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.
  D.h. Knoten, die nahe am Ausgangsknoten der Suche sind, werden zuerst gefunden, zuerst die mit Distanz 1, dann die mit Distanz 2, und so fort.
- Jeder Knoten u von G bekommt
  - $\delta[u]$: Distanz vom Startknoten (minimale Anzahl von Kanten auf Weg vom Startknoten zu u)
  - $\pi[u]$: Vorgänger von u in der Suche, also Vorgänger auf so einem minimalen Weg
- Außerdem: first-in first-out Queue Q

BFS

Algorithm 1: Breitensuche Algorithmus von Quelle s:

```
for jeden Knoten u von G do
    π[u] = NULL
    δ[u] = ∞
end
δ[s] = 0
ENQUEUE(Q, s)
while Q ≠ ∅ do
    u = DEQUEUE(Q)
    for jeden Knoten v mit [u, v⟩ ∈ G do
        if δ[v] == ∞ then
            δ[v] = δ[u] + 1
            π[v] = u
            ENQUEUE(Q, v)
        end
    end
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* $d_s(v)$ 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 $d_s(v) = \infty$. Ein *kürzester Weg* von s und v in G ist ein Pfad von s nach v in G mit $d_s(v)$ Kanten.

Theorem

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

Kürzeste Wege in gewichteten Graphen

Definition

Gewichteter Graph: Ein Graph heißt (Kanten-)gewichtet, wenn jeder Kante e ein reelles Gewicht $w(e)$ zugeordnet wird.

Definition

Sei G ein gewichteter Graph. Die *Länge des Pfads* $p = (e_1, e_2, \ldots, e_k)$ ist $w(p) = \sum_{i=1}^{k} w(e_i)$.

Wir möchten kürzeste (bzw. “leichteste”) Wege von einem Knoten s aus berechnen.

Breitensuche löst dieses Problem für den Fall, dass jede Kante das gleiche Gewicht hat.

Es sei $d_s(v)$ die Länge eines kürzesten Wegs von s nach v.

Grundidee für die Berechnung

Für jeden Knoten v wird aufrechterhalten:

- $\delta[v] \ldots$ Länge des kürzesten schon bekannten Wegs von s
- $\pi[v] \ldots$ der Vorgängerknoten auf diesem Weg

Diese Werte werden initialisiert auf:

$\delta[s] = 0$ und $\delta[v] = \infty$ für $v \neq s$

$\pi[x] = \mathrm{NIL}$ für alle x.

Diese Werte werden mit Hilfe von *Kantenrelaxationen* sukzessive verbessert.

Relaxieren einer Kante $[u, v\rangle$: *relax*(u, v)

$$
\begin{array}{l}
\textbf{if } \delta[v] > \delta[u] + w([u, v\rangle) \textbf{ then} \\
\qquad \delta[v] = \delta[u] + w([u, v\rangle) \\
\qquad \pi[v] = u
\end{array}
$$

Man nennt $[u, v\rangle$ *relaxierbar*, wenn $\delta[v] > \delta[u] + w([u, v\rangle)$, wenn also ihre Relaxation eine Auswirkung auf $\delta[]$ und $\pi[]$ hat.

Naiver Kürzeste-Wege-“Algorithmus”

1. Initialisiere $\delta[]$ und $\pi[]$
2. Solange es eine relaxierbare Kante gibt,
   wähle eine und relaxiere sie.

Probleme:

- Ist das überhaupt ein Algorithmus, also, terminiert diese Methode überhaupt?
  Nein, nicht wenn es einen negativen Zyklus gibt, der von s erreichbar ist.
- Wenn diese Methode terminiert, wie lange braucht sie?
  Das hängt stark von der Wahl der zu relaxierenden Kante ab.

Bellman-Ford-Algorithmus

Idee: Finde für steigendes k die kürzesten Wege mit höchstens k Kanten.

Lemma

Es sei $e_1, \ldots, e_\ell$ ein kürzester Weg von s nach v.
Wenn eine Folge R von Kantenrelaxationen relax(e_1),relax(e_2),$\ldots$,relax(e_ℓ) als Teilfolge enthält (nicht unbedingt zusammenhängend), dann wurde mit R ein kürzester Weg von s nach v berechnet, $\delta[v] = d_s(v)$ und die entsprchenden Vorgänger $\pi[]$ ergeben diesen Weg.

