Informatik im Alltag: Wie funktionieren Navis?

Thema: Wegeprobleme (aus *Grundlagen der Informatik 2*)

Dr. Guido Rößling

TU Darmstadt

Angelehnt an Material von Dr. Jens Gallenbacher

Problemlösen für den Alltag

TECHNISCHE UNIVERSITÄT DARMSTADT

- Informatik ist viel mehr als „Programmieren“!
  - → Systematische Problemanalyse, -lösung und Datenverarbeitung
- Einige Aufgabenbereiche der Informatik:
  - Robotik – von der „Fußball-WM“ über Fabrikation bis Lebensrettung
  - Abstraktion – nicht *ein* Problem lösen, sondern *alle verwandten*
  - Design – für Software, Benutzerschnittstellen, bis zu Computerspielen
  - Modellierung – bei passender Modellierung ist die Lösung einfacher
  - Software-Entwicklung – von Sensoren (Auto, Handy, Apple Watch) über Softwaresysteme (Notepad) zu „großer“ Software (Office, Betriebssysteme)
  - IT-Sicherheit – Verschlüsselung von Mails, sichere Kreditkarten, ...
- Vielfach zusammen mit anderen Disziplinen (Mathe, Physik, Elektrotechnik, Maschinenbau, ...)

Leitbeispiel: Navigationssystem

TECHNISCHE UNIVERSITÄT DARMSTADT

- Jedes „normale“ Navi berechnet schnell und problemlos den *besten* Weg von einem Start- zu einem Zielpunkt
- Dabei gibt es in aller Regel *sehr viele* mögliche Wege
  - Einige sind „naheliegend“: „nimm bei mehr als 50km die Autobahn“
  - Andere weniger: die Route über Bundesstraßen etc. kann *kürzer* sein
- Wie findet man den „besten“ Weg?
- Vor allem: was heißt „bester“ Weg?
  - *Kürzeste Wegstrecke* (in Kilometern)?
  - *Schnellste* Route („Autobahn: 130 km/h, Bundesstraße: 90 km/h“)?
  - *Schönste* Strecke (Sehenswürdigkeiten berücksichtigen)?
  - ...?
- Das *komplex* wirkende Problem ist eigentlich *einfach lösbar*
  - Man muss nur wissen, wie...

„Brute Force-Ansatz“

TECHNISCHE UNIVERSITÄT DARMSTADT

- Der „brutale Ansatz“ („Brute Force“) probiert einfach alle möglichen Wege aus
- Am Ende wird der „günstigste“ Weg gewählt
- **Problem dabei**: Der Ansatz „denkt nicht mit“
  - Der kürzeste Weg von Darmstadt nach Hamburg führt *sicher* nicht über München, auch wenn das ein *möglicher* Weg ist!
  - Keine „Kappung“, sobald der Weg zu „teuer“ wird
- Konsequenzen
  - Kein „intelligentes Verwerfen“ von „offenbar unsinnigen“ Lösungen
  - Es sind *extrem viele* Wege zu berechnen → dauert lange
  - Immerhin: die beste Lösung wird **garantiert** gefunden.

„Branch and Bound“

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ein kurzes Nachdenken zeigt, dass es keine Wege mit *negativem* Wert geben kann (selbst beim Rückwärtsfahren!)

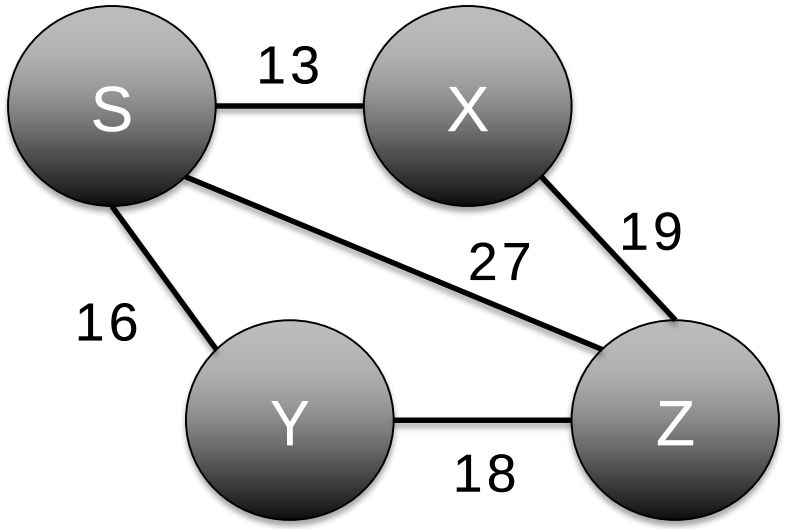
1. Bestimme einen (beliebigen *gültigen*) „besten“ Weg zum Ziel
   - Egal, wie „gut“ oder „schlecht“ dieser ist (!)
2. Setze den „aktuellen Knoten“ auf den Startknoten
3. Wähle einen direkt vom aktuellen Knoten erreichbaren Knoten
4. Betrachte den *aktuellen* Weg nach Hinzufügen eines Teilziels
   - Ist der Weg nun *länger* als der bisher beste Weg?
     - Ja → das kann nicht der beste Weg sein, *verwerfen des Knotens*
     - Nein → Knoten zu aktuellem Knoten machen, weiter mit 3. bis Ziel erreicht ist.

- Der billigste Weg wird sicher gefunden – mit einigem Aufwand

Konkretes Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

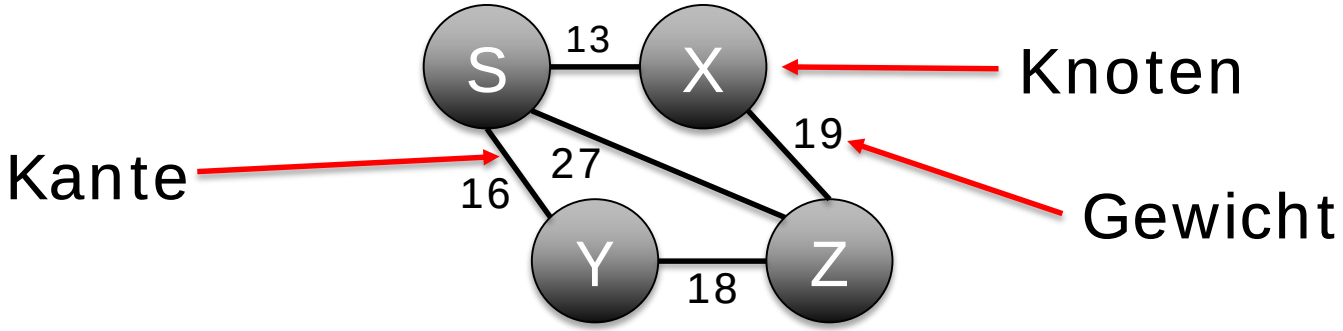
- Wir nutzen eine konkrete (aber erfundene) Karte für ein „echtes Navigationsproblem“
- Wir nutzen ein paar Fachbegriffe aus der *Graphentheorie*
- Das Straßennetz wird durch einen *Graph* repräsentiert



Konkretes Beispiel

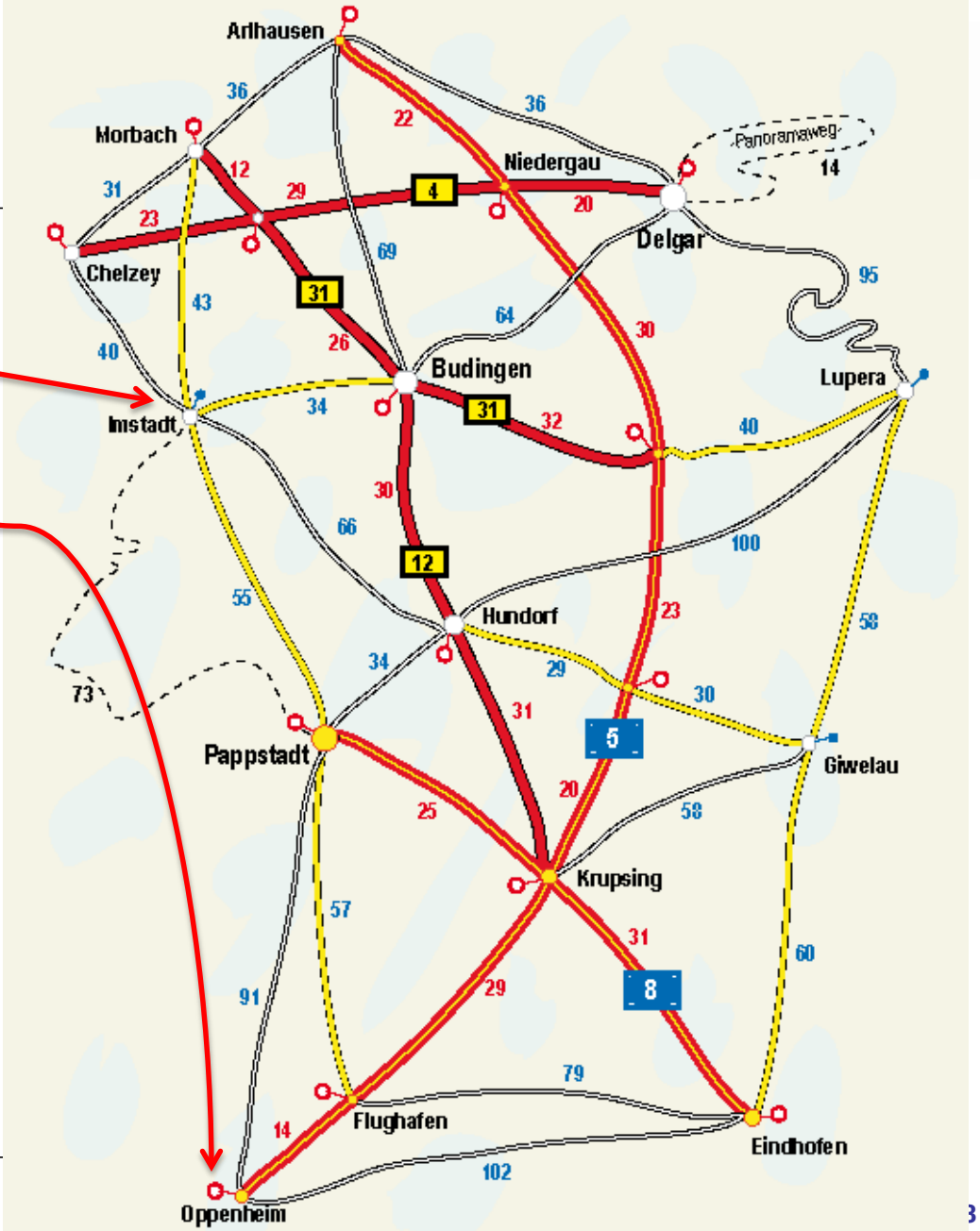
TECHNISCHE UNIVERSITÄT DARMSTADT

- Graphen bestehen aus *Knoten* und *Kanten*
  - *Knoten*: Verkehrsknotenpunkte (Stadt, Kreuzung, Autobahnkreuz, ...)
  - *Kante*: Verbindung zwischen zwei Knoten (Straße, Autobahn, ...)
  - Jede Kante hat ein *Gewicht*: bei uns die „Kosten“ (Entfernung in km) zwischen den beiden Knoten
- Die *Darstellung* des Graphen passen wir an *unser Problem an*
  - Sie muss *nicht* der exakten Wirklichkeit entsprechen
  - Die *Gewichte* müssen natürlich dennoch korrekt sein ☺
- Graphentheorie ist alt: „Erfinder“ *Leonhard Euler* (1707-1783)



