Karlsruher Institut für Technologie

Institut für Theoretische Informatik
T.T.-Prof. Dr. Thomas Bläsius
https://scale.iti.kit.edu

Übungsblatt 07

Algorithmen I – Sommersemester 2022

Abgabe im ILIAS bis 15.06.2022, 14:00 Uhr

Bitte beschrifte Deine Abgabe gut sichtbar mit Deinem Namen und Deiner Matrikelnummer. Achte insbesondere bei handschriftlichen Abgaben auf Lesbarkeit und genügend Platz für Korrektur-Anmerkungen. Die Abgabe erfolgt über das Übungsmodul in der Gruppe Deines Tutoriums im ILIAS. Gib Deine Ausarbeitungen in *einer* PDF-Datei ab. Achte darauf, effiziente Algorithmen zu formulieren, also solche mit möglichst geringer asymptotischer Laufzeit!
Wenn du die Korrektheit eines Algorithmus begründen oder dessen Laufzeit analysieren sollst, tue dies getrennt von der Beschreibung des Algorithmus.
Wenn nicht anders spezifiziert oder aus dem Kontext ersichtlich, bezeichnen wir mit Graph einen einfachen ungerichteten Graphen.

Aufgabe 1 - Fahrplanauskunft (7 Punkte)

Eine elementare Zugverbindung C ist ein Tupel $C = (Z, B_{\text{ab}}, B_{\text{an}}, \tau_{\text{ab}}, \tau_{\text{an}})$ mit der Bedeutung, dass der Zug Z zum Zeitpunkt τ_{ab} am Bahnhof B_{ab} losfährt und ohne Zwischenstopp zum Zeitpunkt τ_{an} am Bahnhof B_{an} ankommt. Sei $\mathcal{C}$ eine Menge solcher elementarer Verbindungen und sei $\mathcal{B}$ die Menge aller Bahnhöfe. Eine Routenanfrage besteht aus einem Startbahnhof $B_s \in \mathcal{B}$, einem Zielbahnhof $B_t \in \mathcal{B}$, sowie einem Startzeitpunkt τ_s. Ziel ist es, eine Zugverbindung von B_s nach B_t zu finden, die nicht vor τ_s startet und möglichst früh endet.

1. Modelliere das Problem der Fahrplanauskunft als kürzeste Wege Problem in einem Graphen. Wende dann deine Modellierung auf die folgende Probleminstanz an und zeichne den resultierenden Graphen: (4 Punkte)

$$\mathcal{B} = \{\text{Growth},\ \text{Progress},\ \text{QualityCity}\}$$

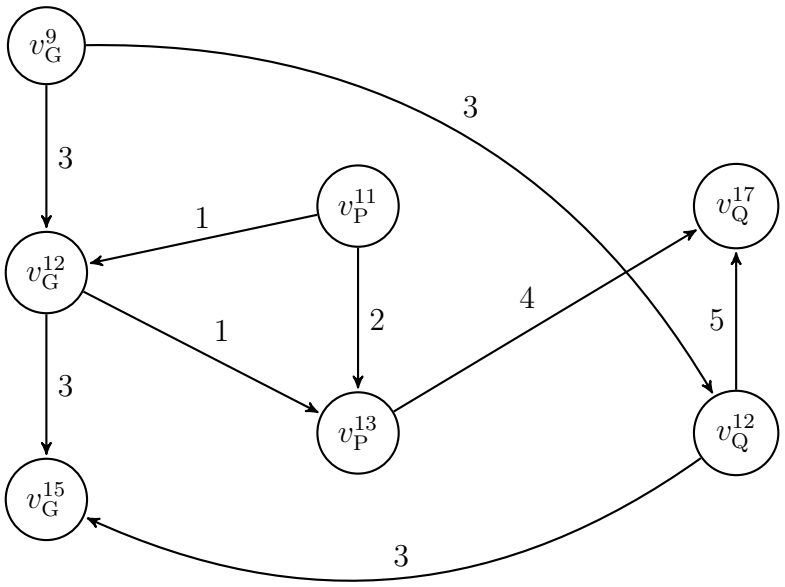
$$
\begin{aligned}
\mathcal{C} = \{&(Z_0, \text{Growth}, \quad \text{QualityCity}, (09\text{ Uhr}), (12\text{ Uhr})) \\
&(Z_1, \text{Growth}, \quad \text{Progress}, \quad (12\text{ Uhr}), (13\text{ Uhr})) \\
&(Z_2, \text{QualityCity}, \text{Growth}, \quad (12\text{ Uhr}), (15\text{ Uhr})) \\
&(Z_3, \text{Progress}, \quad \text{QualityCity}, (13\text{ Uhr}), (17\text{ Uhr})) \\
&(Z_4, \text{Progress}, \quad \text{Growth}, \quad (11\text{ Uhr}), (12\text{ Uhr})) \\
\}&
\end{aligned}
$$

Hinweis: Du darfst annehmen, dass mit Uhrzeiten gerechnet werden kann wie mit ganzen Zahlen.

