Karlsruher Institut für Technologie
Institut für Theoretische Informatik

Prof. Dr. Peter Sanders
Moritz Kobitzsch, Dennis Schieferdecker

2. Übungsblatt zu Algorithmen II im WS 2011/2012

http://algo2.iti.kit.edu/AlgorithmenII.php
{kobitzsch,sanders,schieferdecker}@kit.edu

Aufgabe 1 (*Kleinaufgaben: A* Suche*)

a) Sei pot(·) eine gültige Potentialfunktion für die Suche nach Knoten t in Graph $G(V,\ E)$. Überprüfen Sie, ob

$$\text{pot}^c = \text{pot} + c,\ c = const.$$

ebenfalls eine gültige Potentialfunktion darstellt.

b) Kann es vorkommen, dass eine A* Suche mehr Knoten absucht als eine Suche mit Dijkstras Algorithmus für die gleiche Anfrage? Begründen Sie warum nicht oder geben Sie ein Beispiel an.

Aufgabe 2 (*Rechnen: Monotone ganzzahlige* Priority Queues)

Bei einer Ausführung von *Dijkstras Algorithmus* wird folgender Ausschnitt an *Priority Queue* Operationen protokolliert:

...

- `insert(a, 06 [00110] )` (Parameter: Knotenbezeichnung, Distanz [Distanz binär])
- `insert(b, 10 [01010] )`
- `insert(c, 07 [00111] )`
- `deleteMin()`
- `deleteMin()`
- `insert(d, 12 [01100] )`
- `deleteMin()`
- `insert(e, 16 [10000] )`

...

Zusätzlich wissen Sie, dass das maximale Kantengewicht im Graphen $C = 6$ beträgt und dass vor der ersten protokollierten Operation das letzte enthaltene Element aus der *Priority Queue* entfernt wurde. Dieses hatte den Wert $min = 5$.

a) Führen Sie die Operationen auf einer *Bucket Queue* aus. Geben Sie den Zustand der Datenstruktur nach jeder Operation an.

b) Wieviele *Buckets* werden für eine Ausführung auf einem *Radix Heap* benötigt? Führen Sie die Operationen auf einem *Radix Heap* aus. Geben Sie den Zustand der Datenstruktur und den Wertebereich der *Buckets* nach jeder Operation an.

Aufgabe 3 (*Analyse: Laufzeit von Dijkstras Algorithmus*)

Gegeben sei ein gerichteter Graph $G = (V,\ E)$ mit $|V| = n$ und $|E| = m$, sowie eine Kantengewichtungsfunktion $c : E \to \mathbb{R}_0^+$.

a) Beweisen Sie die Behauptung aus der Vorlesung, dass für $m = \Omega(n \log n \log\log n)$ Dijkstras Algorithmus mit einem *binary heap* eine durchschnittliche Laufzeit von $O(m)$ besitzt.

b) Eine spezielle *Priority Queue* habe folgende Laufzeiteigenschaften:

- `insert`: $O(\log n)$
- `decreaseKey`: $O(1)$
- `deleteMin`: $O(\sqrt{m})$

(ob eine Datenstruktur mit diesen Eigenschaften existiert und Dijkstras Algorithmus mit ihr korrekt arbeitet, ist eine andere Frage, aber wir nehmen für diese Aufgabe an es ginge :-))

Geben sie eine kleinste obere Schranke für die Laufzeit von Dijkstras Algorithmus unter Verwendung dieser *Priority Queue* an. Unter welcher Bedingung an das Verhältnis der Anzahl Knoten n zu Kanten m wird die Laufzeit linear in der Eingabegröße?

c) Geben Sie eine Klasse von Graphen an, für die die Anzahl an `deleteMin` Operationen von einem beliebigen Knoten zu allen erreichbaren Knoten linear von der Pfadlänge $\mu(s, \cdot)$ abhängt, gegeben dass $m = \Omega(n)$.

Aufgabe 4 (*Entwurf:* All Pairs Shortest Paths)

Sie sind von der Finanzaufsichtsbehörde beauftragt worden, einen Algorithmus zu entwickeln, der Irregularitäten im Devisenhandel möglichst zeitnah aufdecken kann. Zu diesem Zweck erhalten Sie die aktuellen direkten Wechselkurse $w_{i,j}$ von Währung i nach Währung j für alle gehandelten Währungen. Dabei bedeutet z.B. $w_{i,j} = 4$, dass man für 1 Einheit aus Währung i genau 4 Einheiten aus Währung j erhält. Eine Unregelmäßigkeit tritt dann auf, wenn eine Möglichkeit existiert, eine Währung i in eine Währung j über mehrere Zwischenwechsel zu tauschen, so dass der Ertrag der Wechsel weniger als die Hälfte des direkten Wechsels von Währung i nach j erzielt. Auf Rückfrage versichert Ihnen Ihr Auftraggeber außerdem, dass eine geldgenerierende Schleife (leider) nicht auftreten kann.