Beispielproblem

- Was ist die „günstigste“ Route von **Imstadt** nach **Oppenheim**?
- „*Faustformel Autobahn*“:
  - Imstadt→Budingen 34km
  - Budingen→A5 32km
  - A5→Oppenheim 86km
  - Insgesamt 152km
- Geht das nicht kürzer?
- → *Wie löst man das?*



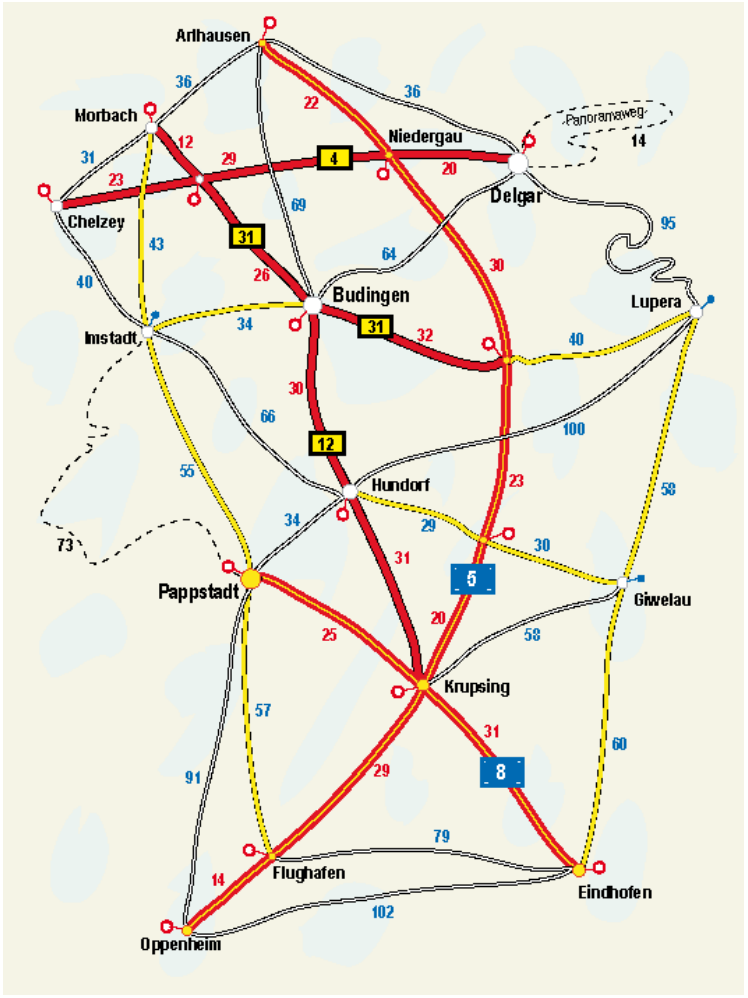
1. Schritt: Abstraktion

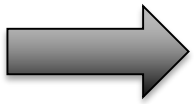
TECHNISCHE UNIVERSITÄT DARMSTADT

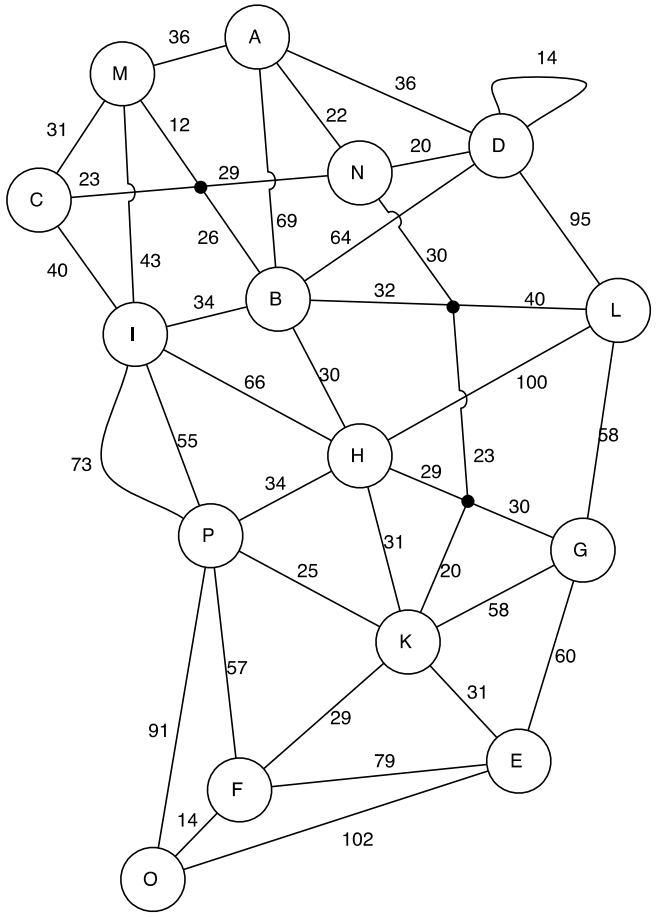
- Welche der Informationen *brauchen* wir für die Wegesuche?

	wichtig	
Namen der Städte	☐ ☺	
Position der Städte	☐ ⊘	Nur Straßenlänge
Größe der Städte	☐ ⊘	
Verlauf der Straßen	☐ ⊘	Nur Straßenlänge
Länge der Straßen	☐ ☺	
Namen und Nummern der Straßen	☐ ⊘	
Straßentyp	☐ ⊘	
Straße führt von ... nach ...	☐ ☺	
Landschaftliche Information	☐ ⊘	

Abstraktion: Namen, Straßentyp





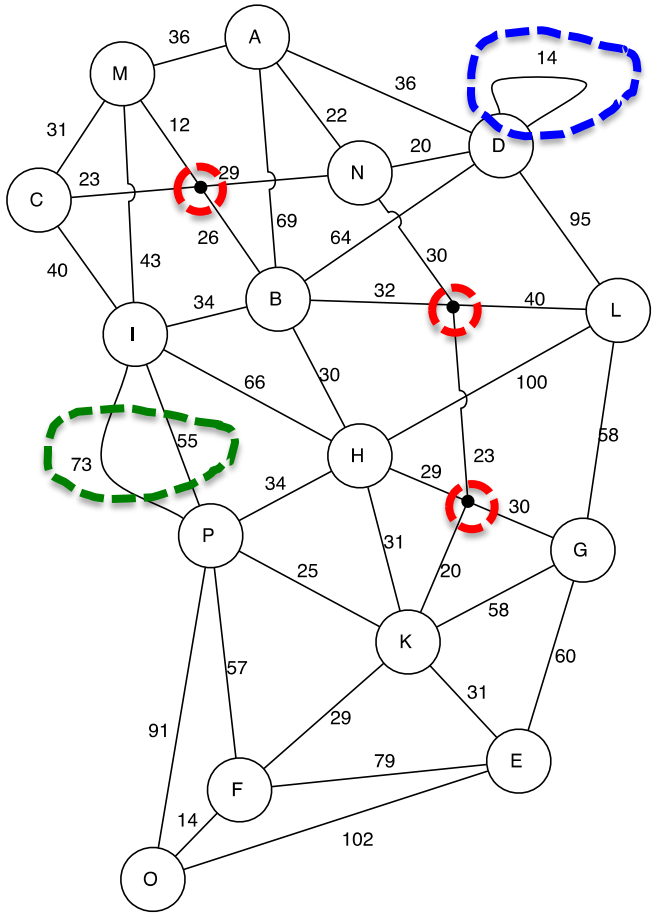


Abstraktion: Namen, Straßentyp

TECHNISCHE UNIVERSITÄT DARMSTADT

Vorgenommene Anpassungen:

1. Ortsnamen durch Buchstaben ersetzt
   - Erleichtert Betrachtung
   - Gesuchte Route nun „I → O“
2. Graph leicht neu gezeichnet
   - Entfernungen besser lesbar
   - Geographische Lage egal



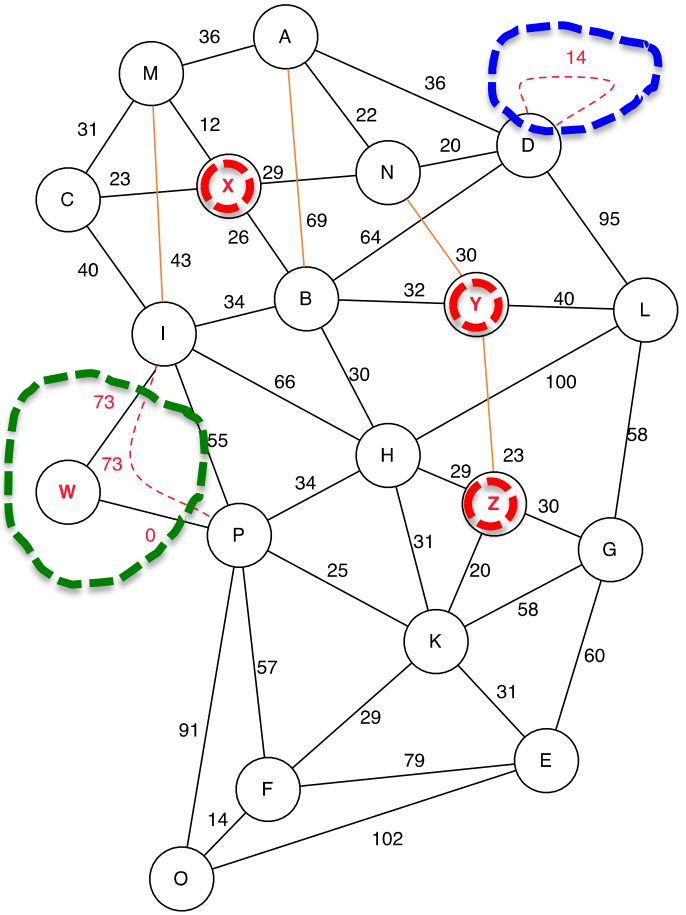
Verbleibende Probleme:

- Kreuzungen ohne Knoten
  - → Knoten einfügen
- „Schleife“ → entfernen
- Längerer „Bogen“ → ummodellieren

Abstraktion: „besserer“ Graph

TECHNISCHE UNIVERSITÄT DARMSTADT

- Kreuzung → zu Knoten X, Y, Z „aufgewertet“
- Kreuzungen ohne Querung → kein „Bogen“ mehr nötig
- „Schleife“ → entfernt
- Längerer „Bogen“ → Knoten W



