Bäume und Wälder

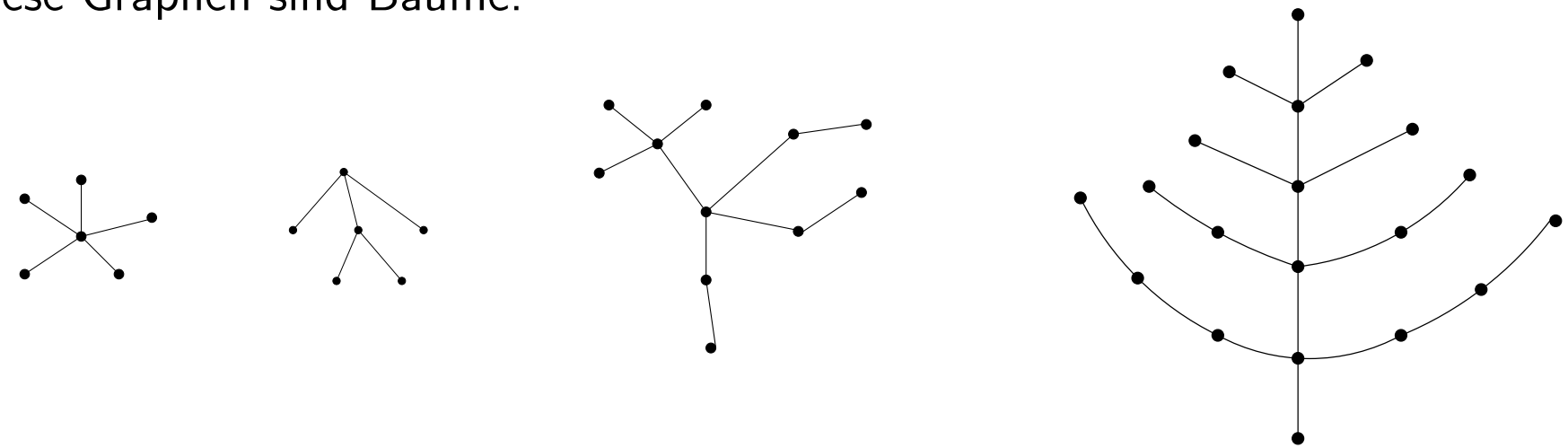
Bäume

Ein (ungerichteter) **Baum** ist ein ungerichteter Graph $G = (V, E)$, der zusammenhängend ist und keine Kreise enthält.

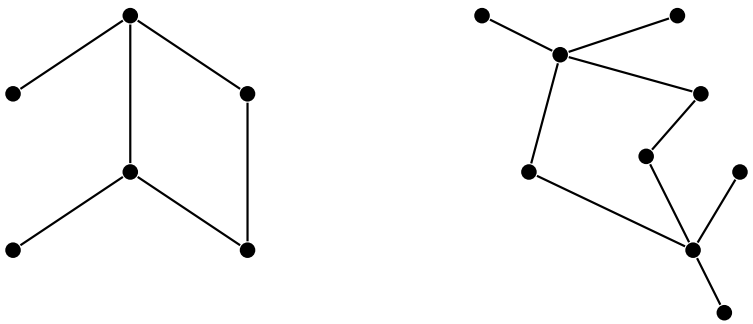
Bäume

Ein (ungerichteter) **Baum** ist ein ungerichteter Graph $G = (V, E)$, der zusammenhängend ist und keine Kreise enthält.

Diese Graphen sind Bäume:



Diese aber nicht:



Wälder

Knoten-disjunkte Vereinigungen von Bäumen heißen Wälder. Präziser:

Die Graphen $B_1 = (V_1, E_1), \ldots, (V_k, E_k)$ seien Bäume und es gelte

$$V_i \cap V_j = \emptyset \text{ für } i \neq j.$$

Dann heißt $G = (V, E)$ ein **Wald**, falls

$$V = \bigcup_{i=1}^{k} V_i \text{ und } E = \bigcup_{i=1}^{k} E_k.$$

.

Wälder

Knoten-disjunkte Vereinigungen von Bäumen heißen Wälder. Präziser:

Die Graphen $B_1 = (V_1, E_1), \ldots, (V_k, E_k)$ seien Bäume und es gelte

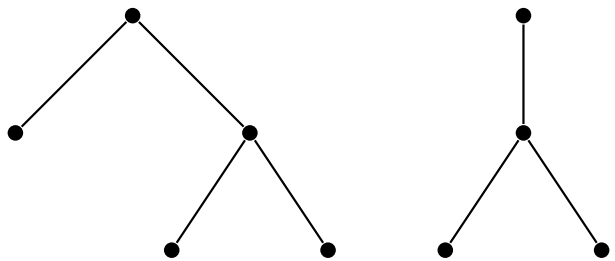
$$V_i \cap V_j = \emptyset \text{ für } i \neq j.$$

Dann heißt $G = (V, E)$ ein **Wald**, falls

$$V = \bigcup_{i=1}^{k} V_i \text{ und } E = \bigcup_{i=1}^{k} E_k.$$

.

Der Graph



ist kein Baum, da nicht zusammenhängend, wohl aber ein Wald.

Eigenschaften von Bäumen (1/2)

Sei $B = (V, E)$ ein Baum und seien $x, y \in V$ Knoten.
Dann gibt es genau einen Weg von x nach y.

Beweis: Siehe Tafel.

Eigenschaften von Bäumen (1/2)

Sei $B = (V, E)$ ein Baum und seien $x, y \in V$ Knoten.
Dann gibt es genau einen Weg von x nach y.

Beweis: Siehe Tafel.

Sei $B = (V, E)$ ein Baum. Wir nennen einen Knoten $b \in V$ ein

Blatt,

wenn b den Grad höchstens 1 besitzt.
Zeige: Jeder Baum besitzt mindestens ein Blatt.

Beweis: Siehe Tafel.

Eigenschaften von Bäumen (2/2)

Sei $B = (V, E)$ ein Baum. Dann hat B genau einen Knoten mehr als Kanten,

d.h. es gilt $|E| = |V| - 1$.

Eigenschaften von Bäumen (2/2)

Sei $B = (V, E)$ ein Baum. Dann hat B genau einen Knoten mehr als Kanten,

$$\text{d.h. es gilt } |E| = |V| - 1.$$

Führe einen Beweis durch

vollständige Induktion nach der Anzahl n der Knoten von B.

(a) Der **Induktionsanfang** für $n = 1$:

Eigenschaften von Bäumen (2/2)

Sei $B = (V, E)$ ein Baum. Dann hat B genau einen Knoten mehr als Kanten,

$$\text{d.h. es gilt } |E| = |V| - 1.$$

Führe einen Beweis durch

vollständige Induktion nach der Anzahl n der Knoten von B.

(a) Der **Induktionsanfang** für $n = 1$: Ein ungerichteter Graph B mit einem Knoten v hat keine Kanten.

Eigenschaften von Bäumen (2/2)

Sei $B = (V, E)$ ein Baum. Dann hat B genau einen Knoten mehr als Kanten,

$$\text{d.h. es gilt } |E| = |V| - 1.$$

Führe einen Beweis durch

vollständige Induktion nach der Anzahl n der Knoten von B.

(a) Der **Induktionsanfang** für $n = 1$: Ein ungerichteter Graph B mit einem Knoten v hat keine Kanten. Also ist v ein Blatt.

Eigenschaften von Bäumen (2/2)

Sei $B = (V, E)$ ein Baum. Dann hat B genau einen Knoten mehr als Kanten,

d.h. es gilt $|E| = |V| - 1$.

Führe einen Beweis durch

vollständige Induktion nach der Anzahl n der Knoten von B.

(a) Der **Induktionsanfang** für $n = 1$: Ein ungerichteter Graph B mit einem Knoten v hat keine Kanten. Also ist v ein Blatt.

(b) Der **Induktionsschritt**: Sei $B = (V, E)$ ein Baum mit $|V| = n + 1$ Knoten. **Zeige**, dass B genau n Kanten besitzt.

Eigenschaften von Bäumen (2/2)

Sei $B = (V, E)$ ein Baum. Dann hat B genau einen Knoten mehr als Kanten,

d.h. es gilt $|E| = |V| - 1$.

Führe einen Beweis durch

vollständige Induktion nach der Anzahl n der Knoten von B.

(a) Der **Induktionsanfang** für $n = 1$: Ein ungerichteter Graph B mit einem Knoten v hat keine Kanten. Also ist v ein Blatt.

(b) Der **Induktionsschritt**: Sei $B = (V, E)$ ein Baum mit $|V| = n + 1$ Knoten. **Zeige**, dass B genau n Kanten besitzt.

- Die **Induktionsannahme**: Jeder Baum mit n Knoten hat $n - 1$ Kanten.

Eigenschaften von Bäumen (2/2)

Sei $B = (V, E)$ ein Baum. Dann hat B genau einen Knoten mehr als Kanten,

d.h. es gilt $|E| = |V| - 1$.

Führe einen Beweis durch

vollständige Induktion nach der Anzahl n der Knoten von B.

(a) Der **Induktionsanfang** für $n = 1$: Ein ungerichteter Graph B mit einem Knoten v hat keine Kanten. Also ist v ein Blatt.

(b) Der **Induktionsschritt**: Sei $B = (V, E)$ ein Baum mit $|V| = n + 1$ Knoten. **Zeige**, dass B genau n Kanten besitzt.

- Die **Induktionsannahme**: Jeder Baum mit n Knoten hat $n - 1$ Kanten.
- Die **Idee**: Entferne ein Blatt b aus B und die eine mit b inzidente Kante. Wende dann die Induktionsannahme auf den neuen Baum mit n Knoten an.

Eigenschaften von Bäumen (2/2)

Sei $B = (V, E)$ ein Baum. Dann hat B genau einen Knoten mehr als Kanten,

d.h. es gilt $|E| = |V| - 1$.

Führe einen Beweis durch

vollständige Induktion nach der Anzahl n der Knoten von B.

(a) Der **Induktionsanfang** für $n = 1$: Ein ungerichteter Graph B mit einem Knoten v hat keine Kanten. Also ist v ein Blatt.

(b) Der **Induktionsschritt**: Sei $B = (V, E)$ ein Baum mit $|V| = n + 1$ Knoten. **Zeige**, dass B genau n Kanten besitzt.

- Die **Induktionsannahme**: Jeder Baum mit n Knoten hat $n - 1$ Kanten.
- Die **Idee**: Entferne ein Blatt b aus B und die eine mit b inzidente Kante. Wende dann die Induktionsannahme auf den neuen Baum mit n Knoten an.
- Das formale Argument: Siehe Tafel.

Spannbäume

Bäume, die „ihren“ Graphen aufspannen

Sei $G = (V, E)$ ein ungerichteter Graph. Ein Baum $B = (V', E')$ heißt

Spannbaum von G,

falls $V' = V$ und $E' \subseteq E$ gilt.

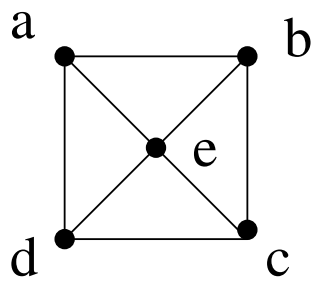
Bäume, die „ihren“ Graphen aufspannen

Sei $G = (V, E)$ ein ungerichteter Graph. Ein Baum $B = (V', E')$ heißt

Spannbaum von G,

falls $V' = V$ und $E' \subseteq E$ gilt.

Der Graph



hat u.a. folgende Spannbäume:

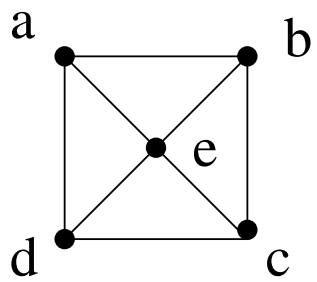
Bäume, die „ihren“ Graphen aufspannen

Sei $G = (V, E)$ ein ungerichteter Graph. Ein Baum $B = (V', E')$ heißt

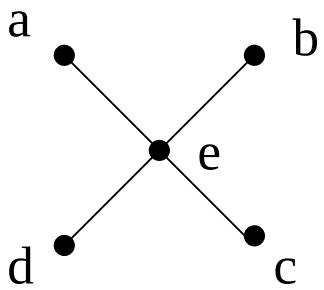
Spannbaum von G,

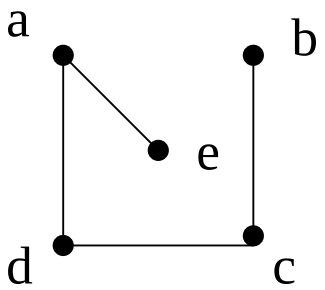
falls $V' = V$ und $E' \subseteq E$ gilt.

