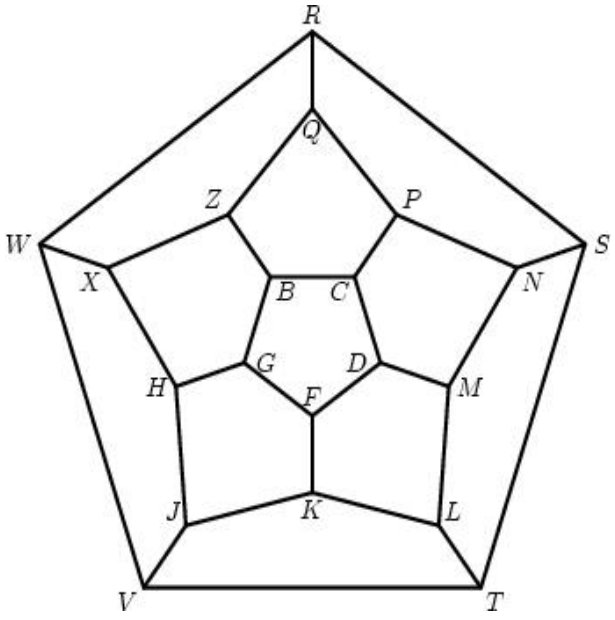
Proseminar
SS 2001

Eulerkreise
Eulerpfade
Hamiltonkreise
Traveling-Salesman-Problem



Inhaltsverzeichnis

Kapitel 1

Einleitung

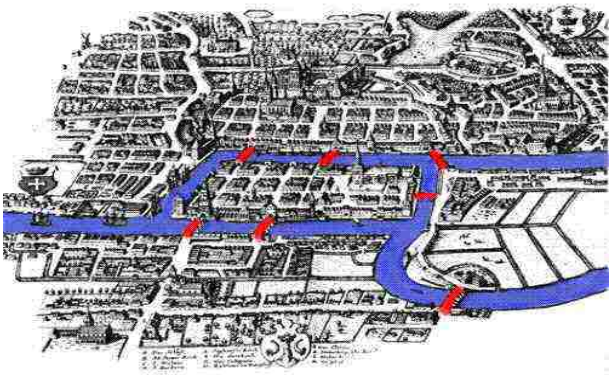


Abbildung 1.1: *Königsberg im 18. Jahrhundert*

Abbildung 1.1 zeigt die Stadt Königsberg im 18. Jahrhundert. Die beiden Arme des Flusses Pregel umfliessen eine Insel, den Kneiphof. Es gibt insgesamt sieben Brücken über den Fluss, die in der Abbildung rot eingezeichnet sind.
Das Problem, das damals gestellt wurde, lautet: „Kann man, beginnend auf einer Landmasse, alle Brücken genau ein Mal überqueren und schliesslich wieder zum Ausgangspunkt zurückkehren?“
Leonard Euler [1] führte dieses Problem 1736 auf ein Graphenproblem zurück. Somit steht das Königsberger Brückenproblem am Anfang der Graphentheorie.

[1] Schweizer Mathematiker, 15. April 1707 in Basel †18. September 1783 in St. Petersburg

Kapitel 2

Eulerweg/-kreis

2.1 Motivation

Warum werden wir uns mit dem Problem der Eulerpfade und Eulerkreise beschäftigen? Handelt es sich dabei um ein rein akademisches Beispiel?
Ein Postbote möchte mit seinem Fahrrad in allen Straßen einer Stadt Briefe austeilen. Da er mit dem Fahrrad unterwegs ist, möchte er keine der Straßen zwei Mal anfahren. Der Postbote wird sich also fragen: „Gibt es eine Möglichkeit, jede Straße genau ein Mal anzufahren, so dass ich wieder am Postamt ankomme?“
Eine weitere Anwendungsmöglichkeit betrifft die Müllabfuhr. Gibt es einen Weg durch eine Stadt, der alle Straßen durchläuft?

2.2 Definitionen

2.2.1 Eulerweg

Sei G = (V,E) ein zusammenhängender Graph.
Ein Weg w heißt **Eulerscher Weg**, wenn w jede Kante aus E genau ein Mal durchläuft.

2.2.2 Eulerkreis

Sei G = (V,E) ein zusammenhängender Graph.
Ein Weg w heißt **Eulerschen Kreis**, falls w jede Kante genau ein Mal berührt und wieder am Ausgangsknoten auskommt.