Prinzip der Gleichformung

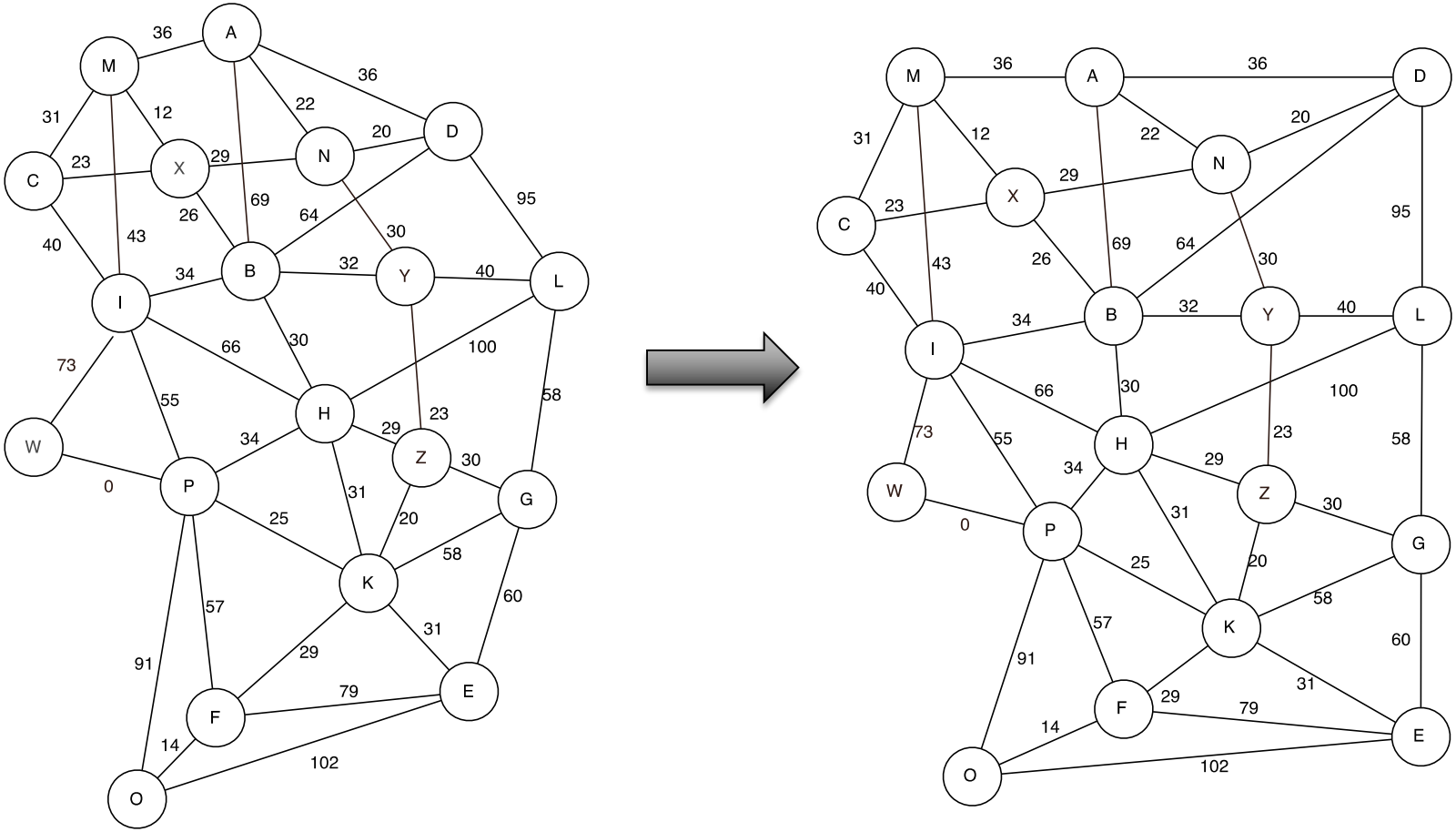
TECHNISCHE UNIVERSITÄT DARMSTADT

- Grundprinzip: „das **Gleiche** in verschiedenen Probleme finden“
- → Aspekte des Problems auf *gleiche Grundelemente* reduzieren
- Bislang: Lösung *eines* Problems
- → Lösung *aller vergleichbaren* Probleme (!)
- Der *gleiche* Lösungsansatz löst *zahlreiche* verwandte Probleme

→„Pragmatische Faulheit“ ☺

- Nächster Schritt:
  - Geographische Lage ist für uns egal
  - Wir können die teilweise „eng gequetschte“ Karte entzerren
    - Wird dadurch übersichtlicher
  - Die „Kantengewichte“ (Straßenlängen) ändern sich *nicht*

Abstraktion: Entzerrung



Navigationstaktik 1: *Erfahrung*

TECHNISCHE UNIVERSITÄT DARMSTADT

- Bestimmen einer Lösung durch *Erfahrung* (Heuristik)
  - „Bei einem hinreichend weit entfernten Ziel, fahre zur nächsten Autobahn, fahre bis zur Nähe des Ziels und fahre dann wieder ab“
- Liefert das *immer* die *günstigste* Strecke?
  - → **Nein!**
- Liefert es wenigstens immer eine *sinnvolle* Strecke?
  - → **Nein!**
- Gegenbeispiel: „Fahre von Erbach nach Darmstadt“
  - → „Fahre nach *Weinheim*, auf die A5 bis Darmstadt, dann zum Ziel“
  - → Weg via B45 bis Dieburg, dann B26 (und Varianten) **viel** kürzer

Navigationstaktik 2: *Brute Force*

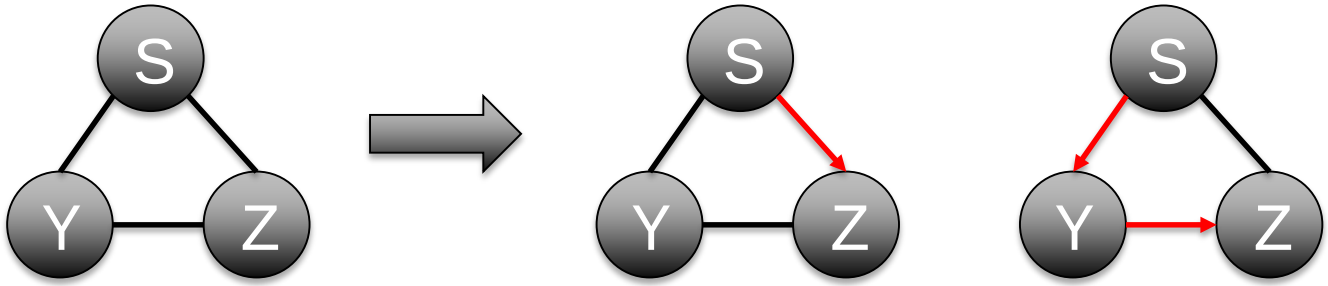
TECHNISCHE UNIVERSITÄT DARMSTADT

- *Alle* möglichen Strecken vom Start zum Ziel berechnen
- → Ziemlich aufwändig
- → Für große Probleme (viele Knotenpunkte) nicht umsetzbar
- „Große Probleme“: wie viele Möglichkeiten gibt es, von der *Hochschulstraße (Darmstadt)* zum *Willy-Brandt-Platz (Frankfurt am Main)* zu kommen?
  - Beide Städte haben ein sehr umfangreiches Straßennetz...

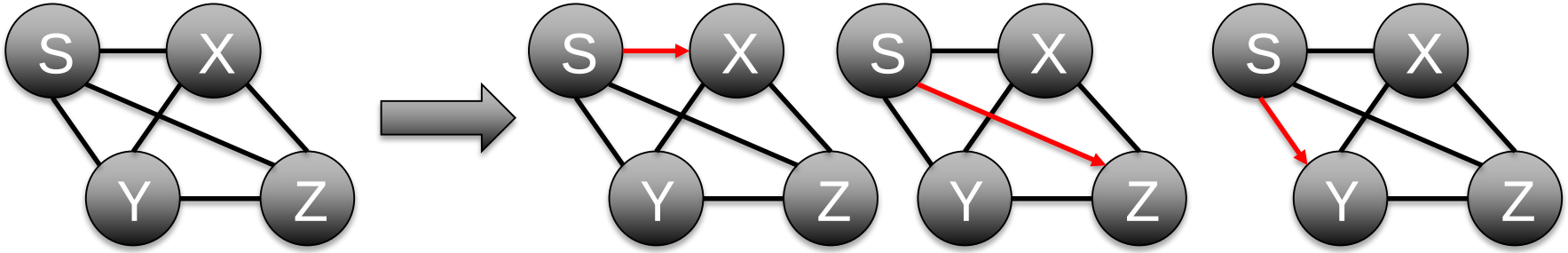
Brute-Force: Aufwand

TECHNISCHE UNIVERSITÄT DARMSTADT

- *Schlimmster Fall:* alle Knoten sind miteinander verbunden
- Lösung für n = 3 Knoten bei Weg S → Z: zwei mögliche Wege



- Lösung für (n + 1) = 4 Knoten mit neuem Startknoten



→ Es stehen 3 (= n – 1) neue Wege zur Verfügung

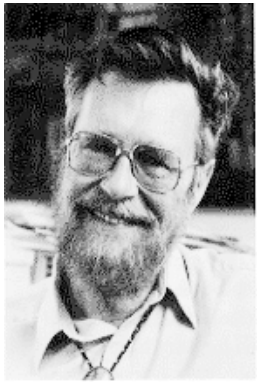
Aufwand: Brute-Force

TECHNISCHE UNIVERSITÄT DARMSTADT

- Verallgemeinert auf *n* Knoten...?
- Der *neue* Knoten kann durch n-1 nutzbare Wege mit dem (n-1) Knoten-System verbunden werden
- → Bei *n* Knoten entstehen bis zu *(n – 1)!* Wege
- Jeder Weg vom Start zum Ziel hat maximal die Länge *n*
- Abgeschätzter Aufwand: *bis zu n * (n-1)! = n!*
- Die Fakultätsfunktion wächst enorm schnell:
  - 6! = 6 * 5 * 4 * 3 * 2 * 1 = 720
  - 7! = 7 * 6! = 5,040
  - 8! = 8 * 7! = 40,320
  - 19! = 19 * ... * 1 = $1{,}216451 * 10^{17}$
    - Unser Graph hat 19 Knoten...
- Deutschland hat ca. 11,418 Gemeinden und unzählige *Straßen*

Lösung: Dijkstra-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT



- Eine Lösung des Problems ist der *Dijkstra-Algorithmus*
- Entwickelt von Edsger Wybe Dijkstra (1930-2002)
  → ca. 1959
- Initialisierung:
  - Setze die Kosten um den Startknoten *S* zu erreichen auf 0
  - Setze die Kosten für alle Knoten *X („Kosten(S→X)“)* auf ∞
  - Beginne mit dem Startknoten als *aktuellem Knoten A*

