Funktionsprinzipien und Anwendungen von Algorithmen zur Pfadplanung

Mohammed Salih Mezraoui
David Gruber
Marius Müller

Informatik
Hauptcampus

HOCHSCHULE TRIER

Einleitung

- **Pfadsuche:** Beschreibt in der Informatik den Algorithmen gestützten Prozess der Suche von einem Start- zu einem Zielpunkt in einem Netzwerk, Graph oder Gitter
- **Uninformierte:** „Einfache" Pfadsuchalgorithmen die die Basis für neuere/bessere Ansätze bilden
- **Optimierungen:** Viele der gezeigten Algorithmen wurden später durch bessere/optimiertere Algorithmen ersetzt, welche in heutigen Projekten nun immer häufiger eingesetzt werden (z.B. der A*-Algorithmus)
- **Anwendung:** In vielen Bereichen wie Internet-Routing, Computerspielen, Robotik, Autonomes Fahren und Navigationssystemen relevant

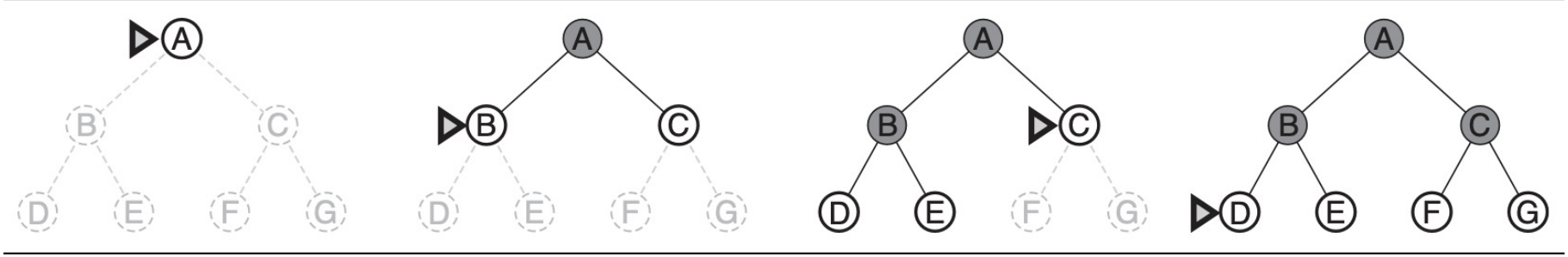
Uninformierte Suche

Uninformierte Algorithmen (oder auch „blinde" Algorithmen) greifen bei ihrer Suche auf keine zusätzlichen Informationen wie z.B. Gewichtungen.

Alle Pfadsuchalgorithmen werden nach den 4 folgenden Hauptkriterien bewertet:

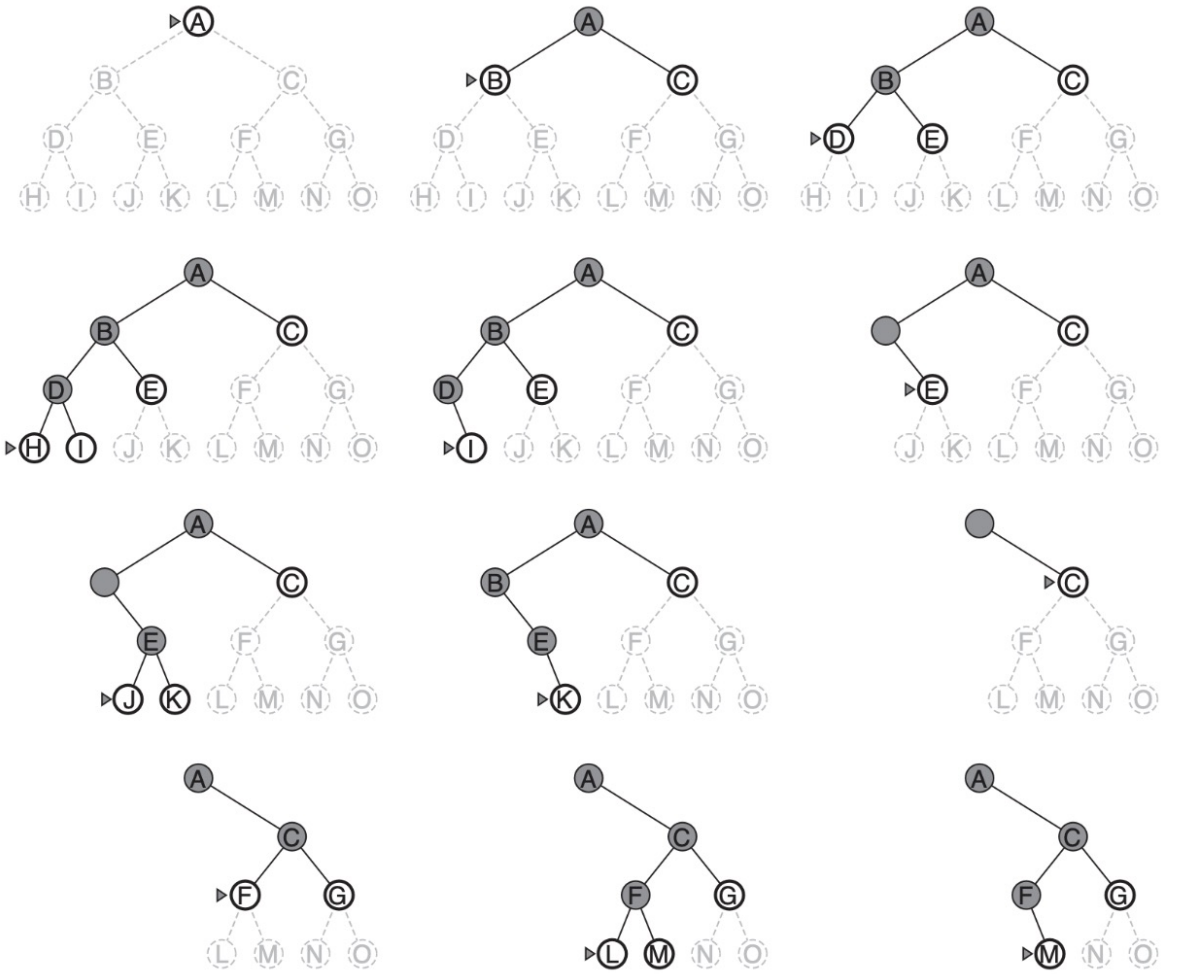
- **Vollständigkeit**: Liefert der Algorithmus immer eine Lösung, wenn es eine Lösung gibt?
- **Optimalität**: Liefert der Algorithmus einen optimalen Weg?
- **Zeitkomplexität**: Wie lange dauert es bis der Algorithmus einen Weg gefunden hat?
- **Raumkomplexität**: Wie viel Speicher wird benötigt um die Lösung zu finden?

- Knoten werden erst innerhalb der aktuellen Ebene besucht
- Erst dann wird eine neue Ebene besucht



Uninformierte Suche
Tiefensuche

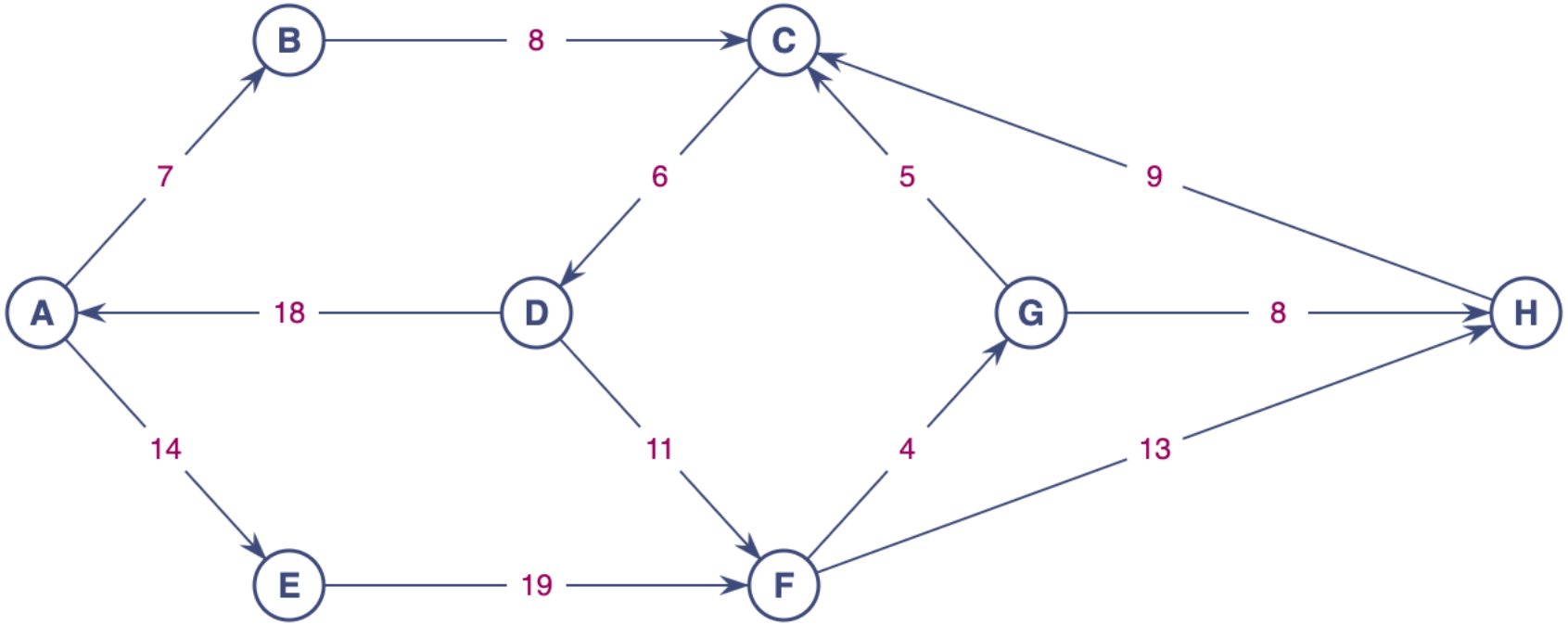
- Knoten werden Ast für Ast zuerst in die Tiefe expandiert bzw. besucht
- erst dann wird der nächste Ast beginnend von der 1. Ebene besucht



