10 Kürzeste Wege

Definition 10.1. Es sei D ein Digraph und $c : A(D) \to \mathbb{R}$ eine Funktion.

Der *gerichtete Abstand* $\text{dist}_{(D,c)}(u, v)$ *von einem Knoten* u *zu einem Knoten* v *in* D *bzgl.* c ist das Minimum von $c(A(P))$ über alle gerichteten Wege P in D von u nach v. Die Funktion c heißt *konservativ*, wenn es in D keinen gerichteten Kreis C mit negativem Gewicht $c(A(C))$ gibt.

Analog definiert man $\text{dist}_{(G,c)}(u, v)$ für Graphen G und Funktionen $c : E(G) \to \mathbb{R}$ sowie den Begriff einer konservativen Funktion.

Lemma 10.2. *Sei* D *ein Digraph. Eine Funktion* $c : A(D) \to \mathbb{R}$ *ist genau dann konservativ, wenn es eine Funktion* $p : V(D) \to \mathbb{R}$ *gibt, für die*

$$c((u, v)) \geq p(v) - p(u)$$

für alle $(u, v) \in A(D)$ *gilt.*

Eine solche Funktion p *nennt man ein Potential für* (D, c).

□

Lemma 10.3. *Sei* D *ein Digraph,* $c : A(D) \to \mathbb{R}$ *konservativ und* $k \in \mathbb{N}$.

Ist $P : u_0 \dots u_{l-1}u_l$ *mit* $u_0 \neq u_l$ *unter allen gerichteten Wegen in* D *von* u_0 *nach* u_l *mit höchstens* k *gerichtete Kanten ein bzgl.* c *kürzester, dann ist* $P' : u_0 \dots u_{l-1}$ *unter allen gerichteten Wegen in* D *von* u_0 *nach* u_{l-1} *mit höchstens* $k-1$ *gerichtete Kanten ein bzgl.* c *kürzester.*

□

Algorithmus 10.4. Dijkstra's Algorithm

Input: *Ein Digraph* D, $c : A(D) \to \mathbb{R}_{>0}$ *und* $s \in V(D)$.

Output: *Kürzeste Wege von* r *zu allen erreichbaren Knoten:* $(l(v), p(v))_{v \in V(D) \setminus \{s\}}$, *wobei* $l(v) = \text{dist}_{(D,c)}(s, v)$ *gilt und* $p(v)$ *ein Vorgänger von* v *auf einem bzgl.* c *kürzesten Weg von* s *nach* v *ist. Ist* v *nicht erreichbar, so ist* $l(v) = \infty$ *und* $p(v)$ *nicht definiert.*

begin
  $l(s) \leftarrow 0$;
  for $v \in V(D) \setminus \{s\}$ **do** $l(v) \leftarrow \infty$;
1 $R \leftarrow \emptyset$;
  while $R \neq V(D)$ **do**
2 *Wähle* $v \in V(D) \setminus R$ *mit* $l(v) = \min\{l(w) \mid w \in V(D) \setminus R\}$;
3 $R \leftarrow R \cup \{v\}$;
    for $w \in V(D) \setminus R$ *mit* $(v, w) \in A(D)$ **do**
      if $l(w) > l(v) + c((v, w))$ **then**
4 $l(w) \leftarrow l(v) + c((v, w))$; $p(w) \leftarrow v$;
      end
    end
  end
  return $(l(v), p(v))_{v \in V(D) \setminus \{s\}}$;
end

Satz 10.5 (Dijkstra 1959)**.** Dijkstra's Algorithm *arbeitet korrekt.*

□

Bemerkung 10.6. Naiv implementiert hat Dijkstra's Algorithm eine Laufzeit von $O(n(D)^2)$. Mit Fibonacci-Heaps erreicht man $O(m(D)+n(D)\log(n(D)))$. In ungerichteten Graphen mit ganzzahligen Kantengewichten kann man kürzeste Wege in linearer Zeit $O(n(G)+m(G))$ finden (Thorup 1999).

Algorithmus 10.7. Moore-Bellman-Ford Algorithmus

Input*: Ein Digraph D, eine konservative Funktion $c : A(D) \to \mathbb{R}$ und $s \in V(D)$.*
Output*: Wie beim* Dijkstra's Algorithm*: $(l(v), p(v))_{v \in V(D)\setminus\{r\}}$ wobei $l(v) = dist_{(D,c)}(s, v)$ gilt und $p(v)$ ein Vorgänger von v auf einem kürzesten Weg von s nach v ist. (Ist v nicht erreichbar, so ist $l(v) = \infty$ und $p(v)$ nicht definiert.)*

```
begin
    l(s) ← 0;
    for v ∈ V(D) \ {s} do
        l(v) ← ∞;
    end
    for i = 1 to n(D) − 1 do
        for (v, w) ∈ A(D) do
            if l(w) > l(v) + c((v, w)) then
                l(w) ← l(v) + c((v, w)); p(w) ← v;
            end
        end
    end
    return (l(v), p(v))_{v∈V(D)\{s}};
end
```

Satz 10.8 (Moore 1959, Bellman 1958, Ford 1956)**.** *Der* Moore-Bellman-Ford Algorithmus *arbeitet korrekt, und seine Laufzeit beträgt $O(m(D)n(D))$.*

□

Algorithmus 10.9. FLOYD-WARSHALL ALGORITHMUS

Input: *Ein Digraph D mit $V(D) = [n]$ und eine konservative Gewichtsfunktion $c : A(D) \to \mathbb{R}$.*

Output: *Matrizen $(l(i,j))_{(i,j)\in[n]^2}$ und $(p(i,j))_{(i,j)\in[n]^2}$. Hier bei gilt $l(i,j) = \text{dist}_{(D,c)}(i,j)$ und für $(i,j) \in [n^2]$ mit $l(i,j) < \infty$ ist $(p(i,j), j)$ die letzte gerichte Kante eines bezüglich c kürzesten gerichteten Weges in D von i nach j.*

```
begin
    for (i,j) ∈ A(D) do l(i,j) ← c((i,j)); p(i,j) ← i;
    for i,j ∈ V(D) mit i ≠ j und (i,j) ∉ A(D) do l(i,j) ← ∞;
    for i ∈ V(D) do l(i,i) ← 0;
    for j = 1 to n do
        for i = 1 to n do
            if i ≠ j then
                for k = 1 to n do
                    if k ≠ j then
                        if l(i,k) > l(i,j) + l(j,k) then
                            l(i,k) ← l(i,j) + l(j,k); p(i,k) ← p(j,k);
                        end
                    end
                end
            end
        end
    end
end
```

Lemma 10.10. *Sei D ein Digraph und $c : A(D) \to \mathbb{R}$ eine Funktion. Sind $P : u_0 \dots u_p$ und $Q : v_0 \dots v_q$ zwei gerichtete Wege in D mit $u_p = v_0$, so existiert ein gerichteter Weg R in D von u_0 nach v_q sowie $s \geq 0$ gerichtete Kreise $C_1, \dots, C_s$ in D mit*

$$c(A(P)) + c(A(Q)) = c(A(R)) + c(A(C_1)) + \dots + c(A(C_s)).$$

Haben P und Q mehr als die Ecke u_p gemeinsam, so kann R so gewählt werden, dass $u_p \notin V(R)$ gilt.

□

Satz 10.11 (Floyd 1962, Warshall 1962)**.** *Der* FLOYD-WARSHALL ALGORITHMUS *arbeitet korrekt in $O(n^3)$ Laufzeit.*

□