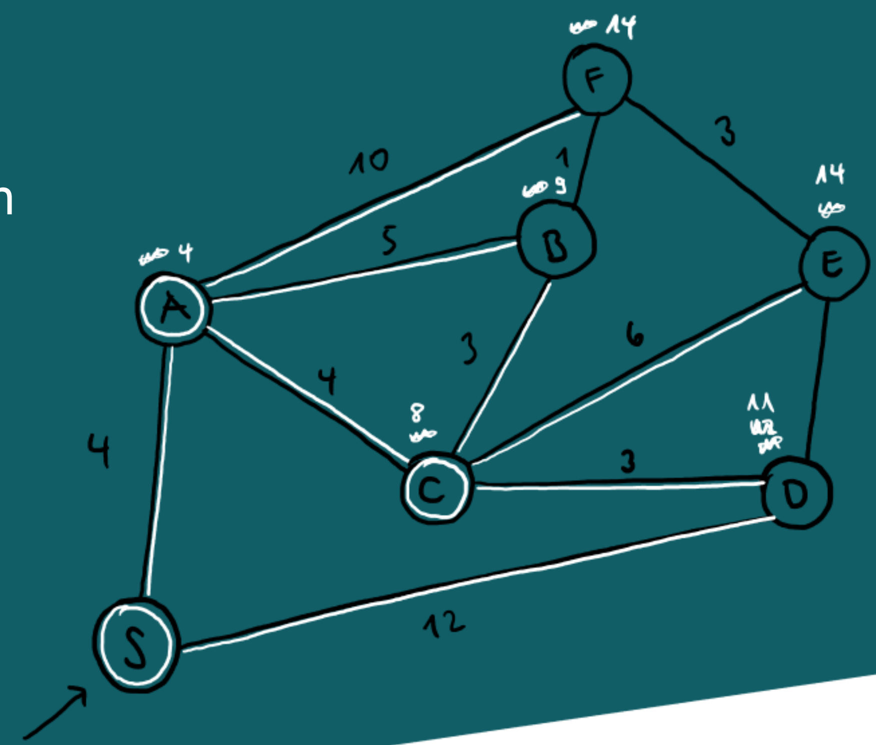
Funktionsprinzipien und Anwendungen von Algorithmen zur Pfadplanung

Tana Bögel, Moritz Hein, Jana Löwen



Informatik
Hauptcampus

HOCHSCHULE TRIER

Funktionsprinzipien und Anwendungen von Algorithmen zur Pfadplanung

Tana Bögel, Moritz Hein, Jana Löwen

Informatik
Hauptcampus

HOCHSCHULE TRIER

Einleitung

- Aufgabe: kostengünstigsten bzw. kürzesten Weg finden

→ Abhängig von Faktoren wie Hindernissen oder variablen Wegekosten

- Vielfältige Anwendungen

BELLMAN-FORD ALGORITHMUS

Informatik
Hauptcampus

HOCHSCHULE TRIER

Funktionsprinzip

Voraussetzungen

- Graph mit einer Menge von Knoten V und Kanten E [1]
- Keine negativen Zyklen
- Startknoten s und Zielknoten t [2]

[1] Struckmann, Wätjen (2016): Mathematik für Informatiker
[2] Ford Jr, Fulkerson (1964): Flows in Networks

Funktionsprinzip

Ablauf

- Initialisierungsphase
- N-1 Runden (N = |V|)
- Suche nach negativen Zyklen

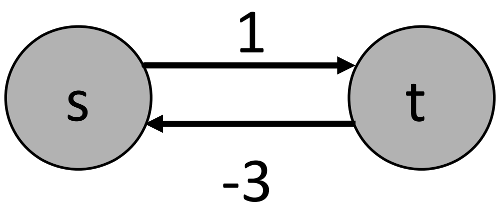
Negativer Zyklus

Negativer Zyklus

- Pfad der als Schleife durchlaufen werden kann und negative Gesamtkosten besitzt

