Ungerichteter Graph: e = ps für s_t(e) = {q,s}
Gerichteter Graph (Digraph): e = qs für s(e) = p und t(e) = s
Multigraph (Ein Graph mit Mehrfachkanten)

- Eine Menge Kanten in einem Graph , deren Anfangs- und Endknoten übereinstimmen, werden als Mehrfachkante bezeichnet.
- Eine Menge von Kanten, deren Knoten übereinstimmen, bzw. im Falle gerichteter Kanten $\forall e1 , e2 \in E$: $s(e1) = s(e2) \wedge t(e1) = t(e2)$ oder $s(e1) = t(e2) \wedge t(e1) = s(e2)$, wird als parallele Kante bezeichnet.

Basic Definitionen

- Ein Graph ohne Schlingen oder Mehrfachkanten heißt ein **schlichter Graph**.
- Ein schlichter Graph ohne parallele Kanten heißt ein **einfacher Graph**.
- Ein schlichter, ungerichteter Graph mit n Knoten heißt **vollständig**, wenn je zwei Knoten durch eine Kante verbunden sind. Bezeichnung **Kn**. |E| ist immer (n(n-1))/2
- Zwei Knoten, die durch eine Kante verbunden sind, heißen **adjazent** (benachbart)
- Wenn v (Anfangs- oder) Endknoten der Kante e ist, heißen v und e **inzident**.
- Sei G = (V , E) ein Graph. Jeder Graph H = (W , F) mit $W \subseteq V$ und $F \subseteq E$ heißt ein **Teilgraph** von G, geschrieben $H \subseteq G$.
- Ein Graph H = (W , F) mit $W \subseteq V$ heißt ein **Untergraph** von G = (V , E), wenn seine Kantenmenge F genau diejenigen Kanten aus E enthält, die zwei Knoten aus W verbinden. Das wird mit H v G oder durch H = G[W] notiert.

Wege und Kreise

- Folge in G = (V , E) , wo abwechselnd Knoten und Kanten sind, ist eine **Kantenfolge**
- Eine Kantenfolge heißt ein **Weg** von v 0 nach v k , wenn alle Knoten und Kanten jeweils voneinander verschieden sind.
- Geschlossene Kantenfolge heißt **Kreis**, wenn alle Knoten und Kanten verschieden
- Ein Knoten u heißt von einem Knoten v aus **erreichbar**, wenn entweder u = v ist oder es eine Kantenfolge gibt, in der v vor u auftritt.
- Wenn $\forall v \in V$: d out (v) > 0 oder $\forall v \in V$: d in (v) > 0, dann besitzt G einen Kreis.

Abbildungen:

- Injektiv: a: A→B Es werden niemals 2 unterschiedliche Elemente aus A auf ein und das selbe Element in B abgebildet
- Surjektiv: A→B $\forall y \in B \; \exists x \in A$: f(x)=y, also alle Elemente in B werden min. 1 mal getroffen
- Bijektiv: Wenn Abbildung injektiv und surjektiv ist

Isomorphie
Zwei Graphen G = (V , E) und H = (W , F) heißen isomorph (geschrieben: $G \cong H$),
wenn es zwei bijektive Abbildungen $\varphi : V \to W , \psi : E \to F$
gibt, so daß für alle u, v ∈ V und e ∈ E gilt: $e = uv \Leftrightarrow \psi(e) = \varphi(u)\varphi(v)$
Nachbarschaft
Sei $W \subseteq V$ eines Graphen G = (V , E).
Dann ist die Nachbarschaft von W : NG(W) = {v |w ∈ W und (w, v) ∈ E}
und die (abgeschlossene) Nachbarschaft von W :cNG(W) = NG(W) ∪ W
Transitive Hülle
Gegeben ein ungerichteter Graph G = (V , E). Transitive Hülle von G der Graph G+ = (V , E +), der genau dann eine Kante e ∈ E + mit s t(e) = {v i , v j } enthält, wenn es in G einen Weg (oder Kreis) von vi nach vj gibt.
Gegeben ein gerichteter Graph G = (V , E). Transitive Hülle von G der Graph G + = (V , E +), der genau dann eine Kante e ∈ E+ mit s(e) = vi und t(e) = vj enthält, wenn es in G einen Weg (oder Kreis) von v i nach v j gibt.

Bipartiter Graph
Ein ungerichteter Graph G = (V,E) heißt bipartit, wenn sich V in zwei disjunkte Teilmengen X , Y zerlegen lässt (V = X ∪ Y ; X ∩ Y = Ø), dass jede Kante e = xy zwischen x∈X und y∈Y ist.

Gewichtete Graphen
Ein schlichter Graph G = (V , E) heißt kantenbewertet, wenn es eine Funktion L : E → R gibt.
Also wenn jeder Kante eine Zahl zugeordnet ist. Bei einem gewichteten Graph, ist jeder Kante ein Wert zugeordnet.

