Vorlesung 1

Eine notwendige Bedingung, die ein Graph erfullen muss, um einen geschlossenen Kantenzug zu besitzen, der jede Kante genau einmal enthalt, ist dass jeder Knoten geraden Grad haben muss (wenn der Kantenzug nicht geschlossen ist, kann man maximal 2 Knoten ungeraden Grades zulassen).

* Eulershe Graph – Kantenzüge die jede Kante eines Graphen genau einmal durchlaufen

Algorithmus zum Finden eines eulerschen Kantenzugs

* Fluery: 1883 eleganten, jedoch ineffizienten

Hierholzer – effizienter als Fluerys Algorithmus:

* Man wählen einen beliebigen Anfangsknoten v und folge einem Kantenzug (ohne Kantenwiederholung) von diesem Knoten aus, bis man wieder bei v ist
* Es geht nicht immer, denn eine sogennanten *Brücke\** wäre ein Problem

Brücke – eine Kante, deren Entfernung den Graphen in zwei oder mehr separate Komponente aufteilt, wenn sie entfernt wird. (deren Entferung dazu führt, dass der Graph nicht mehr zusammenhängend ist)

* Ist unmöglich, an einen anderen Knoten als v zu stoppen, da der gerade Grad aller Konten sicherstellt, dass wenn der Kantenzug einen anderen Knoten erreicht, eine unbenutzte Kante diesen Knoten übrig bleibt
* Es ist nicht möglich, den Algorthmus auf Graphen anweden, die Knoten ungerades Grades haben, weil der gerade Grad aller Knoten sicherstellt, dass wenn der Kantenzug einen anderen Knoten w erreicht, eine unbenutzte Kante von w ubrig bleibt.

Ein Graph G ist ein Paar (V(G), E(G)) disjunkter und endlicher Mengen, wobei E(G) aus Elementen besteht, die 2-elementige Teilmengen von V(G) sind und wir stets annehmen

* Adjazente(benachtbare) Knoten: sind Knoten die miteinander verbunden sind
* Inzidente Kanten: sind Kanten die aus einem Knoten a austreten und in einem Knoten b eintreten.
* Grad eines Knotens: die Anzahl der Kanten, die mit diesem Knoten verbunden sind

Vorlesung 2

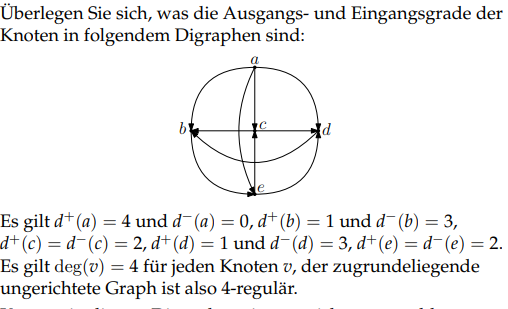
Ein einfacher Graph ist ein Graph ohne Schleifen oder Mehrfachkanten zwischen einem belibigen Paar von Knoten

Ein ungerichteten, einfachen und vollständigen Graphen mit n Knoten bezeichnet man mit Kn.

Gerichtete Graphen

* Eine Kante „ab“ heißt gerichtet, falls a der Startknoten und b der Endknoten ist. Eine gerichtete Kante ist vom Startknoten zum Endknoten gerichtet. Ein gerichteter Graph (directed graph) oder Digraph ist ein Graph, dessen Kanten gerichtet sind.
* Der Eingangsgrad von “v”: die Anzahl der Vorgänger von “v“, also die Anzahl der Kanten, deren Endknoten “v“ ist. d −(v)
* Der Ausgangsgrad von „v“: die Anzahl der Nachfolger von „v“, also die Anzahl der Kanten, deren Anfangsknoten v ist d +(v)
* Der Grad von „v“ ist die Summe die Eingangs- und Ausgangsgrade von „v“, also

deg(v) = d −(v) + d +(v).

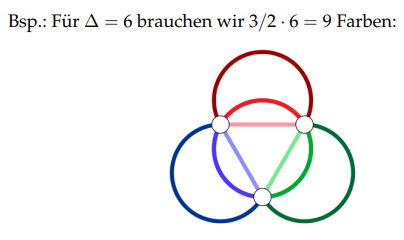
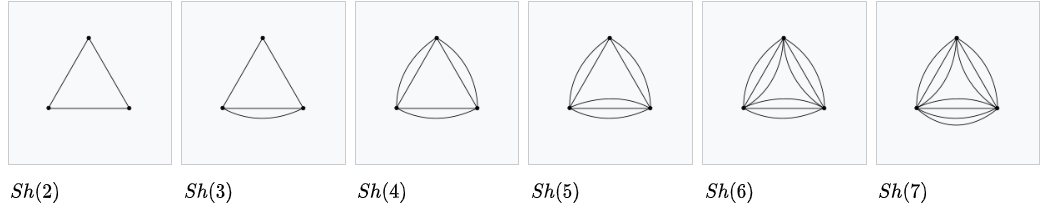


Multigraphen

* Ein Graph, die mindestens zwei Knoten enthalt die durch mehrere Kanten verbunden sind

Die Schannon-Multigraph: ein Multigraph mit 3 Ecken, die mit jeweilse mit der gleichen Anzahl von Kanten verbunden sind oder daruber hinaus eine weitere zusatzliche Kante besitzt.

* Jeder Multigraph mit Maximalgrad ∆ hat einen Kantenfarbung mt hochstens 3∆/2 Farben

Vollständige Graphen (complete graphs)

* einen einfachen Graph, in dem jeder Knoten mit jedem anderen Knoten durch eine Kante vebunden sind
* die Anzahl der Kanten des vollstandigen Graphen Kn entspricht der Dreieckszahl



* der vollstandige Graph Kn, n > 1 hat eine absteigende Gradfolge

Isomorphie von Graphen

* die Eigenschaft zweier Graphen, strukturell gleich zu sein
* zwei Graphen G1 = (v1, e1) und G2 = (v2, e2) heißen isomorph gdw. es eine bijektive Abbildung φ: v1 → v2 gibt (φ(phi) nennen wir einen Isomorphismus von G1 auf G2). Die Funktion nimmt einen Knoten und sieht welche Knoten aus dem anderen Graphen dargestellt wird an
* zwei Graphen sind gdw. Isomorph, wenn eine Graph aus dem anderen Graphen durch Unbennenung der Knoten hervorgeht
* isomorphe Graphen haben die gleichen graphentheoretischen Eigenschafen
* eine Invariante ist ein Objekt f(G), welches mit einem Graphen G assoziiert wird

Aus dem zweiten Seminar

* ein Graph kann in verschiedenen Formen existieren, die die gleiche Anzahl von Komponenten (Knoten und Kanten) und auch die gleiche Kantenkonnektivität aufweisen

Weg

* Kantenzug – eine Folge von Knoten mit Kanten. Ein Kantenzug(Weg-path), bei dem die Kanten paarweise verschieden sind, heißen *Kantenzug ohne Kantenwiederholung*

Kreis

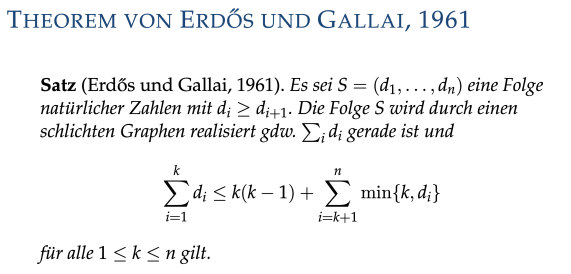
* Wenn ein Graph G einen geschlossenen Kantenzug K, enthält, in dem keine Kante von K mehrfach vorkommt, dann enthält G auch einen Kreis

Realisierten Graphen

* Eine *absteigende Gradfolge* von G ist eine absteigende Folge der Knotenfolge von G.

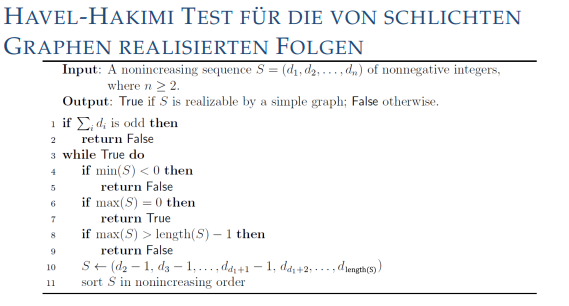
3, 2, 2, 2, 2, 1, 1, 1

* Sei S eine absteigende Folge natürlicher Zahlen. S ist *graphisch* falls es die absteigende Gradfolge irgendeines Graphen ist. Falls G ein Graph mit absteigender Gradfoge S ist, dann S von G *realisiert* wird.



