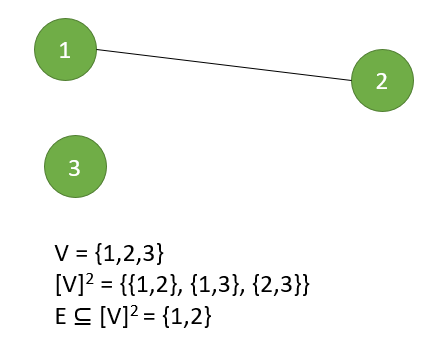
1. **Was ist ein Graph?**

* 2er-Tupel aus Menge V...Knoten und E...Kanten (E ⊆ [V]^2), Menge aus Kanten und eine Menge aus Knoten
* G = (V,E), Reihenfolge der Tupel egal
* <e = {u,v} -> Kante zwischen Knoten u und v

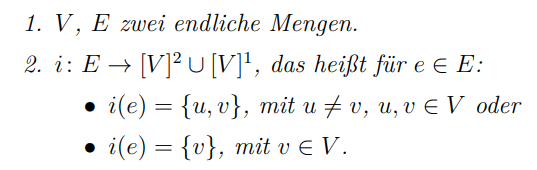


1. **Welche Arten von Graphen kennen Sie?**

* Gerichtet
  + Kanten haben eine Richtung à Menge E besteht aus Tupeln
  + e = (u,v) != e = (v,u)
  + e = (v,v) ist eine Schleife
  + E ⊆ V x V
  + Eingangsgrad (wie viele Kanten kommen zu Knoten)
  + Ausgangsgrad (wie viele Kanten verlassen Knoten)
* Ungerichtet
  + Kanten haben keine Richtung
  + e = {u,v} == e = {v,u}
  + Nur Grad -> keine Unterscheidung Eingang / Ausgang
    - {u ∈ V | v ∈ e}
* Multi Graph
  + Mehrerer Kanten zwischen denselben beiden Knoten
  + e1 = {u,v} != e2 = {u,v} != e3 = {u,v}
* Gewichtet (keine eigene Art – gilt für alle drei oberen)
  + Sei G = (V, E) ein Graph und ω : E → N mit Gewichtsfunktion ω.
  + Das Paar (G, ω) ist dann ein gewichteter Graph.

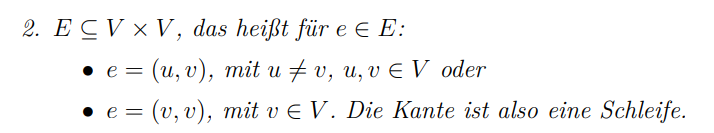
1. **Wie unterscheiden sich gerichtete von ungerichteten Graphen in der Definition?**

* Ungerichtet:
  + Ein (ungerichteter) Graph G ist ein 2-Tupel G = (V,E), wobei V eine endliche Menge ist und E ⊆ [V]2



Kante wird als Menge repräsentiert, da Reihenfolge egal ist

* Gerichtet:



Kante als Tupel repräsentiert, da Reihenfolge ausschlaggebend für Richtung ist.

* Unterschied:
  + Ungerichtet: Kantenmenge ist Teilmenge Menge aller 2-elementigen Teilmengen
  + Gerichtet: Kantenmenge ist Teilmenge vom Kartesischen Produkt der Knoten

1. **Wie werden Graphen dargestellt?**

* Adjazenz Matrix
  + Wir können die Knoten des Graphen als Spalten/Zeilen in der Matrix auffassen
  + Die Kanten werden als die Einträge in der Matrix dargestellt -> 1 es gibt eine Kante 0 es gibt keine (bei gewichteten Graphen wird Gewicht in den Spalten dargestellt)
  + Ungerichtete Graphen immer symmetrisch
* Adjazenz Liste
  + Für jeden Knoten werden die Nachbarn in einer Liste angegeben
* Grafische Darstellung
  + Graph als Menge von Knoten und Kanten
  + Knoten als Punkte
  + Kanten als Striche zwischen Knoten (ungerichtet)
  + Kanten als Pfeile (gerichtet)

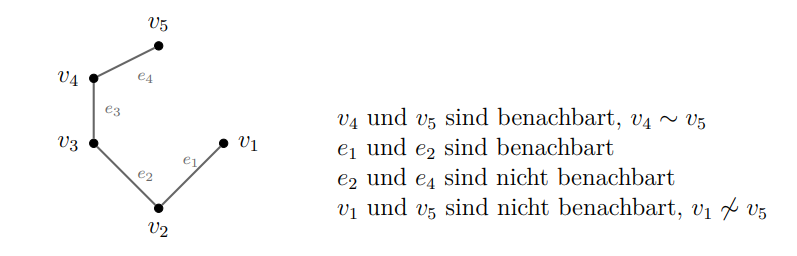
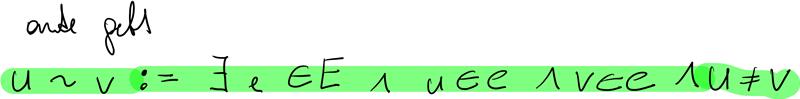
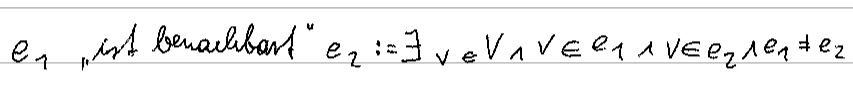
1. **Was ist ein gewichteter Graph?**

* Kanten erhalten Gewicht
* Gewichtsfunktion w weist jeder Kante Zahl zu
* Kann sowohl bei ungerichtetem als auch gerichtetem Graph vorkommen

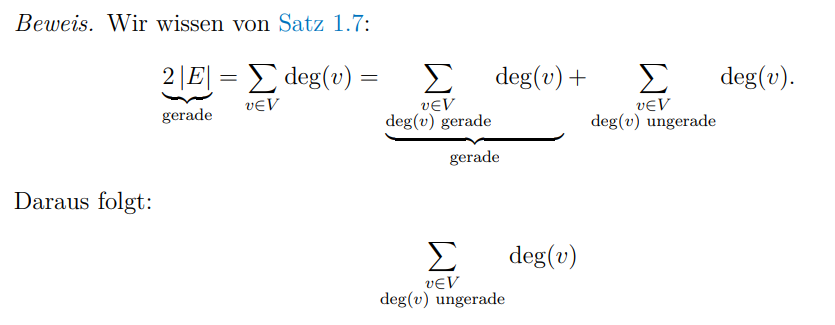
1. **Wie wird das Kantengewicht mathematisch definiert?**

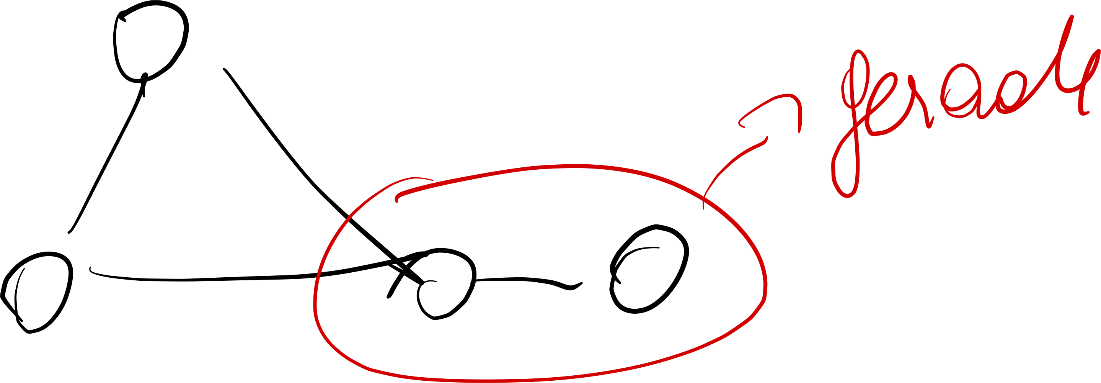
* Über eine Gewichtungsfunktion ω: E -> ℕ
* Gewichtsfunktion = Funktion, die jede Kante in einem Zahlenraum abbildet
* G = (V,E)
* Paar (G, ω) ist ein gerichteter Graph

1. **Was ist Nachbarschaft in Graphen?**

* Alle Knoten die mit Kanten verbunden sind
* Zwei Knoten u,v ∈ V heißen benachbart (= adjazent) wenn es eine Kante gibt die u und v verbindet
* u,v e V, u != v, es existiert ein e = {u,v}
* u∼v heißt benachbart (adjazent)
* Auch Kanten können benachbart sein (= inzident) wenn e1 != e2 aber ein Knoten auf beiden Kante liegt
* 
* 
* Definition für benachbarte Kanten e1 und e2 (Inzidenz):
* 

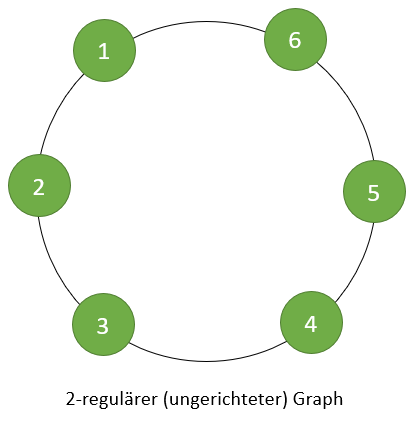
1. **Was besagt das Handshaking Lemma?**

* Die Anzahl an Knoten mit ungeradem Knotengrad ist gerade 



1. **Was ist ein regulärer Graph?**

* Jeder Knoten hat gleich viele Kanten = jeder Knoten hat denselben Knotengrad = alle Knoten haben gleich viele Nachbarn
* deg(v) = r für alle v e V
* Isolierter Knoten -> Grad/Nachbarschaft 0
* r-regulärer Graph: z.B. 2 regulärer Graph



1. **Ist Graph H ein Teilgraph von G (für zwei beliebige Beispiele von G und H)**

* Knoten müssen gleiche Bezeichnung haben
* V2 darf nicht gerichtet sein, wenn V1 ungerichtet ist und vice versa
* Knoten Induziert:
  + Man wählt eine Teilmenge V2 der Knoten aus und übernimmt alle Kanten von G zwischen Knoten aus V2.
  + Teilmenge von V2 und übernimmt **alle Kanten**
  + jeder Knoten in G2 ist gleich verbunden wie in G1Ein Bild, das Uhr enthält.

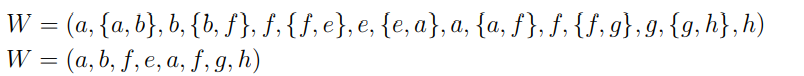
    Automatisch generierte Beschreibung
* Kanten Induziert:
  + Teilmenge E2 von E1
  + übernimmt alle End-Knoten
  + alle Kanten in E2 haben selbe Endpunkte (Start/Endknoten) wie in Graph 1

Ein Bild, das Text enthält.

