Institut für Theoretische Informatik
Lehrstuhl Prof. Dr. D. Wagner

Theorie-Übungsblatt 2 – Lösungsvorschläge

Algorithmen für Routenplanung, Sommer 2010

Problem 1: Bidirektionale Suche **

Sei $G = (V, E, \text{len})$ ein gerichteter, gewichteter Graph mit eindeutigen kürzesten Wegen, das heißt für je zwei Paare $s, t \in V$ existiert genau ein kürzester s-t-Weg. Bei Bidirektionaler Suche werden nun für eine s-t-Anfrage abwechselnd eine Vorwärts-Suche beginnend bei s und eine Rückwärts-Suche beginnend bei t durchgeführt.

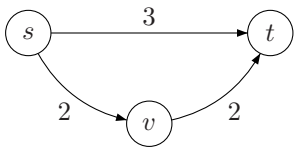
Ein zur Vorlesung alternatives Abbruchkriterium sei wie folgt definiert. Seien $\overrightarrow{S}$ bzw. $\overleftarrow{S}$ die Menge abgearbeiteter Knoten (settled nodes) von der Vorwärts-, bzw. Rückwärtssuche. Sobald eine der beiden Suchen einen Knoten $v \in V$ abarbeitet der bereits von der anderen Suche abgearbeitet wurde, also $|\overrightarrow{S} \cap \overleftarrow{S}| = 1$ erreicht wird, terminiert der Algorithmus und der induzierte Weg

$$P := [s, \ldots, v, \ldots, t]$$

wird ausgegeben.

(a) Zeigen Sie, dass P nicht notwendigerweise der kürzeste s-t-Weg ist.

Lösung. Folgender kleiner Graph liefert ein Gegenbeispiel. Der Ablauf des Algorithmus ist wie folgt.



Vorwärtssuche: `settle`(s)

- `insert`(v,2)
- `insert`(t,3)

Rückwärtssuche: `settle`(t)

- `insert`(s,2)
- `insert`(v,3)

Vorwärtssuche: `settle`(v)

Rückwärtssuche: `settle`(v)

- `stop`. Ausgegebener Pfad ist $P = [s, v, t]$ mit Länge 4.

Der kürzeste Weg P' ist jedoch $P' = [s, t]$ mit Länge 3. □

(b) Sei P' der kürzester Weg von s nach t. Zeigen Sie, dass P' nur Knoten aus $\overrightarrow{S} \cup \overleftarrow{S}$ nach obigem Abbruchkriterium enthält (Der kürzeste Weg ist also im Suchraum enthalten).

Lösung. Wir beweisen die Aussage durch Widerspruch. Sei also $M \in V$ der durch das Abbruchkriterium induzierte Knoten mit $M \in \overrightarrow{S} \cap \overleftarrow{S}$. Angenommen P' enthält einen Knoten $v \in V$ mit $v \notin \overrightarrow{S} \cup \overleftarrow{S}$. Aus der Korrektheit von DIJKSTRA's Algorithmus folgt

$$\begin{aligned} u \in \overrightarrow{S} \quad &\Leftrightarrow \quad \operatorname{dist}(s,u) \leq \operatorname{dist}(s,M), \\ u \in \overleftarrow{S} \quad &\Leftrightarrow \quad \operatorname{dist}(u,t) \leq \operatorname{dist}(M,t). \end{aligned}$$

Damit gilt

$$\begin{aligned} &\operatorname{dist}(s,v) > \operatorname{dist}(s,M) \text{ und} \\ &\operatorname{dist}(v,t) > \operatorname{dist}(M,t). \end{aligned}$$

Also folgt sofort $\operatorname{len} P < \operatorname{len} P'$ was ein Widerspruch dazu ist, dass P' der kürzeste Weg ist. □

(c) Geben Sie einen Algorithmus in Pseudocode an, der auf obigem Abbruchkriterium beruht und dennoch den kürzesten Weg findet.

Lösung. Algorithmus 1 implementiert eine korrekte bidirektionale Suche basierend auf unserem Abbruchkriterium.

Die Notation sei wie folgt. Für die Warteschlangen Q, die Distanzen dist und die Menge abgearbeiteter Knoten S bezeichne $\overline{Q}$, $\overline{\text{dist}}$ und $\overline{S}$ die jeweiligen Größen für die entgegengesetzte Suche. Ist also $Q = \overrightarrow{Q}$, so bezeichnet $\overline{Q} = \overleftarrow{Q}$. Die Abwechslungsstrategie der beiden Suchen wird über `chooseDirection()` gesteuert und kann beliebig gewählt werden.