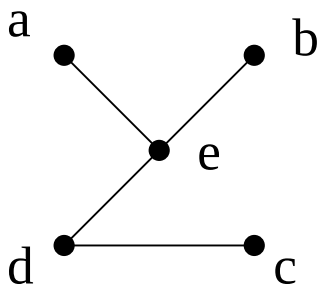
Der Graph



hat u.a. folgende Spannbäume:







Spannbäume: Kleinste zusammenhängende Teilgraphen

Sei $G = (V, E)$ ein ungerichteter Graph. Dann gilt:

Es gibt mindestens einen Spannbaum von $G \iff G$ ist zusammenhängend.

Spannbäume: Kleinste zusammenhängende Teilgraphen

Sei $G = (V, E)$ ein ungerichteter Graph. Dann gilt:

Es gibt mindestens einen Spannbaum von $G \iff G$ ist zusammenhängend.

$\Rightarrow$ G ist zusammenhängend, denn

Spannbäume: Kleinste zusammenhängende Teilgraphen

Sei $G = (V, E)$ ein ungerichteter Graph. Dann gilt:

Es gibt mindestens einen Spannbaum von $G \iff G$ ist zusammenhängend.

$\Rightarrow$ G ist zusammenhängend, denn G hat nach Annahme einen Spannbaum und Bäume sind zusammenhängend.

Spannbäume: Kleinste zusammenhängende Teilgraphen

Sei $G = (V, E)$ ein ungerichteter Graph. Dann gilt:

Es gibt mindestens einen Spannbaum von $G \iff G$ ist zusammenhängend.

- $\Rightarrow$ G ist zusammenhängend, denn G hat nach Annahme einen Spannbaum und Bäume sind zusammenhängend.
- $\Leftarrow$ Wir konstruieren einen Spannbaum:

Spannbäume: Kleinste zusammenhängende Teilgraphen

Sei $G = (V, E)$ ein ungerichteter Graph. Dann gilt:

Es gibt mindestens einen Spannbaum von $G \iff G$ ist zusammenhängend.

- $\Rightarrow$ G ist zusammenhängend, denn G hat nach Annahme einen Spannbaum und Bäume sind zusammenhängend.
- $\Leftarrow$ Wir konstruieren einen Spannbaum: Entferne eine Kante e aus G, wenn e zu einem Kreis gehört.

Spannbäume: Kleinste zusammenhängende Teilgraphen

Sei $G = (V, E)$ ein ungerichteter Graph. Dann gilt:

Es gibt mindestens einen Spannbaum von $G \iff G$ ist zusammenhängend.

- $\Rightarrow$ G ist zusammenhängend, denn G hat nach Annahme einen Spannbaum und Bäume sind zusammenhängend.
- $\Leftarrow$ Wir konstruieren einen Spannbaum: Entferne eine Kante e aus G, wenn e zu einem Kreis gehört.
  - Der neue Graph hat eine Kante weniger, ist aber noch immer zusammenhängend.

Spannbäume: Kleinste zusammenhängende Teilgraphen

Sei $G = (V, E)$ ein ungerichteter Graph. Dann gilt:

Es gibt mindestens einen Spannbaum von $G \iff G$ ist zusammenhängend.

- $\Rightarrow$ G ist zusammenhängend, denn G hat nach Annahme einen Spannbaum und Bäume sind zusammenhängend.
- $\Leftarrow$ Wir konstruieren einen Spannbaum: Entferne eine Kante e aus G, wenn e zu einem Kreis gehört.
  - Der neue Graph hat eine Kante weniger, ist aber noch immer zusammenhängend.
  - Entferne solange Kanten, bis alle Kreise zerstört sind: Wir haben einen (Spann-)Baum erhalten!

Spannbäume: Kleinste zusammenhängende Teilgraphen

Sei $G = (V, E)$ ein ungerichteter Graph. Dann gilt:

Es gibt mindestens einen Spannbaum von $G \iff G$ ist zusammenhängend.

- $\Rightarrow$ G ist zusammenhängend, denn G hat nach Annahme einen Spannbaum und Bäume sind zusammenhängend.
- $\Leftarrow$ Wir konstruieren einen Spannbaum: Entferne eine Kante e aus G, wenn e zu einem Kreis gehört.
  - Der neue Graph hat eine Kante weniger, ist aber noch immer zusammenhängend.
  - Entferne solange Kanten, bis alle Kreise zerstört sind: Wir haben einen (Spann-)Baum erhalten!

Spannbäume sind die **kleinsten** zusammenhängenden Teilgraphen[a] eines zusammenhängenden Graphen.

[a] $G = (V, E)$ ist **Teilgraph** von $H = (V, F)$, wenn $E \subseteq F$ gilt.

Warum?

Spannbäume: Kleinste zusammenhängende Teilgraphen

Sei $G = (V, E)$ ein ungerichteter Graph. Dann gilt:

Es gibt mindestens einen Spannbaum von $G \iff G$ ist zusammenhängend.

- $\Rightarrow$ G ist zusammenhängend, denn G hat nach Annahme einen Spannbaum und Bäume sind zusammenhängend.
- $\Leftarrow$ Wir konstruieren einen Spannbaum: Entferne eine Kante e aus G, wenn e zu einem Kreis gehört.
  - Der neue Graph hat eine Kante weniger, ist aber noch immer zusammenhängend.
  - Entferne solange Kanten, bis alle Kreise zerstört sind: Wir haben einen (Spann-)Baum erhalten!

Spannbäume sind die **kleinsten** zusammenhängenden Teilgraphen[a] eines zusammenhängenden Graphen.

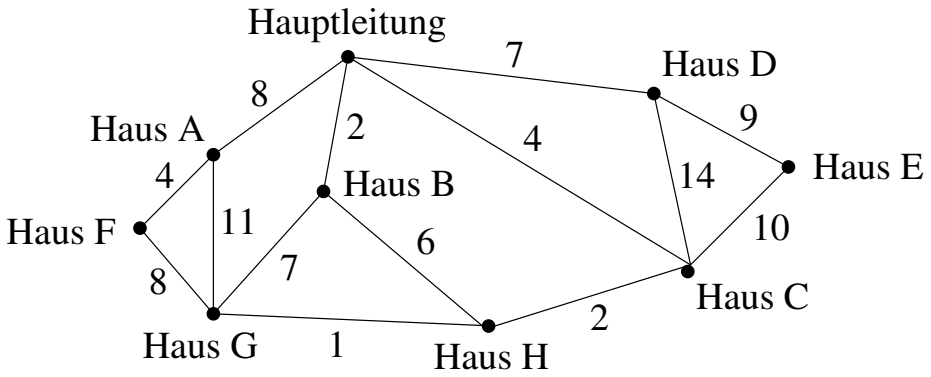
[a] $G = (V, E)$ ist **Teilgraph** von $H = (V, F)$, wenn $E \subseteq F$ gilt.

Warum? Ein kleinster zusammenhängender Teilgraph enthält keine Kreise und muss ein Baum sein!

Kabelfernsehen (1/2)

Eine Firma will Leitungen zum Empfang von Kabelfernsehen in einem neuen Wohngebiet verlegen.

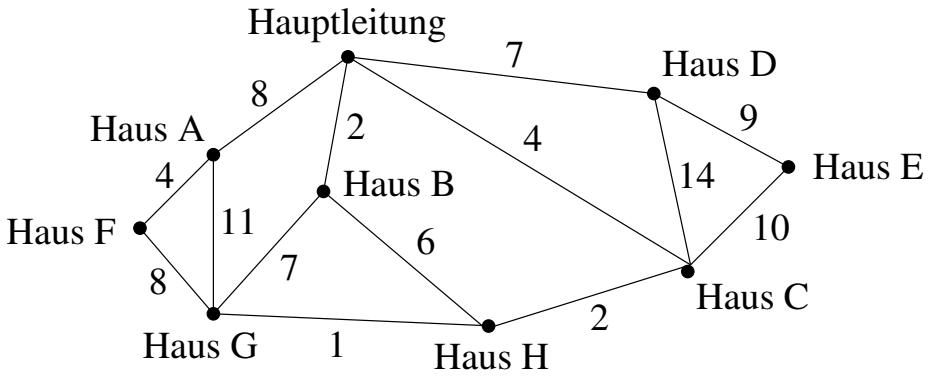
Der folgende Graph skizziert das Wohngebiet:



Kabelfernsehen (1/2)

Eine Firma will Leitungen zum Empfang von Kabelfernsehen in einem neuen Wohngebiet verlegen.

Der folgende Graph skizziert das Wohngebiet:

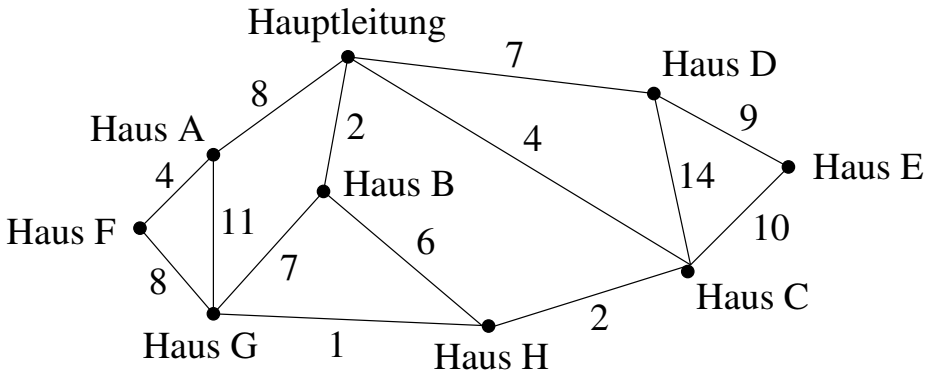


- Knoten entsprechen einzelnen Häusern bzw. der Hauptleitung, die aus einem bereits verkabelten Gebiet herangeführt wird.

Kabelfernsehen (1/2)

Eine Firma will Leitungen zum Empfang von Kabelfernsehen in einem neuen Wohngebiet verlegen.

Der folgende Graph skizziert das Wohngebiet:

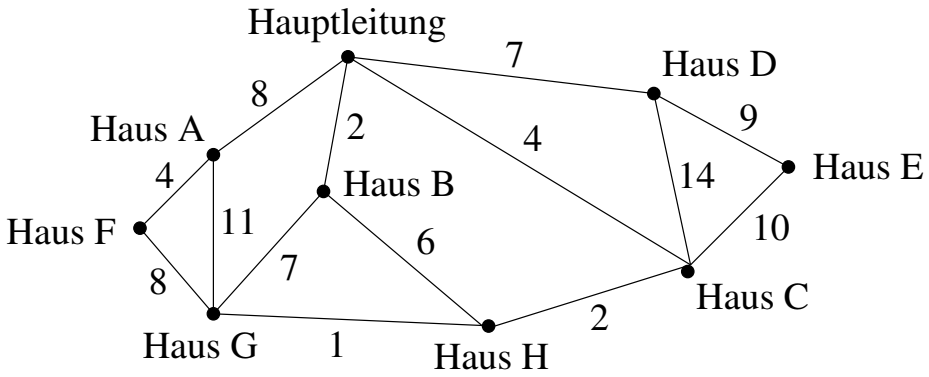


- Knoten entsprechen einzelnen Häusern bzw. der Hauptleitung, die aus einem bereits verkabelten Gebiet herangeführt wird.
- Eine Kante zwischen zwei Knoten zeigt, dass eine Direktleitung gelegt werden kann. Die Kante ist mit den Kosten der Verlegung (in TEuro) beschriftet.

Kabelfernsehen (1/2)

Eine Firma will Leitungen zum Empfang von Kabelfernsehen in einem neuen Wohngebiet verlegen.

Der folgende Graph skizziert das Wohngebiet:



- Knoten entsprechen einzelnen Häusern bzw. der Hauptleitung, die aus einem bereits verkabelten Gebiet herangeführt wird.
- Eine Kante zwischen zwei Knoten zeigt, dass eine Direktleitung gelegt werden kann. Die Kante ist mit den Kosten der Verlegung (in TEuro) beschriftet.

Bestimme eine billigste Verkabelung, die alle Häuser, (möglicherweise über eine Folge von Kabeln) an die Hauptleitung anschließt!