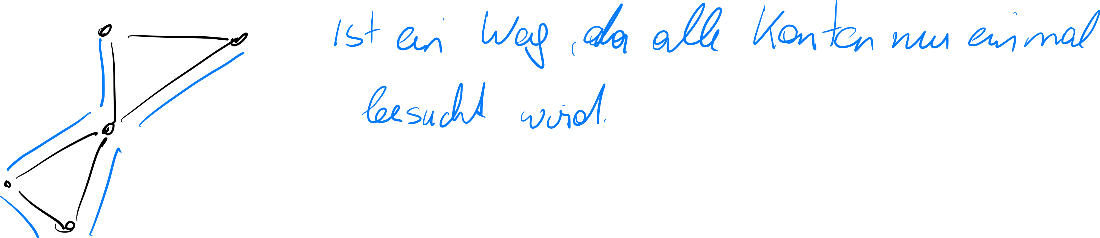
Automatisch generierte Beschreibung

* Spannender Teilgraph: V1 = V

1. **Was ist eine Wanderung?**

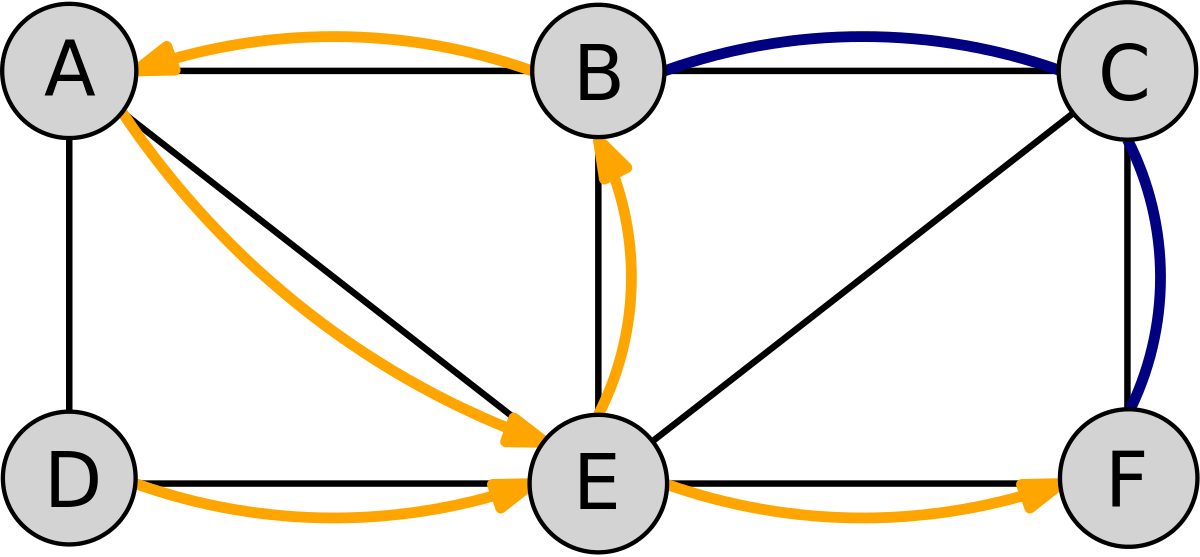
* eine u,v Wanderung in einem Graph ist endliche Abfolge von Knoten und Kanten
* u … Anfangsknoten, v... Endknoten
* Mächtigkeit |W| = n, Länge der Wanderung (Kantenanzahl)
* In einer Wanderung können sowohl Knoten als auch Kanten **mehrfach** vorkommen
* geschlossen - wenn Anfang = Endpunkt
* trivial Wanderung ist W = v, (v Element aus V)
* 
* Es gilt Pfad => Weg => Wanderung
* VORSICHT: Auf Klammerung achten.
* 
  + W1 bis W5 sind Wanderungen als Knoten Abfolge
  + W6 ist einfache Menge

**Was ist Weg?**

* Wanderung ist Weg falls alle KANTEN in W verschieden sind
* 

**Was ist Pfad?**

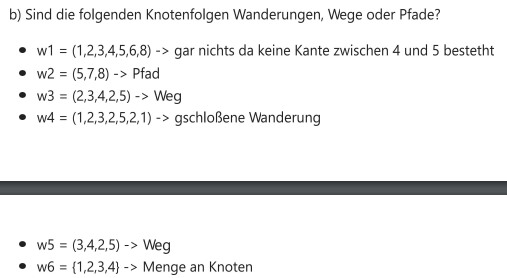
* Wanderung W ist Pfad falls alle KNOTEN und Kanten in W verschieden sind
* Daher kann es keine geschlossenen Pfade außer die Triviale Wanderung geben



1. **Ist die Knotenfolge W (beliebiges Beispiel) eine Wanderung, Weg, Pfad oder Zyklus?**

Ein Bild, das Uhr enthält.

Automatisch generierte Beschreibung



1. **Wann heißt ein Graph zusammenhängend?**

* G ist zusammenhängend, wenn es von je zwei Knoten aus V eine Wanderung vom einen zum anderen Knoten gibt
* Dann gilt:
  + G ist eulerisch, daher es gibt einen Eulerweg
  + Jeder Knoten V in G hat einen geraden Grad à Eulerkreis
  + Eulerweg: Es gibt genau zwei oder keine Knoten mit ungeradem Grad, alle anderen sind gerade à in Knoten mit ungeradem Grad starten und enden



* + Links zusammenhängend
  + Rechts nicht zusammenhängend

1. **Was ist in einem Graph eine Brücke? Geben Sie ein Beispiel in einem zusammenhängenden Graph mit |E| >= 5!**

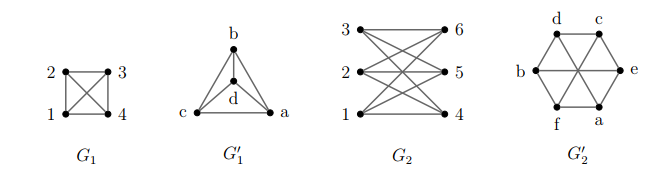
* Brücke ist die Kante die Graph verbindet, wenn diese entfernt wird dann zerfällt Graph in zwei Teilgraphen
* Wenn Graph ohne Kante e nicht zusammenhängend dann ist e eine Brücke
* Zusammenhangskomponente ist Anzahl der nicht zusammenhängenden Graphen Teile (Teilgraphen), wenn zusammenhängend dann 1 sonst n
* G = (V,E) zusammenhängend, e ∈ E dann ist e Brücke wenn G1 = (V,E\{e}) mehr Zusammenhangskomponenten hat als G

Ein Bild, das Text, Uhr, Vektorgrafiken enthält.

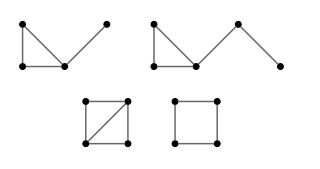
Automatisch generierte Beschreibung

1. **Was bedeutet „zwei Graphen sind isomorph“?**

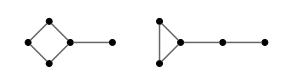
* Gleiche Anzahl an Kanten, Knoten
* Gleiche Nachbarschaften (daher Knotengrade)
* Struktur ident bis auf “Umbenennung” der Knoten
* Naive: die beiden Graphen sind bis auf die Beschriftung gleich
* |V1| = |V2| und |E1| = |E2| und deg(v\_1) = deg(v\_2) für alle Knoten in V1, V2
* Isomorphe Graphen:



* Nicht isomorphe Graphen (weil Kanten bzw. Knoten unterschiedlich sind):

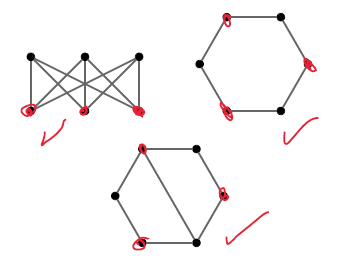


* Nicht isomorpher Graph (weil Nachbarschaftsverhältnis verletzt ist):



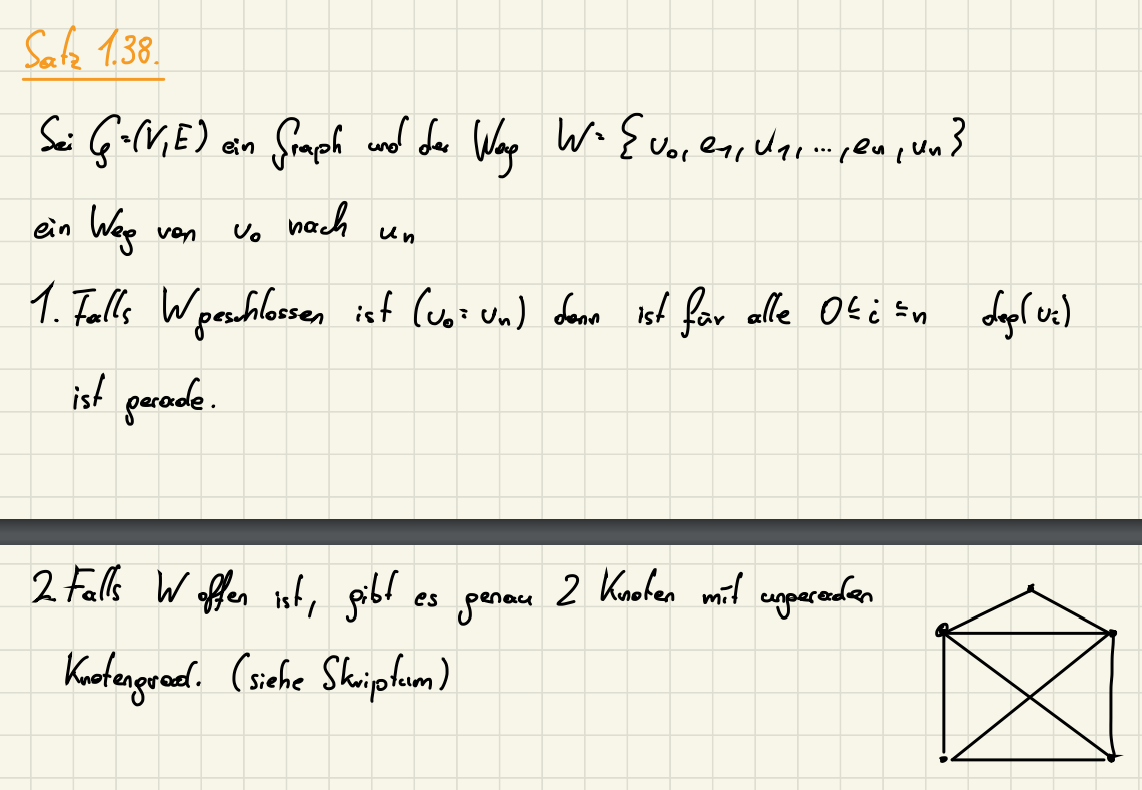
1. **Was ist ein bipartiter Graph?**

* Man kann den Graphen in zwei Knotenmengen teilen, sodass es nur Kanten zwischen den beiden Knotenmengen gibt, aber nicht innerhalb
* Trenne Knotenmenge in zwei Mengen V1, V2 auf; in Folge darf es nur Kanten geben, die zwischen je einem Knoten aus Menge V1 und V2 bestehen, nicht aber zwischen zwei Knoten aus derselben Menge
* Partition {V1,V2} mit V = V1 ∪ V2, V1 ∩ V2 = { } gibt bei der gilt für alle u, v ∈ V1 ⇒ u ≁ v und für alle u, v ∈ V2 ⇒ u ≁ v, also Kanten existieren höchstens zwischen Knoten aus V1 und V2
* Beispiel HÜ roter Knoten darf mit keinem anderen roten Knoten verbunden sein
* Ein Graph ist bipartit, wenn er einen Kreis mit einer geraden Anzahl an Kanten enthält
* Wie evaluieren?
  + Zähle Kanten Anzahl
  + Breitensuche (Ebene für Ebene durchsuchen)-> Starte bei Knoten und schaue alle Nachbarn an, dann die Nachbarn der Nachbarn (Stack)
  + Tiefensuche (Pro Strang bis zu den Blättern durchsuchen) -> wir gehen zuerst einen ganzen Pfad durch, alle Kindknoten aller Kindknoten