Bellman-Ford-Algorithmus

Idee: Finde für steigendes k die kürzesten Wege mit höchstens k Kanten.

Wenn es keine negativen Zyklen gibt, besteht jeder kürzeste Weg aus höchstens $n - 1$ Kanten.

Bellman-Ford-Algorithmus:

1. Initialisiere $\delta[]$ und $\pi[]$
2. wiederhole $n - 1$ mal
   relaxiere jede Kante
3. Falls es noch immer eine relaxierbare Kante gibt, dann gibt es einen von s erreichbaren negativen Zyklus. Andernfalls gitl für alle Knoten x, dass $\delta[x] = d_s(x)$ und $\pi[]$ gibt einen kürzesten Wege Baum.

Bellman-Ford-Algorithmus

Bellman-Ford-Algorithmus:

1. Initialisiere $\delta[]$ und $\pi[]$
2. wiederhole $n-1$ mal
   relaxiere jede Kante
3. Falls es noch immer eine relaxierbare Kante gibt, dann gibt es einen von s erreichbaren negativen Zyklus. Andernfalls gitl für alle Knoten x, dass $\delta[x] = d_s(x)$ und $\pi[]$ gibt einen kürzesten Wege Baum.

Laufzeit:

1. $O(n)$
2. $n-1$ mal $O(m)$, also $O(n \cdot m)$.
3. $O(m)$.

Insgesamt $O(n \cdot m)$.

Bellman-Ford-Algorithmus, Erweiterung

Ziel: Wollen für jeden Knoten x feststellen, ob

1. $d_s(x) = \infty$, also x von s nicht erreichbar, oder
2. $d_s(x) = -\infty$, also x von s über neg. Zyklus erreichbar, oder
3. $d_s(x) \in \mathbb{R}$ sonst.

Der Fall 1 erledigt sich von selbst.
Die Knoten von Fall 2 kann man feststellen, indem man von jedem von s erreichbaren negativen Zyklus einen Knoten wählt, und von dieser Knotenmenge X alle erreichbaren Knoten berechnet (z.B. über DFS).
Wie kann man diese Knotenmenge X berechnen?

Bellman-Ford-Algorithmus, Erweiterung

Lemma: Wenn $v_0, v_1, \ldots, v_{k-1}$ einen Zyklus Z mit negativen Kosten bildet, dann ist mindestens eine der Kanten $[v_k, v_{k+1}\rangle$ relaxierbar. Die Knotenindices werden hier modulo k genommen.

Beweis: $[u, v\rangle$ relaxierbar bedeutet, dass $\delta[u] + w([u, v\rangle) < \delta(v)$, also

$$c_\delta(u, v) = \delta[u] + w([u, v\rangle) - \delta(v) < 0\,.$$

Wenn alle Kanten des Zyklus Z nicht relaxiebar wären, dann würde gelten $c_\delta(v_i, v_{i+1}) \geq 0$ für $0 \leq i < k$ und daher

$$0 \leq \sum_{0 \leq i < k} c_\delta(v_i, v_{i+1}) = \sum_{0 \leq i < k} (\delta[v_i] + w([v_i, v_{i+1}\rangle) - \delta(v_{i+1}))$$

$$= \sum_{0 \leq i < k} w([v_i, v_{i+1}\rangle)\,, \qquad \text{(teleskopierende Summe)}$$

was aber nicht sein, da ja Z ein Zyklus mit negativem Gesamtgewicht sein soll.

Bellman-Ford-Algorithmus, Erweiterung

Erweiterter Bellman-Ford-Algorithmus:

1. Initialisiere $\delta[]$ und $\pi[]$
2. wiederhole $n-1$ mal
   relaxiere jede Kante
3. Teste jede Kante $[u, v\rangle \in E$, ob sie relaxierbar ist, wenn ja, füge v in Menge X ein.
4. Berechne alle von X erreichbaren Knoten und setze deren $\delta[]$ Wert zu $-\infty$.

Laufzeit:

1. $O(n)$
2. $n-1$ mal $O(m)$, also $O(n \cdot m)$.
3. $O(m)$.
4. $O(n+m)$.

Insgesamt $O(n \cdot m)$.