Satz von Havel und Hakimi

* Ein mathematischer Algorithmus, mit dem festgestellt werden kann, ob eine Folge von ganzen Zahlen eine Gradfolge einem einfachen Graphen ist.
* Es funktioniert, indem er den Knoten mit dem höchsten Grad aus der Sequenz entfernt und von den Graden der nächsthöheren Knoten in der Sequenz eine Zahl subtrahiert. Dieser Vorgang wird so lange wierderholt, bis eine Gradfolge erhalten wird, die einem einfachen Graphen enspricht, oder bis festgestellt wird, dass die Folge keine gültige Gradfolge ist (wenn wir eine Folge nur mit Nullen erhalten haben – eine gültige Gradfolge, ansonsten wenn eine negative Zahl in der Folge haben – können wir keinen Graphen mit der gegebener Gradfolge erstellen)



Eine Gradfolge S nicht graphisch ist

* Der Grad einen Knoten kleiner oder gleich als n – 1 sein muss
* Jeder Graph mit mindestens 2 Knoten enthält 2 Knoten gleichen Grades
* Kann nicht eine ungerade Anzahl von Knoten ungeraden Grades enthälten

Operation mit Graphen

* Disjunkte Vereinigung

Es seien G1 = (V1, E1) und G2 = (V2, E2) zwei Graphen mit disjunkten Knotenmengen (die Durchschnitt zwischen Knoten nicht die leere Menge ist (V1 ∩ V2 = ∅)). Die disjunkte Vereinigung und der beiden Graphen ist der Graph G1 ∪ G2 = (V1 ∪ V2, E1 ∪ E2).

* Vereinigung

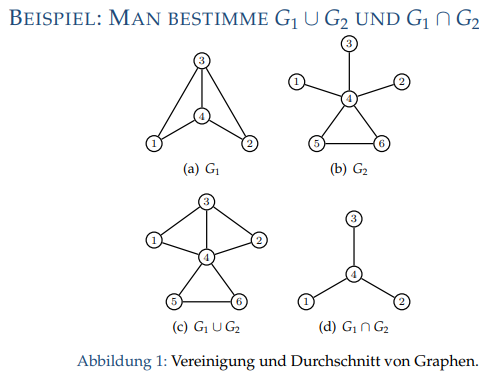
Vereinigung zweier Graphen besteht darin, alle Knoten und Kanten beider Graphen zu kombinieren, um einen neuen Graphen zu erstellen

Im Allgemeinen ist die Vereinigung zweier Graphen G1 = (V1, E1) und G2 = (V2, E2) definiert als G1 ∪ G2 = (V1 ∪ V2, E1 ∪ E2), mit V1 ⊆ V2 oder V2 ⊆ V1 oder V1 = V2 oder V1 ∩ V2 = ∅.

* Durchschnitt

Durchschnitt zweier Graphen besteht darin, nur die gemeinsamen Knoten und Kanten beider Graphen zu behalten und einen neuen Graphen zu erstellen. Die Knotenmenge der resultierenden Graphen enthält nur diejenigen Knoten, die in beiden urprünglichen Graphen vorhanden sind. Die Kantenmenge besteht aus den Kanten, die sowohl im ersten als auch im zweiten Graphen vorhanden sind

Es seien G1 = (V1, E1) und G2 = (V2, E2) zwei Graphen. Der Durchschnitt der Graphen G1 und G2 ist (∅, ∅) oder der Graph G1 ∩ G2 = (V1 ∩ V2, E1 ∩ E2)

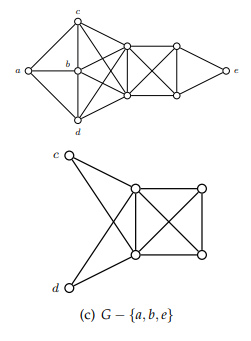


* Symmetrische Differenz

Die symmetrische Differenz zweier Graphen besteht darin, alle Knoten und Kanten beider Graphen zu kombinieren, jedoch ohne diejenigen, die in beiden Graphen vorkommen. Die Knotenmenge des resultierenden Graphen enthält die Knoten, die in mindestens einem der beiden ursprünglichen Graphen vorhanden sind, aber nicht in beiden

* Entfernen von Knoten und Kanten

Beim Entfernen von Knoten und Kanten in einem Graphen werden bestimmte Knoten oder Kanten aus dem Graphen entfernt, um einen neuen Graphen zu erzeugen. Das Entfernen eines Knotens *e* beinhaltet auch das Entfernen aller Kanten, de mit diesem Knoten verbunden sind. Es werden nicht die Endknoten von *e* entfernt



* Fusion

Die Fusion zweier Knoten in einem Graphen besteht darin, diese beiden Knoten zu einem einzigen Knoten zu verschmelzen (kombinieren), wobei alle eingehenden und ausgehenden Kanten beider Knoten erhalten bleiben

* Kontraktion

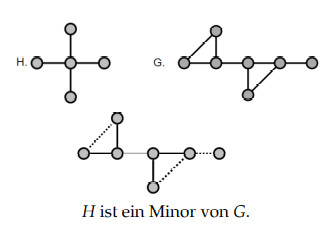
Die Kontraktion einer Kante in einem Graphen besteht darin, diese Kante zu entfernen mit der ansließenden Fusion und die beiden Knoten, die durch diese Kante verbunden sind, zu einem neuen Knoten zu verschmelzen

Das Komplementgraph

* Der Komplementgraph eines Graphen G = (V, E) hat die gleichen Knoten wie der Ursprungsgraph und die Kanten genau jede Kanten, die nicht in G vorhanden sind.
* ein schlichter Graph, der isomorph ist zu seinem Komplementgraphen, heißt *selbstkomplementär (*die beiden Graphen haben die gleiche Anzahl von Knoten und Kanten und jeder Knoten im Graph 𝐺 ist mit genau den Knoten verbunden, mit denen er im Graphen 𝐺(komplemenar) nicht verbunden ist.*)*

Minor

* ein Graph H heißt Minor eines Graphen G, falls H isomorph ist zu einem Graphen welcher als eine Folge von Knotenkontraktionen, Kantenloschungen und Knotenlöschungen aus G entsteht



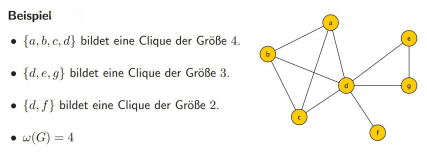
Vorlesung 3

Teilgraph

* ein aufspannender Teilgraph eines Graphen G ist ein Teilgraph, der alle Knoten von G enthält, der durch Enfernen von Kanten aus dem urprünglichen Graphen G entsteht, während alle verbleibende Knoten untereinander verbunden bleiben
* ein induzierter Teilgraph eines Graphen G entsteht, indem man eine Teilmenge der Knoten von G auswählt und alle Kanten behält (keeps all edges), die nur zwischen den ausgewähten Knoten verlaufen.

Clique

* ein Clique ist ein induzierter Teilmenge von Knoten eines Graphen G, der ein vollständiger Graph (schlichter Graph, in dem jede Knoten mit alle anderen Knoten verbunden wird) ist.



Zusammenhang

* zwei Knoten in einem Graphen sind *verbindbar*, wenn es einen Pfad zwischen ihnen gibt

ein Graph heißt zusammenhängend, gdw jedes Paar von Knoten in G verbindbar ist

* jede zusammenhängende Graph mit n Knoten hat mindestens n-1 Kanten

Um eine Äquivalenzrelation zu sein, muss eine Relation die folgenden drei Bedingungen erfüllen:

* Reflexivität: jedes Elements steht in Bezieuhung zu sich selbst. Das bedeutet, dass für jedes Element a in der Menge die Relation aRa gilt
* Symmetrie: Wenn ein Element a in Beziehung zu einem Element b steht, dann steht b auch in Beziehung zu a
* Transitivität: Wenn ein Element a in Beziehung zu einem Element b steht und b in Beziehung zu einem Element c steht, dann steht a auch in Beziehung zu c.

Ein nicht vollständiger, zusammenhängender Graph mit > k Knoten heißt k-zusammenhängend (k-connected), wenn das Löschen beliebiger < k Knoten aus G stets einen zusammenhängenden Graphen liefert.

Bäume

* ein Graph heißt Wald gdw. er keinen Kreis enthält. (Kreisfreie Graphen nennt man auch azyklisch).
* Ein Baum ist ein Graph die keinen Kreis enthält und auch zusammenhängend ist

Charakterisierung von Bäumen

Für einen Graphen G=(V, E) mit |V| = n sind die folgenden Aussagen äquivalent:

* G ist ein Baum
* Je zwei Knoten von G sind durch genau einen Weg verbunden
* G ist zusammenhängend und hat genau n-1 Kanten
* G ist kreisfrei und hat genau n – 1 Kanten, aber für je zwei nicht adjazente Knoten v, w von G enthält G’’= (V, E ∪ {{v, w}}) genau einen Kreis.