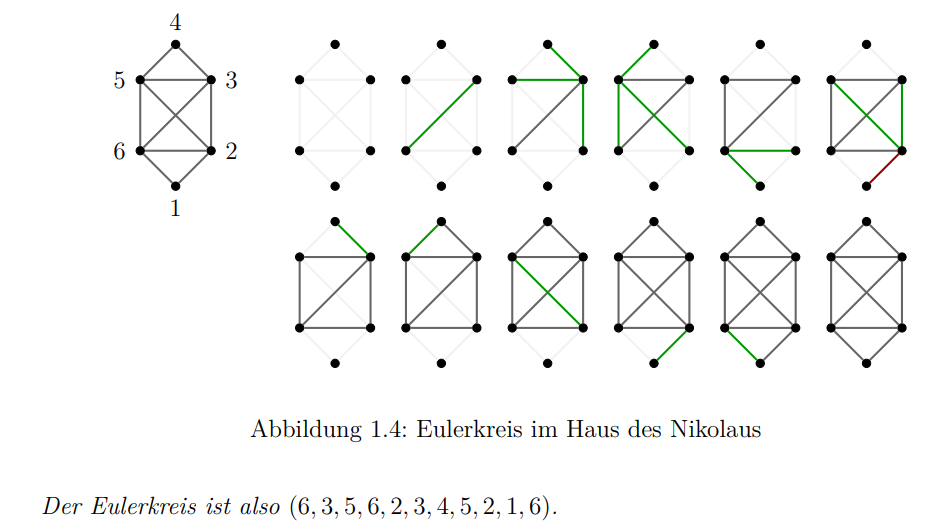
1. **Wann ist ein Graph eulersch?**

* Eulerscher Graph: wenn es einen Eulerkreis gibt -> gibt Eulerkreis immer dann, **wenn der Grad aller Knoten gerade ist** (jeder Knoten muss einen geraden Grad haben) + Graph muss zusammenhängend / geschlossen sein
* Eulersch (geschlossen) vs. Semi-eulersch (offen): das eine ist ein Zyklus und das andere nur ein Weg (Eulerweg -> zwei Knoten dürfen einen ungeraden Knotengrad haben: und zwar der Start und Endknoten)
* Eulerweg wenn Weg/Wanderung alle Kanten in G enthält, alle Kanten GENAU einmal
* Eulerkreis wenn ein geschlossener Eulerweg existiert
* Für einen Eulerkreis in einem vollständigen Graphen muss die Anzahl an Knoten ungerade sein, damit die Grade aller Knoten gerade sind
* Weg geschlossen, dann hat jeder Knoten eine gerade Anzahl an Degree
* Wenn offen dann genau 2 Knoten mit ungeradem Degree, sonst kein Eulergraph



1. Gegeben ein beliebiger Graph G, ist es hier möglich einen Eulerkreis zu konstruieren?
   * Oben genannte Bedingungen prüfen (nur gerade Knotengrade)
2. Geben Sie, falls möglich, den Eulerweg oder Eulerkreis in Graph G (beliebiger Graph) an!
   * Fleury durchspielen:
   * Zufälligen Startknoten wählen, von dort aus zufälliger Kante
     + Wenn Knoten mit ungeradem Grad existieren, eignen sich diese gut als Startknoten (siehe “Haus des Nikolaus”)
   * Kante nur wählen, wenn keine Brücke (außer beim letzten Knoten)
   * Mit Knoten am anderen Ende der Kante fortfahren
3. Welche Algorithmen kennen Sie zur Berechnung von Eulerkreisen? Was sind die Unterschiede?

* **Zwiebelschalen-Algorithmus** (wurde in VO nicht besprochen)
* **Fleury Algorithmus**
  + Findet Eulerkreis
  + muss zusammenhängender Graph sein
  + Initialisiere alle Kanten in Menge F, Lösungsmenge der Kanten ist leer
  + Wir beginnen bei Zufallsknoten und wählen Zufallskante
  + entferne Kante aus Menge F und gib zur Lösungsmenge hinzu (falls KEINE BRÜCKE)
  + neuer Knoten ist Ende von Kante
  + wiederhole Schritte



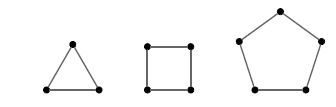
* **Hierholzer Algorithmus (nicht im Skriptum)**
  + identifiziert Zyklen und hängt diese zusammen
  + basiert auf Tiefensuche
  + alle Knoten weg dann Knotenliste
    - wird als zwei Stack aufgebaut, temporär und euler
    - wir nehmen Knoten und entfernen alle seine Kanten, dann hängen wir ihn dazu
    - wenn wir nicht alle Kanten aus dem Knoten löschen konnten, nimm einen Random Nachbarn und wiederhole die Schritte
    - nur Knoten ohne Nachbar kommt in Euler Stack
  + rekursiv, viel komplexer

1. Gegeben der Fleury Algorithmus in folgendem Zustand und nebenstehender Graph mit Kanten E \ F. Welche Kante würde als nächstes ausgewählt und warum?
   * Auf Brücken achten!
2. **Bei der Auswahl der nächsten Kante im Fleury Algorithmus muss eine mögliche Brücke immer zuletzt gewählt werden? Wie viele solcher Brücken kann es ausgehend von ein und demselben Knoten maximal geben (unter der Annahme, dass der Graph eulersch ist)?** (BITTE FEEDBACK) danke

* Während dem Durchlaufen des Graphen darf der aktuelle Knoten maximal eine Brücke haben
* Im Eulergraph selbst darf keine Brücke existieren (Ansonsten kann kein Eulerkreis enthalten sein, denn dann wäre es nur ein Eulerpfad und somit auch kein Eulergraph mehr)
* In einem normalen Graphen kann es so viele Brücken geben, wie es Kanten gibt

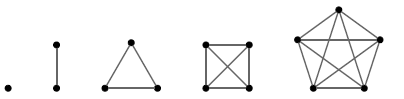
1. **Was bedeutet es, wenn ein Graph die Hamilton Eigenschaft besitzt?**

* Es gibt einen Pfad, der jeden Knoten genau einmal enthält
* Hamilton-Kreis: Zyklus aller Knoten, alles einmal, wir müssen nicht alle Kanten verwenden
* Hamiltopfad wird Kreis, wenn geschlossen
* gibt es einen Hamiltonkreis, ist Graph hamiltonsch
* Jeder Graph ist Hamiltonsch mit wenigstens n >= 3 Knoten, wenn alle Knoten mindestens Grad n/2 aufweisen, kann aber auch ohne gelten
* n-Zyklus Graphen sind hamiltonsch wenn |V| >= 3

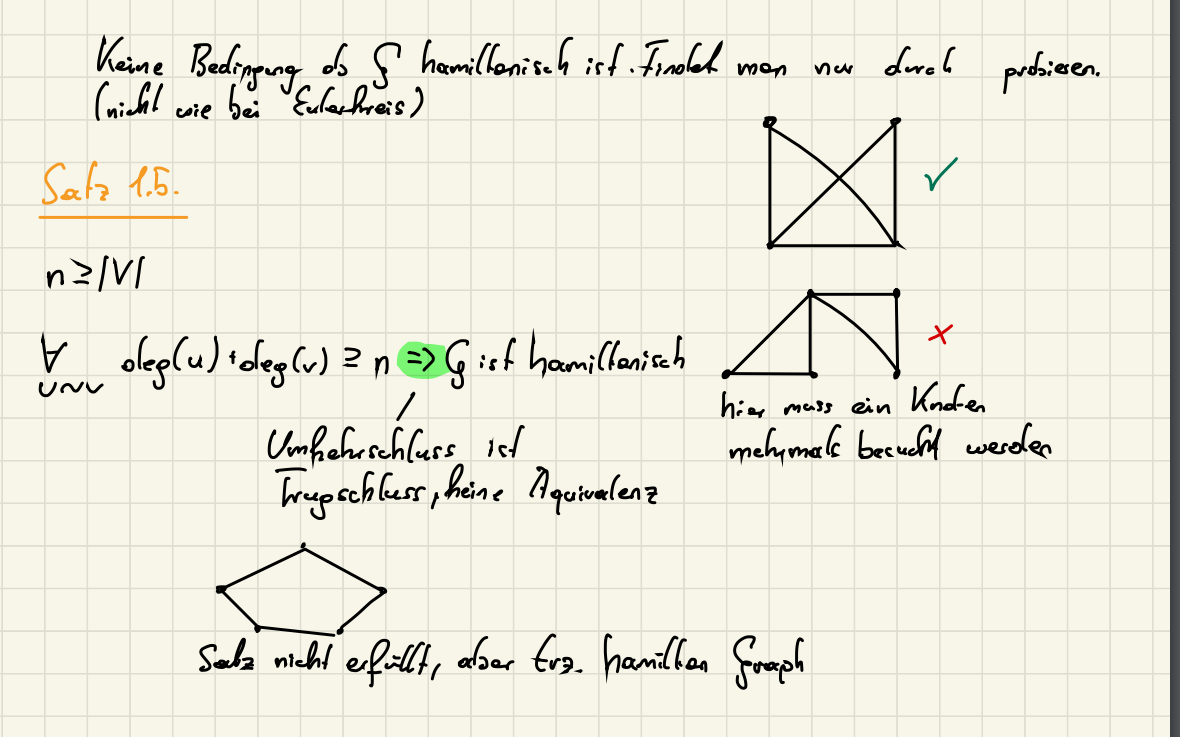


1. **Diskutieren Sie die Euler- bzw. Hamiltoneigenschaft für vollständige Graphen K\_n mit n > 3!**

* Vollständige Graphen Kn sind hamiltonsch wenn |V| >= 3

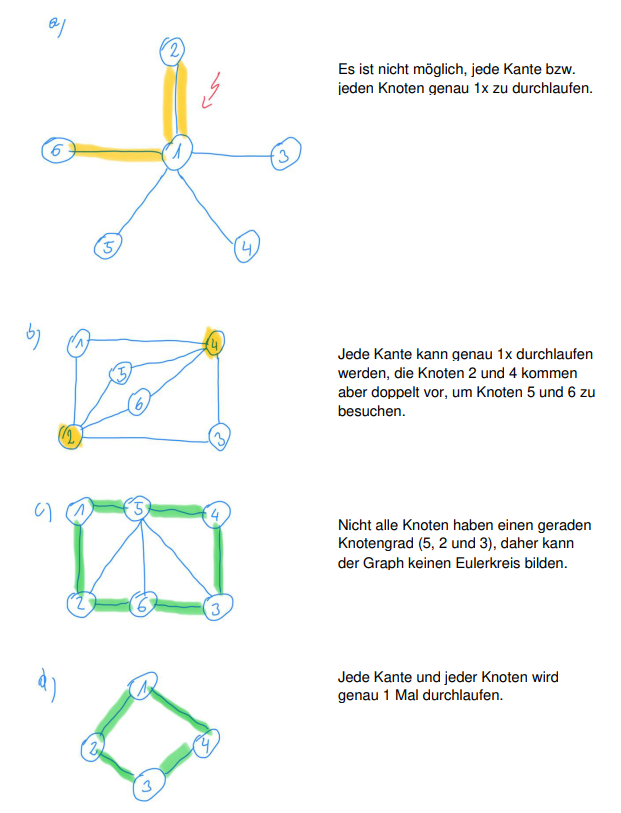


* (Wenn für nicht-benachbarte Knoten gilt, dass die Degrees >= der Dimension n sind, dann ist Graph Hamiltonisch (Satz von Ore) à in einem vollständigen Graphen sind alle Knoten benachbart)
* n ist Anzahl an Knoten
* Eulerkreis: Wenn n ungerade ist, enthält Kn einen Eulerkreis ([Quelle](http://discrete.openmathbooks.org/dmoi2/sec_paths.html#p-2638))
* Graph kann auch ohne diese Eigenschaft Hamiltonsich sein – Umkehrschluss ist Trugschluss



