FreeFlower

Informatik: Data Science und Sicherheit

🚉: **Kürzeste Wege mit Graphen**

Thema und Lernziele

- Thema: Grundlagen der Abstraktion und Modellierung mit Graphen

Thema und Lernziele

- Thema: Grundlagen der Abstraktion und Modellierung mit Graphen
- Lernziele:

Thema und Lernziele

- Thema: Grundlagen der Abstraktion und Modellierung mit Graphen
- Lernziele:
  - Sie beschreiben das Problem, möglichst kurze Wege zu finden, einem Kollegen und verwenden dabei die korrekten Fachbegriffe.

Thema und Lernziele

- Thema: Grundlagen der Abstraktion und Modellierung mit Graphen
- Lernziele:
  - Sie beschreiben das Problem, möglichst kurze Wege zu finden, einem Kollegen und verwenden dabei die korrekten Fachbegriffe.
  - Sie wenden die einzelnen Schritte des berühmten Dijkstra-Algorithmus auf ein konkretes Problem an.

Ablauf der Lektion

- zuerst: Vertraut werden mit dem Thema mit Hilfe einiger Aufgaben (Sie sind gefragt!)

Ablauf der Lektion

- zuerst: Vertraut werden mit dem Thema mit Hilfe einiger Aufgaben (Sie sind gefragt!)
- dazwischen: Grundidee des Dijkstra-Algorithmus und Durchführung an einem Beispiel (gemeinsam)

Ablauf der Lektion

- zuerst: Vertraut werden mit dem Thema mit Hilfe einiger Aufgaben (Sie sind gefragt!)
- dazwischen: Grundidee des Dijkstra-Algorithmus und Durchführung an einem Beispiel (gemeinsam)
- gegen Ende: Sie nutzen die Zeit für Übungen

Bemerkungen zu dem Thema

- Für dieses Thema werden wir die neuen Begriffe aus der ersten Lektion verwenden.

Bemerkungen zu dem Thema

- Für dieses Thema werden wir die neuen Begriffe aus der ersten Lektion verwenden.
- Auch der Inhalt dieses Themas ist verknüpft mit der Thematik von Webseiten und dem Internet.

Bemerkungen zu dem Thema

- Für dieses Thema werden wir die neuen Begriffe aus der ersten Lektion verwenden.
- Auch der Inhalt dieses Themas ist verknüpft mit der Thematik von Webseiten und dem Internet.
- Sie haben in dieser Lektion die Chance die Grundzüge eines Algorithmus kennenzulernen, der sogar einen Namen hat (und somit super wichtig sein muss).

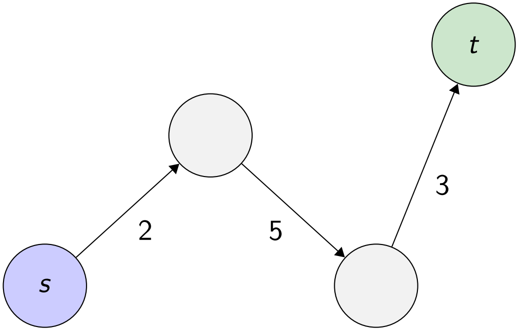
Lektion 2

- Muss mit „kürzestem Weg“ immer die Distanz gemeint sein?

Lektion 2

- Muss mit „kürzestem Weg“ immer die Distanz gemeint sein?
- Sie haben die Auswahl zwischen drei verschiedenen Zugverbindungen. Für welche entscheiden Sie sich vermutlich?

Schreibweise



Der Weg von s nach t hat die Länge (Kosten / Gewicht) 10.

- Die Länge eines Weges p wird mit $c(p)$ bezeichnet.
- Wir schreiben

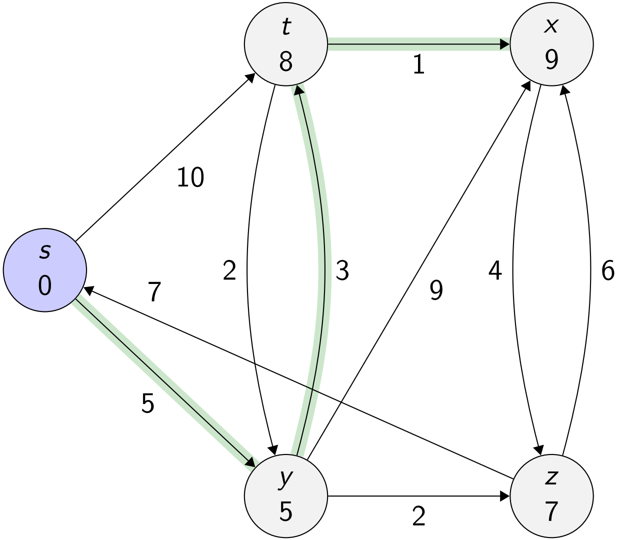
$$u \overset{p}{\leadsto} v$$

und meinen damit einen Weg p von u nach v.