Planare Graphen
Ein Graph G = (V , E) heißt planar, wenn er in der Ebene so gezeichnet werden kann, dass jeder Punkt, den zwei Kanten gemeinsam haben, ein Knoten ist, also wenn er kreuzungsfrei ist.
Kleinste nicht planare Graphen: K3,3 und K5
Es hat keinen Einfluss auf die Planarität, wenn Kante durch Einfügen eines Knoten mit Grad 2 in zwei Kanten zerlegt wird
Ein Graph G ist genau dann planar, wenn weder K 5 noch K 3,3 ein Minor von G ist

Eulersche Polyederformel für die Ebene
Ist G = (V , E) ein planarer Graph,
1. der zusammenhängt, dann gilt: |V | − |E| + |F | = 2
2. mit K Komponenten, dann gilt: |V | − |E| + |F | = 1 + K

$$\chi(C_n) = \begin{cases} 2 & ; \text{ falls } n \text{ gerade} \\ 3 & ; \text{ sonst} \end{cases}$$

Knotengrad
Sei v ∈ V ein Knoten eines Graphen G = (V , E).
Wenn G ungerichtet, dann d(v) = |{e ∈ E|v ∈ s_t(e)}| + |{e ∈ E|v ∈ s_t(e) ∧ |s_t(e)| = 1}|
Falls G gerichtet, dann **Ausgangsgrad** d_out(v) = |{e ∈ E|s(e) = v }|
und **Eingangsgrad** d_in(v) = |{e ∈ E|t(e) = v }|
Maximalgrad Δ(G) = max(v∈V d(v))
Minimalgrad δ(G) = min v ∈V d(v).
Der Graph G heißt **k-regulär**, wenn d(v) = k für alle v ∈ V.
Im ungerichteten Graph ist die Summe aller Knotengrade gleich mit 2 · |E|

Knotenfärbung
Ist G = (V , E) ein ungerichteter Graph ohne Mehrfachkanten und
c : V → S eine Abbildung von V in die Menge S mit c(v) != c(w) für zwei benachbarte Knoten v und w, so nennt man c eine Knotenfärbung von G.
G ist k-färbbar, falls es eine Knotenfärbung c : V → {1, ..., k} von G gibt.
Für jeden Graphen gibt es eine kleinste Zahl k , sodass der Graph k -färbbar ist. Diese Zahl wird die **chromatische Zahl** des Graphen genannt und meist mit **χ(G)** bezeichnet. Der Graph heisst dann k -chromatisch.
G ist 2-färbbar gdw. G ist bipartit gdw. G hat keine Kreise ungerader Länge.
χ(Kn) = n
Ist H ein Untergraph von G, dann gilt: χ(H) ≤ χ(G).
Für leere Graphen G Ø gilt χ(G Ø) = 1.
Ist Graph G = (V , E) planar, dann ist χ(G) ≤ 4.
Es gilt χ(G) ≤ Δ(G) + 1, Bedeutung: Δ(G) = Maximalgrad

Greedy-Färbung
Sei v1 , ..., vn eine Ordnung der Knoten von G.
Definiere c(v) von links nach rechts mit
c(vj) = min(N+ \ { Farben, die schon bei Nachbarn von v i verwendet wurden })
Die Greedy-Färbung benutzt für jeden Knoten die kleinste Farbe, die nicht schon ein Nachbar hat. Jeder hat aber höchstens Δ(G) Nachbarn. → **χ(G) ≤ Δ(G) + 1**
Verbesserter Greedy definiere Ordnung so, dass gilt i ≤ j ⇒ d(vi) ≥ d(vj).

ColorFirst
Wiederhole, bis alle Knoten gefärbt sind:
Wähle eine bisher nicht verwendete Farbe, und färbe damit jeden noch ungefärbten Knoten, falls er nicht mit einem Knoten dieser Farbe verbunden ist.

Färbungsalgorithmus BFS
1. Färbe ersten Knoten mit ”1”.
2. Wiederhole, bis alle Knoten gefärbt sind:
Markiere aktuellen Knoten. Färbe alle noch ungefärbten Nachbarn mit kleinster Farbe, die deren Nachbarn nicht haben. Wähle einen der unmarkierten Nachbarknoten.

Adjazenzmatrix
Adjazenzmatrix A(G) := (aij) des ungerichteten Graphen G(V , E)
ist eine symmetrische |V|×|V|-Matrix mit: aij := Anzahl der Kanten mit den Endknoten vi und vj
Bei gerichtetem Graph wird immer dort +1 gesetzt, wo eine Kante rein kommt

Inzidenzmatrix
Die Inzidenzmatrix M(G) := (m ij) des ungerichteten Graphen G(V , E) ist eine |V| × |E|-Matrix
Also |V| Zeilen und |E| Spalten
Ungerichteter Fall:
- 0, falls vi nicht inzident ist mit ej
- 1, falls vi einer der Endknoten von ej ist
- 2, falls vi der Endknoten der Schlinge ej ist

Gerichteter Fall:
- falls v i nicht inzident ist mit e j
- −1, falls v i die Anfangsknoten von e j ist
- +1, falls v die Endknoten von e ist