Lösung: Dijkstra-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

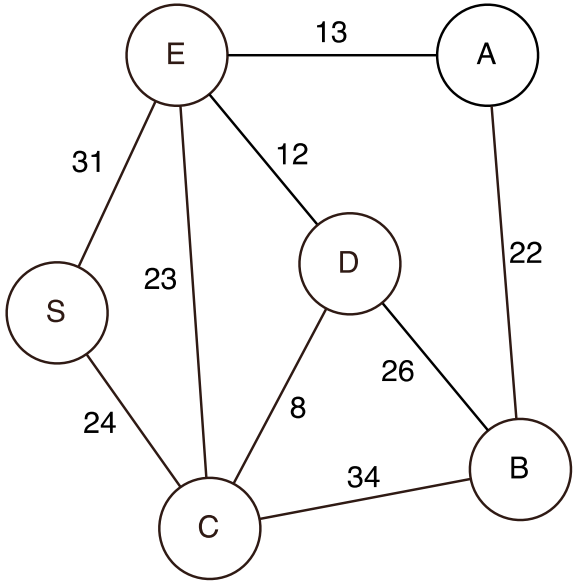
- Wiederhole, bis alle Knoten besucht wurden:
  1. Markiere den aktuellen Knoten *A* als *besucht*
  2. Für alle noch nicht besuchten Knoten *X*:
     a. Falls Kosten(S → A) + Kosten(A → X) < Kosten(S → X), ersetze den Eintrag durch Kosten(S → A) + Kosten(A → X)
  3. Setze den *billigsten erreichbaren, noch nicht besuchten Knoten* zum neuen *aktuellen Knoten A*.
  4. Fahre fort mit Schritt 4, bis alle Knoten besucht wurden.

Lösung: Dijkstra-Algorithmus

TECHNISCHE UNIVERSITÄT DARMSTADT

- Der Algorithmus teilt die Knoten in drei Mengen auf
  - M1: Menge der bereits *besuchten* Knoten
    - Hierfür wurde schon der *günstigste Weg* ab Startknoten berechnet
  - M2: Von aus der Menge M1 *erreichbare unbesuchte* Knoten
  - M3: Alle bislang noch nicht erreichbaren Knoten

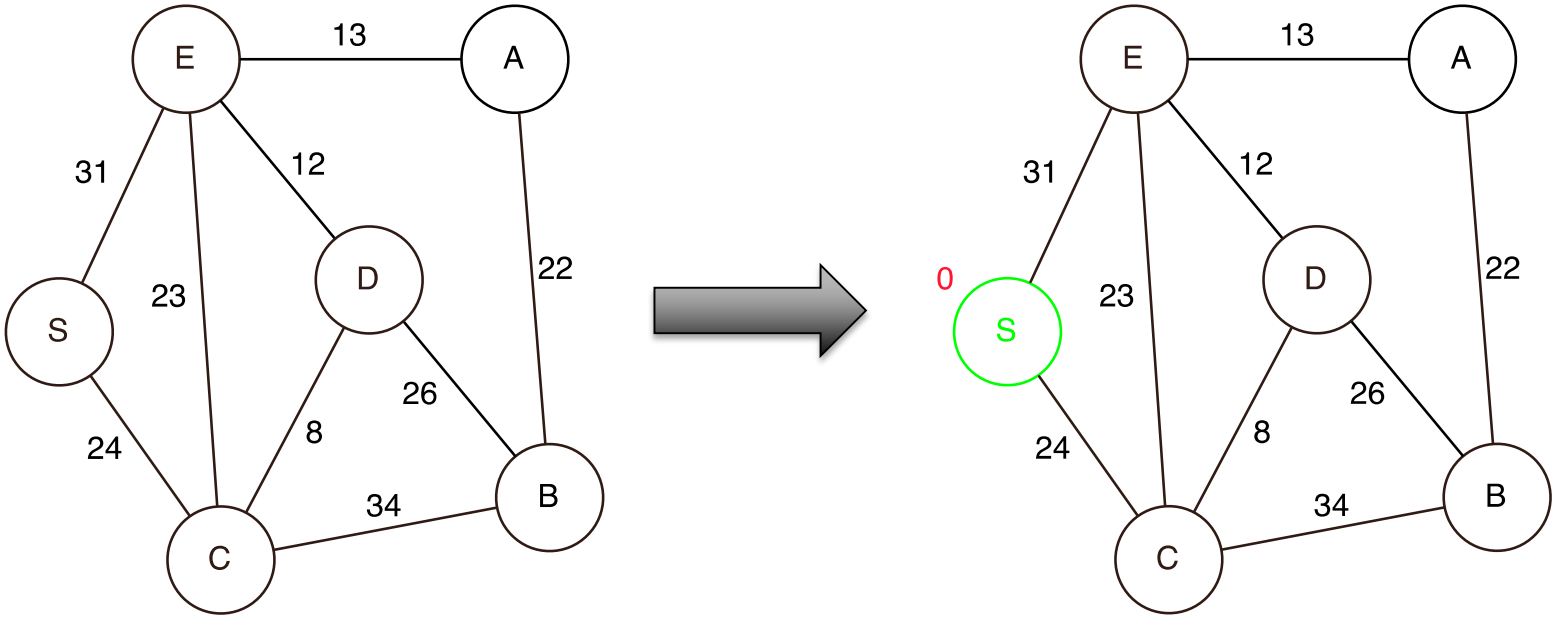
Wir betrachten ein vereinfachtes Beispiel:
Wie ist der kürzeste Weg von S nach A?



Dijkstra-Algorithmus: Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

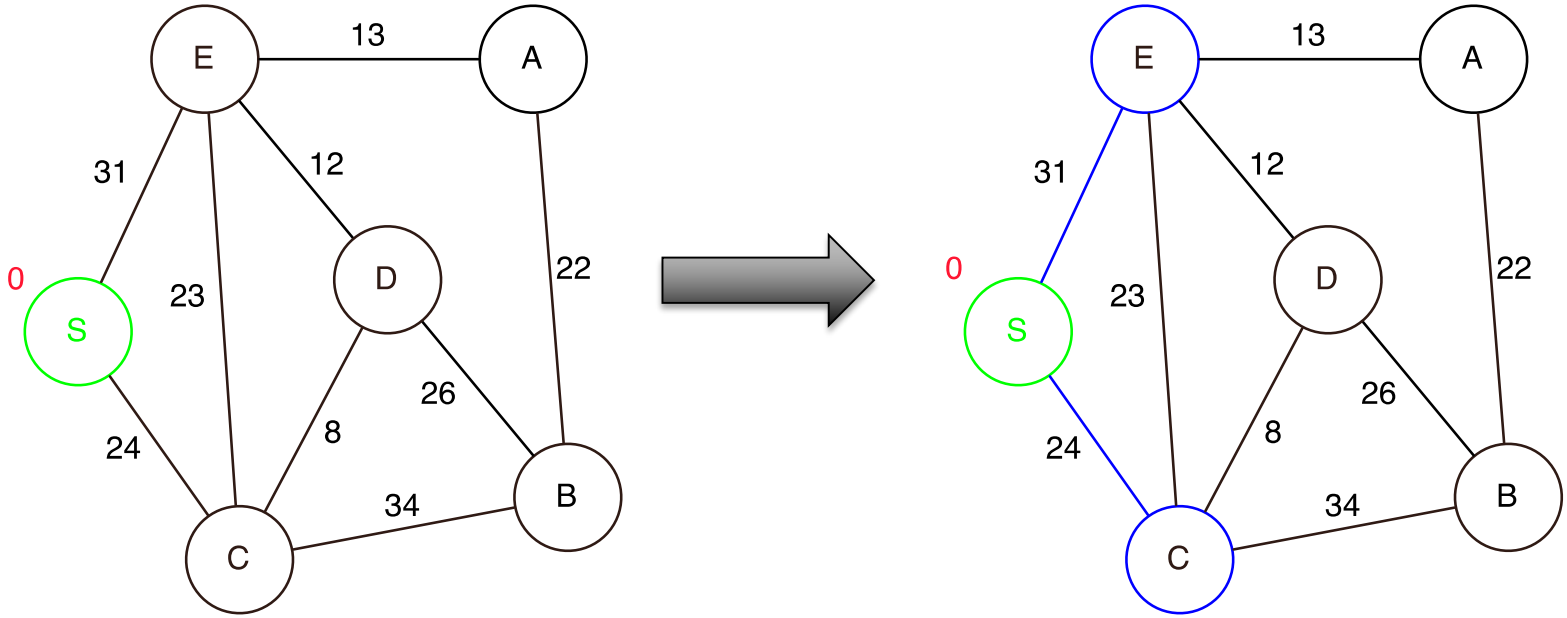
1. Setze die Kosten für Startknoten S = 0
2. Setze die Kosten für alle anderen auf ∞
3. Mache S zum aktuellen Knoten und markiere S als besucht



Dijkstra-Algorithmus: Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

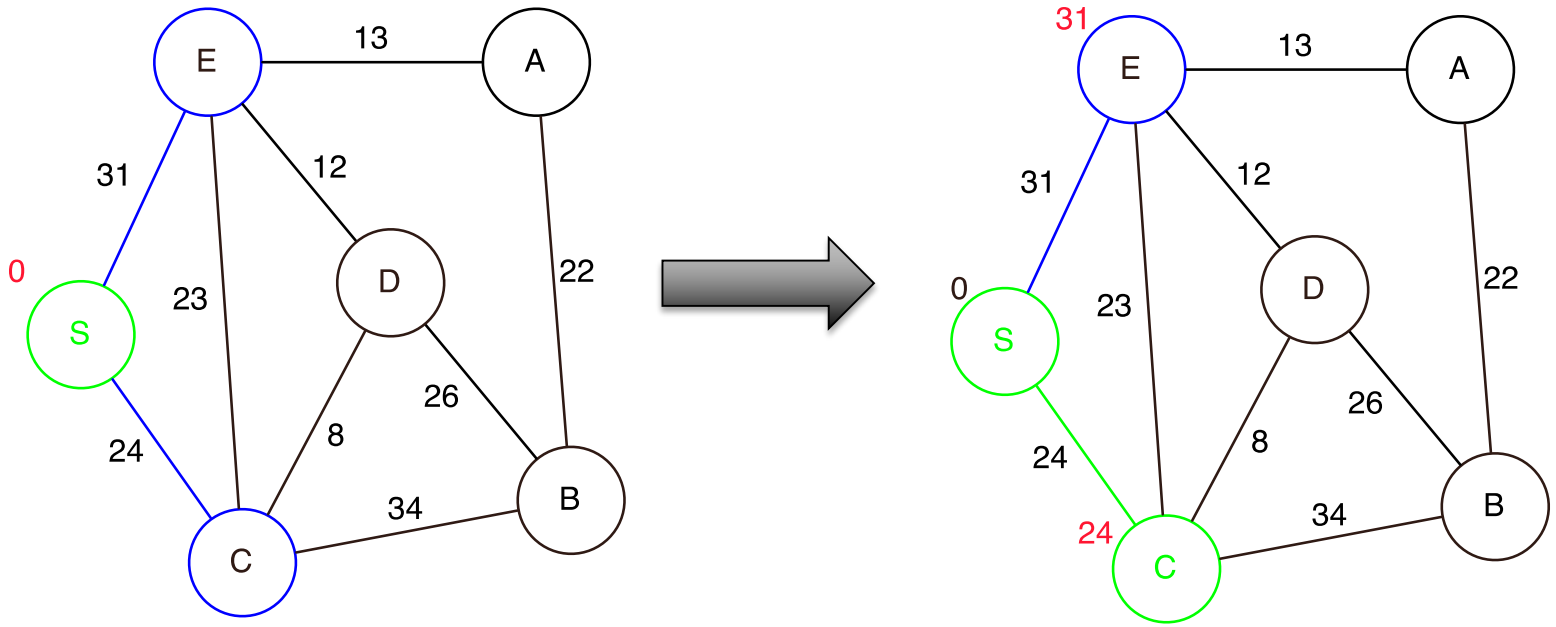
- Aktuell besucht: S
- Betrachte alle von besuchten Knoten erreichbaren Knoten



