel, welcher x enthält, so heißt x **Vorfahre** von y und y **Nachkomme** von x.
  - Der einzige Knoten ohne Vorfahren ist die Wurzel!

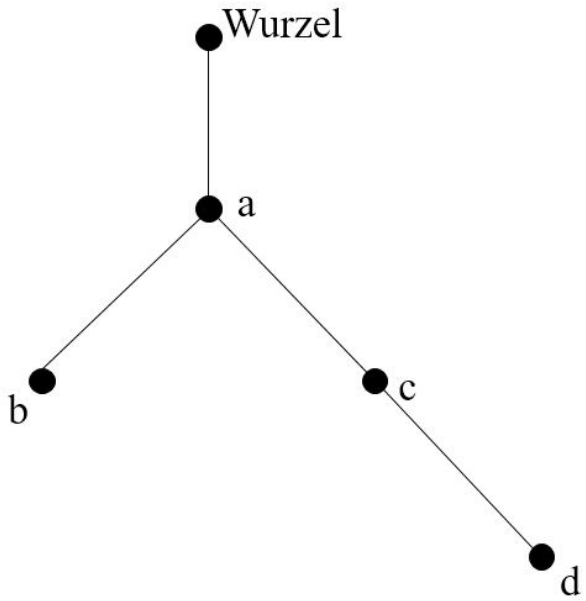
Wurzelbaum – Definitionen

- Sind x und y duch eine Kante verbunden und x ist näher zur Wurzel als y, so ist x **Vater** von y und y **Sohn** von x.
- Existiert ein Weg von y zur Wurzel, welcher x enthält, so heißt x **Vorfahre** von y und y **Nachkomme** von x.
  - Der einzige Knoten ohne Vorfahren ist die Wurzel!
- Knoten mit Nachkommen heißen **innere Knoten**.
- Knoten ohne Nachkommen heißen **Blätter**
  (= die von der Wurzel verschiedenen Knoten mit Grad 1).

Wurzelbaum – Definitionen

- Sind x und y duch eine Kante verbunden und x ist näher zur Wurzel als y, so ist x **Vater** von y und y **Sohn** von x.
- Existiert ein Weg von y zur Wurzel, welcher x enthält, so heißt x **Vorfahre** von y und y **Nachkomme** von x.
  - Der einzige Knoten ohne Vorfahren ist die Wurzel!
- Knoten mit Nachkommen heißen **innere Knoten**.
- Knoten ohne Nachkommen heißen **Blätter** (= die von der Wurzel verschiedenen Knoten mit Grad 1).
- Ist die *Reihenfolge* bei den Söhnen von Bedeutung, so spricht man von einem **geordneten Wurzelbaum**. Die Söhne werden (meist von links nach rechts) durchnummeriert: erster Sohn, zweiter Sohn, usw.

Beispiel: Wurzelbaum



- a ist Vater von b
- d ist Nachkomme von a
- d ist kein Nachkomme von b
- b und d sind Blätter
- a und c sind innere Knoten

Wurzelbaum

Beachten Sie:

In einem Wurzelbaum bildet jeder Knoten mit allen seinen Nachkommen und den dazugehörigen Kanten wieder einen Wurzelbaum.

$\Rightarrow$ (Wurzel)Bäume eignen sich hervorragend für rekursive Bearbeitung!

- Teilbäume haben die gleichen Eigenschaften wie der ursprüngliche Baum, werden aber immer kleiner
- Viele Baumalgorithmen sind daher rekursiv aufgebaut!

Anwendung: Compiler

Syntaxanalyse: Der Compiler erzeugt aus dem zu verarbeitenden Programmcode einen *Syntaxbaum* (= geordneter Wurzelbaum):

- Die inneren Knoten stellen die Operatoren dar;
- die Operanden werden aus den Teilbäumen gebildet, die an den Söhnen dieses Operators hängen.

Anwendung: Compiler

Syntaxanalyse: Der Compiler erzeugt aus dem zu verarbeitenden Programmcode einen *Syntaxbaum* (= geordneter Wurzelbaum):

- Die inneren Knoten stellen die Operatoren dar;
- die Operanden werden aus den Teilbäumen gebildet, die an den Söhnen dieses Operators hängen.

Beispiel: Syntaxbaum für

```
(x%4 == 0) &&
!( (x%100 == 0) &&
   (x%400 != 0) )
```

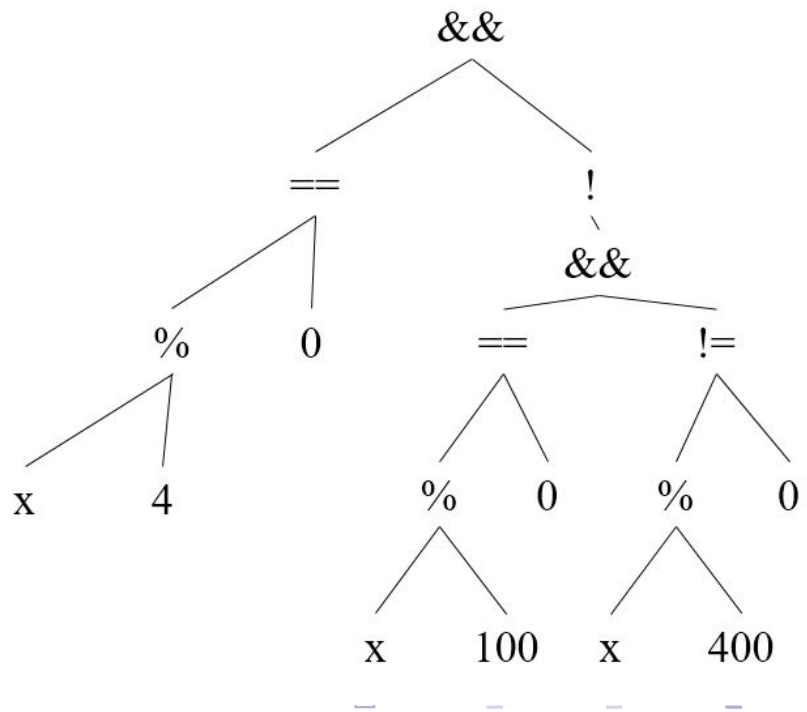
Anwendung: Compiler

Syntaxanalyse: Der Compiler erzeugt aus dem zu verarbeitenden Programmcode einen *Syntaxbaum* (= geordneter Wurzelbaum):

- Die inneren Knoten stellen die Operatoren dar;
- die Operanden werden aus den Teilbäumen gebildet, die an den Söhnen dieses Operators hängen.