2. Sei nun für jeden Bahnhof noch eine Umsteigezeit gegeben und für je zwei Bahnhöfe B_1 und B_2 die Dauer, in der man von B_1 nach B_2 laufen kann. Erweitere deine Modellierung so, dass die Umsteigezeiten eingehalten werden und die Fußwege zwischen Bahnhöfen beachtet werden. (3 Punkte)

Lösung 1

1. Wir bauen basierend auf $\mathcal{C}$ und $\mathcal{B}$ einen gewichteten, gerichteten Graphen G auf. Jeder Knoten in G wird zu einem Bahnhof zu einem bestimmten Zeitpunkt korrespondieren. Den Knoten zu $B \in \mathcal{B}$ und dem Zeitpunkt τ notieren wir als v_B^τ.
   Für jede Zugverbindung $C = (Z, B_{\text{ab}}, B_{\text{an}}, \tau_{\text{ab}}, \tau_{\text{an}}) \in \mathcal{C}$ gilt, dass es die Knoten $v_{B_{\text{ab}}}^{\tau_{\text{ab}}}$ und $v_{B_{\text{an}}}^{\tau_{\text{an}}}$ sowie die Kante $(v_{B_{\text{ab}}}^{\tau_{\text{ab}}}, v_{B_{\text{ab}}}^{\tau_{\text{ab}}})$ mit dem Gewicht $\tau_{\text{an}} - \tau_{\text{ab}}$ in G gibt.
   G enthält weder Knoten noch Kanten, zu denen kein $C' \in \mathcal{C}$ gibt, von dem sie auf diese Weise abgeleitet wurden.
   Zwischen zwei Knoten $v_B^{\tau_1}, v_B^{\tau_2}$ zum selben Bahnhof $B \in \mathcal{B}$ gibt es genau dann die Kante $(v_B^{\tau_1}, v_B^{\tau_2})$, wenn τ_2 der nächste Zeitpunkt nach τ_1 an B ist. Dann hat diese Kante das Gewicht $\tau_2 - \tau_1$.
   Eine Routenanfrage (B_s, B_t, τ_s) erfragt also einen kürzesten Weg von einem Knoten $v_{B_s}^\tau$ mit $\tau \geq \tau_s$ zu einem Knoten $v_{B_t}^{\tau_t}$ so, dass τ_t minimal ist.
   Für die gegebene Probleminstanz ergibt sich der folgende Graph. Die Namen der Bahnhöfe kürzen wir mit ihren Anfangsbuchstaben ab.



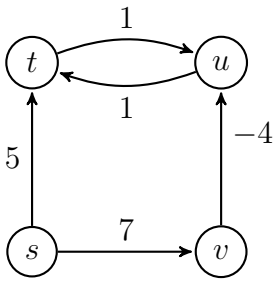
2. Wir betrachten die Knoten in G in aufsteigender Reihenfolge nach ihren Uhrzeiten. Für je zwei Bahnhöfe $B_1, B_2 \in \mathcal{B}$, zwischen denen man Zeit Δ zu Fuß benötigt, und für jeden Knoten $v_{B_1}^{\tau}$ fügen wir den Knoten $v_{B_2}^{\tau+\Delta}$ ein, sollte dieser noch nicht in G enthalten sein. Die beiden Knoten werden dann, gibt es noch keine Kante $(v_{B_1}^{\tau}, v_{B_2}^{\tau+\Delta})$, über eine Kante mit Gewicht Δ verbunden.
   Im Unterschied zur Modellierung in Teilaufgabe 1 besteht nun zwischen zwei Knoten $v_B^{\tau_1}, v_B^{\tau_2}$ zum selben Bahnhof $B \in \mathcal{B}$ genau dann die Kante $(v_B^{\tau_1}, v_B^{\tau_2})$, wenn ihr Gewicht $\tau_2 - \tau_1$ größer oder gleich der Umsteigezeit am Bahnhof B ist.