a) Formulieren Sie das Problem als graphentheoretisches Problem. D.h. bilden Sie die gegebenen Informationen auf Knoten und Kanten eines Graphen ab und interpretieren Sie die gestellte Aufgabe als Problem auf dem von Ihnen definierten Graphen.

b) Beschreiben Sie einen Algorithmus, der das Problem löst.

c) Erweitern Sie Ihren Algorithmus, so dass er auch die Folge an Wechseln ausgeben kann, die eine Unregelmäßigkeit verursacht.

d) Ihr Algorithmus muss k Währungen überwachen. Geben Sie eine Laufzeit für Ihren Algorithmus an, die nur von k abhängt.

Hinweis: $\log ab = \log a + \log b$.

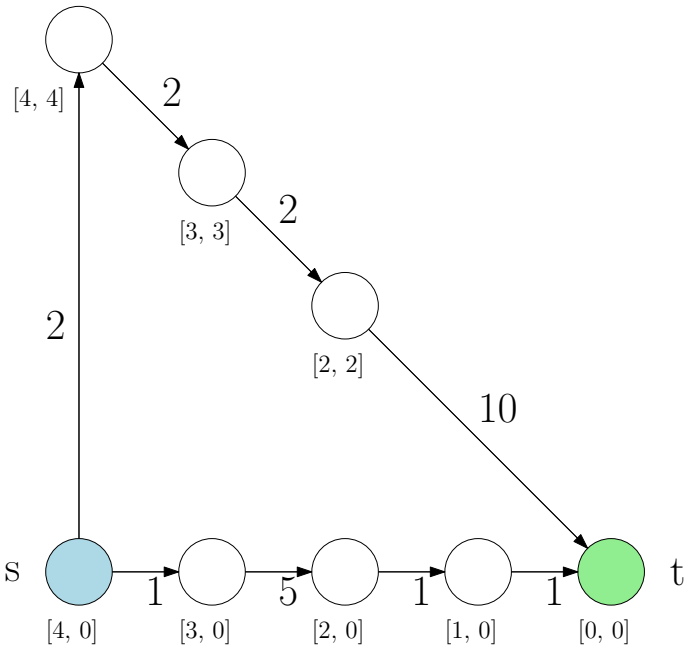
Aufgabe 5 (*Rechnen: A* Suche*)

Gegeben sei der unten abgebildete Graph. An den Kanten sind Kosten für die Nutzung der Verbindung eingetragen und die Knoten tragen Ortskoordinaten.

a) Ergänzen Sie den gegebenen Graphen um Knotenpotentiale für eine A* Suche von s nach t. Verwenden Sie die Manhatten-Distanz ($\hat{=}$ Einsnorm $||\cdot||_1$) als Abschätzung für die Entfernung zum Ziel.
Hinweis: $||\cdot||_1 : ||(x_1, y_1), (x_2, y_2)||_1 = y_2 - y_1 + x_2 - x_1$.

b) Tragen Sie die reduzierten Kantengewichte in den Graphen ein.

c) Wieviele `deleteMin` Operationen führt die A* Suche auf dem Graphen aus? Wieviele eine normale Suche mit Dijkstras Algorithmus?

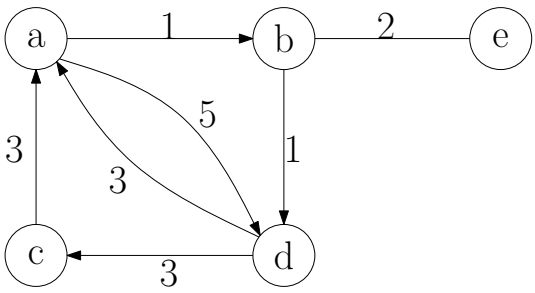


Aufgabe 6 (*Einführung+Analyse: Bidirektionaler Dijkstra*)

In Vorlesung und Saalübung wurde eine bidirektionale Variante von Dijkstras Algorithmus angesprochen, die in dieser Aufgabe näher untersucht werden soll.