□

(d) Studieren Sie das Abbruchkriterium aus der Vorlesung. Zeigen Sie, dass das hier eingeführte Kriterium sogar *schwächer* als das aus der Vorlesung ist, dass also von jeder der beiden Suchen mindestens so viele Knoten abgearbeitet werden wie bei dem Kriterium aus der Vorlesung.

Lösung. Wir beweisen die Aussage in dem wir zeigen dass gilt

$$|\overrightarrow{S} \cap \overleftarrow{S}| = 1 \Rightarrow \operatorname{minKey}(\overrightarrow{Q}) + \operatorname{minKey}(\overleftarrow{Q}) \geq \mu.$$

Sei also $v \in \overrightarrow{S} \cap \overleftarrow{S}$ der durch unser Abbruchkriterium induzierte Knoten. Aus dem Ablauf von DIJKSTRA's Algorithmus folgt

$$\begin{aligned} &\operatorname{minKey}(\overrightarrow{Q}) \geq \operatorname{dist}(s,v) \text{ und} \\ &\operatorname{minKey}(\overleftarrow{Q}) \geq \operatorname{dist}(v,t). \end{aligned}$$

Damit folgt unmittelbar

$$\begin{aligned} \operatorname{minKey}(\overrightarrow{Q}) + \operatorname{minKey}(\overleftarrow{Q}) &\geq \operatorname{dist}(s,v) + \operatorname{dist}(v,t) \\ &\overset{\text{(b)}}{\geq} \mu. \end{aligned}$$

□

Algorithmus 1 : Bidirektionale Suche

Eingabe : Graph $G = (V, E, \text{len})$ und Start- und Zielknoten $s, t \in V$
Ausgabe : Der kürzeste Weg von s nach t

$\overrightarrow{Q}, \overleftarrow{Q} \leftarrow$ Vorwärts- und Rückwärtsschlangen
$\overrightarrow{\text{dist}}, \overleftarrow{\text{dist}} \leftarrow \infty$
$\overrightarrow{S}, \overleftarrow{S} \leftarrow \emptyset$
$\overrightarrow{Q}$.`insert(`s`,0)`, $\overrightarrow{\text{dist}}(s) \leftarrow 0$
$\overleftarrow{Q}$.`insert(`t`,0)`, $\overleftarrow{\text{dist}}(t) \leftarrow 0$
solange nicht $\overrightarrow{Q}$.`empty()` **oder nicht** $\overleftarrow{Q}$.`empty()` **tue**
    $Q \leftarrow$ `chooseDirection()`
    $v \leftarrow Q$.`deleteMin()`
    $S \leftarrow S \cup \{v\}$
    wenn $v \in \overline{S}$ **dann**
        gib aus Pfad $[s, \ldots, M, \ldots, t]$ der Länge μ
        stop
    für alle *ein- bzw. ausgehenden Kanten* $e = (v, w)$ *bzw.* $e = (w, v)$ **tue**
        wenn $w \in \overline{Q}$ **oder** $w \in \overline{S}$ **dann**
            wenn $\mu > \text{dist}(v) + \text{len}(e) + \overline{\text{dist}}(w)$ **dann**
                $\mu \leftarrow \text{dist}(v) + \text{len}(e) + \overline{\text{dist}}(w)$
                $M \leftarrow w$
        wenn $\text{dist}(w) = \infty$ **dann**
            Q.`insert(`w, $\text{dist}(v) + \text{len}(e)$`)`
            $\text{dist}(w) \leftarrow \text{dist}(v) + \text{len}(e)$
        sonst wenn $\text{dist}(v) + \text{len}(e) < \text{dist}(w)$ **dann**
            Q.`decreaseKey(`w, $\text{dist}(v) + \text{len}(e)$`)`
            $\text{dist}(w) \leftarrow \text{dist}(v) + \text{len}(e)$
gib aus Kein Pfad gefunden.

Problem 2: A* Algorithmus **

Sei $G = (V, E, \text{len})$ ein gerichteter, gewichteter Graph. Weiterhin sei $\pi : V \to \mathbb{R}$ eine zunächst beliebige Potentialfunktion. Wie in der Vorlesung eingeführt, seien die *reduzierten Kosten* len_π definiert durch $\text{len}_\pi(u, v) := \text{len}(u, v) - \pi(u) + \pi(v)$. Analog bezeichne $G_\pi := (V, E, \text{len}_\pi)$ den Graph mit reduzierten Kosten.

