

# Tutorium 9

## Algorithmen I SS 14

Institut für Theoretische Informatik



# Kürzeste Wege

- berechnet Entfernung von allen anderen (erreichbaren) Knoten zu Startknoten (single source shortest path)
- Einschränkung: keine negativen Kantengewichte
- Laufzeit abhängig von der verwendeten Priority-Queue
  - naiv  $\mathcal{O}(m + n^2)$
  - Binärer Heap  $\mathcal{O}((m + n) \log n)$
  - mit Fibonacci Heap sogar  $\mathcal{O}(m + n \log n)$
  - noch weitere Verbesserungen mit speziellen Anpassungen möglich

- speichere vorläufige Entfernung zu Startknoten  $s$  in Array  $d$ , zu Beginn  $d[s] = 0, \forall v \in V \setminus \{s\} : d[v] = \infty$
- *Relaxiere Kanten*: Wenn für eine Kante  $(u, v)$  gilt, dass  $d[u] + c(u, v) < d[v]$ , setze  $d[v] := d[u] + c(u, v)$
- Also: Wenn es über eine neue Kante einen kürzeren Weg gibt, nimm diesen

- Problem: Wie oft müssen welche Kanten relaxiert werden, um wirklich kürzeste Wege zu haben?
- Lösung: Zum Knoten mit der kleinsten Entfernung kann kein kürzerer Weg mehr gefunden werden
- Verwalte unbetrachtete Knoten dazu in einer Priority-Queue

- ① Initialisiere  $d$ .
- ② Füge Startknoten in Priority-Queue  $Q$  ein.
- ③ Solange  $Q$  Elemente enthält:
  - ① Nimm minimalen Knoten  $u$  aus  $Q$ .
  - ② Relaxiere alle ausgehenden Kanten  $(u, v)$  von  $u$ . Füge jeden Knoten  $v$  in  $Q$  ein oder aktualisiere die Priorität (Entfernung).
  - ③ Der Knoten  $u$  muss jetzt nicht mehr betrachtet werden.

# Fragen? Wiederholung?

Von einem Startbahnhof aus soll ein Zielbahnhof erreicht werden. Gegeben ist dazu ein Fahrplan, der für jede Verbindung einen Start- und einen Zielbahnhof, sowie eine Abfahrts- und eine Ankunftszeit enthält. Die Zeit, die zum Umsteigen benötigt wird, soll vernachlässigt werden.

**Aufgabe:** Wie kann der früheste Ankunftszeitpunkt ermittelt werden? Wie kann außerdem zusätzlich für diesen frühesten Ankunftszeitpunkt der späteste Abfahrtszeitpunkt berechnet werden?