# WS 2003/04

# Diskrete Strukturen I

Ernst W. Mayr

mayr@in.tum.de
Institut für Informatik
Technische Universität München

01-13-2004

## Das Master Theorem

Bei der Analyse von *Divide-and-Conquer-Verfahren stößt man oft auf Rekursionen, die sich nicht als lineare Rekursionen formulieren lassen. So führt der Mergesort-Algorithmus in der Standardvariante zu der Rekursionsgleichung*

$$C_n = C_{\lfloor n/2 \rfloor} + C_{\lceil n/2 \rceil} + n \quad \textit{für alle } n > 1 \qquad \textit{und} \quad C_1 = 0.$$

*Löst man allgemein ein Problem der Größe $n$ dadurch, dass man es in $a$ Teilprobleme der Größe höchstens $n/b$ aufteilt, so erhält man für die Laufzeit $T(n)$ eine Rekursion der Form*

$$T(n) = a \cdot T(n/b) + f(n)\ ,$$

*wobei $f(n)$ die Laufzeit für die Aufteilung in Teilprobleme und für das Zusammenfügen der Lösungen der Teilprobleme ist.*

## Das Master Theorem

Bei der Analyse von *Divide-and-Conquer-Verfahren stößt man oft auf Rekursionen, die sich nicht als lineare Rekursionen formulieren lassen. So führt der Mergesort-Algorithmus in der Standardvariante zu der Rekursionsgleichung*

$$C_n = C_{\lfloor n/2 \rfloor} + C_{\lceil n/2 \rceil} + n \quad \textit{für alle } n > 1 \qquad \textit{und} \quad C_1 = 0.$$

*Löst man allgemein ein Problem der Größe $n$ dadurch, dass man es in $a$ Teilprobleme der Größe höchstens $n/b$ aufteilt, so erhält man für die Laufzeit $T(n)$ eine Rekursion der Form*

$$T(n) = a \cdot T(n/b) + f(n) \ ,$$

*wobei $f(n)$ die Laufzeit für die Aufteilung in Teilprobleme und für das Zusammenfügen der Lösungen der Teilprobleme ist.*

## Satz (Master Theorem)

*Seien $a \in \mathbb{N}$, $b > 1$ und $C \geq 0$ Konstanten, und sei $f(n)$ eine nichtnegative Funktion. Weiter seien $c_1(n), \ldots, c_a(n)$ Funktionen mit $|c_i(n)| \leq C$ für alle $1 \leq i \leq a$ und $n \in \mathbb{N}_0$. Ist dann $T(n)$ eine Funktion, die für $n = 1$ gleich $0$ ist und die für $n \geq 1$ die Rekursionsgleichung*

$$T(n) = T(n/b + c_1(n)) + \cdots + T(n/b + c_a(n)) + f(n)$$

*erfüllt, dann gilt*

$$T(n) = \begin{cases} \Theta(n^{\log_b a}), & \text{falls } f(n) = O(n^{\log_b a - \epsilon}) \text{ für ein } \epsilon > 0, \\ \Theta(n^{\log_b a} \log n), & \text{falls } f(n) = \Theta(n^{\log_b a} \cdot \log^{\delta} n) \text{ f. } \delta > 0, \\ \Theta(f(n)), & \text{falls } f(n) = \Omega(n^{\log_b a + \epsilon}) \text{ für ein } \epsilon > 0. \end{cases}$$

Für den Beweis des Master-Theorems verweisen wir auf die Literatur, z.B. in:


1. Verma, Rakesh M.:
   *A general method and a master theorem for divide-and-conquer recurrences with applications.*
   J. Algorithms **16**, 1 (1994), pp. 67–79
2. Roura, Salvador:
   *An improved master theorem for divide-and-conquer recurrences.*
   Proceedings of the 24th International Colloquium on Automata, Languages and Programming, ICALP'97 (Bologna, Italy, July 7-11, 1997). LNCS **1256** (1997), pp. 449–459

## Satz ("Baby-Version" des MT)

*Wenn die Funktion $T$ für $x < 1$ gleich $0$ ist und wenn für $x \geq 1$ die Rekursion*

$$T(x) = aT\left(x/b\right) + x$$

*gilt (also $T(1) = 1$), dann gilt für $n = b^t$ eine ganzzahlige Potenz von $b$:*

$$T(n) = (1 + o(1)) \cdot \begin{cases} \frac{b}{b-a} n, & \text{falls } a < b, \\ n \log_b n, & \text{falls } a = b, \\ \frac{a}{a-b} n^{\log_b a}, & \text{falls } a > b\,. \end{cases}$$

**Beweis**

Zuerst wenden wir die Rekursionsgleichung so oft an, bis wir die Anfangsbedingung erreichen. Wir haben also

$$\begin{aligned} T(n) &= n + aT(n/b) \\ &= n + a\frac{n}{b} + a^2 T(n/b^2) \\ &= n + a\frac{n}{b} + a^2\frac{n}{b^2} + a^3 T(n/b^3) \\ &= \cdots \\ &= n + a\frac{n}{b} + a^2\frac{n}{b^2} + \cdots + a^t T(n/b^t), \end{aligned}$$

wobei $t = \log_b n$. Also

$$T(n) = n\left(1 + \frac{a}{b} + \cdots + \frac{a^t}{b^t}\right)$$

**Beweis**

Zuerst wenden wir die Rekursionsgleichung so oft an, bis wir die Anfangsbedingung erreichen. Wir haben also

$$\begin{aligned} T(n) &= n + aT(n/b) \\ &= n + a\frac{n}{b} + a^2 T(n/b^2) \\ &= n + a\frac{n}{b} + a^2\frac{n}{b^2} + a^3 T(n/b^3) \\ &= \cdots \\ &= n + a\frac{n}{b} + a^2\frac{n}{b^2} + \cdots + a^t T(n/b^t), \end{aligned}$$

wobei $t = \log_b n$. Also

$$T(n) = n\left(1 + \frac{a}{b} + \cdots + \frac{a^t}{b^t}\right)$$

## Fallunterscheidung

$a < b$: In diesem Fall konvergiert die Summe und wir erhalten:

$$T(n) \leq n \sum_{k \geq 0} \left(\frac{a}{b}\right)^k = \frac{b}{b-a} n \; .$$