Dijkstra-Algorithmus: Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

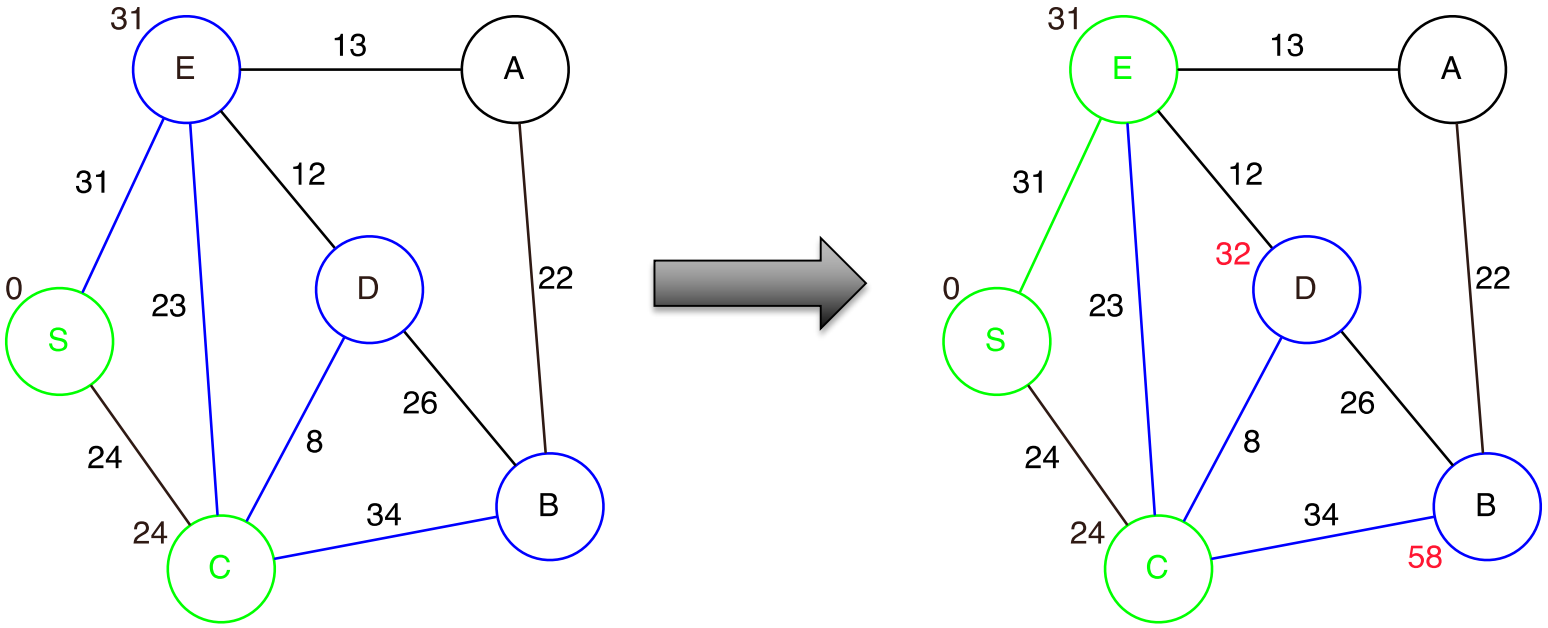
- Aktuell besucht: S
- Falls ein Knoten nun günstiger (oder überhaupt) erreichbar ist, aktualisiere die Kosten.
- Besuche den günstigsten noch nicht besuchten Knoten → C



Dijkstra-Algorithmus: Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

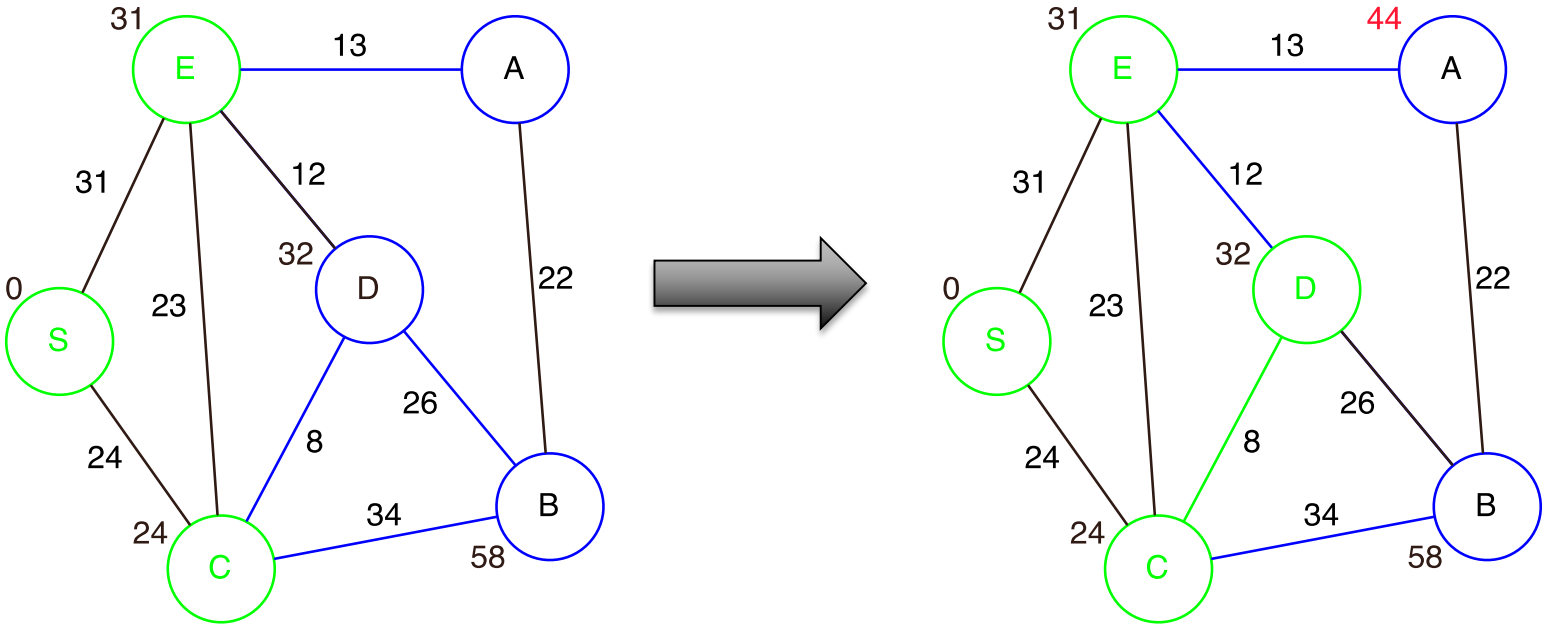
- Aktuell besucht: S, C
- Falls ein Knoten nun günstiger (oder überhaupt) erreichbar ist, aktualisiere die Kosten.
- Besuche den günstigsten noch nicht besuchten Knoten → E



Dijkstra-Algorithmus: Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

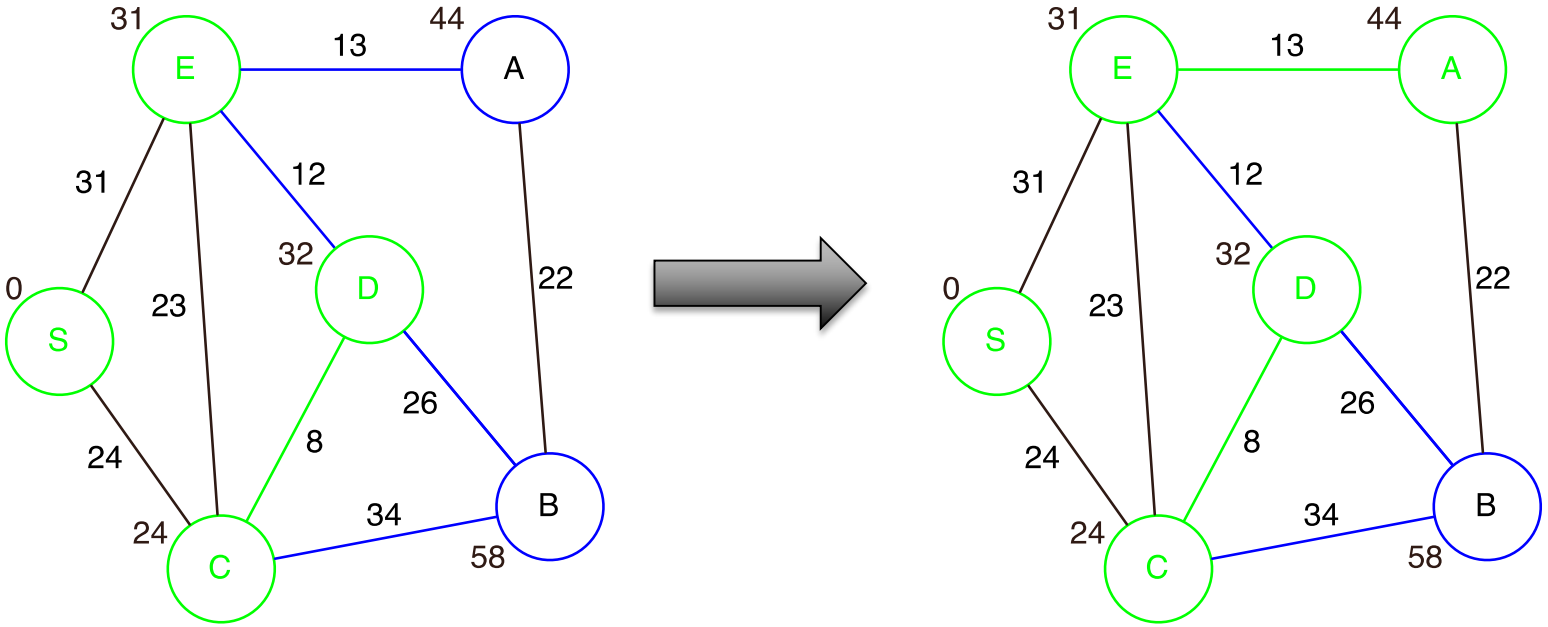
- Aktuell besucht: S, C, E
- Falls ein Knoten nun günstiger (oder überhaupt) erreichbar ist, aktualisiere die Kosten.
- Besuche den günstigsten noch nicht besuchten Knoten → D