Beispiel: Syntaxbaum für

$(x\%4 == 0)$ &&
!($(x\%100 == 0)$ &&
$(x\%400 != 0)$)



Anwendung: Compiler

Nach dem Aufbau des Syntaxbaums kann der Compiler den Ausdruck *rekursiv* auswerten:

- Um den Baum auszuwerten, müssen alle Teilbäume in den Söhnen der Wurzel ausgewertet werden.
- Anschließend kann die Operation der Wurzel durchgeführt werden.
- Rekursion endet, wenn die Wurzel selbst ein Operand ist.

Wurzelbäume

- Das **Niveau** eines Knotens in einem Wurzelbaum ist die Anzahl der Knoten des Weges von der Wurzel zu diesem Knoten
  - Die Wurzel selbst hat das Niveau 1.

Wurzelbäume

- Das **Niveau** eines Knotens in einem Wurzelbaum ist die Anzahl der Knoten des Weges von der Wurzel zu diesem Knoten
  - Die Wurzel selbst hat das Niveau 1.
- Das maximale Niveau aller Knoten heißt **Höhe** des Wurzelbaumes.

Wurzelbäume

- Das **Niveau** eines Knotens in einem Wurzelbaum ist die Anzahl der Knoten des Weges von der Wurzel zu diesem Knoten
  - Die Wurzel selbst hat das Niveau 1.
- Das maximale Niveau aller Knoten heißt **Höhe** des Wurzelbaumes.
- Hat in einem Wurzelbaum jeder Knoten höchstens n Söhne, so heißt er **n-ärer Wurzelbaum**.
- Hat jeder Knoten *genau* n oder 0 Söhne, so heißt er **regulärer n-ärer Wurzelbaum**.
  - Spezialfall $n = 2 \rightarrow$ binäre (reguläre) Wurzelbäume, die Söhne werden *linke* und *rechte* Söhne genannt.

Eigenschaften binärer regulärer Wurzelbäume

Jeder *binäre reguläre Wurzelbaum B* hat folgende Eigenschaften:

1. Falls B mehr als einen Knoten hat, dann gibt es genau einen Knoten vom Grad 2, und alle anderen Knoten haben Grad 1 oder 3.

Eigenschaften binärer regulärer Wurzelbäume

Jeder *binäre reguläre Wurzelbaum* B hat folgende Eigenschaften:

1. Falls B mehr als einen Knoten hat, dann gibt es genau einen Knoten vom Grad 2, und alle anderen Knoten haben Grad 1 oder 3.
2. Ist x ein Knoten von B, so bildet x mit all seinen Nachkommen wieder einen binären regulären Wurzelbaum.

Eigenschaften binärer regulärer Wurzelbäume

Jeder *binäre reguläre Wurzelbaum* B hat folgende Eigenschaften:

1. Falls B mehr als einen Knoten hat, dann gibt es genau einen Knoten vom Grad 2, und alle anderen Knoten haben Grad 1 oder 3.
2. Ist x ein Knoten von B, so bildet x mit all seinen Nachkommen wieder einen binären regulären Wurzelbaum.
3. Die Anzahl der Knoten ist ungerade.

Eigenschaften binärer regulärer Wurzelbäume

Jeder *binäre reguläre Wurzelbaum B* hat folgende Eigenschaften:

1. Falls B mehr als einen Knoten hat, dann gibt es genau einen Knoten vom Grad 2, und alle anderen Knoten haben Grad 1 oder 3.
2. Ist x ein Knoten von B, so bildet x mit all seinen Nachkommen wieder einen binären regulären Wurzelbaum.
3. Die Anzahl der Knoten ist ungerade.
4. Hat B n Knoten, so hat er $(n+1)/2$ Blätter und $(n–1)/2$ innere Knoten. Es gibt also genau ein Blatt mehr als innere Knoten.

Eigenschaften binärer regulärer Wurzelbäume

Jeder *binäre reguläre Wurzelbaum B* hat folgende Eigenschaften:

1. Falls B mehr als einen Knoten hat, dann gibt es genau einen Knoten vom Grad 2, und alle anderen Knoten haben Grad 1 oder 3.
2. Ist x ein Knoten von B, so bildet x mit all seinen Nachkommen wieder einen binären regulären Wurzelbaum.
3. Die Anzahl der Knoten ist ungerade.
4. Hat B n Knoten, so hat er $(n+1)/2$ Blätter und $(n–1)/2$ innere Knoten. Es gibt also genau ein Blatt mehr als innere Knoten.
5. In einem binären Wurzelbaum der Höhe H mit n Knoten gilt:

$$H \geq log_2(n+1)$$

Eigenschaften binärer regulärer Wurzelbäume

- Beweisen wir Aussage 1): *Falls B mehr als einen Knoten hat, dann gibt es genau einen Knoten vom Grad 2, und alle anderen Knoten haben Grad 1 oder 3.*
  - Die Wurzel hat genau zwei Söhne, also Grad 2, jeder andere Knoten ist ein Sohn und hat entweder zwei Söhne (Grad 3) oder keine Nachkommen (Grad 1).

Eigenschaften binärer regulärer Wurzelbäume

- Beweisen wir Aussage 1): *Falls B mehr als einen Knoten hat, dann gibt es genau einen Knoten vom Grad 2, und alle anderen Knoten haben Grad 1 oder 3.*
  - Die Wurzel hat genau zwei Söhne, also Grad 2, jeder andere Knoten ist ein Sohn und hat entweder zwei Söhne (Grad 3) oder keine Nachkommen (Grad 1).
- Beweisen wir Aussage 2): *Ist x ein Knoten von B, so bildet x mit all seinen Nachkommen wieder einen binären regulären Wurzelbaum.*
  - Die charakterisierenden Eigenschaften des binären regulären Baumes bleiben natürlich in allen Teilbäumen erhalten.

Eigenschaften binärer regulärer Wurzelbäume

Zum Beweis von Aussage 3): *Die Anzahl der Knoten ist ungerade.*

Eigenschaften binärer regulärer Wurzelbäume

Zum Beweis von Aussage 3): *Die Anzahl der Knoten ist ungerade.*

- Mittels Induktion über die Anzahl der Knoten!
- Ein minimaler binärer regulärer Wurzelbaum ist ein einzelner Knoten, für diesen stimmt die Aussage.
- Die Anzahl der Knoten des Baumes B ist 1 + Anzahl der Knoten des linken Teilbaums + Anzahl der Knoten des rechten Teilbaums $\rightarrow$ also ungerade, da die beiden kleineren Teilbäume *laut Induktionsannahme* ungerade Knotenzahl haben.