Kabelfernsehen (2/2)

Wir suchen

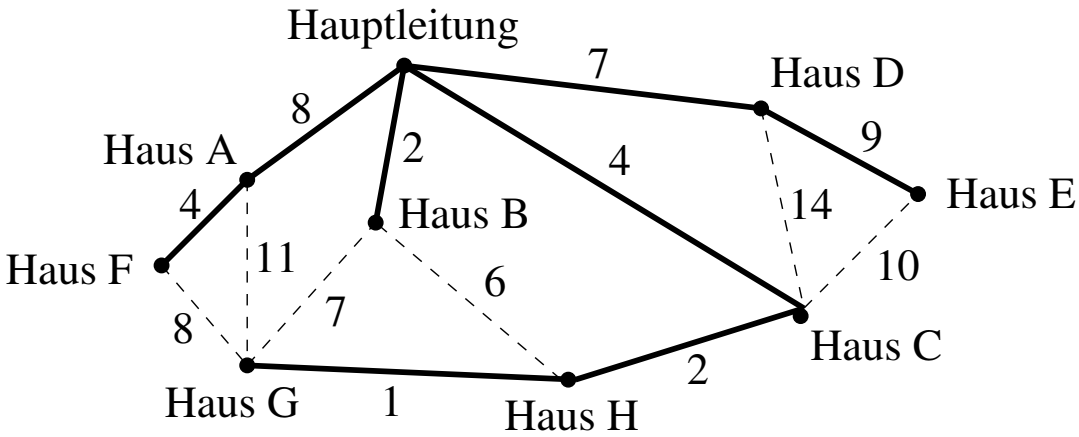
Kabelfernsehen (2/2)

Wir suchen einen **minimalen Spannbaum**,
also einen Spannbaum mit möglichst kleiner Summe von Kantengewichten.

Kabelfernsehen (2/2)

Wir suchen einen **minimalen Spannbaum**,
also einen Spannbaum mit möglichst kleiner Summe von Kantengewichten.

Die **fett** gezeichneten Kanten geben die Kanten eines minimalen Spannbaums an:



Verlegt die Firma genau diese Leitungen, so hat sie das neue Wohngebiet mit den geringstmöglichen Kosten ans Kabelfernsehen angeschlossen.

In den Veranstaltungen „**Datenstrukturen**“ und „**Theoretische Informatik 1**“ werden schnelle Algorithmen zur Bestimmung minimaler Spannbäume besprochen.

Gewurzelte Bäume

Bäume mit Wurzeln

Wir erhalten einen

gewurzelten Baum,

also einen gerichteten Baum mit Wurzel, indem man

1. in einem ungerichteten Baum einen Knoten als „**Wurzel**“ auswählt und
2. alle Kanten in die Richtung orientiert, die von der Wurzel weg führt.

Bäume mit Wurzeln

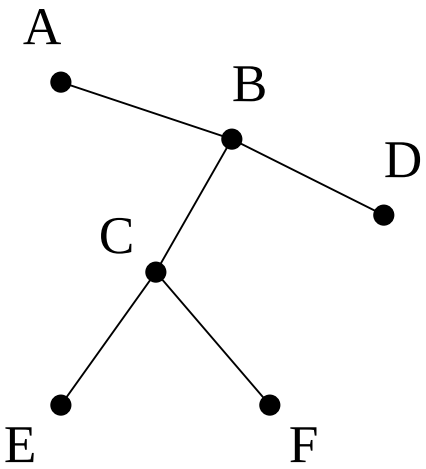
Wir erhalten einen

gewurzelten Baum,

also einen gerichteten Baum mit Wurzel, indem man

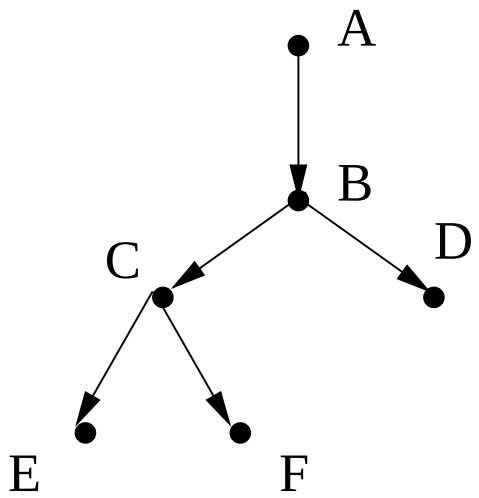
1. in einem ungerichteten Baum einen Knoten als „**Wurzel**“ auswählt und
2. alle Kanten in die Richtung orientiert, die von der Wurzel weg führt.

Ein ungerichteter Baum



und der zugehörige gerichtete Baum mit Wurzel A

$G_A :=$



Gewurzelte Bäume: Die formale Definition

Die präzise Definition des Begriffs „gewurzelter Baum“:

Ein gerichteter Graph $G = (V, E)$ heißt **gewurzelter Baum**,
falls er folgende Eigenschaften hat:

Gewurzelte Bäume: Die formale Definition

Die präzise Definition des Begriffs „gewurzelter Baum“:

Ein gerichteter Graph $G = (V, E)$ heißt **gewurzelter Baum**,
falls er folgende Eigenschaften hat:

(1) G besitzt genau einen Knoten $w \in V$ mit $\text{Ein-Grad}_G(w) = 0$.
Dieser Knoten wird **Wurzel** genannt.

(2) Für jeden Knoten $v \in V$ gilt:
Es gibt in G einen Weg von der Wurzel zum Knoten v.

(3) Für jeden Knoten $v \in V$ gilt: $\text{Ein-Grad}_G(v) \leqslant 1$.

Blätter, innere Knoten, Tiefe, Höhe, Eltern und Kinder

Blätter, innere Knoten und Kinder

Sei $B = (V, E)$ ein gewurzelter Baum mit Wurzel w.

(a) Blätter und innere Knoten:

- Knoten mit Aus-Grad 0 heißen **Blätter**,
- Knoten, die weder Wurzel noch Blätter sind, heißen **innere Knoten**.

Blätter, innere Knoten und Kinder

Sei $B = (V, E)$ ein gewurzelter Baum mit Wurzel w.

(a) Blätter und innere Knoten:

- Knoten mit Aus-Grad 0 heißen **Blätter**,
- Knoten, die weder Wurzel noch Blätter sind, heißen **innere Knoten**.

(b) Sei $v \in V$ ein Knoten von B.

- Die **Höhe von v** ist die Länge eines längsten Weges von v zu einem Blatt.
  Die **Höhe von B** ist die Höhe der Wurzel w.

Blätter, innere Knoten und Kinder

Sei $B = (V, E)$ ein gewurzelter Baum mit Wurzel w.

(a) Blätter und innere Knoten:

- Knoten mit Aus-Grad 0 heißen **Blätter**,
- Knoten, die weder Wurzel noch Blätter sind, heißen **innere Knoten**.

(b) Sei $v \in V$ ein Knoten von B.

- Die **Höhe von** v ist die Länge eines längsten Weges von v zu einem Blatt.
  Die **Höhe von** B ist die Höhe der Wurzel w.
- Die **Tiefe von** v ist die Länge des Weges von der Wurzel w zu v.
  Die **Tiefe von** B ist die maximale Tiefe eines Blatts.

Blätter, innere Knoten und Kinder

Sei $B = (V, E)$ ein gewurzelter Baum mit Wurzel w.

(a) Blätter und innere Knoten:

- Knoten mit Aus-Grad 0 heißen **Blätter**,
- Knoten, die weder Wurzel noch Blätter sind, heißen **innere Knoten**.

(b) Sei $v \in V$ ein Knoten von B.

- Die **Höhe von** v ist die Länge eines längsten Weges von v zu einem Blatt.
  Die **Höhe von** B ist die Höhe der Wurzel w.
- Die **Tiefe von** v ist die Länge des Weges von der Wurzel w zu v.
  Die **Tiefe von** B ist die maximale Tiefe eines Blatts.

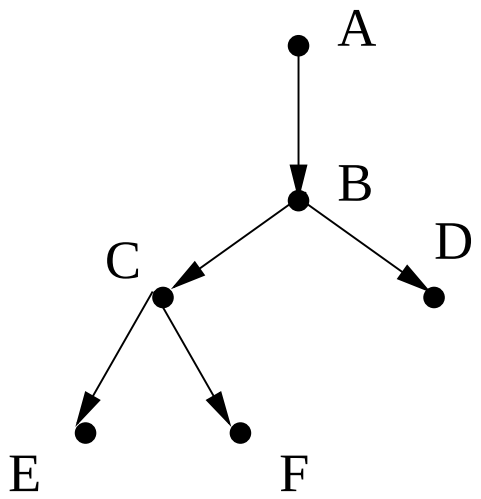
(c) Für jede Kante $(u, v) \in E$ heißt

- u der **Elternknoten** von v und
- v ein **Kind** von u.

Beispiele

Im Graphen

$G_A =$



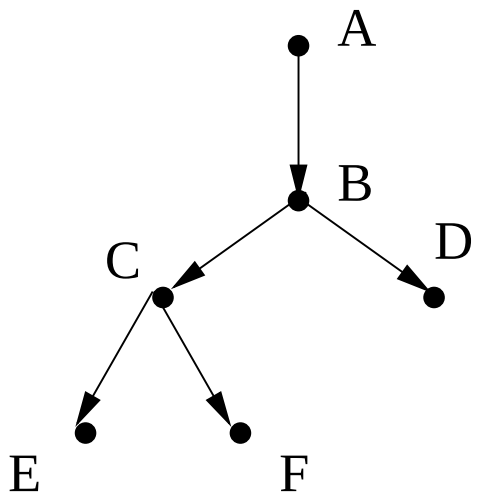
gilt:

- Knoten A hat Tiefe

Beispiele

Im Graphen

$G_A =$

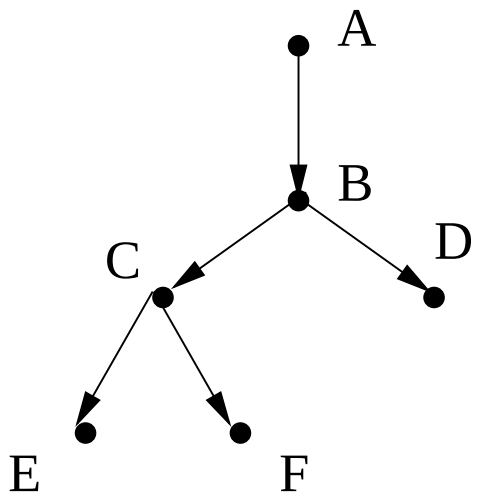


gilt:

- Knoten A hat Tiefe 0, Höhe

Beispiele

Im Graphen

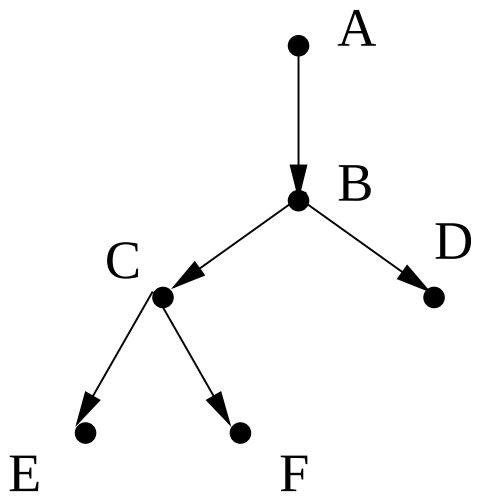
$G_A =$ 

gilt:

- Knoten A hat Tiefe 0, Höhe 3 und A hat genau ein Kind, nämlich B;
- Knoten B hat Tiefe

Beispiele

Im Graphen

$G_A =$ 

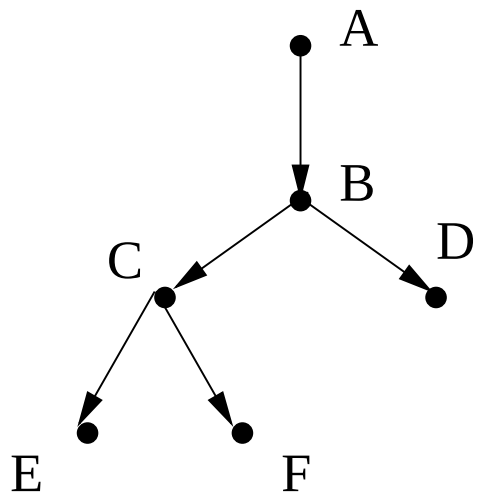
gilt:

- Knoten A hat Tiefe 0, Höhe 3 und A hat genau ein Kind, nämlich B;
- Knoten B hat Tiefe 1, Höhe

Beispiele

Im Graphen

$G_A =$

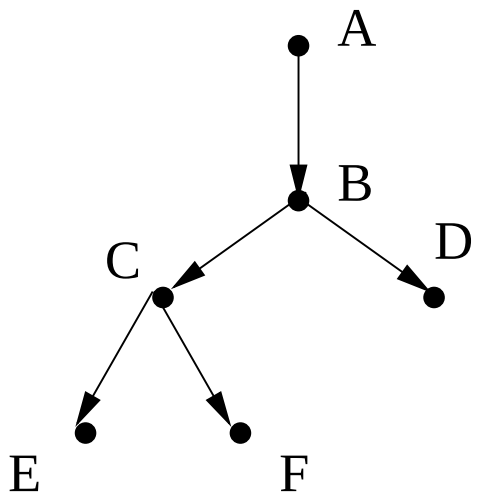


gilt:

- Knoten A hat Tiefe 0, Höhe 3 und A hat genau ein Kind, nämlich B;
- Knoten B hat Tiefe 1, Höhe 2 und B hat genau zwei Kinder, nämlich C und D;
- Knoten C hat Tiefe

Beispiele

Im Graphen

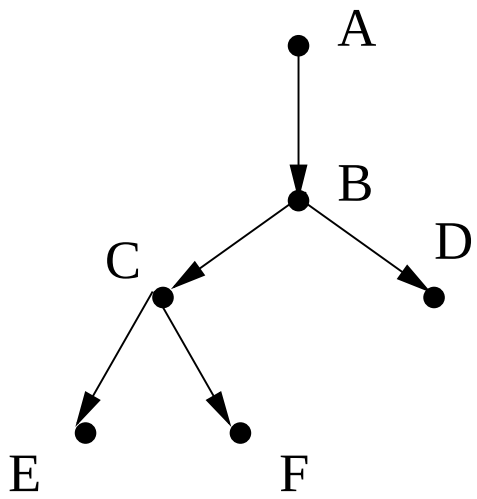
$G_A =$ 

gilt:

- Knoten A hat Tiefe 0, Höhe 3 und A hat genau ein Kind, nämlich B;
- Knoten B hat Tiefe 1, Höhe 2 und B hat genau zwei Kinder, nämlich C und D;
- Knoten C hat Tiefe 2, Höhe

Beispiele

Im Graphen

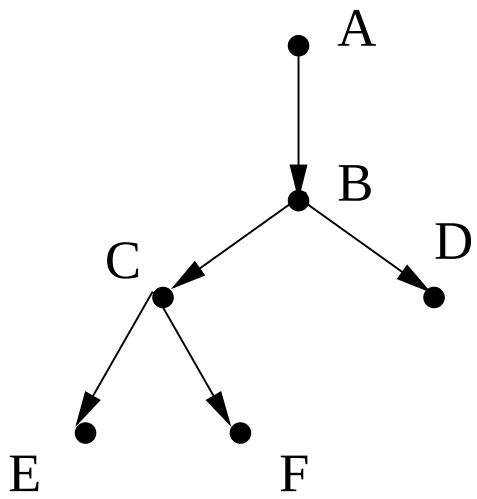
$G_A =$ 

gilt:

- Knoten A hat Tiefe 0, Höhe 3 und A hat genau ein Kind, nämlich B;
- Knoten B hat Tiefe 1, Höhe 2 und B hat genau zwei Kinder, nämlich C und D;
- Knoten C hat Tiefe 2, Höhe 1 und C hat genau zwei Kinder, nämlich C und F;
- und die Knoten D (Tiefe

Beispiele

Im Graphen

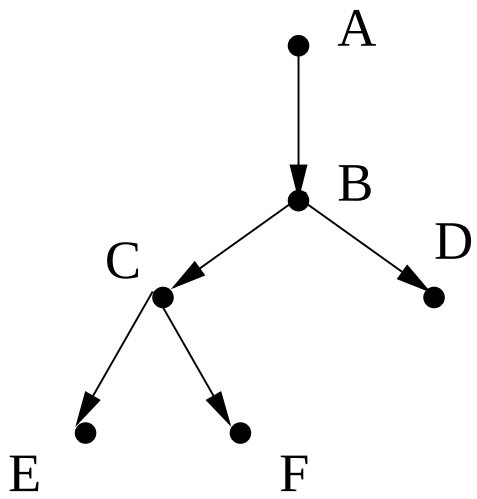
$G_A =$ 

gilt:

- Knoten A hat Tiefe 0, Höhe 3 und A hat genau ein Kind, nämlich B;
- Knoten B hat Tiefe 1, Höhe 2 und B hat genau zwei Kinder, nämlich C und D;
- Knoten C hat Tiefe 2, Höhe 1 und C hat genau zwei Kinder, nämlich C und F;
- und die Knoten D (Tiefe 2), E (Tiefe

Beispiele

Im Graphen

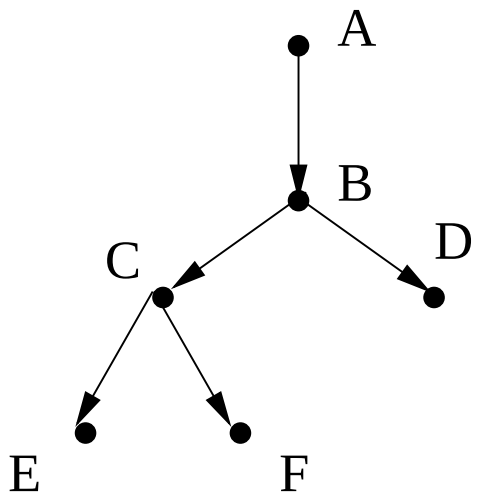
$G_A =$ 

gilt:

- Knoten A hat Tiefe 0, Höhe 3 und A hat genau ein Kind, nämlich B;
- Knoten B hat Tiefe 1, Höhe 2 und B hat genau zwei Kinder, nämlich C und D;
- Knoten C hat Tiefe 2, Höhe 1 und C hat genau zwei Kinder, nämlich C und F;
- und die Knoten D (Tiefe 2), E (Tiefe 3), F (Tiefe

Beispiele

Im Graphen

$G_A =$ 

gilt:

- Knoten A hat Tiefe 0, Höhe 3 und A hat genau ein Kind, nämlich B;
- Knoten B hat Tiefe 1, Höhe 2 und B hat genau zwei Kinder, nämlich C und D;
- Knoten C hat Tiefe 2, Höhe 1 und C hat genau zwei Kinder, nämlich C und F;
- und die Knoten D (Tiefe 2), E (Tiefe 3), F (Tiefe 3) haben keine Kinder.
  Als Blätter haben alle drei die Höhe 0.

Binärbäume

Binärbäume

(a) Ein gewurzelter Baum $B = (V, E)$ heißt **Binärbaum**, falls für jeden Knoten $v \in V$ gilt:

$$\text{Aus-Grad}_B(v) \leqslant 2.$$

Binärbäume

(a) Ein gewurzelter Baum $B = (V, E)$ heißt **Binärbaum**, falls für jeden Knoten $v \in V$ gilt:

$$\text{Aus-Grad}_B(v) \leqslant 2.$$

(b) Ein Binärbaum $B = (V, E)$ heißt **vollständiger Binärbaum**, falls gilt:

(1) Es ist

$$\text{Aus-Grad}(v) = 2$$

für jeden Knoten v, der kein Blatt ist.

Binärbäume

(a) Ein gewurzelter Baum $B = (V, E)$ heißt **Binärbaum**, falls für jeden Knoten $v \in V$ gilt:

$$\text{Aus-Grad}_B(v) \leqslant 2.$$

(b) Ein Binärbaum $B = (V, E)$ heißt **vollständiger Binärbaum**, falls gilt:

(1) Es ist

$$\text{Aus-Grad}(v) = 2$$

für jeden Knoten v, der kein Blatt ist.

(2) Alle Blätter von B haben dieselbe Tiefe.

Binärbäume

(a) Ein gewurzelter Baum $B = (V, E)$ heißt **Binärbaum**, falls für jeden Knoten $v \in V$ gilt:

$$\text{Aus-Grad}_B(v) \leqslant 2.$$

(b) Ein Binärbaum $B = (V, E)$ heißt **vollständiger Binärbaum**, falls gilt:

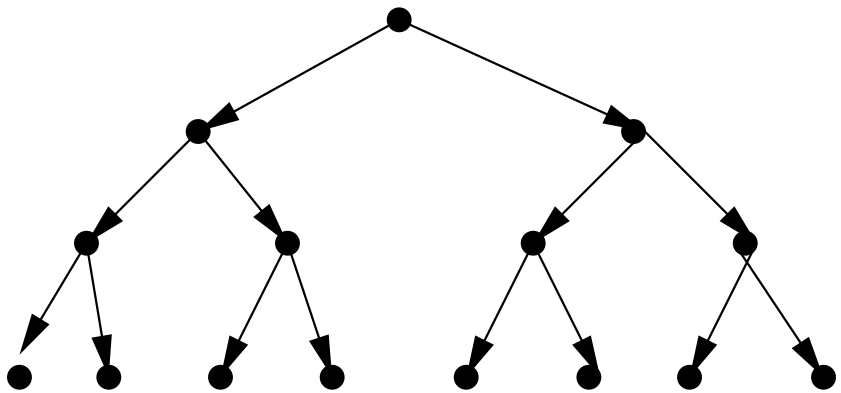
(1) Es ist

$$\text{Aus-Grad}(v) = 2$$

für jeden Knoten v, der kein Blatt ist.

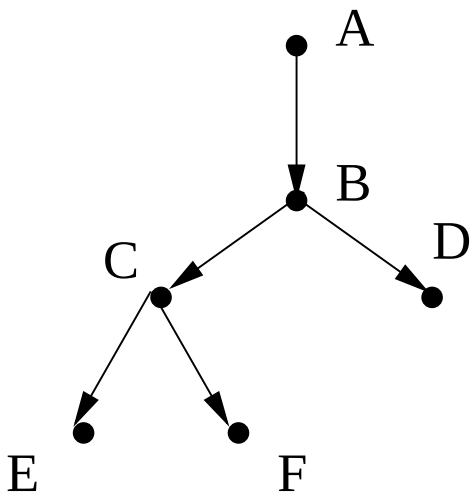
(2) Alle Blätter von B haben dieselbe Tiefe.

Der folgende Graph B_3 ist ein **vollständiger Binärbaum** der Höhe 3:

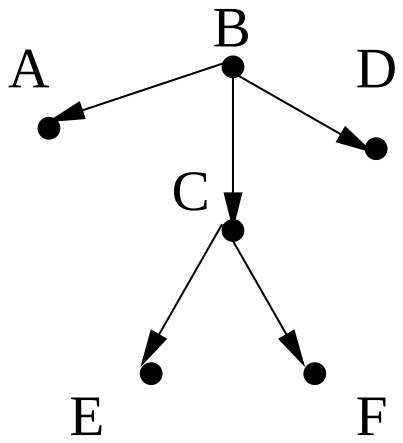


Binärbäume: Beispiele

$G_A =$

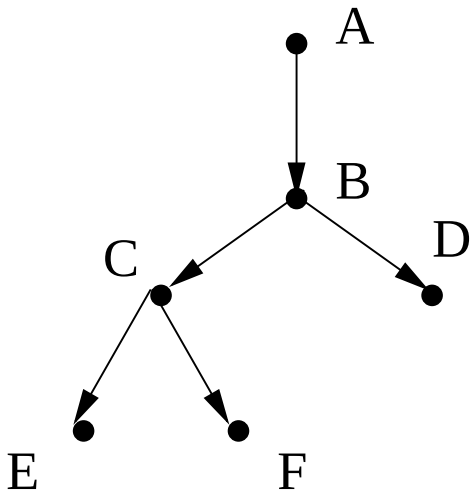


$G_B =$

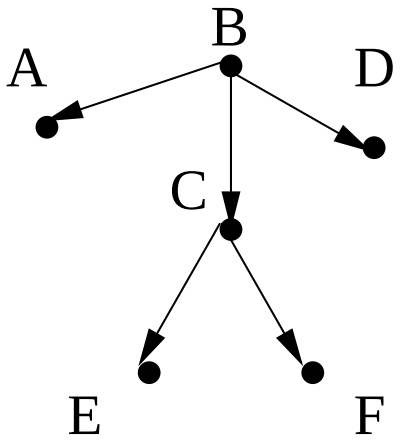


Binärbäume: Beispiele

$G_A =$



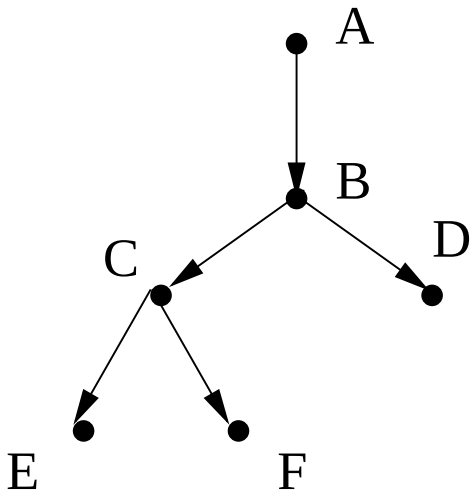
$G_B =$



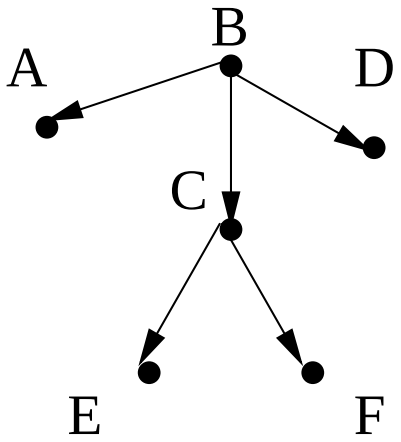
Nur G_A ist binär. Beachte, dass beide Bäume vom selben ungerichteten Baum „abstammen“, nämlich von

Binärbäume: Beispiele

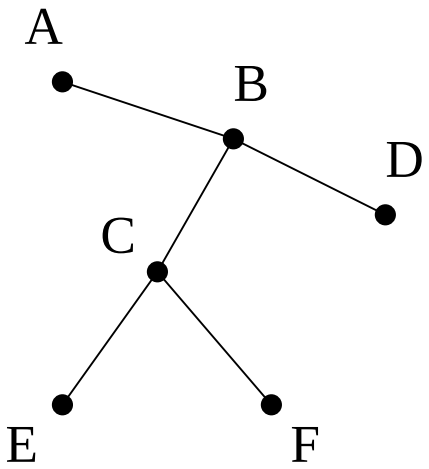
$G_A =$



$G_B =$



Nur G_A ist binär. Beachte, dass beide Bäume vom selben ungerichteten Baum „abstammen“, nämlich von



Wieviele Blätter hat ein Binärbaum der Höhe h? (1/2)

Sei $h \in \mathbb{N}$.

