Technische Hochschule Rosenheim

Grundlagen der Informatik

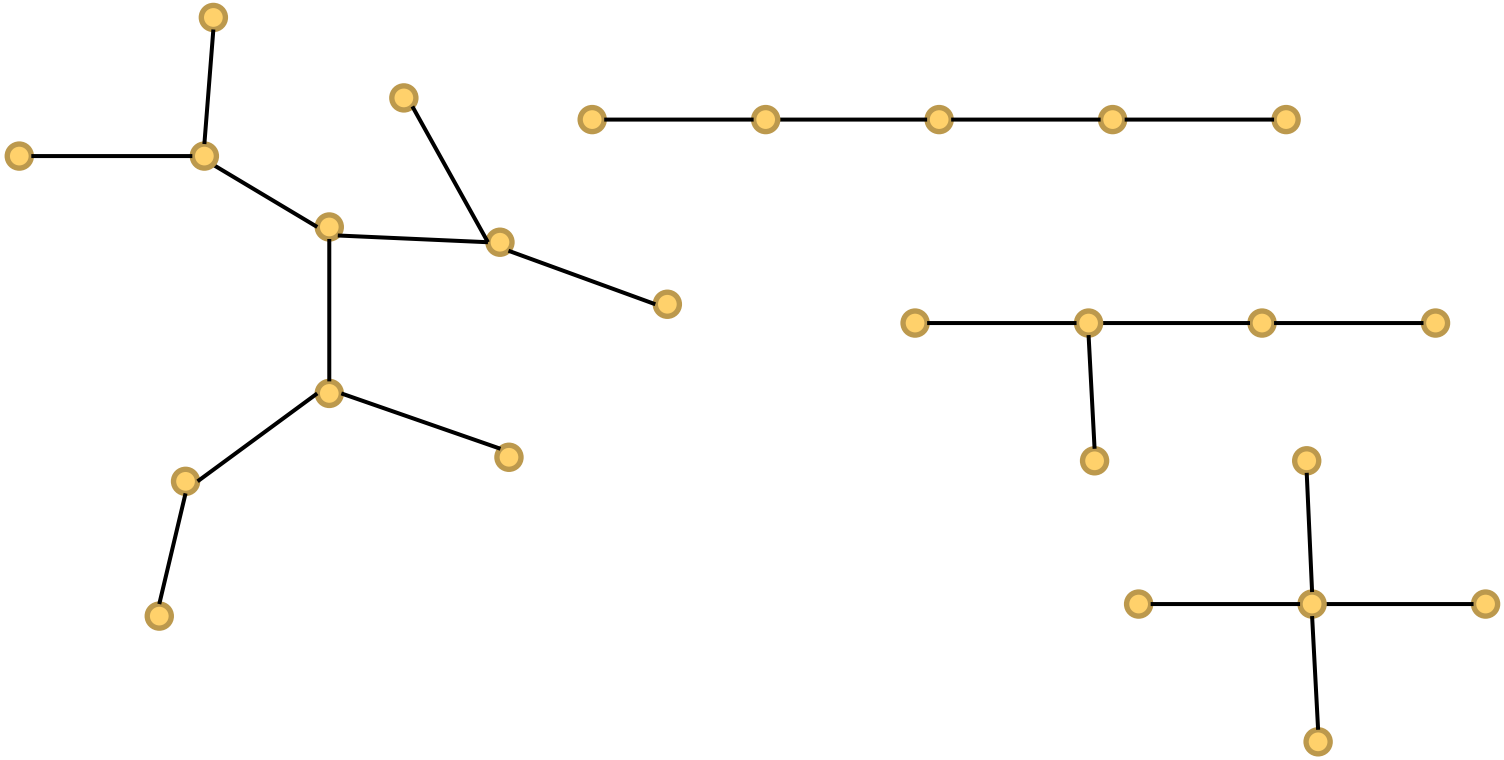
Prof. Dr. J. Schmidt
Fakultät für Informatik

GDI – WS 2020/21

Graphentheorie – Bäume

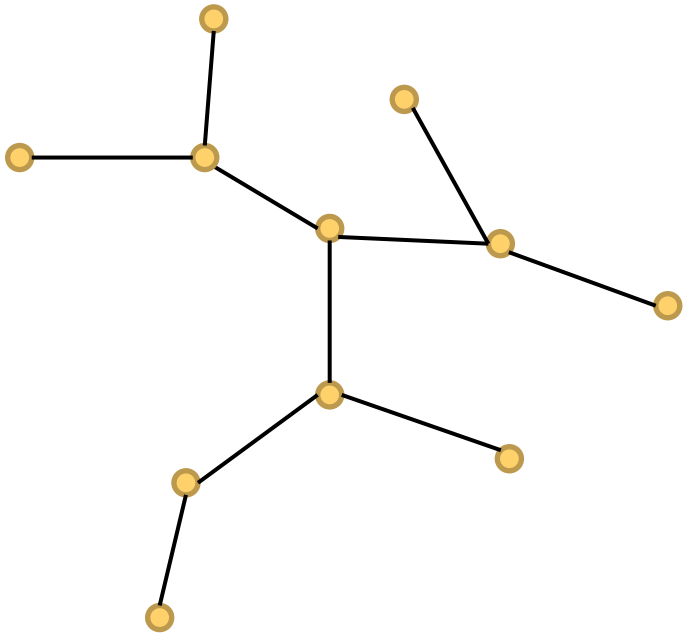
Wald

Ein (schlichter, ungerichteter) Graph G heißt Wald (forest) genau dann wenn G keinen Kreis enthält



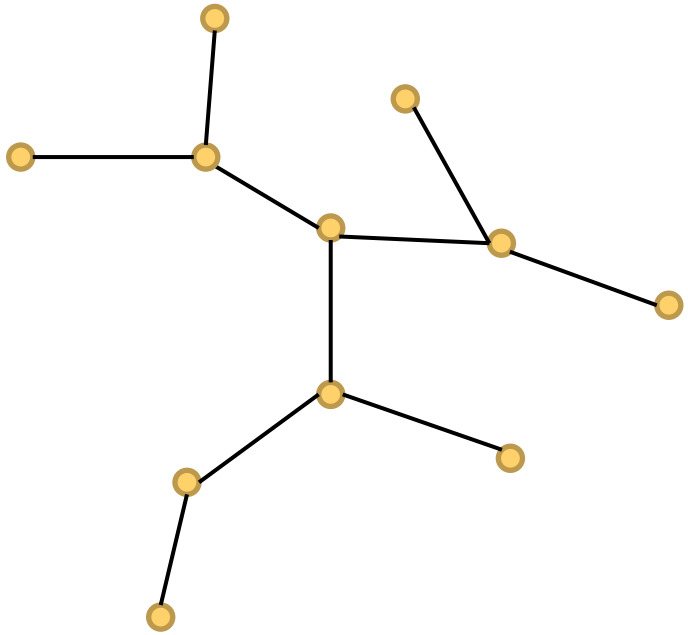
Baum

G heißt Baum (tree) genau dann wenn G ein Wald und zusammenhängend ist



Eigenschaften von Bäumen

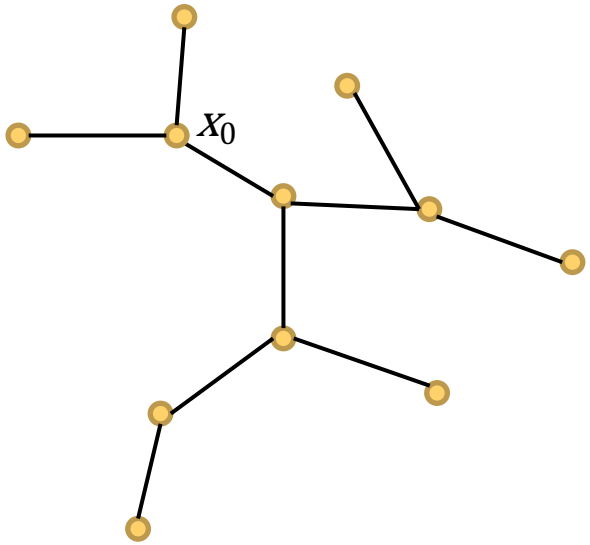
- je zwei Knoten sind durch genau einen Weg verbunden
- entfernt man eine Kante, so ist der Graph nicht mehr zusammenhängend
- ein Baum mit n Knoten hat genau $n - 1$ Kanten

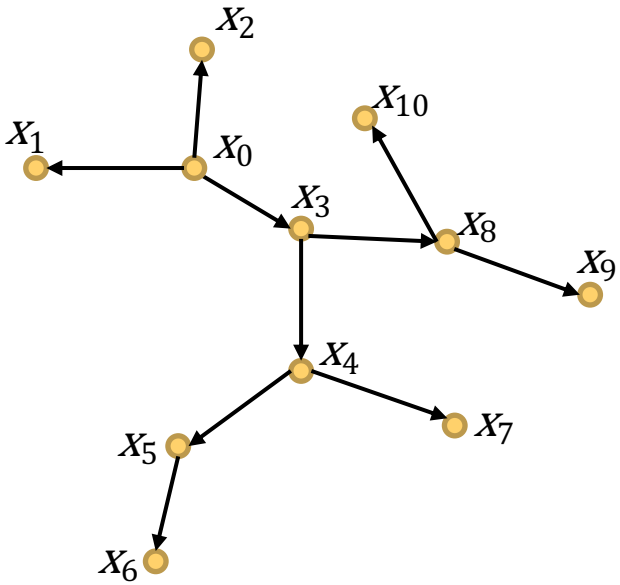


Wurzelbäume

Ein Wurzelbaum entsteht, wenn man in einem Baum

- einen Knoten x_0 als Wurzel (root) auszeichnet
- jede ungerichtete Kante $[x, y]$ durch eine gerichtete (x, y) ersetzt, wobei gilt:
- $d(x_0, x) < d(x_0, y)$



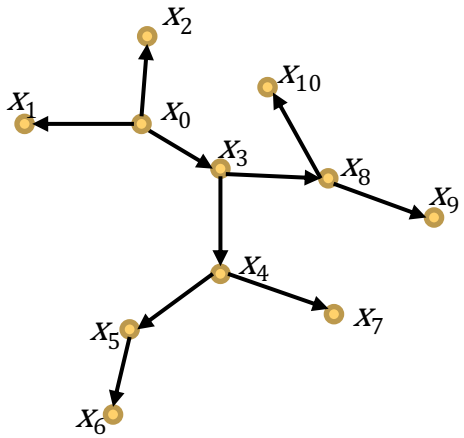


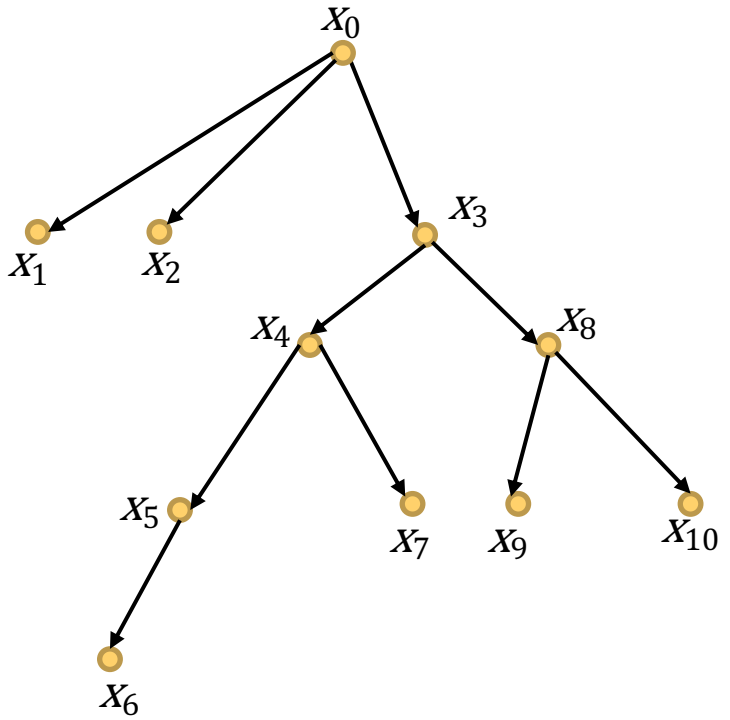
Wurzelbäume

- d(x_0, x) heißt Niveau des Knotens x
- man zeichnet alle Knoten gleichen Niveaus auf gleicher Höhe
  - dadurch wächst der Baum von oben nach unten
- es gilt:

  $d^-(x_0) = 0$

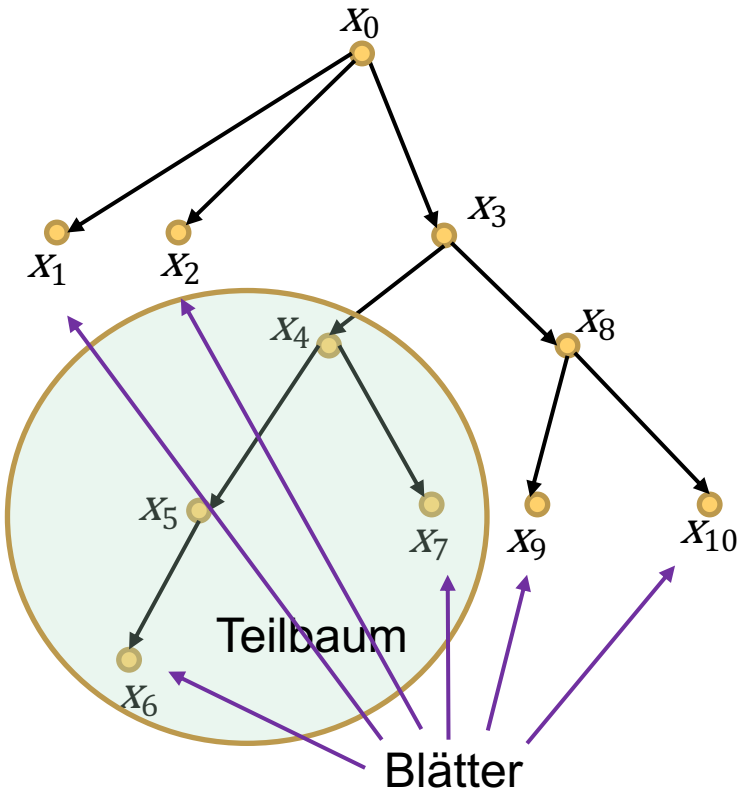
  $d^-(x_i) = 1 \; \forall \; i \neq 0$





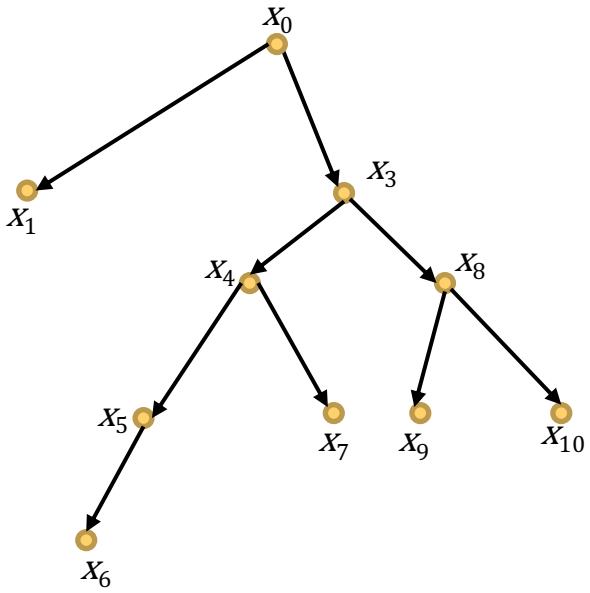
Wurzelbäume

- Teilbaum: Menge der von einem Knoten aus erreichbaren Wege
- Blätter (Endknoten): Knoten mit $d^+(x_i) = 0$