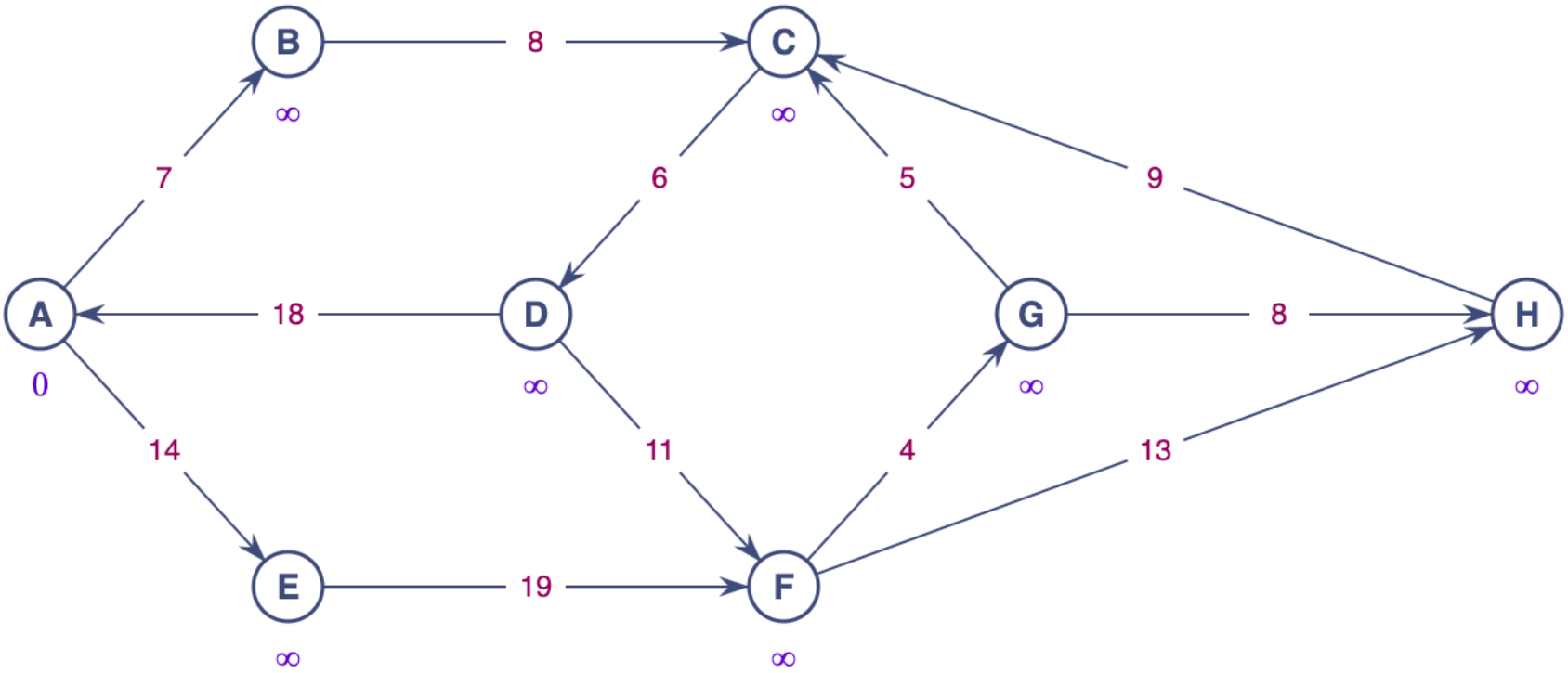
Dijkstra-Algorithmus

- Bekannter Algorithmus auf dem Gebiet der optimalen Pfadwahl
- Wird verwendet, um den kürzesten Pfad von einem Startpunkt in einem Graphen zu einem Zielpunkt zu finden.
- Die Trefferquote liegt bei 100%
- Der Algorithmus besucht die verbleibenden unerwünschten Knoten nicht, wenn der beabsichtigte Zielknoten erreicht ist
- Kann nicht mit negativen Kanten umgehen.