1. Aufgabenblock (10 Minuten)

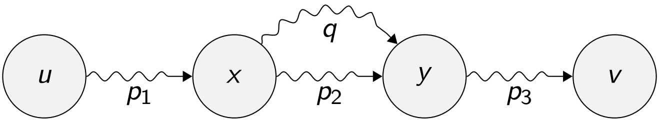
Bearbeiten Sie die vier Aufgaben 1.1, 1.2, 1.3 und 1.4.

Beobachtung



Beobachtung

Sei p der Weg $u \overset{p_1}{\rightsquigarrow} x \overset{p_2}{\rightsquigarrow} y \overset{p_3}{\rightsquigarrow} v$ zusammengesetzt aus den Wegen p_1, p_2, p_3. Angenommen p sei ein kürzester Weg von u nach v. Seien x und y beliebige Knoten entlang dieses Weges. Dann ist der Abschnitt p_2 von x nach y ein kürzester Weg von x nach y.



Dijkstra-Algorithmus

Edsger Wybe Dijkstra

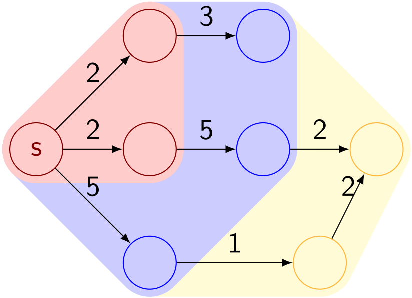
Dijkstra-Algorithmus

- Wir wollen die Längen (Kosten) der jeweils kürzesten Wege von s (Startknoten) zu allen anderen Knoten in einem Graphen finden. Wir werden nur noch Graphen mit nicht-negativen Kantengewichten anschauen.

Dijkstra-Algorithmus

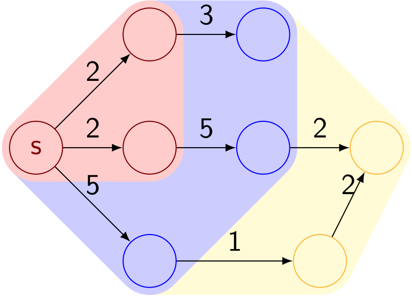
- Wir wollen die Längen (Kosten) der jeweils kürzesten Wege von s (Startknoten) zu allen anderen Knoten in einem Graphen finden. Wir werden nur noch Graphen mit nicht-negativen Kantengewichten anschauen.
- Computer sind enorm schnell. Warum berechnen sie nicht einfach die Länge jedes Weges und wählen einen kürzesten aus?

Grundidee



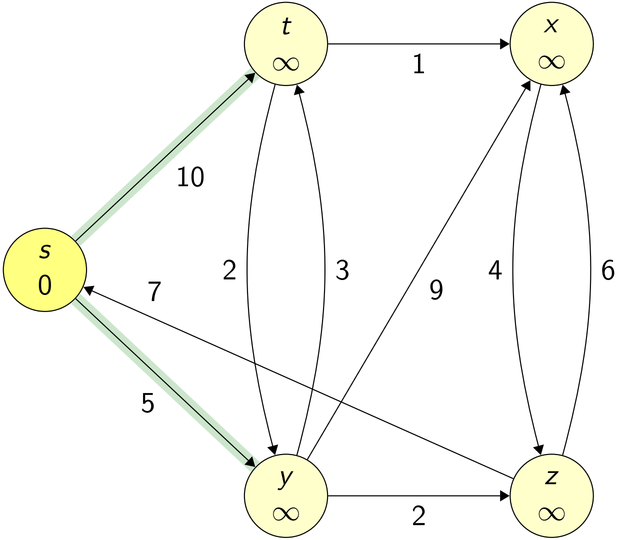
- Die Distanzen der roten Knoten sind bereits optimal und werden sich nicht mehr ändern (nicht offensichtlich).