Spezielle Bäume

- Binärbaum: es gilt $d^+(x_i) \leq 2$ für alle Knoten



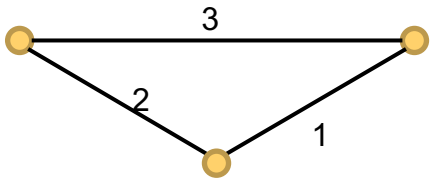
- Liste: es gilt $d^+(x_i) \leq 1$ für alle Knoten

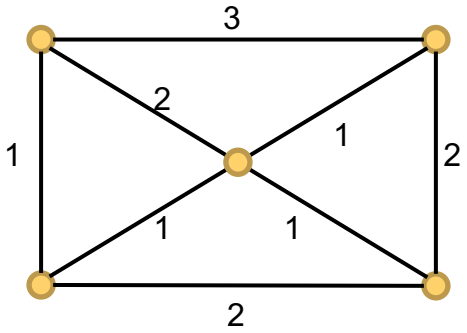
Gerüst

- Gerüst W (aufspannender Wald, spanning forest) eines Graphen G
  - W ist ein Wald
  - die Knotenmengen in W und G stimmen überein
- W entsteht also durch weglassen von Kanten
  - Zusammenhangskomponenten bleiben intakt
  - es treten keine Kreise mehr auf
- Minimalgerüst
  - in gewichteten Graphen
  - Summe der Kantengewichte soll minimal sein

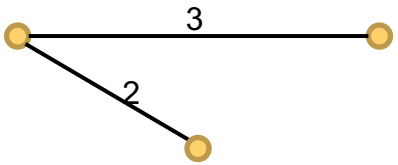
Gerüst – Beispiel

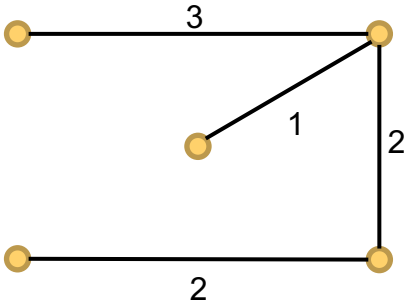
G:

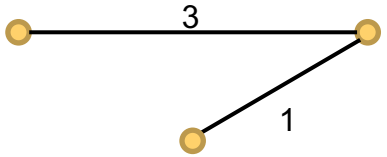


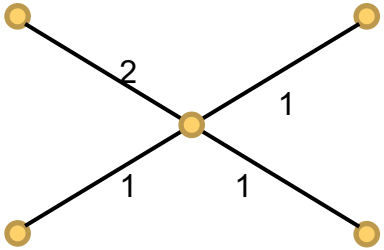


Gerüste z.B.:

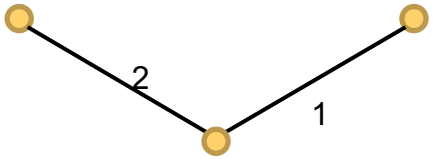


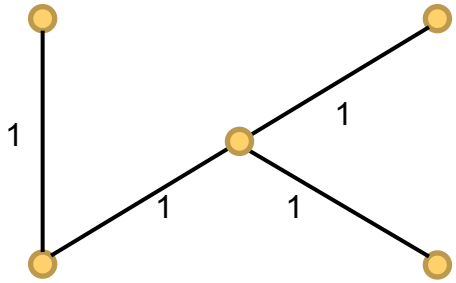






Minimalgerüst:

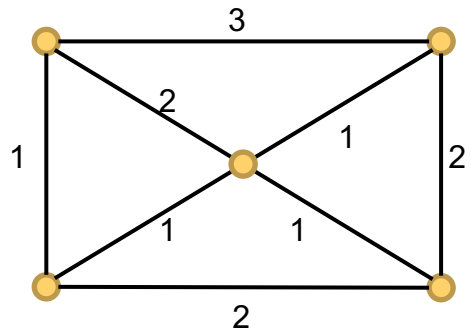




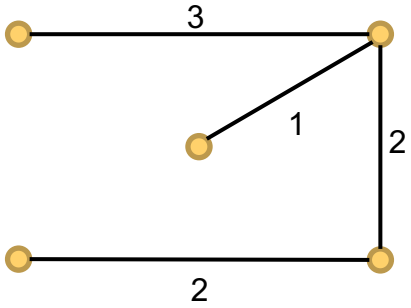
Spannbaum

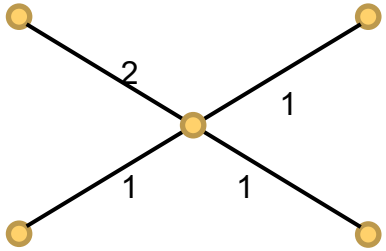
- Spannbaum (spanning tree)
  - Gerüst eines zusammenhängenden Graphen
- Minimaler Spannbaum
  - Minimalgerüst in zusammenhängenden Graphen
- Anwendungsbeispiel
  - Erstellung von kostengünstigen zusammenhängenden Netzwerken (Telefon, Strom, Computer)
- Algorithmen
  - Spannbaum: Tiefen-/Breitensuche
  - minimaler Spannbaum: Kruskal, Prim → A&D

Spannbaum – Beispiel

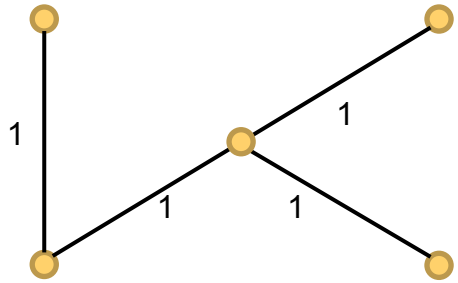


Spannbäume z.B.:

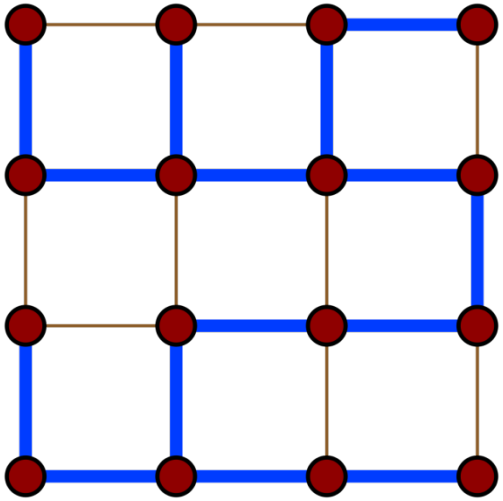


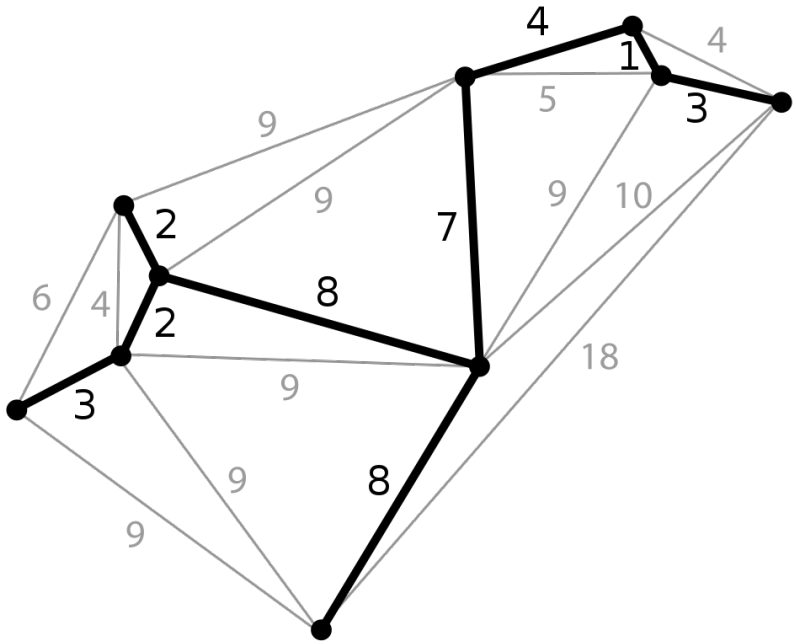


Minimaler Spannbaum:



Spannbaum – Beispiele

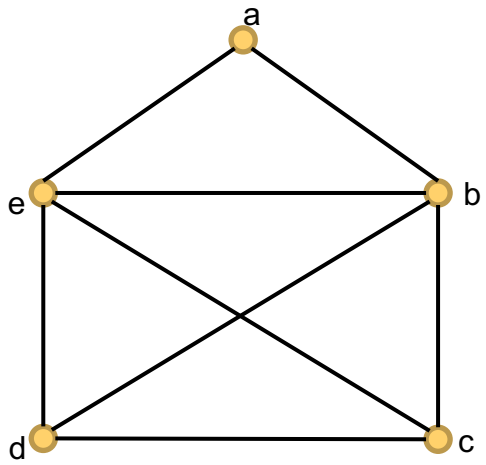




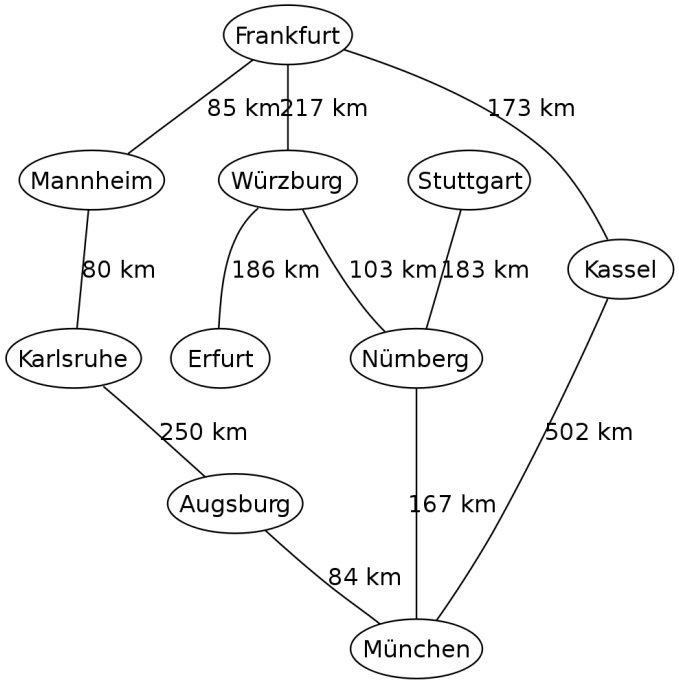
Quelle: Wikipedia, Public Domain

Aufgabe

Bestimmen Sie 3 nicht-isomorphe
Spannbäume des folgenden Graphen:



Suchen in Graphen – Motivation



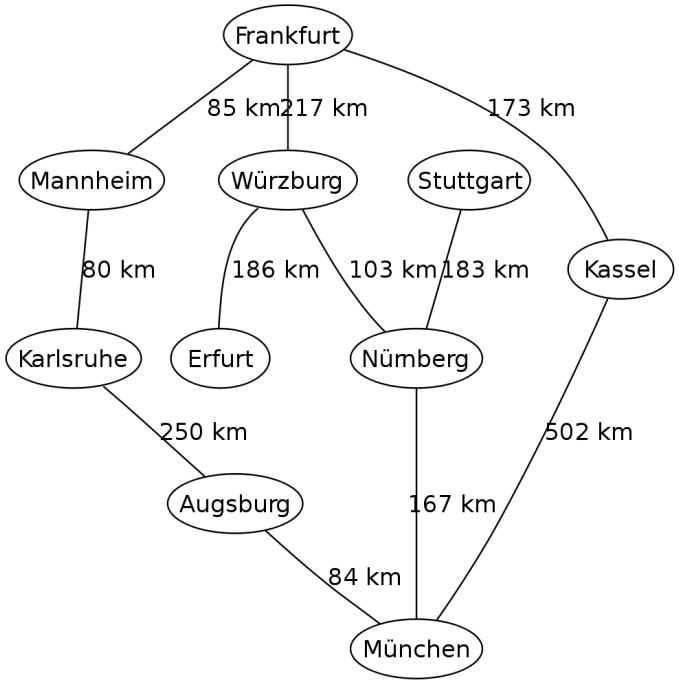
Quelle: Wikipedia, Public Domain

Mögliche Fragestellungen:

- wie komme ich von Frankfurt nach München?
- was ist die kürzeste Verbindung von Frankfurt nach München?