$a = b$: In diesem Fall ist die Lösung

$$T(n) = n \left(\log_b n + 1\right) = (1 + o(1)) \cdot n \log_b n \; .$$

## Fallunterscheidung

$a < b$: In diesem Fall konvergiert die Summe und wir erhalten:

$$T(n) \leq n \sum_{k \geq 0} \left(\frac{a}{b}\right)^k = \frac{b}{b-a} n \ .$$

$a = b$: In diesem Fall ist die Lösung

$$T(n) = n \left(\log_b n + 1\right) = (1 + o(1)) \cdot n \log_b n \ .$$

$a > b$: Wir erhalten:

$$\begin{aligned} T(n) &= n\left(\frac{a}{b}\right)^t \left(1 + \frac{b}{a} + \cdots + \frac{b^t}{a^t}\right) \\ &\leq n\frac{a}{a-b}\left(\frac{a}{b}\right)^t \\ &= \frac{a}{a-b}a^{\log_b n} \\ &= \frac{a}{a-b}n^{\log_b a}\ , \end{aligned}$$

da $t = \log_b n$. *q. e. d.*

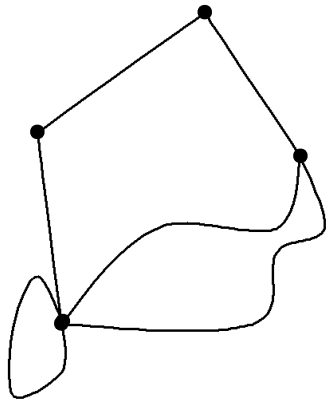
# Kapitel 5: Graphen und Algorithmen

## Grundlagen

### Definition

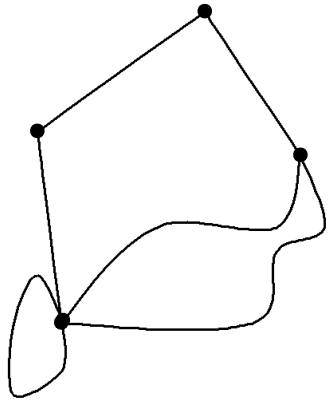
Ein *Graph* $G = (V, E)$ *besteht aus einer Menge* $V$ *von Knoten (aka Ecken, engl. vertex, vertices) und einer (Mehrfach-)Menge* $E \subseteq V \times V$ *von Paaren* $(u, v) \in V \times V$*, genannt Kanten (engl. edges).*

Ein Graph heißt *ungerichteter Graph, falls für alle* $(u, v) \in E$ *auch* $(v, u) \in E$ *ist. Man schreibt dann* $E$ *auch als Menge von ungeordneten Paaren* $\{u, v\}$ *von Kanten.*

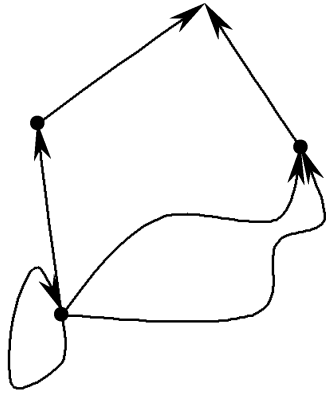


*Ein Graph heißt ein gerichteter Graph, falls* $E$ *(wie in obiger Definition) eine Menge von geordneten Paaren* $(u, v)$ *ist.*

Ein Graph heißt *ungerichteter Graph, falls für alle* $(u, v) \in E$ *auch* $(v, u) \in E$ *ist. Man schreibt dann* $E$ *auch als Menge von ungeordneten Paaren* $\{u, v\}$ *von Kanten.*



*Ein Graph heißt ein gerichteter Graph, falls* $E$ *(wie in obiger Definition) eine Menge von geordneten Paaren* $(u, v)$ *ist.*



## Schlingen

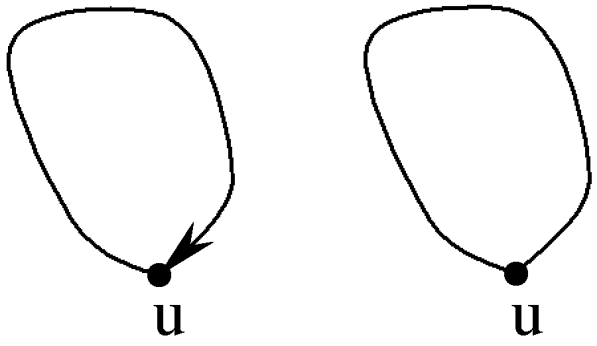
### Definition

Eine *Schlinge ist eine Kante der Form* $(u, u)$ *bzw.* $\{u, u\}$.

## Schlingen

### Definition

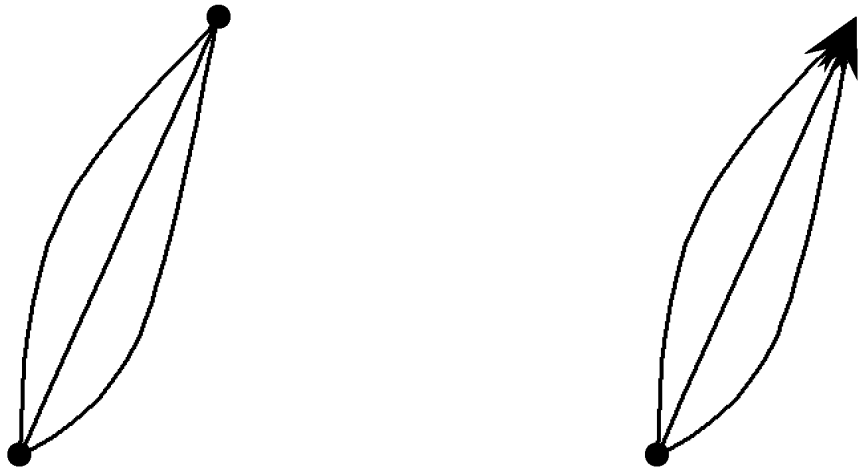
Eine *Schlinge ist eine Kante der Form* $(u, u)$ *bzw.* $\{u, u\}$.