Eigenschaften binärer regulärer Wurzelbäume

Zum Beweis von Aussage 4): *Hat B n Knoten, so hat er $(n+1)/2$ Blätter und $(n–1)/2$ innere Knoten. Es gibt also genau ein Blatt mehr als innere Knoten.*

- Sei p die Anzahl der Blätter von B.

Eigenschaften binärer regulärer Wurzelbäume

Zum Beweis von Aussage 4): *Hat B n Knoten, so hat er $(n+1)/2$ Blätter und $(n–1)/2$ innere Knoten. Es gibt also genau ein Blatt mehr als innere Knoten.*

- Sei p die Anzahl der Blätter von B.
- Wenn B n Knoten hat, so hat B $n–1$ Kanten. [*gilt in jedem Baum!*]

Eigenschaften binärer regulärer Wurzelbäume

Zum Beweis von Aussage 4): *Hat B n Knoten, so hat er $(n+1)/2$ Blätter und $(n–1)/2$ innere Knoten. Es gibt also genau ein Blatt mehr als innere Knoten.*

- Sei p die Anzahl der Blätter von B.
- Wenn B n Knoten hat, so hat B n–1 Kanten. [*gilt in jedem Baum!*]
- Laut Aussage 1 und da in jedem Graphen gilt $\sum d(x) = 2 \cdot |E|$ [*vgl. früher!*]
- $\sum d(x) = p \cdot 1 + (n - p - 1) \cdot 3 + 1 \cdot 2 \quad = \quad 2|E| = 2(n-1)$

Eigenschaften binärer regulärer Wurzelbäume

Zum Beweis von Aussage 4): *Hat B n Knoten, so hat er $(n+1)/2$ Blätter und $(n–1)/2$ innere Knoten. Es gibt also genau ein Blatt mehr als innere Knoten.*

- Sei p die Anzahl der Blätter von B.
- Wenn B n Knoten hat, so hat B n–1 Kanten. [*gilt in jedem Baum!*]
- Laut Aussage 1 und da in jedem Graphen gilt $\sum d(x) = 2 \cdot |E|$ [*vgl. früher!*]
- $\sum d(x) = p \cdot 1 + (n - p - 1) \cdot 3 + 1 \cdot 2 \quad = \quad 2|E| = 2(n-1)$
- Anzahl der Blätter: Nach p auflösen $\Rightarrow p = (n+1)/2$

Eigenschaften binärer regulärer Wurzelbäume

Zum Beweis von Aussage 4): *Hat B n Knoten, so hat er $(n+1)/2$ Blätter und $(n–1)/2$ innere Knoten. Es gibt also genau ein Blatt mehr als innere Knoten.*

- Sei p die Anzahl der Blätter von B.
- Wenn B n Knoten hat, so hat B n–1 Kanten. [*gilt in jedem Baum!*]
- Laut Aussage 1 und da in jedem Graphen gilt $\sum d(x) = 2 \cdot |E|$ [*vgl. früher!*]
- $\sum d(x) = p \cdot 1 + (n-p-1) \cdot 3 + 1 \cdot 2 \quad = \quad 2|E| = 2(n-1)$
- Anzahl der Blätter: Nach p auflösen $\Rightarrow p = (n+1)/2$
- Anzahl der inneren Knoten:

$$n - p = n - (n+1)/2 = (n-1)/2 = p - 1.$$

Eigenschaften binärer regulärer Wurzelbäume

Zur Aussage 5): $[H \geq log_2(n+1)]$

Eigenschaften binärer regulärer Wurzelbäume

Zur Aussage 5): $[H \geq log_2(n+1)]$

- Mittels Induktion kann man leicht zeigen, dass sich auf Niveau k des Baumes maximal 2^{k-1} Knoten befinden.

Eigenschaften binärer regulärer Wurzelbäume

Zur Aussage 5): $[H \geq log_2(n+1)]$

- Mittels Induktion kann man leicht zeigen, dass sich auf Niveau k des Baumes maximal 2^{k-1} Knoten befinden.
- Auf dem untersten Niveau H befinden sich also maximal 2^{H-1} Knoten, die dann alle Blätter sind.

Eigenschaften binärer regulärer Wurzelbäume

Zur Aussage 5): $[H \geq log_2(n+1)]$

- Mittels Induktion kann man leicht zeigen, dass sich auf Niveau k des Baumes maximal 2^{k-1} Knoten befinden.
- Auf dem untersten Niveau H befinden sich also maximal 2^{H-1} Knoten, die dann alle Blätter sind.
- Addiert man dazu die inneren Knoten, so erhält man

  [*beachten Sie Aussage 4)!*]

  $n \leq 2^{H-1} + (2^{H-1}-1)$.

Eigenschaften binärer regulärer Wurzelbäume

Zur Aussage 5): $[H \geq log_2(n+1)]$

- Mittels Induktion kann man leicht zeigen, dass sich auf Niveau k des Baumes maximal 2^{k-1} Knoten befinden.
- Auf dem untersten Niveau H befinden sich also maximal 2^{H-1} Knoten, die dann alle Blätter sind.
- Addiert man dazu die inneren Knoten, so erhält man

  [*beachten Sie Aussage 4)!*]

  $n \leq 2^{H-1} + (2^{H-1}-1)$.
- Daraus ergibt sich also $n \leq 2 \cdot 2^{H-1}-1 = 2^H-1$ und damit

$$\log_2(n+1) \leq H$$

Suchbäume

Aussage 5) von vorher impliziert:

- In einem binären Wurzelbaum der Höhe H lassen sich bis zu $2^H - 1$ Knoten unterbringen (z.B. $H = 18 \rightarrow$ ca. 262 000 Knoten!)
- Zu jedem dieser Knoten gibt es von der Wurzel aus einen Weg, der höchstens 18 Knoten besucht !

Suchbäume

Aussage 5) von vorher impliziert:

- In einem binären Wurzelbaum der Höhe H lassen sich bis zu $2^H - 1$ Knoten unterbringen (z.B. $H = 18 \rightarrow$ ca. 262 000 Knoten!)
- Zu jedem dieser Knoten gibt es von der Wurzel aus einen Weg, der höchstens 18 Knoten besucht !

$\Rightarrow$ Datenspeicherung in den Knoten von Bäumen, auf die sehr schnell zugegriffen werden kann

$\Rightarrow$ Suchbäume

Suchbäume

- Jedem Datensatz wird ein (eindeutiger) Schlüssel zugeordnet.
- Jedem Knoten wird ein Datensatz zugeordnet.

- *Eintragen von Schlüssel und Daten in den Suchbaum*:
  Alle Schlüssel des linken Teilbaums des Knotens p sind kleiner als der Schlüssel von p, die des rechten Teilbaums sind größer als der von p.