Suchstrategie: Festgelegt durch die Reihenfolge der Knotenexpansion

Suchen in Graphen – allgemein



function TREE-SEARCH(*Problem, Strategie*) **return** Lösung oder fehlgeschlagen
    Initialisiere Suchbaum mit dem *Anfangszustand* des *Problems*
    do
        if keine Kandidaten zur Expansion **then return** *fehlgeschlagen*
        Wähle Blatt zur Expansion basierend auf *Strategie*
        if Knoten enthält den Zielzustand **then return** *Lösung*
        else expandiere Knoten und füge resultierende Knoten dem Suchbaum hinzu
    enddo

Bewertungskriterien für Suchalgorithmen

- Kriterien:
  - Vollständigkeit: *Wird immer eine Lösung gefunden, sofern diese existiert?*
  - Optimalität: *Wird immer die Lösung mit den geringsten Kosten gefunden?*
  - Zeitkomplexität: *Anzahl der generierten/expandierten Knoten*
  - Speicherkomplexität: *Anzahl der während der Suche gespeicherten Knoten*

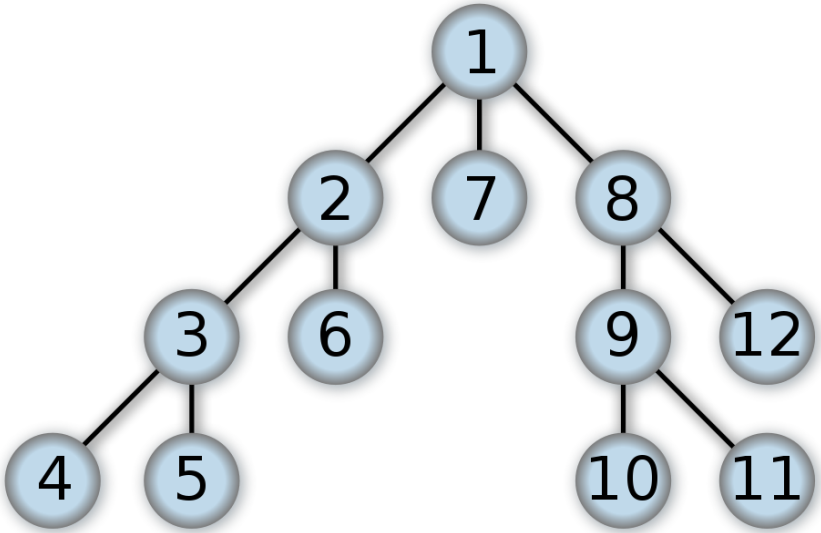
- Zeit- und Speicherkomplexität werden basierend auf der Schwierigkeit des Problems gemessen, definiert durch:
  - *b – maximaler Verzweigungsgrad des Suchbaums*
  - *d – Tiefe der Lösung mit geringsten Kosten*
  - *m – maximale Tiefe des Zustandsraums (kann ∞ sein)*

Uninformierte Suchstrategien

- uninformierte Suche = blinde Suche
  - verwende nur Information, die in der Problemdefinition verfügbar ist
  - wenn Strategie feststellen kann, ob ein nicht-Zielknoten besser ist als ein anderer $\rightarrow$ informierte Suche
- je nach Expansionsalgorithmus:
  - Tiefensuche
  - Breitensuche
  - uniforme Kosten Suche

Tiefensuche (Depth-First Search)

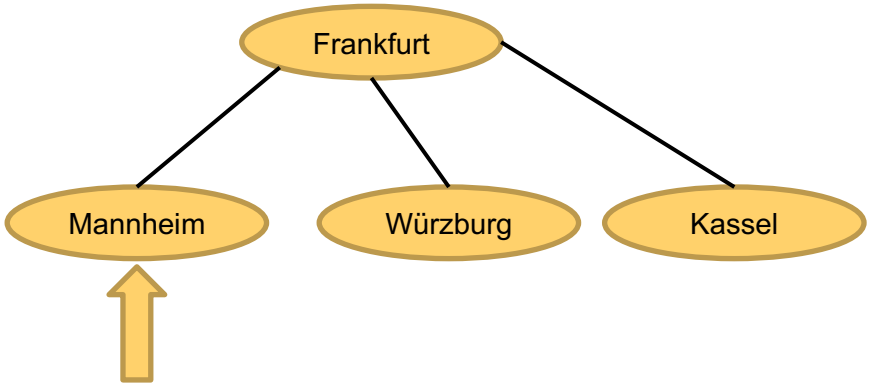
- Strategie
  - expandiere das tiefste Blatt
- Datenstruktur für Blätter: Stack (LIFO)

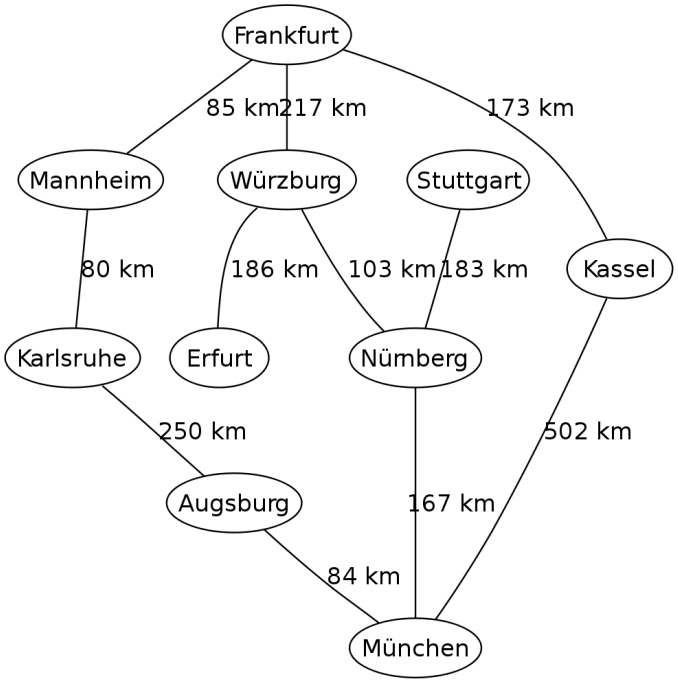


Quelle: Wikipedia, A. Drichel

Tiefensuche – Beispiel

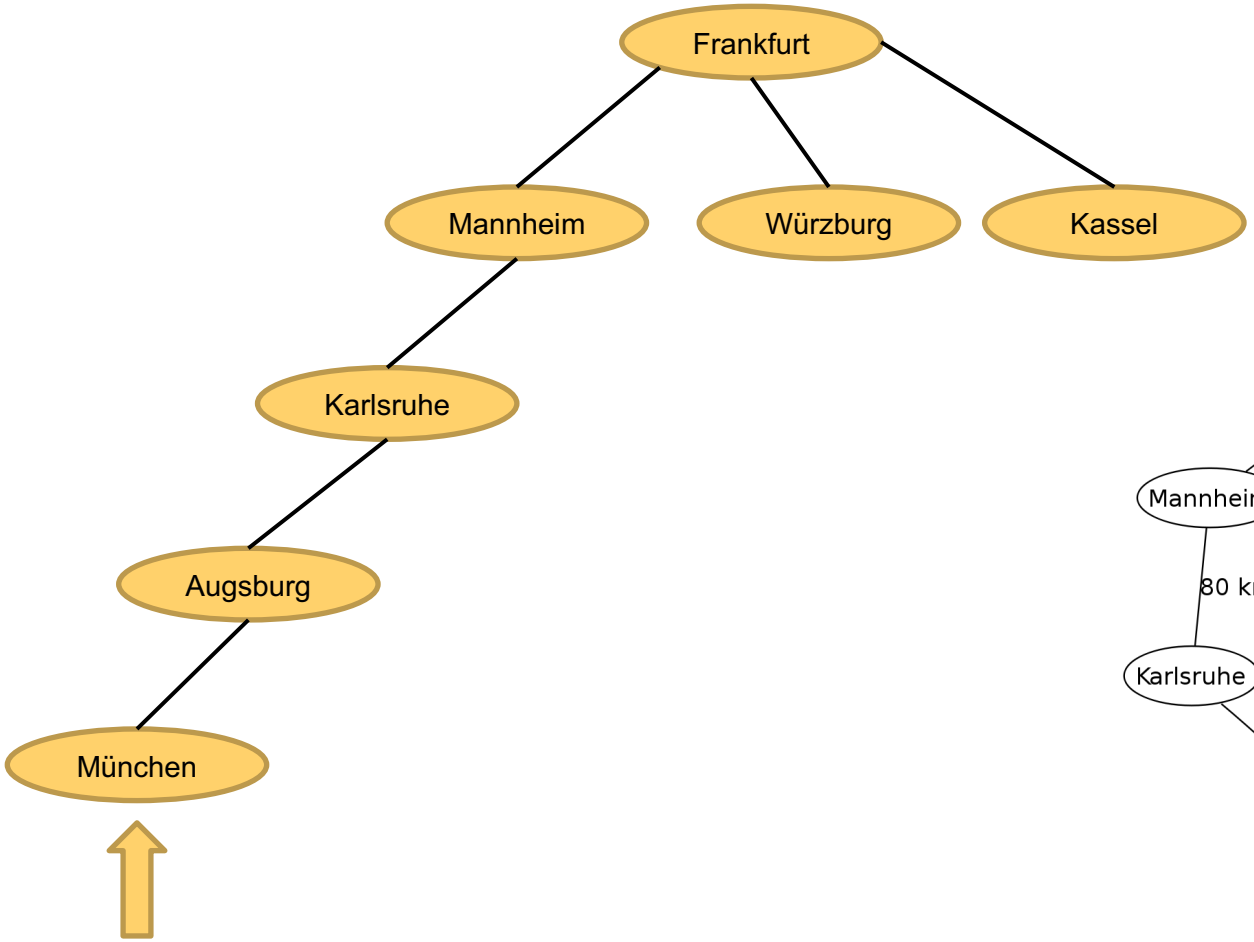
wie komme ich von Frankfurt nach München?



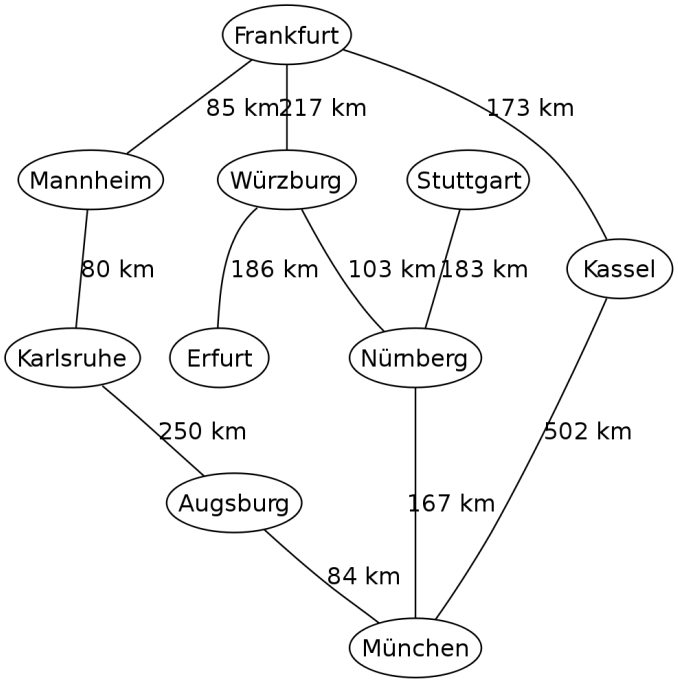


Tiefensuche – Beispiel

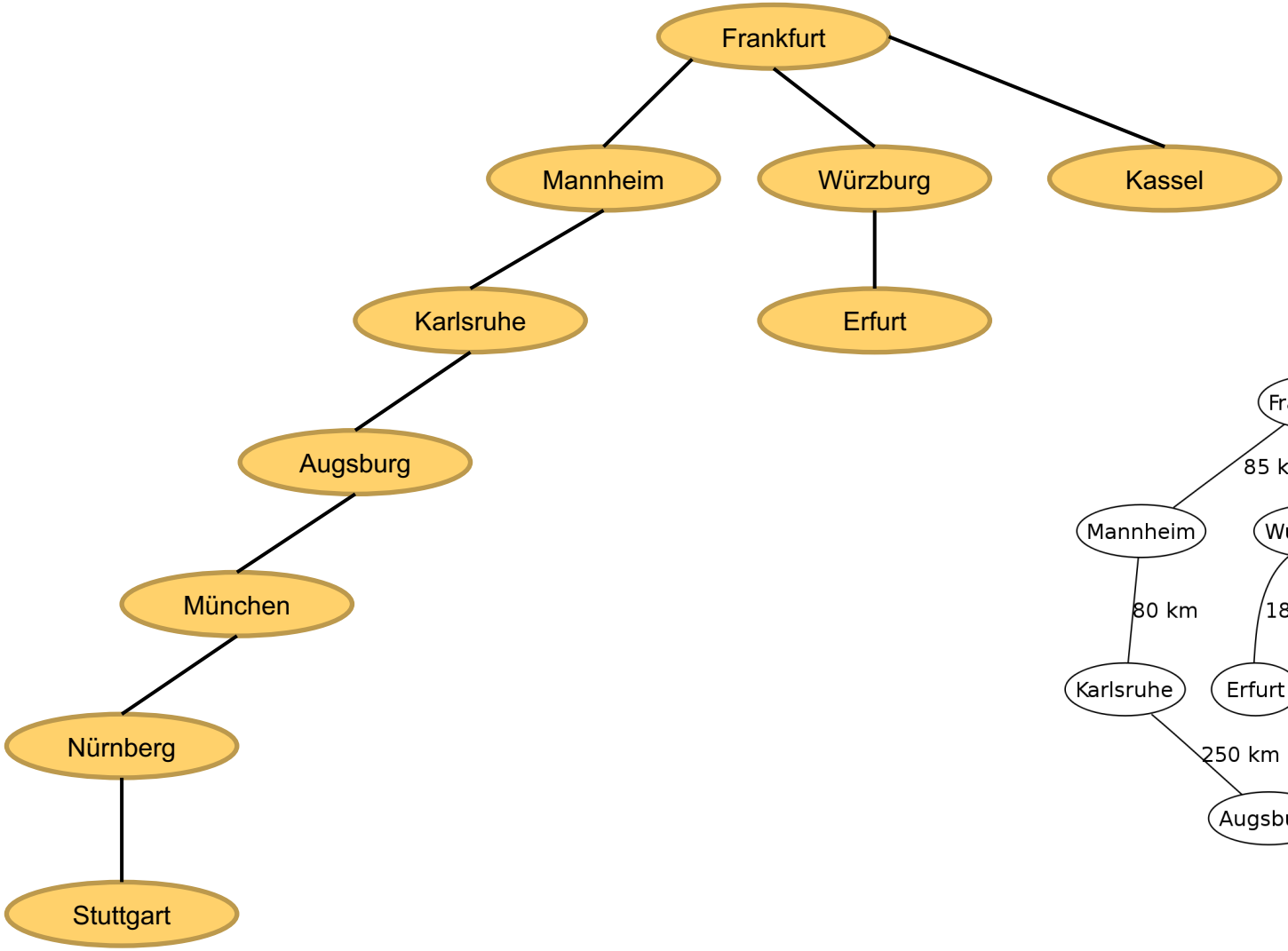
wie komme ich von Frankfurt nach München?

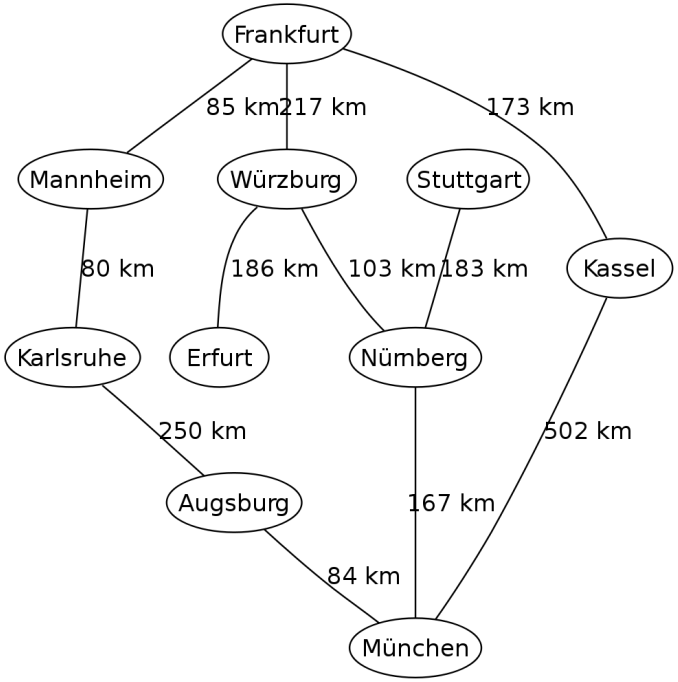


ist das die kürzeste Route?