Zusammenhang von Knoten
In ungerichteten G sind Knoten u und v **zusammenhängend**, wenn u=v ist oder Weg von u nach v vorhanden. In einem gerichteten Graphen sind Knoten u und v **stark zusammenhängend**, wenn u = v ist oder es einen Weg von u nach v gibt und es einen Weg von v nach u gibt. In einem gerichteten Graphen heißen zwei Knoten u und v **schwach zusammenhängend**, wenn sie in dem zugrundeliegenden ungerichteten Graphen zusammenhängend sind. In einem ungerichteten Graphen G = (V , E) heißen die größten Untergraphen von G, die nur zusammenhängende Knoten enthalten **(Zusammenhangs-)Komponenten**. Der Graph G heißt **zusammenhängend**, falls G genau eine Komponente besitzt. In einem gerichteten Graphen G = (V , E) heißen die größten Untergraphen von G, die nur stark zusammenhängende Knoten enthalten **starke (Zusammenhangs-)Komponenten**. Der Graph G heißt **stark zusammenhängend**, falls G genau eine starke Komponente besitzt. In einem gerichteten Graphen G = (V , E) heißen die Zusammenhangskomponenten auf dem zugrundeliegenden ungerichteten Graph **schwache (Zusammenhangs-)Komponenten**. Der Graph G ist **schwach zusammenhängend**, falls G genau eine schwache Komponente besitzt.

Schnitte
In einem zusammenhängenden Graphen G = (V , E) heißt ein Knoten v ein **Schnittknoten**, falls der Untergraph H von G mit Knotenmenge V \ {v} nicht zusammenhängend ist. Entsprechend heißt eine Kante e eine **Schnittkante**, falls der Teilgraph F (V , E \ {e}) von G nicht zusammenhängend ist.

Flüsse in Netzwerken
Modellierung mit schwach zusammenhängenden, schlichten gerichteten Graphen G(V , E) mit |V | = n Knoten
c(e) Kapazität von Kante,
O(v) = {e ∈ E|s(e) = v } **output** von Knoten, **I(v)** = {e ∈ E|t(e) = v } **input** von Knoten
Es gibt **Quelle** q mit d_out (q) = 0 und **Senke** s mit d_in(s) = 0
für alle v ∈ V \ {q, s} gilt d_in (v) > 0 und d_out (v) > 0.
f(e) ist der **Fluss.** Für e ∈ E gilt f(e) ≤ c(e) (**Kapazitätsbeschränkung**),
Innere Knoten, leiten Mengen des Gutes verlustlos weiter, d.h. es gilt die **Flusserhaltung**
Der **Wert d** des Flusses ist die Summe der outputs von q bzw. die Summe der inputs bei s
Schnitt
Wenn X und Y beliebige Untermengen von Knoten eines Graphen G sind, bezeichnet
A(X , Y) die Menge der Kanten, die Knoten aus X mit Knoten aus Y verbinden.
A + (X , Y) ist die Menge der Kanten, ausgehend von Knoten aus X , diese mit Knoten aus Y verbinden.
A − (X , Y) ist die Menge der Kanten, ausgehend von Knoten aus Y , diese mit Knoten aus X verbinden.
Sei g eine beliebige Funktion, die den Kanten eines Graphen G nichtnegative rationale Zahlen zuordnet, dann ist für zwei beliebige Knotenmengen X , Y von G: g(X , Y) = Summe g(e) von allen e∈A+(X ,Y) .
!X ist das Komplement von X in V , d.h. !X = V \ X .
Ein **Schnitt** ist eine Menge von Kanten A(X , !X), wobei q ∈ X und s ∈ !X.
Es sei f ein Fluss in einem Netzwerk G = (V , E), und es sei d der Wert des Flusses.
Wenn A(X , !X) ein Schnitt in G ist, dann gilt d = f (X , !X) − f (!X , X) und d ≤ c(X , !X).
Ein Graph G hat 2 hoch n mögliche Schnitte, wenn es n innere Knoten in G gibt.
Ein Fluss, dessen Wert min{c(X, !X)|A(X, !X)ist ein beliebiger Schnitt} entspricht, ist ein **maximaler Fluss**.
Ein ungerichteter Weg von q zur s ist **vergrößernder Weg**, wenn für jede Vorwärtskante gilt f (e) < c(e) und für jede Rückwärtskante gilt f (e) > 0
Inkrement von Flüssen
Wenn f ein Fluss in G ist, wird einer Kantenfolge W eine nichtnegative Zahl i(W), das
Inkrement von W , zugeordnet, wobei i(W) = min{i(e)|eist eine Kante der KantenfolgeW }
i(e) = c(e) − f (e) falls e eine Vorwärtskante von W
i(e) = f (e) falls e eine Rückwärtskante von W
Wenn in einem Graphen G ein Fluss der Stärke d von der Quelle q zur Senke s fließt,
gilt genau eine der beiden Aussagen:
1. Es gibt einen vergrößernden Weg.
2. Es gibt einen Schnitt A(X , X) mit c(X , X) = d.

BFS:
Gegeben sei ein Graph G mit zwei ausgezeichneten Knoten s und t.

1. Man kennzeichne den Knoten s mit 0 und setze i = 0.
2. Man ermittle alle nichtgekennzeichneten Knoten in G, die mit den mit i gekennzeichneten Knoten benachbart sind.

Falls es derartige Knoten nicht gibt, ist t nicht mit s über einen Weg verbunden.
Falls es derartige Knoten gibt, sind sie mit i + 1 zu kennzeichnen.

3. Wenn t gekennzeichnet, folgt 4., wenn nicht, erhöhe man i+=1 und gehe zu 2.
4. Die Länge des kürzesten Weges von s nach t ist i + 1. Ende.