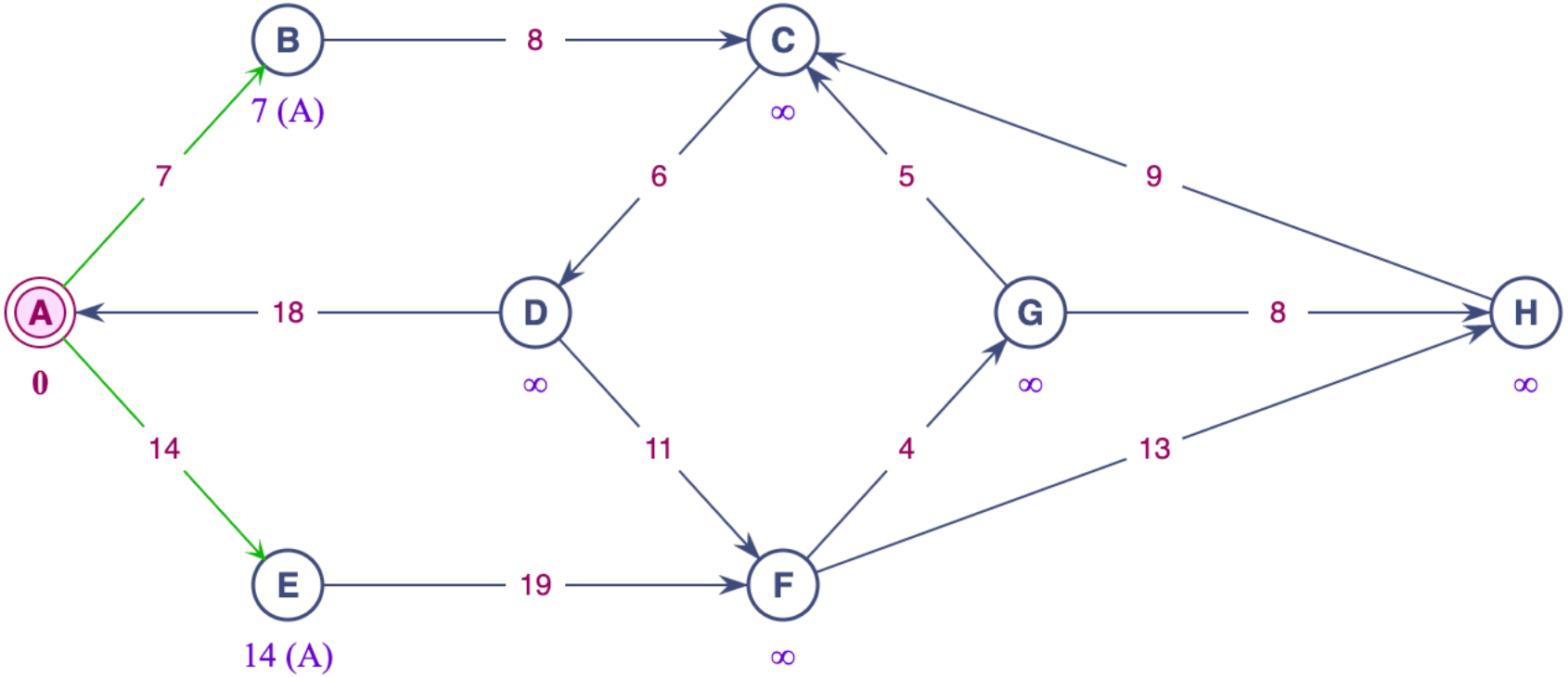
Dijkstra-Algorithmus Beispiel



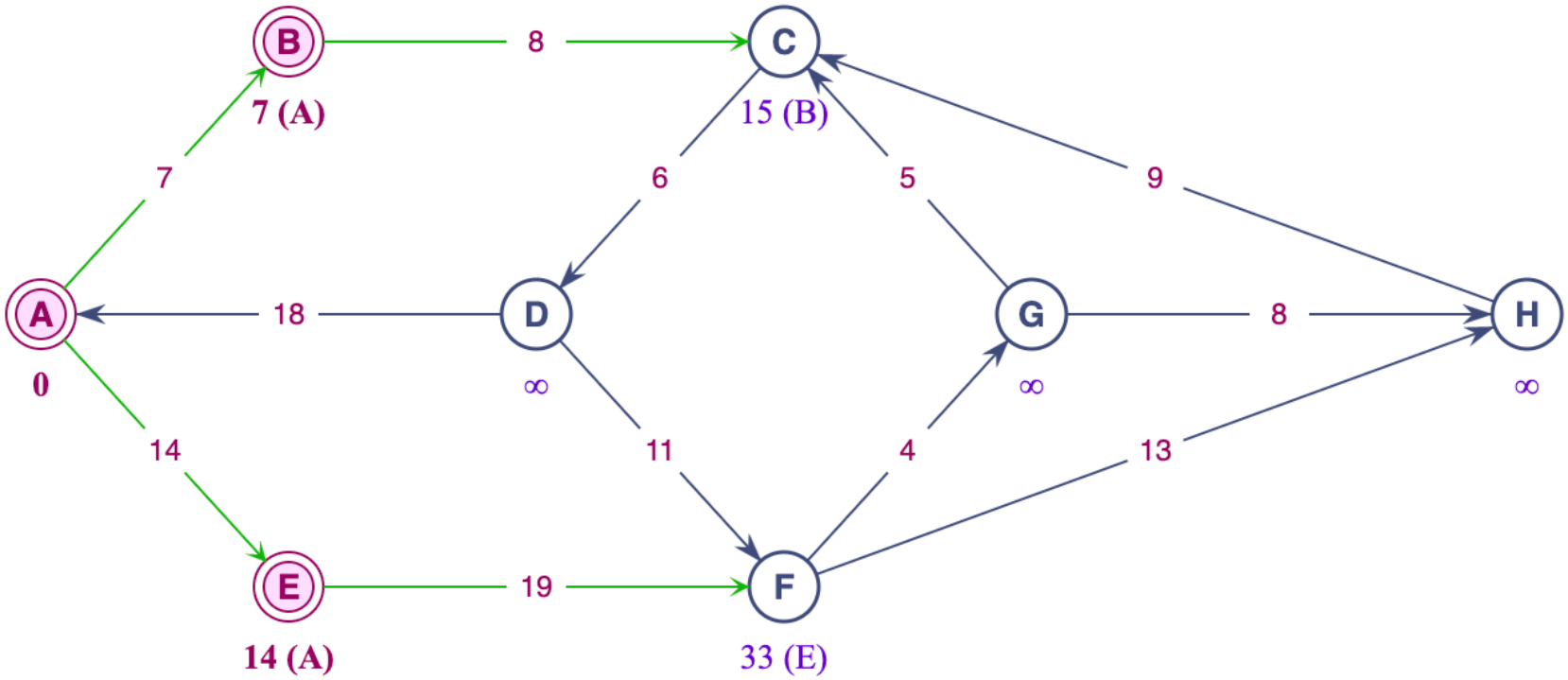
Dijkstra-Algorithmus Beispiel



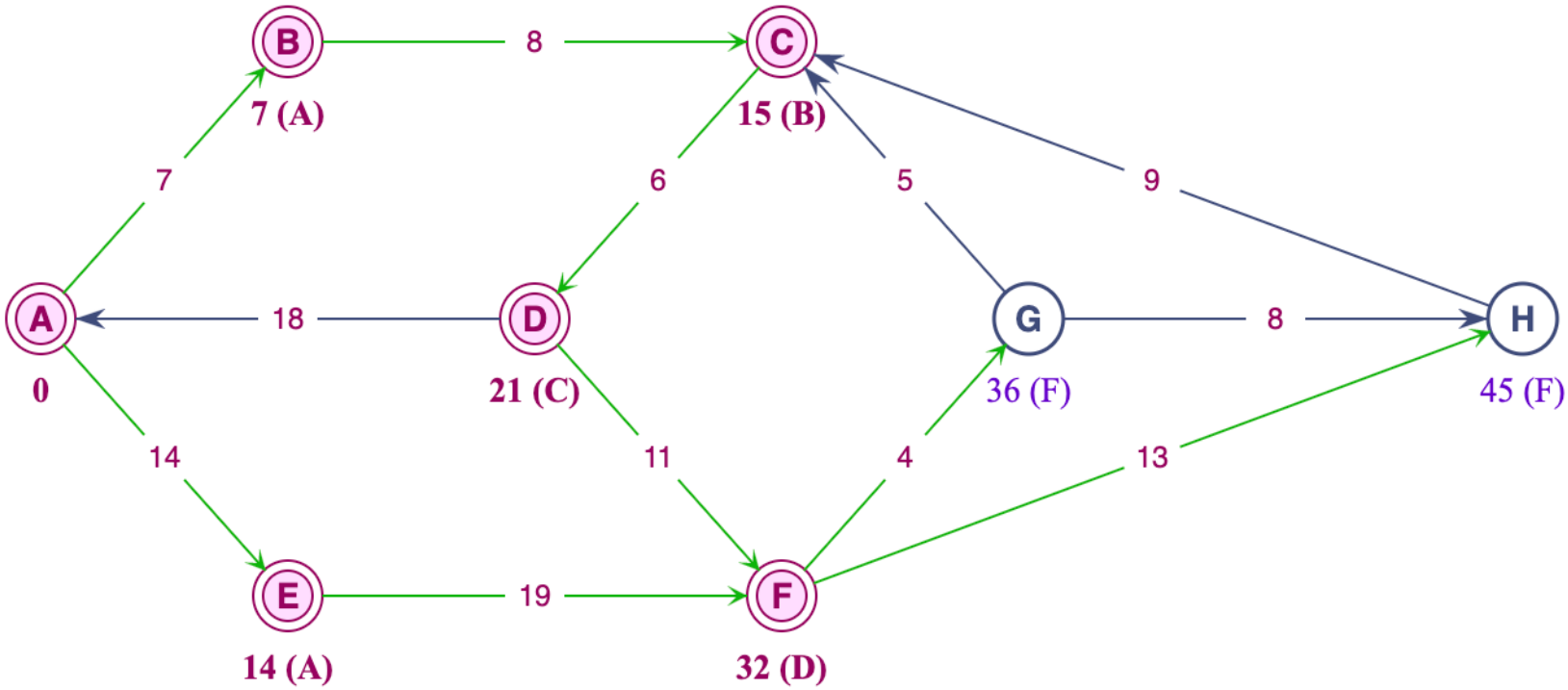
Dijkstra-Algorithmus Beispiel



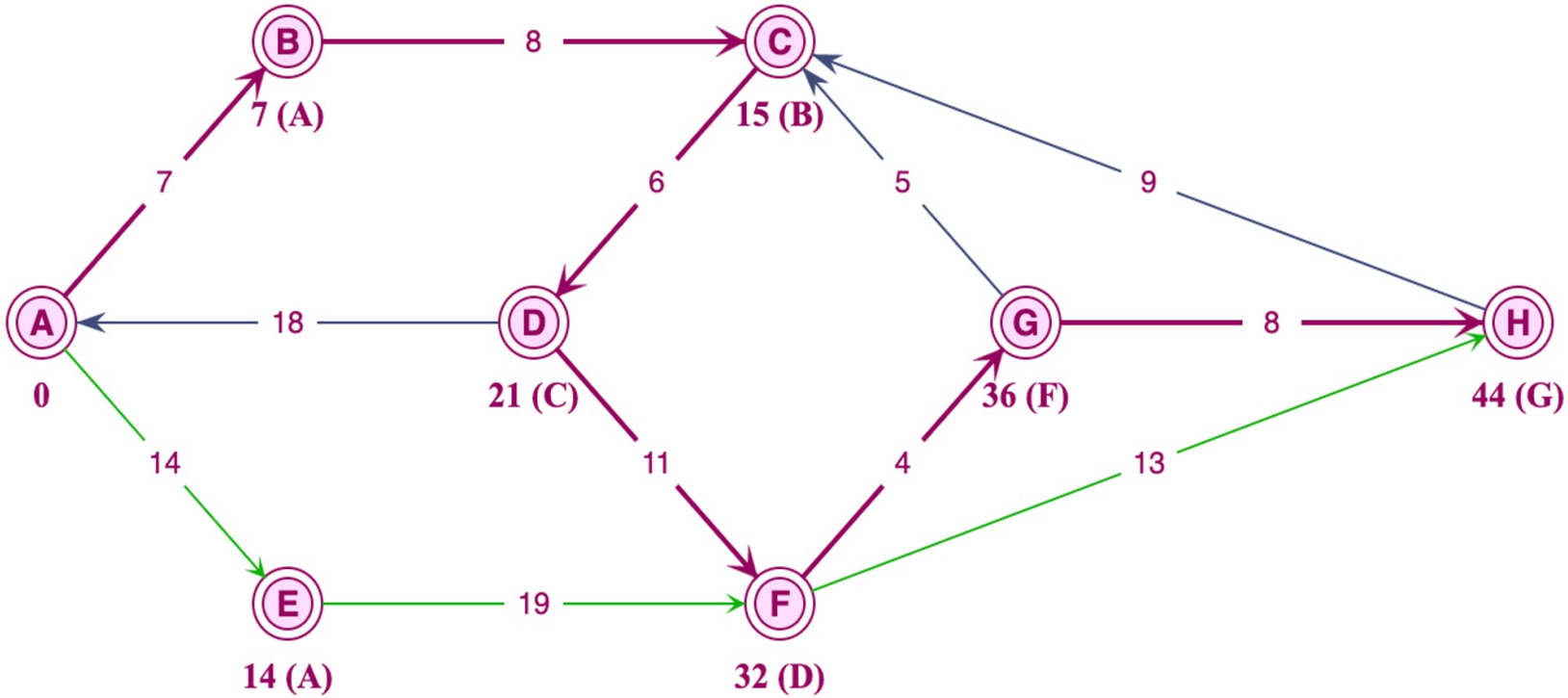
Dijkstra-Algorithmus Beispiel



Dijkstra-Algorithmus Beispiel



Dijkstra-Algorithmus Beispiel

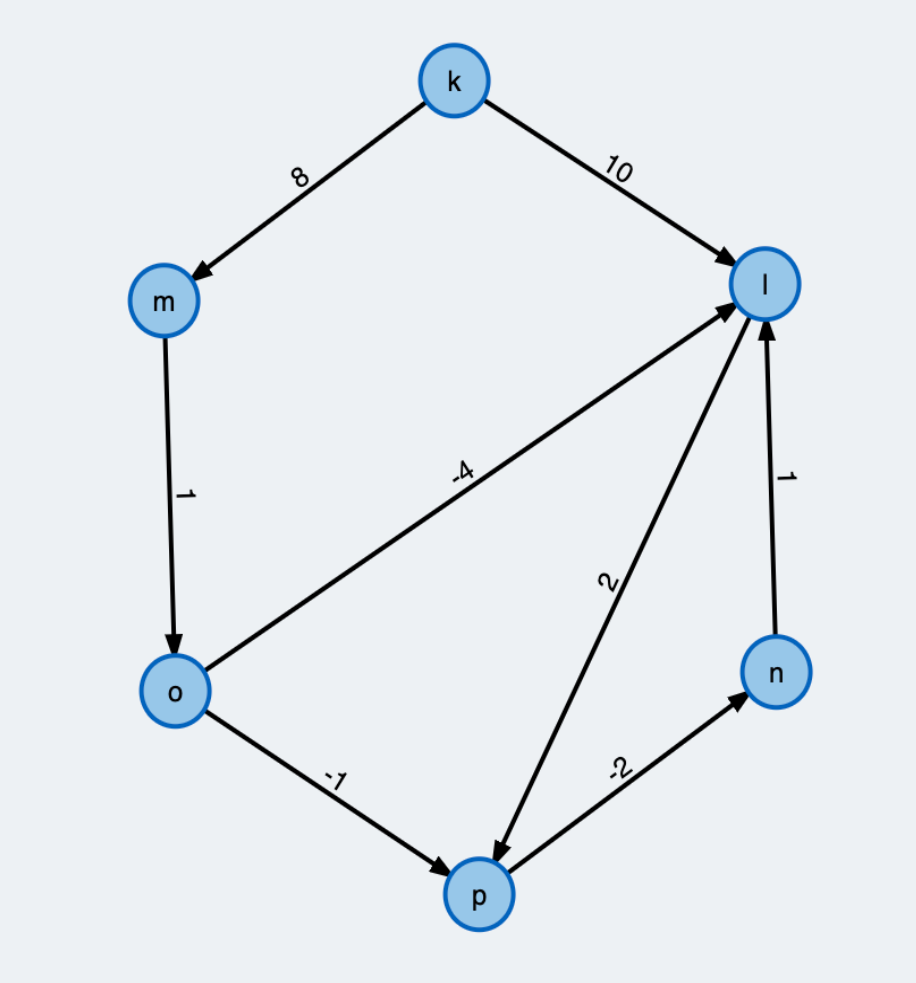