Tiefensuche – Beispiel Spannbaum



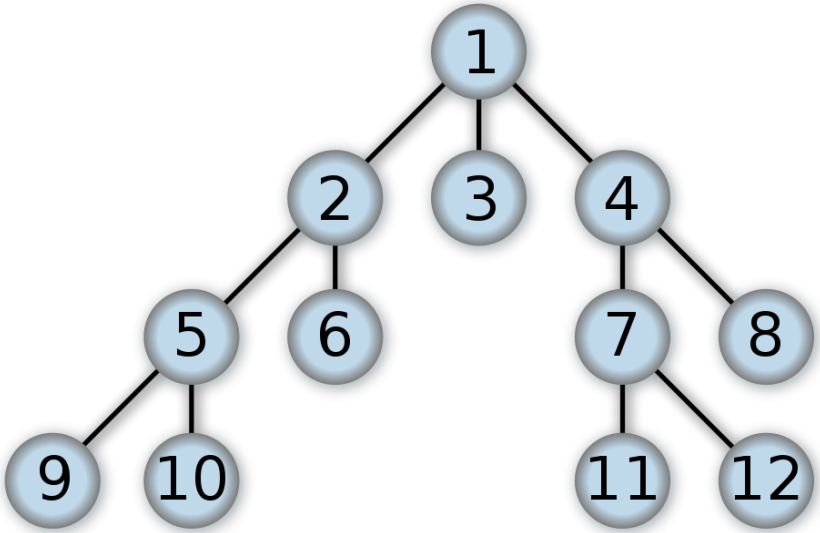


Bewertung Tiefensuche

- Vollständigkeit:
  - NEIN
  - außer der Suchraum ist endlich und es gibt keine Schleifen
- Optimalität:
  - NEIN
- Zeitkomplexität: $O(b^m)$
  - sehr schlecht wenn m (maximale Tiefe des Suchraums) sehr viel größer ist als d (Tiefe der optimalen Lösung)
  - aber: schneller als Breitensuche, falls es mehrere Lösungen gibt
- Speicherkomplexität: $O(bm + 1)$

Breitensuche (Breadth-First Search)

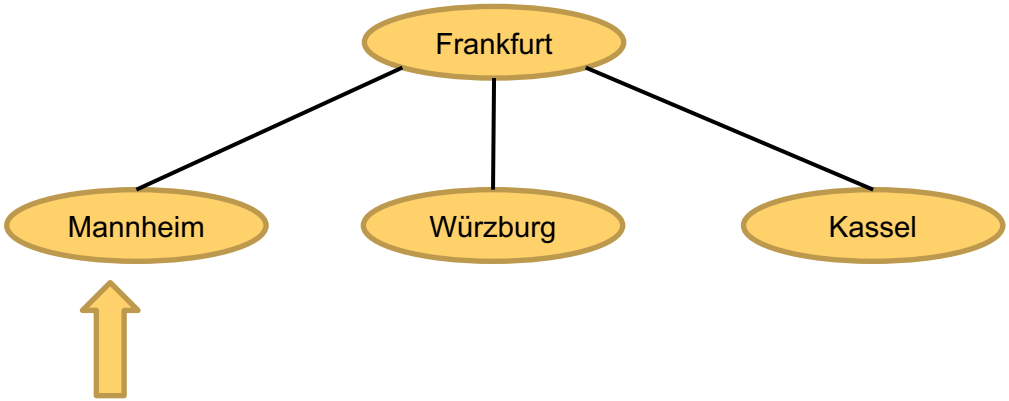
- Strategie
  - expandiere das flachste Blatt
- Datenstruktur für Blätter: Warteliste (Queue, FIFO)

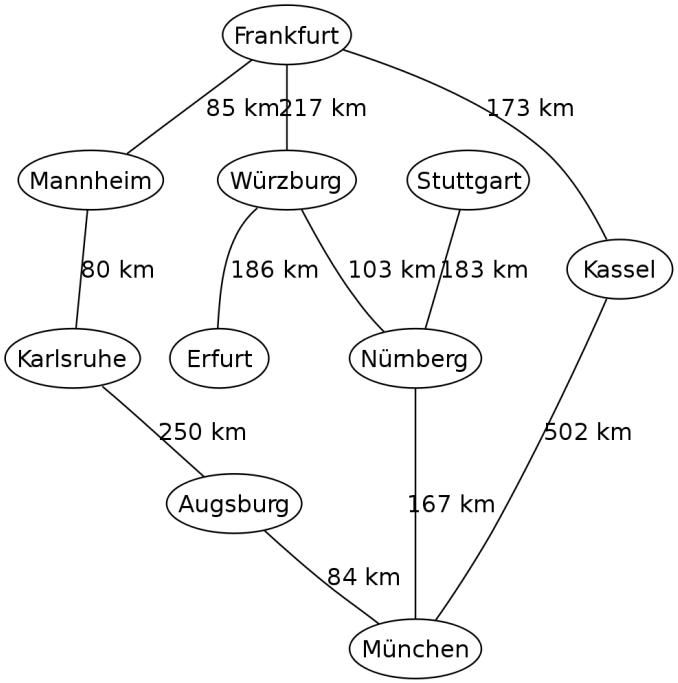


Quelle: Wikipedia, A. Drichel

Breitensuche – Beispiel

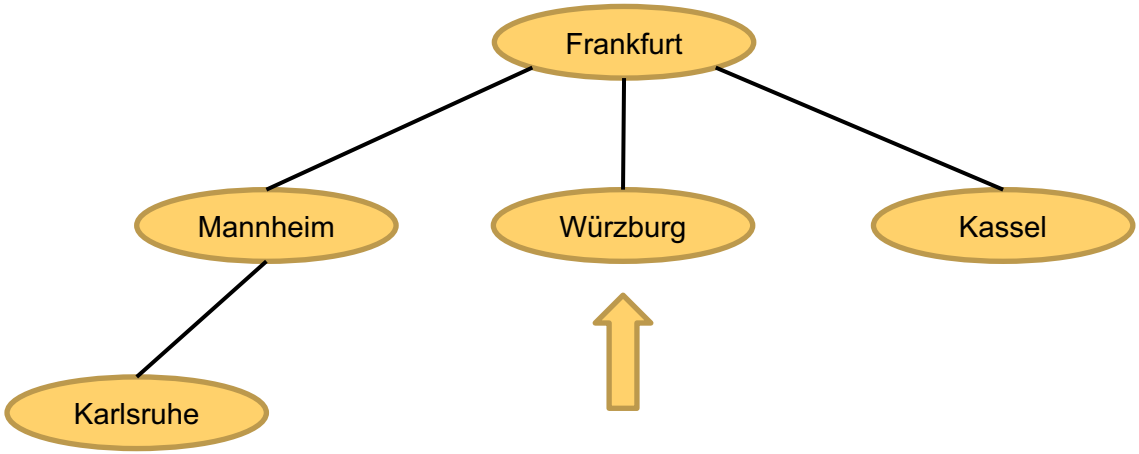
wie komme ich von Frankfurt nach München?

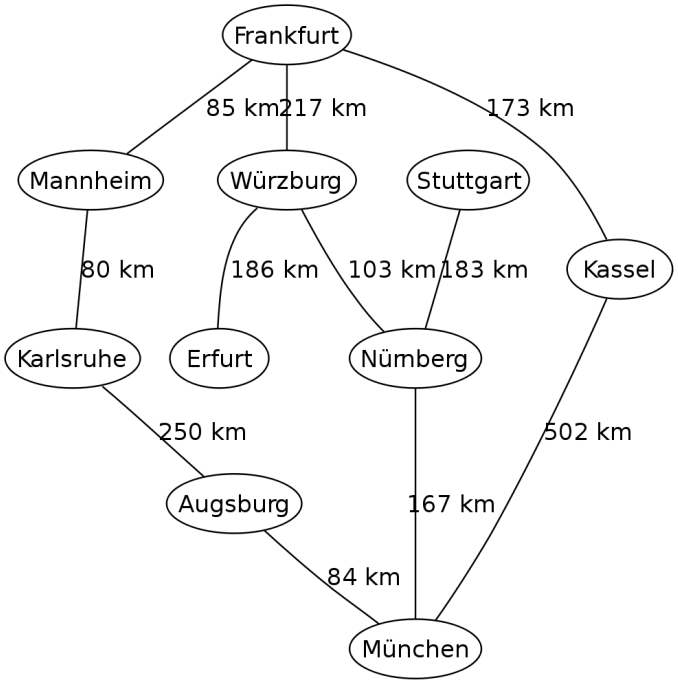




Breitensuche – Beispiel

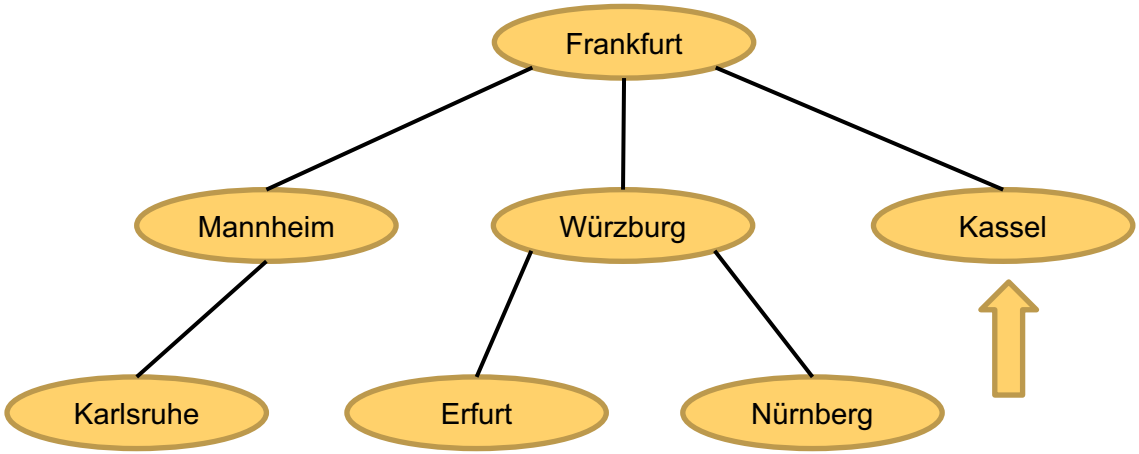
wie komme ich von Frankfurt nach München?

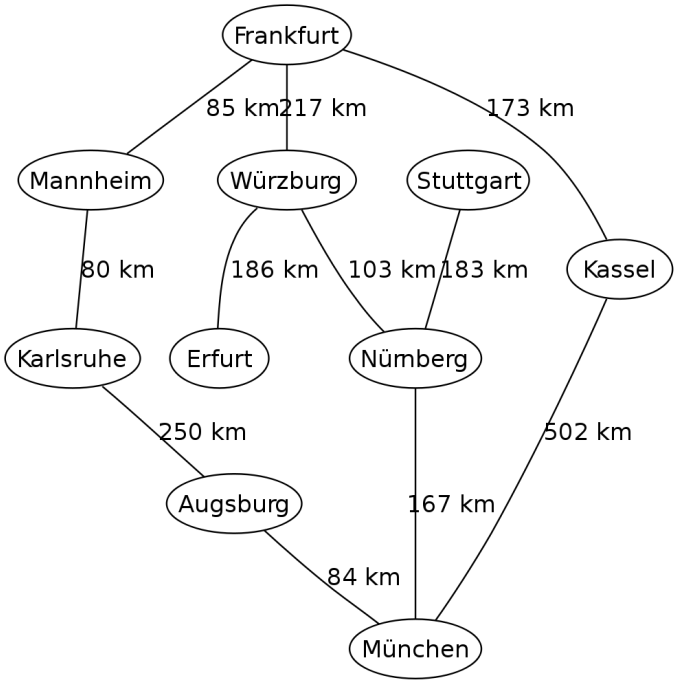




Breitensuche – Beispiel

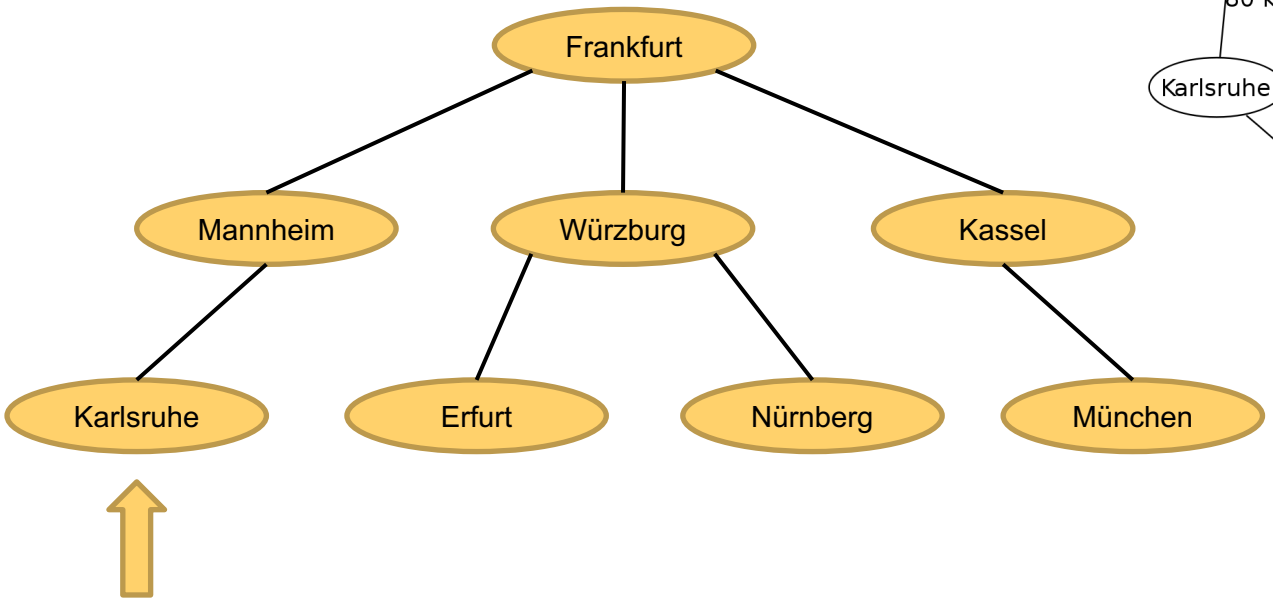
wie komme ich von Frankfurt nach München?