Dijkstra-Algorithmus: Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

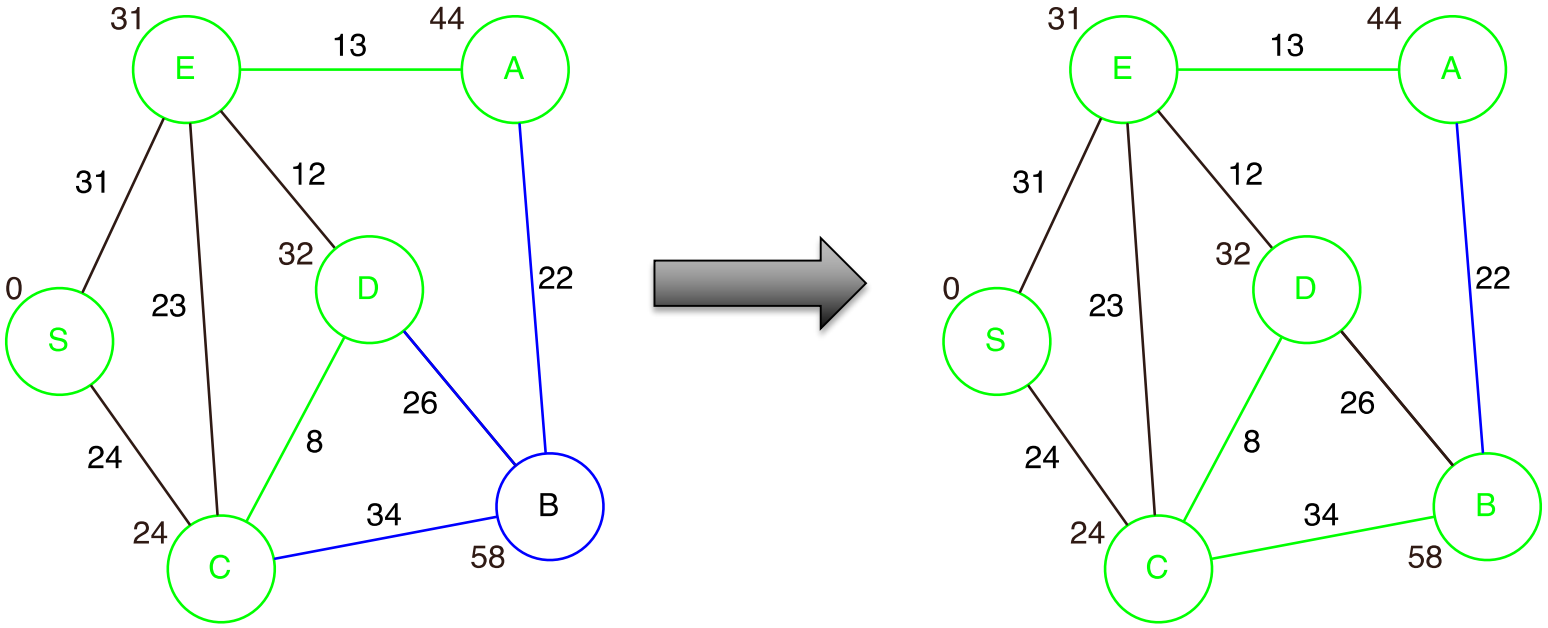
- Aktuell besucht: S, C, E, D
- Falls ein Knoten nun günstiger (oder überhaupt) erreichbar ist, aktualisiere die Kosten.
- Besuche den günstigsten noch nicht besuchten Knoten → A



Dijkstra-Algorithmus: Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

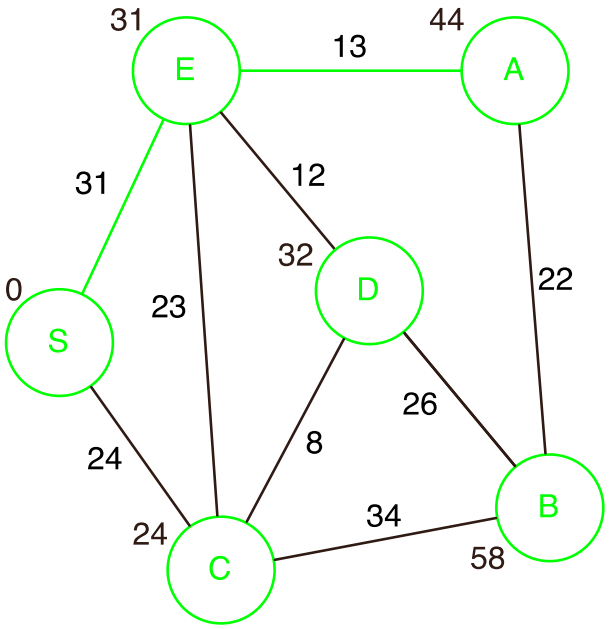
- Aktuell besucht: S, C, E, D, A
- Falls ein Knoten nun günstiger (oder überhaupt) erreichbar ist, aktualisiere die Kosten.
- Besuche den günstigsten noch nicht besuchten Knoten → B



Dijkstra-Algorithmus: Beispiel

TECHNISCHE UNIVERSITÄT DARMSTADT

- Alle Knoten sind besucht
- Der kürzeste Weg von S → A ist S → E → A mit Kosten 44
- Wir „kennen“ auch alle anderen günstigsten Wege



Dijkstra: Aufwand?

TECHNISCHE UNIVERSITÄT DARMSTADT

- Der Rechenaufwand des Verfahrens ist proportional zum *Quadrat der Anzahl Knoten*
  - „In jedem Schritt alle Kosten der Knoten durchlaufen, um den billigsten zu finden“
  - Sei *E*=Anzahl *Kanten („edge“), V*=Anzahl Knoten („vertex“)
  - → Aufwand in Größenordnung $O(V^2 + E)$
- Durch clevere Verwaltung der Nachbarknoten und –kosten geht es schneller
  - → Größenordnung $O(E * \log V)$

- Dijkstra betrachtet „stur“ alle Wege
- Etwas „menschliches Verständnis“ wäre durchaus sinnvoll
  - Der kürzeste Weg von Frankfurt nach München geht sicher nicht über *Hannover*…

Modifikation: A* („A-Stern“)

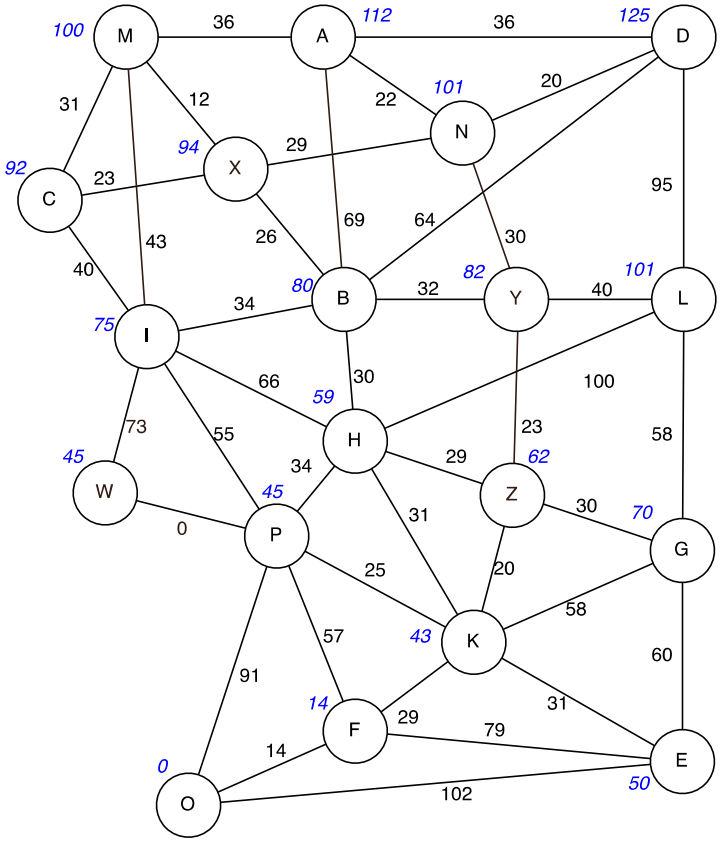
TECHNISCHE UNIVERSITÄT DARMSTADT

- Dijkstra verwendet immer den *unbesuchten* Knoten *mit den geringsten Kosten* als nächsten Knoten
- A* fügt noch eine „Messfunktion“ *f* ein
- Gewählt wird dann der unbesuchte aber schon erreichbare Knoten X, für die f(x) *am kleinsten* ist
- Wie wählt man f(x) bei Navigation...?
  - Summe Wegstrecke + „**Entfernung vom Ziel/Luftlinie**“
- Natürlich müssen diese Daten mit erfasst sein!

Modifikation: A* („A-Stern“)

TECHNISCHE UNIVERSITÄT DARMSTADT

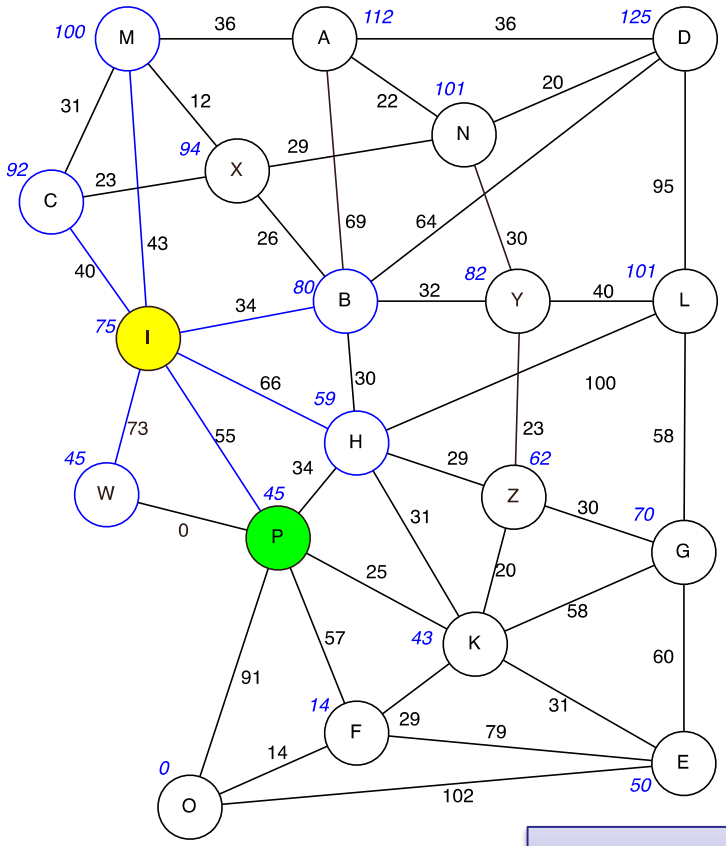
- Die Karte gibt die Entfernung in km *Luftlinie* vom Ziel O an.