Initialisierungsphase

- d[s]=0 → parent[s]=s
- d[t]=∞ → parent[t]=-



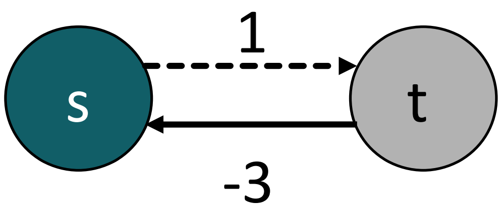
V	d	parent
s	0	s
t	∞	-

Negativer Zyklus

Runde 1

- d[t]=1
- parent[t]=s

V	d	parent
s	0	s
t	**1**	**s**



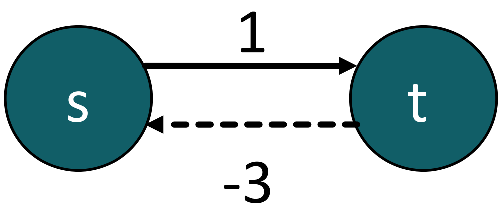
Negativer Zyklus

Suche nach negativen Zyklen

- d[s]=1+(-3)=-2<0
- parent[s]=t

- → Negativer Zyklus

V	d	parent
s	**-2**	**t**
t	1	s



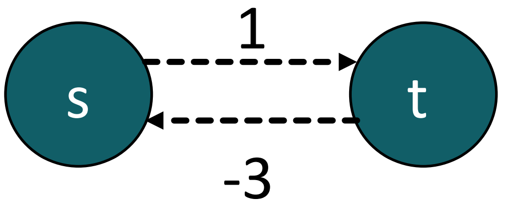
Dietzfelbinger, Mehlhorn, Sanders (2014): Algorithmen und Datenstrukturen

Negativer Zyklus

Infiziere

- d[s]=-∞
- d[t]=-∞

- Es existiert kein kürzester Weg



V	d	parent
s	**-∞**	**t**
t	**-∞**	**s**

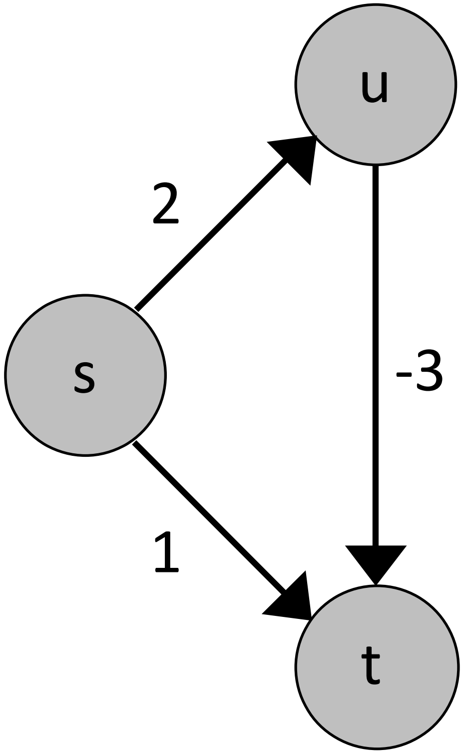
Beispiel

Beispiel

Initialisierungsphase

- d[s]=0 → parent[s]=s
- Alle anderen Distanzen auf ∞ setzen
- Alle anderen Vorgänger auf - setzen

V	d	parent
s	0	s
u	∞	-
t	∞	-

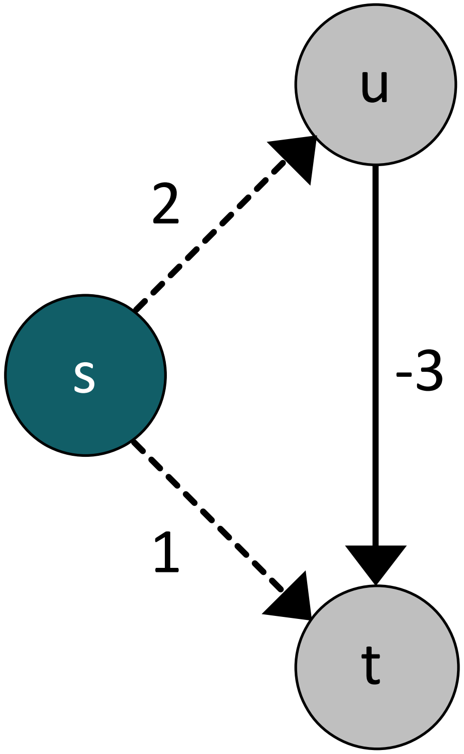


Beispiel

Runde 1

- $d[u]=0+2=2<\infty \rightarrow parent[u]=s$
- $d[t]=0+1=1<\infty \rightarrow parent[t]=s$