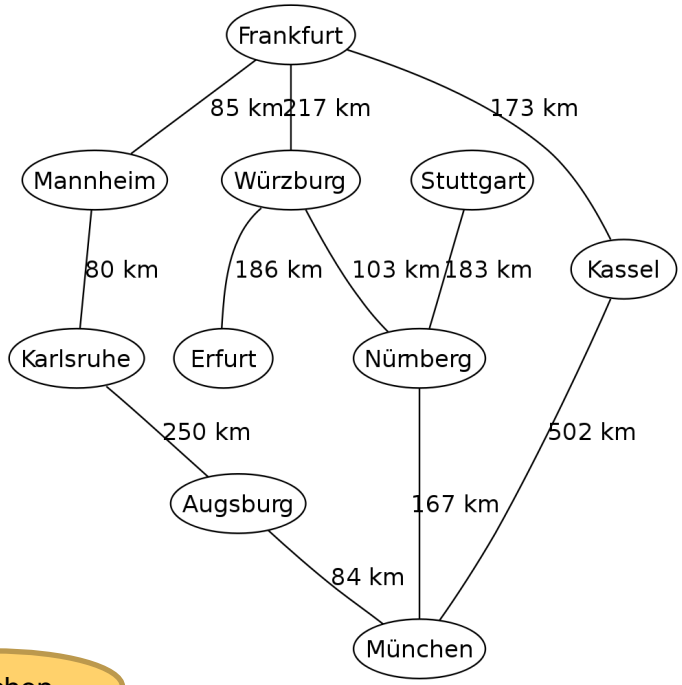




Breitensuche – Beispiel

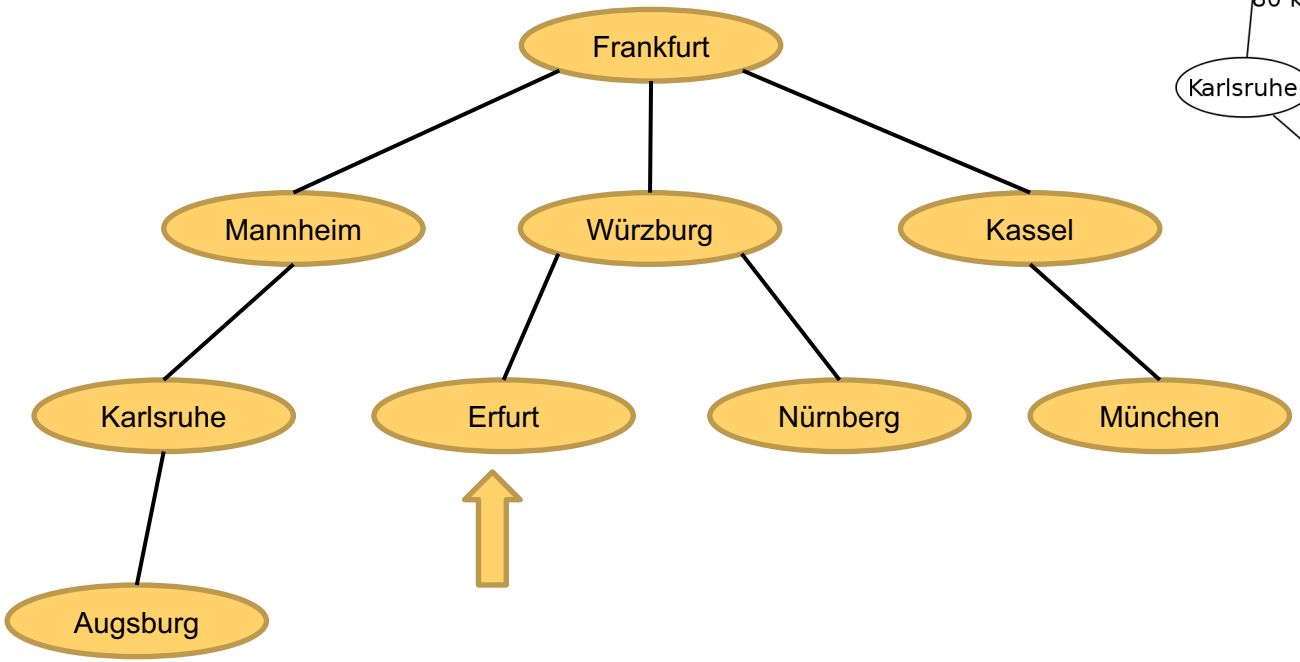
wie komme ich von Frankfurt nach München?

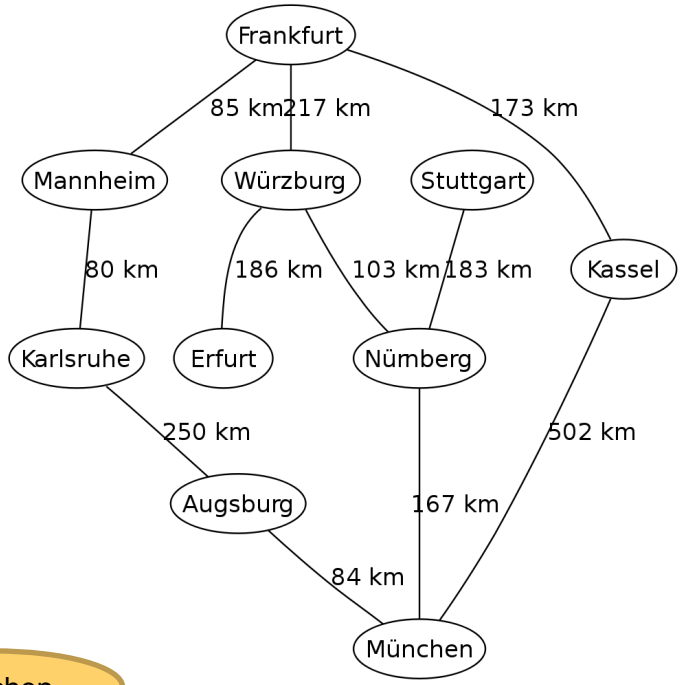




Breitensuche – Beispiel

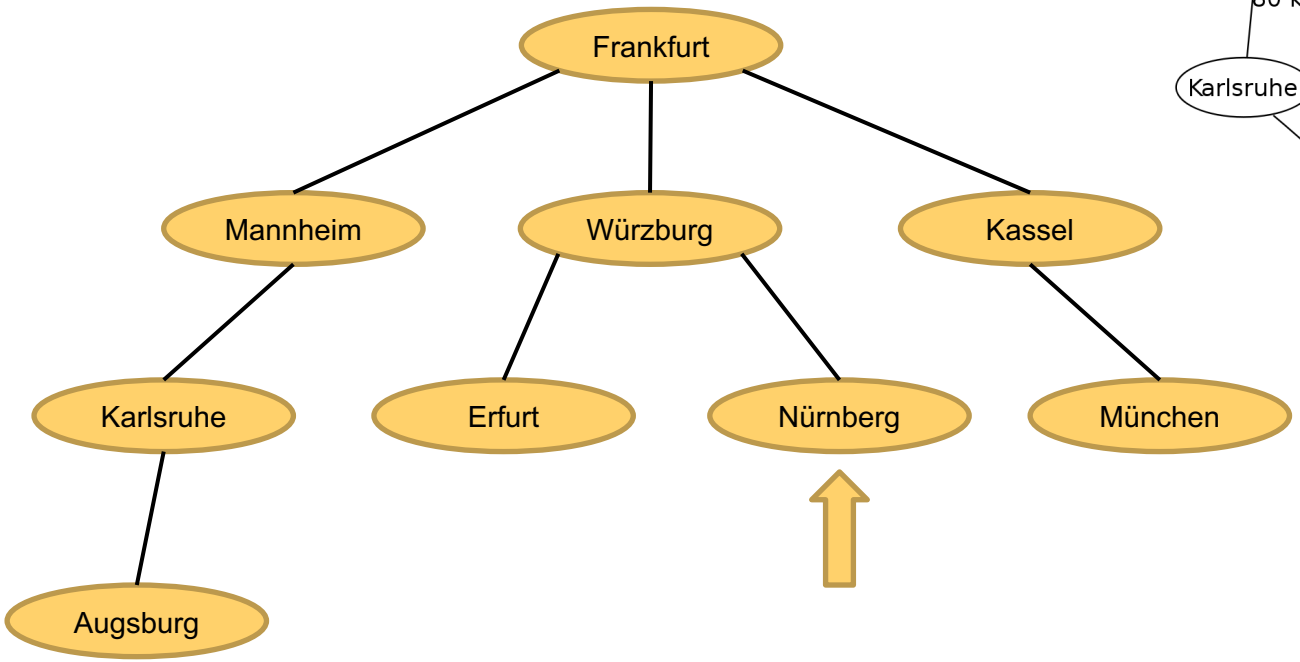
wie komme ich von Frankfurt nach München?

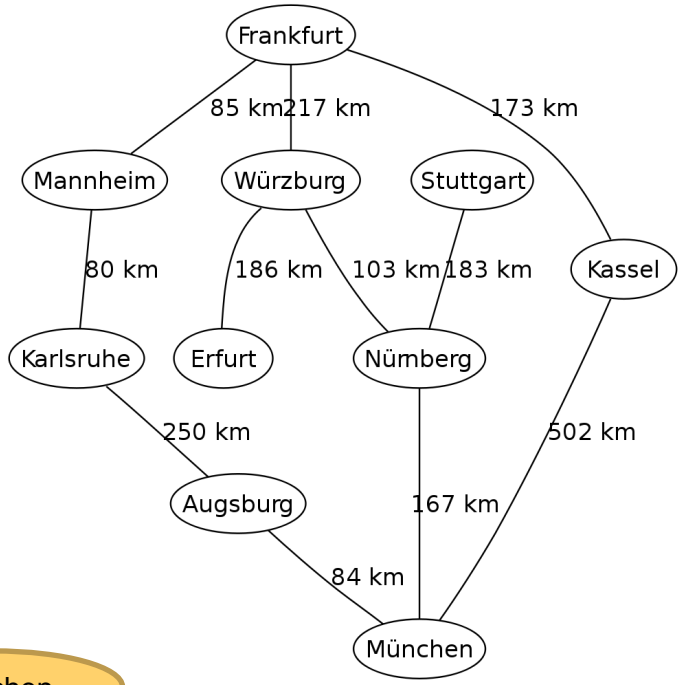




Breitensuche – Beispiel

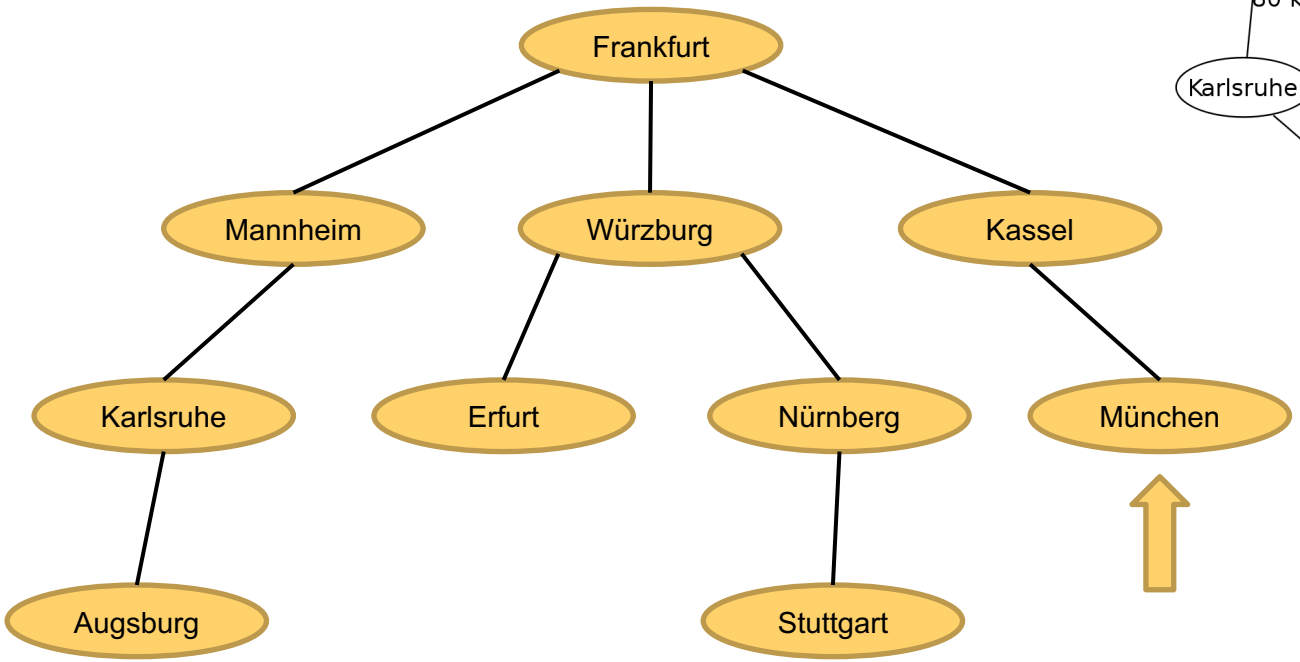
wie komme ich von Frankfurt nach München?

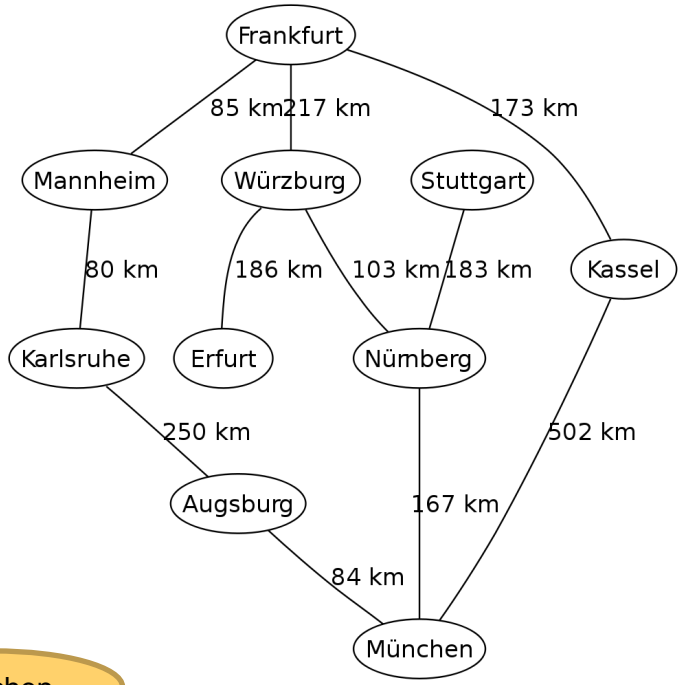




Breitensuche – Beispiel

wie komme ich von Frankfurt nach München?





ist das die kürzeste Route?