2.3 Satz von Euler

Leonhard Euler (1707 – 1783) zeigte 1736, dass es keinen Weg gibt, der des Brückenproblem lösen könnte. Er führte das Brückenproblem auf ein Graphenproblem zurück und formulierte folgenden **Satz:**
Sei G = (V,E) ein zusammenhängender Graph. Dann gilt:

1. *G enthält eulerschen Pfad $\Leftrightarrow$ # Knoten mit ungeradem Grad = 2.*
2. *G enthält eulerschen Kreis $\Leftrightarrow$ # Knoten mit ungeradem Grad = 0.*

Beweis: [1]
"$\Rightarrow$": Sei w der zu G gehörende eulersche Kreis.
Bewegt man sich entlang von w, so liefert das Besuchen eines Knoten v den Beitrag 1 zum Grad und das Verlassen von v den Beitrag 1 zum Aussengrad von v. Da w alle Kanten enthält und alle Knoten berührt, folgt daraus die Behauptung.
"$\Leftarrow$":
Man wählt sich einen Knoten v_1 beliebig. Ausgehend von diesem Knoten durchläuft man die Kanten im Graphen in beliebiger Reihenfolge mit der Einschränkung, dass man keine Kante mehr als ein Mal benutzt. Die Voraussetzung, dass alle Knotengrade gerade sind, garantiert, dass es zu jedem Knoten, der auf diesem Weg betreten wird, mindestens eine bislang noch unbenutzte Kante gibt, über die der Knoten wieder verlassen werden kann. Der so konstruierte Weg w_1 muss daher wieder in v_1 enden.
Falls noch nicht alle Kanten des Graphen durchlaufen wurden: Da G zusammenhängend ist, muss es in diesem Fall aber mindestens einen Knoten v_2 in w_1 geben, der zu einer unbenutzten Kante inzident ist. In diesem Knoten startet man das Durchlaufen des Teilgraphen. Dann kann ein Weg w_2 gefunden werden, der in v_2 beginnt und wieder endet. Die beiden Wege w_1 und w_2 können zu einem neuen Weg verschmolzen werden, indem man in w_1 das Teilstck von v_1 zu v_2 durchläuft und dann nach w_2 schwenkt, diesen Weg komplett durchläuft und nach Rckkehr zu v_2 das verbliebene Teilstück w_1 bereist. Enthält der so gefundene Weg noch immer nicht alle Kanten, wählt man analog zu v_2 einen Knoten v_3 und wiederholt obiges Verfahren so lange, bis der Weg alle Kanten des Graphen enthält. □

[1] Es wird nur die erste Aussage des Satzes bewiesen. Der Beweis zum zweiten Teil des Satzes ist analog.

2.4 Die Lösung des Königsberger Brückenproblems

Mit Hilfe des Satzes von Euler kann man nun entscheiden, dass es keinen Weg durch Königsberg gibt, der alle Brücken genau ein Mal überquert und wieder am Ausgangspunkt ankommt.
Die Begründung hierfür ist, dass die Knoten A, B und D den Grad 3 haben (siehe Abbildung 2.1). Da drei Knoten mit ungeradem Grad im Graph sind, gibt es auch keinen Eulerpfad.

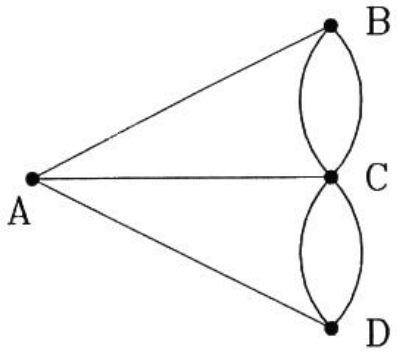


Abbildung 2.1: *Die abstrakte Darstellung der Landkarte von Königsberg*

2.5 Algorithmen

2.5.1 Bestimmung eines Eulerkreises

- Wähle einen Knoten A als Startknoten
- Durchlaufe den Graphen, bis A wieder erreicht ist
- Blende die besuchten Kanten aus Wende dasselbe Verfahren auf die übrigbleibenden Teilgraphen an
- Konstruiere den Eulerschen Kreis aus den Teillösungen

2.5.2 Bestimmung eines Eulerpfades

- wähle einen Knoten mit ungeradem Grad = Startknoten

- Durchlaufe den Graphen, bis der andere Knoten mit ungeradem Grad erreicht ist
- Blende die besuchten Kanten aus
- rufe den Algorithmus zum Finden des Eulerkreises auf den Zusammenhangskomponenten der übrigbliebenden Teilgraphen auf
- Konstruiere den Eulerweg aus den Teillösungen

2.5.3 Komplexität

Da jede Kante beim Algorithmus genau ein Mal berührt wird, handelt es sich bei der Suche nach Eulerpfaden bzw. Eulerkreisen um ein Problem, das sich in linearer Zeit lösen lässt. Die Laufzeit des Algorithmus ist deshalb von der Ordnung O(n).