V	d	parent
s	0	s
u	**2**	**s**
t	**1**	**s**

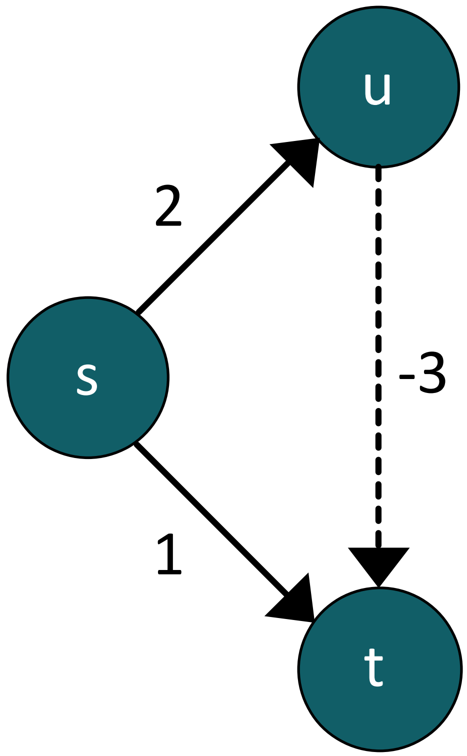


Beispiel

Runde 2

- d[t]=2+(-3)=-1<1 → parent[t]=u

V	d	parent
s	0	s
u	2	s
t	**-1**	**u**



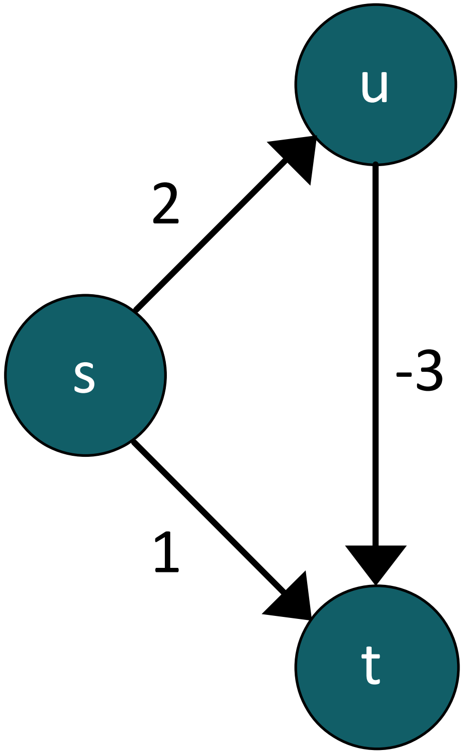
Dietzfelbinger, Mehlhorn, Sanders (2014): Algorithmen und Datenstrukturen

Beispiel

Suche nach negativen Zyklen

- → Keine negativen Zyklen gefunden
- Kürzester Weg: s → u → t, d[t]=-1

V	d	parent
s	0	s
u	2	s
t	-1	u



Dietzfelbinger, Mehlhorn, Sanders (2014): Algorithmen und Datenstrukturen

Anwendungen

Informatik
Hauptcampus

HOCHSCHULE TRIER

Distance-Vector Routing

- Runden sind „hops“
- Startknoten ist „root“
- Nachfolger statt Vorgänger
- Router sind die Knoten und Verbindungen zwischen diesen sind die Kanten

Vorteile

- Gute Nachrichten verbreiten sich schnell

Nachteile

- Schlechte Nachrichten verbreiten sich langsam
- Count-To-Infinity Problem
- Router kennen nur Teile der Routing-Tabelle

Logistik- und Distributionsprobleme

- Für neue Knoten muss nicht gesamtes Netz neu berechnet werden
- Negative Kantengewichte sind erlaubt

DIJKSTRA-ALGORITHMUS

Informatik
Hauptcampus

HOCHSCHULE TRIER

Funktionsprinzip

- Lösung des Single-Source Shortest Path Problems
  → findet kürzeste Wege vom Startknoten zu allen anderen Knoten im Graphen

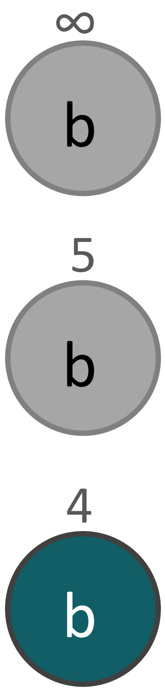
- Voraussetzungen:
  - Graph mit einer Menge von Knoten V und Kanten E
  - Nichtnegative Kostenfunktion c
  - Startknoten s

- Liefert einen Baum mit den kürzesten Wegen

Gerald Teschl (2013): Mathematik für Informatiker, Springer Vieweg

Funktionsprinzip

- Knoten erhalten nach jedem Schritt Markierungen
  - Noch unbekannte Knoten:
  - Temporär markierte Knoten:
  - Permanent markierte Knoten:

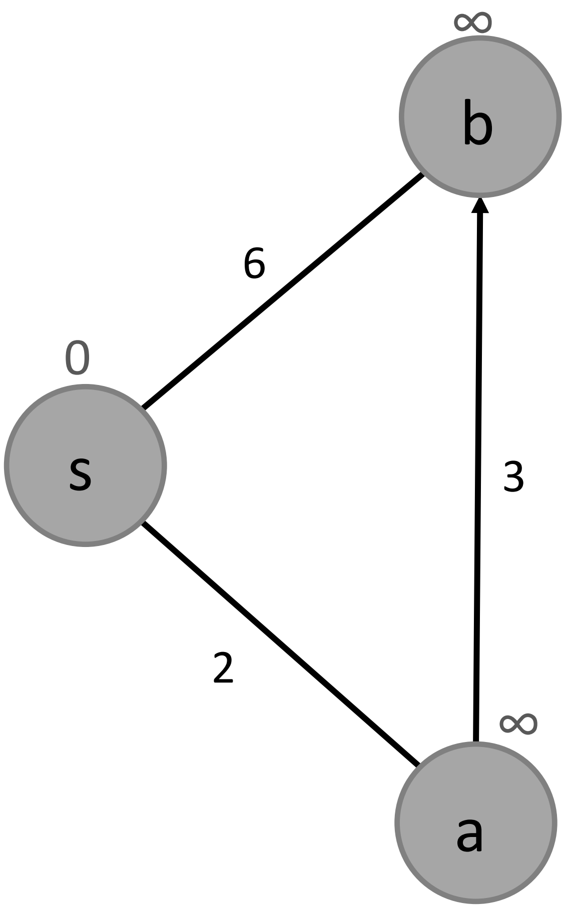


Funktionsprinzip

- Initialisierung:

- Der Startknoten s temporär markieren mit
  d[s] = 0, parent[s] = s

- Alle anderen Distanzen sind unendlich und die Vorgänger noch unbekannt



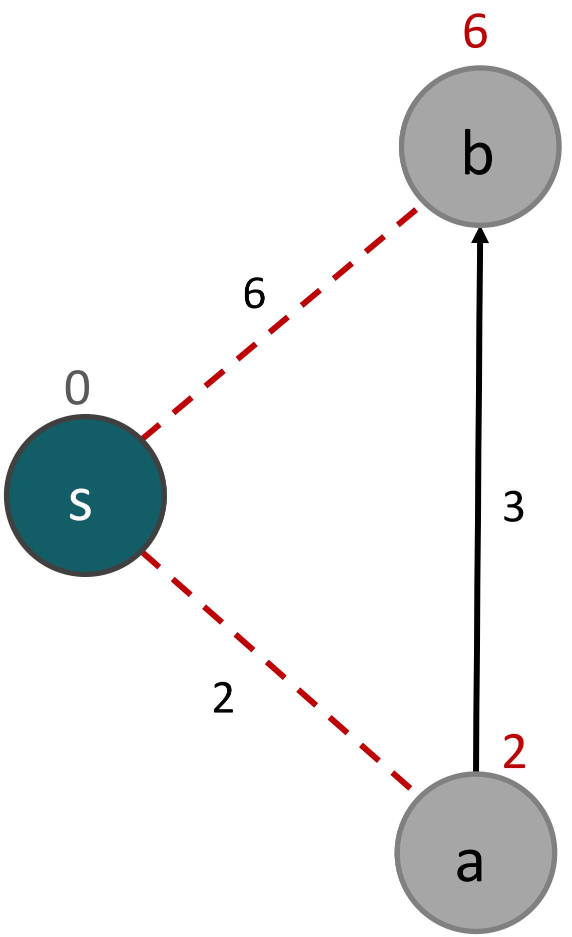
Funktionsprinzip

- Knoten s besuchen und permanent markieren

- Entfernungen vom Startknoten zu dessen Nachbarknoten gemäß der Kostenfunktion anpassen:

  d[a] = 2, parent[a] = s
  d[b] = 6, parent[b] = s

- Knoten a und b temporär markieren



Funktionsprinzip

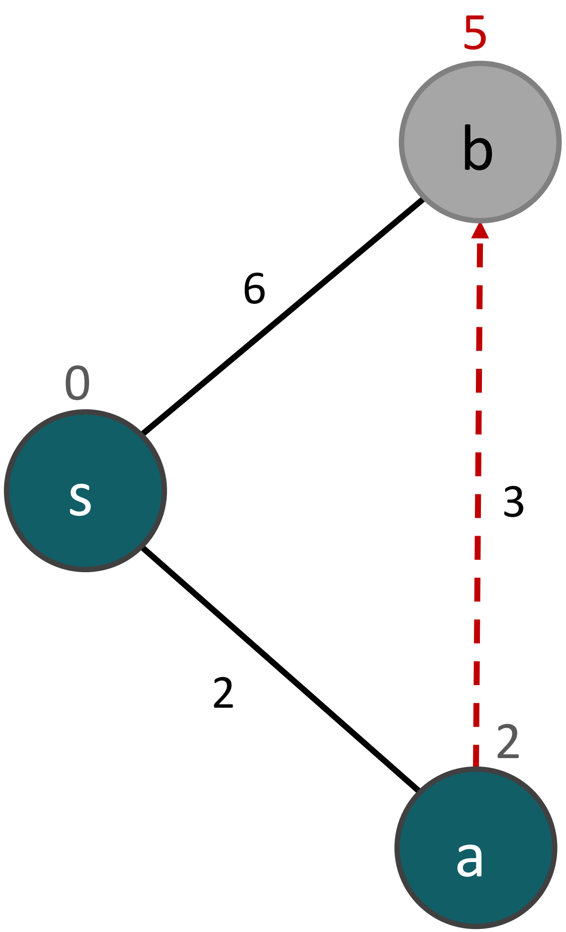
- Temporär markierten Knoten mit geringster Entfernung zu s besuchen und permanent markieren