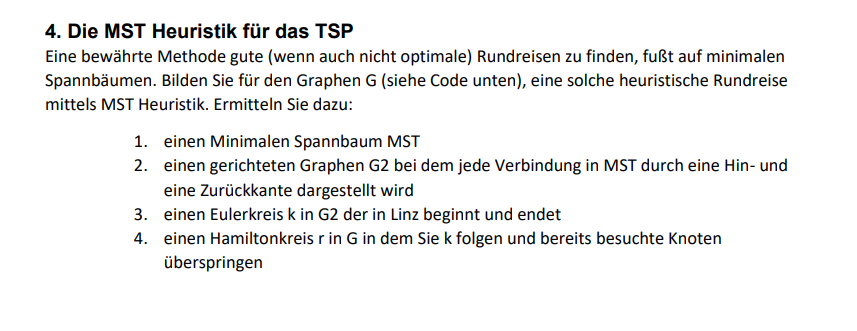
1. **Vergleichen Sie die Lösbarkeit des Eulerkreisproblems mit dem des Hamiltonkreisproblems!**

* Hamilton nur mit Probieren, keine allgemein gültige Algorithmische Lösung, viel schwieriger à np-schwer Problem (nicht in Polynomialzeit lösbar)
* Euler geht algorithmisch, zB mit Fleury und es kann mit allgemein gültigen Regeln geprüft werden, ob es überhaupt einen Eulerkreis gibt
* Achtung, nicht jeder Eulergraph ist ein Hamiltongraph und umgekehrt
  1. kein Eulerkreis und kein Hamiltonkreis
  2. Eulerkreis, aber kein Hamiltonkreis
  3. Hamiltonkreis, aber kein Eulerkreis
  4. Eulerkreis und Hamiltonkreis



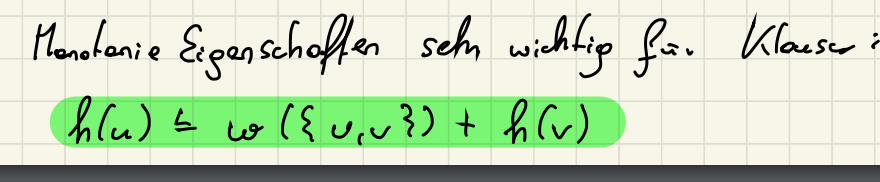
1. **Skizzieren Sie einen Algorithmus zur Lösung des Hamiltonkreisproblems!**

* Backtracking à Tiefensuche: Alles durchprobieren
* Minimum Spanning Tree Heuristic
  + Zuerst MST durch Prim oder Kruskal
  + Mache ungerichteten zu gerichteten Graphen à verdopple jede Kante in MST (Gegenkanten einfügen)
  + Bestimme Eulerkreis in MST (zB Fleury) à da Knoten immer mit 2 Kanten (hin- und zurück) verbunden sind, ist der Knotengrad jedes Knoten gerade und es MUSS einen Eulerkreis geben
  + neuer Graph mit allen Kanten, lösche alle Kanten, die bereits besucht wurden, neue Kante ist nächster Knoten aus dem Eulerkreis (Übung)



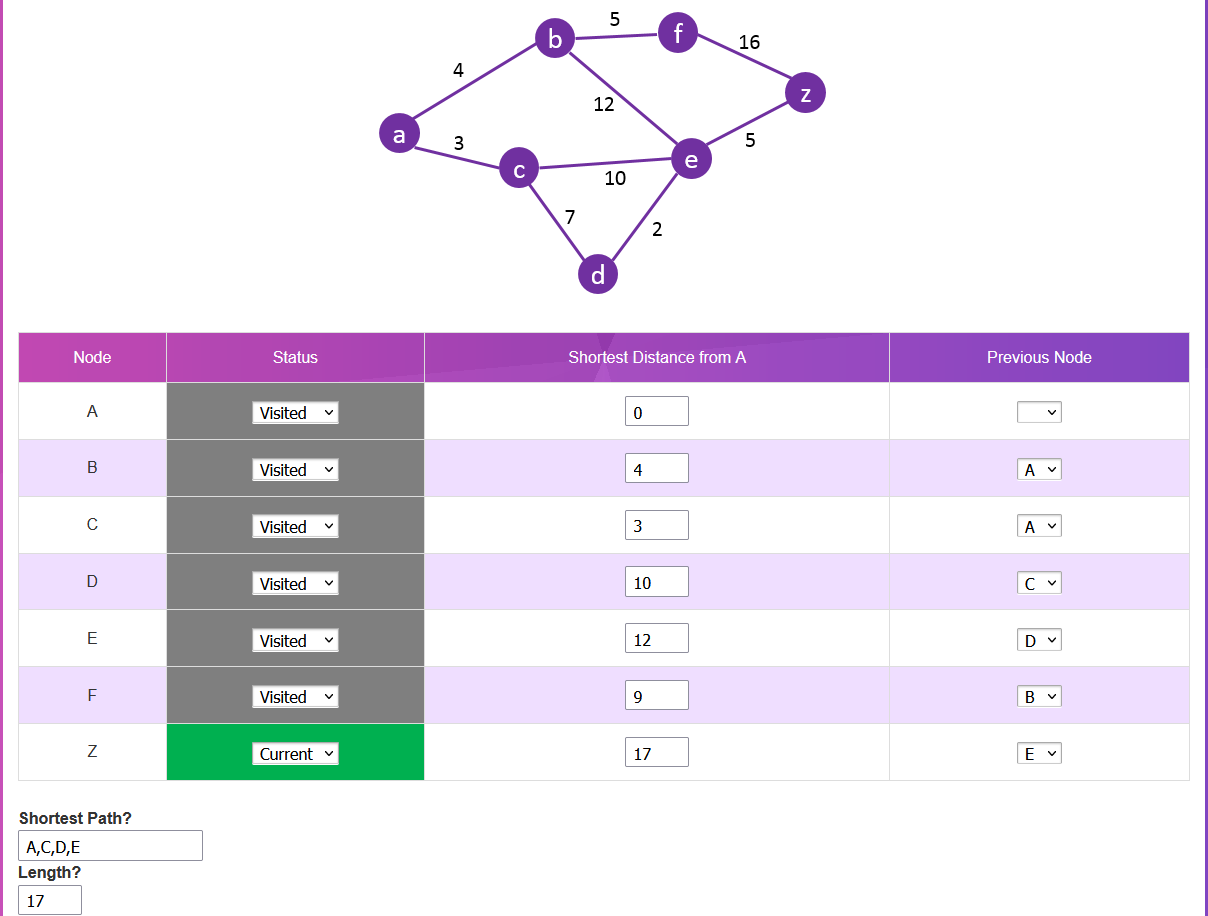
1. **Welche Algorithmen für die kürzeste Wegfindung kennen Sie? Vergleichen Sie diese und diskutieren Sie die Anwendbarkeit dieser Algorithmen!**

* Dijstra – Bidirectionaler Dijstra (Breitensuche)
  + Kürzester Weg von Start zum Ziel
  + Der Dijkstra Algorithmus ist ein sogenannter Greedy Algorithmus, der immer den günstigsten Weg vom Startknoten zu einem anderen Knoten wählt. Es werden alle anderen Knoten (in konzentrischen Kreisen = gleichmäßig in alle Richtungen) durchsucht, bis man den günstigsten Weg zum Zielknoten gefunden hat. Er hilft dir die kürzesten beziehungsweise kostengünstigsten Wege zu berechnen. Die Kantengewichte, so nennt man die Kosten, um von einem Punkt zum nächsten zu kommen, dürfen beim Dijkstra-Algorithmus nicht negativ sein.
  + Update: Berücksichtigt die Kosten vom Start bis zum aktuellen Knoten + die zusätzlichen Kosten für den nächsten Schritt
  + Bidirektional gehe vom Start und Zielpunkt gleichzeitig los. Sobald ein Punkt in beiden Dijkstras vorkommt, dient dieser als obere Schranke. Findet man erneut einen Punkt, der in beiden vorkommt und höhere Gesamtkosten als die obere Schranke hat, kann man den Algorithmus beenden.
  + Nur bei positiven Kantengewichten
* Bellman Moore Ford (Breitensuche)
  + nicht negative Zyklen, aber negative Kantengewichte erlaubt
  + kürzester Weg von Start zu allen anderen Knoten (arbeitet mit Stack)
  + Dijkstra ist fertig, sobald Zielknoten abgearbeitet wurde (in F ist). BMF muss so lange weiterlaufen, bis kein Knoten mehr am Stack liegt
* A\*
  + Der A\*-Algorithmus („A Stern“ oder englisch „a star“, auch A\*-Suche) gehört zur Klasse der informierten Suchalgorithmen. Er dient in der Informatik der Berechnung eines kürzesten Pfades zwischen zwei Knoten in einem Graphen mit positiven Kantengewichten und berücksichtigt zusätzlich eine Heuristik. Ist die Heuristik gut, findet A\* den kürzesten Pfad deutlich schneller als Dijkstra; bei einer falschen Heuristik findet A+ evtl. gar nicht den kürzesten Pfad.
  + Heuristic: unterschätzt die tatsächlichen Kosten; (streng) monoton fallend à je näher man zum Ziel kommt, desto kleiner wird die Heuristik
  + h(u) ≤ w(u,v) + h(v) ∀ u,v ∈ E



Breiten und Tiefensuche:

1. **Gegeben der Dijkstra Algorithmus in folgendem Zustand. Welche Kante würde als nächstes ausgewählt? Wie lautet der kürzeste Weg? Welchen Weg findet der Dijkstra Algorithmus?**



<https://www.101computing.net/dijkstras-shortest-path-algorithm/>

* Nimmt Wert aus der Liste “dist” mit den geringsten Kosten.

1. **Wie können Sie den Dijkstra Algorithmus potenziell noch beschleunigen?**

* Bidirektionale Suche
  + Wir starten bei zwei verschiedenen Punkten (Start & Ziel)
  + Machen von beiden aus einen Dijkstra
  + Wenn wir uns zum ersten Mal treffen, sind wir noch nicht fertig!
  + Sobald ein Punkt in beiden Dijkstras vorkommt, dient dieser als obere Schranke. Findet man erneut einen Punkt, der in beiden vorkommt und höhere Gesamtkosten als die obere Schranke hat, kann man den Algorithmus beenden.