Eintragen eines Schlüssels s

Existiert der Baum noch nicht $\Rightarrow$ erzeuge die Wurzel und trage s bei der Wurzel ein.

Ansonsten: Sei w der Schlüssel der Wurzel.

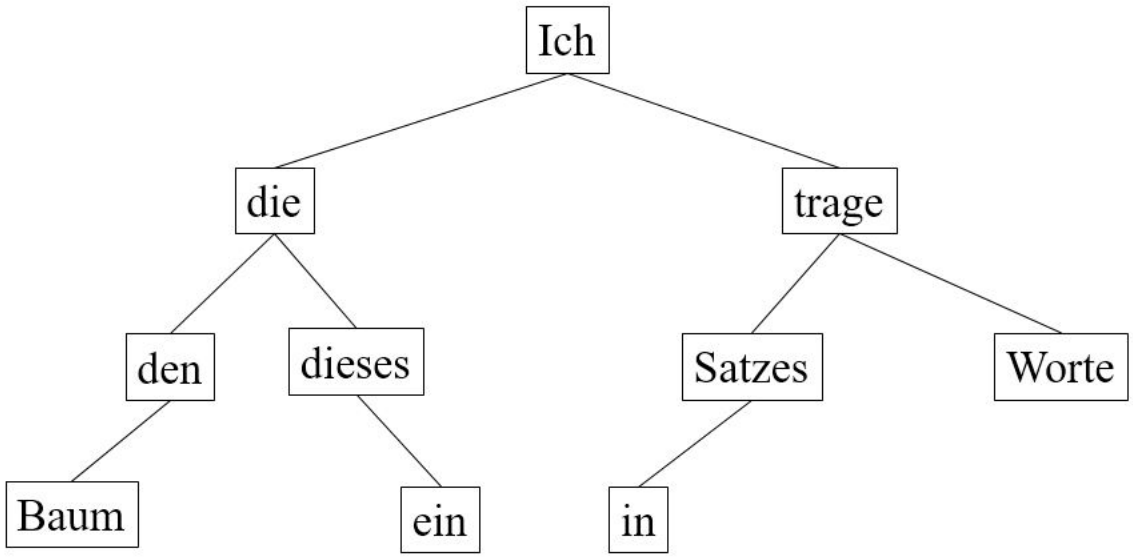
$s < w \Rightarrow$ Trage Schlüssel s im linken Teilbaum der Wurzel ein.

$s > w \Rightarrow$ Trage Schlüssel s im rechten Teilbaum der Wurzel ein.

Beispiel: Suchbäume

Suchbaum für „Ich trage die Worte dieses Satzes in den Baum ein“:

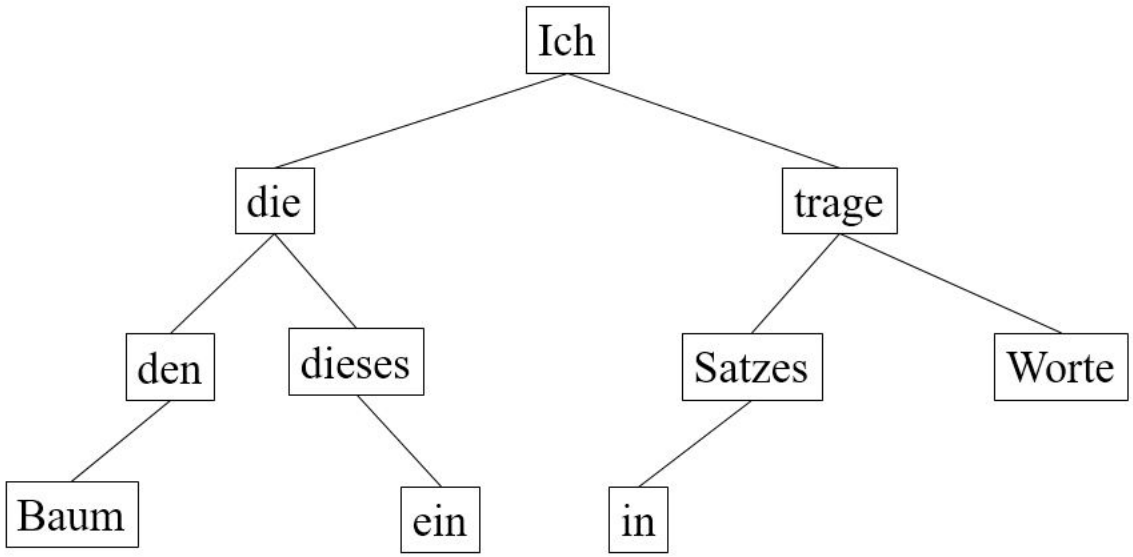
- Worte werden den Knoten zugeordnet.
- Schlüssel sind die Worte selber, alphabetische Ordnung.



Beispiel: Suchbäume

Suchbaum für „Ich trage die Worte dieses Satzes in den Baum ein“:

- Worte werden den Knoten zugeordnet.
- Schlüssel sind die Worte selber, alphabetische Ordnung.



Anmerkung: Suche umso effizienter, je weniger sich die Höhen linker und rechter Teilbäume unterscheiden $\Rightarrow$ *höhenbalancierte Bäume*

Suche nach Daten mit Schlüssel s

Die Suche nach dem Schlüssel s beginnend bei dem Knoten x
(x hat den Schlüssel s_x)

$s = s_x \Rightarrow$ Daten gefunden.

$s < s_x \Rightarrow$ Setze die Suche beim linken Sohn von x fort.

$s > s_x \Rightarrow$ Setze die Suche beim rechten Sohn von x fort.

. . . kleine Erweiterung für den Fall, dass s nicht vorhanden ist.