## Mehrfachkanten

### Definition

Ist $E$ eine Multimenge (d. h. Kanten treten mit Vielfachheit auf), sind die Kanten mit Vielfachheit 2 oder größer *Mehrfachkanten.*

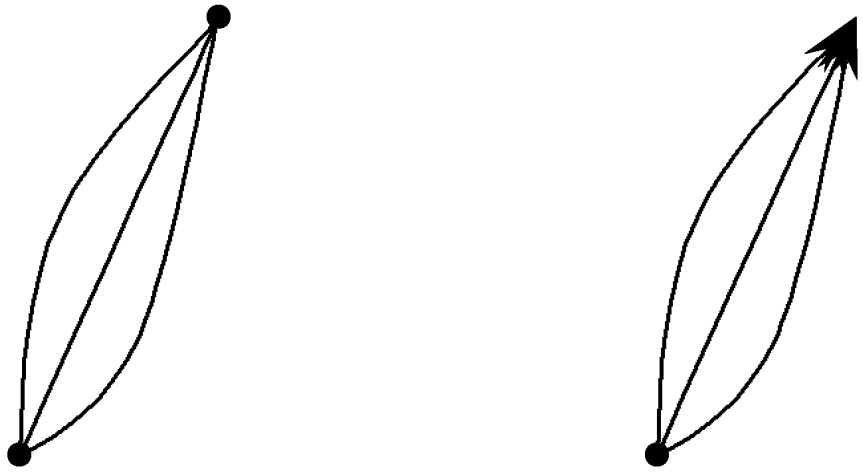


Ein Graph, der Mehrfachkanten enthält, heißt auch *Multigraph.*

## Mehrfachkanten

### Definition

Ist $E$ eine Multimenge (d. h. Kanten treten mit Vielfachheit auf), sind die Kanten mit Vielfachheit 2 oder größer *Mehrfachkanten.*



Ein Graph, der Mehrfachkanten enthält, heißt auch *Multigraph.*

## Einfache Graphen

### Definition
Ein Graph heißt *einfach, falls er keine Schlingen oder Mehrfachkanten enthält.*

### Definition
Ein Graph $G = (V, E)$ $(=: K_n)$ mit $|V| = n$ Knoten heißt *vollständig (der vollständige Graph mit $n$ Knoten), falls* $E = \{\{u, v\}; u, v \in V, u \neq v\}$ *bzw.* $E = \{(u, v); u, v \in V, u \neq v\}$.

## Einfache Graphen

### Definition
Ein Graph heißt *einfach, falls er keine Schlingen oder Mehrfachkanten enthält.*

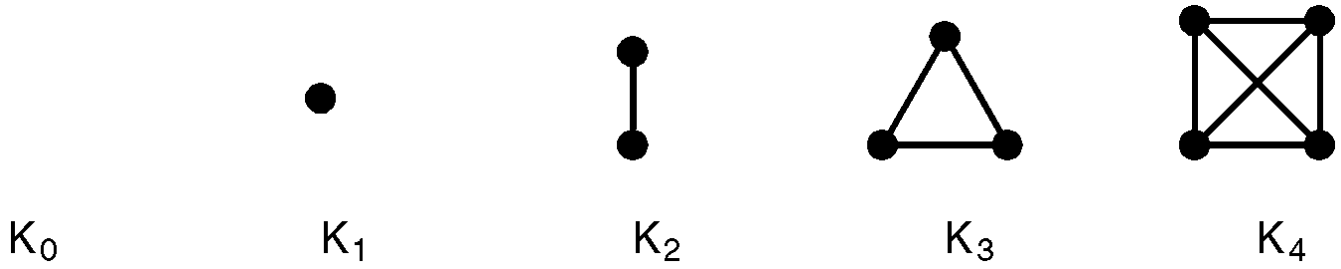
### Definition
Ein Graph $G = (V, E)$ $(=: K_n)$ mit $|V| = n$ Knoten heißt *vollständig (der vollständige Graph mit $n$ Knoten), falls* $E = \{\{u, v\}; u, v \in V, u \neq v\}$ *bzw.* $E = \{(u, v); u, v \in V, u \neq v\}$.

## Einfache Graphen

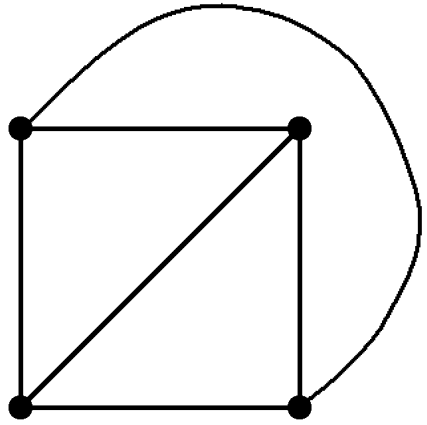
### Definition
Ein Graph heißt *einfach, falls er keine Schlingen oder Mehrfachkanten enthält.*

### Definition
Ein Graph $G = (V, E)$ $(=: K_n)$ mit $|V| = n$ Knoten heißt *vollständig (der vollständige Graph mit $n$ Knoten), falls* $E = \{\{u, v\}; u, v \in V, u \neq v\}$ *bzw.* $E = \{(u, v); u, v \in V, u \neq v\}$.

### Beispiel



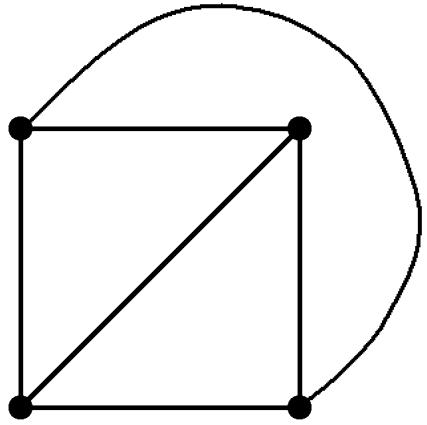
Der $K_4$ lässt sich auch kreuzungsfrei zeichnen:



Für die Anzahl der Kanten in einem vollständigen Graphen (und damit für die *maximale* Anzahl von Kanten in einem einfachen Graphen) gilt:

$$|E| = \binom{n}{2} = \frac{n \cdot (n-1)}{2}$$

Der $K_4$ lässt sich auch kreuzungsfrei zeichnen:



Für die Anzahl der Kanten in einem vollständigen Graphen (und damit für die *maximale* Anzahl von Kanten in einem einfachen Graphen) gilt:

$$|E| = \binom{n}{2} = \frac{n \cdot (n-1)}{2}$$

## Bipartiter Graph

### Definition

Ein Graph heißt *bipartit, falls sich* $V$ *in* $V_1 \uplus V_2$ *mit* $V_1 \neq \emptyset \neq V_2$ *so partitionieren lässt, dass gilt:*

$$(\forall e \in E)\big[e \in (V_1 \times V_2) \cup (V_2 \times V_1)\big]$$

### Beispiel ($C_8$, Kreis mit 8 Knoten)

**Bemerkung:**

Schreibweise für bipartite Graphen:

$$G = (V_1, V_2, E)$$

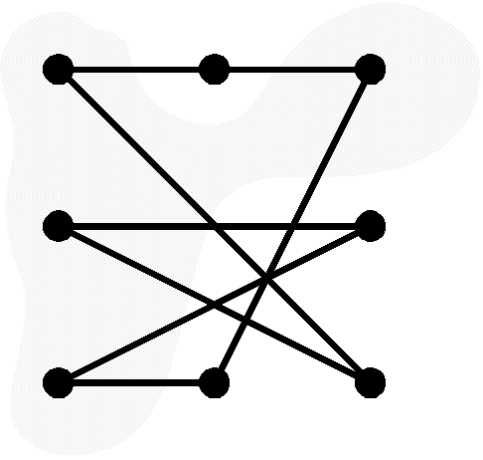
## Bipartiter Graph

### Definition

Ein Graph heißt *bipartit, falls sich* $V$ *in* $V_1 \uplus V_2$ *mit* $V_1 \neq \emptyset \neq V_2$ *so partitionieren lässt, dass gilt:*

$$(\forall e \in E)\big[e \in (V_1 \times V_2) \cup (V_2 \times V_1)\big]$$

### Beispiel ($C_8$, Kreis mit 8 Knoten)



**Bemerkung:**

Schreibweise für bipartite Graphen:

$$G = (V_1, V_2, E)$$

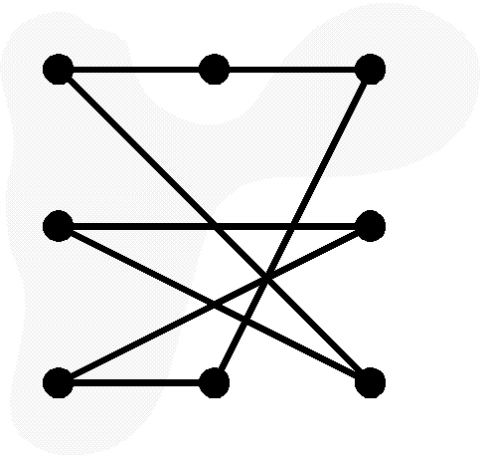
## Bipartiter Graph

### Definition

Ein Graph heißt *bipartit, falls sich* $V$ *in* $V_1 \uplus V_2$ *mit* $V_1 \neq \emptyset \neq V_2$ *so partitionieren lässt, dass gilt:*

$$(\forall e \in E)\big[e \in (V_1 \times V_2) \cup (V_2 \times V_1)\big]$$

### Beispiel ($C_8$, Kreis mit 8 Knoten)



**Bemerkung:**

Schreibweise für bipartite Graphen:

$$G = (V_1, V_2, E)$$

# Vollständiger bipartiter Graph

## Definition

Ein bipartiter Graph $G = (V_1, V_2, E)$ heißt *vollständig, falls* $E = V_1 \times V_2 \cup V_2 \times V_1$.
*(Notation:* $K_{m,n}$ *mit* $m = |V_1|, n = |V_2|$*)*

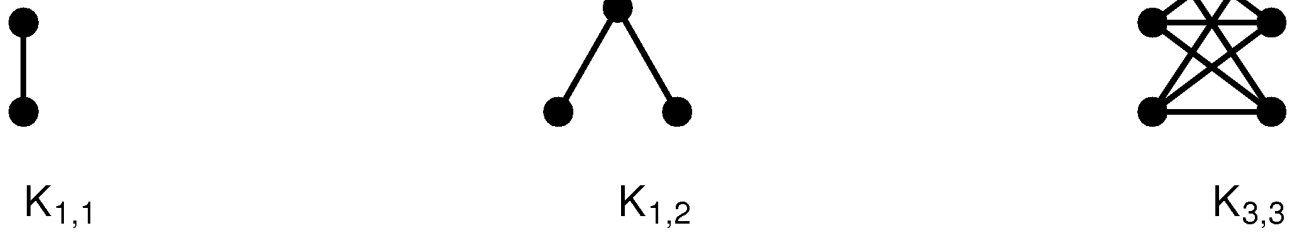
## Beispiel

# Vollständiger bipartiter Graph

## Definition

Ein bipartiter Graph $G = (V_1, V_2, E)$ heißt *vollständig, falls* $E = V_1 \times V_2 \cup V_2 \times V_1$.
*(Notation: $K_{m,n}$ mit $m = |V_1|, n = |V_2|$)*

## Beispiel



## $k$-partiter Graph

### Definition

Ein Graph heißt *$k$-partit ($k \in \mathbb{N}, k \geq 2$), falls es eine Partition $V = V_1 \uplus V_2 \uplus \ldots \uplus V_k$ mit $V_i \neq \emptyset, i = 1, \ldots, k$ gibt, so dass*

$$(\forall e \in E)\big[e \in V_i \times V_j; 1 \leq i, j \leq k, i \neq j\big]$$

### Beispiel (Vollständiger tripartiter Graph $K_{2,2,2}$)

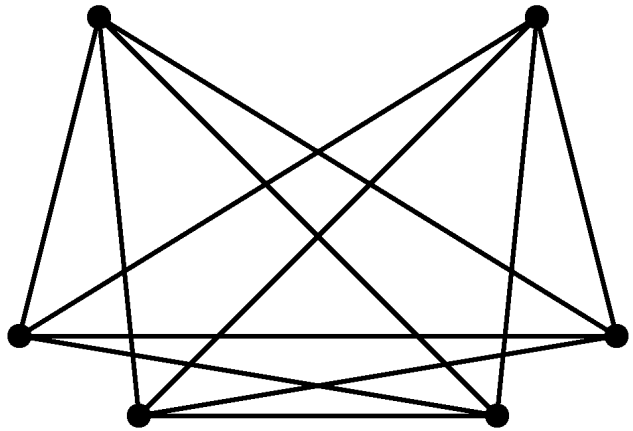
## $k$-partiter Graph

### Definition

Ein Graph heißt *$k$-partit ($k \in \mathbb{N}, k \geq 2$), falls es eine Partition $V = V_1 \uplus V_2 \uplus \ldots \uplus V_k$ mit $V_i \neq \emptyset, i = 1, \ldots, k$ gibt, so dass*

$$(\forall e \in E)\big[e \in V_i \times V_j; 1 \leq i, j \leq k, i \neq j\big]$$