Gerichtete Graphen

* Ein gerichterter Graph ist ein Paar G = (V, A) bestehend aus dem Mengen V, der Menge von Knoten und A der Menge der gerichteten Kanten manchmal auch Pfeile, die aus geordneten Paaren (v, w) besteht
* Fur eine gerichtete Kante heißt v der Startknoten und w der Endknoten von e
* Man kann ungerichtete Graphen als gerichtete Graphen betrachten, bei denen die Relation A symmetrisch ist]
* G heißt stark zusammenhängend gdw. es für je zwei Knoten v, w ∈ V einen gerichteten Weg von v nach w gibt
* Für einen gerichteten Graph, die Summe den Eingangsgraden ist gleich mit die Summe den Ausgangsgraden
* Ein [gerichteter Graph](https://de.wikipedia.org/wiki/Gerichteter_Graph) heißt *(schwach) zusammenhängend*, falls der zugehörige ungerichtete Graph (also der Graph, der entsteht, wenn man jede gerichtete Kante durch eine ungerichtete Kante ersetzt) zusammenhängend ist.

Darstellung von Graphen in Computern

graph6

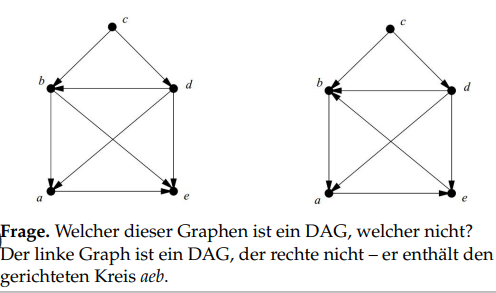
* Benutzt Bitvektoren (Liste von Bits) und ASCII mit Dezimalcode von 63 (?) bis 126 (~), um Graphen zu repräsentieren
* Es wurde entwikelt, um Graphen effizient zu speichern und zu übertragen, insbesondere für kleine bis mittelgroße Graphen
* wird benutzt, um einfache, ungerichtete Graphen mit maximal 2^36 – 1 darzustellen

Schritten

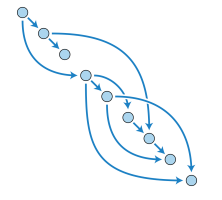
* erstelle die Adjazenzmatrix
* Nehmen wir die untere Dreiecksmatrix ohne Hauptdiagonale, die wri nun zeilenweise aufschreiben
* Berechnen die numerischen Wert N(n) = n + 63 und R(x) = R(010010 100100) = 63+18 63+36 = 81 99
* Zusammenfassend is dieser Graph 68 81 99-> Ascii DQc

DAG (directed acyclic graph)

* ein gerichteter Graph G heißt DAG gdw. G keinen gerichteten Kreis der Länge großer oder gleich 2 enthält



Ein DAG ist ein gerichteter Graph mit einer *topologischen Sortierung,* d.h einer Reihenloge der Knoten derart, dass jede gerichtete Kante von einem früheren zu einem späteren Knoten verläuft. Ein gerichteten Graph G heißt DAG gdw. G keinen gerichteten Kreis der Länge >= 2 enthält.



Topologisches Sortieren

Es sei V eine Menge. Eine binäre Relation heißt Ordnung auf V gdw. folgende Eingenschaften gelten: Reflexivität, Antissymmetrie, Transitivität

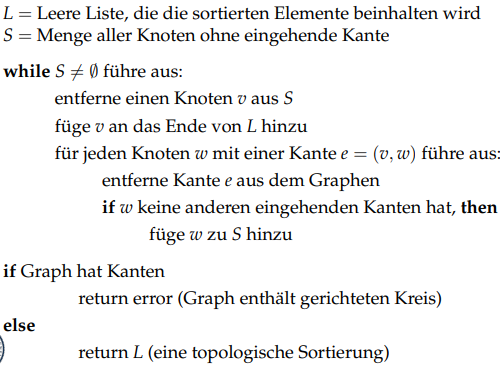
Algorithmus von Kahn

Der Algorithmus von Kahn ist ein Algorithmus, der in der Graphentheorie verwendet wird, um eine topologische Sortierung eines gerichteten azyklischen Graphen (DAG) zu finden. Die Knoten wird in der gleichen Reihenfolge gewählt wie in einer potentiell existierende topologischen Sortierung.

Der Algorithmus beginnt alle Knoten ohne eingehende Kante in einer Warteschlange einzufügen. Dann nimmt der Algorithmus für jede Knoten in der Warteschlange den Knoten aus der Warteschlange und fügt ihn in einer Liste, die die sortierten Elemente beihalten wird. Danach für den Knoten die aus der Wartenschlange entfernt wird, besucht man alle Nachbarn, entfernen die zusätzliche Kanten die diese zwei Kanten verbinden und wenn die Nachbarn keine eingehenden Kanten hat, dann wird den Knoten in der Warteschlange eingefügt.

Dieser Vorgang wird so lange fortgesetzt, bis alle Knoten sortiert sind oder es keine Knoten ohne einghende Kanten mehr gibt.

Am Ende des Algorithmus enthält die Liste die topologische Sortierung des Graphen, sofern der Graf ein gerichteter azyklischer Graph ist. Wenn der Graph einen Zyklus enthält, wird keine vollständige topologische Sortierung möglich sein. Der Komplexitat O(|v|+|e|).



Vorlesung 4 – silde 58 Hamiltonweg

Äquivalente Aussagen zu Bäumen

Es sei G=(V,E) ein gerichteter Graph. Die folgenden Aussagen sind äquivalent

* G ist ein Baum
* G ist ein minimal zusammenhängend (wenn man eine beliebige Kante e entfernt, ist der resultierende Graph nicht mehr zusammenhängend)
* G ist maximal kreisfrei (wenn man eine beliebige Kante hinzufügt, ist der entstehende Graph nicht mehr kreisfrei)

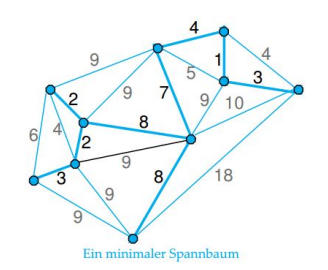
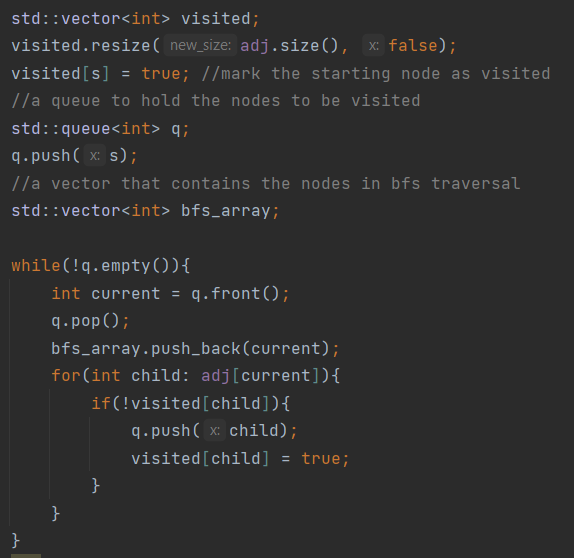
Minimale Spannbäume

* ein aufspanneder Teilgraph des Graphen G ist ein Teilgraph, der alle Knoten von G enthält, der durch das Entfernen von Kanten aus dem ursprünglichen Graph entsteht, während alle Knoten untereinander verbunden bleiben
* Spannbaum oder aufspannender Baum (spanning tree) heißt ein Teilgraph des Graphen G, der alle Knoten von G beinhaltet und auch ein Baum ist
* Ein minimaler Spannbaum beschreibt den Teilgraph, der die Kanten beinhaltet, die die kostengünstige Verbindung aller Knoten innerhalb des Graphen entspricht

Kruskal Algorithmus

* Gehört zur Gruppe der Greedy Algorithmen. Er kann bei zusammenhängenden, gewichteten Graphen angewendet werden, um den minimalen Spannbaum zu ermitteln.
* Wird ein neuer Graph erstellt, der zunächst nur die Knoten des urprünglichen Graphen ohne die jeweiligen Kanten enthält. Dann werden die Kanten mit minimaler Gewicht eingefügt. Diesen Vorgang wiederholen sich nun für die nächsten Kanten mit das geringste Gewicht, die keine Kreise erzeugen. Der Algorithmus endet, wenn man n – 1 Kante ausgewählt hat. Aufwand einer naiven Implementierung: O(m\*logn+n^2)

BREADTH-FIRST-SEARCH

Vorlesung 4

Suchalgorithmen

Breitensuche (Breadth-first search) ist ein Suchalgoritmus, der einen Graphen oder ein Baumdatenstruktur durchläuft und dabei alle Knoten auf derselben Ebene untersucht, bevor er zum nächsten Ebene weitergeht.