Modifikation: A* („A-Stern“)

TECHNISCHE UNIVERSITÄT DARMSTADT

- In der Tabelle steht *Wegstrecke / Entfernung Luftlinie*



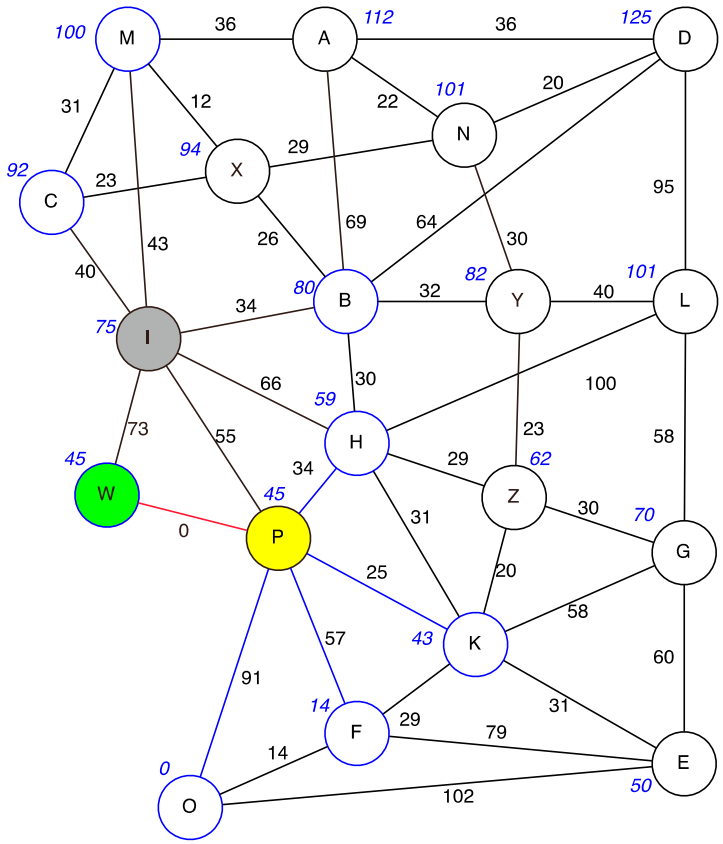
Knoten	Wert	Knoten	Wert
A	∞	L	∞
B	*34/80*	M	*43/100*
C	*40/92*	N	∞
D	∞	O	∞
E	∞	P	*55/45*
F	∞	W	*73/45*
G	∞	X	∞
H	*66/59*	Y	∞
I	**0 / 75**	Z	∞
K	∞		

gewählt

aktuell

besucht

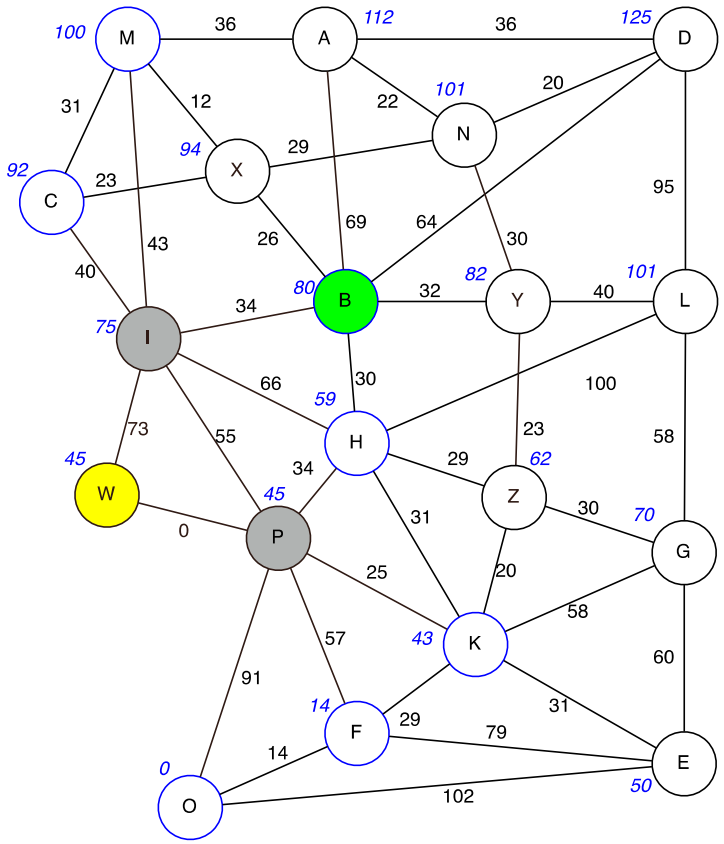
Modifikation: A* („A-Stern“)



Knoten	Wert	Knoten	Wert
A	∞	L	∞
B	34/80	M	43/100
C	40/92	N	∞
D	∞	O	*146/0*
E	∞	**P**	**55 / 45**
F	*112/14*	W	*55/45*
G	∞	X	∞
H	66/59	Y	∞
I	**0 / 75**	Z	∞
K	*80/43*		

Modifikation: A* („A-Stern“)

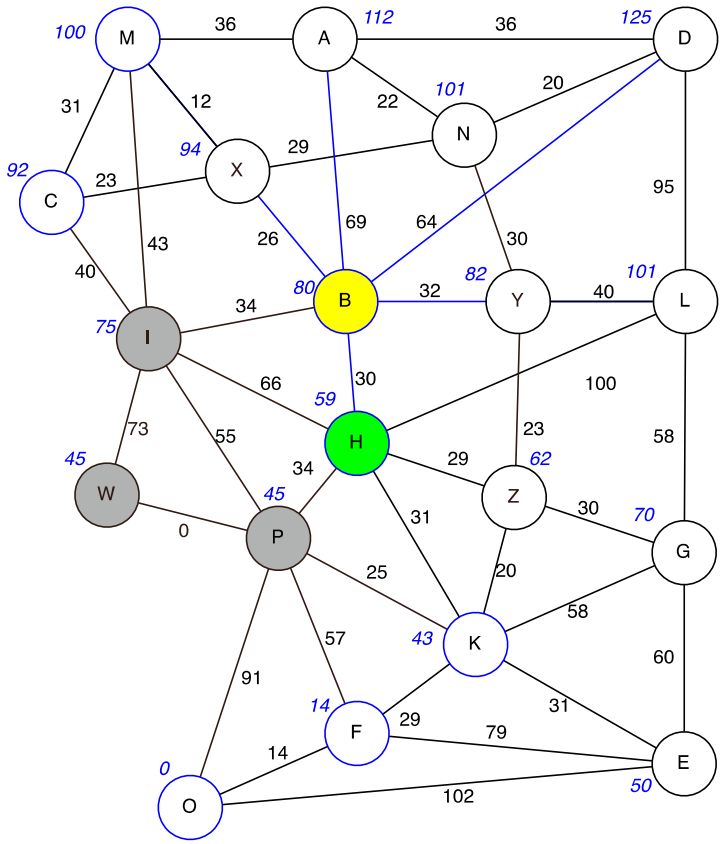
TECHNISCHE UNIVERSITÄT DARMSTADT



Knoten	Wert	Knoten	Wert
A	∞	L	∞
B	34/80	M	43/100
C	40/92	N	∞
D	∞	O	146/0
E	∞	**P**	**55 / 45**
F	112/14	**W**	**55 / 45**
G	∞	X	∞
H	66/59	Y	∞
I	**0 / 75**	Z	∞
K	80/43		

Modifikation: A* („A-Stern“)

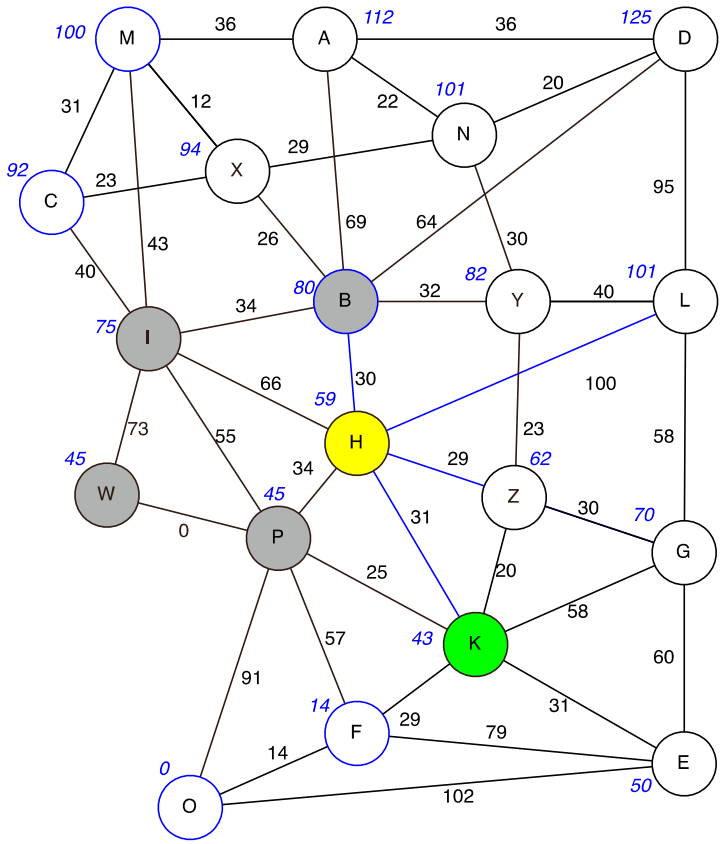
TECHNISCHE UNIVERSITÄT DARMSTADT



Knoten	Wert	Knoten	Wert
A	*103/112*	L	∞
B	**34 / 80**	M	43/100
C	40/92	N	∞
D	*98/125*	O	146/0
E	∞	**P**	**55 / 45**
F	112/14	**W**	**55 / 45**
G	∞	X	*60/94*
H	*64/59*	Y	*66/82*
I	**0 / 75**	Z	∞
K	80/43		

Modifikation: A* („A-Stern“)