### Beispiel (Vollständiger tripartiter Graph $K_{2,2,2}$)



# (Binäre) Hyperwürfel

## Definition

Ein Graph $G = (V, E)$ heißt *$n$-dimensionaler binärer Hyperwürfel (aka $Q_n$), falls $V = V_n = \{0,1\}^n$ mit*

$$E = \Big\{ \{v, w\} \in {V_n}^2;\ \textit{Hamming-Abstand}(v, w) = 1 \Big\}.$$

## Beispiel

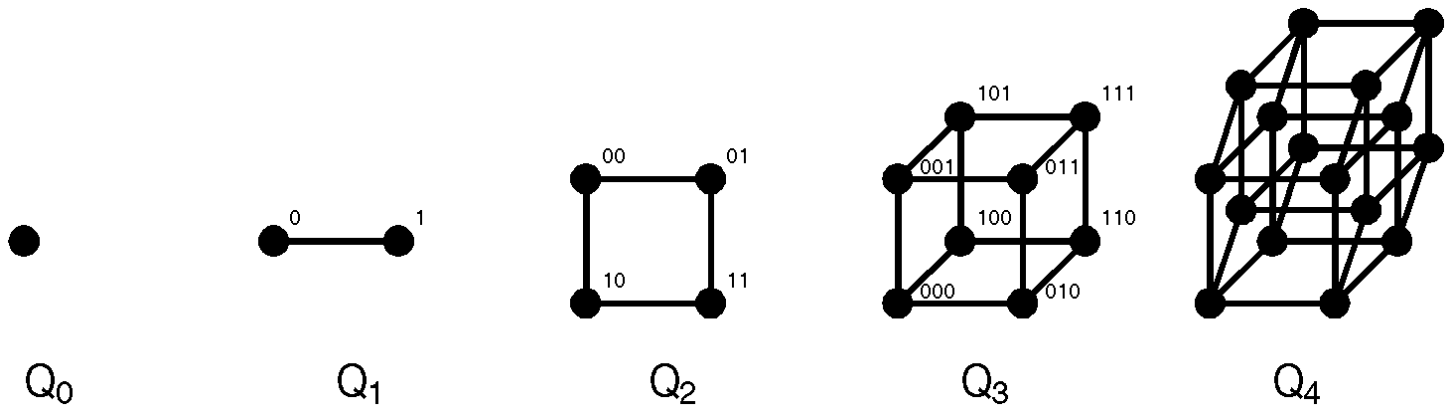
# (Binäre) Hyperwürfel

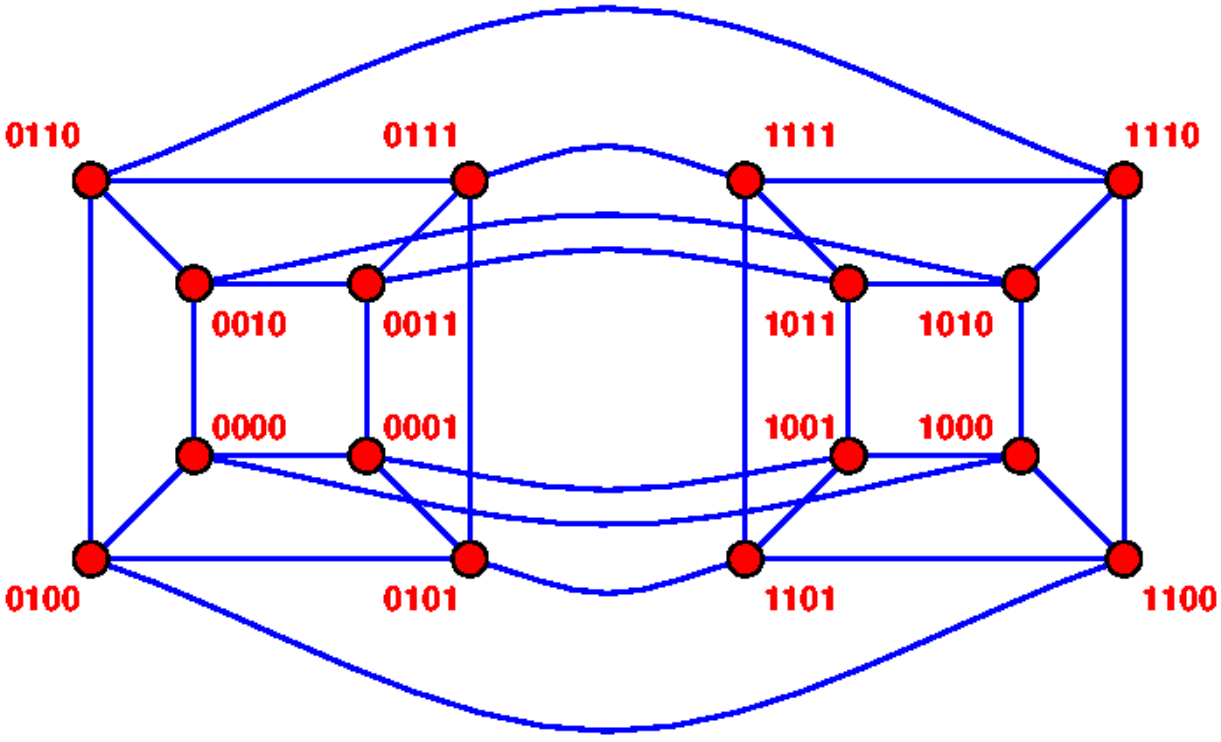
## Definition

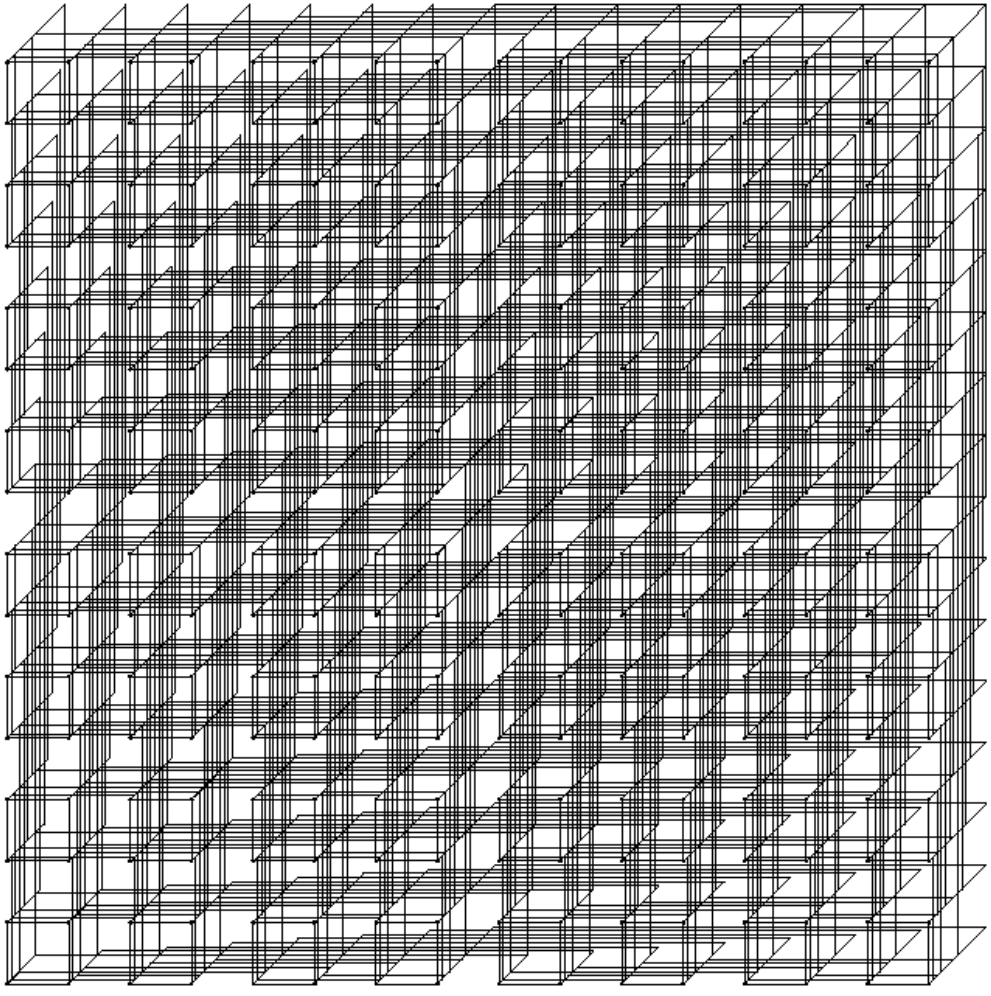
Ein Graph $G = (V, E)$ heißt *$n$-dimensionaler binärer Hyperwürfel (aka $Q_n$), falls $V = V_n = \{0,1\}^n$ mit*

$$E = \Big\{ \{v, w\} \in {V_n}^2;\ \textit{Hamming-Abstand}(v, w) = 1 \Big\}.$$

## Beispiel




0110
0111
1111
1110
0010
0011
1011
1010
0000
0001
1001
1000
0100
0101
1101
1100



Für die Anzahl der Knoten in $Q_n$ gilt:

$$|V| = 2^n$$

Für die Anzahl der Kanten in $Q_n$ gilt:

$$|E| = n \cdot \frac{2^n}{2} = n \cdot 2^{n-1}$$

Für die Anzahl der Knoten in $Q_n$ gilt:

$$|V| = 2^n$$

Für die Anzahl der Kanten in $Q_n$ gilt:

$$|E| = n \cdot \frac{2^n}{2} = n \cdot 2^{n-1}$$

# Pfade

## Definition

1. Ein *Pfad der Länge $n$ ist eine Folge $(v_1, v_2, \ldots, v_n)$ von Knoten eines Graphen $G = (V, E)$, so dass $(v_i, v_{i+1}) \in E$ für alle $i = 1, \ldots, n-1$.*