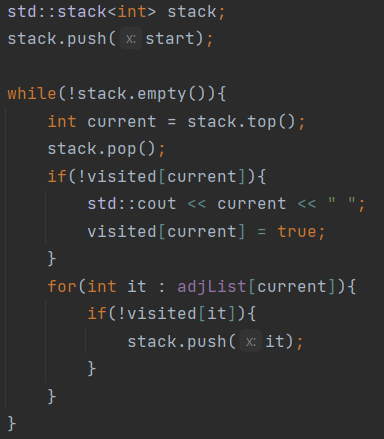
Während des Suchprozesses wird eine Warteschlangen - Datenstruktur (queue) verwendet, um die Reihenfolge, in der die Knoten besucht werden, einzuhalten.Der Algorithmus beginnt damit, den Wurzelknoten in der Warteschlange einzufügen und ihn als besucht zu markieren. Dann nimmt der Algorithmus für jede Knoten in der Warteschlange den Knoten aus der Warteschlange und besucht alle seine Nachbarn, die noch nicht besucht wurden. Jeder dieser nicht besuchten Nachbarn wird in der Warteschlange aufgenommen und als besucht markiert.

Dieser Vorgang wird so lange fortgesetzt, bis alle Knoten, die von Startknoten aus erreichbar sind, besucht worden sind.

Teifensuche (Depth-first search) ist ein Suchalgorithmus, der einen Graphen oder eine Baumdatenstruktur durchläuft und dabei jeden Zweig so weit wie möglich erkundet (exploreaza fiecare ramura), bevor er zurückgeht.

Während des Suchprozesses wird eine Stack-Datenstruktur verwendet, um die Reihenfolge der besuchten Knoten einzuhalten(a mentine). Der Algorithmus beginnt damit, den Wurzelknoten auf dem Stapel einzufügen und ihn als besucht zu markieren. Dann untersucht der Algorithmus für jeden Knoten am oberen Ende des Stapels seine nich besuchten Nachbarn und wählt einen aus, der besucht werden soll. Der ausgewählte Nachbar wird auf dem Stapel entfernt und als besucht markiert.

Dieser Vorgang wird so lange fortgesetzt , bis alle Knoten am oberen Ende des Stapels keine unbesuchten Nachbarn mehr hat; dann geht der Algorithmus zurück, indem er den Knoten vom Stapel nimmt und zum vorherigen Knoten zurückkehrt. Dieser Vorgang wird so lange fortgesetzt, bis alle Knoten, die vom Startknoten aus erreichbar sind, besucht worden sind



Durchlaufen Binären Bäume (Traversierung)

* Pre-Order (Wurzel-L-R): Bei dieser Traversierung, auch Tiefensuche gennant, wird zuerst den Wurzel betrachtet. Anschließend wird der linke und dann der rechte Teilbaum durchlaufen
* In-Order (L-Wurzel-R): Bei dieser Traversierung wird der linke Teilbaum durchlaufen. Danach folgt der Wurzel und am Ende der rechte Teilbaum
* Post-Order (L-R-Wurzel): Bei dieser Traversierung wird als erstes der linke und dann der rechte Telbaul durchlaufen. Zum Schluss wird die Wurzel betrachtet

Vorlesung 5

Ein gerichteter Graph, der eine Wurzel besitzt, ist immer schwach zusammenhängend, aber nicht jeder schwach zusammenhängende Graph besitzt eine Wurzel. Ein gerichteter Graph ist stark zusammenhängend genau dann, wenn jeder Knoten ein Wurzel ist.

Ein gerichteter Graph ist ein Baum genau dann, wenn er zusammenhängend und kreisfrei ist

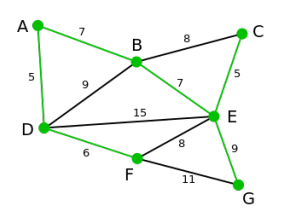
Gewichtete Graphen und minimale aufspannende Bäume

* Ein ungerichteter oder gerichteter heißt *gewichtet,* wenn jeder Kante ein Gewicht zugeordnet ist.
* Ein Teilgraph von einen gerichteter oder ungerichteter Graph heißt minimaler Spannbaum (minimum spanning tree) von G, wenn sein Gewicht minimal unter allen Spannbäumen von G ist.
* Unter den minimalen Spannbaum eines Graphen versteht man den Teilgraph, der alle Knoten am kostengünstigen miteinander verbindet.

Algorithmus von Prim

* Gehört zur Gruppe der Greedy Algorithmen. Er kann bei zusammenhängenden, gewichteten Graphen angewendet werden, um den minimalen Spannbaum zu ermitteln.

Der Prim Algorithmus beginnt bei einem Knoten (beliebig ausgewählt) und fügt die jeweils günstige Kante an den aktuellen Knoten ein. Nun hat man einen Teilgraphen mit zwei Knoten und einer Kante. Von diesem Teilgraphen muss wiederum eine Kante ausgehen zu einem noch nicht enthalteten Knoten. Deshalb wählt man hierfür wieder die Kante mit kleinstem Gewicht und setze das Verfahren fort, bis alle Knoten verbunden sind

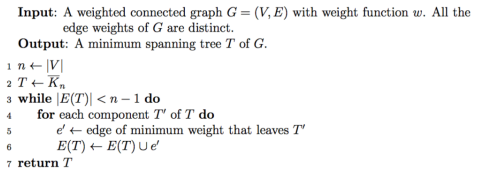


Algorithmus von Boruvka

-Gehört zur Gruppe der Greedy Algorithmen. Er kann bei zusammenhängendenn, gewichteten Graphen angewendet werden, deren Kantengeweichte paarweise verschieden sind, um den minimaler Spannbaum zu ermitteln

Der Algorithmus beginnt mit einem leeren Graphen MST, der den minimalen Spannbaum darstellen wird. Für jeden Teilgraphen G in der MST finde die Kante mit dem kleinsten Gewicht, die von G zu einem anderen Teilgraphen führt. Diese Kanten wird als “leichteste Kante” bezeichnet. Dann füge die leichtesten Kanten aus dem Teilgraphen G zu MST ein. Danach verbinde alle Teilgraphen, die durch die hinzufügten Kanten verbunden sind.

Dieser Vorgang wird so lange wiederholt, bis alle Knoten in der MST enthält werden.



Algorithmus von Dijkstra

Dijkstra Algorithmus ist eine Methode zur Lösung von Optimierungsproblemen, bei denen nach dem kostengünstigsten oder kürzesten Pfad in einem gerichteten Graphen mit nichtnegativen Kantengewichten gesucht wird. Der Dijkstra Algorithmus ist Greedy und sucht nach dem kürzesten Weg.

Der Algorithmus beginnt mit der Intialisierung den Startknoten mit einem Abstand von 0 und allen anderen Knoten mit einem Abstand von unendlich. Setze den Startknoten als den aktuellen Knoten.

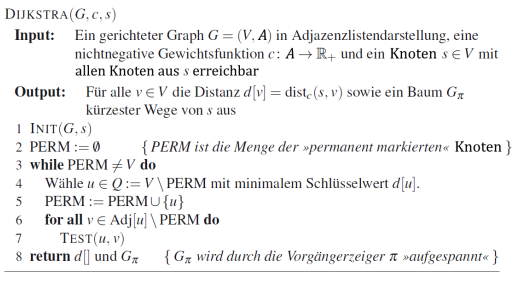
Im Verfahren halten wir eine Menge PERM von permanent markierten Knoten, d.h Knoten für die bereits die Distanz gilt, wobei Distanz der kürzeste Abstand zwischen den Startknoten und den jeweiligen Knoten ist.

Während des Suchprozesses wird eine Warteschlangen - Datenstruktur (queue) verwendet, um die Knoten nach ihrem Abstand zu verwalten. Wird. der Startknoten mit seinem Abstand zur Warteschlange hinzugefugt. Solange die Warteschlange nicht leer ist:

* Entfernen den Knoten mit dem kleinsten Abstand aus der Warteschange
* Dieser Knoten wird in der Menge PERM eingefügt
* Für jeden benachbarten Knoten des aktuellen Knotens: Berechne den vorläufigen Abstand zum benachbarten Knoten, indem du den Abstand des aktuellen Knotens und das Gewicht der Kante zwischen den beiden Knoten addierst. Wenn der vorläufige Abstand kleiner als der bisherigen Abstand des benachbarten Knotens, aktualisiere den Abstand des benachbarten Knotens mit dem vorläufigen Abstand. Füge den benachbarten Knoten mit seinem aktualisierten Abstand zur Warteschlange hinzu.

Der Algorithmus endet, wenn alle Knoten besucht wurden oder der Zeilknoten erreicht.

Der Algorithmus von Dijkstra ist korrekt, weil er schrittweise den kürzesten Pfad zu jedem Knoten berechten, basierend auf den bereits berechneten kürzesten Pfaden zu anderen Knoten, und sicherstellt, dass die berechneten Astände tatsächlich die kürzesten Pfade sind.