(a) Zeigen Sie: Dijkstra's Algorithmus auf G_π entspricht genau A* mit Potentialen π.

Lösung. Bevor wir die Aussage beweisen zeigen wir zunächst ein Lemma.
Lemma 1. *Sei* $P := [v_1, \ldots, v_k]$ *ein Pfad in* G*. Dann gilt in* G_π

$$\text{len}_\pi P = \text{len} P - \pi(v_1) + \pi(v_k).$$

Beweis. Der Beweis erfolgt durch nachrechnen:

$$\begin{aligned}\operatorname{len}_\pi P &= \sum_{i=1}^{k-1} \operatorname{len}_\pi(v_i, v_{i+1}) \\ &= \sum_{i=1}^{k-1} \big(\operatorname{len}(v_i, v_{i+1}) - \pi(v_i) + \pi(v_{i+1})\big) \\ &= \Big(\sum_{i=1}^{k-1} \operatorname{len}(v_i, v_{i+1})\Big) - \pi(v_1) + \pi(v_k) \\ &= \operatorname{len} P - \pi(v_1) + \pi(v_k)\end{aligned}$$

□

Beide Algorithmen arbeiten gleich, genau dann wenn sie die Knoten in der gleichen Reihenfolge abarbeiten. Für zwei Knoten $u, v \in V$ bezeichne $u \prec v$ dass u vor v abgearbeitet wird. Es gilt:

$$\begin{aligned}\textsc{Dijkstra}: \quad u \prec v \quad &\Leftrightarrow \quad \operatorname{dist}_\pi(s,u) < \operatorname{dist}_\pi(s,v) \\ \text{A*}: \quad u \prec v \quad &\Leftrightarrow \quad \operatorname{dist}(s,u) + \pi(u) < \operatorname{dist}(s,v) + \pi(v)\end{aligned}$$

Damit ist nun

$$\begin{aligned}&\operatorname{dist}_\pi(s,u) < \operatorname{dist}_\pi(s,v) \\ \overset{\text{lem}}{\Leftrightarrow}\ &\operatorname{dist}(s,u) - \pi(s) + \pi(u) < \operatorname{dist}(s,v) - \pi(s) + \pi(v) \\ \Leftrightarrow\ &\operatorname{dist}(s,u) + \pi(u) < \operatorname{dist}(s,v) + \pi(v).\end{aligned}$$

□

(b) Welche Anforderung(en) müssen an π zusätzlich gestellt werden, damit beide Algorithmen den kürzesten Weg finden? Begründen Sie Ihre Behauptung(en).

Lösung. Dijkstra's Algorithmus ist korrekt wenn der Graph keine negativen Zyklen enthält. Wir stellen also sicher, dass alle reduzierten Kosten ≥ 0 sind. Also

$$\begin{aligned}\operatorname{len}_\pi(u,v) \geq 0 &\Leftrightarrow \operatorname{len}(u,v) + \pi(u) - \pi(v) \geq 0 \\ &\Leftrightarrow \operatorname{len}(u,v) + \pi(u) \geq \pi(v).\end{aligned}$$

□

Problem 3: Der ALT-Algorithmus ***

Gegeben sei ein gerichteter, gewichteter Graph $G = (V, E, \operatorname{len})$ mit $|V| = n$ und eindeutigen kürzesten Wegen. Weiterhin sei $\ell \in V$ *eine* Landmarke.

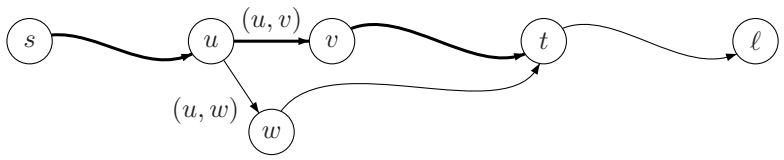
(a) Zeigen Sie, dass für s-t-Anfragen, wobei t auf dem kürzesten Weg von s nach ℓ liegt, der ALT-Algorithmus ausschließlich Knoten entlang des kürzesten s-t-Weges abarbeitet.

Lösung. Um die Aussage zu beweisen betrachten wir wieder den Graphen mit reduzierten Kosten G_π unter Dijkstra's Algorithmus. Um zu beweisen dass Dijkstra's Algorithmus nur Knoten entlang des kürzesten Weges abarbeitet, beweisen wir zwei Aussagen.