(a) Jeder **vollständige Binärbaum der Höhe h** hat genau 2^h Blätter und genau $2^{h+1} - 1$ Knoten.

(b) Jeder **Binärbaum der Höhe h** hat höchstens 2^h Blätter und höchstens $2^{h+1} - 1$ Knoten.

Wieviele Blätter hat ein Binärbaum der Höhe h? (1/2)

Sei $h \in \mathbb{N}$.

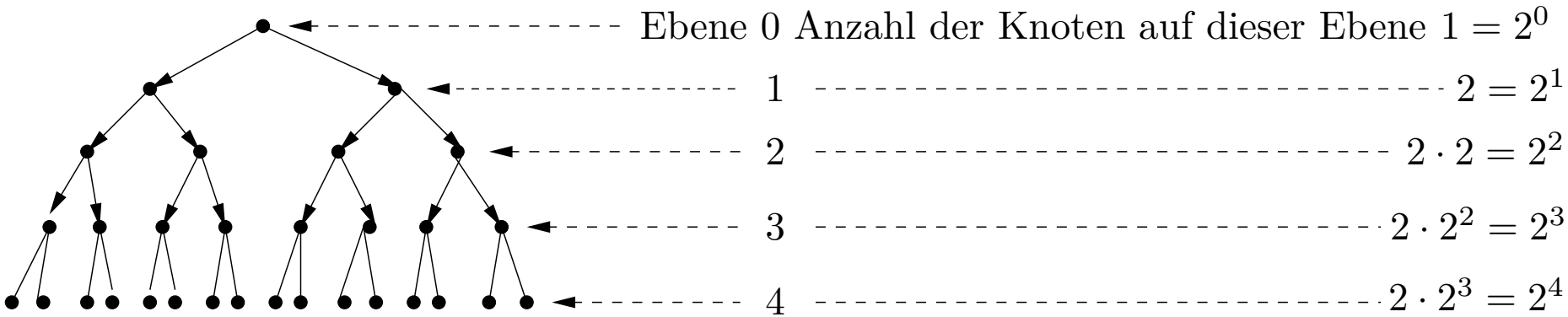
(a) Jeder **vollständige Binärbaum der Höhe h** hat genau 2^h Blätter und genau $2^{h+1} - 1$ Knoten.

(b) Jeder **Binärbaum der Höhe h** hat höchstens 2^h Blätter und höchstens $2^{h+1} - 1$ Knoten.

Intuition: Nach der Skizze sieht es so aus, dass ein vollständiger Binärbaum der Höhe h genau 2^t Knoten der Tiefe t (für $t \leqslant h$) besitzt und deshalb gibt es genau

$$2^0 + 2^1 + 2^2 + \cdots + 2^h \;=\; 2^{h+1} - 1$$

Knoten.



Wieviele Blätter hat ein Binärbaum der Höhe h? (2/2)

Aber wie formalisiert man die Intuition?

Wieviele Blätter hat ein Binärbaum der Höhe h? (2/2)

Aber wie formalisiert man die Intuition?

Sei B ein vollständiger Binärbaum der *Höhe* h. Wir bestimmen die Anzahl der Knoten der *Tiefe* t in B mit vollständiger Induktion nach t für alle $t \leqslant h$.

1. Wir beobachten, dass es (im Vergleich zur Anzahl der Knoten der Tiefe t), **genau** doppelt so viele Knoten der Tiefe $t+1$ gibt, falls $t+1 \leqslant h$.

Wieviele Blätter hat ein Binärbaum der Höhe h? (2/2)

Aber wie formalisiert man die Intuition?

Sei B ein vollständiger Binärbaum der *Höhe* h. Wir bestimmen die Anzahl der Knoten der *Tiefe* t in B mit vollständiger Induktion nach t für alle $t \leqslant h$.

1. Wir beobachten, dass es (im Vergleich zur Anzahl der Knoten der Tiefe t), **genau** doppelt so viele Knoten der Tiefe $t+1$ gibt, falls $t+1 \leqslant h$.
   - Wenn a_t die Anzahl der Knoten der Tiefe t ist, dann ist

$$a_0 = 1 \text{ und } a_{t+1} = 2a_t$$

Wieviele Blätter hat ein Binärbaum der Höhe h? (2/2)

Aber wie formalisiert man die Intuition?

Sei B ein vollständiger Binärbaum der *Höhe* h. Wir bestimmen die Anzahl der Knoten der *Tiefe* t in B mit vollständiger Induktion nach t für alle $t \leqslant h$.

1. Wir beobachten, dass es (im Vergleich zur Anzahl der Knoten der Tiefe t), **genau** doppelt so viele Knoten der Tiefe $t+1$ gibt, falls $t+1 \leqslant h$.
   - Wenn a_t die Anzahl der Knoten der Tiefe t ist, dann ist

$$a_0 = 1 \text{ und } a_{t+1} = 2a_t$$

2. Sodann zeige mit einer vollständigen Induktion nach t, dass $a_t = 2^t$ gilt.

Wieviele Blätter hat ein Binärbaum der Höhe h? (2/2)

Aber wie formalisiert man die Intuition?

Sei B ein vollständiger Binärbaum der *Höhe* h. Wir bestimmen die Anzahl der Knoten der *Tiefe* t in B mit vollständiger Induktion nach t für alle $t \leqslant h$.

1. Wir beobachten, dass es (im Vergleich zur Anzahl der Knoten der Tiefe t), **genau** doppelt so viele Knoten der Tiefe $t+1$ gibt, falls $t+1 \leqslant h$.
   - Wenn a_t die Anzahl der Knoten der Tiefe t ist, dann ist

$$a_0 = 1 \text{ und } a_{t+1} = 2a_t$$

2. Sodann zeige mit einer vollständigen Induktion nach t, dass $a_t = 2^t$ gilt.
3. B hat 2^h Blätter und $2^0 + 2^1 + \cdots + 2^h = 2^{h+1} - 1$ Knoten.

Wieviele Blätter hat ein Binärbaum der Höhe h? (2/2)

Aber wie formalisiert man die Intuition?

Sei B ein vollständiger Binärbaum der *Höhe* h. Wir bestimmen die Anzahl der Knoten der *Tiefe* t in B mit vollständiger Induktion nach t für alle $t \leqslant h$.

1. Wir beobachten, dass es (im Vergleich zur Anzahl der Knoten der Tiefe t), **genau** doppelt so viele Knoten der Tiefe $t+1$ gibt, falls $t+1 \leqslant h$.
   - Wenn a_t die Anzahl der Knoten der Tiefe t ist, dann ist

$$a_0 = 1 \text{ und } a_{t+1} = 2a_t$$

2. Sodann zeige mit einer vollständigen Induktion nach t, dass $a_t = 2^t$ gilt.
3. B hat 2^h Blätter und $2^0 + 2^1 + \cdots + 2^h = 2^{h+1} - 1$ Knoten.

Aber warum hat jeder Binärbaum der Höhe h höchstens 2^h Blätter und höchstens $2^{h+1} - 1$ Knoten?

Unter allen Binärbäumen der Tiefe h hat der vollständige Binärbaum die meisten Blätter und Knoten.

Modellierungsbeispiele für gewurzelte Bäume mit Knoten-/Kantenmarkierungen

Modellierung mit Bäumen

Gewurzelte Bäume werden eingesetzt, um

- **Entwicklungen** (bzw. Generationen) zu zeigen:
  - Stammbäumen (in der Genealogie) geben die Familiengeschichte wieder,
  - phylogenetische Bäume (in der Bioinformatik) zeigen evolutionäre Beziehungen zwischen verschiedenen Arten auf.

Modellierung mit Bäumen

Gewurzelte Bäume werden eingesetzt, um

- **Entwicklungen** (bzw. Generationen) zu zeigen:
  - Stammbäumen (in der Genealogie) geben die Familiengeschichte wieder,
  - phylogenetische Bäume (in der Bioinformatik) zeigen evolutionäre Beziehungen zwischen verschiedenen Arten auf.
- **hierarchische Strukturen** zu repräsentieren wie etwa
  - eine Verzeichnisstruktur oder die
  - Organisationsstrukur einer Firma.

Modellierung mit Bäumen

Gewurzelte Bäume werden eingesetzt, um

- **Entwicklungen** (bzw. Generationen) zu zeigen:
  - Stammbäumen (in der Genealogie) geben die Familiengeschichte wieder,
  - phylogenetische Bäume (in der Bioinformatik) zeigen evolutionäre Beziehungen zwischen verschiedenen Arten auf.
- **hierarchische Strukturen** zu repräsentieren wie etwa
  - eine Verzeichnisstruktur oder die
  - Organisationsstrukur einer Firma.
- **Datenstrukturen** zu entwickeln. (Siehe die gleichnamige Veranstaltung.)
  - Beispiele sind binäre Suchbäume, AVL-Bäume oder B-Bäume.

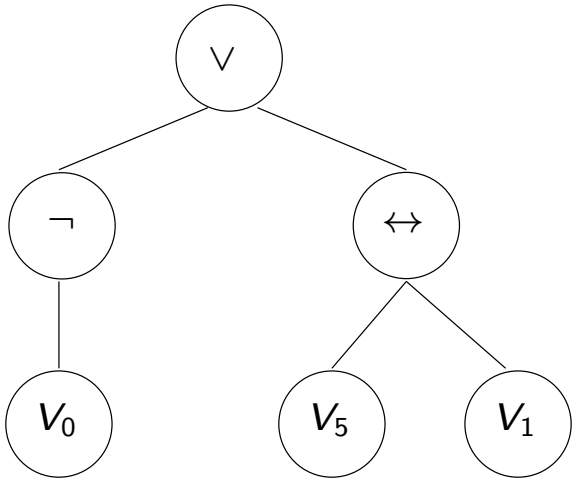
Modellierung mit Bäumen

Gewurzelte Bäume werden eingesetzt, um

- **Entwicklungen** (bzw. Generationen) zu zeigen:
  - Stammbäumen (in der Genealogie) geben die Familiengeschichte wieder,
  - phylogenetische Bäume (in der Bioinformatik) zeigen evolutionäre Beziehungen zwischen verschiedenen Arten auf.
- **hierarchische Strukturen** zu repräsentieren wie etwa
  - eine Verzeichnisstruktur oder die
  - Organisationsstrukur einer Firma.
- **Datenstrukturen** zu entwickeln. (Siehe die gleichnamige Veranstaltung.)
  - Beispiele sind binäre Suchbäume, AVL-Bäume oder B-Bäume.
- **rekursive Definitionen** zu veranschaulichen, bzw. zu verstehen. Beispiele sind rekursiv definierte Mengen oder Funktionen wie etwa:
  - (aussagenlogische) Formeln, arithmetische Ausdrücke, XML-Dokumente oder die Syntax einer Programmiersprache,
  - rekursive Programme.