Dies ist ein Spannbaum

Bewertung Breitensuche

- Vollständigkeit:
  - JA
  - wenn flachster Zielknoten sich in endlicher d Tiefe befindet
  - und b (maximale Anzahl Nachfolger eines Knotens) endlich
- Optimalität:
  - JA, wenn alle Kanten gleiches Gewicht haben

Bewertung Breitensuche

- Zeitkomplexität
- Angenommen, jeder Knoten hat b Nachfolger
  - Wurzel hat b Nachfolger, jeder Knoten der nächsten Ebene hat wieder b Nachfolger (gesamt: b^2), …
  - angenommen, die Lösung ist auf Tiefe d
  - schlimmster Fall: alle Knoten außer dem letzten auf Tiefe d werden expandiert
  - Gesamtzahl generierter Knoten:
    $b + b^2 + b^3 + \cdots + b^d + \left(b^{d+1} - b\right) = O(b^{d+1})$

Bewertung Breitensuche

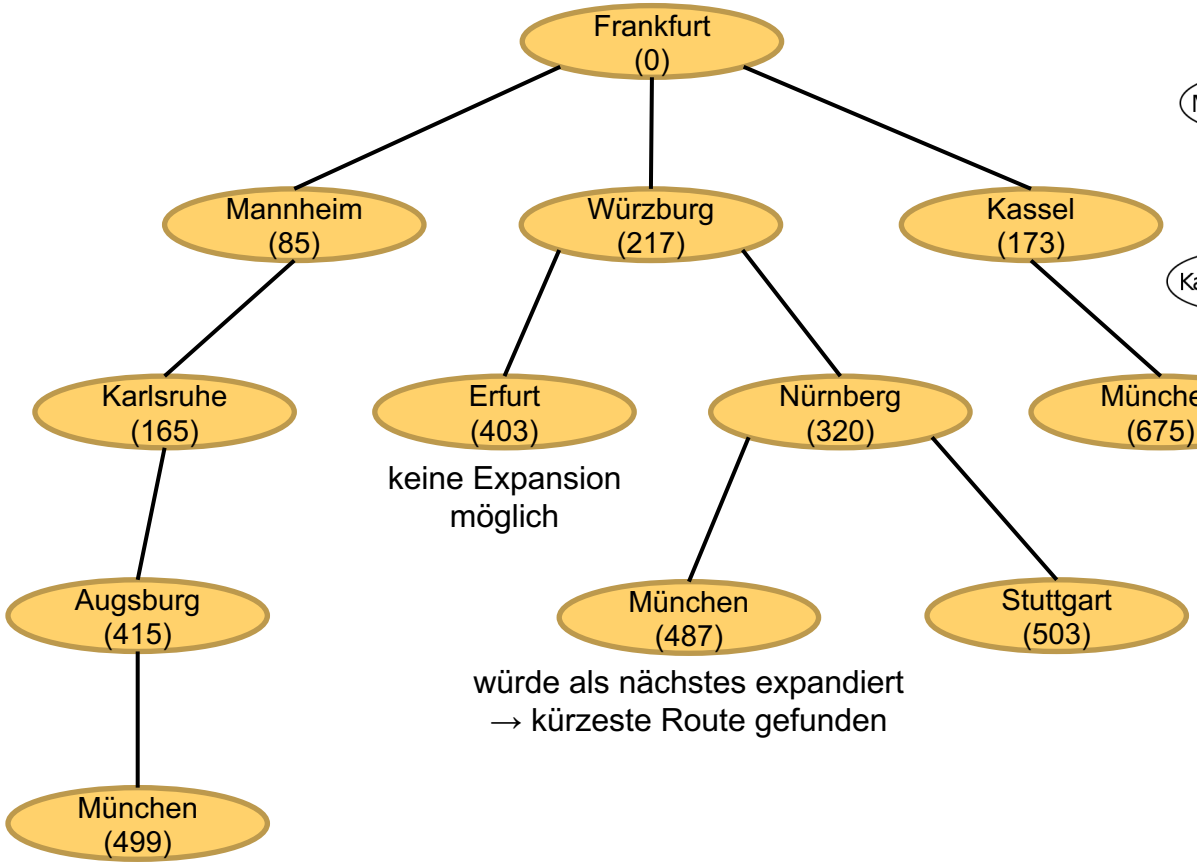
- Vollständigkeit:
  - JA
  - wenn flachster Zielknoten sich in endlicher d Tiefe befindet
  - und b (maximale Anzahl Nachfolger eines Knotens) endlich
- Optimalität:
  - JA
  - wenn alle Kanten gleiches Gewicht haben
- Zeitkomplexität: $O(b^{d+1})$
- Speicherkomplexität:
  - wie Zeitkomplexität, wenn alle Knoten im Speicher gehalten werden

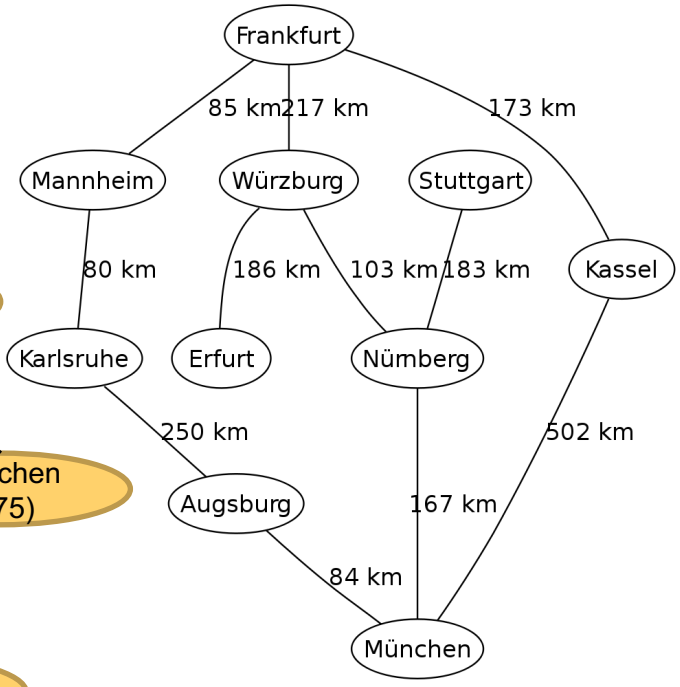
Uniforme Kosten Suche (Uniform Cost Search)

- Erweiterung der Breitensuche für gewichtete Graphen
- Strategie
  - ordne jedem Knoten die Gesamtkosten von der Wurzel aus zu
  - expandiere den Knoten, der die bisher geringsten Gesamtkosten hat
  - Ende, wenn als nächstes der Zielknoten expandiert würde (d.h. er steht vorne in der Warteliste)
- Datenstruktur für Blätter: Prioritätswarteliste (Priority Queue)

Uniforme Kosten Suche – Beispiel

wie komme ich von Frankfurt nach München?





Bewertung Uniforme Kosten Suche

- Vollständigkeit:
  - JA
  - wie Breitensuche
- Optimalität:
  - JA
  - wenn Kantengewichte nicht negativ
- Zeitkomplexität: $O(b^{1+c/e})$
  - c: Kosten der optimalen Lösung
  - e: minimal mögliches (positives) Kantengewicht
- Speicherkomplexität:
  - wie Zeitkomplexität, wenn alle Knoten im Speicher gehalten werden

Bewertung Breitensuche / uniforme Kosten Suche

- Speicheranforderungen sind hier oft ein größeres Problem als Laufzeit
- Suchprobleme mit exponentieller Komplexität können durch uninformierte Suchalgorithmen nur für sehr kleine Probleme gelöst werden
- Beispiel: Breitensuche
  - Annahme: b = 10; 100 000 Knoten/sec; 1000 byte/Knoten

Lösung auf Tiefe	expandierte Knoten	Zeit	Speicher
2	1100	0,011 sec	1 MB
4	111100	1,1 sec	106 MB
6	10^7	1,9 min	10 GB
8	10^9	3,1 h	1 TB
10	10^{11}	13 Tage	101 TB
12	10^{13}	3,5 Jahre	10 Petabyte
14	10^{15}	352 Jahre	1 Exabyte

Informierte Suche

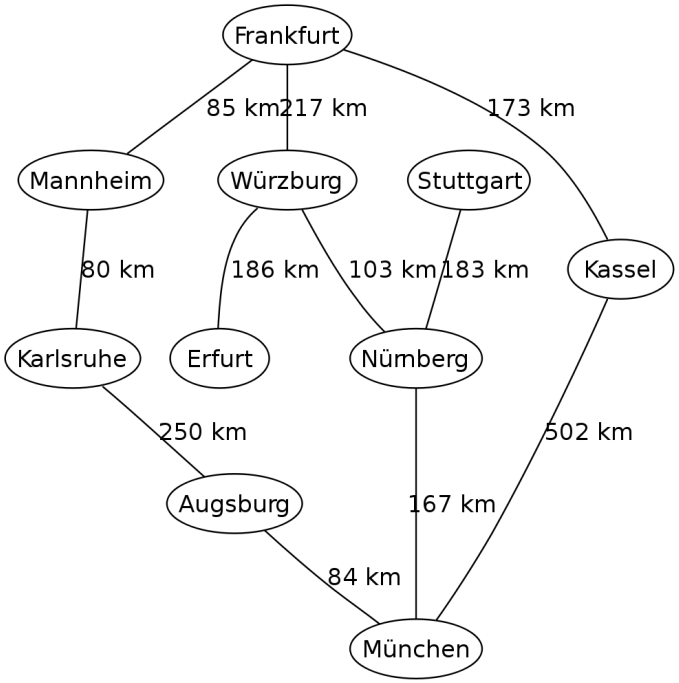
- Idee: Bestensuche (best-first search)
  - zu expandierender Knoten wird auf Basis einer *Evaluierungsfunktion* $f(n)$ ausgewählt
- Evaluierungsfunktion misst Entfernung zum Ziel
  - wähle den Knoten, der am besten erscheint
- Implementierung:
  - Datenstruktur ist eine Warteliste absteigend sortiert nach Bewertung
  - hier betrachtet: A* Suche

Heuristische Funktion

- $h(n)$ = geschätzte Kosten des kürzesten/billigsten Pfads vom Knoten n zum Zielknoten
- ist n der Zielknoten, dann gilt $h(n) = 0$

- wichtig:
  - $h(n)$ muss zusätzliche Information bereitstellen,
  - d.h. es darf nicht aus dem Graphen selbst berechenbar sein

Beispiel



Entfernungen der Städte von München – Luftlinie

Stadt	Entfernung
Frankfurt	304
Mannheim	272
Würzburg	218
Stuttgart	189
Kassel	382
Karlsruhe	253
Erfurt	317
Nürnberg	149
Augsburg	57
München	0

Bestensuche

- Idee: vermeide die Expansion von Pfaden, die bereits teuer sind

- Auswahl basierend auf **Evaluierungsfunktion** $f(n)$
- $f(n) = g(n) + h(n)$
  - $g(n)$ bisherige Kosten vom Start zum Knoten n
  - $h(n)$ geschätzte Kosten von n zum Ziel
  - $f(n)$ geschätzte Gesamtkosten von Start zum Ziel über n

- wähle den Knoten zur Expansion, der die geringsten Gesamtkosten hat

- Spezialfall: $f(n) = h(n)$
  - Greedy-Suche (greedy = gierig)
  - expandiere den Knoten, der die kleinsten geschätzten Kosten zum Ziel hat

A*-Suche

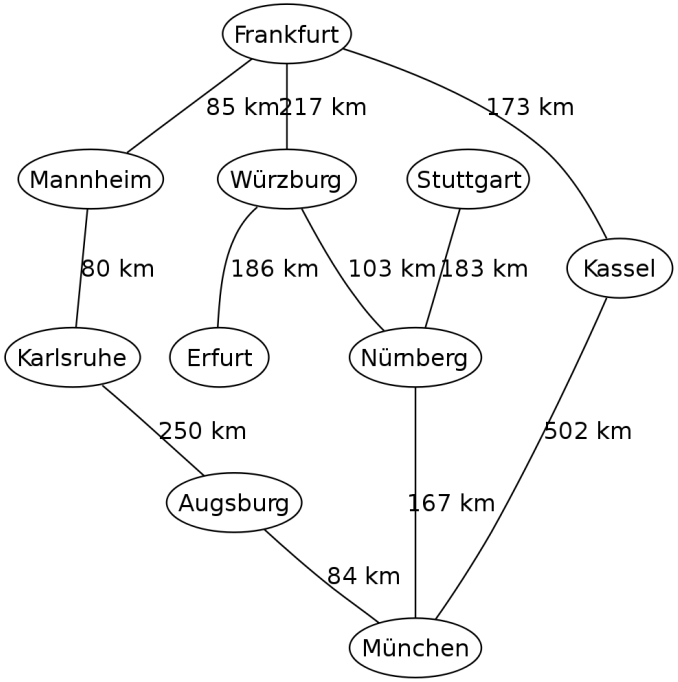
- A* verwendet eine optimistische Heuristik
  - die Kosten zum Ziel werden niemals überschätzt
  - $h(n) \leq h^*(n)$, wobei $h^*(n)$ die tatsächlichen Kosten von n zum Ziel sind

- meist: monotone Heuristik
  - stärker einschränkend
  - zusätzlich muss gelten: $h(n) \leq t(n, n') + h(n')$
    - n und n' sind Nachbarknoten
    - $t(n, n')$ sind die tatsächlichen Kosten von n nach n'
    - die geschätzten Kosten zum Ziel müssen kleiner sein als die Kosten über einen beliebigen Nachbarknoten (Dreiecksungleichung)
  - euklidischer Abstand ist monoton (z.B. Luftlinie)

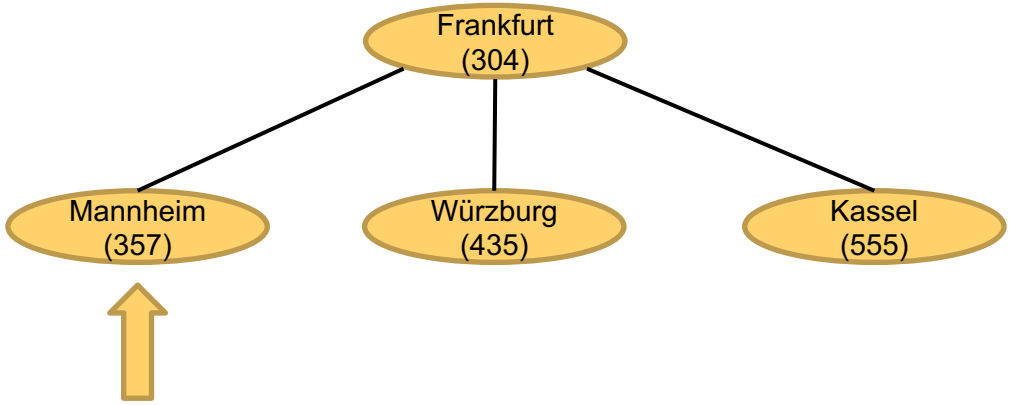
Beispiel – von Frankfurt nach München

$f(F) = g(F, F) + h(F) = 0 + 304 = 304$

Stadt	Entfernung
Frankfurt (F)	304
Mannheim (MA)	272
Würzburg (WÜ)	218
Stuttgart (S)	189
Kassel (KS)	382
Karlsruhe (KA)	253
Erfurt (EF)	317
Nürnberg (N)	149
Augsburg (A)	57
München (M)	0



Beispiel – von Frankfurt nach München

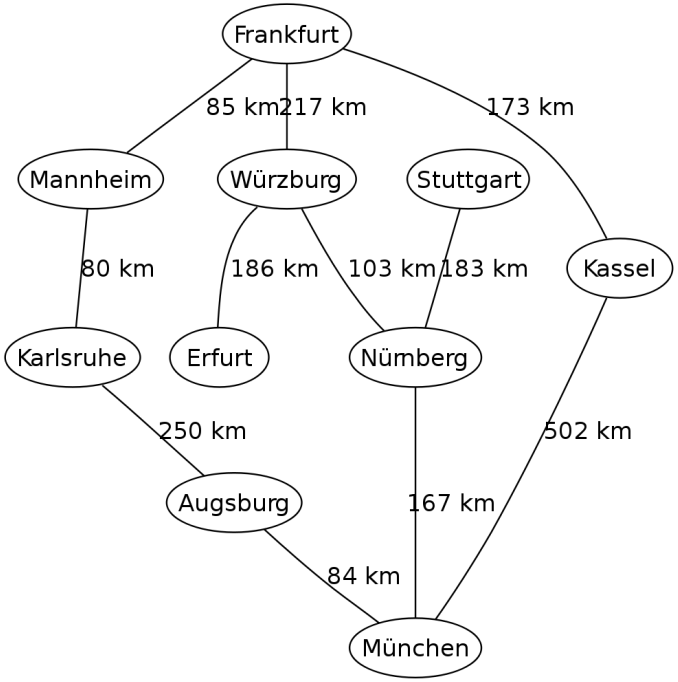


Stadt	Entfernung
Frankfurt (F)	304
Mannheim (MA)	272
Würzburg (WÜ)	218
Stuttgart (S)	189
Kassel (KS)	382
Karlsruhe (KA)	253
Erfurt (EF)	317
Nürnberg (N)	149
Augsburg (A)	57
München (M)	0