1. Ist $(u, v) \in E$ eine Kante auf dem kürzesten s-t-Weg, dann gilt $\text{len}_\pi(u, v) = 0$.

2. Ist $u \in V$ ein Knoten auf dem kürzesten s-t-Weg und $(u, w) \in E$ eine *abzweigende* Kante, das heißt (u, w) liegt nicht auf dem kürzesten s-t-Weg, so ist $\text{len}_\pi(u, w) > 0$.

Damit ergibt sich $\text{dist}_\pi(s, t) = 0$ und $\text{dist}_\pi(s, w) > 0$ für alle Knoten $w \in V$ die nicht auf dem kürzesten s-t-Weg liegen. Das heißt DIJKSTRA's Algorithmus auf G_π arbeitet nur Knoten entlang des kürzesten Weges ab.



Um den Beweis zu vereinfachen zeigen wir zunächst folgende Hilfsaussage.
Lemma 2. *Sei $u \in V$ und der Knoten t liegt auf dem kürzesten Weg von u nach ℓ. Dann ist das Potential von u gerade $\pi(u) = \text{dist}(u, t)$.*

Beweis. Wir rechnen nach:

$$\begin{aligned}\pi(u) &= \max(\underbrace{\text{dist}(u, \ell) - \text{dist}(t - \ell)}_{=\text{dist}(u,t)}, \text{dist}(\ell, t) - \text{dist}(\ell, u)) \\ &= \text{dist}(u, t)\end{aligned}$$

Der zweite Eintrag der Maximumsbildung kann nicht weiter zur Verbesserung des Potentials beitragen, da $\text{dist}(u, t)$ bereits die bestmögliche untere Schranke (Dreiecksungleichung) für den Abstand von u nach t ist. □

Wir zeigen nun die zwei Aussagen:

1. Nachrechnen liefert für Kanten $(u, v) \in E$ auf dem kürzesten s-t-Weg:

$$\begin{aligned}\text{len}_\pi(u, v) &= \text{len}(u, v) - \pi(u) + \pi(v) \\ &= \text{len}(u, v) - \text{dist}(u, t) + \text{dist}(v, t) \\ &= \text{len}(u, v) - \text{len}(u, v) \\ &= 0\end{aligned}$$

2. Die Aussage $\text{len}_\pi(u, w)$ für abzweigende Kanten $(u, w) \in E$ vom kürzesten Weg zeigen wir durch Widerspruch. Nehmen wir also an, dass $\text{len}_\pi(u, w) = 0$ gilt, das heißt

$$\begin{aligned}& 0 = \text{len}_\pi(u, w) \\ \Rightarrow\quad & 0 = \text{len}(u, w) - \pi(u) + \pi(w) \\ \Rightarrow\quad & 0 = \text{len}(u, w) - \text{dist}(u, t) + \pi(w) \\ \Rightarrow\quad & \text{dist}(u, t) = \text{len}(u, w) + \pi(w)\end{aligned}$$

   Da für $\pi(w)$ gelten muss $\pi(w) \leq \text{dist}(w, t)$ erhalten wir für den Fall $\pi(w) = \text{dist}(w, t)$ dass $\text{dist}(u, t) = \text{len}(u, w) + \text{dist}(w, t)$ gilt, das heißt der Weg $[u, w, \ldots, t]$ hat die gleiche

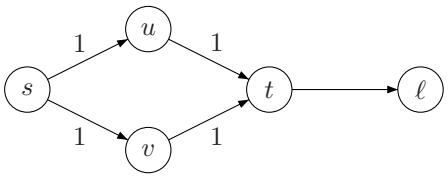
Länge wie der Weg $[u, v, \ldots, t]$. Somit existieren zwei kürzeste Wege von u nach t was nach Aufgabenstellung ausgeschlossen ist.

Bleibt also nur der Fall $\pi(w) < \text{dist}(w, t)$. Damit erhalten wir allerdings $\text{dist}(u, t) > \text{len}(u, w) + \text{dist}(w, t)$, das heißt der Weg über $[u, w, \ldots, t]$ ist kürzer als der Weg $[u, v, \ldots, t]$ was ein Widerspruch zu der Annahme ist, dass (u, v) auf dem kürzesten Weg liegt.

Insgesamt erhalten wir also dass DIJKSTRA's Algorithmus auf G_π nur Kanten entlang des kürzesten s-t-Weges abarbeitet. □

(b) Geben Sie ein Gegenbeispiel zu (a) an, falls die Eigenschaft dass alle kürzesten Wege eindeutig sind nicht gilt.