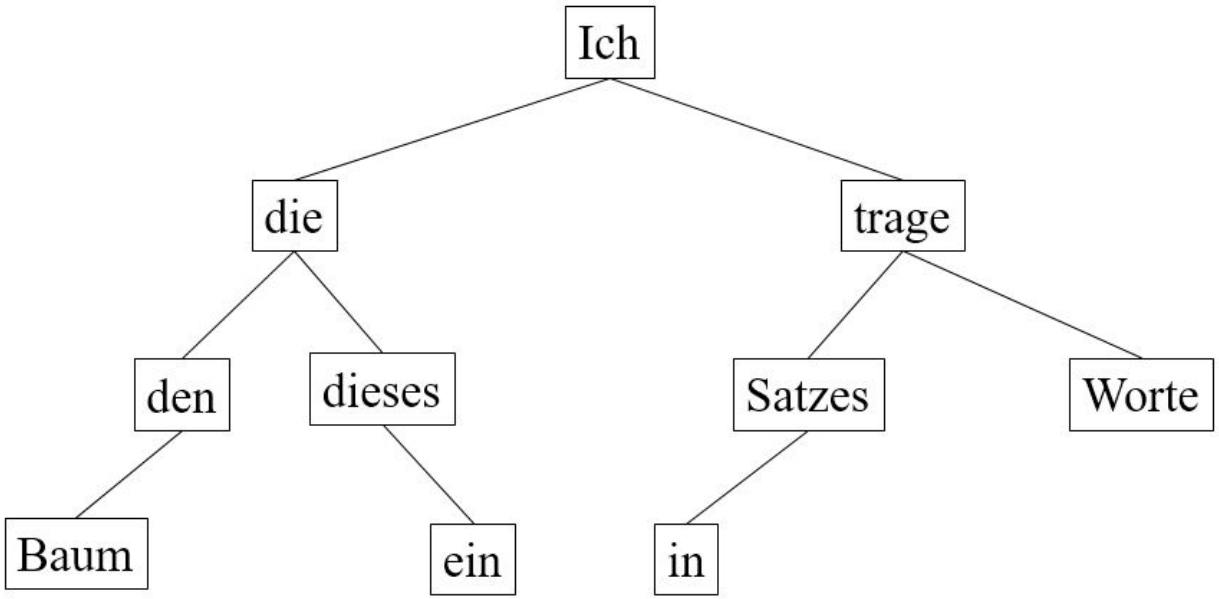
Traversierungsverfahren – Durchlaufen von Suchbäumen

Inorder-Durchlauf: Die Daten des Baums im Beispiel mit Wurzel x werden in alphabetischer Reihenfolge ausgegeben.

1. Wenn es einen linken Sohn gibt: Gib den Baum mit Wurzel linker Sohn von x alphabetisch aus.
2. Gib den Datensatz der Wurzel x aus.
3. Wenn es einen rechten Sohn gibt: Gib den Baum mit Wurzel rechter Sohn von x alphabetisch aus.

Beispiel Durchlaufen von Suchbäumen

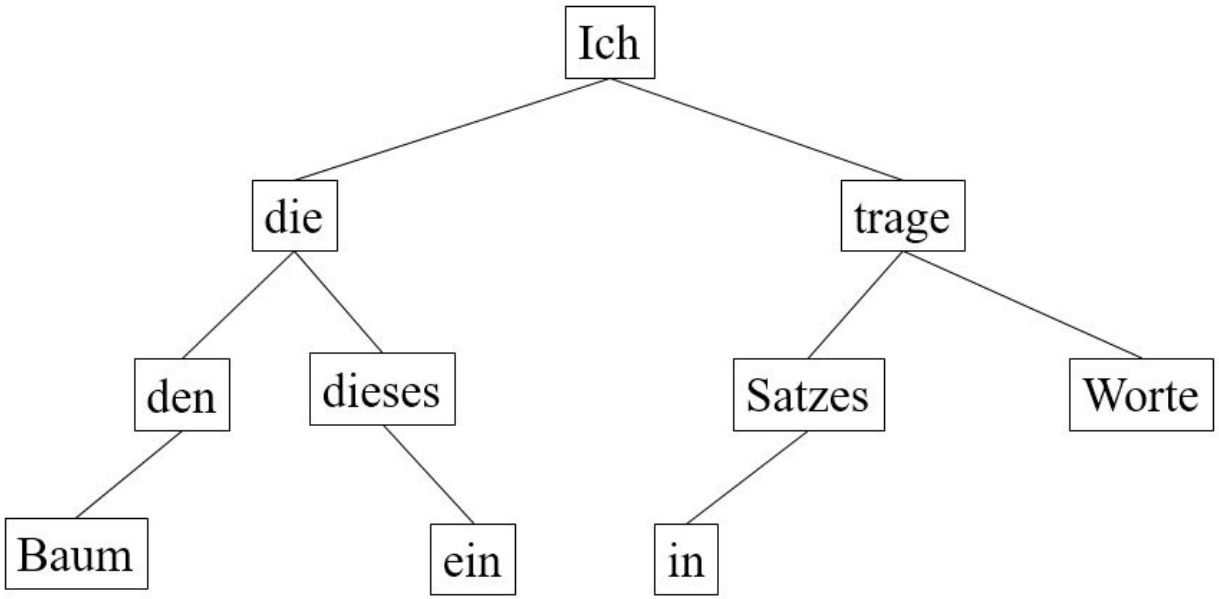
Inorder (L-W-R): „Baum den die dieses ein Ich in Satzes trage Worte“