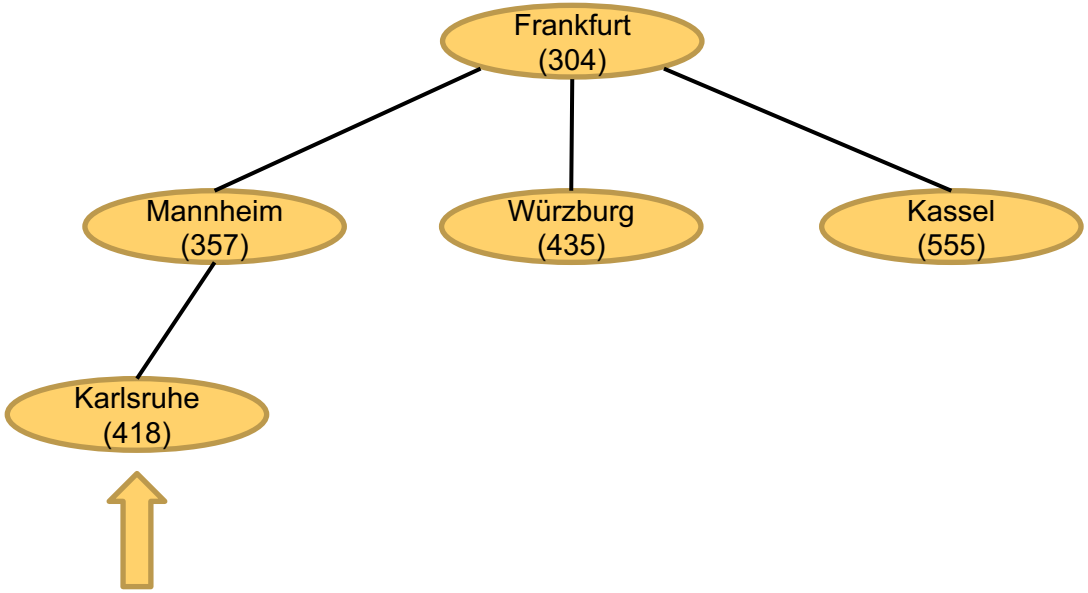
f(MA) = g(F, MA) + h(MA) = 85 + 272 = 357

f(WÜ) = g(F, WÜ) + h(WÜ) = 217 + 218 = 435

f(KS) = g(F, KS) + h(KS) = 173 + 382 = 555

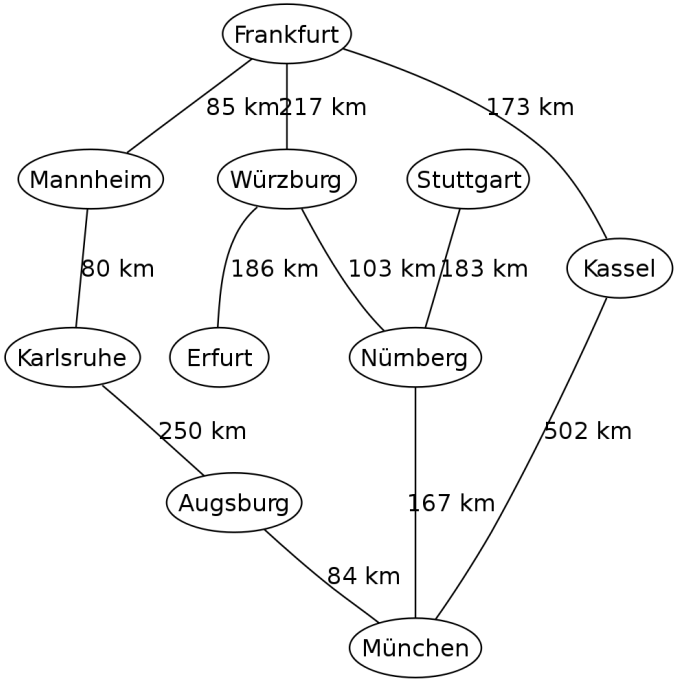


Beispiel – von Frankfurt nach München

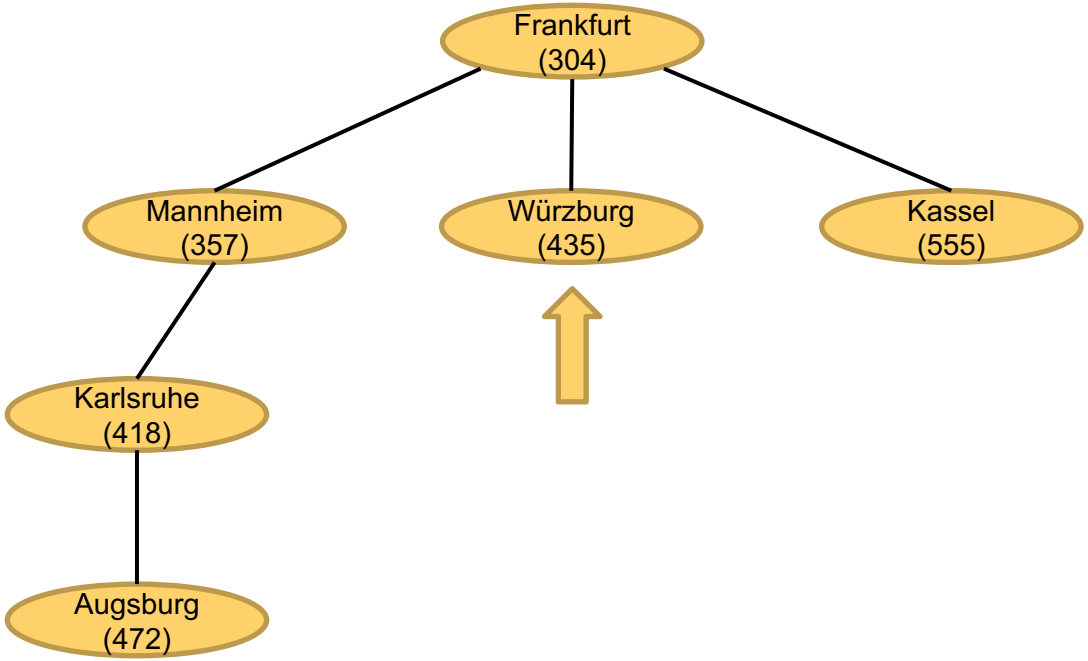


Stadt	Entfernung
Frankfurt (F)	304
Mannheim (MA)	272
Würzburg (WÜ)	218
Stuttgart (S)	189
Kassel (KS)	382
Karlsruhe (KA)	253
Erfurt (EF)	317
Nürnberg (N)	149
Augsburg (A)	57
München (M)	0

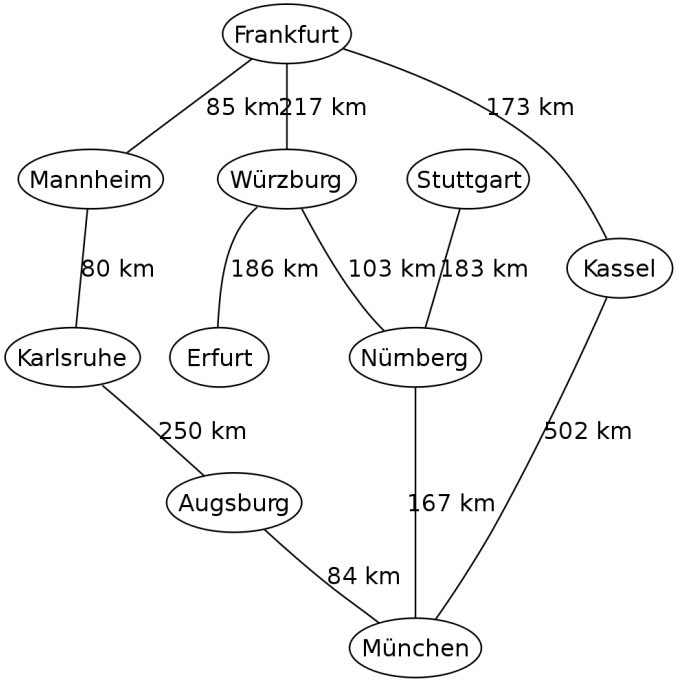
f(KA) = g(F, KA) + h(KA) = 165 + 253 = 418



Beispiel – von Frankfurt nach München

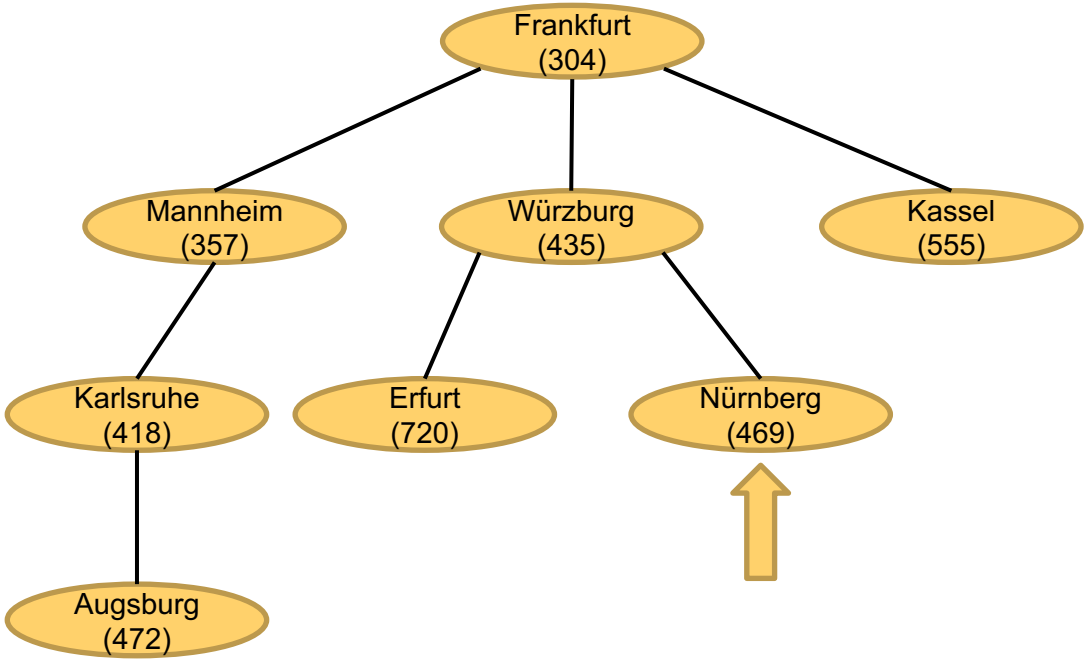


Stadt	Entfernung
Frankfurt (F)	304
Mannheim (MA)	272
Würzburg (WÜ)	218
Stuttgart (S)	189
Kassel (KS)	382
Karlsruhe (KA)	253
Erfurt (EF)	317
Nürnberg (N)	149
Augsburg (A)	57
München (M)	0

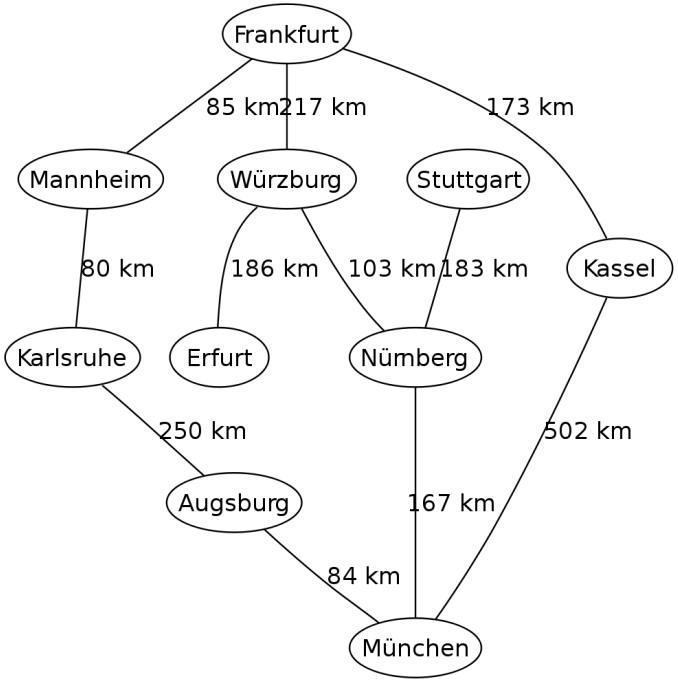


f(A) = g(F, A) + h(A) = 415 + 57 = 472

Beispiel – von Frankfurt nach München



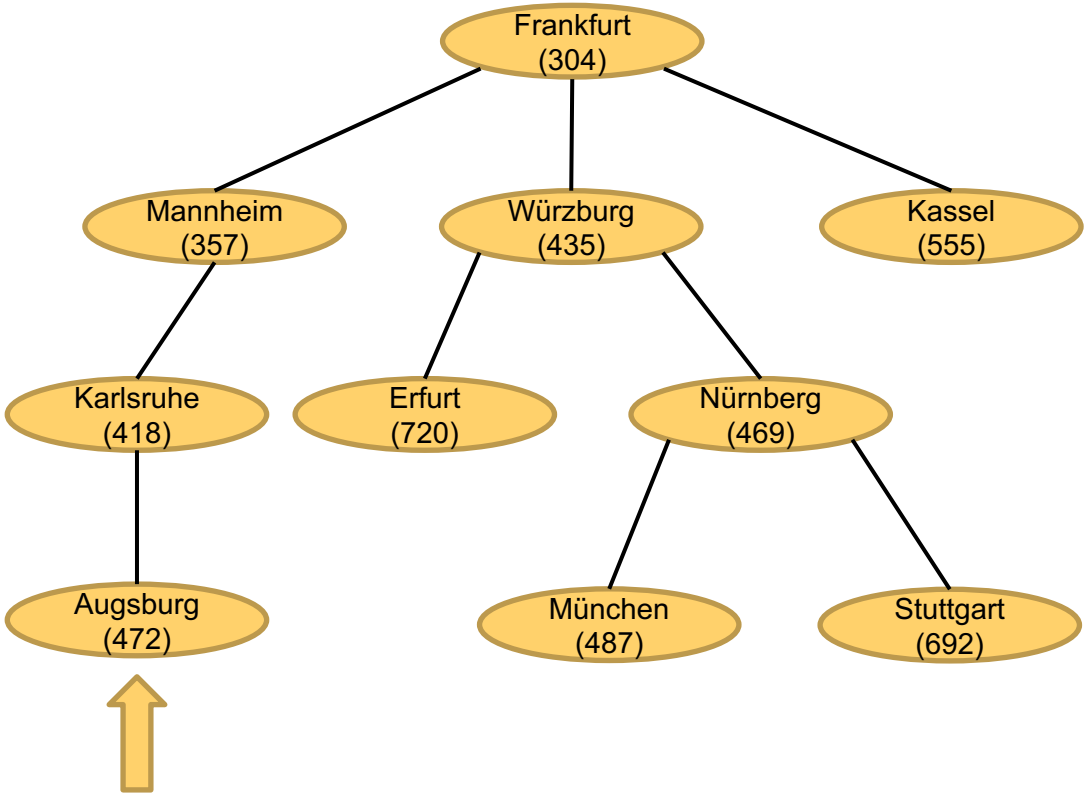
Stadt	Entfernung
Frankfurt (F)	304
Mannheim (MA)	272
Würzburg (WÜ)	218
Stuttgart (S)	189
Kassel (KS)	382
Karlsruhe (KA)	253
Erfurt (EF)	317
Nürnberg (N)	149
Augsburg (A)	57
München (M)	0