2.6 Beispiele

Das Haus vom Nikolaus

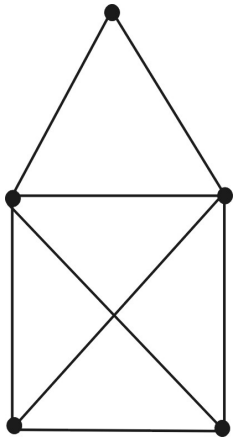


Abbildung 2.2: *Das Haus vom Nikolaus*

Kapitel 3

Das Hamiltonsche Problem

3.1 Das Problem

Ein dem Königsberger Brückenproblem sehr ähnliches, aber ungleich komplexeres Problem wurde im Jahre 1859 von Sir William Hamilton (1805 – 1865) gestellt.
Gibt es einen Weg in einem gegebenen Graphen, der jeden Knoten genau ein Mal passiert und schliesslich wieder am Ausgangspunkt ankommt?

3.2 Definition

Gegeben sei ein Graph.

3.2.1 Hamiltonpfad

Ein Weg, der jeden Knoten des Graphen genau ein Mal berührt, nennt man **Hamiltonschen Pfad**.

3.2.2 Hamiltonkreis

Ein Weg, der jeden Knoten des Graphen genau einmal passiert und schliesslich wieder am Ausgangsknoten ankommt, nennt man **Hamiltonschen Kreis**.

3.3 Algorithmus

Im Gegensatz zum Königsberger Brückenproblem sind beim Hamiltonschen Problem keine hinreichenden Bedingungen bekannt, die charakterisieren, wann ein Graph einen Hamiltonscher Kreis enthält.

Dennoch kann ein Algorithmus angegeben werden, der feststellt, ob in einem Graphen ein Hamiltonscher Kreis enthalten ist.

3.3.1 Hamiltonsche Kreise

Systematisches Ausprobieren aller Möglichkeiten, um den Graphen zu durchlaufen:

- Wähle und markiere einen Knoten A (Startknoten)
- Führe Tiefensuche aus
- Falls Nachbarknoten nicht markiert, markiere ihn und führe erneut Tiefensuche aus
- Falls alle Nachbarknoten markiert, führe Backtracking [1] aus. Verfolge den Ast des Suchbaums, für den gilt:
- Länge dieses Astes = # Knoten und Blatt des Suchbaumes = Nachbarknoten von A

3.3.2 Komplexität

Dieser Algorithmus hat eine Laufzeit der Ordnung O(n!). [2] Als Begründung dient das genauere Betrachten des entstehenden Suchbaums. Seien im Graphen n Knoten. Ein Pfad durch den Graph hat die maximale Länge n, da sonst die Voraussetzung, dass jeder Knoten genau ein Mal besucht wird, verletzt wird. Des weiteren muss jeder Pfad im Suchbaum paarweise disjunkt zu einem anderen Pfad sein.
Daraus kann man nun folgern: Für die erste Stelle im Pfad kommen n Knoten in Frage (man kann sich den ersten Knoten frei wählen). An der zweiten Position im Pfad können noch (n-1) Knoten stehen, an der dritten (n-2) usw. Wenn wir den Pfad bis zum vorletzten Knoten durchlaufen haben, bleibt für den letzten Knoten noch genau eine Möglichkeit.
Also: $\prod_{k=1}^{n} k = \text{n!}$

[1] Wenn eine partielle Lösung eines Problems sukzessive erweitert werden kann, um eine vollständige Lösung zu erzeugen, kann eine rekursive Implementierung geeignet sein.
Das Finden eines Hamiltonschen Kreises kann mit Hilfe eines Suchbaumes beschrieben werden, dessen Knoten den partiellen Lösungen entsprechen. Eine Bewegung im Baum abwärts entspricht somit der Konstruktion einer vollständigeren Lösung, während eine Bewegung aufwärts im Baum der Rückkehr zu einer vorher erzeugten partiellen Lösung entspricht.