Bellman-Ford-Algorithmus

- Graphen-Suchalgorithmus zur Ermittlung des kürzesten Pfades.
- Geeignet für Graphen mit negativen Kanten
- Der Bellman-Ford-Algorithmus kann gerichtete und ungerichtete Graphen mit nicht-negativen Gewichten verarbeiten.

Bellman-Ford Beispiel

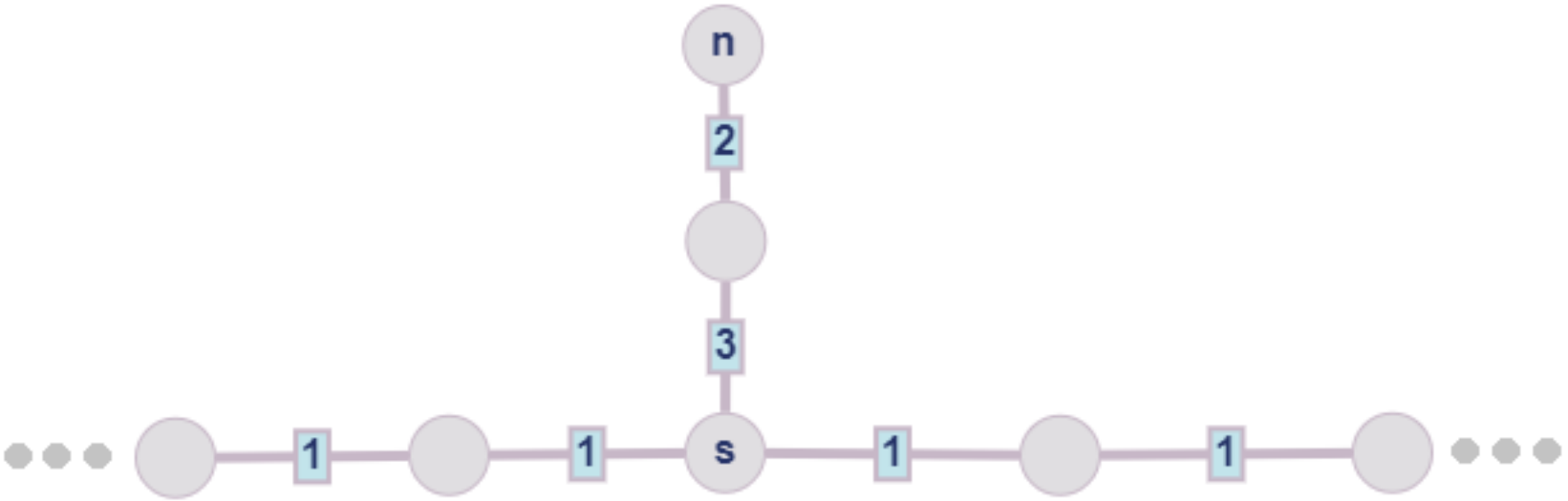
K	I	N	P	O	M
0	∞	∞	∞	∞	∞
0	10	10	12	9	8
0	5	10	8	9	8
0	5	5	7	9	8
0	5	5	7	9	8