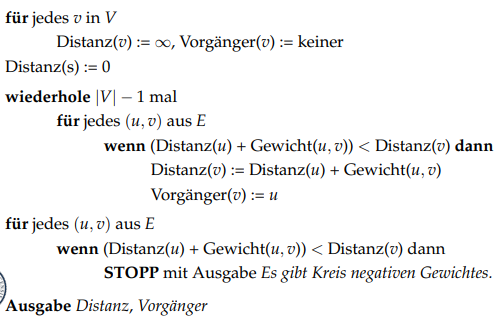
Vorlesung 6

Algorithmus von Bellman-Ford

Der Bellman-Ford Algorithmus dient zur Finden nach dem kürzesten Pfad von einem Startknoten aus zu allen anderen Knoten in einme gewichteten(auch negativen), gerichteren Graphen. Der Bellman-Ford Algorithmus durchsucht alle Kanten und merkt sich die leichteste.

Der Algorithmus beginnt mit der Intialisierung den Startknoten mit einem Abstand von 0 und allen anderen Knoten mit einem Abstand von unendlich. Setze den Startknoten als den aktuellen Knoten.

* Für jede Kante im Graphen, aktualisiere den Abstand des Zeilknotes, falls der Abstand des aktuellen Knotes plus das Gewicht der Kante kleiner ist als des bisherige Abstand des Zeilknotens. Dieser Vorgang wird für die Anzahl von Knoten minus 1 Mal fortgesetzt.
* Um herauszufinden, ob es einen negativen Zyklen gibt, führt man eine weiter Iteration über alle Kanten im Graphen durch und wenn der Abstand des Zielknotens aktualisiert wird, gibt es einen negativen Zyklus im Graphen. (um zu verhindern, dass der Algorithmus in einer Endlosschleife landet)



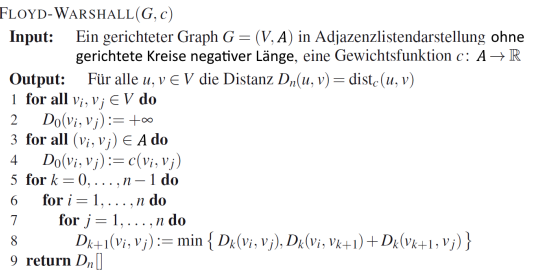
Algorithmus von Floyd-Warshall

Der Floyd-Warshall Algorithmus ist ein Algorithmus zum Find der kürzesten Pfade zwischen allen Paaren von Knoten in einem gerichteten Graphen. Der Algorithmus basiert auf der Idee der dynamischen Programmierung. Indem er schrittweise den Abstand zwischen Knotenpaaren aktualisiert, verwendet er bereits kürzesten Pfade, um neue kürzeste Pfade zu finden.

Zuerst im Initialisierenung erstellen wir eine Adjazenzmatrix, die die Kanten und ihre Gewichten im Graphen represäntiert. Setze alle Kanten für die es eine Verbindung gibt, auf ihre entsprechenden Gewichte, und alle anderen Kanten auf unendicht. Setze die Diagonalelemente der Matrix auf 0.

Für jedes Knotenpaar (u,v) überprüfe, ob der Pfad der Knoten u zum Knoten v durch den aktuelen Knoten führt und einen kürzeren Abstand bietet als der bischerige Pfad. Wenn ja, aktualisiere den Abstand.

Am Ende des Algorithmus enthält die Adjazenzmatrix die kürzesten Pfade zwischen allen Paaren von Knoten im Graphen.

^

Eulersche Kantenzüge

Ein Graph hat einen eulerschen Kantenzug gdw. G bis auf isolierte Knoten zusammenhängend ist und für die Zahl u der Knoten mit ungeradem Grad gilt u = 0 oder u = 2

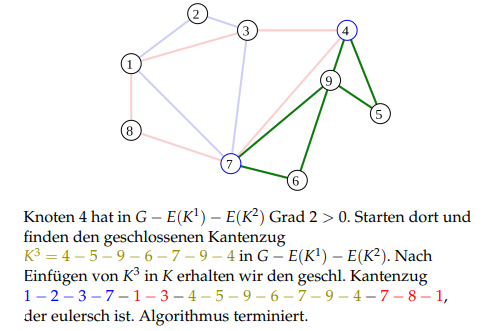
Algorithmus von Hierholzer

Algorithmus von Hierholzer ist ein Algorithmus zum Finden eines Eulerkreises in einem gerichteten oder ungerichtete Graphen, sofern dieser existiert. Ein Eulerkreis ist ein geschlossener Pfad der alle Kanten des Graphen genau einmal enthält.

* Man wählen einen beliebigen Anfangsknoten v0 und folge einem Kantenzug (ohne Kantenwiederholung) von v0 aus, bis man wieder bei derselben Knoten (v0) erreicht.
* Überprüfe, ob der Kantenzug K ein geschlossener eulerscher Kantenug ist, das heißt, ob alle Kanten in des Graphen G in K enthalten sind. Wenn dies der Fall ist, sind wir fertig und der Algorithmus bricht ab.

Ansonsten

* Wähle ein belibigen Knoten v1 in K, der noch unbesuchten Kanten in dem verbleibenden Teilgraphen aus G hat (Kanten, die in G existieren, aber nicht in K enthalten sind)
* Konstruiere einen neuen Kantenzug, der von v1 ausgeht und keinen Kanten in K doppelt durchläuft.
* Füge den neuen Kantenzug in K ein, indem man den Startpunkt v1 in K durchläuft und alle Knoten von den neuen Kantenzug in den richtigen Reihenfoge einfügt
* Nenne den aktualisierten Kantenzug K und kehre zu Schritt 2 zurück, um zu überprüfen , ob K nun ein geschlossener eulerscher Kantenzug ist

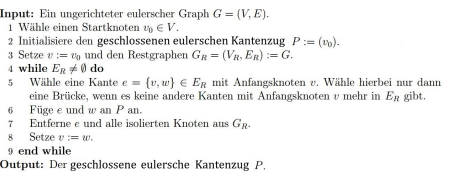


Algorithmus von Fluery

Algorithmus von Hierholzer ist ein Algorithmus zum Bestimmung eines Eulerkreises in einem gerichteten oder ungerichtete Graphen, sofern dieser existiert. Ein Eulerkreis ist ein geschlossener Pfad der alle Kanten des Graphen genau einmal enthält.

Der Fluery-Algorithmus funktioniert nicht bei Graphen mit Brücken oder Graphen mit Knoten von Grad 0 oder 2.

* Man wählen einen beliebigen Anfangsknoten als aktuellen Knoten
* Wähle unter den unmarkierten, mit dem aktuellen Knoten inzidenten Kanten eine beliebige Kante aus. Dabei sind zuerst Kanten zu wählen, die im unmarkierten Graphen keine Brückenkanten sind
* Markiere die gewählte Kante und füge sie der Kantenfolge hinzu
* Wähle den anderen Knoten der gewählen Kante als neuen aktuellen Knoten
* Wenn noch unmakierte Kanten existieren setze aus den Schritt 2 fort



Eulerisierung

Das Einfügen einer neuen Kanten in einem Graphen ist *legal,* falls die verbundene Knoten bereits durch eine alte Kante, also eine Kante, die im ursprünglichen Graphen bereits vorhanden war ß verbunden sind

Die Eulerisierung des Graphen G ist den Prozess des Einfügens von legalen Kanten in einem Graphen G, bis ein Graph entsteht, der einem geschlossenen eulerschen Kantenzug enthält.

Man kann jeden zusammenhängenden Graphen eulerisieren: Verdoppele alle Kanten und wähle dann Wege, so dass diese Kanten nicht mehrfach gehen

Wenn G‘ enthält keine Knoten ungeraden Grades und ist daher eulersch

Vorlesung 7

Graphendarstellung in der Ebene

Ein Polygonzug ist eine zum abgeschlossenen Einheitsintervall [0, 1] homöomorphe Teilmenge des R^2 die die Vereiningung endlich vieler Strecken ist.

F homöomorph (f bijektiv, f und f^-1 stetig)

Ein Paar (v, e) mit Knoten und Kanten endlicher Mengen heißt ebener Graph (plane graph), wenn gelten

* V ist eine Menge von Punkten im R^2
* Jede Kante e ist ein Polygonzug und die beiden Endpunkte von e liegen in V
* Die Endpunkte einer Kante verschieden sind und es zwischen je zwei Knoten höchstens eine Kante gibt

Planarität

Ein gerichteter oder ungerichteter Graph G heißt *planar*, wenn G isomorph zu einem ebenen Graph ist. Also G heißt *planar*, wenn man G so in der Ebene zeichnen kann, dass sich seine Kanten nicht kreuzen

Flächen

Für einen ebenen Graphen G ist die Menge R^2 \ G offen; ihre Gebiete sind die Flächen von G

Da G beschränkt ist, hat genau eine seiner Flächen einen unendlichen Flächeninhalt. Diese Fläche wird unbeschränkt (unbounded) oder unendlich (infinite) gennant.