[2] worst case running time

3.4 Der Algorithmus an einem Beispiel

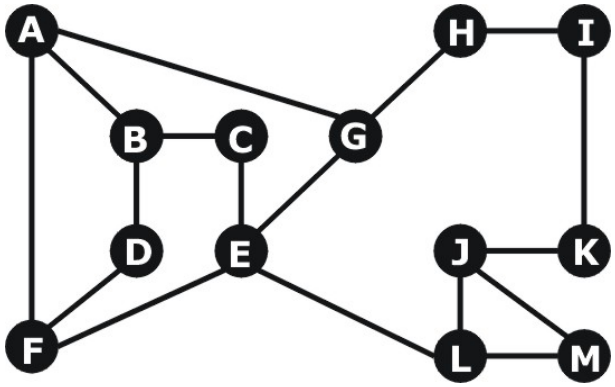


Abbildung 3.1: *Der Ausgangsgraph*

Abbildung 3.1 zeigt den Graphen, auf den der Algorithmus angewandt wird. Abbildung 3.2 zeigt den zum Graphen gehörigen Suchbaum. Jeder Pfad von der Wurzel zu einem Blatt entspricht einem Weg im Graphen.
Der erste geprüfte Pfad im Beispiel ist A B C E F D. Zu diesem Zeitpunkt sind alle zu D benachbarten Knoten markiert. Also wird die Markierung für D gelöscht. Anschließend kehrt der Algorithmus zu F zurück und prüft, ob F außer D noch andere nicht markierte Knoten hat. Da dies nicht der Fall ist, wird für F die Markierung gelöscht. Der aktuelle Knoten ist nun E. Danach besucht der Algorithmus den Knoten G, anschließend H u.s.w. Dies ergibt schließlich den Suchbaum von Abbildung 5.
Für die Entscheidung, ob G einen Hamiltonschen Kreis enthält, genügt es, die Blätter des Suchbaumes zu betrachten. In einem Knoten des Suchbaumes wird die Länge des Pfades abgespeichert. Hat man nun den Suchbaum entwickelt, sucht man sich die Blätter, deren abgespeicherte Tiefe der Anzahl der Knoten im Graph entspricht.
Endet ein Ast, dessen Länge maximal ist, mit einem zu A benachbarten Knoten, so repräsentiert dieser Ast einen Hamiltonschen Kreis.

3.4.1 Verbesserung des Algorithmus

Knoten in bestimmter Reihenfolge

Es gibt keine allgemeingültige Verfahren, die den obenstehenden Algorithmus verbessern können. Allerdings kann man unter bestimmten Umständen gewisse Forderungen an die Suchreihenfolge stellen.
Will man in einem Graph einen Hamiltonschen Kreis finden, so kann man

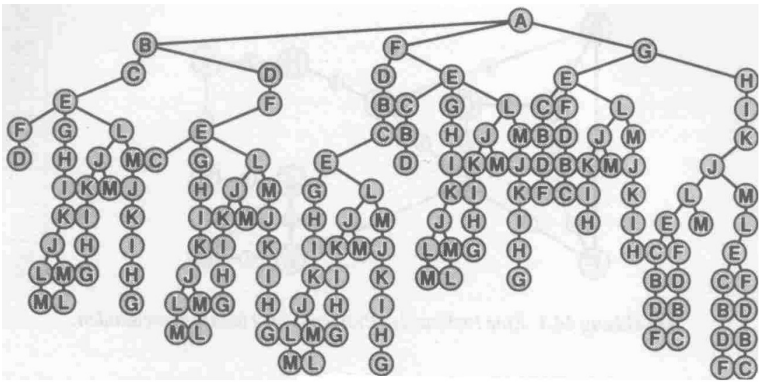


Abbildung 3.2: *Der Suchbaum zu Abbildung 3.1*

fordern, dass eine der Nachbarknoten immer nach allen anderen Knoten besucht werden soll.
Im Beispiel von Abbildung 3.1 muss der Knoten C vor dem Knoten B im Suchbaum erscheinen, denn es gibt keine Kante von A nach C. Der Algorithmus muss am Knoten B nur aufgerufen werden, wenn sich C bereits im Suchbaum auf dem Pfad befindet.
Allerdings ist dieses Verfahren nicht immer anwendbar. Wenn wir annehmen, dass kein Hamiltonscher Zyklus, sondern nur ein Hamiltonscher Pfad gefunden werden muss, so kann das obige Schema nicht angewandt werden, da wir nicht im Voraus wissen, ob ein Pfad zu einem Zyklus führen wird.