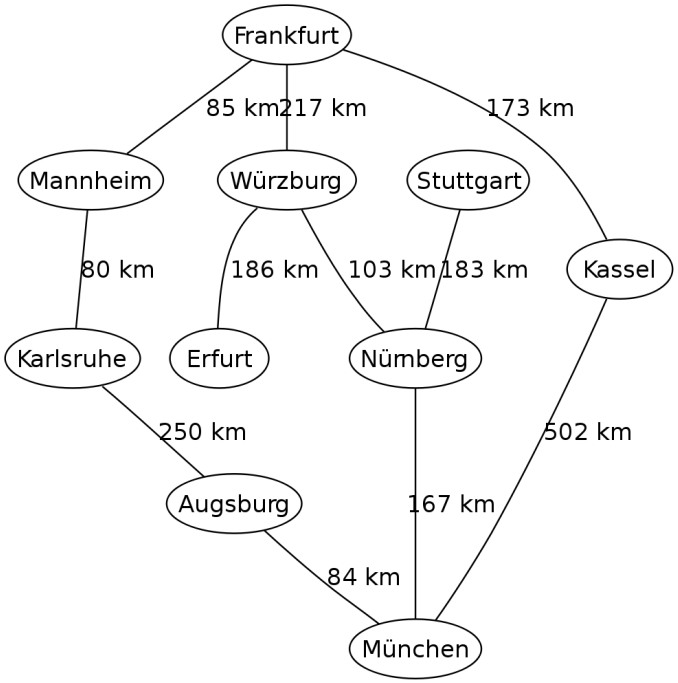
f(EF) = g(F, EF) + h(EF) = 403 + 317 = 720

f(N) = g(F, N) + h(N) = 320 + 149 = 469

Beispiel – von Frankfurt nach München



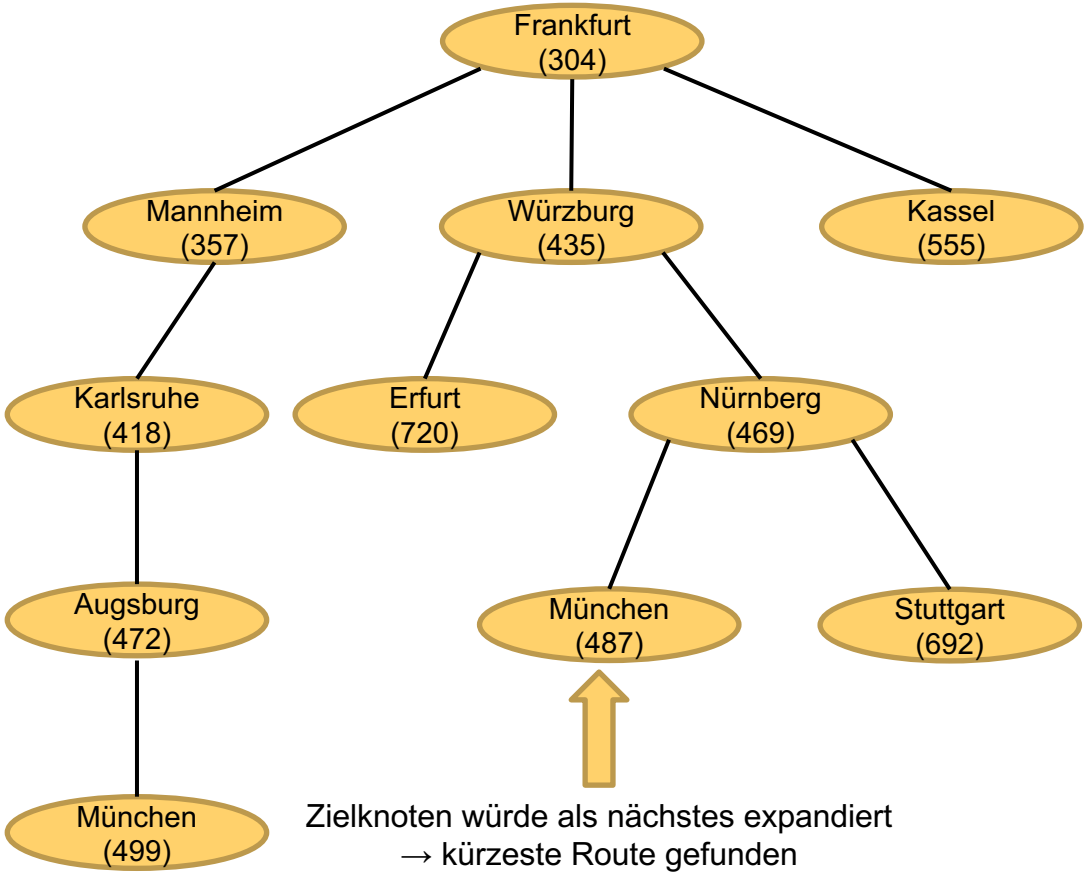
Stadt	Entfernung
Frankfurt (F)	304
Mannheim (MA)	272
Würzburg (WÜ)	218
Stuttgart (S)	189
Kassel (KS)	382
Karlsruhe (KA)	253
Erfurt (EF)	317
Nürnberg (N)	149
Augsburg (A)	57
München (M)	0



f(M) = g(F, M) + h(M) = 487 + 0 = 487

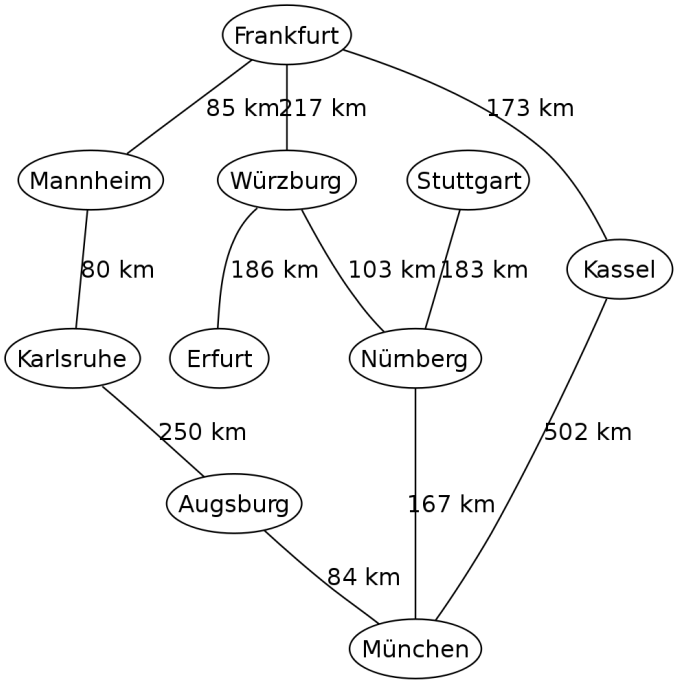
f(S) = g(F, S) + h(M) = 503 + 189 = 692

Beispiel – von Frankfurt nach München



Zielknoten würde als nächstes expandiert
→ kürzeste Route gefunden

Stadt	Entfernung
Frankfurt (F)	304
Mannheim (MA)	272
Würzburg (WÜ)	218
Stuttgart (S)	189
Kassel (KS)	382
Karlsruhe (KA)	253
Erfurt (EF)	317
Nürnberg (N)	149
Augsburg (A)	57
München (M)	0



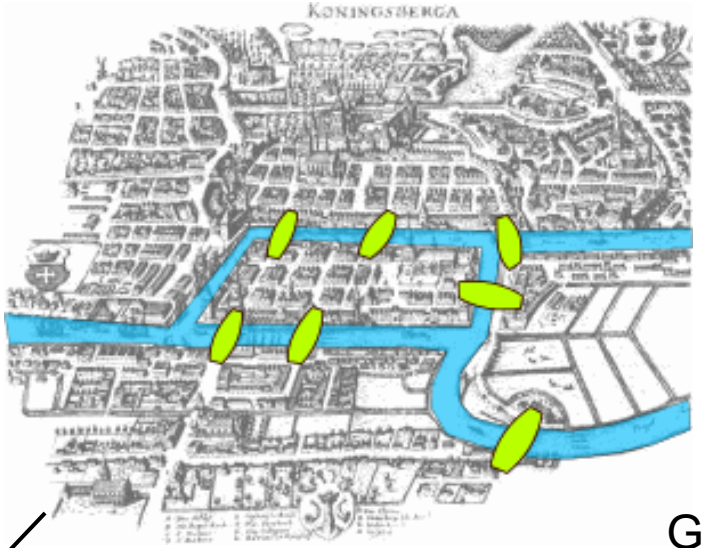
f(M) = g(F, M) + h(M) = 499 + 0 = 499

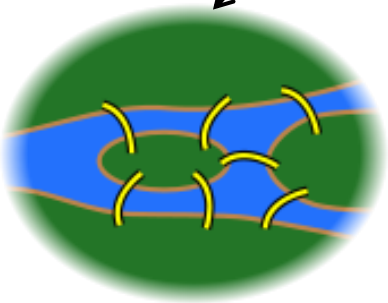
Bewertung A*-Suche

- Vollständigkeit:
  - JA
- Optimalität:
  - JA
  - wenn die Bedingungen für $h(n)$ eingehalten werden
  - A* ist sogar optimal-effizient: kein anderer optimaler Algorithmus expandiert weniger Knoten (Ausnahme: gleiche Bewertung verschiedener Knoten)
- Zeitkomplexität:
  - exponentiell in der Länge des kürzesten Pfads
- Speicherkomplexität:
  - hält alle Knoten im Speicher
  - daher ist Speicherverbrauch das Hauptproblem, nicht die Rechenzeit

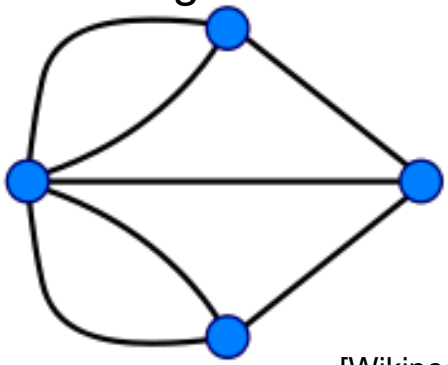
Zurück zum Start

Euler 1736: Gibt es einen Rundweg durch Königsberg, der jede der sieben Brücken über die Pregel genau einmal überquert?





Gibt es einen Rundweg, der jede Kante genau einmal enthält?



[Wikipedia, Public Domain]

Eulersche Graphen

- Eulerkreis: geschlossener Weg, der alle Kanten des Graphen genau einmal enthält
- Eulerscher Graph: zusammenhängender Graph, der einen Eulerkreis besitzt

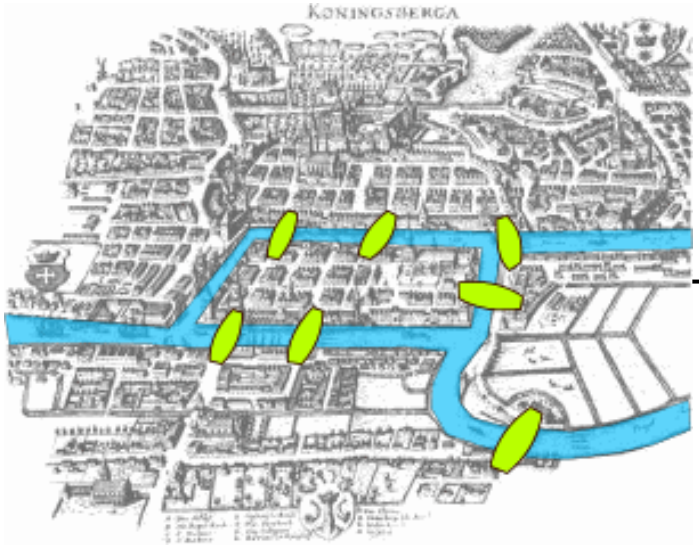
- ungerichtete Graphen:
  Graph ist eulersch genau dann wenn
  - Graph zusammenhängend und Grad jedes Knotens ist gerade oder
  - Graph zusammenhängend und Kantenmenge des Graphen besteht aus der Vereinigung paarweise disjunkter Kreise

Eulersche Graphen

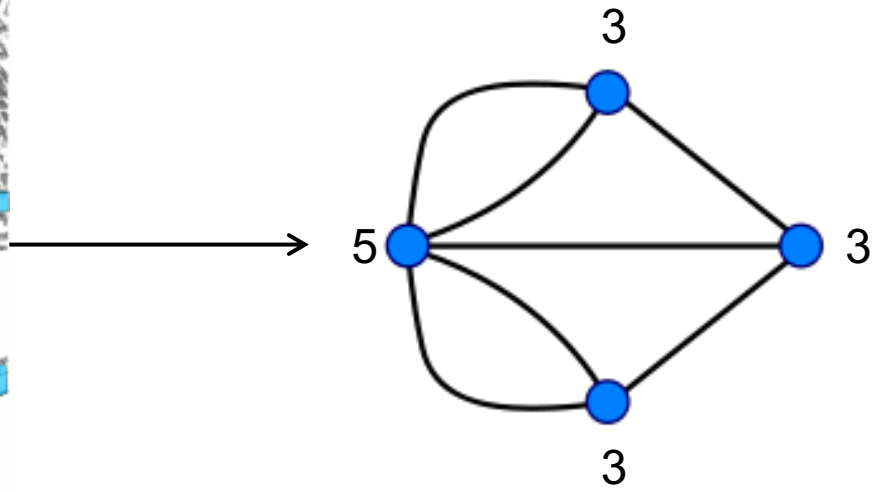
- das Entscheidungsproblem ist damit relativ leicht zu lösen
- Ist ein gegebener Graph eulersch?
  - Man prüft Zusammenhang und Grad jedes Knotens
- für das tatsächliche Finden eines Eulerkreises gibt es effiziente Algorithmen
  - soll hier nicht betrachtet werden

Königsberger Brückenproblem

Euler 1736: Gibt es einen Rundweg durch Königsberg, der jede der sieben Brücken über die Pregel genau einmal überquert?



Gibt es einen Rundweg, der jede Kante genau einmal enthält?



NEIN

Hamiltonsche Graphen

- gibt es einen geschlossenen Pfad, der jeden Knoten genau einmal enthält?
- Spezialfall: Problem des Handlungsreisenden
  - kürzester Hamiltonkreis
- dieses Problem ist ungleich schwieriger
  - NP-vollständig (→ Theoretische Informatik 2. Sem.)
  - außer in einigen Spezialfällen
    - z.B. ist jeder vollständige Graph mit 3 Knoten oder mehr hamiltonsch