- Entfernungen vom Startknoten über den besuchten Knoten zu dessen Nachbarknoten berechnen [1]:

  d[b] = d[a] + c(a,b)

- Relaxierung bei Knoten b [2]:

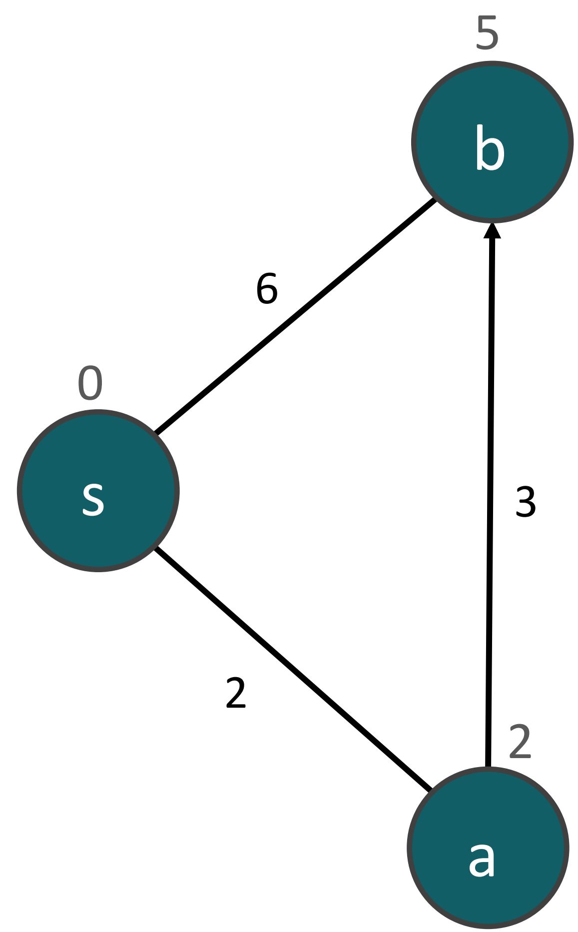
  d[b] = 5, parent[b] = a



[1] Brigitte Werners (2013): Grundlagen des Operation Research, Springer Gabler
[2] Martin Dietzfelbinger (2014): Algorithmen und Datenstrukturen, Springer Vieweg

Funktionsprinzip

- Temporär markierten Knoten mit geringster Entfernung zu s besuchen permanent markieren
- Da alle Knoten nun permanent markiert sind, ist der Algorithmus beendet



Sven Oliver Krumke (2012): Graphentheoretische Konzepte & Algorithmen, Springer Vieweg

Laufzeit

- Alle N Knoten erhalten genau einmal eine permanente Markierung
- Jeder Knoten hat maximal N-1 Nachbarn, für die die Distanz berechnet werden muss
- Damit ergibt sich: $O(N \cdot N-1) = O(N^2)$
- Die exakte Laufzeit ist von der Wahl der Priorityqueue abhängig → Verbesserung möglich

Anwendungen

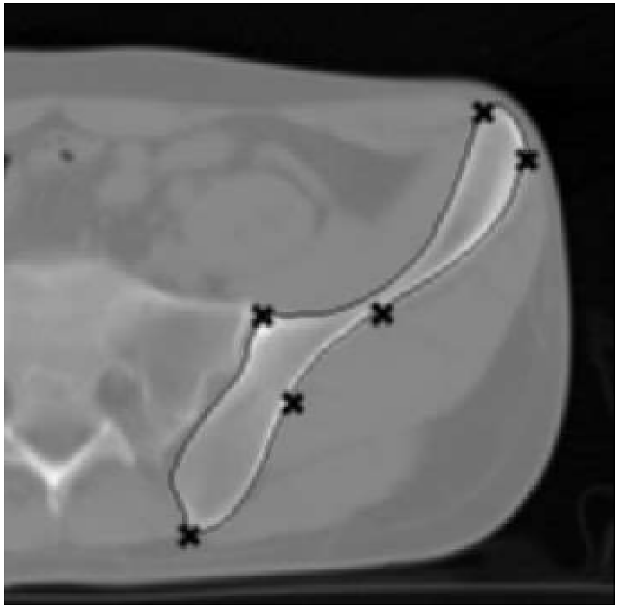
Routenplanung

- Straßennetz wird durch den Graphen repräsentiert
- Lösung des Single-Pair Shortest Path Problems
  → findet kürzesten Weg von s zu t
- Angabe der Fahrtzeit anhand von Durchschnitts-geschwindigkeiten → Berechnung von
  - Entfernung auf schnellstem Weg sowie
  - Fahrtzeit auf kürzestem Weg
- Effizientere Varianten: frühzeitiges Stoppen, bidirektionale Suche