* Die Rand (marginea) einer Fläche ist immer ein Untergraph von G
* Eine Kante, die zu einem Kreis C in G gehört, liegt auf dem Rand von genau zwei Flächen von G, und diese sind in verschiedenen Flächen von C enthalten

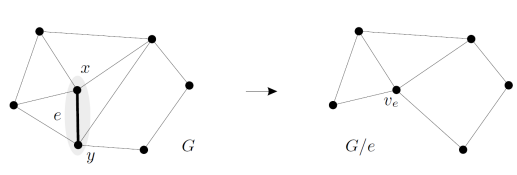
Ein eingebetteter Graph ist ein zusammenhängender Graph G mit einer zyklischen Ordnung der orientierten Kanten, die am gleichen Knoten beginnt, die wir als im Uhrzeigersinn interpretieren werden.

Wenn man einen eingebetteten Graphen zeichnet und eine Flache als die von Kanten begrenzte Region betrachtet, ist aufgrund der rechtsdrehenden Interpretation der Ordnung um die Knoten eine Flache die Folge von orientierten Kanten, wobei man die links liegende Flache durch eine Traversierung der orientierten Randkanten gegen den Uhrzeigersinn erhalt.

Ene Fläche in einem eingebetteten Graph ist eine zyklische Folge von orientierten Kanten.

Ein Graph heißt eben, falls |V(G)|-|E(G)|+F(G)=2

*Kontrakion* der Kante *e* ergibt den Graphen G/e (der Graph G außer e), indem in G die Kante *e* entfernt wird und die Knoten x, y zu einem neuen Knoten ve verschmolzen werden, der zu allen Kanten indizent ist, die in G zu x oder y indizent waren



Ein Minor in ein Graph, der aus einem anderen Graphen durch wiederholtes Löschen von Knoten und Kanten und Kontrahieren von Kanten entsteht

Ein ungerichteter, einfacher Graph G heißt planar, wenn alle Minoren von G planar sind.

Eulersche Formel

Es sei ein ebener Graph mit n Knoten, k Kanten, f Flächen (inklusive der unbeschränkte) und z Zussamenhangskomponenten. Dann gilt: n–k+f=z+1

Beweis:

* Zuerst gilt k = 0, so besteht es(Γ-Gamma) nur aus isolierten Knoten, d.h die Anzahl der Zusammenhangskomponenten ist z = n, es gibt nur eine Fläche, nähmlich die unbeschrämkte
* Es sei nun für ein k großer als 1 die Formel richtig für alle Graphen mit weniger als k Kanten. Hier gibt es zwei Fälle:
  + Sind alle z Zusammenhangskomponenten von Gamma Bäume (Graphen die keinen Kreisen enthalten und zusammenhängend sind), so gilt k=n-z
  + Da Bäume keine Kreise enthalten, gibt es auch hier wieder nur die unbeschränkte Fläche, d.h es gilt f=1. Insgesamt folgt dann n-k+f=n-(n-z)+1=z+1
* Es sein eine Kante auf diesem Kreis. Dann liegt e auf dem Rand von zwei verschiedenen Flächen F1 und F2

Folien 29-39

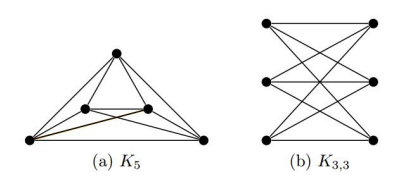
Eulersche Polyederformel. Es sei ein ebener, eiinfacher, zusammenhängender Graph. Gamma hat Ordnung n, Größe m und f Flächen. Dann gilt n-m+f=2.

Bipartite Graphen

-ein Graph heißt bipartit, falls es diskunkte Mengen A ⊂ V(gehört zu) und B ⊂ V gibt mit die die Vereinigung von A und B gleich mit V (A und B bilden also eine Partition von V), sodass keine Kanten von G zwischen zwei Knoten von A oder Knoten von B verläuft.

A und B heißen Partitionsklassen. Sind alle mögliche Kanten vorhanden, so nennen wird den Graphen *vollständig bipartit* und schreiben wir K|A|,|B| .

Der vollständige Graph K5 und der vollständige bipartite Graph K33 sind nicht planar.



Beweis

* In K5 gilt (Ordnung) n=5 und k = 10. Folglich ist die Bedingund 10 =k <=3n-6 = 9
* In K33 gilt gilt (Ordnung) n=6 und k = 9. Da K33 zusammenhängend ist, folgt durch die eulersche Polyederformel f=2-n+k=5. Es gibt also 4f<=2k=18 beziehungsweise f<=4, ein Widerspruch

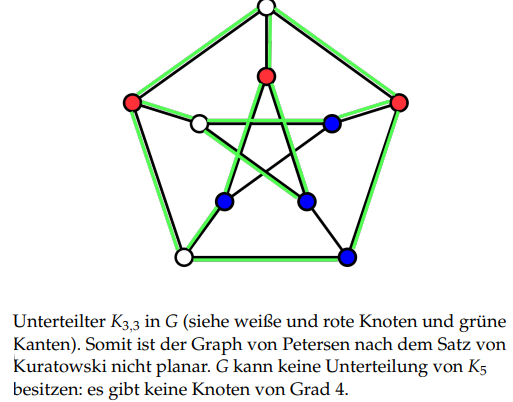
Satz von Kuratowski und Wagner

Ein Unterteilungsgraph ist ein Graph, der durch Kantenunterteilungen(subdividing edges) aus einem anderen Graphen entsteht. Zwei Graphen heißen homöomorph, falls sie isomorphe Unterteilungsgraphen besitzen.(homeomorphin-by removing or adding vertices of degree 2, you be able look like K5 or K33)

Ein ungerichteter, einfacher Graph ist genau dann planar, wenn er keinen Teilgraphen besitzen, der ein Unterteilungsgraph des K5 oder K33 ist.(Kuratowski). Unterteilungen von K5 und K33 nennt man auch Kuratowski Teilgraphen.

G ist planar gdw. G keinen der beiden Kuratowski-Graphen als Minor enthält. (Wagner)

Ein Petersen Graph ist ein 3-regulärer Graph mit 10 Knoten. Das bedeutet, dass jeder Knoten drei Nachbarn hat.



Planaritätsalgorithmus nach Demoucron, Malgrange und Pertuiset

Es sie G=(V,E) ein 2-zusammenhängender Graph (eingebettet is einem größeren Graphen) und P ein Weg, sodass nur seine Endknoten in V liegen. Dann ist G vereinigt mit P auch 2-zusammenhängend.

Definieren Sie die Fragment von dem Teilgraph in den größten Graph. Dann haben wir 2 Arte von Fragmente: trivial (Kante die in E ist aber nicht in E‘ aber den Knoten in den Teilgraph sind) und nicht trivial (Zusammenhangskomponente, was ist in E und nicht in E‘).

32:50

Der Algoritmus nach D, M und P berechnet eine planare Einbettung für einen planaren Graphen G=(V,E). (computes a planar embedding for a planar graph). Ist ein enkrementeller Algorithmus, da die Einbettung Schritt für Schritt die Einbetung eines neuen Zyklus des Diagramms ist. Wir haben bereits eine Einbettung eines Teilgraphen G’ von G berechnet, müssen wir uns die sogennanten Fragmente ansehen.

Für einen Teilgraph G’=(V’,E’) von G=(V,E) definieren wir ein Fragment von G(in Bezug auf G’) als einen Teilgraph C=(Vc, Ec) von G mit entweder Ec = {u,v} mit {u,v} ∈ G/G‘ oder ein Zusammenhagskomponente (connected component) von G/G‘ mit alle Kanten in G zwischen C und G‘

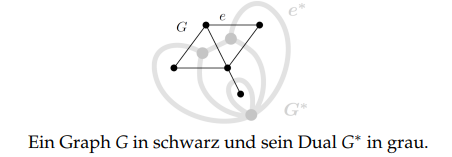
Kreisplanare Graphen

Es sei G=(V,E) ein ungerichteter, einfacher, planarer Graph

* G heißt kreisplanar(outerplanar), wenn G ein ebene Graphendarstellung besitzt, in der alle Knoten am Rand der (gleichen) unberschränkten Fläche liegen
* G ist genau dann kreisplanar, wenn er weder K4 noch K2,3 als Minor enthält

Dualer Graph

Es sei G=(V,E) die ebene Darstellung eines planaren Graphen. Dann heißt G\*=(V\*,E\*) der dualer Graph wenn gelten: In jeder Fläche von G liegt genau ein *v\*,* in jeder Flache von G\* liegt genau ein *v*, jede Kante *e* schneidet genau eine Kante *e\** und jede Kante *e\** schneidet eine Kante *e.*



Vorlesung 8

Es sei G ein Graph

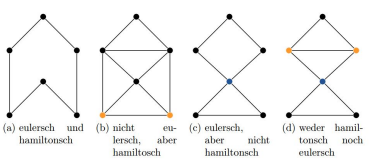
* Ein Weg heißt hamiltonsch (hemiltonian), wenn er jeden Knoten von G enthält
* G heißt hamiltonsch gdw. ein hamiltonschen Kreis (ein Pfad, der alle Knoten genau einmal enthält) enthält

Ziel ist, eine Reiseroute entlang der Kanten des Dodekaeders zu finde, die jede Stadt genau einmal besucht und dort aufhört, wo sie b eginnt.