Vermeiden von Graphzerlegungen

Man kann beobachten, dass manche Pfade den Graph so zerlegen, dass die nicht markierten Knoten nicht mehr verbunden sind. Es kann kein Zyklus gefunden werden.
In Abbildung 3.1 kann kein Hamiltonscher Kreis mit den Knoten A B F E beginnen, da dieser Pfad den Graph in “zwei Hälften“ teilt.

3.5 Beispiele

Hamiltonkreis in einem Dokedaeder

Ein Hamiltonkreis durch einen Graph mit 20 Knoten zeigt Abbildung 3.3.

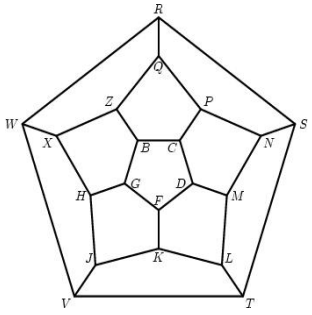


Abbildung 3.3: Hamiltonscher Kreis in einem Dodekaeder

Das Problem des Rösselsprunges auf dem Schachbrett:

Hierbei handelt es sich um das Problem, mit einem Springer alle Felder eines Schachbretts genau einmal zu erreichen und wieder zum Ausgangsfeld zurückzukehren.
Der Rösselsprunggraph R(8) wird wie folgt definiert: Jedem der $8 \times 8 = 64$ Felder lässt man eine Ecke von R(8) entsprechen und verbindet zwei Ecken durch eine Kante genau dann, wenn zwischen den entprechenden Feldern ein Springerzug möglich ist. Das Rösselsprungproblem zu lösen ist äquivalent damit, in R(8) einen Hamiltonkreis zu finden.
Abbildung 3.4 zeigt eine mögliche Lösung des Problems.

58	43	60	37	52	41	62	35
49	46	57	42	61	36	53	40
44	59	48	51	38	55	34	63
47	50	45	56	33	64	39	54
22	7	32	1	24	13	18	15
31	2	23	6	19	16	27	12
8	21	4	49	10	25	14	17
3	30	9	20	5	28	11	26

Abbildung 3.4: Eine von Euler gefundene Lösung des Rösselsprungproblems

Kapitel 4

Travelling Salesman-Problem (TSP)

4.1 Definition des Problems

Ein Handlungsreisender soll nacheinander n Städte besuchen. Es soll keine Städte geben, die nicht besucht werden können und von jeder Stadt aus soll jede andere Stadt angefahren werden können. Ausserdem soll jede Strasse eine bestimmte Länge haben. In welcher Richtung die Strasse befahren wird, ist nicht von Bedeutung.
Wir stellen also drei Bedingungen an den Graphen:
Der Graph ist

1. vollständig
2. gewichtet
3. ungerichtet

Auf seiner Rundreise will der Handelsreisende alle Städte besuchen. Am Ende der Reise soll er wieder am Ausgangspunkt ankommen. Gesucht ist dabei eine kürzeste mögliche Route.
Das TSP ist äquivalent mit der Suche nach dem kürzesten Hamiltonschen Kreis in einem ungerichteten Graphen mit einem gegebenen Startknoten.

4.2 Approximation der TSP-Tour mit Hilfe des MST

Da jeder bisher bekannte Algorithmus eine Laufzeit der Ordnung O(2^n) hat, [1] kann man bereits bei Knotenzahlen von n = 50 mit heutigen Supercomputern keine Lösung mehr berechnen. Deswegen sind Algorithmen vonnöten, die eine TSP-Tour möglichst gut approximieren.
Eine Möglichkeit besteht in der Approximation der TSP-Tour mit einem Minimum Spanning Tree (MST). Dafür muss jedoch die Dreiecksungleichung gelten. [2]
Der Handlungsreisende bewegt sich aussen um den MST herum. Da ein MST alle Knoten des zugehörigen Graphs beinhaltet und bei diesem Rundweg um den MST jeder Knoten besucht wird, eignet sich dieses Verfahren zur Approximation der TSP-Tour.
Für die erheblich schnellere Berechnung der TSP-Tour muss in Kauf genommen werden, dass nicht die optimale TSP-Tour berechnet wird. Das zeigt das folgende Theorem.

