



Wintersemester 2017/2018

Lineare und kombinatorische Optimierung

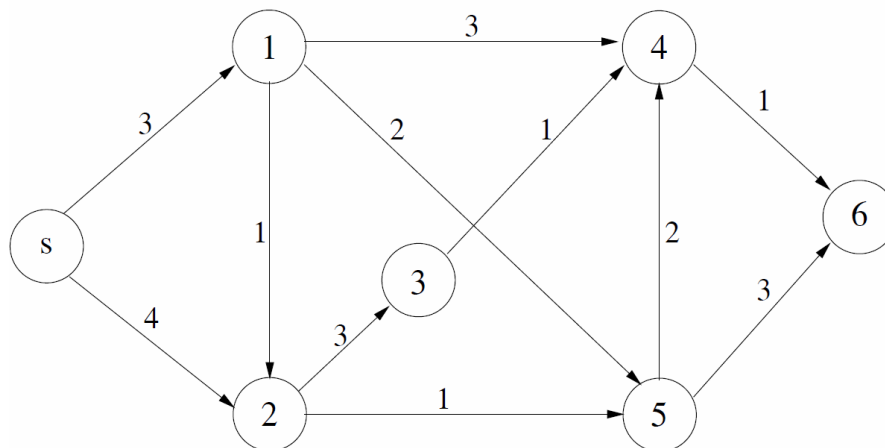
Übungsblatt 5

Gruppenübungen

Aufgabe 1. (Dijkstra, Kürzester-Wege-Baum, Grundversion Moore–Bellman)

(0 Punkte)

Gegeben sei der folgende Graph:

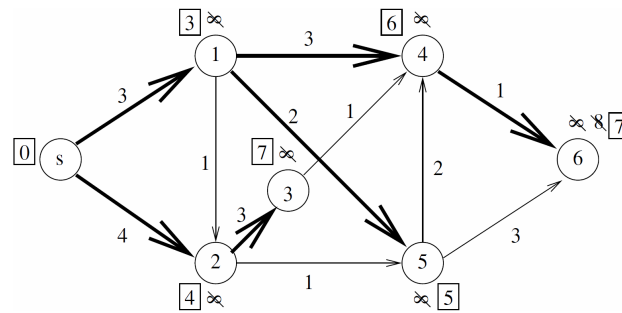


1. Berechnen Sie mithilfe Algorithmus 12 (Dijkstra) einen kürzesten Weg von s zu allen anderen Knoten und geben Sie den Kürzester-Wege-Baum an.
2. Ist der Kürzester-Wege-Baum eindeutig?
3. Zeigen oder widerlegen Sie: Für einen (ungerichteten) Graphen $G = (V, E)$ mit ausgezeichnetem Knoten $s \in V$ ist ein Kürzester-Wege-Baum für Knoten s zu allen anderen Knoten derselbe wie ein minimal aufspannender Baum.
4. Verändern Sie das Gewicht von Bogen $(3, 4)$ auf -2 . Wenden Sie zur Berechnung der kürzesten Wege den Algorithmus 13 (Grundversion Moore–Bellmann) an und zeigen Sie, dass der Dijkstra-Algorithmus in diesem Fall nicht korrekt funktioniert.

Lösung:

1. Die Lösung ist im folgenden Bild angegeben. Die Zahlen, die bei den Knoten stehen, sind die d -Werte und das Durchstreichen und neu Schreiben bedeutet, ein Update hat stattgefunden. Die

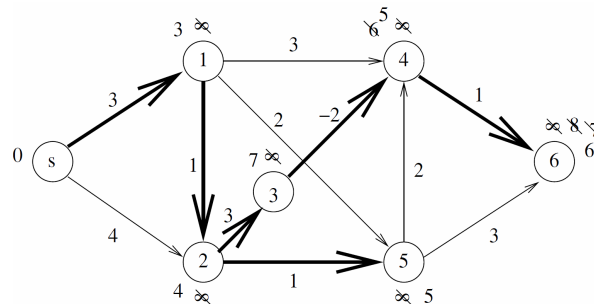
Werte in den Kästchen sind dann die endgültigen d -Werte, also die Länge der kürzesten Wege. Der Kürzeste-Wege-Baum ist dick eingezeichnet. Die Reihenfolge, in der die Knoten markiert werden ist $s, 1, 2, 5, 4, 6, 3$ oder genauso gut $s, 1, 2, 5, 4, 3, 6$.