Beispiel Durchlaufen von Suchbäumen

Inorder (L-W-R): „Baum den die dieses ein Ich in Satzes trage Worte“

Preorder (W-L-R): „Ich die den Baum dieses ein trage Satzes in Worte“

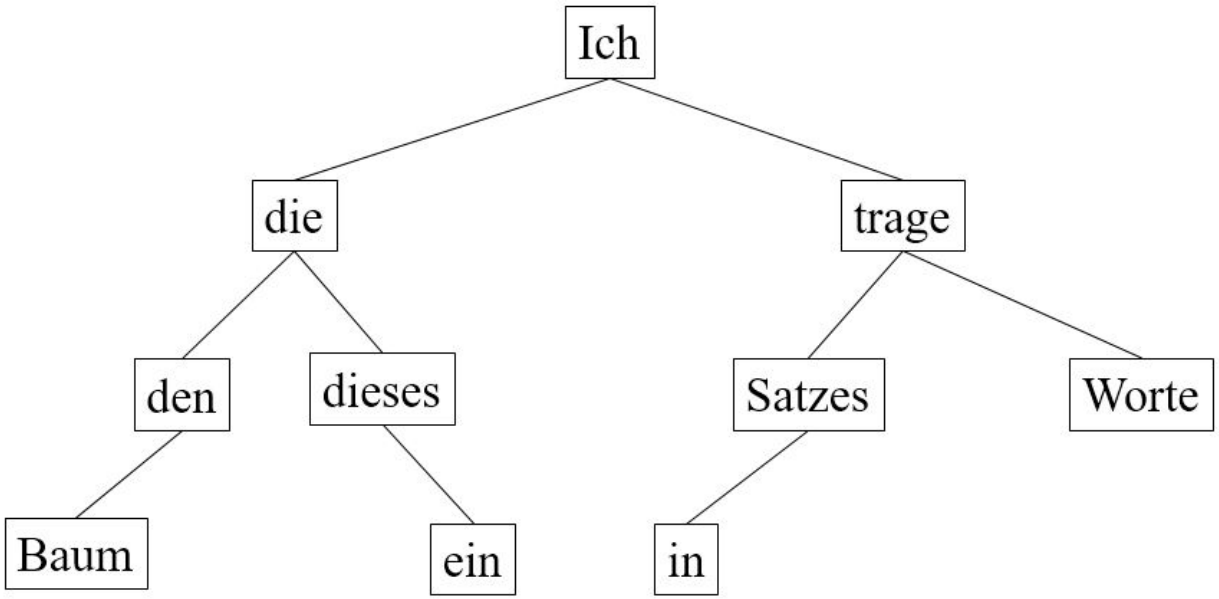


Beispiel Durchlaufen von Suchbäumen

Inorder (L-W-R): „Baum den die dieses ein Ich in Satzes trage Worte“

Preorder (W-L-R): „Ich die den Baum dieses ein trage Satzes in Worte“

Postorder (L-R-W): „Baum den ein dieses die in Satzes Worte trage Ich“



Anwendung binäre Bäume: Huffman-Code

Grundidee: Kapazität bei Datenübertragung einsparen

- Häufig vorkommende Zeichen durch kurze Codeworte ersetzen
- Selten vorkommende Zeichen durch längere Codeworte ersetzen
- Beispiel: Morsealphabet

Ziel: Minimierung von benötigtem Speicherplatz durch *variable Codewortlänge*

- Häufige Zeichen möglichst kurz codieren
- Im Gegensatz dazu: ASCII-Code – für jedes Zeichen genau ein Byte

Präfix Codes

Problem bei Codes variabler Länge: Trennung von Codewörtern („wo hört eines auf, wo beginnt das nächste?“)

Lösungen:

- Trennzeichen (z.B. Morsecode)
- *Präfix-Codes – kein Codewort ist Anfangsteil eines anderen Codeworts*
  - Bsp: Telefonnummernsystem - keine Telefonnummer beginnt mit “110”

Präfix Codes

Problem bei Codes variabler Länge: Trennung von Codewörtern („wo hört eines auf, wo beginnt das nächste?“)

Lösungen:

- Trennzeichen (z.B. Morsecode)
- *Präfix-Codes – kein Codewort ist Anfangsteil eines anderen Codeworts*
  - Bsp: Telefonnummernsystem - keine Telefonnummer beginnt mit “110”

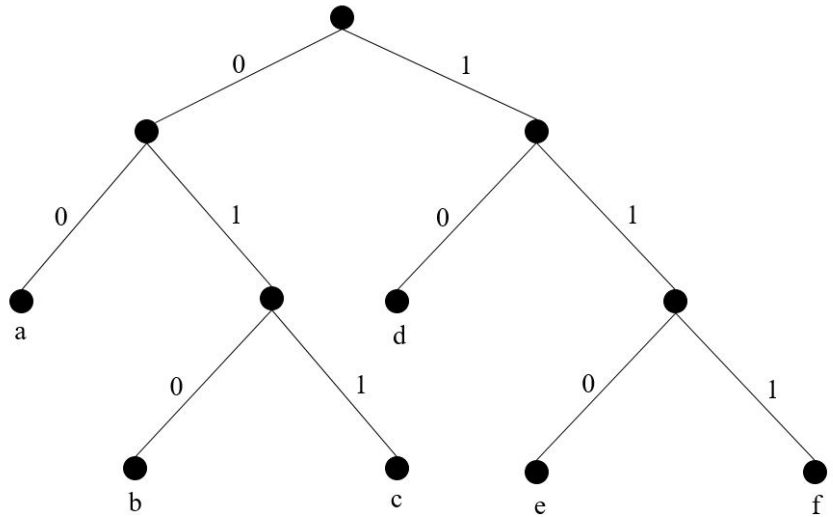
Präfix-Codes können mithilfe von Wurzelbäumen konstruiert werden.

Konstruktion von Präfix-Codes mittels Wurzelbäumen

- Codeworte bestehen nur aus „0“ und „1“ = Kantenfolgen in einem binären regulären Wurzelbaum
  - Kanten nach links: “0”, Kanten nach rechts: “1”
- Blätter enthalten die Zeichen, die kodiert werden sollen

Konstruktion von Präfix-Codes mittels Wurzelbäumen

- Codeworte bestehen nur aus „0“ und „1“ = Kantenfolgen in einem binären regulären Wurzelbaum
  - Kanten nach links: “0”, Kanten nach rechts: “1”
- Blätter enthalten die Zeichen, die kodiert werden sollen

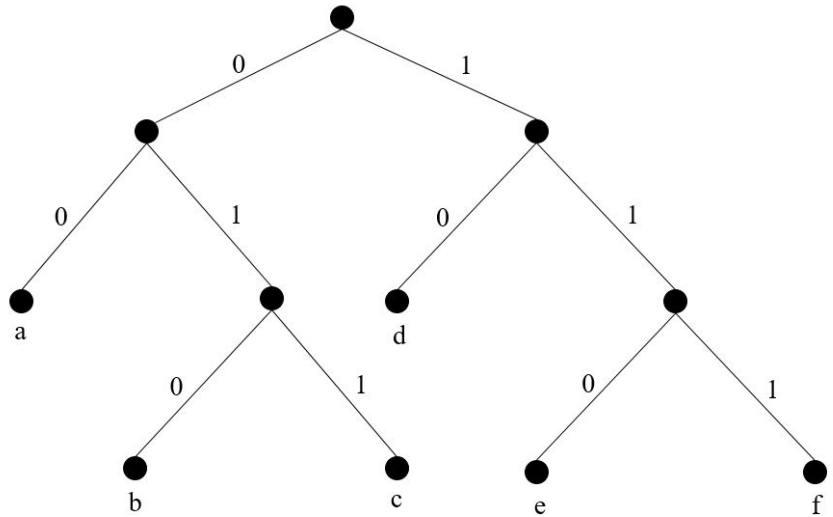


$\Rightarrow$ Kein Codewort ist Anfangsteil eines anderen
(weil sich kodierte Zeichen nur in den Blättern befinden!)

