

Grundlagen der theoretischen Informatik

Vorlesung 4: Günstigste Wege

Günstigste Wege

Definition

Sei $G = (V, E)$ ein gerichteter (oder ungerichteter) Graph. Für $e \in E$ bezeichne $k(e)$ die Kosten (Länge) der Kante e . Für einen Kantenzug W bezeichne $k(W)$ die Kosten (Länge) des Kantenzuges, definiert als die Summe der Kosten der enthaltenen Kanten. Der *günstigste Weg* (*legrövidebb út*) zwischen zwei Knoten ist der Weg zwischen ihnen mit minimalen Kosten. Für $x, y \in V$ sei der *Abstand* von x, y

$$d(x, y) = \begin{cases} 0, & \text{falls } x = y; \\ \infty, & \text{falls es keinen} \\ & x \sim y \text{ Weg gibt;} \\ \text{Kosten des günstigsten } x \sim y \text{ Weges,} & \text{sonst.} \end{cases}$$

Definition

Eine Gewichtsfunktion heisst *konservativ* für den Graphen G , falls es in G keinen gerichteten Kreis K mit $k(K) < 0$ gibt.

Lemma

Sei $G = (V, E)$ ein gerichteter Graph, $k : E \rightarrow \mathbb{R}$ konservativ. Wenn es einen Kantenzug mit Kosten L zwischen den Knoten x und y gibt, dann gilt $d(x, y) \leq L$.

Beweis. Man kann aus dem Kantenzug eventuelle Zyklen entfernen, bis er ein $x \sim y$ -Weg wird. Da die Kosten der entfernten Kreise nicht negativ sind, werden dadurch die Kosten des Kantenzuges nicht erhöht.

Korollar

Sei G und k wie im Lemma. Sei W ein günstigster Weg zwischen den Knoten x und y . Sei z ein weiterer Knoten in W . Dann ist $W[x, z]$ ein günstigster Weg zwischen x und z .

Relax-Schritt

Definition

Seien G, k wie oben, s ein ausgezeichneteter Knoten von G . Sei $D : V \rightarrow \mathbb{R}$ eine Funktion, die wie folgt initialisiert wird:

$$D(v) = \begin{cases} 0, & \text{falls } v = s; \\ \infty, & \text{falls } sv \notin E; \\ k(sv), & \text{sonst.} \end{cases}$$

Sei $xy \in E$, dann ist der *Relax-Schritt* für die Kante xy :

$\text{Relax}(xy)$: wenn $D(y) > D(x) + k(xy)$, dann sei $D(y) = D(x) + k(xy)$.

Lemma

Nach einer beliebigen Sequenz von Relax-Schritten gilt für alle Knoten: $D(v) \geq d(s, v)$. Wenn für einen Knoten $D(v) = d(s, v)$ gilt, dann ändert sich $D(v)$ bei weiteren Relax-Schritten nicht mehr

Beweis. Vollständige Induktion. Nach der Initialisierung von D gilt die Aussage. Man muss nur beweisen, dass sie nach jeder Änderung von D auch gilt. Wenn bei $\text{Relax}(xy)$ der Wert von $D(y)$ geändert wird, dann gilt:

$$D^{\text{neu}}(y) = D(x) + k(xy) \geq d(s, x) + k(xy) \geq d(s, y).$$

$d(s, x)$ ist gleich den Kosten eines $s \sim x$ Weges, der zusammen mit der Kante xy einen $s \sim y$ Kantenzug mit Kosten $d(s, x) + k(xy)$ ergibt.

Die zweite Aussage folgt daraus, dass $D(v)$ bei einem Relax-Schritt immer nur verringert werden kann, was in diesem Fall nicht mehr möglich ist

Algorithmus von Dijkstra

Seien alle Kosten nicht-negativ.

Algorithmus von Dijkstra

Input: $G = (V, E)$ gerichteter/ungerichteter Graph, $k : E \rightarrow \mathbb{R}_0^+$, $s \in V$.

Output: $d(s, v)$ für alle $v \in V$.