- Nein, er ist i.A. nicht eindeutig. Z.B., wenn man im obigen Baum die Kante $(s, 2)$ mit der Kante $(1, 2)$ ersetzt, bekommt man einen anderen Kürzeste-Wege-Baum.
- Gegenbeispiel: vollständiger Graph $G = (V, E)$ mit $V = \{s, 1, 2\}$ und $c_{s1} = c_{s2} = 2$ und $c_{12} = 1$.

- Der Dijkstra-Algorithmus funktioniert nicht korrekt, da Folgendes geschieht: Die Lösung sieht wieder genauso aus, wie in Teilaufgabe (a), aber der kürzeste Weg zu Knoten (4) sollte über den Knoten (3) gehen und die Länge 5 (statt 6) haben. Aber wenn die Kante $(3, 4)$ betrachtet wird, ist Knoten (4) schon markiert und deshalb wird kein Update durchgeführt.

Der Moore-Bellmann-Algorithmus in der Grundversion schaut sich jede Kante immer wieder an und falls ein Update möglich ist, führt er das durch. Das tut er so lange, bis kein Update mehr im Graphen möglich ist. Eine mögliches Ergebnis von diesem Algorithmus ist unten angegeben.



Bemerkung: Dijkstra läuft in der Regel auch schief, wenn man auf alle Kanten das Gewicht $c = \max\{-c_{ij} : c_{ij} < 0\}$ addiert, um alle Kanten positiv zu machen. Siehe nächste Aufgabe!

Aufgabe 2. (Kürzeste Wege auf modifizierten Graphen)

(0 Punkte)

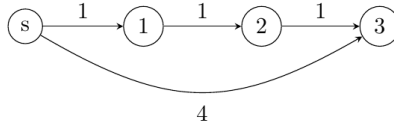
Gegeben sei ein gerichteter Graph $D = (V, A)$ mit positiven Kantengewichten sowie die kürzesten Wege von einem Startknoten $s \in V$ zu allen Knoten $i \in V$ mit Längen $d(i)$. Nun wird der Graph folgendermaßen modifiziert:

- Auf jedes Kantengewicht wird 1 addiert.
- Jedes Kantengewicht wird mit $\lambda > 0$ multipliziert.

Bleiben *jeweils* die bisherigen kürzesten Wege kürzeste Wege des geänderten Graphen? Begründen Sie Ihre Entscheidung oder geben Sie ein jeweils ein Gegenbeispiel an.

Lösung:

- Gegenbeispiel:



2. Diese Aussage ist richtig:

Sei P_i^* der kürzeste Weg von s nach i für $i \in V$. Dann gilt

$$d(P_i^*) \leq d(P_i)$$

für jeden Weg P_i von s nach i . Seien d^{neu} die neuen Distanzlabel. Dann gilt:

$$d^{neu}(P_i^*) = \lambda d(P_i^*) \leq \lambda d(P_i) = d^{neu}(P_i).$$

Also ist P_i^* auch kürzester Weg bzgl. d^{neu} .

Aufgabe 3. (Kürzeste Wege und ungerade Kreise)

(0 Punkte)

Zeigen Sie, dass das Problem, einen kürzesten ungeraden Kreis in einem Digraphen mit nicht-negativen Bogengewichten zu finden, mit einem Algorithmus zur Bestimmung kürzester Wege gelöst werden kann.

Lösung:

Wir betrachten unseren Graphen $D = (V, A)$ mit Gewichten c_a . Wir bilden nun einen neuen Graphen D' nach folgender Vorschrift: Die Eckenmenge V' besteht aus zwei disjunkten Kopien von V : $V' := V \times \{0\} \cup V \times \{1\}$. Wir verbinden zwei Ecken (v_1, σ_1) und (v_2, σ_2) in V' genau dann, wenn die beiden Ecken in verschiedenen Kopien von V sind und wenn sie in D verbunden waren, also $\sigma_1 = 1 - \sigma_2$ und $v_1 v_2 \in A$ gilt. Als Gewicht wählen wir in diesem Fall das Gewicht der ursprünglichen Kante in D : $c_{v_1 v_2}$.

Das ursprüngliche Problem lösen wir nun, indem wir für alle $v \in V$ kürzeste Wege in D' von $(v, 0)$ nach $(v, 1)$ suchen. Der minimale hierunter ergibt dann unseren minimalen ungeraden Kreis. Dieser Kreis ist ungerade, da eine ungerade Anzahl von Wechslen zwischen den Graphen stattgefunden hat. Auf den Beweis dieser Behauptung verzichten wir hier.