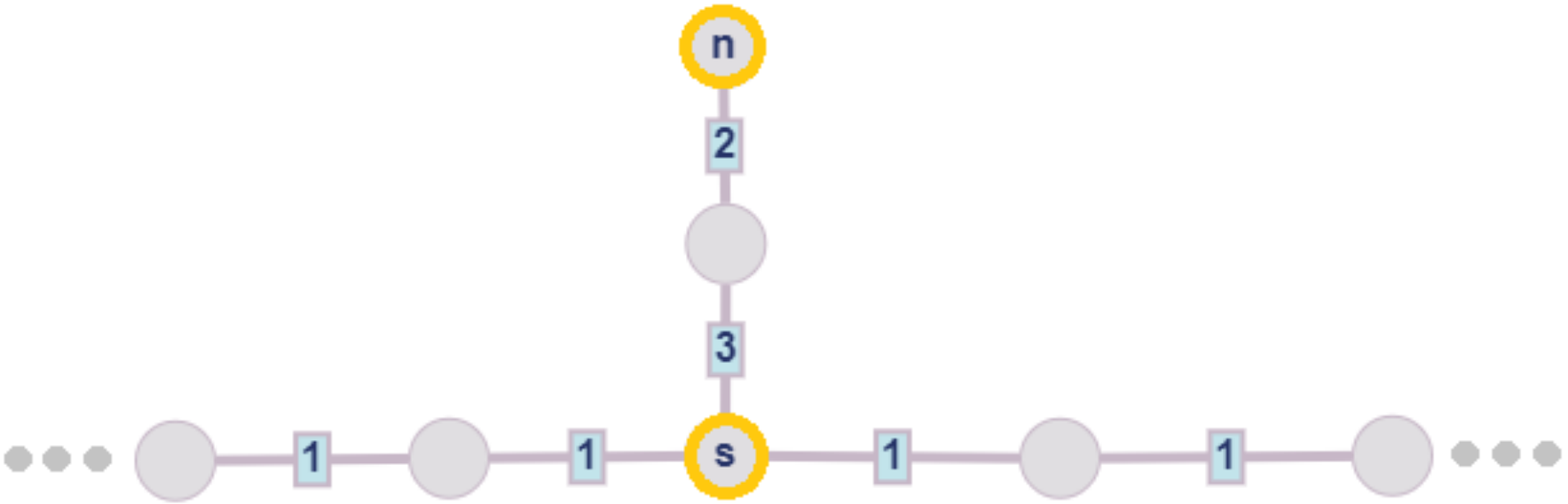
Quelle: [Eigene Darstellung]

Optimierung

Problem: Dijkstra

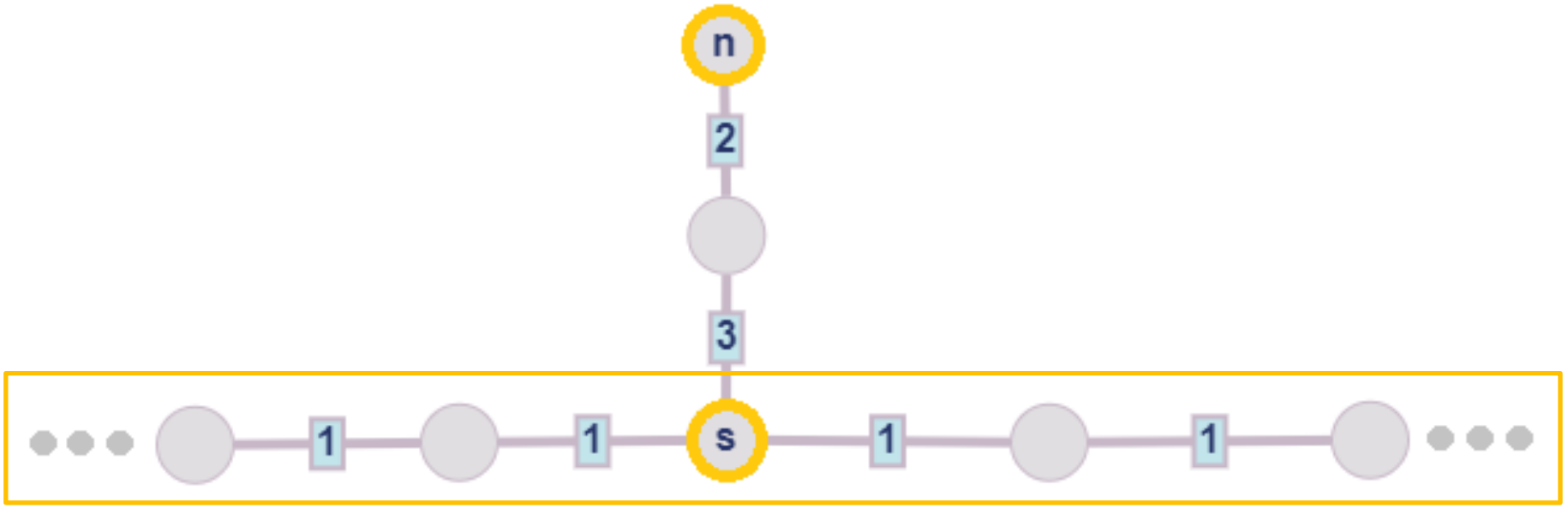


Quelle: [Eigene Darstellung]



Quelle: [Eigene Darstellung]