1. **Wie lautet die Abbruchbedingung der bidirektionalen Suche?**

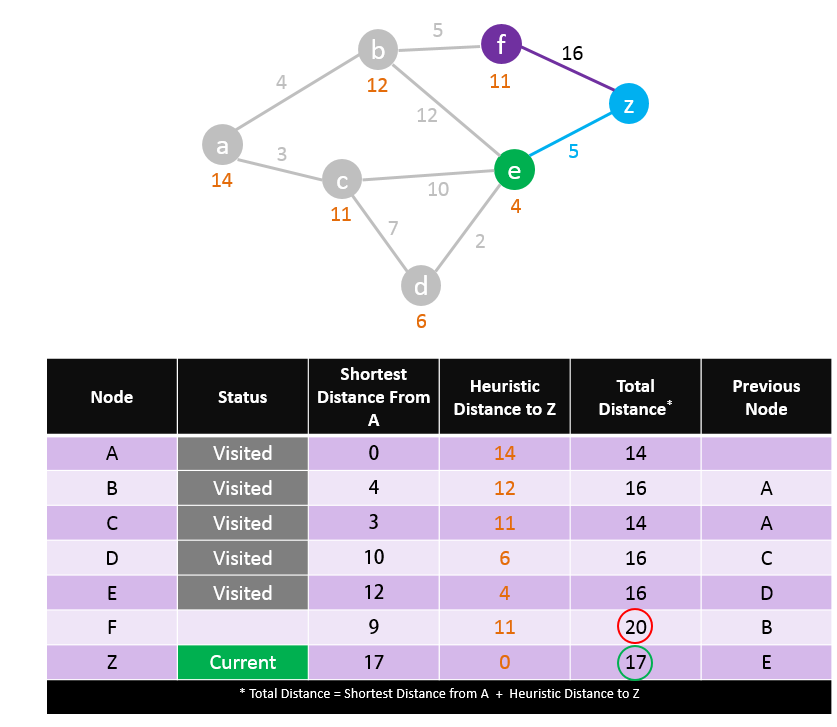
* Die Suche ist vorbei, wenn sich die beiden Suchräume treffen UND die Gesamtkosten des Weges (Summe aus beiden Dijkstras) höher sind als die obere Schranke
* Daher es gibt einen Pfad, der beide Punkten verbindet
* Dann berechnen wir zum ersten Mal die Schranke S mit der Entfernung des Pfades, den wir gefunden haben
  + nun berechnen wir einen neuen Pfad
  + wir wiederholen dies so lange, bis wir die Schranke nicht mehr reduzieren können
  + Der gefundene Pfad dient uns als obere Schranke S
  + Wir setzen die oberste Schranke initial auf unendlich
  + Wenn wir ein Update zu einem kürzeren Weg machen, der sich bereits in der Menge der jeweils anderen Suchrichtung befindet, aktualisieren wir S
  + Wir können die Suche abbrechen, wenn der kürzest mögliche Pfad aus beiden Richtungen größer als diese Schranke ist

1. **Welche Eigenschaft muss die Heuristik für den A\* erfüllen?**

* Monoton fallend
* Daher jeder folgende Wert muss kleiner oder gleich dem vorherigen sein
* Heuristik muss kleiner werden, je näher man zum Zielknoten kommt
* Heuristik schätzt bzw. unterschätzt die tatsächlichen Kosten

1. **Gegeben der A\* Algorithmus in folgendem Zustand. Welche Kante würde als nächstes ausgewählt? Wie lautet der kürzeste Weg? Welchen Weg findet der A\* Algorithmus?**

* Einfach Pfad mit kürzester Heuristik + Gewicht nehmen
* Obacht: Gewicht ist nicht zwingend gleich Heuristik!!



<https://www.101computing.net/a-star-search-algorithm/>

1. **Was ist ein Spannbaum, was ist ein minimaler Spannbaum?**

* Spanngraph
  + Teilgraph mit selber Knotenmenge
  + Unterschied ist Anzahl der Kanten
* Baum
  + Teilgraph der keine Kreise enthält
  + muss aber nicht selbe Knotenmenge haben
* Spannbaum
  + ungerichtet
  + maximal azyklisch à fügt man eine Kante hinzu, entsteht ein Zyklus
  + minimal zusammenhängend à entfernt man eine Kante, ist der Baum nicht mehr zusammenhängend
  + muss dieselbe Knotenmenge wie der Graph haben
* MST
  + Spannbaum mit minimalem Kantengewicht

1. **Ist der Spannbaum aus dem Dijkstra Algorithmus minimal?**
2. nicht immer minimal (berücksichtigt nicht die geringsten Kantengewichte von Knoten zu Knoten, sondern nur den insgesamt kürzesten Pfad vom Start zum Ziel à sobald Dijkstra alle Knoten durchsucht hat und beim Ziel ist, bricht er ab)
3. Dijkstra geht von einer source aus und verbindet sich von dort zu jedem Knoten.
4. stattdessen Prim / Kruskal
5. Bei Klausur -> NEIN!

Ein Bild, das Text, Gerät enthält.

Automatisch generierte Beschreibung

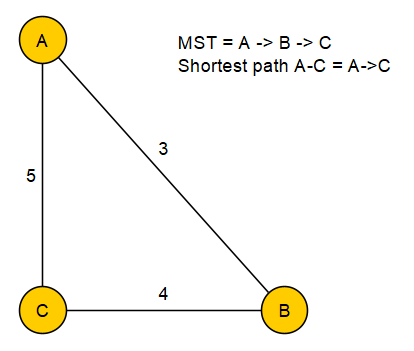
1. **Welche Algorithmen für die Berechnung von Spannbäumen kennen Sie? Diskutieren Sie die Unterschiede!**

* Prim
  + Beginne bei Startknoten
  + nimm in Nachbarschaft den Knoten mit geringstem Gewicht
  + gehe diese Kante und hänge sie in die Lösungsmenge
  + gehe zu Nachbarknoten
  + Solange bis bei Zielknoten
* Kruskal
  + alle Kanten
  + sortiere Gewichte aufsteigend
  + initialisiere alle Knoten
  + setzte Kanten mit niedrigstem Gewicht wieder ein
  + vermeide Zyklen
* Unterschiede
  + Kruskal muss im Vorfeld schon jede Kante betrachten -> Speicher und Laufzeit
  + Kruskal verwendet Union Find, schnelles Finden durch Repräsentanten
  + Prim merkt sich immer nur die laufende Distanz und die Knoten seiner Nachbarn

1. **Ist der minimale Spannbaum eindeutig, wann / wann nicht?**

* Eindeutig dann, wenn Kantengewichte eindeutig sortiert nur einmal vorkommt
* Nicht eindeutig, wenn es Zyklen im Graphen gibt, bei denen die einzelnen Kanten dieselben Gewichte aufweisen, dann mehrere Graphen
  + Quasi Wege durch den Graphen

1. **Genügt es einen minimalen Spannbaum zu finden, um in einem Graph einfach alle kürzesten Wege zu berechnen? Warum bzw. warum nicht?**

* BITTE UM FEEDBACK: Nein, da der MST nicht den Weg von einem bestimmten Start- zu einem bestimmten Zielknoten berücksichtigt, das übernimmt der Dijkstra Algorithmus.  
  Da der Dijkstra nicht zwingend einen minimalen Spannbaum liefert, muss der MST nicht zwingend den kürzesten Weg liefern => bin mir nicht sicher ob die Begründung wirklich schlüssig ist.
  + Bei einem MST müssen alle Knoten enthalten sein, bei einem kürzesten Pfad aber nicht. Somit kann es kürzeste Pfade geben, welche nicht (vollständig) auf dem MST liegen.
* Beispiel: Minimaler Spannbaum enthält nicht den kürzesten Pfad:
* MST kann Kante enthalten die hohen Kosten verursacht. Minimaler Pfad muss nicht alle Knoten besuchen.   
  

1. **Was ist ein Webgraph?**

* Gerichteter Graph
* Beschreibt Webseiten als Knoten und die Verbindungen/Links zwischen diesen als Kanten

1. **Erklären Sie das Random Surfer Modell, welche Annahmen sind in dem Modell enthalten?**

* Berechnungsgrundlage für PageRank.
* Verhalten des Internet
* Surfer bewegt sich auf zwei arten:
  + Url eingabe oder Lesezeichen -> Dämpfungsfaktor
  + Über andere Seiten auf die Seite
* Beim Random Surfer Model wird nun angenommen, dass der Link, der als nächstes angeklickt wird, zufällig ausgewählt wird.
* Wahrscheinlichkeiten lassen sich durch den PageRank ableiten
* In der Realität hat ein Nutzer ein Ziel und bewegt sich deshalb nicht zufällig über Links durchs Netz.

1. **Was bedeutet der Dämpfungsfaktor im Random Surfer Modell?**

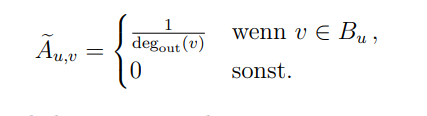
* Für den zufälligen Surfer bedeutet dieser Dämpfungsfaktor die Wahrscheinlichkeit weiterhin Links zu klicken, während mit Wahrscheinlichkeit d−1 auf eine zufällige Webseite gesprungen wird.

1. **Der Page Rank ist eine spezielle Form der Eigenvektorzentralität, was beschreibt der Page Rank?**

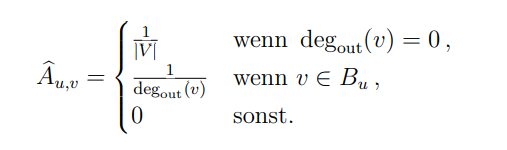
* Page Rank beschreibt Wahrscheinlichkeit, dass sich der Random Surfer auf einer bestimmten Website befindet
* Pagerank ist Sortierung der Knoten in unserem Webgraphen
* Seiten mit vielen Backlinks (eingehende Kanten) sollen weiter nach vorn sortiert werden
* Pagerank ist die Wahrscheinlichkeit, dass der User auf dem Knoten landet, wenn er immer zufällige Knoten ansteuert
* Rang kann durch den Grad an ausgehenden Knoten berechnet werden
* Immer zwischen 0 und 1
* Wird durch Dämpfungsfaktor erweitert
* Matrixdarstellung ist das Eigenvektorproblem, Matrix muss den Eigenwert 1 haben und Vektor darf nur positive Zahlen enthalten

1. **Wie kann der Page Rank für die Suche verwendet werden?**

* Für den Page Rank wird eine **Adjadenzmatrix** aufgestellt, in der die einzelnen Kanten aufsummiert eingetragen werden
* aus dieser leitet sich die Matrix **A\_tilde** ab, bei der wir die einzelnen Wahrscheinlichkeiten eintragen (transponiert und normalisiert)