Bedingungen:

* Es sei G ein Graph mit n>=3 Knoten und Minimalgrad >= n/2. Dann ist G hamiltonsch.(Dirac)
* Ein graph mit Zusammenhang 1 (deg(v) < 2) kann nich hamiltonsch sein

Im Gegensatz zur Bestimmung von (geschlossenen) eulerschen Kantenzügen ist die Bestimmung von Hamiltonkreisen rechnerisch ungleich schwerer (much more difficult) und es gibt keine schöne Charakterisierung von hamiltonschen Graphen.

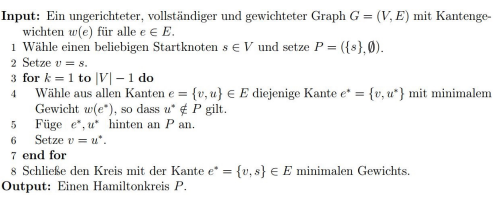


Nächster-Nachbar-Heuristik (nearest neighbor)

Gegeben sei ein ungerichteter, vollständiger und gewichteter Graph G=(V,E). Es wird ein Hamiltonkreis minimaler Länge gesucht. Der Algoritmus versucht eine gute Lösung zu finden, aber nicht garantieren kann, eine optimale Lösung zu finden

1. Zuert wähle einen beliebigen Startknoten von V (Knotenmenge von G)
2. Wähle den nächsten unbesuchten Knoten mit minimales Gewicht, der dem aktuellen Knoten am nächsten liegt
3. Füge diesen Knoten der Rundreise hinzu und markiere ihn als besucht
4. Wiederhole Schritt 3,bis alle Knoten besucht wurden und die Rundreise abgeschlossen ist
5. Verbinde den letzten Knoten der Rundreise mit dem Startpunkt, um den Kreis zu schließen

Die Nachbar-Heuristik wählt bei jedem Schritt den nächsten Punkt, der die geringste Entfernung zum aktuellen Punkt hat, und fügt ihn der Rundreise hinzu. Dies wird solange wiederholt, bis alle Punkte besucht wurden und die Rundreise abgeschlossen ist.



Korrektheit: Da G vollständig ist, gibt es in jeder Iteration mindestens eine Kante zu einem noch nicht besuchten Knoten und folglich auch (mindestens) eine mit minimaler Gewicht. Der Algorithmus findet zwar immer einen Hamlitonkreis, nicht aber immer den kürzesten.

Metrische Graphen

Es sei G=(V,E) ein ungerichteter, gewichteter Graph mit positiven Kantengewichten. Der Graph G heißt metrisch, wenn für alle Knoten u, v, z mit {u,v} {v,z} {z,u} die Dreiecksungleichung w({u,v}) <= w({v,z}) + w({z,u}) gilt. Beispiel: jeder ebener Graph mit Strecken als Kanten und dem euklidischen Abstand als Gewicht

Minimaler Spannbaum Heuristik

Input: ein ungerichteter, vollständiger und metrischer Graph min positiven Kantengewichten

1. Bestimme ein minimaler Spannbaum T\* von G (Algorithmus von Prim oder dem Algorithmus von Kruskal)
2. Vordeppele in T\* alle Kanten
3. Bestimme in dem entstandenen Graphen einen geschlossenen eulerschen Kantenzug (Algorithmus von Hierholzer)
4. Erzeuge aus dem geschlossenen eulerschen Kantenzug K einen Hamiltonkreis durch Überspringen bereits besuchter Knoten

Vom geschlossenen eulerschen Kantenzug zum Hamiltonkreis

Wähle einem beliebigen Startknoten auf dem geschlossenen eulerschen Kantenzug und folge diesem so lange, bis ein Knoten zum zwei Mal besucht würde. Da der Graph G vollständig ist, ist es möglich, stattdessen direkt zum nächsten unbesuchten Knoten auf dem geschlossenen eulerschen Kantenzug wiederzugehen

Vorlesung 9

Chromatische Zahl

Es sei G=(V,E) ein ungerichteter Graph.

Eine Abbildung c:V -> {1, ...,k} heißt Knotenfärbung mit k Farben, wenn für jeder Paar benachbarter Knoten unterschiedliche Farben haben.

Die minimum Anzahl den Farben in dem Knotenfärbung von G heißt der chromatiche Zahl (chromatic number) von G und ist als χ(G) (Chi) dargestellt.

Bsp: ein bipartiter Graph ist nichts anderes als ein Graph mit chromatischer Zahl 2.

Sequentieller Färbungsalgorithmus

Ein einfaches Verfahren, mit dem man eine Knotenfärbung bestimmen kann, ist das sequentielle Färbungsalgorithmus. Dieser verwendet aber nicht notwendigerweise die minimale Anzahl Farben.

Der Algorithmus erhaltet ein ungerichteter, einfacher Graph G=(V,E).

1. Lege eine beliebige Reihenfolge v1, v2 ... vn der Knoten fest
2. Weise dem ersten Knoten in dem Reihenfonge die Farbe 1 zu
3. Gehe die Knoten des Graphen in einer bestimmten Reigenfolge (außer der ersten Knoten) durch und für jeden Knoten betrachte alle bereits gefärbten Nachbarn. Weise dem Knoten die kleinste verfügbare Farbe zu, die von seinen Nachbarn noch nicht verwendet wurde

Output: Die Knotenfärbung c des Graphen G

(Satz) Der Algorithmus ist wohldefiniert und erzeugt eine Knotenfärbung mit höchstens maximal Grad eines Knotens von G + 1.

Wiederholung: eing Graph G ist regulär, falls alle Knoten in G denselben Grad haben.

Verbesserter Färbungsalgorithmus

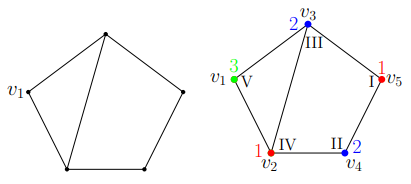
Im verbesserten Färnungsalgoritmus wird die Reihenfolge der Knoten nicht mehr beliebig festgelegt.

Der Algorithmus erhaltet ein ungerichteter, einfacher, zusammenhängender Graph G=(V,E). G hat mindestes einen Knoten s mit der Grad kleiner als die maximal Grad eines Knotens von G. (G sei also nicht rägular) ->(G ist regulär, falls alle Knoten in G denselben Grad haben)

1. Führe Breiten- oder Tiefensuche mit Startknoten s durch und nummeriere die Knoten v1, ... vn von G in der Reihenfolge ihres Auftretens
2. Weise dem letzten Knoten in dem bestimmten Reihenfolge die Farbe 1 zu.
3. Gehen die Knoten des Graphen in ungekehrter Reihenfolge außer der letzten Knoten (von hinten nach vorne) durch und betrachte alle bereits gefärbten Nachbarn. Weise dem Knoten die kleinste verfügbare Farbe zu, die von seinem Nachbarn noch nicht verwendet wurde.

Die Ausgabe ist die endgültige Knotenfärbung c des Graphen G

(Satz) Es G ein ungerichteter, einfacher, zusammenhängender Graph mit (mindestens) einem Knoten s ... Dann sist der verbesserte Färbungsalgorithmus wohldefiniert und erzeugt eine Knotenfärbung mit höchstens ∆(G) Farben.-> Da G zusammenhängend ist, sind Breitenß und Tiefensuche wohldefiniert und finden Wege zu allen Knoten von



Vierfarbensatz

Jeder ungerichtete, einfache, planare Graph erfüllt χ(G)<=4 : Satz(Appel und Haken 1976). Um diesen Satz in seiner dualen Fassung zu visualisieren: man kann die Länder und Meere einer Landkarte stets mit höchstens vier Farben färben derart, dass keine zwei benachbarten Flächen dieselbe Farbe erhalten

Museumwärterproblem

Vorlesung 10 : Matchings

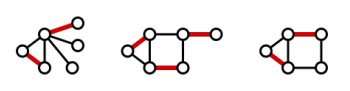
Kekule-Struktur

-erstellte die Struktur von Benzol als einen Ring aus sechs Kholenstoffatomen dar, wobei jede zweite Kante gedoppelt wird. Da jeder Knoten in einer Doppelkante liegt, hat jedes Kohlenstoffatom dann 4 Bindungen

-diese Menge von Doppelkanten ist das, was wir in de Graphenthorie ein perfektes Matching nennen.

Es sei G=(V,E) ein gerichteter oder ungerichteter Graph. Eine Menge M (gehört zu E) heißt *Matching*, wenn keine zwei Kanten adjazent sind.

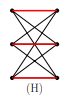
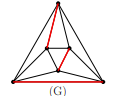
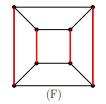
Ein Matching heißt maximal, wenn es nicht weiter erweitert werden kann, das heißt, es gibt keine weiteren Kanten im Graphen, die in dem Matching hinzugefügt werden können.