Ablauf: Initialisierung: D wird initialisiert. Sei weiterhin $F = \{s\}$. In einer allgemeinen Iteration macht man Folgendes:

- (1) Man wählt $v \in V \setminus F$, wofür $D(v)$ minimal ist.
 - (2) $F = F \cup \{v\}$.
 - (3) Für jedes $x \in V \setminus F$ mit $vx \in E$ wird $\text{Relax}(vx)$ durchgeführt.
- Der Algorithmus terminiert wenn $F = V$, nach $n - 1$ Iterationen.

Bemerkung: Dieser Algorithmus kann so erweitert werden, dass er nicht nur die Kosten der günstigsten Wege, sondern die günstigsten Wege selbst liefert.

Bemerkung: Die Anzahl der Schritte ist $\text{konst} \cdot n^2$.

Satz

Während der gesamten Durchführung des Algorithmus gilt für alle Knoten in F : $D(v) = d(s, v)$.

Algorithmus von Bellman und Ford

Algorithmus von Bellman und Ford

Input: $G = (V, E)$ gerichteter Graph, $s \in V$, $k : E \rightarrow \mathbb{R}$ konservativ.

Output: $d(s, v)$ für alle $v \in V$.

Ablauf: Initialisierung: D wird initialisiert.

Wiederhole $n - 1$ -mal:

Wiederhole für jede Kante xy : $\text{Relax}(xy)$.

Bemerkung: Der Algorithmus funktioniert auch für ungerichtete Graphen, aber die Längefunktion soll dann nicht-negativ sein. Dieser Algorithmus kann so erweitert werden, dass er nicht nur die Kosten der günstigsten Wege, sondern die günstigsten Wege selbst liefert.

Satz

Man betrachte einen Zwischenstand nach j Iterationen der äußeren Schleife in der Durchführung des Algorithmus. Sei v ein Knoten, für den es einen günstigsten $s \sim v$ Weg W gibt, der aus höchstens j Kanten besteht. Dann gilt $D(v) = d(s, v)$.

Korollar

Nach $n - 1$ Iterationen der äußeren Schleife liefert der Algorithmus von Bellman und Ford für alle Knoten $D(v) = d(s, v)$.

Beweis. Ein Weg besteht aus höchstens $n - 1$ Kanten.

Bemerkung: Dieser Algorithmus kann so erweitert werden, dass er nicht nur die Kosten der günstigsten Wege, sondern die günstigsten Wege selbst liefert.

Bemerkung: Die Anzahl der Schritte ist *konst* · nm .

Algorithmus von Floyd

Aufgabe: Berechnung der günstigsten Wege zwischen allen Knotenpaaren.

Definition

Sei $G = (V, E)$ ein gerichteter Graph, $k : E \rightarrow \mathbb{R}$ eine konservative Kostenfunktion. Sei weiterhin $V = \{v_1, v_2, \dots, v_n\}$. Dann bezeichne man für $1 \leq x, y \leq n$, $0 \leq z \leq n$ mit $D^{(z)}(x, y)$ die Kosten des günstigsten $x \sim y$ Weges, in dem alle inneren Knoten in $v_1, \dots, v_z$ sind.

Algorithmus von Floyd

Input: $G = (V, E)$ gerichteter Graph, $k : E \rightarrow \mathbb{R}$ konservativ.

Output: $d(x, y)$ für alle $x, y \in V$.

Ablauf: Initialisierung:

$$D^{(0)}(x, y) = \begin{cases} 0, & \text{falls } x = y; \\ \infty, & \text{falls } xy \notin E; \\ k(xy), & \text{sonst.} \end{cases}$$

Wiederhole für $z = 1, 2, \dots, n$:

Wiederhole für alle Paare $x, y \in V$:

$$D^{(z)}(x, y) = \min\{D^{(z-1)}(x, y), D^{(z-1)}(x, z) + D^{(z-1)}(z, y)\}.$$

Output: $D^{(n)}(x, y)$ für alle $x, y \in V$.

Bemerkung: Der Algorithmus funktioniert auch für ungerichtete Graphen, aber die Längefunktion soll dann nicht-negativ sein.

Satz

Der Algorithmus von Floyd berechnet korrekt die Kosten der günstigsten Wege zwischen allen Knotenpaaren.

Bemerkung: Dieser Algorithmus kann so erweitert werden, dass er nicht nur die Kosten der günstigsten Wege, sondern die günstigsten Wege selbst liefert.

Bemerkung: Die Anzahl der Schritte ist $\textit{konst} \cdot n^3$.