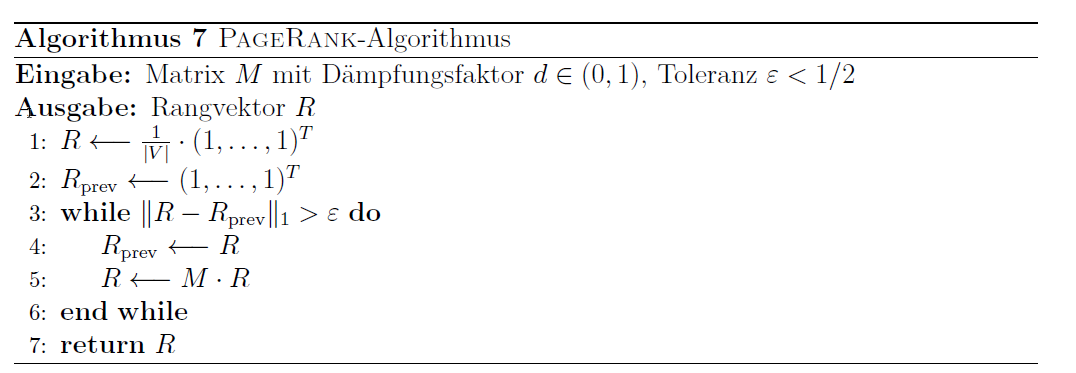
* **A\_hat** wird erweitert um die Bereinigung von Senken
  + Senke: Knoten der keine Out Degrees aufweist
  + von diesem springen wir mit bestimmter Wahrscheinlichkeit zu jedem anderen Knoten
  + Matrix wird um Rückverbindungen ergänzt, zB ¼ bei 4 Seiten



* Durch den Page Rank werden relevantere Webseiten weiter oben sortiert, da für sie die Wahrscheinlichkeit höher ist, besucht zu werden
* Page-Rank ist eine Variante der **Eigenvektor-Zentralität** und ist ein Maß für den Einfluss eines Knotens in einem Netzwerk.
* Der PageRank-Algorithmus ist eine spezielle Methode, die Linkpopularität einer Seite bzw. eines Dokumentes festzulegen. Das Grundprinzip lautet: Je mehr Links auf eine Seite verweisen, desto höher ist das Gewicht dieser Seite. Je höher das Gewicht der verweisenden Seiten ist, desto größer ist der Effekt. Das Ziel des Verfahrens ist es, die Links dem Gewicht entsprechend zu sortieren, um so eine Ergebnisreihenfolge bei einer Suchabfrage herzustellen, d. h. Links zu wichtigeren Seiten weiter vorne in der Ergebnisliste anzuzeigen.

1. **Welches Verfahren wird für die Berechnung des Page Rank verwendet?**

* Potenz/Power Methode
  + Diese Methode berechnet den dominanten Eigenwert einer Matrix und den dazugehörigen Eigenvektor unter der Annahme, dass der Eigenwert einfach ist.
  + dominanter Eigenwert einer Matrix und deren Eigenvektor
  + einfacher Eigenwert
  + Abbruchkriterium ist von Toleranz abhängig, wird als Norm dargestellt



* Vergleiche Iterationen aus HÜ 3

1. **Gegeben ein beliebiger Graph G, schätzen Sie die Reihenfolge der Knoten die sich durch den Page Rank ergeben würde! Begründen Sie Ihre Schätzung!**

* Hier immer auf Wahrscheinlichkeit achten
* Was ist immer die höchste Wahrscheinlichkeit? -> den Knoten nehme ich
* Aus Graph
  + Adjadenzmatrix erstellen
  + A\_tilde erstellen
  + A\_hat erstellen ohne Senken
  + Iterationen durchrechnen wäre immer bedingte Wahrscheinlichkeit
* Knoten mit höherer Anzahl an Eingangskanten werden einen höheren Rang haben
* Startknoten mit höchster Wahrscheinlichkeit?

1. **Beschreiben Sie 3 Zentralitätsmaße für Graphen!**

* Zwischenzentralität
  + Wie stark Knoten in kürzesten Pfaden zwischen beliebigen Knoten im Netzwerk vorkommt
* Eigenverktorzentralität (Pagerank)
  + Matrix mit Eigenwert und Eigenvektor
  + Mit welcher Wahrscheinlichkeit befindet sich das System in einem bestimmten Zustand?
* Nähezentralität
  + Ähnlich wie bei der Zwischenzentralität benötigen wir wieder die Distanz zwischen zwei Knoten. Diesmal wird jedoch die durchschnittliche Nähe zu allen anderen Knoten gemessen

1. **Was ist ein Netzwerk und was ist ein Fluss?**

* Netzwerk
  + Kann als Graph dargestellt werden
  + Knoten und Leitungen, diese entsprechen den Kanten
  + jede Leitung hat gewisse Kapazität
  + hat Quelle (Start) und Senke
  + gerichtet, gewichtet und orientiert
* Fluss
  + Fluss ist Abbildung eines Weges in einem Netzwerk
    - Ähnlich zu einem Pfad durch unseren Graphen
  + s-t Fluss: Bewegungen durch den Graphen von der Quelle zur Senke
  + Flusserhaltungsgesetzt: Summe was von einem Knoten weg geht ist Summe aus dem was in den Knoten eingegangen ist
  + Maximaler Fluss: Maximaler Wert aller möglichen Flüsse

1. **Geben Sie eine Definition des maximalen Flussproblems!**

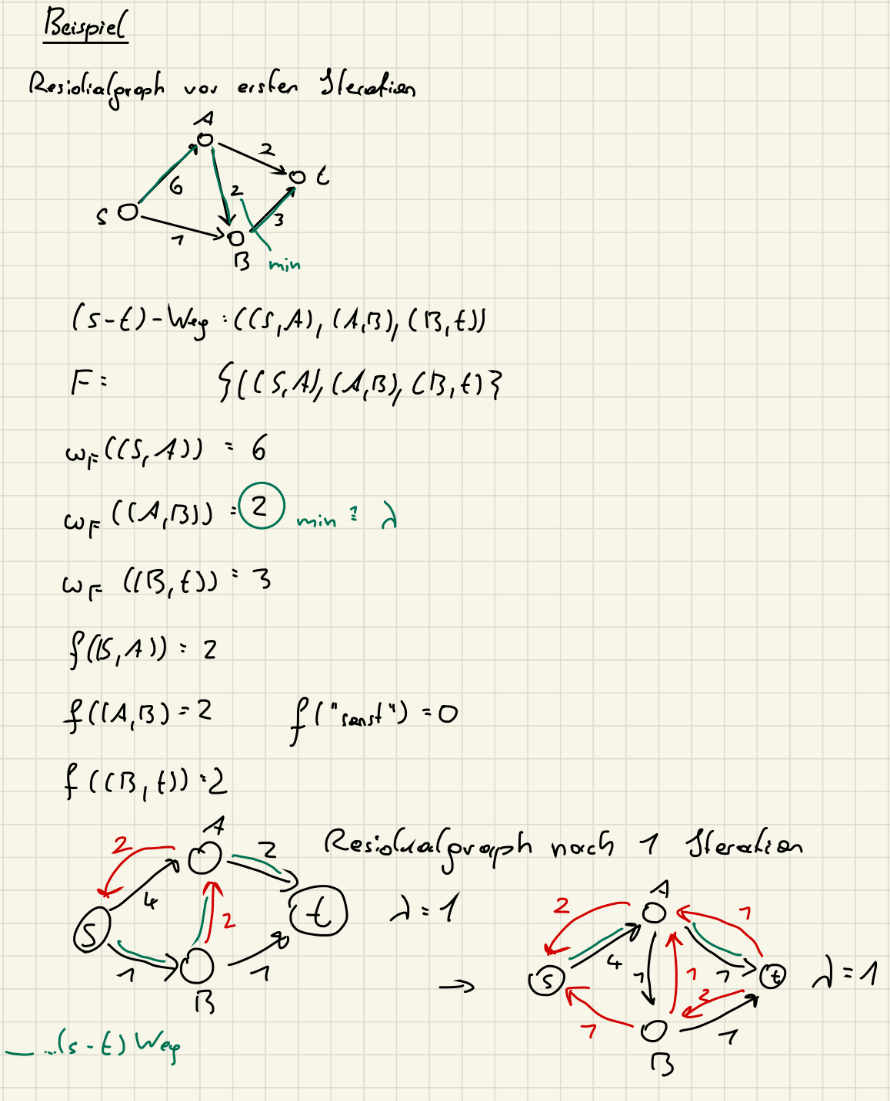
* Optimierungsproblem
  + Finde (s,t) Fluss mit maximalem Wert

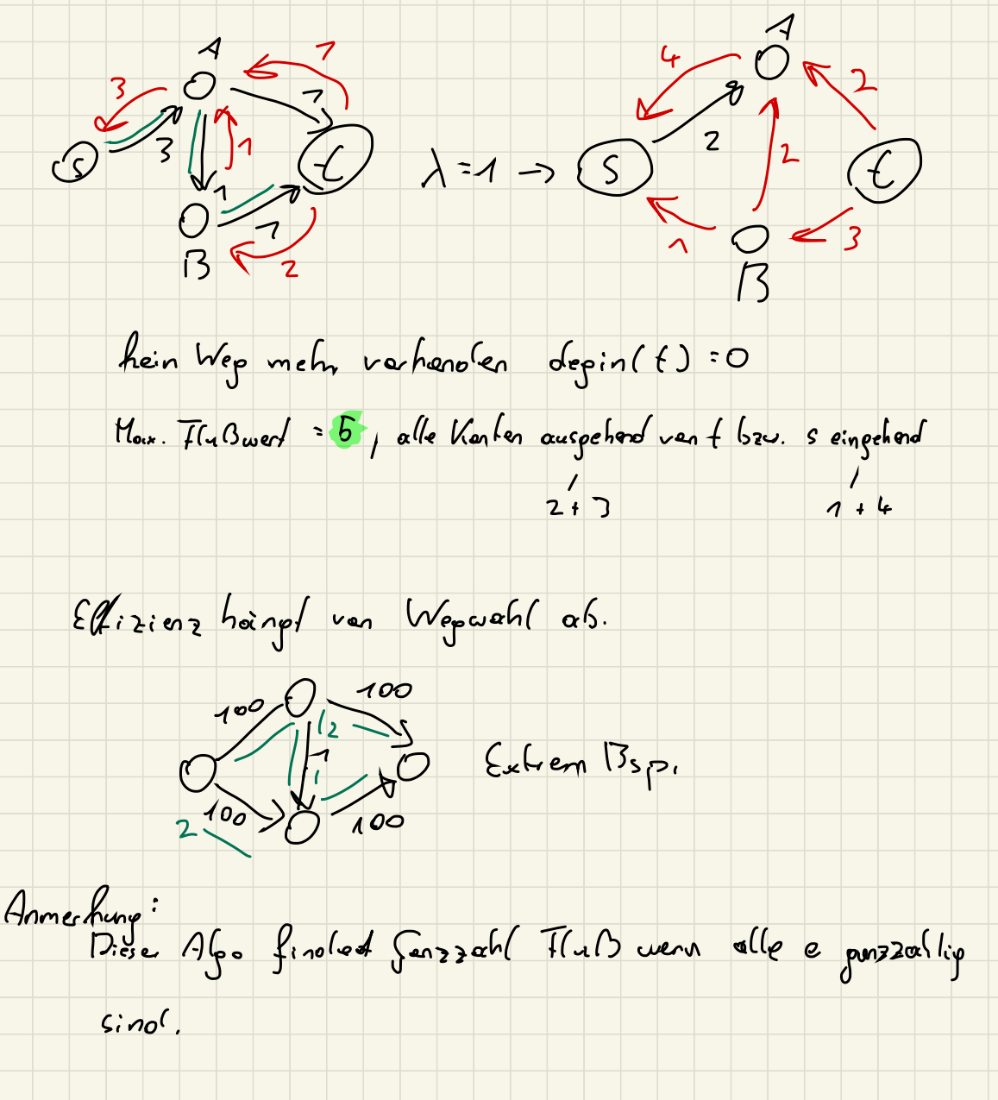
Ein Bild, das Text, drinnen, Screenshot enthält.