Bsp.: „0100110010“ eindeutig auflösbar in „ “

Konstruktion von Präfix-Codes mittels Wurzelbäumen

- Codeworte bestehen nur aus „0“ und „1“ = Kantenfolgen in einem binären regulären Wurzelbaum
  - Kanten nach links: “0”, Kanten nach rechts: “1”
- Blätter enthalten die Zeichen, die kodiert werden sollen

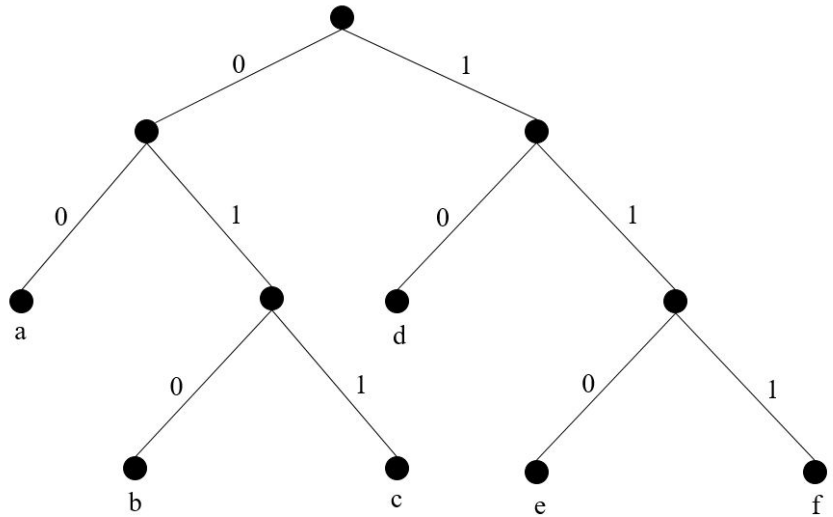


$\Rightarrow$ Kein Codewort ist Anfangsteil eines anderen
(weil sich kodierte Zeichen nur in den Blättern befinden!)

Bsp.: „0100110010“ eindeutig auflösbar in „010 “

Konstruktion von Präfix-Codes mittels Wurzelbäumen

- Codeworte bestehen nur aus „0“ und „1“ = Kantenfolgen in einem binären regulären Wurzelbaum
  - Kanten nach links: “0”, Kanten nach rechts: “1”
- Blätter enthalten die Zeichen, die kodiert werden sollen

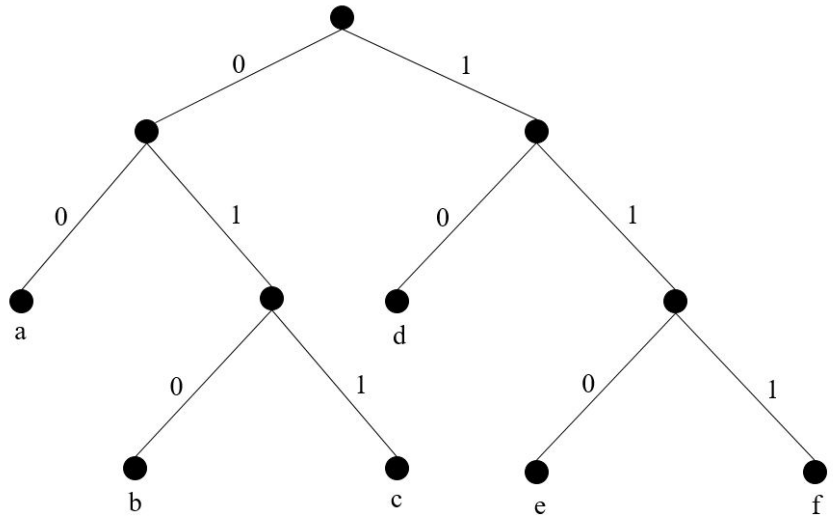


$\Rightarrow$ Kein Codewort ist Anfangsteil eines anderen
(weil sich kodierte Zeichen nur in den Blättern befinden!)

Bsp.: „0100110010“ eindeutig auflösbar in „010 011 “

Konstruktion von Präfix-Codes mittels Wurzelbäumen

- Codeworte bestehen nur aus „0“ und „1“ = Kantenfolgen in einem binären regulären Wurzelbaum
  - Kanten nach links: “0”, Kanten nach rechts: “1”
- Blätter enthalten die Zeichen, die kodiert werden sollen

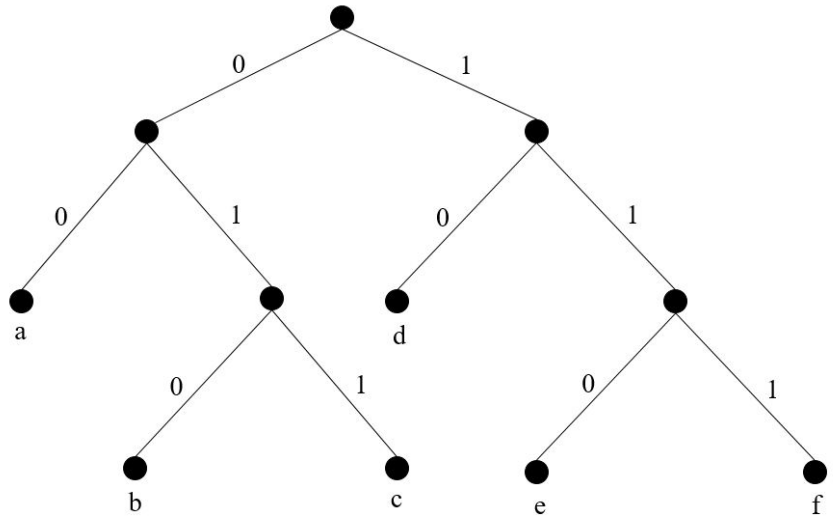


$\Rightarrow$ Kein Codewort ist Anfangsteil eines anderen
(weil sich kodierte Zeichen nur in den Blättern befinden!)

Bsp.: „0100110010“ eindeutig auflösbar in „010 011 00 “

Konstruktion von Präfix-Codes mittels Wurzelbäumen

- Codeworte bestehen nur aus „0“ und „1“ = Kantenfolgen in einem binären regulären Wurzelbaum
  - Kanten nach links: “0”, Kanten nach rechts: “1”
- Blätter enthalten die Zeichen, die kodiert werden sollen



$\Rightarrow$ Kein Codewort ist Anfangsteil eines anderen
(weil sich kodierte Zeichen nur in den Blättern befinden!)

Bsp.: „0100110010“ eindeutig auflösbar in „010 011 00 10“

Optimaler Präfix-Code: Huffman-Code

Häufig vorkommende Zeichen kurze Codewörter, seltene Zeichen längere Codewörter $\Rightarrow$ Huffman-Algorithmus (produziert optimalen Präfix-Code)

- **Gegeben:** Quellalphabet, Häufigkeit des Auftretens der Zeichen, z. B. $a : 4\%$ $b : 15\%$ $c : 10\%$ $d : 15\%$ $e : 36\%$ $f : 20\%$

Optimaler Präfix-Code: Huffman-Code

Häufig vorkommende Zeichen kurze Codewörter, seltene Zeichen längere Codewörter $\Rightarrow$ Huffman-Algorithmus (produziert optimalen Präfix-Code)

- **Gegeben:** Quellalphabet, Häufigkeit des Auftretens der Zeichen, z. B. $a : 4\%$ $b : 15\%$ $c : 10\%$ $d : 15\%$ $e : 36\%$ $f : 20\%$
- **Algorithmus:**
  1. Die Zeichen werden die Blätter eines binären Baumes. Die Blätter enthalten Häufigkeiten der Zeichen (aufsteigend geordnet).