Aufgabe 2 - Dijkstra und negative Kanten (2 Punkte)

In dieser Aufgabe wollen wir untersuchen, was passiert wenn Dijkstra's Algorithmus auf einem gerichteten gewichteten Graphen mit (unter anderem) negativen Kantengewichten aufgerufen wird. Wir beschränken uns hierbei auf Instanzen ohne negative Kreise (d.h. die Summe der Kantengewichte in jedem Kreis ist nicht negativ). Gib einen Graphen G und zwei Knoten s und t an, sodass Dijkstra's Algorithmus aus der Vorlesung auf (G, s) nicht den kürzesten s-t-Weg findet und begründe kurz weshalb. (2 Punkte)

Lösung 2

Betrachte den folgenden Graphen $G = (V, E)$ mit $s, t \in V$:



G enthält keinen negativen Zyklus, jedoch bestimmt Dijkstra's Algorithmus $\text{dist}(s, t)$ fälschlicherweise als 5 statt 4. Das liegt daran, dass von s aus zunächst t und anschließend u exploriert werden. Beiden Knoten werden danach nicht mehr in die Queue eingefügt. Wird nun v exploriert, so wird zwar die Distanz von s zu u aktualisiert, jedoch nicht die zu t.

Aufgabe 3 - Das Orakel von Dijkstra (6 Punkte)

Zu einem Graphen $G = (V, E)$ und einem Knoten $s \in V$ bezeichnen wir mit dem *Kürzeste-Wege-Baum* von s einen Baum $T = (V, E')$ mit $E' \subseteq E$ bei dem jeder Pfad von s nach $t \in V$ in T auch einem kürzesten Pfad von s nach t in G entspricht. Entscheide für jede der folgenden Aussagen, ob sie wahr oder falsch ist und begründe deine Antwort.

1. Wenn alle Kanten im (gerichteten) Graph paarweise verschiedene Gewichte haben, dann ist der Kürzeste-Wege Baum von jedem Startknoten s aus eindeutig. (1 Punkt)

2. Wenn wir die Kantenrichtungen in einem gerichteten Graphen mit positiven Kantengewichten entfernen (indem man jede Kante als ungerichtet auffasst), dann ändern sich die Distanzen (bezüglich kürzester Wege) nicht. (1 Punkt)

3. Wenn wir das Gewicht jeder Kante um $k > 0$ vergrößern, dann vergrößern sich die Distanzen zwischen Knotenpaaren um Vielfache von k. (1 Punkt)

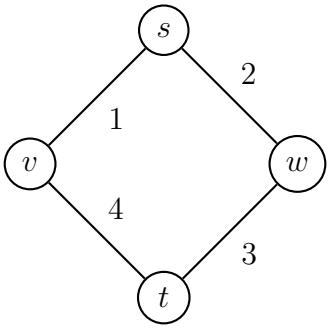
4. Wenn wir das Gewicht jeder Kante um $k > 0$ verkleinern, dann verkleinern sich die Distanzen zwischen Knotenpaaren um Vielfache von k. (1 Punkt)

5. Unter allen kürzesten Wegen zwischen zwei Knoten in einem (gerichteten) gewichteten Graph findet Dijkstra's Algorithmus immer denjenigen mit den wenigsten Kanten. (1 Punkt)

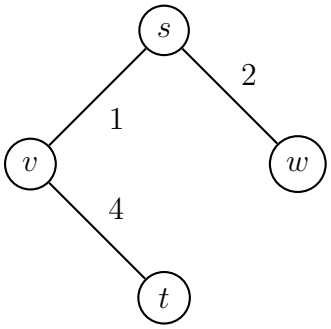
6. Falls ein zusammenhängender Graph keine negativen Kreise enthält, dann gibt es zwischen jedem Knotenpaar einen kürzesten Pfad, der einfach ist. Zur Erinnerung: ein Pfad ist einfach, wenn er keine Kreise enthält. (1 Punkt)

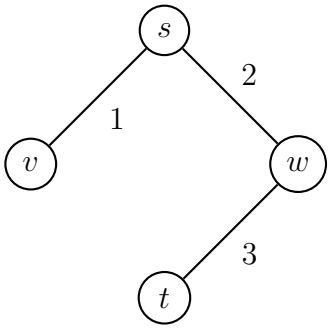
Lösung 3

1. **Falsch.** Ein Gegenbeispiel ist der folgende Graph:

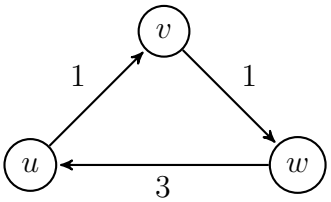


   Beide der folgenden Graphen sind Kürzeste-Wege-Baum zu s:

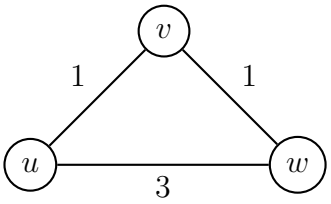




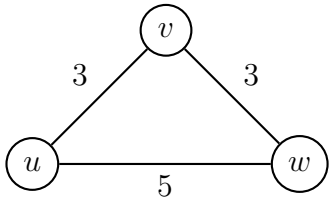
2. **Falsch.** Der folgende Graph ist ein Gegenbeispiel, denn hier hat der kürzeste Weg von w nach u Länge 3, durch Aufheben der Kantenrichtungen ergibt sich ein kürzester Weg der Länge 2.



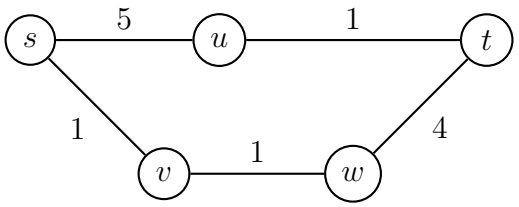
3. **Falsch.** Ein Gegenbeispiel ist das folgende: Im unten gegebenen Graphen ist $\text{dist}(u, w) = 2$. Wählen wir $k = 3$, so ändert sich diese Distanz zu 6 und damit um 4, also kein Vielfaches von k.



4. **Falsch.** Ein Gegenbeispiel ist das folgende: Im unten gegebenen Graphen ist $\text{dist}(u, w) = 5$. Wählen wir $k = 2$, so ändert sich diese Distanz zu 2 und damit um 3, also kein Vielfaches von k.



5. **Falsch.** Ein Gegenbeispiel ist der folgende Graph. Dijkstra's Algorithmus mit Startknoten s wird auf diesem zunächst die Knoten v und danach w explorieren. Damit wird t von w aus, d.h. über den Weg mit 3 Kanten, gefunden.



6. **Wahr.** Sein $G = (V, E)$ ein Graph ohne negative Kreise. Angenommen, es gibt $v, w \in V$ so, dass der kürzeste Pfad p von v nach w einen Kreis C enthält. Wir zeigen, dass es dann auch einen kürzesten Pfad ohne Kreis von v nach w gibt.
Der Pfad p enthält einen Knoten $u \in V$ mindestens zwei Mal, also

$$p = \langle v = v_0, v_1, \ldots, v_{k-1}, v_k = u, v_{k+1}, \ldots, v_{l-1}, v_l = u, v_{l+1}, \ldots, v_m = w \rangle$$

wobei für alle $i \in \{1, \ldots, k-1\} \cup \{l+1, \ldots, m-1\}$ gilt, dass $v_i \neq u$. Da die Summe der Kantengewichte in C mindestens 0 sein muss, gibt es einen Pfad in G von v nach w, welcher höchstens die gleiche Länge wie p hat und keinen Kreis enthält, nämlich

$$p' = \langle v = v_0, v_1, \ldots, v_{k-1}, u, v_{l+1}, \ldots, v_m = w \rangle$$

Aufgabe 4 - Viele Wege führen nach Rom (5 Punkte)