Automatisch generierte Beschreibung

1. **Zeichnen Sie die Residualgraphen nach jeder Iteration des Ford-Fulkerson-Algorithmus für ein beliebiges Netzwerk N für die ersten 3 Iterationen!**

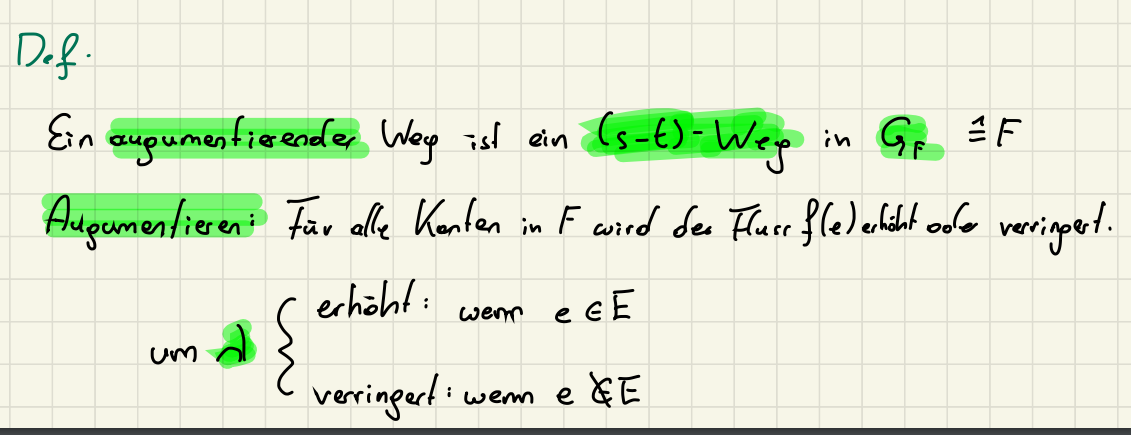
* Wir zeichnen immer gegen Pfad von der wir die entsprechenden Kosten abziehen
* Suche Weg zwischen S und T
* Zeichne Pfade gegenläufig ein
* Passe Gewicht an
  + ziehe kleinsten Wert Lambda überall ab
  + Kante kann wegfallen oder bleibt, wenn sie noch positive Kapazität hat
* Wiederhole solange bis keine Pfade mehr durch Graph von s nach t möglich





1. **Wie lässt sich ein augmentierender Pfad finden?**

* Eigentlich nur Lambda abziehen und Pfeil in Gegenrichtung zeichnen mit neuem Gewicht
* Angepasst wird originale Kante
* Finden
  + Neuer Weg zwischen Start- und Endknoten
  + Check, ob es noch einen Weg zwischen s und t im Residualgraph gibt (residual = neue Leitung Kapazitäten)
  + Kann man mit Shortest Path (Dijkstra/A\*) -> Bellman-Moore-Ford statt Dijkstra, wenn negative Distanzen machen



1. **Wann resultiert der Ford-Fulkerson-Algorithmus in einer ganzzahligen Lösung für maxflow?**

* Wenn alle Kapazitäten ganzzahlig sind, dann gibt es ganzzahligen Maximalen Fluss, wird von Fulkerson gefunden

1. **Was ist ein Schnitt und wie hängen maximaler Fluss und minimaler Schnitt zusammen?**

* Der (s-t)-Schnitt mit dem geringsten Wert entspricht dem maximalen  
  (s-t)-Fluss
* Ein Schnitt teilt einen Graph in zwei Zusammenhangskomponenten  
  Jene Kanten deren Knoten in unterschiedlichen
* Zusammenhangskomponenten bilden die "Schnitt-menge" (cut-set)
* Der Wert eines Schnitts ist die Summe der Kantengewichte der Schnittmenge
* Ein (s-t)-Schnitt ist ein Schnitt der den Graph so aufteilt, dass s in der einen und t in der anderen Zusammenhangskomponente liegt

52. Wie lautet die Standardform der linearen Programmierung?

Ein Bild, das Handschrift, Text, Kalligrafie, Schrift enthält.

Automatisch generierte Beschreibung

53. Aus welchen 5 Teilen ist ein lineares Programm aufgebaut?

Ein Bild, das Text, Handschrift, Kalligrafie, Schrift enthält.

Automatisch generierte Beschreibung

54. Welches Problem ergibt sich wenn in einem linearen Programm diskrete Entscheidungsvariablen definiert werden?

Wenn in einem linearen Programm diskrete Entscheidungsvariablen definiert werden, ergibt sich das Problem der sogenannten gemischt-ganzzahligen linearen Programmierung (Mixed Integer Linear Programming, MILP). In MILP-Modellen werden einige oder alle Entscheidungsvariablen auf diskrete Werte beschränkt, während andere Variablen weiterhin kontinuierlich sein können.

55. Wie werden Ungleichungen bzw. Gleichungen in einem linearen Programm abgebildet?

Ungleichungen:

Ungleichungen werden im linearen Programm typischerweise in der Form "linker Ausdruck ≤ rechter Ausdruck" oder "linker Ausdruck ≥ rechter Ausdruck" dargestellt. Der linke Ausdruck besteht aus den Entscheidungsvariablen, die multipliziert und addiert werden können, während der rechte Ausdruck eine Konstante oder eine lineare Funktion der Entscheidungsvariablen sein kann.

Beispiel für eine Ungleichung: 2x + 3y ≤ 10

Gleichungen:

Gleichungen werden im linearen Programm in der Form "linker Ausdruck = rechter Ausdruck" dargestellt. Der linke Ausdruck besteht aus den Entscheidungsvariablen, die multipliziert und addiert werden können, während der rechte Ausdruck eine Konstante oder eine lineare Funktion der Entscheidungsvariablen sein kann.

Beispiel für eine Gleichung: x + 2y = 5

Die Entscheidungsvariablen im linearen Programm repräsentieren die zu bestimmenden Größen oder Entscheidungen, die optimiert werden sollen. Sie können kontinuierlich oder diskret sein, je nach den Anforderungen des Problems.

Die Zielfunktion im linearen Programm wird in der Regel als lineare Funktion der Entscheidungsvariablen dargestellt, die entweder maximiert oder minimiert werden soll. Das Ziel besteht darin, die Werte der Entscheidungsvariablen zu finden, die die Zielfunktion optimieren, während gleichzeitig die Nebenbedingungen erfüllt werden

56. Was bedeutet es wenn künstliche Variablen in der optimalen Lösung eines linearen Programms einen Wert ungleich 0 annehmen?

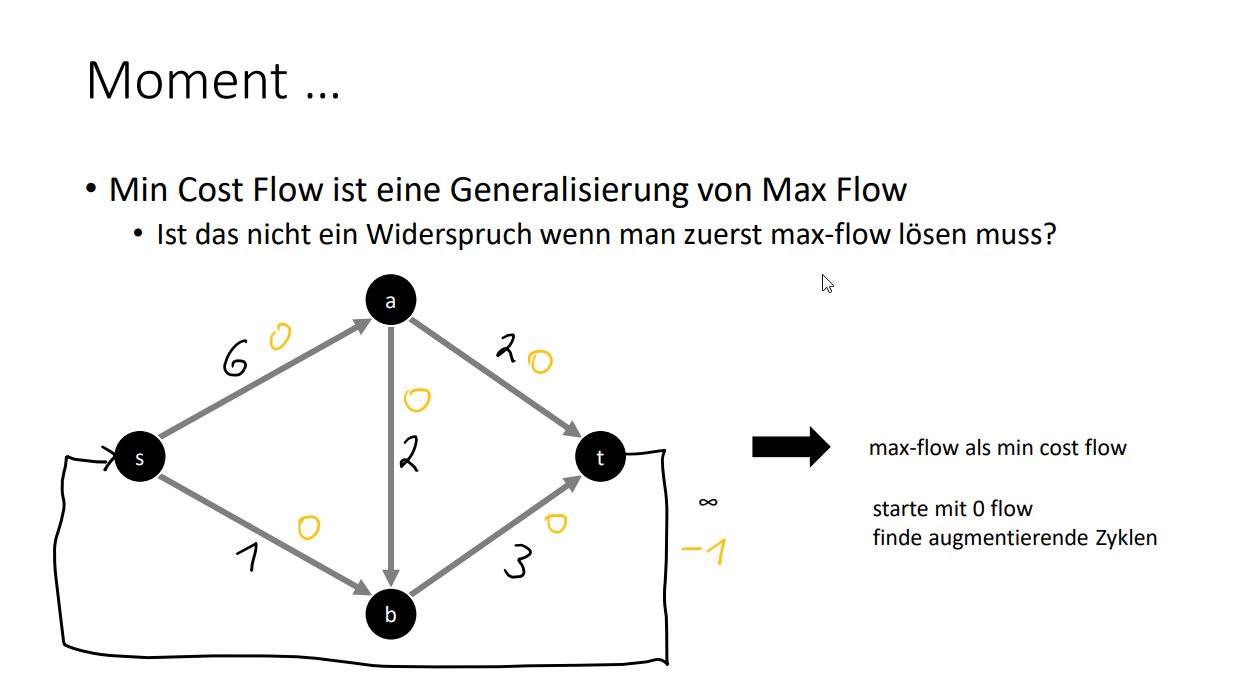
Wenn künstliche Variablen in der optimalen Lösung eines linearen Programms einen Wert ungleich 0 annehmen, bedeutet dies, dass das ursprüngliche lineare Programm keine zulässige Lösung hat. Künstliche Variablen werden eingeführt, um die Lösbarkeit des linearen Programms zu überprüfen und sicherzustellen, dass alle Nebenbedingungen erfüllt sind.

Im Simplex-Verfahren, einem verbreiteten Algorithmus zur Lösung linearer Programme, werden künstliche Variablen verwendet, um die Ungleichungen in Gleichungen umzuwandeln.

Idealerweise sollten die künstlichen Variablen in der optimalen Lösung des linearen Programms den Wert 0 haben. Das bedeutet, dass sie keinen Einfluss auf die Lösung haben und nur als Hilfsvariablen verwendet wurden, um das Problem zu formulieren.