Optimaler Präfix-Code: Huffman-Code

Häufig vorkommende Zeichen kurze Codewörter, seltene Zeichen längere Codewörter $\Rightarrow$ Huffman-Algorithmus (produziert optimalen Präfix-Code)

- **Gegeben:** Quellalphabet, Häufigkeit des Auftretens der Zeichen, z. B. $a : 4\%$ $b : 15\%$ $c : 10\%$ $d : 15\%$ $e : 36\%$ $f : 20\%$
- **Algorithmus:**
  1. Die Zeichen werden die Blätter eines binären Baumes. Die Blätter enthalten Häufigkeiten der Zeichen (aufsteigend geordnet).
  2. Suche die zwei kleinsten Knoten ohne Vater. Falls es mehrere gibt, wähle zwei beliebige. Erstelle einen neuen Vaterknoten zu diesen beiden Knoten und ordne diesem die Summe deren Häufigkeiten zu.

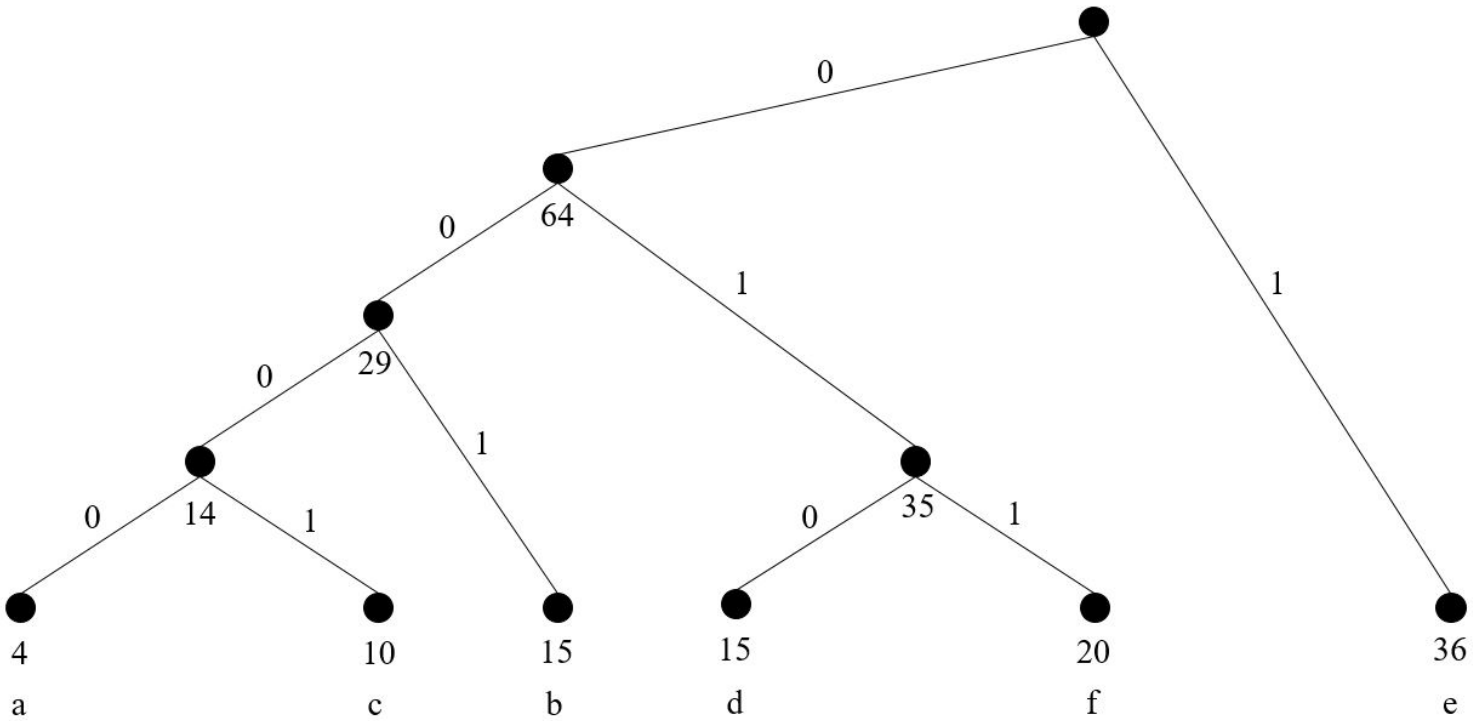
Optimaler Präfix-Code: Huffman-Code

Häufig vorkommende Zeichen kurze Codewörter, seltene Zeichen längere Codewörter $\Rightarrow$ Huffman-Algorithmus (produziert optimalen Präfix-Code)

- **Gegeben:** Quellalphabet, Häufigkeit des Auftretens der Zeichen, z. B. $a : 4\%$ $b : 15\%$ $c : 10\%$ $d : 15\%$ $e : 36\%$ $f : 20\%$

- **Algorithmus:**
  1. Die Zeichen werden die Blätter eines binären Baumes. Die Blätter enthalten Häufigkeiten der Zeichen (aufsteigend geordnet).
  2. Suche die zwei kleinsten Knoten ohne Vater. Falls es mehrere gibt, wähle zwei beliebige. Erstelle einen neuen Vaterknoten zu diesen beiden Knoten und ordne diesem die Summe deren Häufigkeiten zu.
  3. Wiederhole Schritt 2 so oft, bis nur noch ein Knoten ohne Vater übrig ist. Dieser ist die Wurzel des Baums.

Beispiel: Huffman-Code



Anwendung binäre Bäume: Huffman-Code

- Der Huffman-Code wird auch bei modernen Kompressionsverfahren eingesetzt (meist als Teil mehrstufiger Algorithmen).
  - z. B. bei der Faxcodierung oder als Teil des jpeg-Algorithmus zur Bildkomprimierung

Wer sich dafür interessiert, tiefer in die Überlappungsbereiche Informatik – Mathematik einzutauchen → bitte um Kontaktaufnahme!

Forschungsprojekte zu Algorithmen in verschiedenen Anwendungsbereichen:

- *Sicherheit (crime forecasting, . . .)*
- *Fehlertoleranz (exascale computing, Internet-of-things, . . .)*
- *Mobilfunk/Telekommunikation (5G, . . .)*