BFS Rückverfolgung: (Die Kennzeichnung des Knoten a wird dabei mit λ(a) bezeichnet)
Algorithmus erzeugt einen Weg v0, v1 , … , vλ(t) , so dass v0 = s; v λ(t) = t ist.

1. Man setze i = λ(t) und ordne vi = t zu.
2. Man ermittle einen Knoten u, der zu vi benachbart ist und mit λ(u) = i − 1 gekennzeichnet ist. Man ordne v i−1 = u zu.
3. Wenn i = 1 ist, ist der Algorithmus beendet. Wenn nicht, i-=1 und gehe zu Schritt 2.

Dijkstra
lij ist Länge der Kante vi->vj . Falls keine Kante lij := ininity
Für jeden Knoten vi ∈ V des zu untersuchenden Graphen werden drei Variable angelegt:
1. Entfi kürzeste Entfernung von v1 nach vi. Der Startwert 0 für i = 1 sonst infinity
2. Vorgi gibt den Vorgänger von vi an. Startwert ist v1 für i = 1 und undefiniert sonst.
3. OKi zeigt, ob kürzeste Entfernung von v1 nach vi bekannt. Startwert für alle false.
Algorithmus:

- Suche Knoten vh wo OK = false und Enf am kleinsten
- Setzte OK = true für vh
- Nachbarn vj, wo OK = false und Enfj > Enfh + länge(h->j)
- Für alle vj Enfj = Enfh + länge(h->j) und Vorgj = vh
- Wiederhole so lang es Knoten mit OK = false gibt

i	1	2	3	4	5	6
Entf	0	∞	∞	∞	∞	∞
Vorg	1					
OK	*f*	*f*	*f*	*f*	*f*	*f*

A* Algorithmus

Knoten $V = \{v_1, \ldots, v_n\}$
offenen Liste: OL nur der Startknoten v_1 mit $f_1 = h_1$, $g_1 = 0$ und $p_1 = v_1$
geschlossene Liste: CL leer
heuristische Knotenwerte : $h_i \in \mathbb{R}^+$
Kantenlängen zwischen v_i und v_j: $l_{ij} \in \mathbb{R}^+$
Vorgänger: $p_i \in V$
aktueller Schätzwert $f_i = \infty$ für $i > 1$
zurückgelegter Weg $g_i = \infty$ für $i > 1$

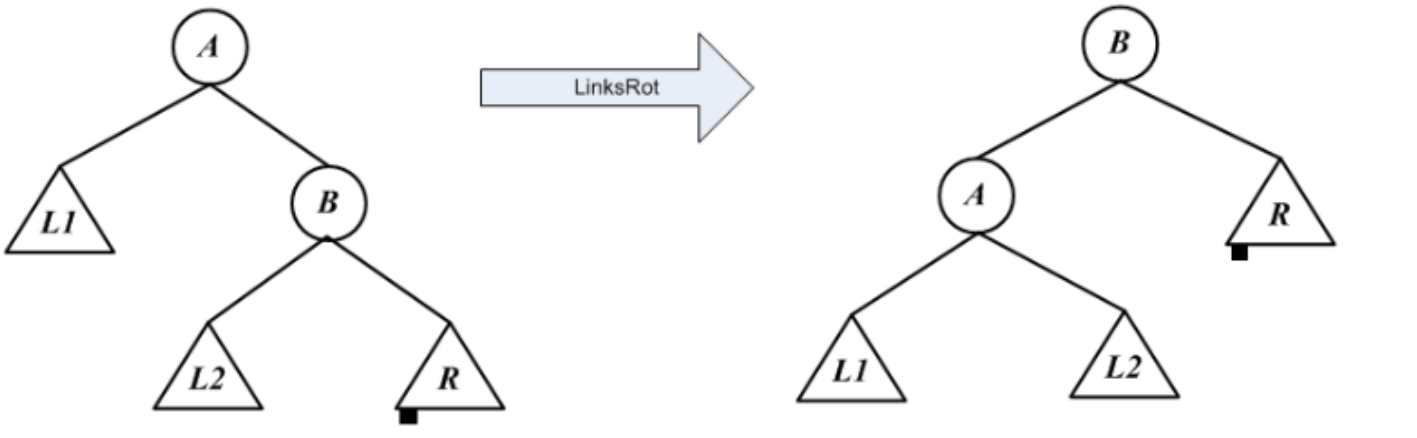
	v_1	v_2	v_3	v_4
Vorgänger (p)	v_1			
heuristik(h)	h1			
Zurückgel. Weg (g)	0	∞	∞	∞
Schätzwert (f)	h1	∞	∞	∞
Closed List	*t*			