Dirk Mattfeld (2014): Logistiknetzwerke, Springer Gabler

Segmentierung medizinischer Bilder

- Zur Auswertung medizinischer Bilder für Diagnosen und Therapien
- Abgrenzung von relevanten Strukturen, beispielsweise Tumoren
- Verwendung des Live-Wire-Verfahrens:
  - Hervorhebung der Objektkontur ausgehend vom Startpunkt über gewählte Saatpunkte bis zum Mauszeiger [1]



Segmentierung des Darmbeins [2]

[1] Sebastian Dörn (2017): Programmieren für Ingenieure und Naturwissenschaftler, Springer Vieweg
[2] Heinz Handels (2009): Medizinische Bildverarbeitung, Vieweg + Teubner

Segmentierung medizinsicher Bilder

- Live-Wire-Verfahren:
  - Transformation des Bildes in einen Graphen:

    Bildpunkt ≙ Knoten, Kontur ≙ Pfad
  - Bei der Kostenfunktion entspricht kostengünstigster Weg entspricht möglichst der Objektkontur
  - Kostengünstigsten Weg mit Hilfe des Dijkstra-Algorithmus berechnen und optisch hervorheben

Heinz Handels (2009): Medizinische Bildverarbeitung, Vieweg + Teubner

Routing im Internet

- Ermöglicht die Kommunikation und Datenübertragung zweier Rechner aus verschiedenen lokalen Netzwerken (LANs) [1]
- Router speichern Nachbarn und Distanzen in Link-State-Paketen → Verteilung an Router im Netzwerk per Flooding [2]
- Berechnung des kürzesten Weges zu allen andere Routern mit Hilfe des Dijkstra-Algorithmus
- Verkürzung der Laufzeit durch Aufteilung in Teilnetzwerke [1]

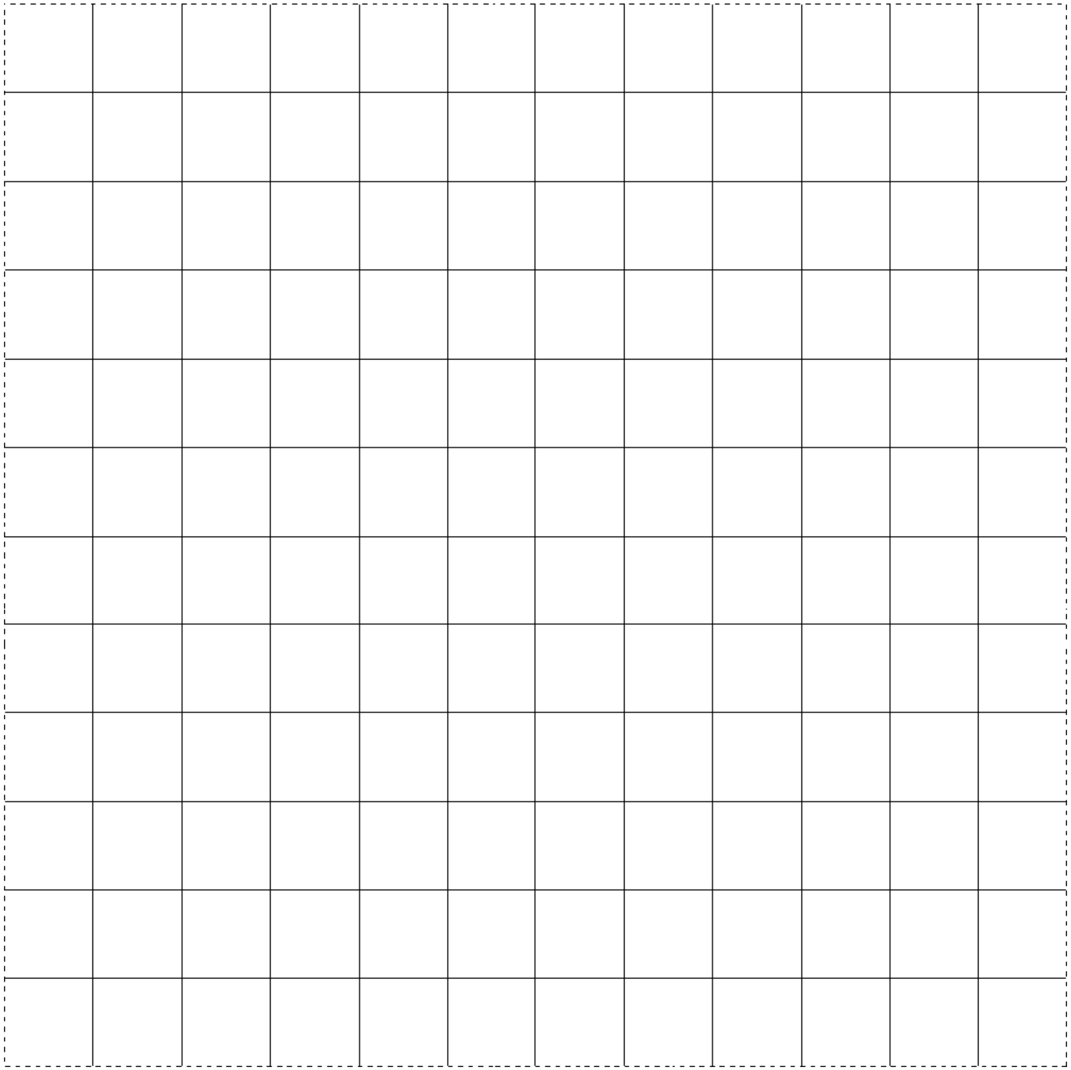
[1] Gerald Teschl (2013): Mathematik für Informatiker, Springer Vieweg
[2] Baun (2019): Computer Networks, Springer Vieweg

