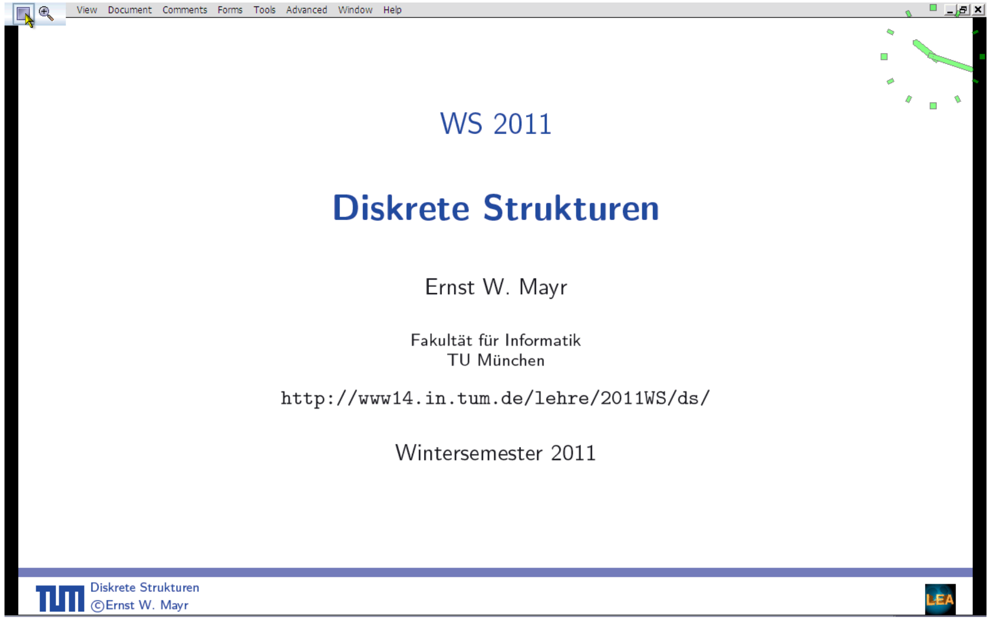
Script generated by TTT

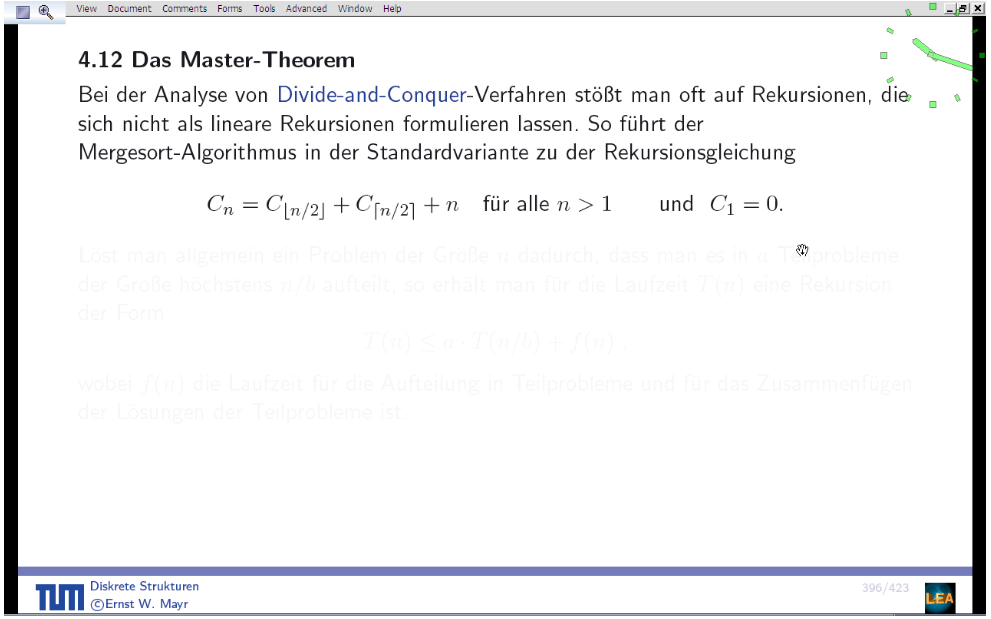
Title: Mayr: 2011 ds (19.01.2012)