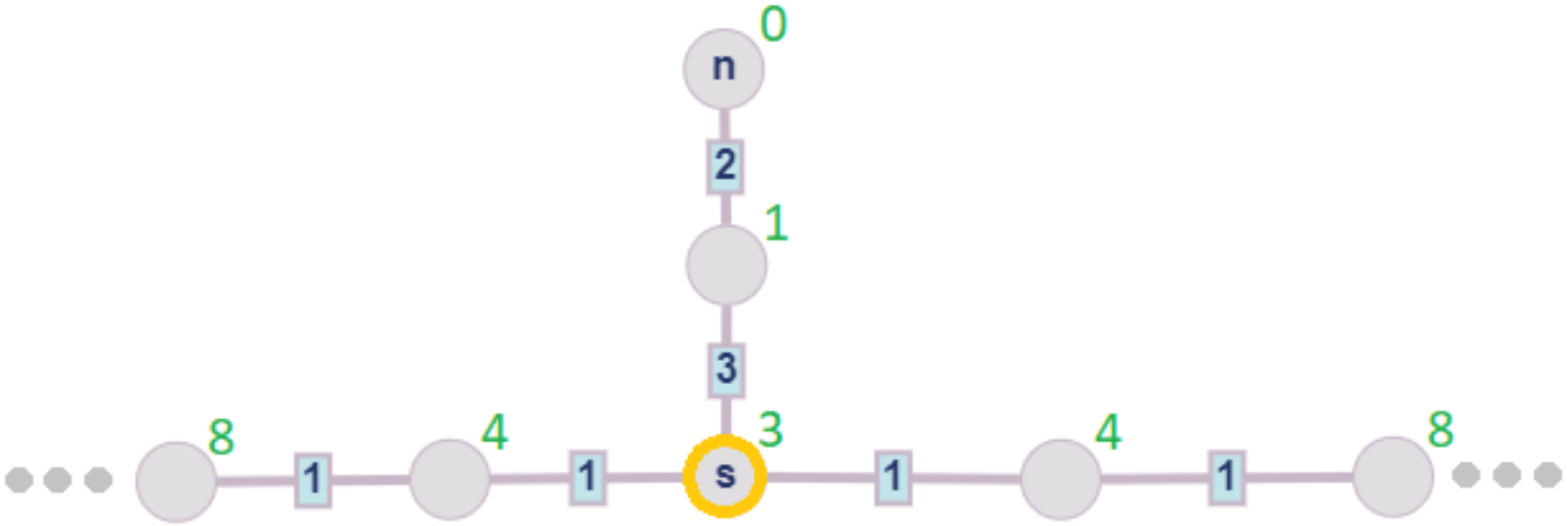
Problem: Dijkstra

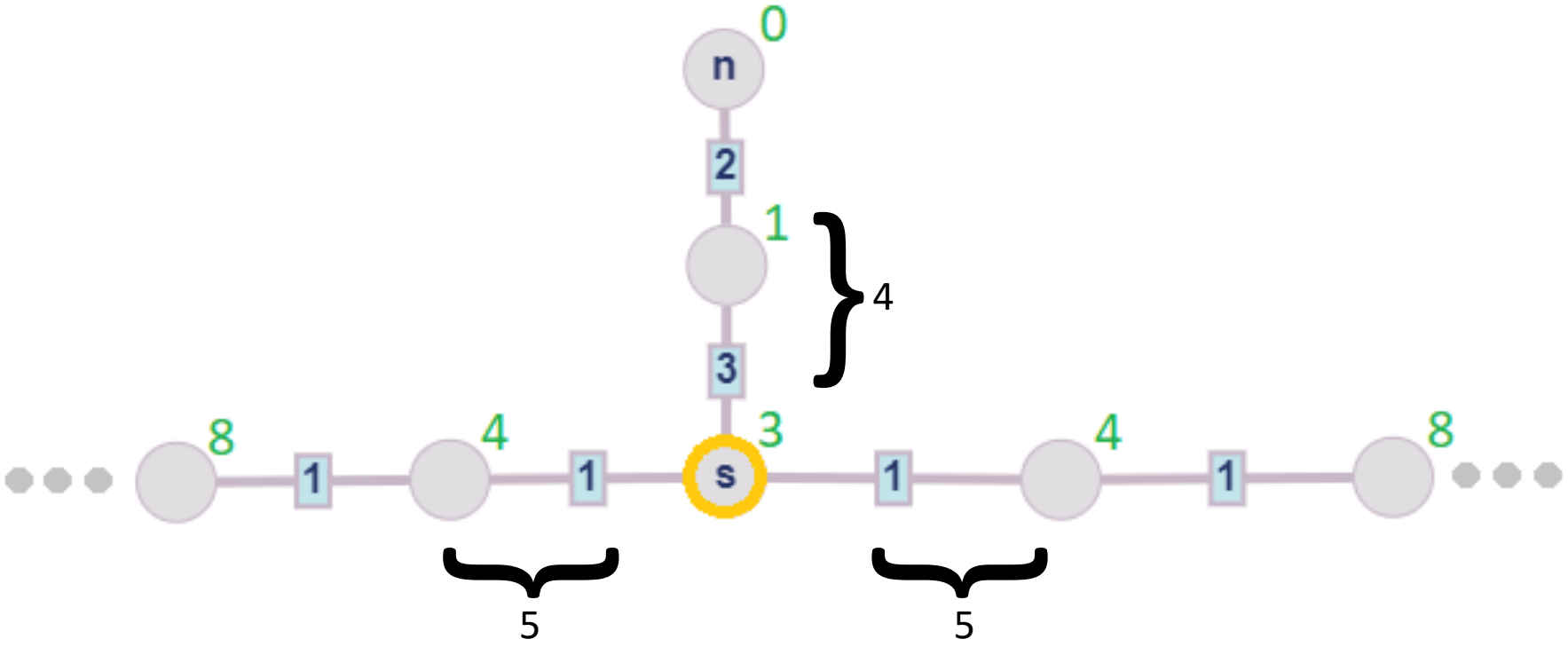


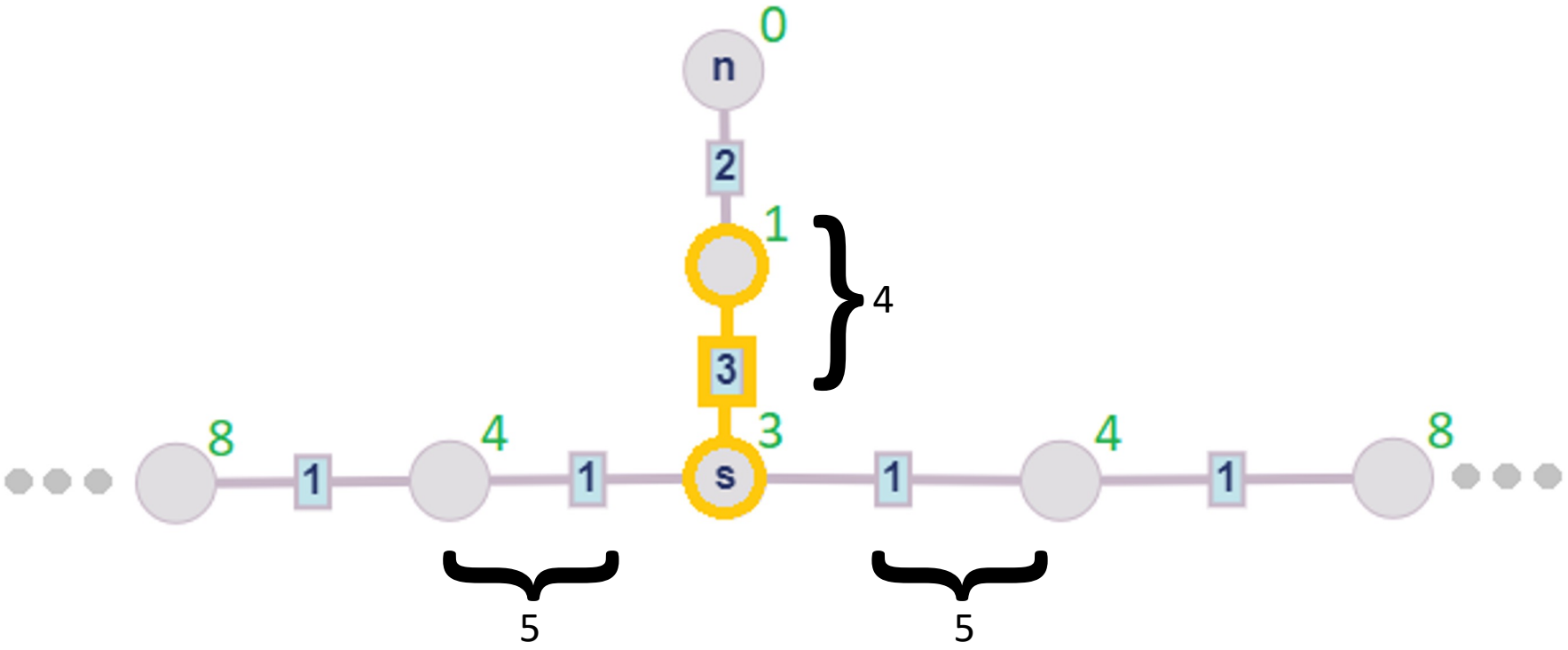
Quelle: [Eigene Darstellung]

A*-Suche

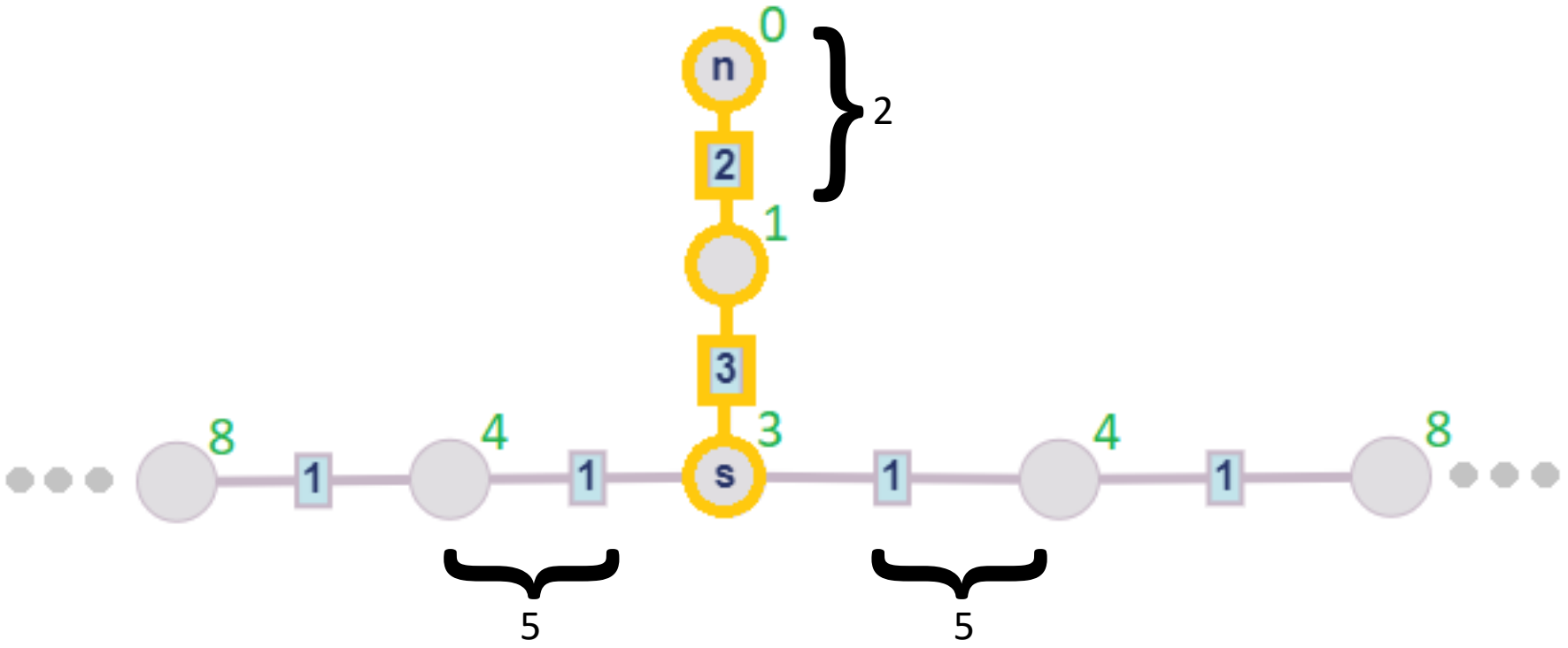
- Informierte Suche
  → Berücksichtigung von problemspezifischem Wissen
- Erweitert Dijkstra-Algorithmus um heuristische Funktion
- Heuristik berechnet die zu erwartenden Kosten des optimalen Pfades von einem Knoten *s* zu einem Zielknoten *n*
- Mögliche Heuristik beim Straßennetz: Luftlinie


