Kürzeste Wege

Zur Berechnung kürzester Wege hilft folgendes Lemma

Lemma

Nach einer Sequenz R von Kantenrelaxierungen ist $v.d$ die Länge eines kürzesten Weges von s nach v falls die Sequenz R einen kürzesten Weg p von s nach v als Subsequenz enthält. In diesen Fall enthält die predecessor Information einen kürzesten Weg von s nach v.

Einfacher Algorithmus zur Berechnung kürzester Wege

Für Graphen mit nicht-negativen Gewichten

Algorithm 1: Simple SSSP von Startpunkt s:

```
1 Initialisierung(s)
2 for i = 1 ⟶ n − 1 do
3 |   for alle Kanten (u, v) ∈ G do
4 |   |   Kantenrelaxierung((u, v), ω_(u,v))
5 |   end
6 end
```

Einfacher Algorithmus zur Berechnung kürzester Wege

Korrektheit folgt aus Eigenschaft, dass jeder kürzeste Weg maximal $n - 1$ Kanten enthält.
Laufzeit ist $O(mn)$ (folgt direkt aus Laufzeit von Initialisierung und Kantenrelaxierung).

Berechnung kürzester Wege in Graphen mit negativen Gewichten

Algorithm 2: Bellman-Ford Algorithmus von Startpunkt s:

```
1 Simple SSSP(s)
2 for alle Kanten (u, v) ∈ G do
3 |   if v.d > u.d + ω_(u,v) then
4 |   |   Infiziere(v)
5 |   end
6 end
```

Algorithm 3: Infiziere einen Knoten v:

```
1 if v.d > −∞ then
2 |   v.d = −∞
3 |   for Kanten (v, w) ausgehend von v do
4 |   |   Infiziere(w)
5 |   end
6 end
```

Bellman-Ford Algorithmus

- Fall von negativem Zyklus: nach Ausführung von Simple SSSP(s) gibt es einen Knoten v mit $v.d > u.d + \omega_{(u,v)}$
- Bellman-Ford Algorithmus setzt die Distanz $v.d$ auf $-\infty$
- Anschließend werden Gewichte der von v erreichbaren Knoten auf $-\infty$ gesetzt
- Methode *Infiziere*(v) ist der Tiefensuche ähnlich
- Laufzeit: $O(mn)$

Berechnung kürzester Wege in Graphen mit nicht-negativen Gewichten

Frage Kann man im Fall nicht-negativer Gewichte kürzeste Wege auch schneller berechnen?

Idee Reihenfolge der Kantenrelaxierungen gut auswählen

DS
- Q: Min-Priority Queue, die Knoten v mit Prioritäten $v.d$ enthält
- S: Menge an Knoten, für die bereits ein kürzester Weg gefunden wurde

Dijkstra Algorithmus

Für Graphen mit nicht-negativen Gewichten

Algorithm 4: Dijkstra Algorithmus von Startpunkt s:

```
1  Initialisierung(s)
2  S = ∅
3  Q = alle Knoten in G
4  while Q ≠ ∅ do
5  |   u = EXTRACT-MIN(Q)
6  |   S = S ∪ u
7  |   for Kanten (u, v) ausgehend von u do
8  |   |   Kantenrelaxierung((u, v), ω_(u,v))
9  |   end
10 end
```

Dijkstra Algorithmus

Theorem

Für einen gewichteten Graphen G mit nicht-negativen Gewichten berechnet der Dijkstra Algorithmus alle kürzesten Wege vom Startpunkt s. Bei der Terminierung des Algorithmus sind die Längen der kürzesten Wege $v.d = \delta(s, v)$, und die kürzesten Wege sind in den predecessor Informationen enthalten.

Beweisidee verwendet die Invariante, dass bei Beginn der While-Schleife (in Zeile 5) gilt, dass $u.d = \delta(s, u)$.

Dijkstra Algorithmus

Laufzeit

- Initialisierung: $O(n)$
- Min-Priority Queue Operationen
  - $O(n)$ Mal Kosten für Einfügen
  - $O(n)$ Mal Kosten für EXTRACT-MIN
  - $O(m)$ Mal Kosten für DECREASE-KEY (in Kantenrelaxierung)