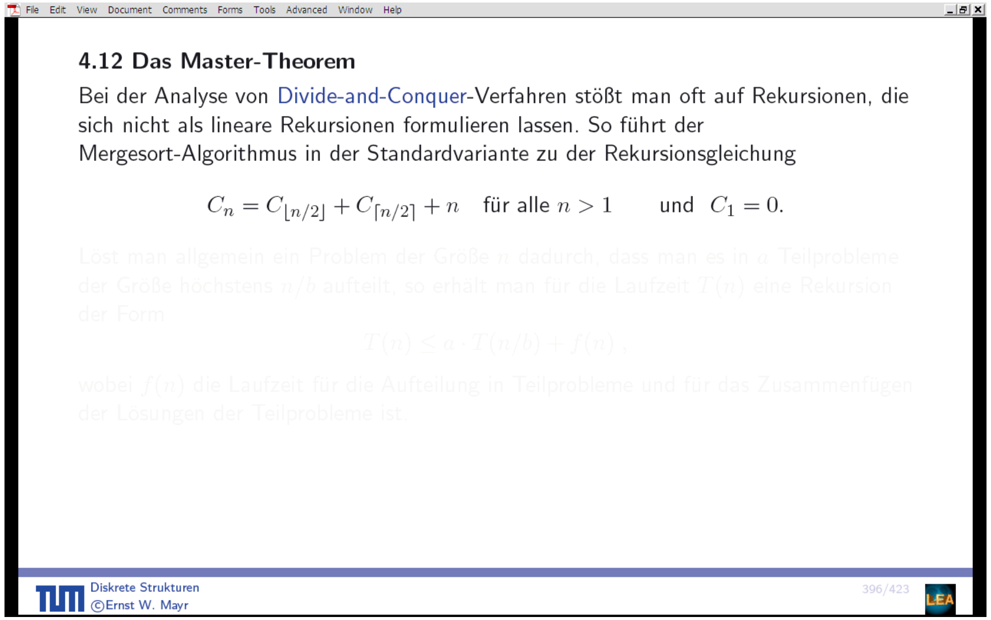
Date: Thu Jan 19 10:18:10 CET 2012

Duration: 89:34 min

Pages: 40







4.12 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

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

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

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

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

4.12 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

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

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

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

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

4.12 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

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

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

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

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

Satz 227 (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

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

Satz 227 (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:

Verma, Rakesh M.:
A general method and a master theorem for divide-and-conquer recurrences with applications.
J. Algorithms **16**(1), pp. 67–79, 1994

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**, pp. 449–459, 1997

Satz 228 ("Baby-Version" des MT)

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

$$T(x) = aT(x/b) + x$$

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

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

Satz 228 ("Baby-Version" des MT)

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

$$T(x) = aT(x/b) + x$$

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

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

Diskrete Strukturen ©Ernst W. Mayr 4.12 Das Master-Theorem 399/423

Beweis:

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

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

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

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

wobei $t = \log_b n$. Also

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

Beweis (Forts.):

Fallunterscheidung:

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

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

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

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

Beweis:

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

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

wobei $t = \log_b n$. Also

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

Beweis (Forts.):

Fallunterscheidung:

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

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

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

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

Beweis (Forts.):

$a > b$: Wir erhalten:

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

da $t = \log_b n$. □

Beweis (Forts.):

$a > b$: Wir erhalten:

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

da $t = \log_b n$. □

Kapitel IV Graphen und Algorithmen

1. Grundlagen

Definition 229

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).

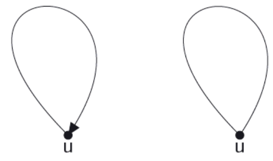
Ein Graph heißt ungerichteter Graph, falls für alle $(u, v) \in E$ auch $(v, u) \in E$ ist. Man schreibt dann E auch als Menge von ungeordneten Paaren $\{u, v\}$ von Kanten.

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

1.1 Schlingen

Definition 230

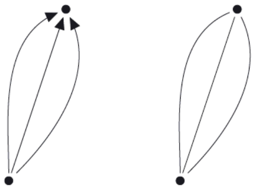
Eine Schlinge ist eine Kante der Form (u, u) bzw. $\{u, u\}$.



1.2 Mehrfachkanten

Definition 231

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.

1.3 Einfache Graphen

Definition 232

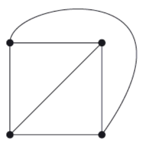
Ein Graph heißt einfach, falls er keine Schlingen oder Mehrfachkanten enthält.

Definition 233

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 234

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:

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

1.4 Bipartiter Graph

Definition 235

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 236 (C_8, Kreis mit 8 Knoten)

Bemerkung:
Schreibweise für bipartite Graphen:

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

1.5 Vollständiger bipartiter Graph

Definition 237

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 238

$K_{1,1}$ $K_{1,2}$ $K_{3,3}$

1.6 k-partiter Graph

Definition 239

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 240 (Vollständiger tripartiter Graph $K_{2,2,2}$)