4.2.1 Theorem

Die TSP-Tour um einen minimalen Spannbaum ist weniger als doppelt so lange wie eine optimale TSP-Tour.
Gegeben: Ein Graph G, eine optimale TSP-Tour T*, eine approximierte TSP-Tour T und ein minimaler Spannbaum MST.
Beweis:
Streiche Kante k zwischen zwei Punkten aus G $\Rightarrow$ Spannbaum des Graphen G.

$$
\begin{aligned}
&\Rightarrow \quad |MST| \leq |\mathrm{T}^* \backslash k| < |T*| \\
&\Rightarrow \quad |MST| < |\mathrm{T}^*| \qquad (4.1) \\
&\qquad \text{Der Rundweg um den Spannbaum besitzt die Länge } 2 \cdot |MST| \\
&\Rightarrow \quad 2 \cdot |MST| = |T| \\
&\Rightarrow \quad |MST| = \frac{1}{2} \cdot |T| \qquad (4.2)
\end{aligned}
$$

(4.2) in (4.1): $\frac{1}{2} \cdot |T| < |\mathrm{T}^*| \Leftrightarrow |T| < 2 \cdot |\mathrm{T}^*|$ □

[1] Das TSP ist NP-vollständig.
[2] Eine Aussage, die für Graphen im Zweidimensionalen trivial ist. Die Dreiecksungleichung gilt auf der Ebene.

Anhang A

Hilfsalgorithmen

A.1 (Minimale) Spannende Bäume

A.1.1 Definition: Spannender Baum

Ein **spannender Baum** (*engl.: spanning tree*) eines zusammenhängenden Graphen ist ein Teilgraph von G, der alle Knoten von G enthält.

A.1.2 Definition: Minimaler Spannender Baum

Ein **minimaler spannender Baum** (*engl.: minimum spanning tree, MST*) eines zusammenhängenden Graphen V ist ein spannender Baum von V, dessen Summe der Kantenkosten minimal ist.

A.1.3 Algorithmus von Kruskal

- Initialisiere einen Wald mit n isolierten Knoten
- Initialisiere einen Heap mit allen Kanten
- **do** entferne die billigste Kante aus dem Heap
- falls durch Einfügen von (x,y) zwei Zusammenhangskomponenten verbunden werden, füge (x,y) in den Wald ein, bis
- Wald besteht nur aus einem Baum

A.2 Tiefensuche

A.2.1 Algorithmus

- Initialisiere einen Stack
- Wähle einen Knoten A und lege ihn auf den Stack
- Besuche einen zu A adjazenten, noch nicht markierten Knoten, markiere ihn und schreibe den Knoten auf den Stack
- Wende dasselbe Verfahren auf den Nachbarknoten an
- Falls es keine nicht markierten Nachbarknoten zum aktuellen Knoten gibt, dann entferne einen Knoten vom Stack
- Wende das Verfahren so lange an, bis der Stack leer ist.

Anhang B

Literatur

- A. Brandtstädt, Graphen und Algorithmen, Teubner-Verlag, Stuttgart, 1994 (S. 40 ff.)
  ⇒ *Mathematische Beschreibung von Eulerkreisen, etwas für Theorieliebhaber*

- A. Aho, J.E. Hopcraft und J.D. Ullman, Data Structures and Algorithms, Addison-Wesley Publishing Company, Reading (Massachusetts), 1983
  ⇒ *Ausführlich und gut lesbar*

- R. Sedgewick, Algorithmen, Addison-Wesley, Bonn, 1991 (S. 703 ff.)
  ⇒ *leicht verständlicher Überblick über erschöpfendes Durchsuchen in Graphen*

- T.H. Cormen, C.E. Leiserson, R.I. Rivest, Introduction to Algorithms, The MIT Press, Cambridge (Massachusetts), 1990
  ⇒ *leicht verständlich, behandelt Approximation für TSP*

- Duden: Informatik, Dudenverlag, Mannheim, 1993
  ⇒ *Ein guter Ausgangspunkt für die Suche nach Begriffen der Informatik.*

- A. Steger, Diskrete Strukturen - Band 1, Springer-Verlag, Berlin u.a., 2001 (S. 70 ff.)
  ⇒ *Kurze mathematische Einführung in Hamilton- und Eulertouren*