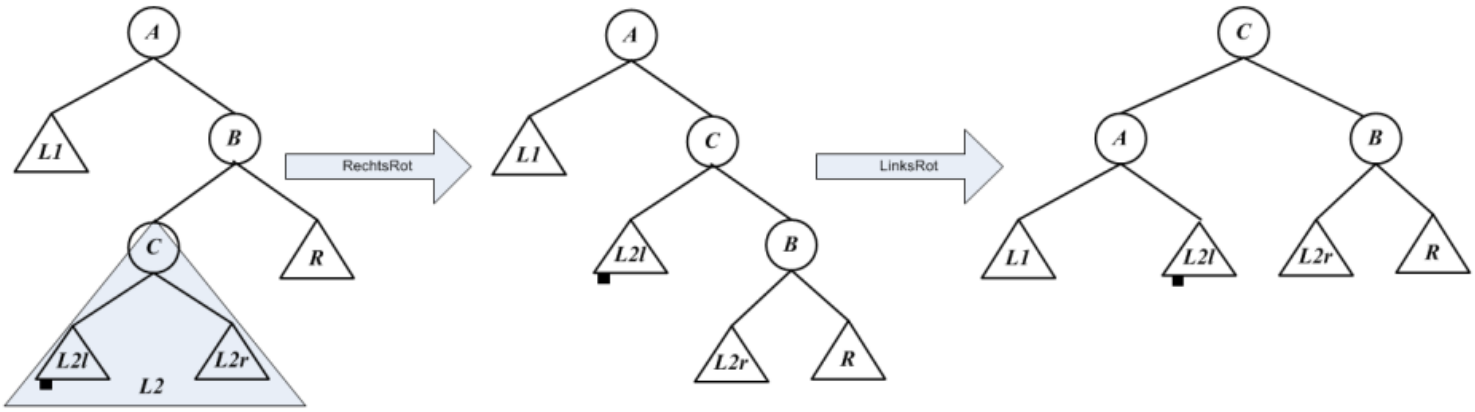
Algorithmus

1. Knoten v_k mit niedrigsten f_k in OL suchen
2. Knoten v_k in die CL schieben.
3. adjazente Knoten v_j, die nicht in der CL sind, in die OL schreiben und prüfen, ob $g_j > g_k + l_{k,j}$. Falls
   JA, wird der aktuelle Knoten v_k Vorgängerknoten: $p_j := v_k$
   und neuer g- und der f-Wert: $g_j := g_k + l_{k,j}$ und $f_j := g_j + h_j$
4. Falls der Zielknoten in der geschlossenen Liste; gehe zu 5.
   Falls kein Zielknoten gefunden und offene Liste leer; gehe zu 6.
   Sonst; gehe zu 1.
5. Der Pfad läßt sich vom Zielknoten aus mittels der Vorgänger bis zum Startknoten zurück verfolgen.
6. Es gibt keinen Pfad.

Heuristik: Für jeden Knoten k und jeden Nachfolger j von k muss gelten h(k) <= lk,j + h(j), damit optimal. Nicht optimal, wenn Ziel unbekannt oder Heuristik sehr komplex

Bäume
Ein ungerichteter zusammenhängender kreisloser Graph heißt ein **Baum.**
Ein ungerichteter Graph, dessen Komponenten Bäume sind, heißt ein **Wald**.
Ein gerichteter kreisloser Graph heißt **azyklisch**.
Eigenschaften
Zwischen je zwei verschiedenen Knoten aus G gibt es genau einen Weg.
G ist zusammenhängend, und jede Kante aus E ist eine Schnittkante.
G ist zusammenhängend, und es ist |E| = |V | − 1.
G besitzt keinen Kreis, aber durch Hinzufügen einer beliebigen Kante zu E entsteht ein Graph mit genau einem Kreis.
Der **Abstand** a(x, y) von zweier Knoten x,y im Baum ist Anzahl der Kanten zwischen ihnen
Sei B ein Wurzelbaum. max(a(x, x0)), wobei x ∈ B und Wurzel = x0 heisst **Länge** L.
Die Anzahl der Knoten des längsten Weges zur Wurzel x 0 heißt Höhe H des Wurzelbaumes. Höhe= Länge+1
Suchbaum Eintragen Algorithmus
Trage Schlüssel s in den Baum ein:
Existiert Baum noch nicht, so erzeuge Wurzel und trage s als Wurzelschlüssel ein.
Sei s0 der Wurzelschlüssel.
Falls s < s 0 : Trage Schlüssel s im linken Teilbaumder Wurzel ein.
Falls s > s 0 : Trage Schlüssel s im rechten Teilbaum der Wurzel ein.
Suche im Suchbaum Algorithmus
Ist s gleich dem Schlüssel x, so liefere x zurück. Sonst:
Falls s < x: Suche s beginnend beim linken Sohn von x
Falls s > x: Suche s beginnend beim rechten Sohn von x
Man braucht maximal H − 1 Vergleiche, um alle 2**H−1 Knoten zu finden.
AVL Bäume(erst LinksRot, Dann Problem Links)





Gerüste
Ein Baum H = (W, F) heißt ein Gerüst von G, wenn H ein Teilgraph von G ist und alle Knoten von G enthält (wenn also gilt $F \subseteq E$ und $W = V$)
Ein Graph G besitzt ein Gerüst, gdw. er zusammenhängend ist.
τ(G) ist die Anzahl der verschiedenen Gerüste von G.
Der vollständige Graph Kn mit n Knoten hat $\tau(K_n) = n^{**}(n-2)$ verschiedene Gerüste.