A-STERN ALGORITHMUS

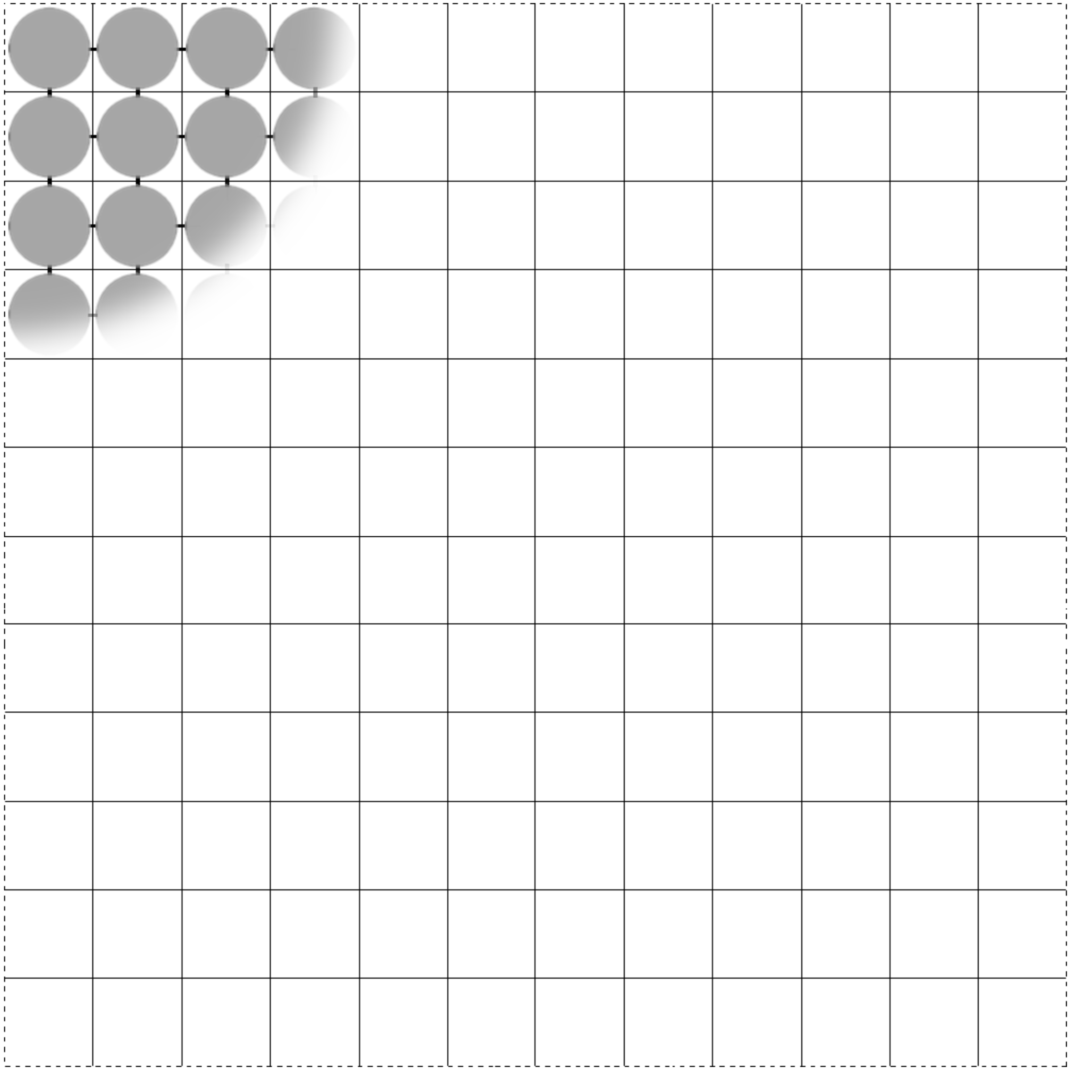
Funktionsprinzip

- Berechnet kürzesten Pfad eines kantengewichteten Graphen
- Basiert auf Dijkstra-Algorithmus
- Unterstützt keine negativ gewichteten Kanten
- Nutzt eine heuristische Funktion um effizienter zu suchen