0
n
2
1
3
8
1
4
1
s
3
1
4
1
8


0
n
2
1
4
3
8
4
3
4
8
1
1
s
1
1
5
5



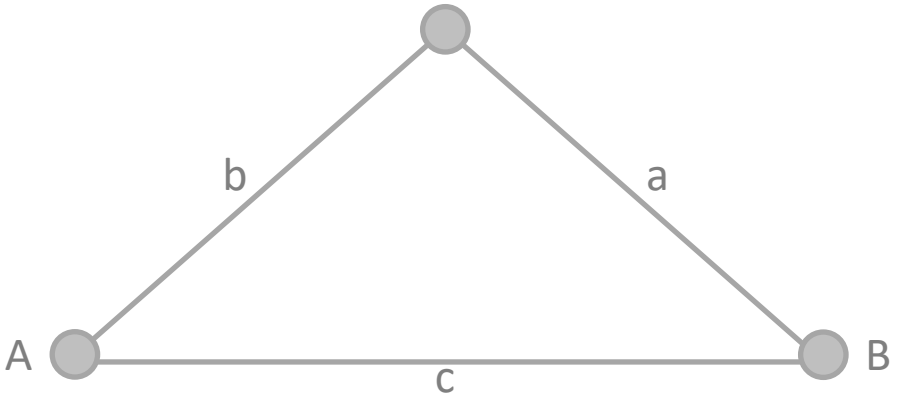
Quelle: [Eigene Darstellung]


0
n
2
2
1
3
8
1
4
1
3
s
1
4
1
8
5
5

Vorverarbeitung

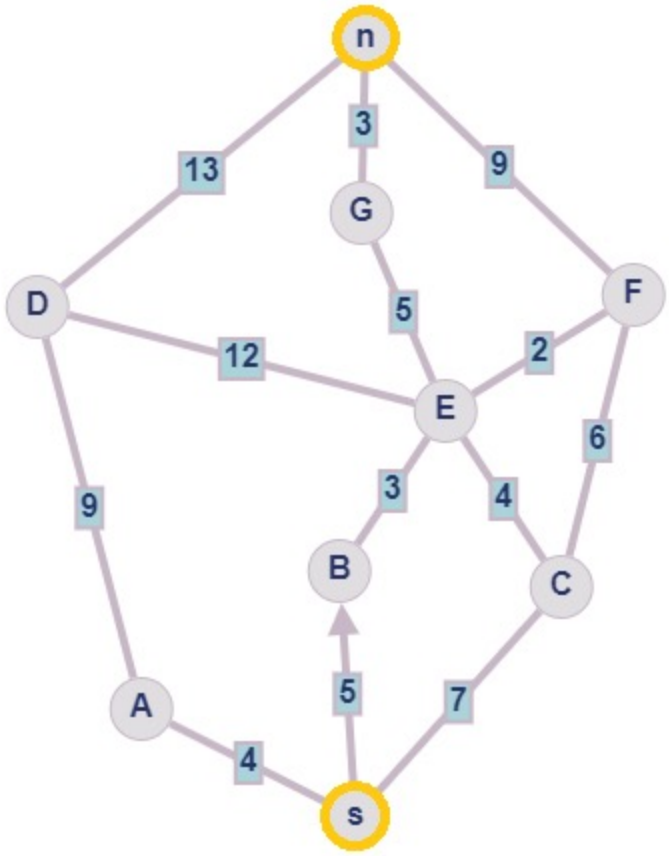
ALT-Algorithmen

- **ALT** ist ein Akronym für **A***-Search, **L**andmarks und **T**riangle Inequality
- In Vorverarbeitungs-Phase konstante Anzahl von Landmarken im Graphen auswählen
- Von Landmarken kürzesten Pfad zu allen anderen Knoten bestimmen
- Distanz und Dreiecksungleichung als Heuristik

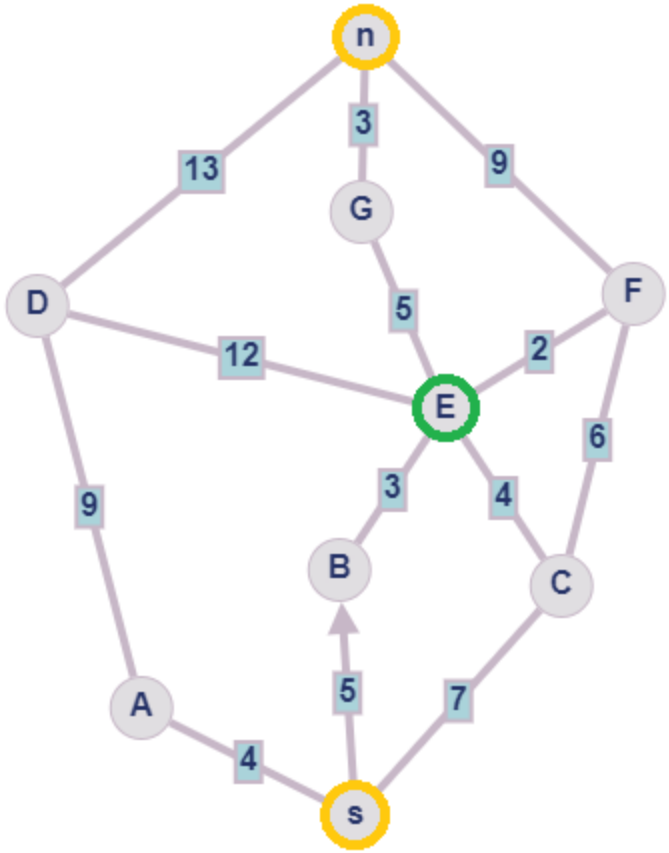


$$|a - b| \leq c$$

ALT-Algorithmus

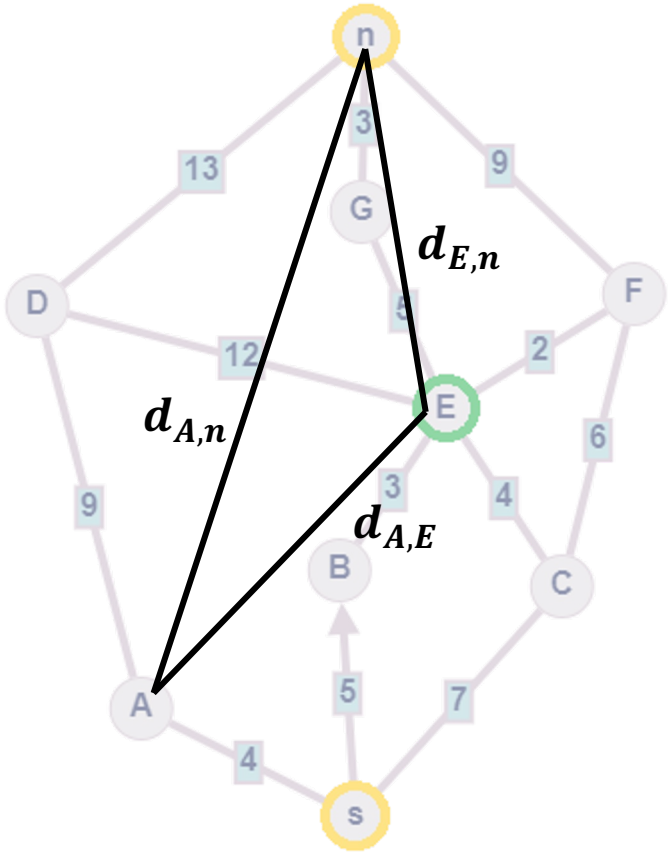


Quelle: [Eigene Darstellung]



Quelle: [Eigene Darstellung]