Grundidee



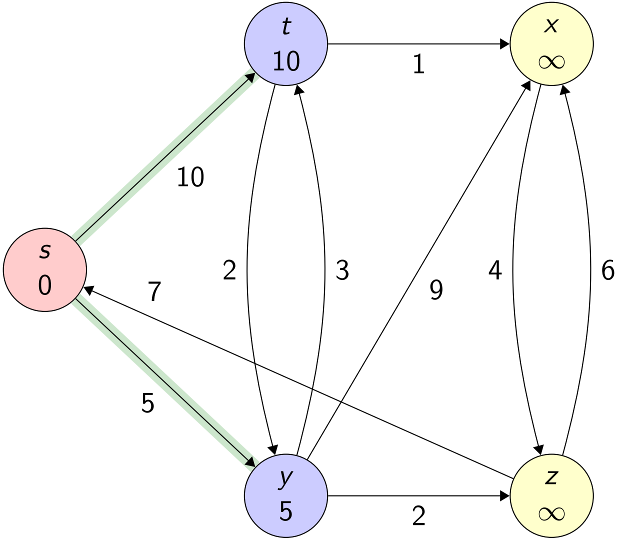
- Die Distanzen der roten Knoten sind bereits optimal und werden sich nicht mehr ändern (nicht offensichtlich).
- Ebenfalls sehr wichtig: Updates der Distanzen!