Prüfer-Code → Graph
|V| = n, dann aus n−2-Tupel T wird Spannbaum S = {1, 2, …, n} konstruiert.
ki = min(Xi \ Ti)
ei = (ki, erstes Element aus Ti) → Edge
Xi+1 = Xi \ {ki}, Ti+i = Ti \ {erstes Element aus Ti}
Wiederhole bis |Xi| = 2. Dann verbinde die letzten beiden in Xi
Graph → Prüfer-Code
Suche Blatt x mit niedrigstem Wert.
Suche Knoten y, der über Kante mit y verbunden ist. → Schreibe y in Prüfer-Code.
Ignoriere die letzten beiden übrig gebliebenen Knoten.
Kruskal Algorithmus für Minimalgerüst
Numeriere die Kanten nach steigender Länge. Setze F := $\emptyset$.
Für i := 1, … , |E|:
Falls $F \cup \{ei\}$ nicht die Kantenmenge eines Kreises in G enthält, setze $F := F \cup \{ei\}$.

Euler Tour und Pfad
Eine geschlossene Kantenfolge, die jede Kante eines Graphen genau einmal enthält, heißt eine Eulertour.
Ein Graph, der eine Eulertour besitzt, heißt ein eulerscher Graph.
Eine Kantenfolge, die jede Kante eines Graphen genau einmal enthält und nicht geschlossen ist, heißt ein Eulerpfad.
Ein ungerichteter Graph besitzt genau dann eine Eulertour, wenn jeder Knoten einen geraden Grad besitzt.
Ein ungerichteter Graph besitzt genau dann einen Eulerpfad, wenn genau zwei Knoten einen ungeraden Grad besitzen. Diese beiden Knoten sind der erste und der letzte Knoten des Eulerpfads.
Kann G in kantendisjunkte Kreise zerlegt werden, dann gehen durch einen Knoten v genau i kantendisjunkte Kreise, so gilt $d(v) = 2i$

Fleury Algorithmus für Euler Kreis
Gegeben sei ein eulerscher Graph G = (V , E).
Ausgabe ist die Eulertour W

1. Man wähle einen beliebigen Knoten v0 in G und setze W0 = v0.
2. Wenn der Kantenzug W i = v0,e1,v1 . . . ei,vi gewählt worden ist (so daß alle Kanten unterschiedlich, wähle man eine unbenutzte Kante e i+1 , so daß
    1. e i+1 inzident mit vi ist und
    2. ausgenommen keine Alternative, e i+1 keine Schnittkante ist.
3. Beende, wenn Wi jede Kante von G beinhaltet. Andernfalls gehe zu Schritt 2.

Algorithmus von Hierholzer

1. Wähle einen beliebigen Knoten v0 des Graphen und konstruiere von v0 ausgehend einen Kreis a K in G. Vernachlässige nun alle Kanten dieses Kreises.
2. Am ersten Knoten des ersten Kreises, dessen Grad größer 0 ist, lässt man nun einen weiteren Kreis entstehen. Erstelle so viele Kreise, bis alle Kanten von einem Kreis durchlaufen wurden.
3. Nun erhält man den Eulerkreis, indem man mit dem ersten Kreis beginnt und bei jedem Schnittpunkt mit einem anderen Kreis, den letzteren einfügt, und danach den nächsthöheren Kreis wieder bis zu einem weiteren Schnittpunkt oder dem Endpunkt fortsetzt.

Hamiltonkreis

Ein Hamiltonscher Weg in einem Graphen G ist ein Weg, der jeden Knoten von G genau einmal enthält.
Ein Hamiltonscher Kreis (oder Hamiltonischer Zyklus) in einem Graphen G ist ein Kreis, der jeden Knoten von G enthält.
Ein Graph heißt hamiltonsch, wenn er einen hamiltonschen Kreis enthält.
Sei G ein schlichter Graph mit n Knoten für $3 \leq n \in N+$, und der Minimalgrad von G betrage $\delta(G) \geq n/2$, dann ist G hamiltonsch.
Ein schlichter Graph G ist dann und nur dann hamiltonsch, wenn seine Hülle c(G) hamiltonsch ist.
Also G ist hamiltonisch, wenn $\delta(c(G)) \geq n/2$

Metrisches TSP

liegt vor, wenn zusätzlich die Kantenlängen die Dreiecksungleichung erfüllen; also die direkte Verbindung von i nach j ist nie länger als der Weg von i nach j über einen dritten Knoten k : $c\ ij \leq c\ ik + c\ kj$

Minimum-Spanning-Tree-Heuristik

Berechnung eines minimalen Gerüstes
Konstruktion einer Tour, durch Verdopplung aller Baumkanten und dann Suche einer Eulertour.
Abkürzung durch direkte Kanten, falls Knoten doppelt besucht werden falls metrisches TSP ohne Kontrolle, sonst mit.
höchstens doppelt so lang ist wie eine kürzeste Tour.

Nearest Insertion für TSP

Es wird ein vollständiger Graph benötigt
Gegeben sei ein Graph Kn = (V , E) mit V = {v1 , v2 , … vn}. Der aktuelle gewählte Kreis Wi wird stets neu numeriert und ist gegeben durch Wi = u1, u2 . . . ui,u1 . Diese neue Numerierung verändert nicht die Reihenfolge der gewählten Knoten vi!