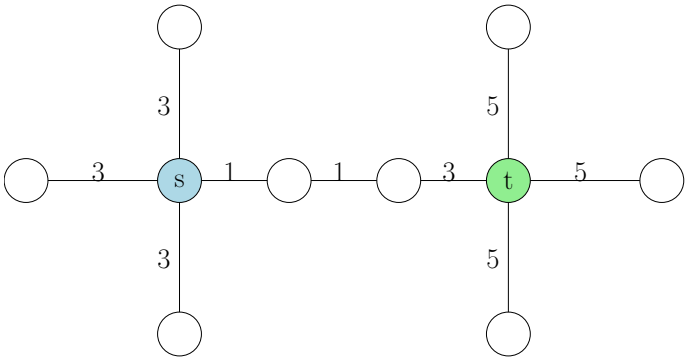
Zur Wiederholung:
Gegeben sei –wie üblich– ein gerichteter Graph $G = (V,\ E)$ mit $|V| = n$ und $|E| = m$, sowie eine Kantengewichtungsfunktion $c : E \to \mathbb{R}_0^+$. Gesucht ist der kürzeste Pfad $p = \langle s, \ldots, t \rangle$ zwischen zwei Punkten $s,\ t \in V$.
Eine bidirektionale Suche löst dieses Problem wie folgt: Es werden zwei unidirektionale Suchen mit Dijkstras Algorithmus gestartet. Die *Vorwärtssuche* beginnt bei Knoten s und operiert auf dem normalen Graphen G, auch *Vorwärtsgraph* genannt. Die *Rückwärtssuche* beginnt bei Knoten t und operiert auf dem *Rückwärtsgraph* $G^r = (V, E^r)$ mit Kantengewichtungsfunktion c^r. Dieser Graph entsteht aus G durch Umkehrung aller Kanten. Der Algorithmus scannt abwechselnd einen Knoten in der Vorwärtssuche und in der Rückwärtssuche, beginnend mit der Vorwärtssuche.
Wird während des Scans von Knoten u Kante $(u,\ v)$ relaxiert, so wird überprüft, ob die Distanz $d_{\text{forward}}[v] + d_{\text{backward}}[v]$ kleiner ist als die momentan minimale gefundene Distanz von s nach t und diese gegebenenfalls angepasst ($d_{\text{forward}}[v]$ gibt die bisher kürzeste gefundene Distanz von s nach v in der Vorwärtssuche und $d_{\text{backward}}[v]$ die bisher kürzeste gefundene Distanz von v nach t in der Rückwärtssuche an).
Sobald ein Knoten in einer Richtung gescannt werden soll, der bereits in der anderen Richtung gescannt worden ist, kann die Suche beendet werden (*Abbruchbedingung*). Die aktuelle minimale gefundene Distanz ist dann die tatsächliche minimale Distanz zwischen s und t.

a) Zeichnen Sie den Rückwärtsgraph G^r zum angegebenen Graphen. Geben Sie die Kantengewichte $c(a, d)$, $c^r(a, d)$ sowie $c(b, e)$, $c^r(b, e)$ an.



(Kante (b, e) ist eine bidirektionale [bzw. ungerichtete] Kante)

b) Geben Sie an, in welcher Reihenfolge der unten angegebene Graph durchlaufen wird.

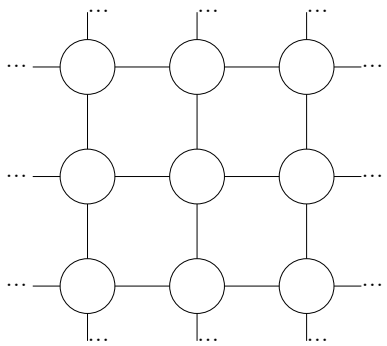


c) Zeigen Sie, dass die Abbruchbedingung korrekt ist.

d) Wann kann es passieren, dass die Suche nach dem Scan von Knoten u beendet wird, dieser aber nicht Teil des kürzesten Weges ist. Geben Sie ein Beispiel an.

Aufgabe 7 (*Analyse: Bidirektionaler Dijkstra*)

a) Gegeben sei ein Gittergraph G mit allen Kantengewichten gleich 1. Wieviele Knoten wird eine bidirektionale Suche besuchen in Abhängigkeit von Abstand d zwischen Start und Ziel? Wieviele die unidirektionale Suche?



Beispiel eines Gittergraphen

b) Geben Sie ein Beispiel an, in dem die bidirektionale Suche von s nach t exponentiell weniger Knoten besucht als die unidirektionale Suche.

c) Geben Sie ein Beispiel, in dem die bidirektionale Suche von s nach t mehr Knoten besucht als die unidirektionale Suche.

d) Zeigen Sie, dass die bidirektionale Suche nie mehr als doppelt so viele Knoten besucht als die unidirektionale Suche.

Ausgabe: 08.11.2011
Abgabe: keine Abgabe, keine Korrektur