Djkstra: *step by step*



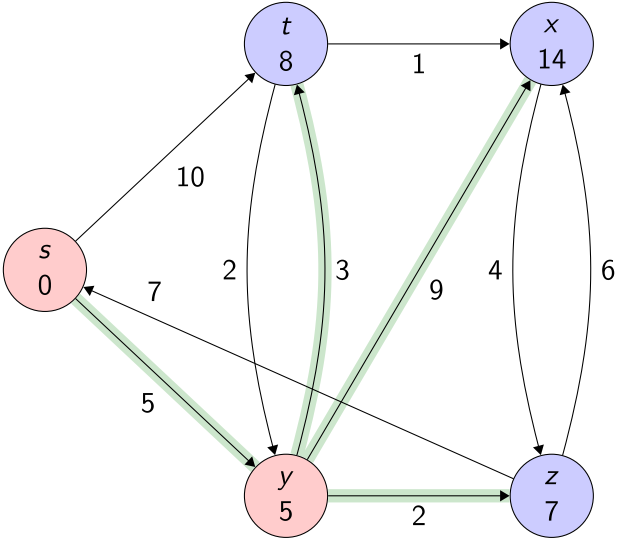
Schritt 1

Djkstra: *step by step*



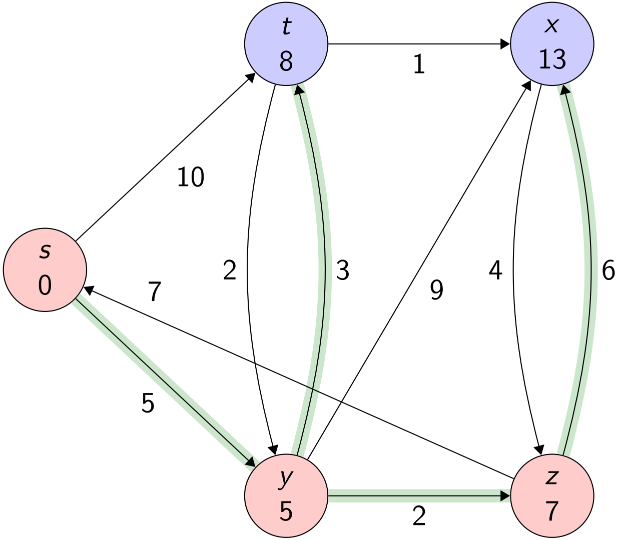
Schritt 2

Djkstra: *step by step*



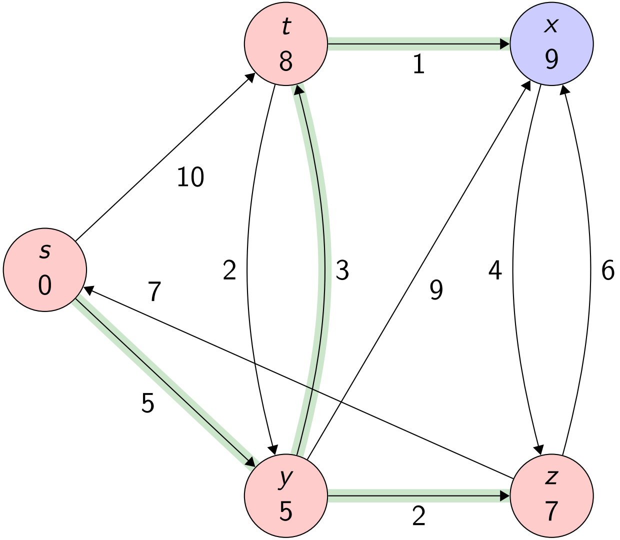
Schritt 3

Djkstra: *step by step*



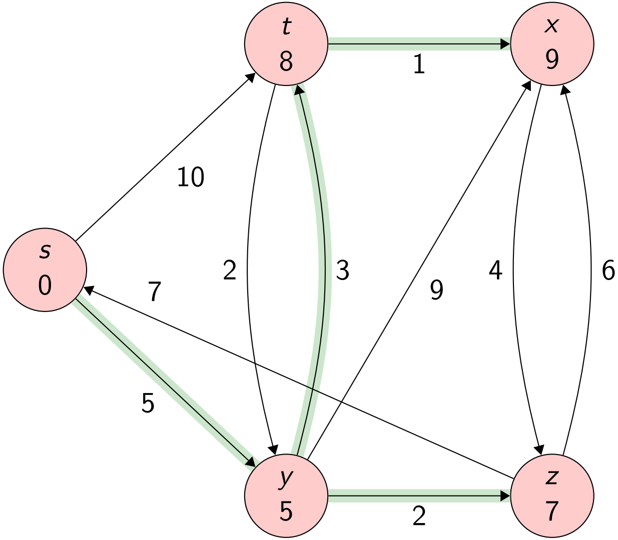
Schritt 4

Djkstra: *step by step*



Schritt 5

Djkstra: *step by step*



Schritt 6

Initialisierung (Startbedingungen)

Wir bezeichnen mit $d_s[v]$ die Länge des kürzesten **bisher** gefunden Weges vom Startknoten s zum Knoten v. Mit $c(u, v)$ bezeichnen wir das Gewicht (cost) der Kante von Knoten u zum Knoten v.

- Zu Beginn: $d_s[v] = \infty$ für alle Knoten des Graphen. (Initialisierung)
- Ziel: Für jeden Knoten v des Graphen soll am Ende $d_s[v]$ die Länge eines kürzesten Weges von s nach v sein. (rote Knoten)

Dijkstra-Algorithmus (informal)

- Wir fügen $u := s$ zur roten Menge hinzu und betrachten alle Knoten, welche direkt über eine einzige Kante von s aus erreichbar sind (also genau die Knoten in der blauen Menge).
- Für jeden dieser blauen Knoten v schauen wir, ob die Bedingung

$$d_s[u] + c(u, v) < d_s[v]$$

erfüllt ist. Diese Bedingung hat die Bedeutung, dass die bislang berechneten Länge $d_s[v]$ grösser ist, als wenn wir von s über u nach v gehen würden. Trifft diese zu, so führen wir das Update

$$d_s[v] = d_s[u] + c(u, v)$$

durch und verbessern dadurch den bisherigen Wert $d_s[v]$.

- Nun wählen wir denjenigen blauen Knoten u, welchen den (einen der) kleinsten Werte $d_s[u]$ unter allen blauen Knoten hat, als den neuen Knoten u und fügen u zur roten Menge hinzu.
- Wir arbeiten nun so weiter, bis keine blauen Knoten mehr übrig sind.

wichtige Eigenschaft

Eine entscheidende Tatsache in dem Algorithmus (dies ist nicht völlig offensichtlich) ist, dass die roten Knoten v ihren Wert $d_s[v]$ nie mehr ändern werden und dass für sie in der Tat $d_s[v]$ die Länge eines kürzesten Weges von s nach v ist.

2. Aufgabenblock (Zeit: Rest der Lekltion)

Bearbeiten Sie die drei Aufgaben 1.5, 1.6 und 1.7.

Vielen Dank!

Pseudocode

```
## Initialisierung ##
für alle Knoten u des Graphen setze:
    d_s[u] = ∞
d_s[s] ← 0
S ← ∅
Q ← alle Knoten des Graphen

## Schritte ##
solange Q nicht leer ist, mach:
    u ← Knoten aus Q mit kleinstem d_s[u]
    S ← S ∪ {u}

    für alle von u direkt erreichbaren Knoten v mach:
        # ist der Weg von s nach v kürzer über u?
        falls d_s[u] + c(u, v) < d_s[v] dann:
            d_s[v] ← d_s[u] + c(u, v)
```