1. Man wähle eine beliebige Knoten u 1 = v j ∈ V als Startknoten und setze W 1 = u 1 .
2. Aus der Zahl der n − i Knoten, die bisher noch nicht gewählt worden sind, ermittle man einen Knoten u+1 = vk , die am dichtesten zu Wi liegt. Sei Wi = u1u2 . . . uiu1 . Man bestimme dann, welche der Kantenfolgen (Kreise) die kürzeste ist. Es sei W i+1 die kürzeste Kantenfolge. Man kennzeichne sie, wenn nötig, neu als u 1 . . . u i+1 u1 . Man setze i := i + 1.
3. Wenn W i alle Knoten beinhaltet, beende den Algorithmus, sonst führe Schritt 2 aus.

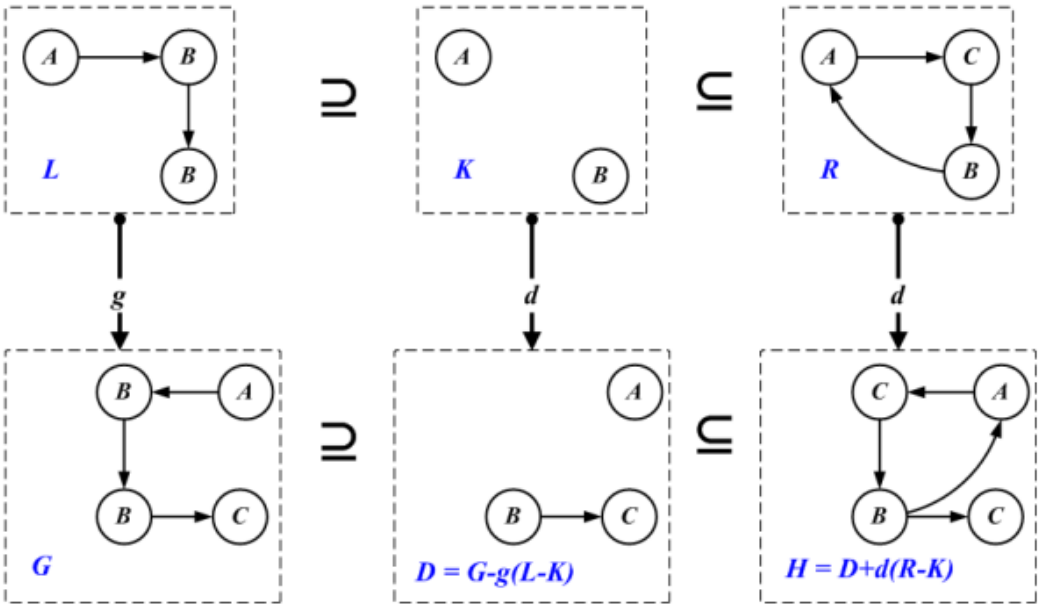
Graphmorphismen

Graphmorphismen bestehen aus struktur- und markierungerhaltenden Abbildungen zwischen Graphen:

- bilden Knoten auf Knoten und Kanten auf Kanten ab,
- bewahren Quelle und Ziel von Kanten,
- bewahren Markierungen.

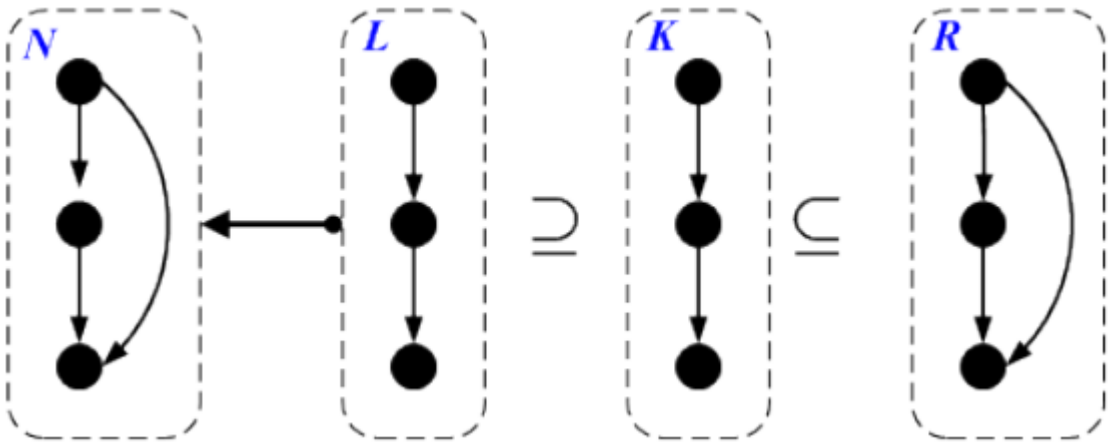
Seien G und H Graphen über C. Ein Graphmorphismus f : G → H von G
nach H ist ein Paar von Abbildungen f = (fV : VG → VH , fE : EG → EH), so dass für alle e ∈ E G und alle v ∈ V G gilt:
Bewahrung von Quelle und Ziel und Bewahrung von Markierungen

Graphersetzung Regeln



Wähle ein Vorkommen von L in G, d.h. einen Graph- morphismus g : L →G.
Überprüfe die Kontakt- und Identifikationsbedingung.
D = G − g(L−K).
H = D+(R−K).