1.7 (Binärer) Hyperwürfel

Definition 241

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

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

Beispiel 242

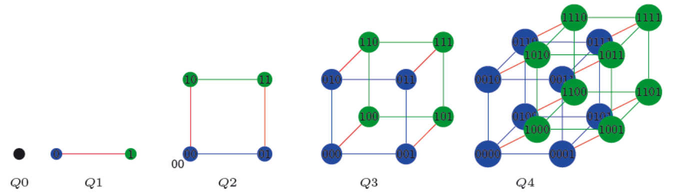
1.7 (Binärer) Hyperwürfel

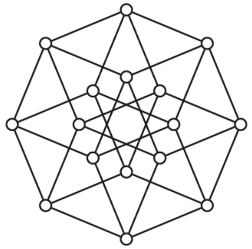
Definition 241

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

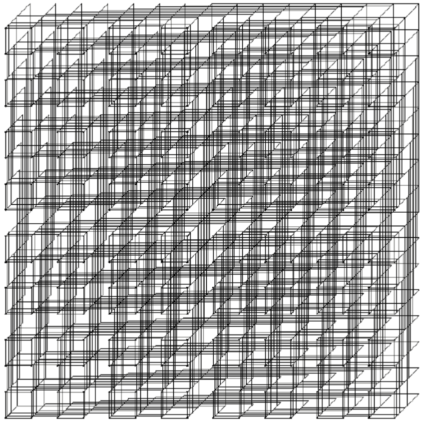
$$E = \Big\{\{v, w\} \in {V_n}^2;\ \text{Hamming-Abstand}(v, w) = 1\Big\}.$$

Beispiel 242





Q_4: 4-dimensionaler Hyperwürfel



Q_8: 8-dimensionaler Hyperwürfel

Für die Anzahl der Knoten in Q_n gilt:

$$|V| = 2^n$$

Für die Anzahl der Kanten in Q_n gilt:

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

1.8 Pfade

Definition 243

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 $E = \{\{v_i, v_{i+1}\}; i = 1, \ldots, n-1\}$.

Beispiel 244

P_0 P_1 P_2 P_3 P_4

Definition 245

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

Beispiel 246 (Pfad, aber *nicht einfacher* Pfad der Länge 7)

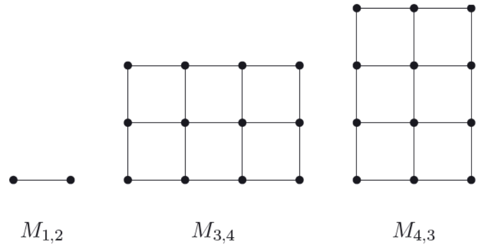
1.10 Gitter

Definition 248

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 249



1.11 Torus

Definition 250

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

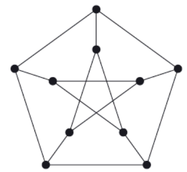
$$\{(i,j),(k,l)\} \in E \iff |(i-k) \bmod m| + |(j-l) \bmod n| = 1$$

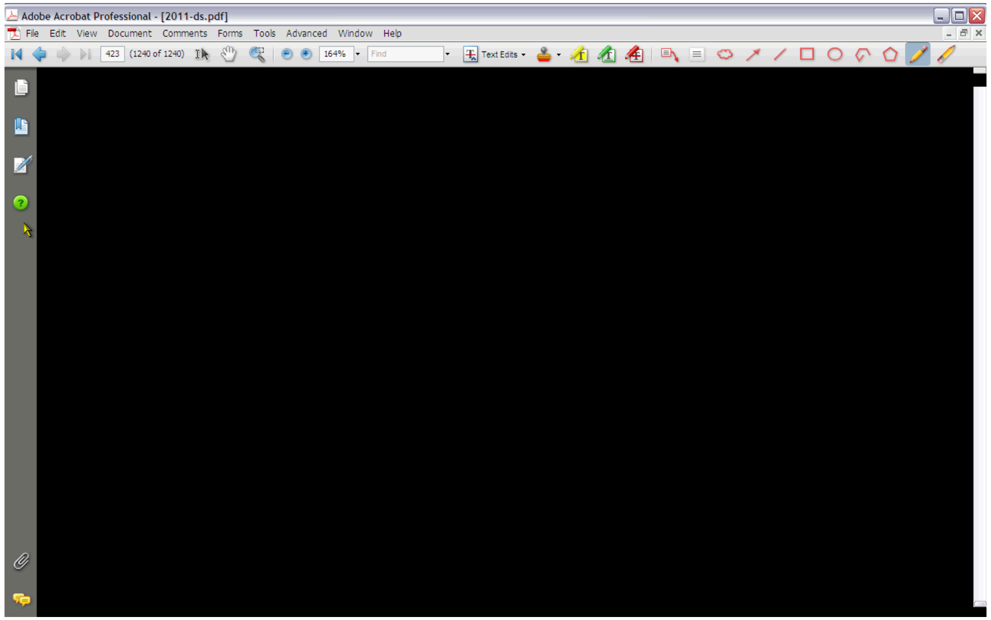
Beispiel 251

1.12 Petersen-Graph

Definition 252

Der folgende Graph heißt Petersen-Graph:




Adobe Acrobat Professional - [2011-ds.pdf]
File Edit View Document Comments Forms Tools Advanced Window Help
423 (1240 of 1240)
164%
Find
Text Edits