Karlsruher Institut für Technologie

Tutorium 9

Algorithmen I SS 14

Institut für Theoretische Informatik

Kürzeste Wege

Dijkstra-Algorithmus

- 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

Dijkstra: Funktionsweise (1)

- 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

Dijkstra: Funktionsweise (2)

- 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

Dijkstra: Algorithmus

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.

Karlsruher Institut für Technologie

Fragen? Wiederholung?