1. **In welcher Beziehung stehen die Probleme maxflow und mincostflow?**

* Jedes Maxflow Problem kann in ein Mincostflow Problem überführt werden
* Um mincostflow zu berechnen, muss man zuerst den maxflow berechnen
* Weil Maxflow äquivalent zu Min-Cut ist
  + Max-Flow-Min-Cut-Theorem
* Der (s-t) Schnitt mit dem geringsten Wert entspricht dem Maximalen (s-t) Fluss
* Transformieren des Graphen



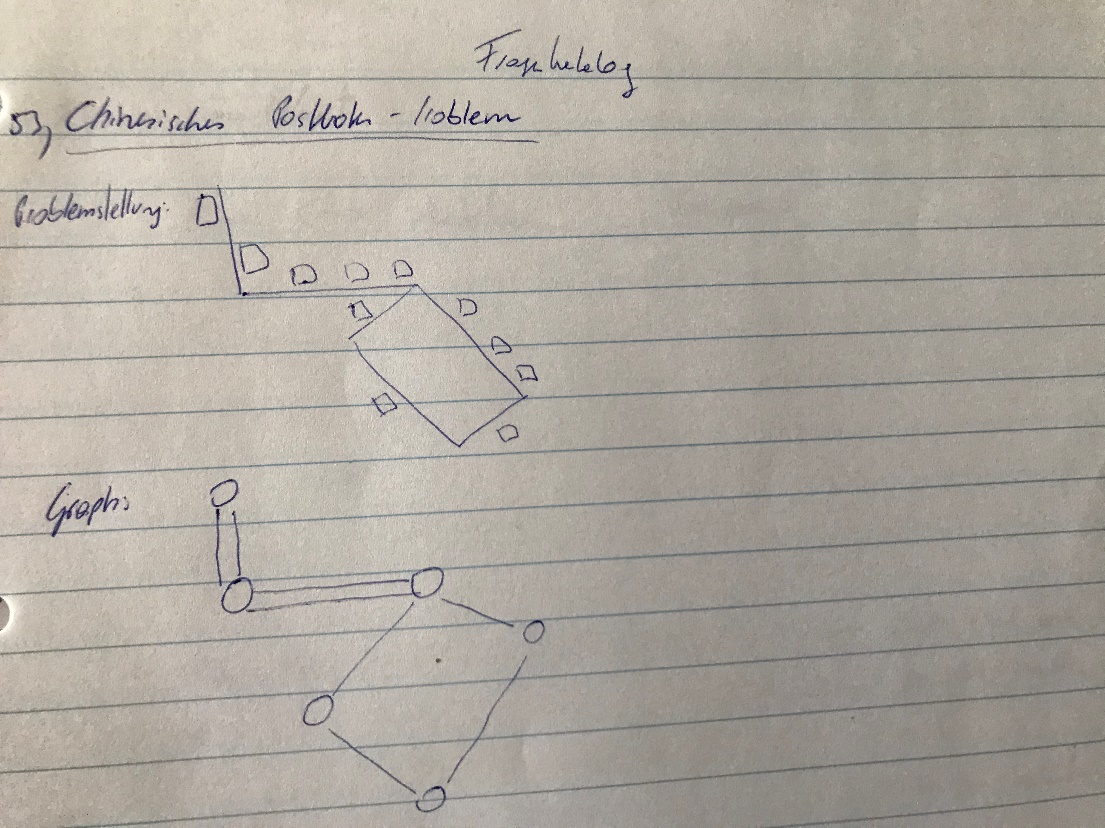
Behams Anmerkungen in der Übung zu den Textbeispielen:

* Genau beschreiben, wie man den Graphen / das Modell aufstellt (z.B. Knoten = Kreuzung, Straße = Kante)
* Evtl. kleine Skizze machen
* Einschränkungen / Erweiterungen anführen

1. **Eine chinesische Postbotin muss in einem Dorf Information zu einer bevorstehenden Wahl austragen, die bereits in wenigen Tagen stattfinden soll. In jeder Straße des Dorfes stehen Häuser, die sie besuchen muss. Sie ist besorgt, dass möglicherweise nicht alle die Information rechtzeitig erhalten. Wie kann man das Austragen der Information effizient gestalten?**  
   **Welches Problem verbirgt sich dahinter bzw. wie könnte man es erweitern?**

Lösung hier Eulerpfad; Erweiterung zum Eulerkreis (wenn man wieder zurück will).

Problem: Nicht jeder Graph enthält einen Eulerpfad à Graphen erweitern, sodass die Bedingungen für einen Eulerkreis erfüllt sind (nur 2 Knoten mit ungeradem Knotengrad)



1. **Eine Stadt möchte die Wasserversorgung in einem bestimmten Gebiet verbessern. Bewohner beschweren sich, dass an heißen Tagen zu wenig Wasser aus der Leitung kommt. Die Stadt stellt jedoch fest, dass genügend Wasser im Speicher vorhanden ist. Wie könnten Sie die Stadt bzw. die Bewohner hier unterstützen?**

Max-flow Problem (wieviel bekomme ich durch; sehe auch wo in meinem Netzwerk die Engstelle ist), z.B. mit Ford-Fulkerson-Algorithmus.

Speicher = Quelle, Häuser = Senke, Wasserleitung = Kante.

Evtl. zusätzliche Bedingungen: Zu jeder Senke muss eine Mindestmenge kommen bzw. es darf nur eine Maximalmenge zu jeder Senke kommen

1. **Das Management eines Einkaufszentrums stellt fest, dass trotz attraktiver Geschäfte in einem Bereich nur vergleichsweise wenige Kunden zu finden sind. Das Management möchte herausfinden warum und wie das zu verbessern wäre. Wie könnten Sie das Management unterstützen?**

Geschäfte als Knoten und Gänge als Kanten modellieren à Einkaufszentrum als Graph in dem es unterschiedliche Wege zwischen den Geschäften gibt.

Zentralitätsmaße (z.B. Zwischenzentralität liegt auf zu wenig kürzesten Wegen zwischen den Geschäften) à Geschäfte stärker bewerben oder weniger attraktive Marken in diesem Bereich platzieren;

man könnte benachbarte Geschäfte als Links betrachten, dann wäre es ein Page Rank Problem)

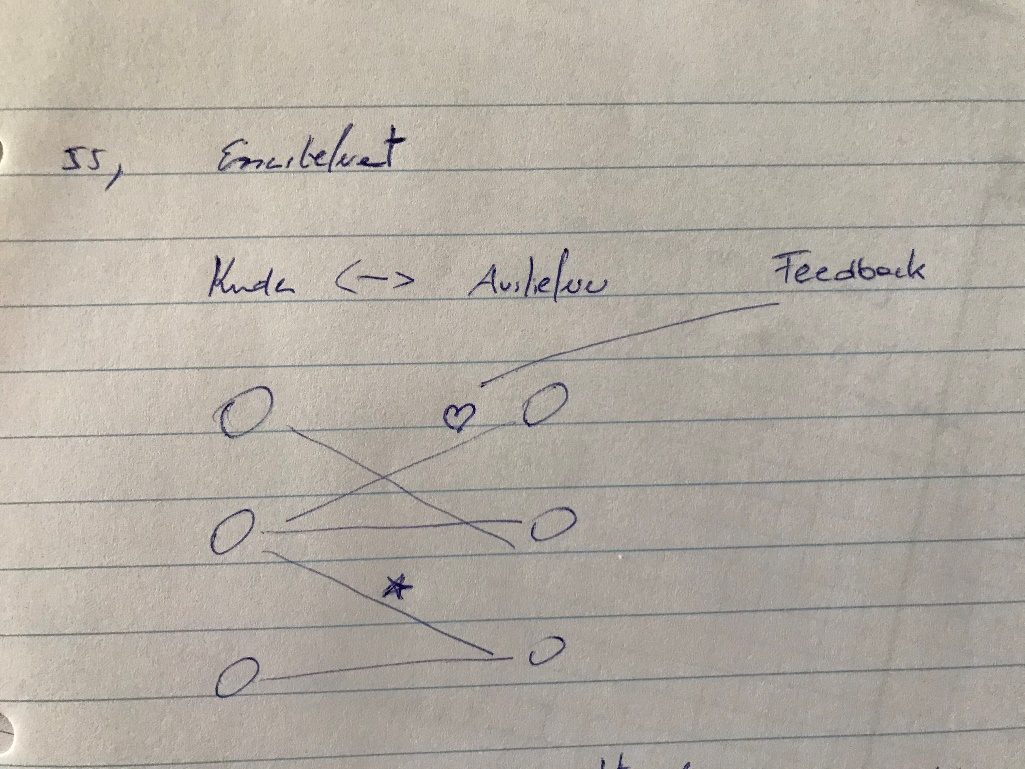
1. **Ein Essenslieferant beobachtet anhand des Kundenfeedbacks nach einer Lieferung, dass bestimmte Auslieferer besser mit bestimmten Kunden zurechtkommen als mit anderen. Der Lieferant möchte die Kundenzufriedenheit steigern. Wie könnten Sie den Lieferanten hier**  
   **unterstützen?**

Assignment Problem:

* Bipartite Paarung à Welcher Kunde soll von welchem Auslieferer beliefert werden?
* Zwei Knotenmengen: 1) Auslieferer, 2) Kunden
* Bewertungen als Kantengewichte darstellen

Mögliche Einschränkungen:

* Ein Auslieferer kann nicht alle Kunden beliefern, falls er z.B. besonders beliebt ist
* Arbeitslast soll möglichst ausgeglichen sein.
* Jeder Auslieferer soll ausliefern



Vorschlag Werth: Als Mincost – Maxflow modellieren:

* Allgemein ist es ein Assignment Problem: Etwas der Menge 1 der Menge 2 zuweisen. Das kann als Maxflow Problem modelliert werden. (wie oben schon erwähnt, hier nur noch eine Ergänzung)

Kundenzufriedenheit sagt mir, wie viel ich über die Kante liefern kann und das maximiere ich dann.

* Maxflow: Kapazität überall 1 auf allen Kanten. Die Kundenzufriedenheit dann als negative Kosten daraufmodellieren.
* Rechts und links limitieren: jeder Kunde genau einmal und jeder Lieferant einmal. Dazwischen den Zusammenhang eben mit den negativen Kosten für Kundenzufriedenheit modellieren und mit min cost max flow arbeiten, sodass die Kosten gering sind und somit die Kundenzufriedenheit hoch ist.

Nachstehende Fragen nicht im offiziellen Fragenkatalog. Habe nur bei der Diskussion mit Werth mitgeschrieben:

61. Die Bestimmung von Lehrveranstaltungszeiten, sowie die Raumbelegung sind an einer Universität für die administrativen Kräfte große Herausforderungen und nehmen jedes Semester etliche Stunden in Anspruch. Versuchen Sie das Problem so weit als möglich als lineares Programm zu skizzieren. Beschreiben Sie Mengen, Parameter, Entscheidungsvariablen, Ziele und mögliche Nebenbedingungen, möglichst in mathematischer Form. Ein paar Anhaltspunkte: (1) ein Raum kann zu einem Zeitpunkt nur eine LVA abhalten, (2) Personen können sich zur selben Zeit in nur einem Raum befinden, (3) der Raum muss genügend Kapazität haben damit alle Platz finden, (4) der Raum muss die Anforderungen an die LVA berücksichtigen (PC Plätze, Beamer, Tafel, Whiteboard, onlinefähig). Was könnten mögliche Ziele für so ein Modell sein?

Good to know:

1. **Breitensuche vs. Tiefensuche**

Alle Suchen funktionieren mit Queue. Bei Breitensuche muss ich bei .. was hinzufügen; bei Tiefensuche bei …

Bei Tiefensuche nehme ich die Nachbarn vom jetzigen Knoten und haue sie vorne in die Queue rein (die arbeite ich als nächstes ab)

Bei Breitensuche wird “hinten” in der Queue eingefügt

Wenn nach Kantengewicht sortiert = Prim

Nach Kantengewicht und Distanz zur Source sortiert = Dijkstra

Nach Kantengewicht und Distanz zur Source und Heuristic = A\*

1. **Stark zusammenhängende Komponenten**

Von jedem Knoten erreiche ich jeden anderen Knoten (in dieser Komponente). Macht nur bei gerichteten Graphen Sinn. Bei ungerichtet heißt es nur zusammenhängend. Man darf keine Senke haben.

1. **Ordnung vs. Größe eines Graphen**

* Ordnung = Anzahl an Knoten
* Größe = Anzahl an Kanten