*2. Der Graph $P_n$ ist der Graph $(V, E)$ mit $V = \{v_1, \ldots, v_n\}$ und $E = \big\{\{v_i, v_{i+1}\}; i = 1, \ldots, n-1\big\}$.*

## Beispiel

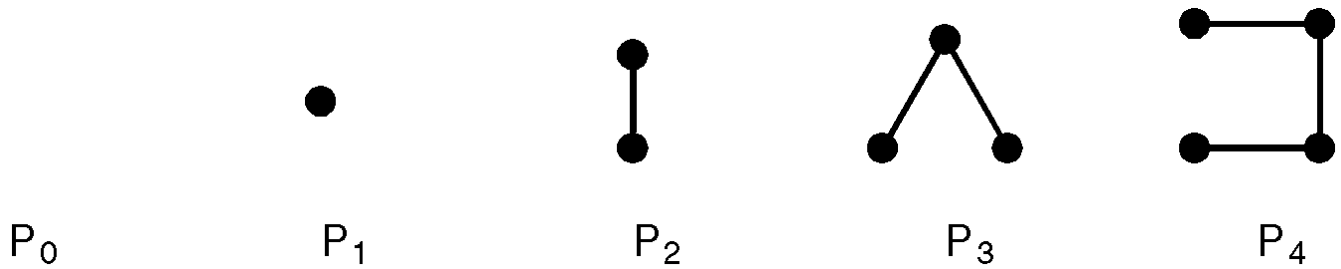
# Pfade

## Definition

1. Ein *Pfad der Länge $n$ ist eine Folge $(v_1, v_2, \ldots, v_n)$ von Knoten eines Graphen $G = (V, E)$, so dass $(v_i, v_{i+1}) \in E$ für alle $i = 1, \ldots, n-1$.*

*2. Der Graph $P_n$ ist der Graph $(V, E)$ mit $V = \{v_1, \ldots, v_n\}$ und $E = \big\{\{v_i, v_{i+1}\}; i = 1, \ldots, n-1\big\}$.*

## Beispiel



## Definition

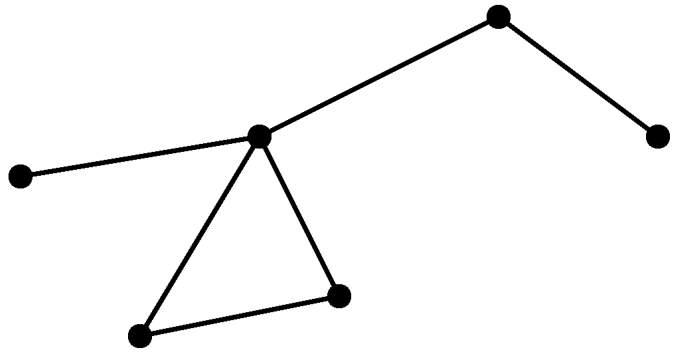
Ein Pfad heißt *einfach, falls alle Knoten paarweise verschieden sind.*

Beispiel (Pfad, aber *nicht einfacher* Pfad der Länge 7)

## Definition

Ein Pfad heißt *einfach, falls alle Knoten paarweise verschieden sind.*

## Beispiel (Pfad, aber *nicht einfacher* Pfad der Länge 7)



## Kreise

### Definition

Ein Graph $G = (V, E)$ heißt *(einfacher) Kreis der Länge* $n$ *(i. Z.* $C_n, n \geq 3$*), falls* $V = \{v_0, \ldots, v_{n-1}\}$ *und*
$E = \big\{\{v_i, v_{(i+1) \bmod n}\}; i = 0, \ldots, n-1\big\}$.