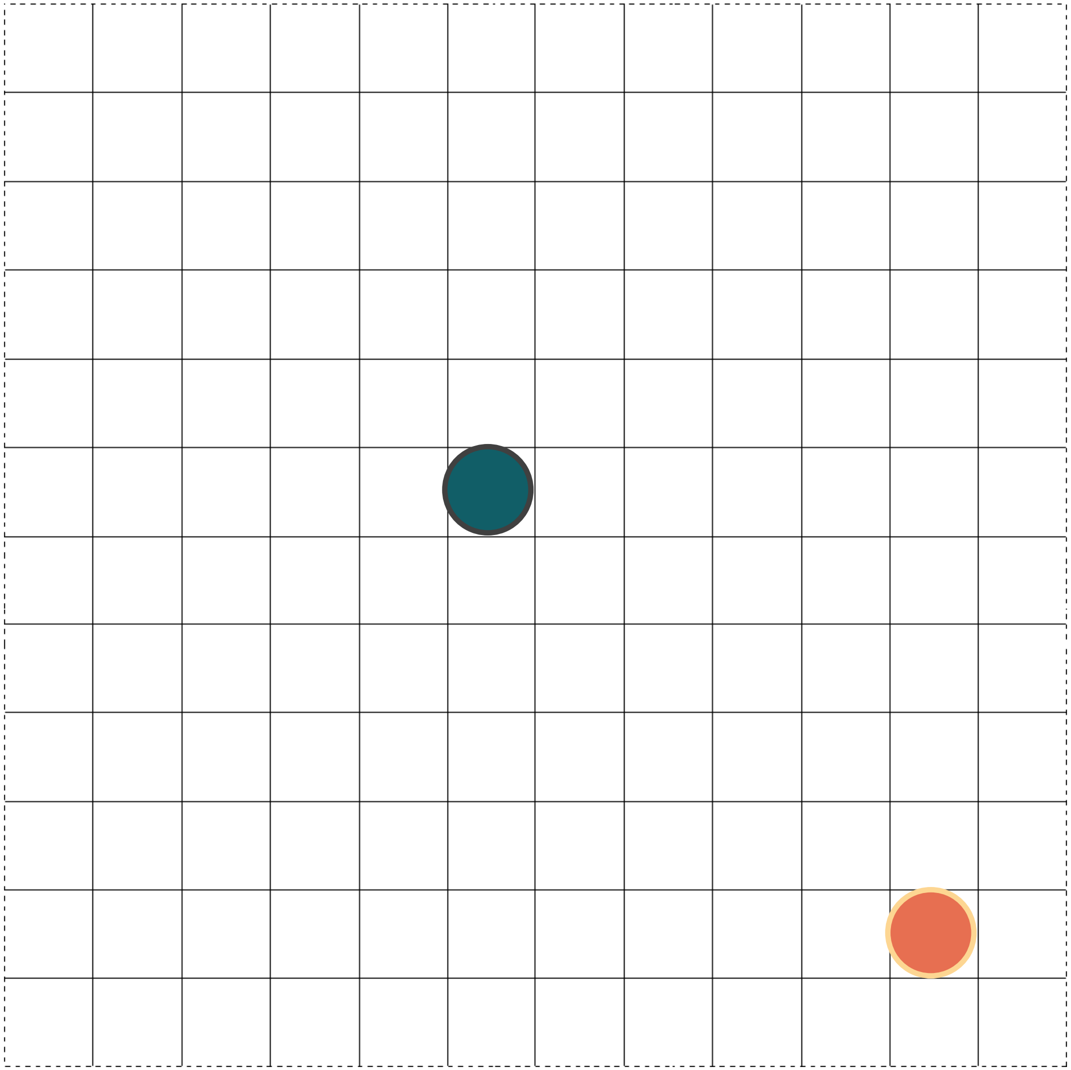
Funktionsprinzip



Funktionsprinzip



Funktionsprinzip



Funktionsprinzip

- PLACEHOLDER - Dijkstra erkundete Nachbarn zum Ziel, Pfeil direkt zum Ziel mit kleinerer erkundeten Fläche auf der nächsten Folie

Heuristische Funktion

Heuristische Funktion

„Mit begrenztem Wissen und wenig Zeit dennoch zu wahrscheinlichen Aussagen oder praktikablen Lösungen zu kommen."

- „Simple heuristics that make us smart", G. Gigerenzer und P. M. Todd (1999)

Veränderte Kostenfunktion

$$f(n) = g(n) + h(n)$$

Kosten vom Startknoten

Geschätzte Kosten bis zum Zielknoten

Zusammenfassung & Ausblick

Zusammenfassung

- Bellman-Ford Algorithmus: Umgang mit negativen Kantengewichten
- Dijkstra-Algorithmus: universell einsetzbar
- A*-Algorithmus: Anpassung an Problemdomäne

→ Hohe Relevanz auch in Zukunft

Gibt es noch Fragen?

Vielen Dank für eure Aufmerksamkeit!