Breitensuche (Breadth First Search - kurz BFS)

- Möglichkeit zur Sortierung eines generellen Graphen
- Idee: G wird in einer Reihenfolge abgesucht, die immer zuerst in die Breite geht
- Jeder Knoten u von G bekommt
  - $u.d$: Zeitstempel der ersten Prüfung des Knotens
  - $u.predecessor$: Vorgänger von u in der Suche
- Außerdem: first-in first-out Queue Q

BFS

Algorithm 1: Breitensuche Algorithmus von Quelle s:

```
1  for jeden Knoten u von G do
2  |   u.predecessor = NULL
3  |   u.d = ∞
4  end
5  s.d = 0
6  ENQUEUE(Q, s)
7  while Q ≠ ∅ do
8  |   u = DEQUEUE(Q)
9  |   for jeden Knoten v mit (u, v) ∈ G do
10 |   |   if v.d == ∞ then
11 |   |   |   v.d = u.d + 1
12 |   |   |   v.predecessor = u
13 |   |   |   ENQUEUE(Q, v)
14 |   |   end
15 |   end
16 end
```

Laufzeitanalyse

- Initialisierung $O(n)$
- Jeder Knoten wird nur ein Mal in Q eingefügt
- Operationen ENQUEUE und DEQUEUE in $O(1)$, alle Operationen insgesamt in $O(n)$
- Die Adjazenzliste jedes Knotens wird ein Mal untersucht, Gesamtlänge $O(m)$
- Insgesamte Laufzeit: $O(n+m)$

Zusammenhang zu kürzesten Wegen

Definition

Die *Länge eines kürzesten Weges* $\delta(s, v)$ zwischen s und v in G ist die minimale Anzahl der Kanten in allen Pfaden von s nach v in G. Falls es keinen solchen Pfad gibt ist $\delta(s, v) = \infty$. Ein *kürzester Weg* von s und v in G ist ein Pfad von s nach v in G mit $\delta(s, v)$ Kanten.

Theorem

Bei Ende der Breitensuche enthält die Variable $v.d$ die Länge eines kürzesten Weges $\delta(s, v)$. Ein kürzester Weg kann gefunden werden, indem man den predecessor Knoten folgt bis s erreicht wird.

Kürzeste Wege in gewichteten Graphen

Definition

Gewichteter Graph: Ein Graph heißt gewichtet, wenn jeder Kante ein reelles Gewicht zugeordnet wird. Ansonsten heißt der Graph ungewichtet.

Definition

Sei G ein gewichteter Graph, und sei ω_e das Gewicht der Kante e. Die *Länge des Pfads* $p = (e_1, e_2, \ldots, e_k)$ ist $\omega_p = \sum_{i=1}^{k} \omega_{e_i}$.

Wir möchten kürzeste Wege von einem Knoten s aus berechnen.

Grundoperationen für die Berechnung

Algorithm 2: Initialisierung s:

```
1 for jeden Knoten u von G do
2 |   u.predecessor = NULL
3 |   u.d = ∞
4 end
5 s.d = 0
```

Algorithm 3: Kantenrelaxierung $(u, v), \omega_{(u,v)}$:

```
1 if v.d > u.d + ω(u,v) then
2 |   v.d = u.d + ω(u,v)
3 |   v.predecessor = u
4 end
```

Gewichtete kürzeste Wege

Lemma

Falls nach einer Sequenz von Kantenrelaxierungen $v.d < \infty$, dann gibt es einen Weg der Länge $v.d$ von s nach v.

Frage: In welcher Reihenfolge sollten die Kanten relaxiert werden?

Sonderfall: Kürzeste Wege in DAGs

- Beginne bei s und relaxiere alle ausgehenden Kanten
- Nehme den Nachfolger von s in der topologischen Sortierung und relaxiere alle ausgehenden Kanten
- Nehme den Nachfolger und wiederhole

Laufzeit: $O(m+n)$