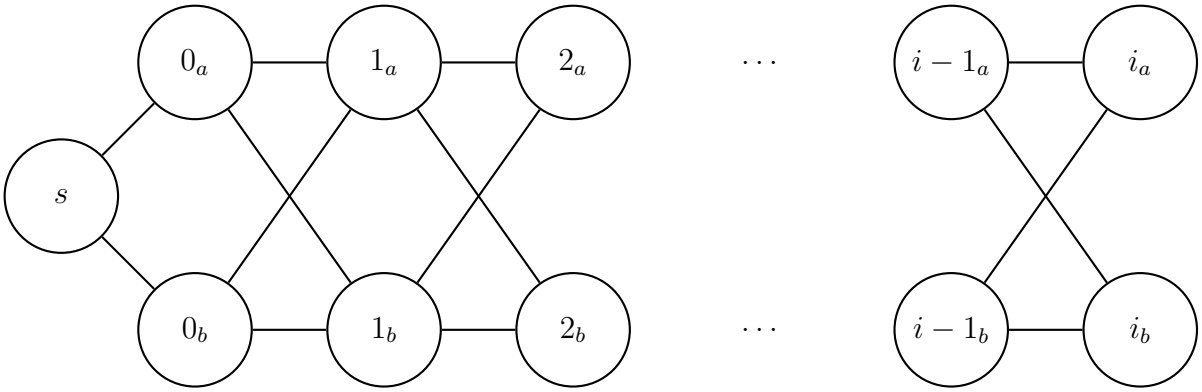
1. Gibt es Graphen mit n Knoten, in denen es zwischen zwei Knoten $2^{\Theta(n)}$ kürzeste Wege gibt? Begründe deine Antwort. (2 Punkte)

   Hinweis: Um eine asymptotische untere Schranke zu zeigen genügt es nicht einen einzelnen Graphen anzugeben. Versuche stattdessen eine Konstruktion zu finden, die für beliebig große n funktioniert.

2. Gegeben sei ein ungewichteter Graph $G = (V, E)$ mit $|V| = n$ und $|E| = m$. Beschreibe einen Algorithmus der in $O(n + m)$ zu einem gegebenen $s \in V$ für alle $v \in V$ die Anzahl kürzester s-v-Wege bestimmt. Begründe die Korrektheit deines Algorithmus und wieso er das geforderte Laufzeitverhalten hat. (3 Punkte)

Lösung 4

1. Ja, betrachte zum Beispiel die folgende Familie von Graphen für $i > 1$:



Von s aus gibt es zu den Knoten k_a, k_b mit $0 \leq k \leq i$ jeweils 2^k kürzeste Wege. Insbesondere gibt es zwischen s und i_a, i_b jeweils $2^i = 2^{\frac{n-1}{2}-1}$ kürzeste Wege.
Da $\frac{n-1}{2} - 1 \in \Theta(n)$, haben diese Graphen die gewünschte Eigenschaft.

2. Wir können eine angepasste Version der Breitensuche verwenden. Dabei nutzen wir aus, dass die Breitensuche auf ungerichteten ungewichteten Graphen kürzeste Wege findet. Für jeden Knoten v bestimmen wir sowohl $\text{dist}(s, v)$ als auch die Anzahl kürzester s-v-Wege. Das geschieht wie folgt:
Auf G wird eine Breitensuche durchgeführt. Wird in der Nachbarschaft eines Knoten u ein Knoten v gefunden, der noch nicht gefärbt ist, so bestimmen wir $\text{dist}(s, v) = \text{dist}(s, u) + 1$ und setzen die Anzahl kürzester s-v-Wege gleich der Anzahl kürzester s-u-Wege. War v bereits gefärbt und es gilt $\text{dist}(s, v) = \text{dist}(s, u)+1$, so erhöhen wir die Anzahl kürzester s-v-Wege um die Anzahl kürzester s-u-Wege. Dieser Algorithmus bestimmt die Länge eines kürzesten s-v-Weges für jeden Knoten $v \in V$, weil dieser Wert genau dann bestimmt wird, wenn v von der normalen Breitensuche gefunden wird. Für jede Kante $e = \{u, w\} \in E$ gilt zudem, dass jeder s-u-Weg um e erweitert einen s-v-Weg ergibt. Somit zählen wir die Anzahl kürzester s-v-Wege.
Wir speichern pro Knoten eine konstant viel zusätzliche Informationen. Das Anlegen der dazu benötigten Arrays hat einen Zeitbedarf in $\Theta(n)$. Die Anpassungen an die Breitensuche, also das Abfragen und aktualisieren von Distanzen und Anzahlen von kürzesten Wegen, laufen alle in konstanter Zeit ab. Die Laufzeit der Breitensuche liegt in $\Theta(m)$. Somit liegt die Gesamtlaufzeit unseres Algorithmus in $\Theta(n + m)$.

Dr. Meta hat sich bereits an einem geheimen Urlaubsort zurückgezogen, um sich von dem beruflichen Stress der letzten Wochen zu erholen.
Er wünscht eine erholsame Pfingst-Pause!