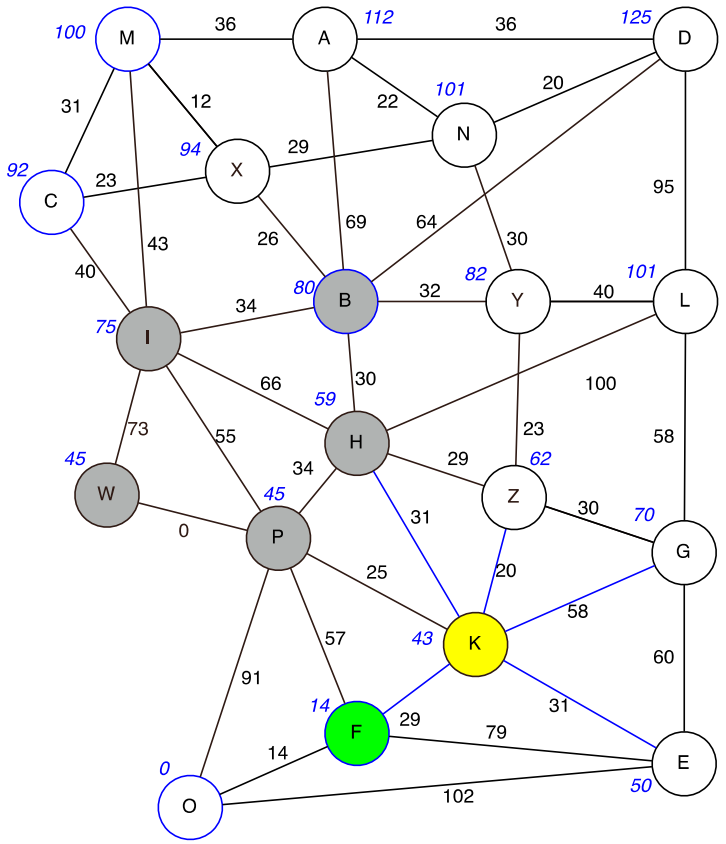
TECHNISCHE UNIVERSITÄT DARMSTADT



Knoten	Wert	Knoten	Wert
A	103/112	L	*164/101*
B	**34 / 80**	M	43/100
C	40/92	N	∞
D	98/125	O	146/0
E	∞	**P**	**55 / 45**
F	112/14	**W**	**55 / 45**
G	∞	X	60/94
H	**64 / 59**	Y	66/82
I	**0 / 75**	Z	*95/43*
K	80/43		

Modifikation: A* („A-Stern“)

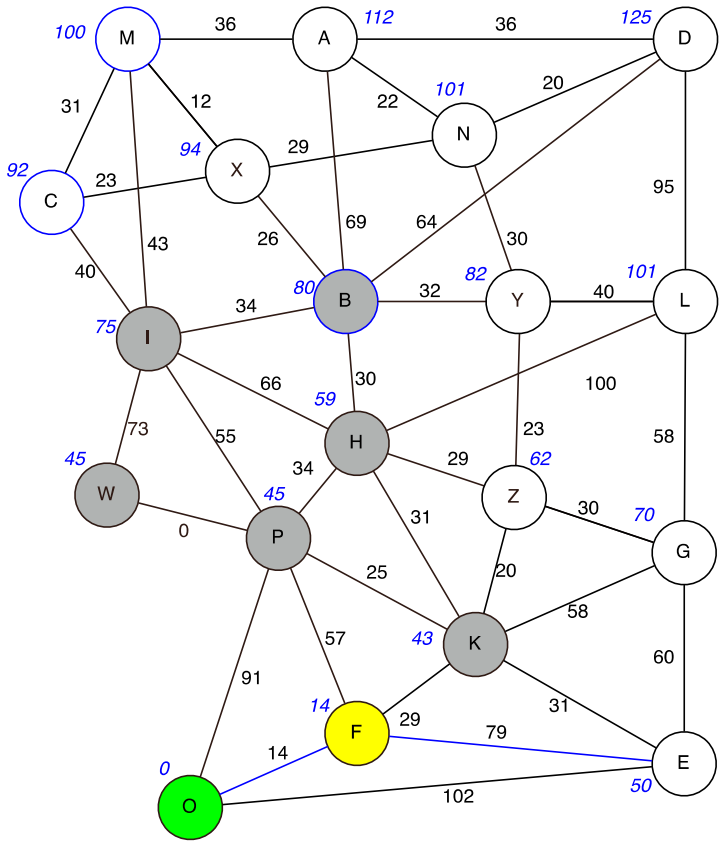
TECHNISCHE UNIVERSITÄT DARMSTADT



Knoten	Wert	Knoten	Wert
A	103/112	L	164/101
B	**34 / 80**	M	43/100
C	40/92	N	∞
D	98/125	O	146/0
E	*111/50*	**P**	**55 / 45**
F	*109/14*	**W**	**55 / 45**
G	*138/70*	X	60/94
H	**64 / 59**	Y	66/82
I	**0 / 75**	Z	95/43
K	**80 / 43**		

Modifikation: A* („A-Stern“)

TECHNISCHE UNIVERSITÄT DARMSTADT

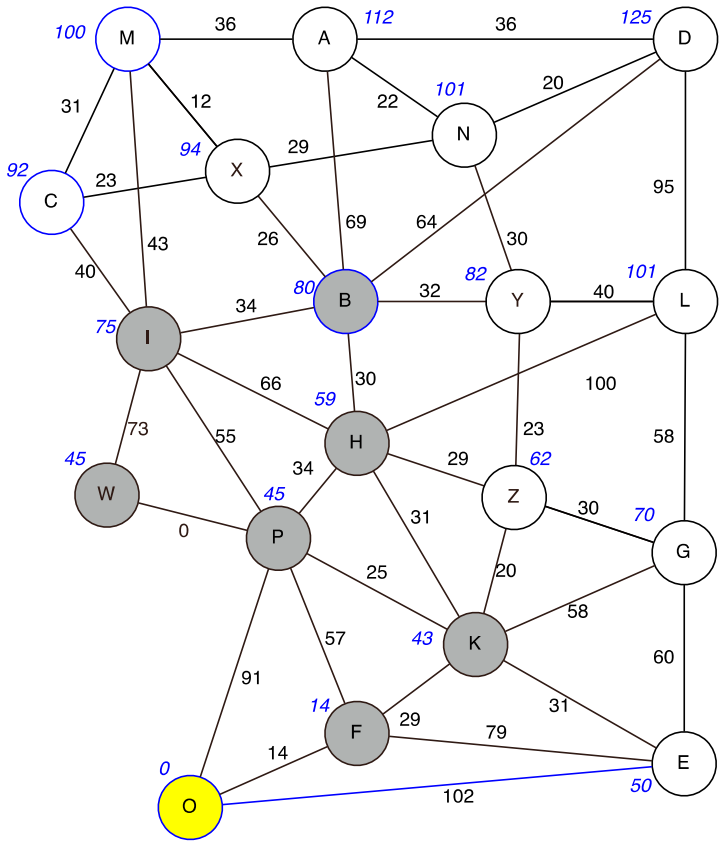


Knoten	Wert	Knoten	Wert
A	103/112	L	164/101
B	**34 / 80**	M	43/100
C	40/92	N	∞
D	98/125	O	*123/0*
E	111/50	**P**	**55 / 45**
F	109/14	**W**	**55 / 45**
G	138/70	X	60/94
H	**64 / 59**	Y	66/82
I	**0 / 75**	Z	95/43
K	**80 / 43**		

Modifikation: A* („A-Stern“)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ziel O ist billigster Knoten → Ende (vorzeitiger Abbruch!)

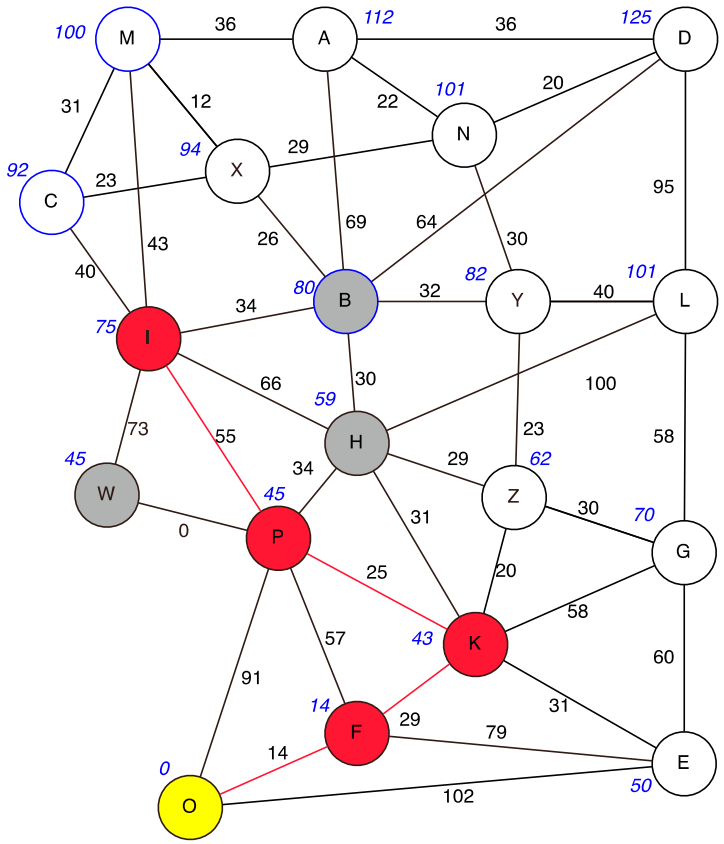


Knoten	Wert	Knoten	Wert
A	103/112	L	164/101
B	**34 / 80**	M	43/100
C	40/92	N	∞
D	98/125	**O**	**123 / 0**
E	111/50	**P**	**55 / 45**
F	**109 / 14**	**W**	**55 / 45**
G	138/70	X	60/94
H	**64 / 59**	Y	66/82
I	**0 / 75**	Z	95/43
K	**80 / 43**		

Modifikation: A* („A-Stern“)

TECHNISCHE UNIVERSITÄT DARMSTADT

- Günstigster Weg ist I → P → K → F → O mit Kosten 123



Knoten	Wert	Knoten	Wert
A	103/112	L	164/101
B	**34 / 80**	M	43/100
C	40/92	N	∞
D	98/125	**O**	**123 / 0**
E	111/50	**P**	**55 / 45**
F	**109 / 14**	**W**	**55 / 45**
G	138/70	X	60/94
H	**64 / 59**	Y	66/82
I	**0 / 75**	Z	95/43
K	**80 / 43**		

Bewertung A*; Varianten

TECHNISCHE UNIVERSITÄT DARMSTADT

- Ist A-Stern wirklich Optimierungsalgorithmus oder Heuristik?
- A* liefert immer das Optimum
  - Jeder Knoten mit potenzieller Verbesserung als Zwischenziel – also „Strecke + Luftlinie < Strecke zum Ziel“ – wird auf dem Weg besucht

- Und bei „*schnellster* Strecke“ o. Ä.?
  - Statt Luftlinie andere Kriterien hinzuziehen
    - „Geschwindigkeit auf *Straßentyp*“ (Autobahn vs. Ortschaft)
    - „Stauinformationen → Geschwindigkeit anpassen“
    - „Sehenswürdigkeiten“ für die „schönste“ Strecke
  - Es muss nur die Hilfsfunktion *f(x)* angepasst werden, *nicht* A*