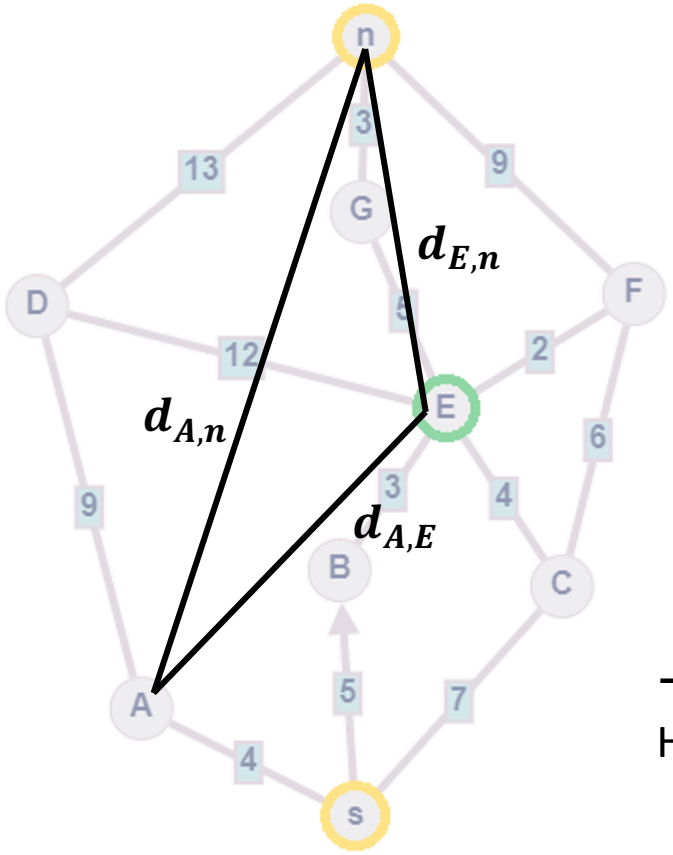
ALT-Algorithmus



$$d_{A,n} \geq \left| d_{A,E} - d_{E,n} \right|$$

Quelle: [Eigene Darstellung]

ALT-Algorithmus



→ $|d_{A,E} - d_{E,n}|$ kann als Heuristik verwendet werden

$$d_{A,n} \geq |d_{A,E} - d_{E,n}|$$

Quelle: [Eigene Darstellung]

Reach-Based Pruning

- Vereinfachung in der Vorverarbeitungsphase
- Ziel: kürzester Pfad vom Startknoten s zum Zielknoten n
- Reach von v auf P:

$$r(v,P) = \min\big(dist(s,v,P), dist(v,n,P)\big)$$

- Reach von v in G :

$$r(v,G) = \max\big(r(v,Q)\big)$$

- Knoten muss nur beachtet werden, wenn

$$r(v,G) \geq \underline{dist(s,v)} \quad \vee \quad r(v,G) \geq \underline{dist(v,n)}$$

Vergleich der Laufzeit

- Input: Straßennetz der San Francisco Bay Area
  → 330.024 Knoten und 793.681 Kanten
- Mittelwert der Laufzeit bei Anwendung auf jeweils 10.000 zufällig gewählten Paaren von Knoten

Algorithmus	Vorverarbeitung	Laufzeit
Dijkstra	-	30,49ms
ALT	5,7s	2,91ms

Anwendungen

- GIS ist ein System zum Erstellen, Verwalten, Analysieren und Kartieren verschiedener Arten von Daten.
- Die Verwendung des Standard-Dijkstra-Algorithmus führt bei der Ermittlung des kürzesten Weges zu einer längeren Zeitspanne
- Ein besserer Ansatz wäre die Erstellung eines temporären Datensatzes, bevor der Dijkstra-Algorithmus gestartet wird

- Mobiler Roboter ist ein wichtiges Thema in intelligenten Städten und die Pfadplanung ist eine wichtige Komponente unter den mobilen Roboter-Technologien
- Das Rolling-Window-Prinzip wird in einer unbekannten Umgebung verwendet, um Kollisionen zu vermeiden.
- Dijkstra und A* in erster Linie für die Pfadplanung in einer bekannten Umgebung eingesetzt werden.
- Die Idee ist eine Kombination von Dijkstra, A* und dem Rolling-Window-Prinzip.

- Verwendung bei Google Maps und ähnlichen Navigations-/Kartenanbietern
- Pfadsuchalgorithmen sind auch bei Autonomen System relevant
- Wahl des Algorithmus durch Faktoren bestimmt:
  - Sensorische Ausstattung des Roboters/Fahrzeugs
  - Kinetische Bewegungsgestaltung
  - Zur Verfügung stehenden Rechenressourcen
- Neben „Klassische" statische Algorithmen
- Gibt es auch dynamische Pfadsuchalgorithmen
  - Sind auf sich ändernde Umgebungen optimiert

Zusammenfassung

Uninformierte Pfadsuchalgorithmen	Informierte Pfadsuchalgorithmen
Breitensuche vollständig und optimal für einheitliche Pfadkosten, hat jedoch eine exponentielle Raumkomplexität	**A*-Suche** vollständig und optimal, für zulässige heuristische Funktionen
Tiefensuche weder vollständig noch optimal, hat dafür aber eine lineare Raumkomplexität	**ALT-Algorithmen** A* durch Preprocessing noch weiter optimiert
Greedy Dijkstra-Algorithmus ist optimal und hat eine Trefferquote von 100%	**Reach-based Pruning** erweitert den Dijkstra Algorithmus um Metrik und optimiert ihn dadurch

Quellen

- **[RN10]:** Russel, Stuart J. und Peter Norvig: Articial Intelligence - A Modern Approach, Pearson Education, Inc, 2010.
- **[HNR68]:** Hart, Peter, Nils Nilsson und Bertram Raphael: A Formal Basis for the Heuristic Determination of Minimum Cost Paths. IEEE Transactions on Systems Science and Cybernetics, 4(2):100 107, 1968.
- **[GH05]:** Goldberg, Andrew V. und Chris Harrelson: Computing the shortest path: A search meets graph theory. SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, Seiten 156165, Januar 2005.
- **[Gol07]:** Goldberg, Andrew V.: Point-to-Point Shortest Path Algorithms with Preprocessing. In: Lecture Notes in Computer Science, Seiten 88102. Springer Berlin Heidelberg, 2007.
- **[FGK+21]:** Foeada, Daniel, Alifio Ghifaria, Marchel Budi Kusumaa, Novita Hanafiahb und Eric Gunawanb: A Systematic Litera- ture Review of A* Pathfinding. Elsevier B.V, 2021.
- **[ESR22]:** Esri GIS Dictionary: Pathfinding. online, Juli 2022. https://support.esri.com/en/other-resources/gis- dictionary/term/7f861382-d88c-4828-8272-c3da4bdc8fa6.
- **[KSDS21]:** Karur, Karthik, Nitin Sharma, Chinmay Dharmatti und Joshua E. Siegel: A Survey of Path Planning Algorithms for Mobile Robots. Vehicles, 3(3):448–468, aug 2021.

Quellen

- **[Jav13]:** Javaid, Adeel: Understanding Dijkstra Algorithm. SSRN Elec- tronic Journal, 01 2013.
- **[AIS+20]:** Abusalim, Samah, Rosziati Ibrahim, Mohd Saringat, Sa- piee Jamel und Jahari Wahab: Comparative Analysis between Dijkstra and Bellman-Ford Algorithms in Shortest Path Optimiza- tion. IOP Conference Series: Materials Science and Engineering, 917:012077, 09 2020.
- **[ZG19]:** Zhou, Minhang und Nina Gao: Research on Optimal Path based on Dijkstra Algorithms. Proceedings of the 3rd International Con- ference on Mechatronics Engineering and Information Technology (ICMEIT 2019), 01 2019.
- **[MS20]:** Mukhlif, Fadhil und Abdu Saif: Comparative Study On Bellman-Ford And Dijkstra Algorithms. Int. Conf. Comm. Electric Comp. Net., 2020.
- **[HAMTMA20]:** Hamid Ali, Abed Alasadi, Azizand Mohammed Talib, Dhiya Mohammed und Abdulmaje Ahmed: A Network Analysis for Finding the Shortest Path in Hospital Information System with GIS and GPS. Journal of Network Computing and Applications (2020) 5: 10-22 Clausius Scientific Press, Canada, 2020.
- **[VB14]:** Vaibhavi, Patel und Prof. Chitra Baggar: A survey paper of Bellman-ford algorithm and Dijkstra algorithm for finding shortest path in GIS application. International Journal of P2P Network Trends and Technology (IJPTT), 2014.
- **[MW21]:** Myung, Hyun und Yang Wang: Robotic Sensing and Systems for Smart Cities. Sensors, 21(9), 2021.

Quellen

- **[mZIL17]:** Zhang, Hong mei und Ming long Li: Rapid path planning algorithm for mobile robot in dynamic environment. Advances in Mechanical Engineering, 9(12):168781401774740O, 2017.
- **[MKL19]:** Mehta, Heeket, Pratik Kanani und Priya Lande: Google Maps. International Journal of Computer Applications, 178(8):41– 46, may 2019.