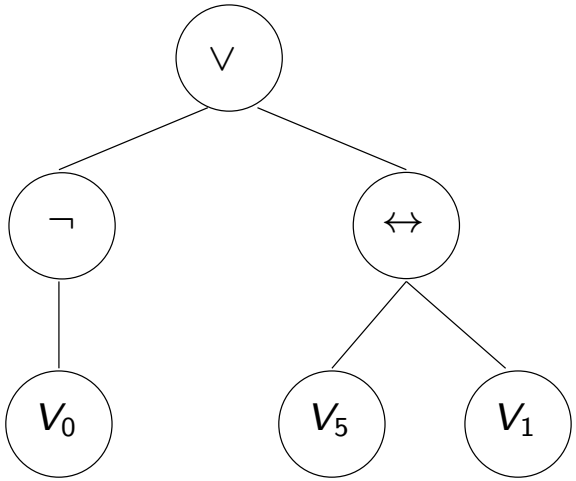
Die Syntax aussagenlogischer Formeln

Die Bedeutung der *aussagenlogischen Formel* $\big(\neg V_0 \vee (V_5 \leftrightarrow V_1)\big)$ wird sofort klar, wenn wir ihren **Syntaxbaum** betrachten:



Die Syntax aussagenlogischer Formeln

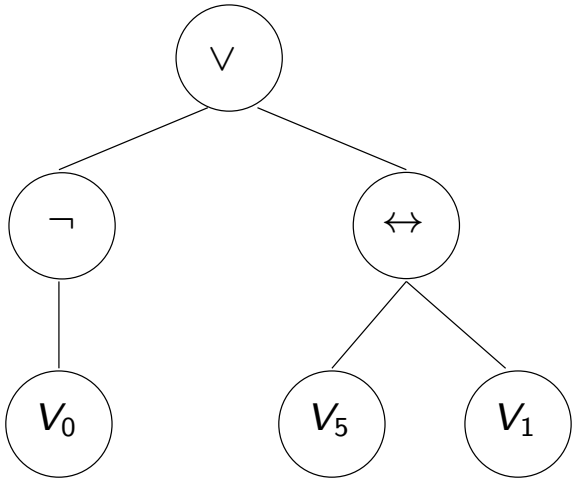
Die Bedeutung der *aussagenlogischen Formel* $\big(\neg V_0 \vee (V_5 \leftrightarrow V_1)\big)$ wird sofort klar, wenn wir ihren **Syntaxbaum** betrachten:



Für aussagenlogische Formeln ist der Syntaxbaum stets binär. Warum?

Die Syntax aussagenlogischer Formeln

Die Bedeutung der *aussagenlogischen Formel* $\big(\neg V_0 \vee (V_5 \leftrightarrow V_1)\big)$ wird sofort klar, wenn wir ihren **Syntaxbaum** betrachten:



Für aussagenlogische Formeln ist der Syntaxbaum stets binär. Warum?

Syntaxbäume treten auch bei der Compilierung auf.

Um ein *Programm* auszuführen, wird der Compiler den **Syntaxbaum** des Programms erzeugt. (Mehr Details im Kapitel über „kontextfreie Grammatiken“.)

- Auch diesmal ist der Syntaxbaum gewurzelt, aber nicht mehr binär.

Syntaxbaum $\leftrightarrow$ Formel

Wir haben aussagenlogische Formeln durch

eine rekursive Definition über den Aufbau der Formel

hergeleitet.

Syntaxbaum $\leftrightarrow$ Formel

Wir haben aussagenlogische Formeln durch

eine rekursive Definition über den Aufbau der Formel

hergeleitet.

(a) Der Syntaxbaum bildet die Rekursion nach.

(b) Die inneren Knoten wie auch die Wurzel sind jeweils mit einem Junktor **markiert**, die Blätter sind mit Variablen **markiert**.

Rekursive Programme und ihre Rekursionsbäume

Ein rekursives Programm $R(\vec{p})$ (mit den Parametern $\vec{p}$ und einer festgelegten Eingabe) kann mit Hilfe seines Rekursionsbaums untersucht werden.

Rekursive Programme und ihre Rekursionsbäume

Ein rekursives Programm $R(\vec{p})$ (mit den Parametern $\vec{p}$ und einer festgelegten Eingabe) kann mit Hilfe seines Rekursionsbaums untersucht werden.

1. Beschrifte die Wurzel von B mit den Parametern $\vec{p}$ des Erstaufrufs.
   - Wenn innerhalb von $R(\vec{p})$ keine rekursiven Aufrufe getätigt werden, dann wird die Wurzel zu einem Blatt.
   - Ansonsten erhält die Wurzel für jeden rekursiven Aufruf innerhalb von $R(\vec{p})$ ein neues Kind, das mit den Parametern des rekursiven Aufrufs beschriftet wird.

Rekursive Programme und ihre Rekursionsbäume

Ein rekursives Programm $R(\vec{p})$ (mit den Parametern $\vec{p}$ und einer festgelegten Eingabe) kann mit Hilfe seines Rekursionsbaums untersucht werden.

1. Beschrifte die Wurzel von B mit den Parametern $\vec{p}$ des Erstaufrufs.
   - Wenn innerhalb von $R(\vec{p})$ keine rekursiven Aufrufe getätigt werden, dann wird die Wurzel zu einem Blatt.
   - Ansonsten erhält die Wurzel für jeden rekursiven Aufruf innerhalb von $R(\vec{p})$ ein neues Kind, das mit den Parametern des rekursiven Aufrufs beschriftet wird.
2. Wenn ein Knoten v von B mit den Parametern $\vec{q}$ beschriftet ist, gehen wir mit v genauso wie mit der Wurzel um.
   - Wenn innerhalb von $R(\vec{q})$ keine rekursiven Aufrufe getätigt werden, dann wird v zu einem Blatt.
   - Ansonsten erhält v für jeden rekursiven Aufruf innerhalb von $R(\vec{q})$ ein neues Kind, das mit den Parametern des rekursiven Aufrufs beschriftet wird.

Rekursive Programme und ihre Rekursionsbäume

Ein rekursives Programm $R(\vec{p})$ (mit den Parametern $\vec{p}$ und einer festgelegten Eingabe) kann mit Hilfe seines Rekursionsbaums untersucht werden.

1. Beschrifte die Wurzel von B mit den Parametern $\vec{p}$ des Erstaufrufs.
   - Wenn innerhalb von $R(\vec{p})$ keine rekursiven Aufrufe getätigt werden, dann wird die Wurzel zu einem Blatt.
   - Ansonsten erhält die Wurzel für jeden rekursiven Aufruf innerhalb von $R(\vec{p})$ ein neues Kind, das mit den Parametern des rekursiven Aufrufs beschriftet wird.
2. Wenn ein Knoten v von B mit den Parametern $\vec{q}$ beschriftet ist, gehen wir mit v genauso wie mit der Wurzel um.
   - Wenn innerhalb von $R(\vec{q})$ keine rekursiven Aufrufe getätigt werden, dann wird v zu einem Blatt.
   - Ansonsten erhält v für jeden rekursiven Aufruf innerhalb von $R(\vec{q})$ ein neues Kind, das mit den Parametern des rekursiven Aufrufs beschriftet wird.

Z. B. stimmt die Anzahl der Knoten von B mit der Anzahl aller rekursiven Aufrufe (fÃ¼r die zu Anfang festgelegte Eingabe) überein.

Das Array A sei aufsteigend sortiert.

Wir möchten überprüfen, ob der „Schlüssel“ x in A vorkommt.

Binärsuche: Die Mutter aller Algorithmen (1/2)

```
void Suche( int unten, int oben){
   if (oben < unten)
     std::cout « x « " wurde nicht gefunden."
      « std::endl;
```

Binärsuche: Die Mutter aller Algorithmen (1/2)

```
void Suche( int unten, int oben){
   if (oben < unten)
     std::cout « x « " wurde nicht gefunden."
      « std::endl;
   int mitte = (unten+oben)/2;
```

Binärsuche: Die Mutter aller Algorithmen (1/2)

```
void Suche( int unten, int oben){
   if (oben < unten)
     std::cout « x « " wurde nicht gefunden."
      « std::endl;
   int mitte = (unten+oben)/2;
   if (A[mitte] == x)
```

Binärsuche: Die Mutter aller Algorithmen (1/2)

```
void Suche( int unten, int oben){
   if (oben < unten)
     std::cout « x « " wurde nicht gefunden."
      « std::endl;
   int mitte = (unten+oben)/2;
   if (A[mitte] == x)
     std::cout « x « " wurde in Position "
      « mitte « " gefunden." « std::endl;
```

Binärsuche: Die Mutter aller Algorithmen (1/2)

```
void Suche( int unten, int oben){
   if (oben < unten)
     std::cout « x « " wurde nicht gefunden."
      « std::endl;
   int mitte = (unten+oben)/2;
   if (A[mitte] == x)
     std::cout « x « " wurde in Position "
      « mitte « " gefunden." « std::endl;
     else {
```

Binärsuche: Die Mutter aller Algorithmen (1/2)

```
void Suche( int unten, int oben){
   if (oben < unten)
     std::cout « x « " wurde nicht gefunden."
      « std::endl;
   int mitte = (unten+oben)/2;
   if (A[mitte] == x)
     std::cout « x « " wurde in Position "
      « mitte « " gefunden." « std::endl;
     else {
       if (x < A[mitte])
         Suche(unten, mitte-1);
```

Binärsuche: Die Mutter aller Algorithmen (1/2)

```
void Suche( int unten, int oben){
   if (oben < unten)
     std::cout « x « " wurde nicht gefunden."
      « std::endl;
   int mitte = (unten+oben)/2;
   if (A[mitte] == x)
     std::cout « x « " wurde in Position "
      « mitte « " gefunden." « std::endl;
     else {
       if (x < A[mitte])
         Suche(unten, mitte-1);
       else
         Suche(mitte+1,oben);}}
```

? Wie sieht der Rekursionsbaum B für ein Array A und einen Schlüssel x aus, wenn x kleiner als der kleinste Schlüssel von A ist?

Binärsuche: Die Mutter aller Algorithmen (1/2)

```
void Suche( int unten, int oben){
   if (oben < unten)
     std::cout « x « " wurde nicht gefunden."
      « std::endl;
   int mitte = (unten+oben)/2;
   if (A[mitte] == x)
     std::cout « x « " wurde in Position "
      « mitte « " gefunden." « std::endl;
     else {
       if (x < A[mitte])
         Suche(unten, mitte-1);
       else
         Suche(mitte+1,oben);}}
```

? Wie sieht der Rekursionsbaum B für ein Array A und einen Schlüssel x aus, wenn x kleiner als der kleinste Schlüssel von A ist?

? Es sei $n = \mathit{oben} - \mathit{unten} + 1$ mit $n = 2^k - 1$ für eine Zahl $k \in \mathbb{N}$. Wie groß ist die Höhe von B? Wieviele rekursive Aufrufe $T(n)$ verursacht Binärsuche also?

Binärsuche: Die Mutter aller Algorithmen (2/2)

Es ist:

Rekursionsanfang: $H(0) =$

Binärsuche: Die Mutter aller Algorithmen (2/2)

Es ist:

Rekursionsanfang: $H(0) = 0$.

Rekursionsschritt: $H(n) =$

Binärsuche: Die Mutter aller Algorithmen (2/2)

Es ist:

Rekursionsanfang: $H(0) = 0$.

Rekursionsschritt: $H(n) = H(\frac{n-1}{2}) + 1$.

Binärsuche: Die Mutter aller Algorithmen (2/2)

Es ist:

Rekursionsanfang: $H(0) = 0$.

Rekursionsschritt: $H(n) = H(\frac{n-1}{2}) + 1$.

- Wir haben $n = 2^k - 1$ gefordert. Beachte, dass $\frac{n-1}{2} = \frac{2^k-2}{2} = 2^{k-1} - 1$ gilt.

Binärsuche: Die Mutter aller Algorithmen (2/2)

Es ist:

Rekursionsanfang: $H(0) = 0$.

Rekursionsschritt: $H(n) = H(\frac{n-1}{2}) + 1$.

- Wir haben $n = 2^k - 1$ gefordert. Beachte, dass $\frac{n-1}{2} = \frac{2^k-2}{2} = 2^{k-1} - 1$ gilt.
- Nach jedem rekursiven Aufruf wird der Exponent k also um 1 erniedrigt.

Binärsuche: Die Mutter aller Algorithmen (2/2)

Es ist:

Rekursionsanfang: $H(0) = 0$.

Rekursionsschritt: $H(n) = H(\frac{n-1}{2}) + 1$.

- Wir haben $n = 2^k - 1$ gefordert. Beachte, dass $\frac{n-1}{2} = \frac{2^k-2}{2} = 2^{k-1} - 1$ gilt.
- Nach jedem rekursiven Aufruf wird der Exponent k also um 1 erniedrigt.
  - Für $n = 2^k - 1$ „sollte“ es höchstens k rekursive Aufrufe geben
  - und $H(2^k - 1) = k$ „sollte“ folgen.