# Gitter

## Definition

Ein Graph $G = (V, E)$ heißt ein *$m$-$n$-Gitter (zweidimensionales Gitter mit den Seitenlängen $m$ und $n$, i. Z. $M_{m,n}$), falls* $V = \{1, \ldots, m\} \times \{1, \ldots, n\}$ *und*

$$\underbrace{\{(i,j),(k,l)\}}_{\substack{\text{Kante zwischen} \\ \text{Knoten } (i,j) \\ \text{und Knoten } (k,l)}} \in E \iff |i-k| + |j-l| = 1$$

## Beispiel

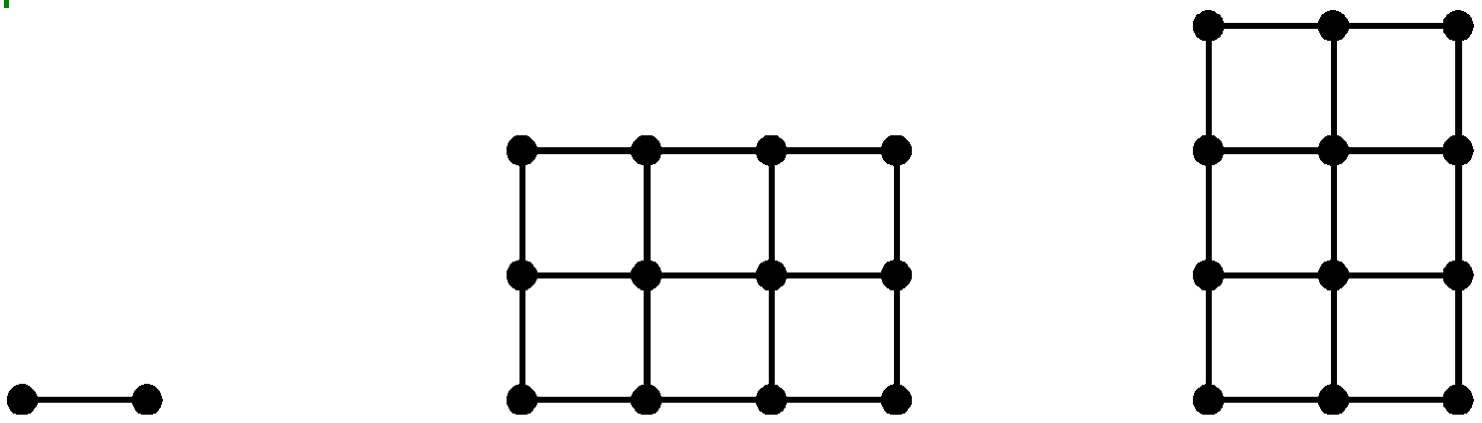
# Gitter

## Definition

Ein Graph $G = (V, E)$ heißt ein $m$-$n$-*Gitter (zweidimensionales Gitter mit den Seitenlängen* $m$ *und* $n$*, i. Z.* $M_{m,n}$*), falls* $V = \{1, \ldots, m\} \times \{1, \ldots, n\}$ *und*

$$\underbrace{\{(i,j),(k,l)\}}_{\substack{\text{Kante zwischen} \\ \text{Knoten } (i,j) \\ \text{und Knoten } (k,l)}} \in E \iff |i-k| + |j-l| = 1$$

## Beispiel



## Torus

### Definition

Ein Graph $G = (V, E)$ heißt *zweidimensionaler Torus (pl. Tori) mit den Seitenlängen $m$ und $n$, falls $V = \{1, \ldots, m\} \times \{1, \ldots, n\}$ und*

$$\{(i, j), (k, l)\} \in E \iff |(i - k) \bmod m| + |(j - l) \bmod n| = 1$$

### Beispiel

# Torus

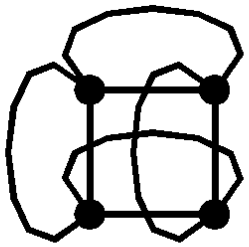
## Definition

Ein Graph $G = (V, E)$ heißt *zweidimensionaler Torus (pl. Tori) mit den Seitenlängen $m$ und $n$, falls $V = \{1, \ldots, m\} \times \{1, \ldots, n\}$ und*

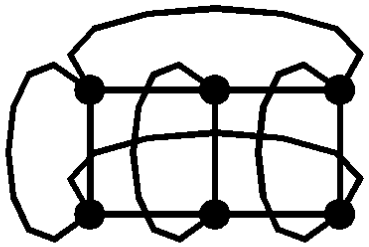
$$\{(i, j), (k, l)\} \in E \iff |(i - k) \bmod m| + |(j - l) \bmod n| = 1$$

## Beispiel

$T_{1,2}$



$T_{2,2}$

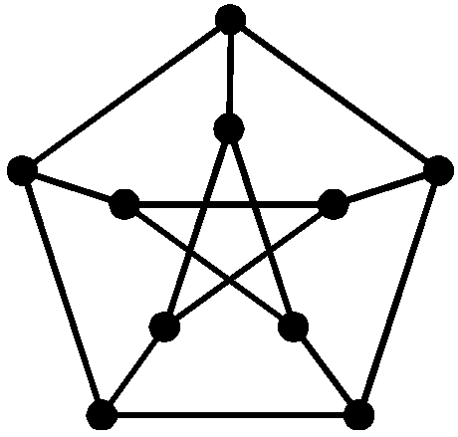


$T_{2,3}$

# Petersen-Graph

## Definition

Der folgende Graph heißt *Petersen-Graph:*



## Definitionen für ungerichtete Graphen

### $P_n$: Der Pfad mit $n$ Knoten

**Definition**

Ein *Pfad (Weg) in einem Graphen ist eine Folge von Knoten* $v_0, v_1, \ldots, v_k$ *mit* $\{v_i, v_{i+1}\} \in E, i = 0, \ldots, k-1$. *Ein Pfad heißt einfach, wenn alle* $v_i$ *paarweise verschieden sind. Ein Kreis ist ein Pfad, bei dem gilt:* $v_0 = v_k$. *Ein Kreis heißt einfach, wenn die Knoten* $v_0, \ldots, v_{k-1}$ *paarweise verschieden sind.*