Kontaktbedingung: Es dürfen keine hängenden Kanten entstehen.
Identifikationsbedingung: Je 2 Knoten sind identifizierbar, oder beide sind in K
Beide Bedingungen zusammen werden **Klebebedingungen** genannt



NAC:
Bilden der transitiven Hülle eines (gerichteten) Graphen.

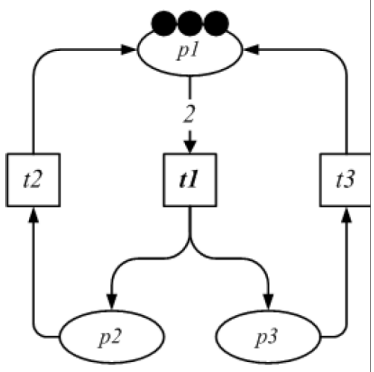
Keine Operation, wenn transitive Kante schon vorhanden

Definition (Markiertes S/T-Netz)

Ein (markiertes) *S/T-Netz* ist ein 4-Tupel $N = (P, T, W, M_0)$, für das gilt:

1. P und T sind Mengen, deren Elemente *Stellen (places)* bzw. *Transitionen* genannt werden mit $P \cap T = \emptyset$.
2. $W : (P \times T) \cup (T \times P) \to \mathbb{N}_0$ ordnet jeder Kante ihr *Kantengewicht* zu.
3. Und die Anfangsmarkierung $M_0 : P \to \mathbb{N}_0$ beschreibt die Verteilung der Token.

Beschreiben Sie das folgende S/T-Netz formal:



$N = (P, T, W, M_0)$ mit:
$P = \{p1, p2, p3\}$
$T = \{t1, t2, t3\}$

$$W(x,y) = \begin{cases} 2 & ; \text{falls } (x,y) = (p1,t1) \\ 1 & ; \text{falls } (x,y) \in \{(p2,t2),(p3,t3), \\ & \quad (t1,p2),(t1,p3),(t2,p1),(t3,p1)\} \\ 0 & ; \text{sonst} \end{cases}$$

$$M_0(x) = \begin{cases} 3 & ; \text{falls } x = p1 \\ 0 & ; \text{sonst} \end{cases}$$

Aktivierung

Eine Transition t ist unter einer Markierung M **aktiviert** $M[t\rangle$, wenn jede Stelle im Vorbereich der Transition mindestens soviele Token enthält, wie das Gewicht der entsprechenden eingehenden Kante vorschreibt.

Schalten

Eine Transition t **schaltet** $M[t\rangle M'$, wenn Token aus dem Vorbereich entfernt werden und Token im Nachbereich hinzugefügt werden. Die Anzahl der entfernten bzw. hinzugefügten Token wird NUR von den entsprechenden Kantengewichten bestimmt. M' ist die **Folgemarkierung**.

Definition (Vorbereich, Nachbereich)

Für einen Knoten $x \in P \cup T$ eines S/T-Netzes $N = (P, T, W)$
bezeichnet $\bullet x = \{y \mid W(y,x) > 0\}$ den *Vorbereich* und
$x\bullet = \{y \mid W(x,y) > 0\}$ den *Nachbereich* von x.

Definition (Schaltverhalten)

Sei $N = (P, T, W)$ ein S/T-Netz.

1. Eine Transition $t \in T$ heißt *M-aktiviert*, falls für alle $p \in \bullet t : M(p) \geq W(p,t)$ ... wird durch $M[t\rangle$ notiert.
2. Eine M-aktivierte Transition $t \in T$ bestimmt eine *Folgemarkierung* M' von M durch $M'(p) = M(p) - W(p,t) + W(t,p)$ für alle $p \in P$.
   t schaltet von M nach M'
   ..... wird durch $M[t\rangle M'$ oder $M \xrightarrow{t} M'$ notiert.

Eigenschaften von S/T-Netzen
Sei N ein S/T-Netz und MG sein Markierungsgraph, EG sein Erreichbarkeitsgraph
Beschränktheit
Für endliche markierte Netze NM0 gilt: NM0 ist beschränkt gdw. der
Erreichbarkeitsgraph EG von NM0 endlich ist. Wenn also nicht unendlich neue Token generiert werden
Erreichbarkeit
Eine Markierung M eines markierten Netzes NM0 heißt **erreichbar**, falls M ∈ EG
Eine Transaktion is erreichbar, wenn es im Markierungsgraph einen Pfad zu dieser gibt.
Lebendigkeit
N ist *lebendig*: Von jeder erreichbaren Markierung aus ist für jede Transition t eine Markierung erreichbar, die t aktiviert.
Markierung M ∈ MG **lebendig** in N bzw. NM0 , falls jede Transition t ∈ T, M-erreichbar ist.
Transition t ∈ T **lebendig** in NM0 , wenn sie für alle Markierungen M ∈ EG M-erreichbar ist.
Eine Transition t 2 T heißt **tot** in einer Markierung M, wenn sie nie erreicht werden kann
Verklemmung
Markierung M ∈ MG heißt **Verklemmung**, wenn kein t ∈ T, M-aktiviert ist.
Verklemmungsfreiheit
N heißt **verklemmungsfrei** (auch: schwach lebendig), falls N keine Verklemmung M ∈ EG besitzt.
Reversibilität