Lösung. Betrachte folgenden Beispielgraph.

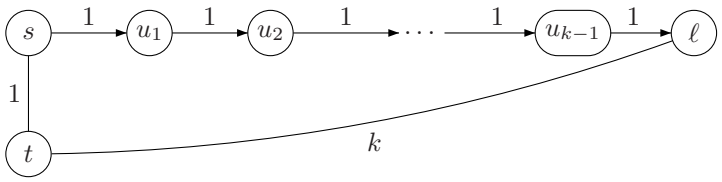


Da die kürzesten Wege nicht eindeutig sind ergeben sich für die Kanten (s, u) und (s, v) jeweils reduzierte Kosten 0. Es ist also möglich, dass der ALT Algorithmus sowohl u als auch v abarbeitet. □

(c) **Knobelaufgabe.**
Zeigen Sie, dass der Suchraum (Anzahl abgearbeiteter Knoten) von ALT um einen Faktor von $\mathcal{O}(n)$ größer sein kann als bei Anwendung des normalen DIJKSTRA Algorithmus.

Hinweis: Finden Sie eine Familie von Graphen mit Knoten s, t, ℓ für die die Suchraumgröße unter DIJKSTRA's Algorithmus konstant ist, die reduzierten Kosten unter dem ALT-Algorithmus jedoch bewirken dass zunächst (fast) der ganze Graph vor t abgearbeitet wird.

Lösung. Wir geben eine Familie $\mathcal{G}$ von Graphen an die die geforderten Eigenschaften erfüllt. Für jedes $k \in \mathbb{N}$ konstruieren wir einen Graphen G_k mit $\Theta(k)$ Knoten wie in der folgenden Abbildung illustriert.

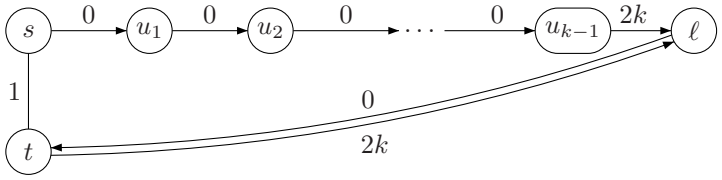


Der Quellknoten s kann mit u_0 und die Landmarke ℓ mit u_k identifiziert werden.

Wir berechnen nun die reduzierten Kosten von G_k. Die Potentiale ergeben sich wie folgt:

$$\begin{aligned}
\pi(u_i) &= \max(\underbrace{\underbrace{\operatorname{dist}(u_i,\ell)}_{=k-i} - \underbrace{\operatorname{dist}(t,\ell)}_{=k}}_{=-i}, \underbrace{\underbrace{\operatorname{dist}(\ell,t)}_{=k} - \underbrace{\operatorname{dist}(\ell,u_i)}_{=k+i+1}}_{=-i-1}) \\
&= -i \\
\pi(\ell) &= \max(\underbrace{\underbrace{\operatorname{dist}(\ell,\ell)}_{=0} - \underbrace{\operatorname{dist}(t,\ell)}_{=k}}_{=-k}, \underbrace{\underbrace{\operatorname{dist}(\ell,t)}_{=k} - \underbrace{\operatorname{dist}(\ell,\ell)}_{=0}}_{=k}) \\
&= k \\
\pi(s) &= \pi(u_0) \\
&= 0 \\
\pi(t) &= \max(\operatorname{dist}(t,\ell) - \operatorname{dist}(t,\ell), \operatorname{dist}(\ell,t) - \operatorname{dist}(\ell,t)) \\
&= 0
\end{aligned}$$

Mit $\operatorname{len}_\pi(u,v) = \operatorname{len}(u,v) - \pi(u) + \pi(v)$ ergeben sich die reduzierten Kosten auf den Graphen wie folgt.



Der Suchraum von Dijkstra's Algorithmus auf dem ursprünglichen Graph besteht demnach nur aus $\{s, u_1, t\}$ während der ALT Algorithmus den Suchraum $V \setminus \{\ell\}$ hat, da die reduzierten Kosten entlang aller Kanten (u_{i-1}, u_i) für $i < k$ gerade 0 sind.

Mit $|V| \in \Theta(k)$ ergibt sich somit eine Suchraumverschlechterung um einen Faktor von $\Theta(|V|)$ des ALT Algorithmus gegenüber dem einfachen Dijkstra. □