## Definitionen für ungerichtete Graphen

### $P_n$: Der Pfad mit $n$ Knoten

**Definition**

Ein *Pfad (Weg) in einem Graphen ist eine Folge von Knoten* $v_0, v_1, \ldots, v_k$ *mit* $\{v_i, v_{i+1}\} \in E, i = 0, \ldots, k-1$. *Ein Pfad heißt einfach, wenn alle* $v_i$ *paarweise verschieden sind. Ein Kreis ist ein Pfad, bei dem gilt:* $v_0 = v_k$. *Ein Kreis heißt einfach, wenn die Knoten* $v_0, \ldots, v_{k-1}$ *paarweise verschieden sind.*

## Definitionen für ungerichtete Graphen

### $P_n$: Der Pfad mit $n$ Knoten

**Definition**

Ein *Pfad (Weg) in einem Graphen ist eine Folge von Knoten* $v_0, v_1, \ldots, v_k$ *mit* $\{v_i, v_{i+1}\} \in E, i = 0, \ldots, k-1$. *Ein Pfad heißt einfach, wenn alle* $v_i$ *paarweise verschieden sind. Ein Kreis ist ein Pfad, bei dem gilt:* $v_0 = v_k$. *Ein Kreis heißt einfach, wenn die Knoten* $v_0, \ldots, v_{k-1}$ *paarweise verschieden sind.*

## Definitionen für ungerichtete Graphen

### $P_n$: Der Pfad mit $n$ Knoten

**Definition**

Ein *Pfad (Weg) in einem Graphen ist eine Folge von Knoten* $v_0, v_1, \ldots, v_k$ *mit* $\{v_i, v_{i+1}\} \in E, i = 0, \ldots, k-1$. *Ein Pfad heißt einfach, wenn alle* $v_i$ *paarweise verschieden sind. Ein Kreis ist ein Pfad, bei dem gilt:* $v_0 = v_k$. *Ein Kreis heißt einfach, wenn die Knoten* $v_0, \ldots, v_{k-1}$ *paarweise verschieden sind.*

## Isomorphe Graphen

### Definition

Zwei Graphen $G_i = (V_i, E_i)$, $i = 1, 2$ heißen *isomorph, falls es eine Bijektion $\varphi : V_1 \to V_2$ gibt, so dass gilt:*

$$\big(\forall v, w \in V_1\big)\Big[\{v, w\} \in E_1 \iff \big\{\varphi(v), \varphi(w)\big\} \in E_2\Big].$$

### Beispiel

$K_{2,2} \cong C_4 \cong Q_2$ oder $T_{4,4,4} \cong Q_6$

### Beispiel

## Isomorphe Graphen

### Definition

Zwei Graphen $G_i = (V_i, E_i)$, $i = 1, 2$ heißen *isomorph, falls es eine Bijektion* $\varphi : V_1 \to V_2$ *gibt, so dass gilt:*

$$\big(\forall v, w \in V_1\big)\Big[\{v, w\} \in E_1 \iff \big\{\varphi(v), \varphi(w)\big\} \in E_2\Big].$$

### Beispiel

$K_{2,2} \cong C_4 \cong Q_2$ oder $T_{4,4,4} \cong Q_6$

### Beispiel

## Isomorphe Graphen

### Definition

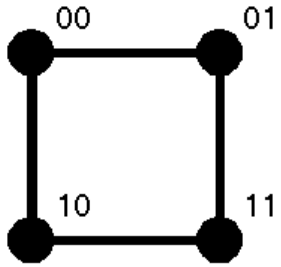
Zwei Graphen $G_i = (V_i, E_i)$, $i = 1, 2$ heißen *isomorph, falls es eine Bijektion* $\varphi : V_1 \to V_2$ *gibt, so dass gilt:*

$$\big(\forall v, w \in V_1\big)\Big[\{v, w\} \in E_1 \iff \big\{\varphi(v), \varphi(w)\big\} \in E_2\Big].$$

### Beispiel

$K_{2,2} \cong C_4 \cong Q_2$ oder $T_{4,4,4} \cong Q_6$

### Beispiel




$Q_2$

## Adjazenz

### Definition

Sei $G = (V, E)$, $u, v \in V$ und $\{u, v\} \in E$. Dann heißen $u$ und $v$ *adjazent (aka benachbart).* $u$ *und* $v$ *sind Endknoten von* $\{u, v\}$*;* $u$ *und* $v$ *sind inzident zur Kante* $\{u, v\}$*. Zwei Kanten heißen adjazent, falls sie einen Endknoten gemeinsam haben.*

## Nachbarschaft

**Definition**
Sei $u \in V$.

$$N(u) := \{v \in V; u \neq v, \{u, v\} \in E\}$$

heißt die *Nachbarschaft von* $u$. $d(u) := \deg(u) := \left|N(u)\right|$ *heißt Grad von* $u$. *Falls* $d(u) = 0$, *so heißt* $u$ *isoliert.*

## Nachbarschaft

### Definition

Sei $u \in V$.

$$N(u) := \{v \in V; u \neq v, \{u, v\} \in E\}$$

heißt die *Nachbarschaft von* $u$. $d(u) := \deg(u) := \left|N(u)\right|$ *heißt Grad von* $u$. *Falls* $d(u) = 0$, *so heißt* $u$ *isoliert.*

## Nachbarschaft

### Definition

Sei $u \in V$.

$$N(u) := \{v \in V; u \neq v, \{u, v\} \in E\}$$

heißt die *Nachbarschaft von* $u$. $d(u) := \deg(u) := \left|N(u)\right|$ *heißt Grad von* $u$. *Falls* $d(u) = 0$, *so heißt* $u$ *isoliert.*