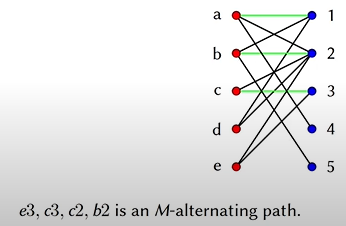
Ein Matching maximaler Kardinalität, auch bekannt als maximum Matching, ist ein Matching, bei dem die Anzahl der gepaarten Kanten so groß wie möglich ist.



Alle zu M indizenten Knoten heißen *M-überdeckt*, alle nicht zu M indizenten Knoten heißen *M-frei*. Ein Matching in G heißt *perfekt*, wenn es alle Knoten überdeckt. Ein Graph besitzt genau dann ein *perfektes Matching*, wenn *die Anzahl der Knoten im Graphen halb so groß ist wie die Anzahl der Kanten*. Eine bestimmte Bedingung, die erfüllt sein muss, damit ein perfektes Matching existiert, ist dass *die Anzahl der Knoten* im Graphen *gerade* sein muss.



Ein M-alternierender Weg ist ein einfacher Weg in einem Graphen, bei dem die Kanten abwehselnd in M und nicht in M liegen. Ein M-alternierender Weg in G, dessen beide Endknoten M-frei sind, heißt M-vergrößernder Weg (augmenting path).



Ein Graph heißt bipartiet, falls er zwei

Matchings maximaler Kardinalität in bipartiten Graphen

Der ungarischer Algorithmus

Der Algorithmus ist ein effizienter Algorithmus zum Finden eine Matching maximaler Kardinalitat in einem ungerichteten bipartiten Graphen mit zugehörigen Knotenpartition V1 und V2.

1. Wähle ein beliebiges Matching M, z.B ein leeren Matching oder der nur eine Kante enthält. Orientiere alle Kanten, so dass sie von V2 nach V1 zeigen und alle Kanten die E/M (die in der Kantenmenge existieren, aber nicht in der Matching), so dass sie von V1 zu V2 zeigen
2. Bestimme die Menge F1 aller M-freier Konten in V1 und die Menge F2 aller M-freien Knoten in V2

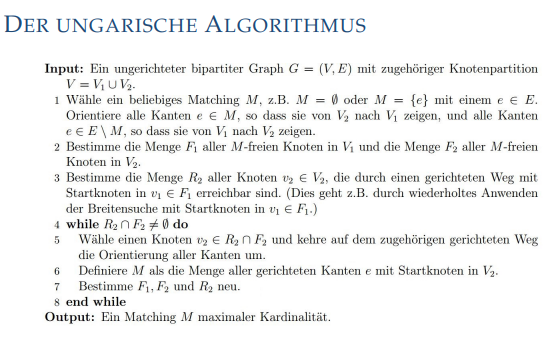
3.Bestimme die Menge R2 aller Knoten v2 ∈V2, die durch ein gerichteten Weg mit Startknoten in v1 ∈ F1 erreichbar sind

4.Wahrend die Durchsnitt zwischen R2 und F2 nicht leer ist

- wählen einen Knoten v2 und kehre auf den zugehörigen gerichteten Weg die Orientierung aller Knoten um

- definiere M als die Menge aller gerichteten Kanten e mit Startknoten n V2

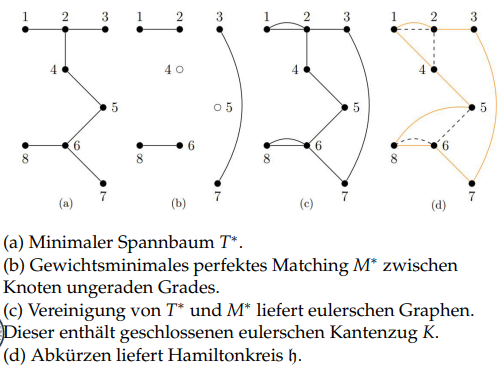
-bestimme F1,F2 und R2 neu



Starte ich irgendwie und dann wenn ich eine Kante wähle

Christofides-Heuristik  
Ein Algorithmus, de zur Lösung des Traveling Salesman Problemns verwendet wird. Das TSP ist ein bekanntes kombinatorisches Optimierungsproblem.Die Christofides-Heuristik basiert auf der Idee, das TSP in zwei Teilprobleme zu zerlegen: das Minimum Spanning Tree Problem und das Perfect Matching Problem. Der Algorithmus folg den folgenden Schritten

* Bestimme einen minimalen Spannbaum T\*
* Bestimme die Mengen in MST mit ungeradem Grad und erstelle ein Perfektes Matching mit min. Gewicht
* Verbinde des MST mit dem Perfektes Matching, um einen eulerschen Graphen zu erhalten. Danach bestimme einen geschlossenen eulerschen Kantenzug K (Algorithmus von Hierholzer). Ein eulerschen Graph ist ein Graph, der einen geschlossenen Rundweg enthält, der jeder Kante genau einmal durchläuft
* Erzeuge aus K einen Hamiltonkreis durch Überspringen bereits besuchter Knoten



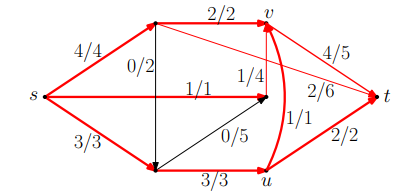
Vorlesung 11

Ein Flussnetzwerk besteht aus einem gerichteten Graphen, der aus Knoten und Kanten besteht. Jede Kante hat eine Kapazität, die angibt, wie viel Fluss sie maximal tragen kann. Die zwei ausgezeichneten Knoten sind die Quelle von dem Fluss ausgeht,und die Senke in dem der Fluss eintritt.

Es sei N=(G,c,s,t) ein Flussnetzwerk. Eine Abbildung heißt (s,t)-Fluss (flow) auf N, wenn die folgenden Bedingungen erfüllt sind:

* Kapazitätsbedingung (capacity constraints): die Kapazität wird für keine Kante überschrittem
* Flusserhaltungsbedingung (flow conservation constraints): Der Fluss, der in einem Knoten eintritt, muss gleich dem Fluss sein, der aus diesem Knoten austritt, außer in der Quelle und der Senke(1. Kirchhoffsche Gesetz)

Der Maximalfluss ist der größmögliche Fluss, der von der Quelle zur Senke im Netzwerk geleitet werden kann, unter Beachtung der Kapazitäten und der Erhaltungsbedingungen. Das Ziel besteht darin, den Flusswert zu maximieren, um die größmögliche Leistung im Netzwerk zu erreichen. (Wenn der Fluss maximal ist, sind alle Kanten, die von s ausgehen, voll ausgelastet, es gibt keine Möglichkeit, einen Fluss zu finden, der mehr transportiert).



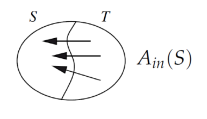
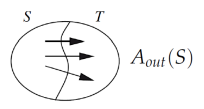
Ein Schnitt ist eine Aufteilung der Knoten eines Netzwerkes in zwei disjunkten Mengen, wobei die Quelle in einer Menge liegt und die Senke in der anderen Menge. Der Wert der Schnitts ist die Summe der Kapazitäten aller Schnittkanten (die Kanten, die den Schnitt überqueren).

Der Vorwärtsteil des Schnittes besteht aus den Knoten, die von der Senke aus erreichbar sind wenn man nur vorwärts entlang der Kanten im Schnittnetzwerk geht. (alle Kanten die aus der ersten Partition in der zweiter gehen)

Der Rückwärtsteil des Schnittes besteht aus den Knoten, die vor der Quelle aus erreichbar sind, wenn man nur rückwärts entlang der Kanten im Schnittnetzwerk geht. (naxh zurück fließt, aus der zweiter Menge in der ersten Menge)

Für jede Kante wird ein reelle Zahl geordnet

Definiren wir für solchen einen Schnitt der Überschuss, (addiere die Kante für die Vorwärtsteil, addiere die Kante für die Rückwertsteil subtrahiere den Wert de Vörwartsteil mit der Wert der Rückwertsteil)



Lemma: ist f ein (s,t) -Fluss und (S,T) ein (s,t)-Schnitt in einem Graphen, so gilt:

Der Kapazität gleich mit den Überschuss. Die Kapazität gibt den oberen Schranke für den Flusswert.

Der maximale Flusswert ist höchstens so groß wie die Kapazitat eines minimale

Ford-Fulkerson Algorithmus

* Wege finden die vergrößern sind und darauf basiert den Algorithmen. Erstens finden sie ein Maximalfluss finde, dass heißt, ich suche immer sogennante Fluss vergrößerte Wege bis ich keine mehr finde. Wenn ich diese gefunden habe, bin ich fertig
* Der Algorithmus basiert auf der Idee

Repeadedly finds vergrößerten Wege through the residual graph and augments the flow until no more augmenting paths can be find