Binärsuche: Die Mutter aller Algorithmen (2/2)

Es ist:

Rekursionsanfang: $H(0) = 0$.

Rekursionsschritt: $H(n) = H(\frac{n-1}{2}) + 1$.

- Wir haben $n = 2^k - 1$ gefordert. Beachte, dass $\frac{n-1}{2} = \frac{2^k-2}{2} = 2^{k-1} - 1$ gilt.
- Nach jedem rekursiven Aufruf wird der Exponent k also um 1 erniedrigt.
  - Für $n = 2^k - 1$ „sollte“ es höchstens k rekursive Aufrufe geben
  - und $H(2^k - 1) = k$ „sollte“ folgen.
- Wir zeigen $H(2^k - 1) = k$ mit vollständiger Induktion nach k.
  - **Induktionsanfang:** $H(2^0 - 1) =$

Binärsuche: Die Mutter aller Algorithmen (2/2)

Es ist:

Rekursionsanfang: $H(0) = 0$.

Rekursionsschritt: $H(n) = H(\frac{n-1}{2}) + 1$.

- Wir haben $n = 2^k - 1$ gefordert. Beachte, dass $\frac{n-1}{2} = \frac{2^k-2}{2} = 2^{k-1} - 1$ gilt.
- Nach jedem rekursiven Aufruf wird der Exponent k also um 1 erniedrigt.
  - Für $n = 2^k - 1$ „sollte“ es höchstens k rekursive Aufrufe geben
  - und $H(2^k - 1) = k$ „sollte“ folgen.
- Wir zeigen $H(2^k - 1) = k$ mit vollständiger Induktion nach k.
  - **Induktionsanfang:** $H(2^0 - 1) = H(0) =$

Binärsuche: Die Mutter aller Algorithmen (2/2)

Es ist:

Rekursionsanfang: $H(0) = 0$.

Rekursionsschritt: $H(n) = H(\frac{n-1}{2}) + 1$.

- Wir haben $n = 2^k - 1$ gefordert. Beachte, dass $\frac{n-1}{2} = \frac{2^k-2}{2} = 2^{k-1} - 1$ gilt.
- Nach jedem rekursiven Aufruf wird der Exponent k also um 1 erniedrigt.
  - Für $n = 2^k - 1$ „sollte“ es höchstens k rekursive Aufrufe geben
  - und $H(2^k - 1) = k$ „sollte“ folgen.
- Wir zeigen $H(2^k - 1) = k$ mit vollständiger Induktion nach k.
  - **Induktionsanfang:** $H(2^0 - 1) = H(0) = 0$. ✓
  - **Induktionsschritt:** $H(2^{k+1} - 1) =$

Binärsuche: Die Mutter aller Algorithmen (2/2)

Es ist:

Rekursionsanfang: $H(0) = 0$.

Rekursionsschritt: $H(n) = H(\frac{n-1}{2}) + 1$.

- Wir haben $n = 2^k - 1$ gefordert. Beachte, dass $\frac{n-1}{2} = \frac{2^k-2}{2} = 2^{k-1} - 1$ gilt.
- Nach jedem rekursiven Aufruf wird der Exponent k also um 1 erniedrigt.
  - Für $n = 2^k - 1$ „sollte“ es höchstens k rekursive Aufrufe geben
  - und $H(2^k - 1) = k$ „sollte“ folgen.
- Wir zeigen $H(2^k - 1) = k$ mit vollständiger Induktion nach k.
  - **Induktionsanfang:** $H(2^0 - 1) = H(0) = 0$. ✓
  - **Induktionsschritt:** $H(2^{k+1} - 1) = H(2^k - 1) + 1$

Binärsuche: Die Mutter aller Algorithmen (2/2)

Es ist:

Rekursionsanfang: $H(0) = 0$.

Rekursionsschritt: $H(n) = H(\frac{n-1}{2}) + 1$.

- Wir haben $n = 2^k - 1$ gefordert. Beachte, dass $\frac{n-1}{2} = \frac{2^k-2}{2} = 2^{k-1} - 1$ gilt.
- Nach jedem rekursiven Aufruf wird der Exponent k also um 1 erniedrigt.
  - Für $n = 2^k - 1$ „sollte“ es höchstens k rekursive Aufrufe geben
  - und $H(2^k - 1) = k$ „sollte“ folgen.
- Wir zeigen $H(2^k - 1) = k$ mit vollständiger Induktion nach k.
  - **Induktionsanfang:** $H(2^0 - 1) = H(0) = 0$. ✓
  - **Induktionsschritt:** $H(2^{k+1} - 1) = H(2^k - 1) + 1 \overset{\text{Induktionsannahme}}{=} k + 1$. ✓

Binärsuche: Die Mutter aller Algorithmen (2/2)

Es ist:

Rekursionsanfang: $H(0) = 0$.

Rekursionsschritt: $H(n) = H(\frac{n-1}{2}) + 1$.

- Wir haben $n = 2^k - 1$ gefordert. Beachte, dass $\frac{n-1}{2} = \frac{2^k-2}{2} = 2^{k-1} - 1$ gilt.
- Nach jedem rekursiven Aufruf wird der Exponent k also um 1 erniedrigt.
  - Für $n = 2^k - 1$ „sollte“ es höchstens k rekursive Aufrufe geben
  - und $H(2^k - 1) = k$ „sollte“ folgen.
- Wir zeigen $H(2^k - 1) = k$ mit vollständiger Induktion nach k.
  - **Induktionsanfang:** $H(2^0 - 1) = H(0) = 0$. ✓
  - **Induktionsschritt:** $H(2^{k+1} - 1) = H(2^k - 1) + 1 \overset{\text{Induktionsannahme}}{=} k + 1$. ✓

Und was bedeutet das jetzt?

Im schlimmsten Fall muss Binärsuche muss höchstens $\lceil \log_2 n \rceil$ Zahlen inspizieren, gegenüber n Zahlen für die lineare Suche.

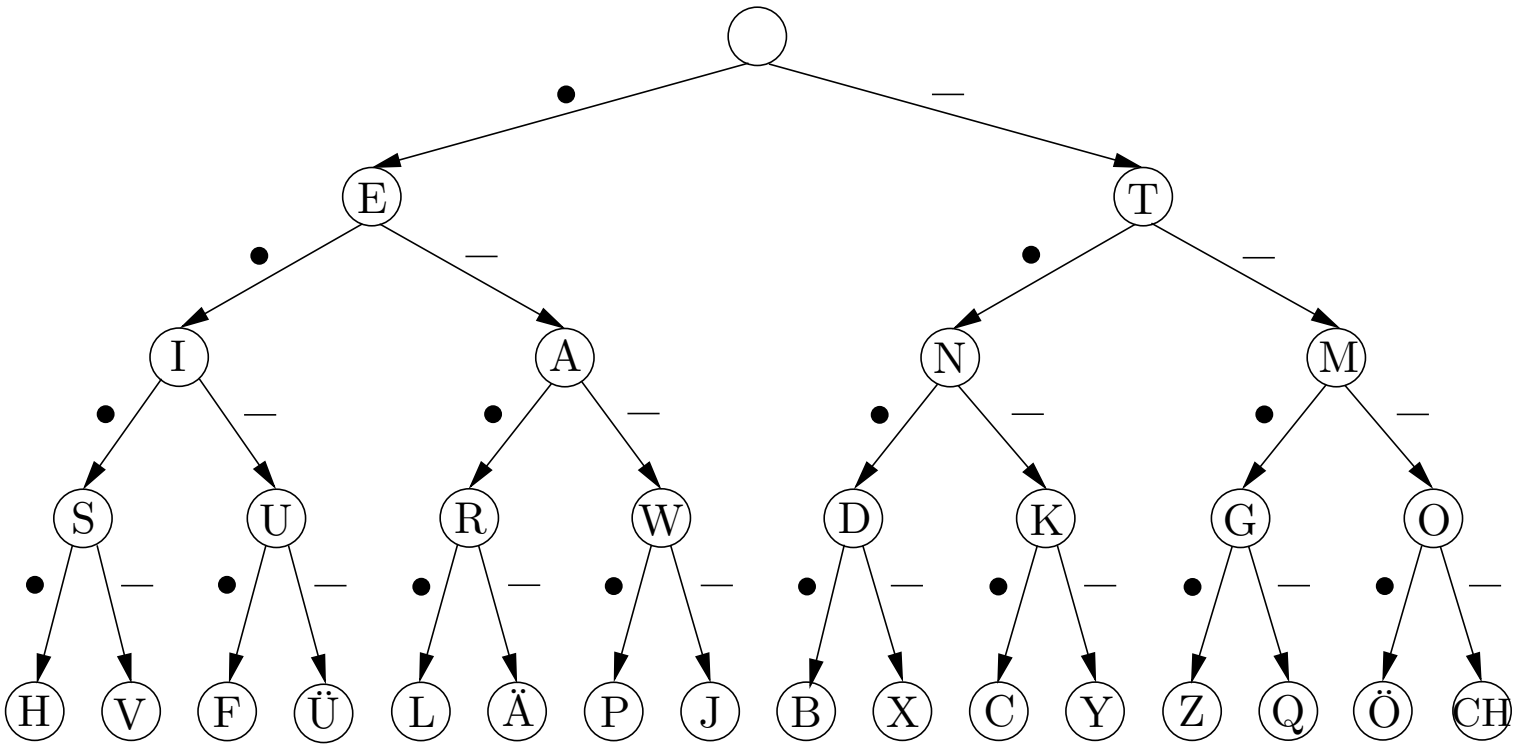
Die lineare Suche ist exponentiell langsamer als Binärsuche.

Entscheidungsbäume

Entscheidungsbäume und der Morse-Code

Folgen von Entscheidungen können häufig durch gewurzelte, markierte Bäume modelliert werden. Diese Bäume nennt man auch **Entscheidungsbäume**.

Der Entscheidungsbaum für den Morse-Code: Im Morse-Code wird jeder Buchstabe durch eine Folge von kurzen und langen Signalen repräsentiert. Ein „kurzes Signal“ wird im folgenden Baum als Kantenmarkierung “•” dargestellt, ein „langes Signal“ als “—” .



Die Struktur von Entscheidungsbäumen

Entscheidungsbäume zählen systematisch alle endgültigen Entscheidungen auf.

(a) Die Knoten modellieren einen **Zwischenstand** (oder eine partielle Entscheidung) bei der Entscheidungsfindung.

- Im Beispiel des Morse-Code werden Knoten mit dem –momentan– kodierten Buchstaben beschriftet.

Die Struktur von Entscheidungsbäumen

Entscheidungsbäume zählen systematisch alle endgültigen Entscheidungen auf.

(a) Die Knoten modellieren einen **Zwischenstand** (oder eine partielle Entscheidung) bei der Entscheidungsfindung.
   - Im Beispiel des Morse-Code werden Knoten mit dem –momentan– kodierten Buchstaben beschriftet.

(b) Die von einem Knoten des Entscheidungsbaums ausgehenden Kanten modellieren die “momentan“ möglichen, einander ausschließenden **Alternativen**.

Die Struktur von Entscheidungsbäumen

Entscheidungsbäume zählen systematisch alle endgültigen Entscheidungen auf.

(a) Die Knoten modellieren einen **Zwischenstand** (oder eine partielle Entscheidung) bei der Entscheidungsfindung.
   - Im Beispiel des Morse-Code werden Knoten mit dem –momentan– kodierten Buchstaben beschriftet.

(b) Die von einem Knoten des Entscheidungsbaums ausgehenden Kanten modellieren die "momentan" möglichen, einander ausschließenden **Alternativen**.

(c) Alle Blätter, möglicherweise aber auch innere Knoten (wie etwa im Beispiel des Morse-Codes) entsprechen **endgültigen Entscheidungen**.

Die Struktur von Entscheidungsbäumen

Entscheidungsbäume zählen systematisch alle endgültigen Entscheidungen auf.

(a) Die Knoten modellieren einen **Zwischenstand** (oder eine partielle Entscheidung) bei der Entscheidungsfindung.
- Im Beispiel des Morse-Code werden Knoten mit dem –momentan– kodierten Buchstaben beschriftet.

(b) Die von einem Knoten des Entscheidungsbaums ausgehenden Kanten modellieren die "momentan" möglichen, einander ausschließenden **Alternativen**.

(c) Alle Blätter, möglicherweise aber auch innere Knoten (wie etwa im Beispiel des Morse-Codes) entsprechen **endgültigen Entscheidungen**.
- Wenn ein Knoten v im Baum einer endgültigen Entscheidung entspricht, dann wird diese Entscheidung durch die Folge der Kantenmarkierungen auf dem Weg von der Wurzel nach v repräsentiert.
  - ⋆ Im Beispiel des Morse-Code: Ist der Knoten v mit einem Buchstaben α markiert, dann beschreibt der Weg von der Wurzel nach α den Morse-Code von α.
- Jede endgültige Entscheidung e entspricht genau einem Weg W_e von der Wurzel. Die Beschriftung des Endknotens von W_e erklärt die Bedeutung von e.

Entscheidungsbäume: Anwendungsbeispiele

(a) In **Optimierungsproblemen der kombinatorischen Optimierung** gibt es viele Lösungen unterschiedlicher Qualität: Gesucht ist eine optimale Lösung.

- Alle „in Frage kommenden“ Lösungen sind **systematisch aufzuzählen**:
  - ⋆ Jede mögliche Lösung L muss irgendeiner endgültigen Entscheidung –und damit einem Knoten v_L im Baum– entsprechen.
  - ⋆ Der Knoten v_L wird mit der Qualität der Lösung L beschriftet.
- Wir schauen uns gleich das Problem des Handlungsreisenden an.

Entscheidungsbäume: Anwendungsbeispiele

(a) In **Optimierungsproblemen der kombinatorischen Optimierung** gibt es viele Lösungen unterschiedlicher Qualität: Gesucht ist eine optimale Lösung.

- Alle „in Frage kommenden“ Lösungen sind **systematisch aufzuzählen**:
  - ⋆ Jede mögliche Lösung L muss irgendeiner endgültigen Entscheidung –und damit einem Knoten v_L im Baum– entsprechen.
  - ⋆ Der Knoten v_L wird mit der Qualität der Lösung L beschriftet.
- Wir schauen uns gleich das Problem des Handlungsreisenden an.

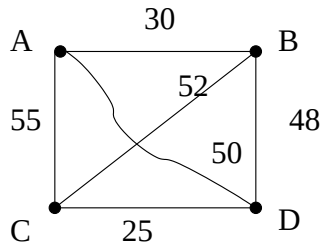
(b) Die in der technischen Informatik verwandten

binären Entscheidungsgraphen

(engl: binary decision diagrams) entstehen, wenn Knoten eines Entscheidungsbaums identifiziert werden.

Das Problem des Handlungsreisenden (1/2)

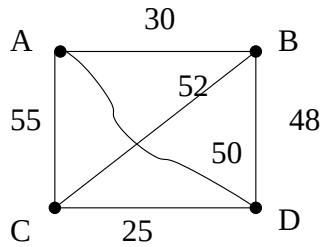
Der folgende Graph gibt Entfernungen (in km) zwischen den Städten A,B,C und D an:



Ein Handlungsreisender soll einen **möglichst kurzen** Rundweg finden, auf dem er alle Städte A, B, C, D besucht.

Das Problem des Handlungsreisenden (1/2)

Der folgende Graph gibt Entfernungen (in km) zwischen den Städten A,B,C und D an:

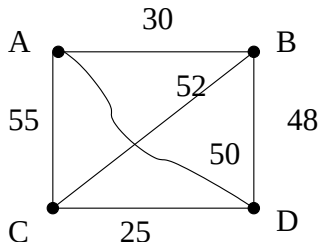


Ein Handlungsreisender soll einen **möglichst kurzen** Rundweg finden, auf dem er alle Städte A, B, C, D besucht.

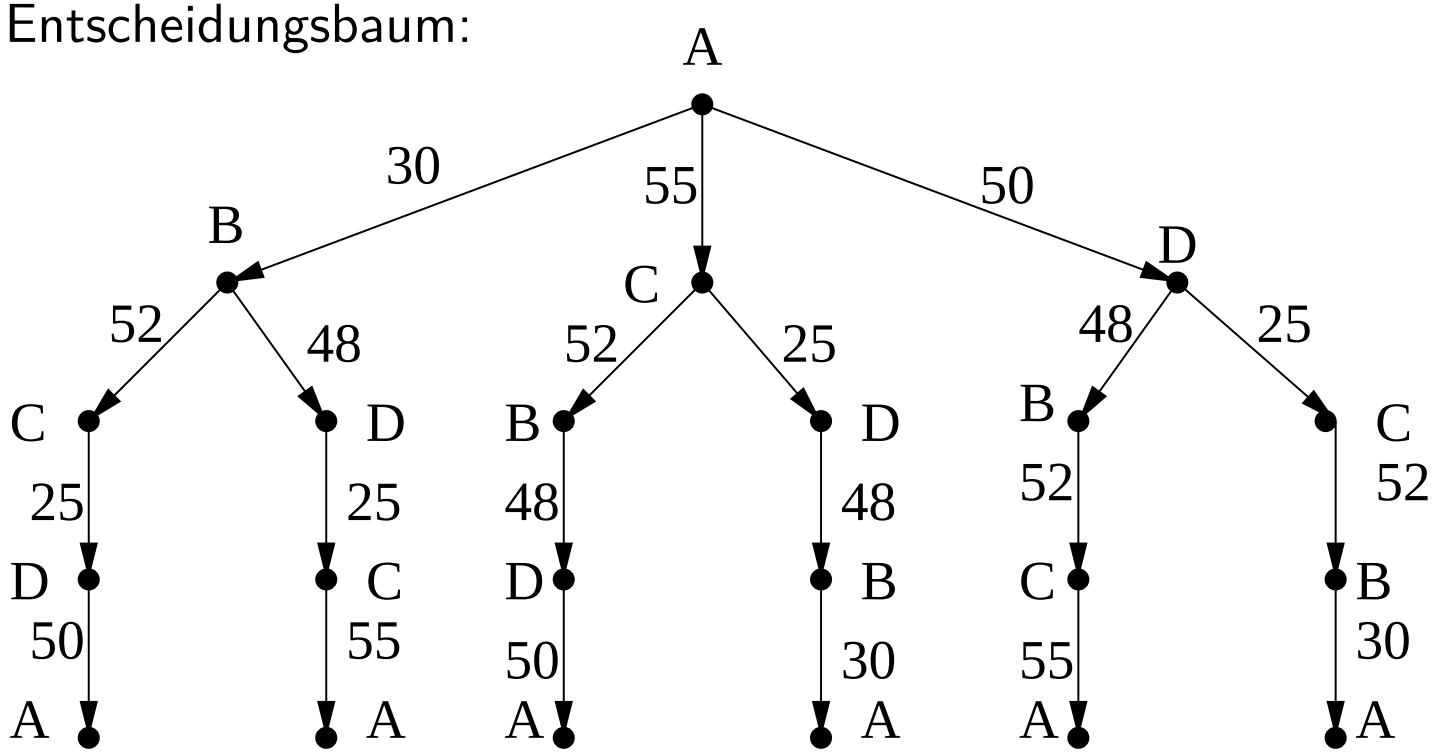
Wir möchten einen Entscheidungsbaum bauen, der systematisch alle Rundwege aufzählt.

Das Problem des Handlungsreisenden (2/2)

Zur Erinnerung:



Und hier ist der Entscheidungsbaum:



Und wenn es sehr viel mehr Städte gibt?

(a) Es gibt $(n-1)!$ viele verschiedene Rundwege, die alle in einer bestimmten Stadt „beginnen“.

Und wenn es sehr viel mehr Städte gibt?

(a) Es gibt $(n-1)!$ viele verschiedene Rundwege, die alle in einer bestimmten Stadt „beginnen“.

(b) Für $n = 4$ haben wir gerade einmal $3! = 6$ Rundwege und unser Entscheidungsbaum ist recht klein.

Aber was können wir noch für $n = 50$ ausrichten?

Und wenn es sehr viel mehr Städte gibt?

(a) Es gibt $(n-1)!$ viele verschiedene Rundwege, die alle in einer bestimmten Stadt „beginnen“.

(b) Für $n = 4$ haben wir gerade einmal $3! = 6$ Rundwege und unser Entscheidungsbaum ist recht klein.

Aber was können wir noch für $n = 50$ ausrichten?

Branch & Bound-Verfahren schneiden möglichst große Teilbäume aus dem Baum heraus, sobald es klar ist, dass dort keine optimale Lösungen „sitzen“.

Branch & Bound Verfahren werden in der Bachelor-Veranstaltung „**Approximationsalgorithmen**“ untersucht.

Spielbäume und der Minimax-Algorithmus

In einem zwei-Personen Spiel wie **Schach, Dame, Mühle, Go, Othello** ... spielen Alice und Bob gegeneinander. Können wir in einer vorgegebenen Spielsituation einen **gewinnenden Zug** für den ziehenden Spieler bestimmen?

Wir bauen einen **Spielbaum**.

Spielbäume und der Minimax-Algorithmus

In einem zwei-Personen Spiel wie **Schach, Dame, Mühle, Go, Othello** ... spielen Alice und Bob gegeneinander. Können wir in einer vorgegebenen Spielsituation einen **gewinnenden Zug** für den ziehenden Spieler bestimmen?

Wir bauen einen **Spielbaum**.

(a) Ein Knoten (S, X) im „Spielbaum“ entspricht einer Spielsituation S und einem ziehenden Spieler $X \in \{\,\text{Alice}, \text{Bob}\,\}$.

(b) Vom Knoten (S, X) aus wird für jeden möglichen Zug z des Spielers X eine mit z beschriftete Kante angelegt.

Spielbäume und der Minimax-Algorithmus

In einem zwei-Personen Spiel wie **Schach, Dame, Mühle, Go, Othello** ... spielen Alice und Bob gegeneinander. Können wir in einer vorgegebenen Spielsituation einen **gewinnenden Zug** für den ziehenden Spieler bestimmen?

Wir bauen einen **Spielbaum**.

(a) Ein Knoten (S, X) im „Spielbaum“ entspricht einer Spielsituation S und einem ziehenden Spieler $X \in \{\,\text{Alice}, \text{Bob}\,\}$.

(b) Vom Knoten (S, X) aus wird für jeden möglichen Zug z des Spielers X eine mit z beschriftete Kante angelegt.

(c) Ein Blatt wird mit 1 oder -1 beschriftet, falls Alice das Spiel gewinnt oder verliert.

Spielbäume und der Minimax-Algorithmus

In einem zwei-Personen Spiel wie **Schach, Dame, Mühle, Go, Othello** ... spielen Alice und Bob gegeneinander. Können wir in einer vorgegebenen Spielsituation einen **gewinnenden Zug** für den ziehenden Spieler bestimmen?

Wir bauen einen **Spielbaum**.

(a) Ein Knoten (S, X) im „Spielbaum“ entspricht einer Spielsituation S und einem ziehenden Spieler $X \in \{\,\text{Alice}, \text{Bob}\,\}$.

(b) Vom Knoten (S, X) aus wird für jeden möglichen Zug z des Spielers X eine mit z beschriftete Kante angelegt.

(c) Ein Blatt wird mit 1 oder -1 beschriftet, falls Alice das Spiel gewinnt oder verliert.

(d) Für einen inneren Knoten u definiere den Spielwert

$$\textbf{wert(u)} = \begin{cases} 1 & \text{Alice hat eine Gewinnstrategie} \\ -1 & \text{Bob hat eine Gewinnstrategie.} \end{cases}$$

- Angenommen, Alice ist in v am Zug und wir kennen „**wert(w)**“ für jedes Kind w von v. Wie sollten wir den Spielwert „**wert(v)**“ bestimmen?

Spielbäume und der Minimax-Algorithmus

In einem zwei-Personen Spiel wie **Schach, Dame, Mühle, Go, Othello** ... spielen Alice und Bob gegeneinander. Können wir in einer vorgegebenen Spielsituation einen **gewinnenden Zug** für den ziehenden Spieler bestimmen?

Wir bauen einen **Spielbaum**.

(a) Ein Knoten (S, X) im „Spielbaum“ entspricht einer Spielsituation S und einem ziehenden Spieler $X \in \{\,\text{Alice}, \text{Bob}\,\}$.

(b) Vom Knoten (S, X) aus wird für jeden möglichen Zug z des Spielers X eine mit z beschriftete Kante angelegt.

(c) Ein Blatt wird mit 1 oder -1 beschriftet, falls Alice das Spiel gewinnt oder verliert.

(d) Für einen inneren Knoten u definiere den Spielwert

$$\textbf{wert(u)} = \begin{cases} 1 & \text{Alice hat eine Gewinnstrategie} \\ -1 & \text{Bob hat eine Gewinnstrategie.} \end{cases}$$

- Angenommen, Alice ist in v am Zug und wir kennen „**wert(w)**“ für jedes Kind w von v. Wie sollten wir den Spielwert „**wert(v)**“ bestimmen?
- Und wenn Bob am Zug ist?