Hausübungen

Aufgabe 4. (Charakterisierung Kürzester Wege, Beweis Lemma 5.1)

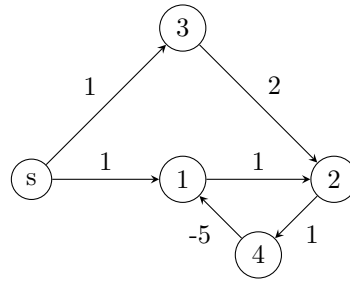
(2+2 Punkte)

Es sei $D = (V, A)$ ein gerichteter Graph, der keine negativen Kreise enthält.

1. Zeigen Sie: Ist $(s = i_0, i_1, \dots, i_k = t)$ ein kürzester Weg von s nach t , so ist auch jeder Teilweg (i_0, \dots, i_ℓ) für $\ell = 1, \dots, k - 1$ ein kürzester Weg von s nach i_ℓ .
2. Zeigen Sie, dass diese Aussage unter Umständen nicht gilt, wenn der Graph negative Kreise enthält.

Lösung:

1. Wir nehmen an, der Graph beinhaltet keine negativen Kreise und es gibt einen kürzeren Weg P' von s nach i_ℓ . Dann hätten wir, durch das Zusammensetzen von P' und $(i_{\ell+1}, \dots, i_k = t)$ eine *Kette* von s nach t , die kürzer ist als der gegebene kürzeste Weg von s nach t . Wir können alle Kreise aus dieser Kette entfernen und da alle diese Kreise nicht-negatives Gewicht haben, erhalten wir einen Weg von s nach t , dessen Gewicht kleiner ist als der kürzeste Weg von s nach t . Widerspruch.
2. Beachte, dass Wege keine Knoten und Kanten doppelt benutzen. Gegenbeispiel:



Der Graph hat offensichtlich einen negativen Kreis. Der kürzeste Weg von s nach 2 ist der Weg $s - 1 - 2$ der Länge 2. Allerdings ist $s - 1$ mit Länge 1 nicht der kürzeste Weg von s nach 1, da $s - 3 - 2 - 4 - 1$ mit Länge -1 kürzer ist.

Aufgabe 5. (Kürzeste Wege)

(2+2 Punkte)

Die Nachricht **babbbbaabba** der Länge $n = 10$ soll mit dem folgenden Codebuch codiert werden:

Text	Code	Länge
a	00	2
b	010	3
ba	0110	4
bb	0111	4
abb	1	1

Beispiele für mögliche Codierungen der Nachricht wären also die folgenden:

b abb b a a b ba oder ba bb ba a bb a
 010 1 010 00 00 010 0110 0110 0111 0110 00 0111 00

Die Gesamtlänge der ersten Codierung ist 18, die Länge der zweiten Codierung ist 20.

Für das angegebene Wort soll eine Codierung mit minimaler Gesamtlänge gefunden werden. Gehen Sie hierzu wie folgt vor:

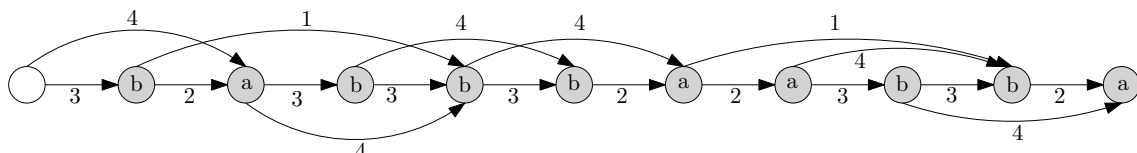
1. Formulieren Sie das Problem als ein Kürzeste-Wege-Problem. Gebe eine exakte Beschreibung des Digraphen oder eine detaillierte Zeichnung an.
2. Berechnen Sie eine Lösung mithilfe des Dijkstra-Algorithmus. Geben Sie die berechnete optimale Codierung sowie die Länge der Codierung an.

Hinweis: Man kann das Problem auf einen “kleinen“ Graphen mit $\mathcal{O}(n)$ Knoten modellieren. Größere Formulierungen sind erlaubt, erschweren aber die Berechnung im zweiten Teil der Aufgabe.

Lösung:

1. Sei $\text{text} = \text{babbbbaabba} = c_1 \dots c_{10}$. Wir definieren einen gerichteten azyklischen Graphen $D = (V, A)$ mit $V = \{0, \dots, 10\}$ und $A = \{(i, j) | c_{i+1} \dots c_j \text{ ist im Codebuch}\}$.

Die Länge $w(a)$ einer gerichteten Kante $a \in A$ ergibt sich durch die Länge der zugehörigen Codierung im Codebuch.



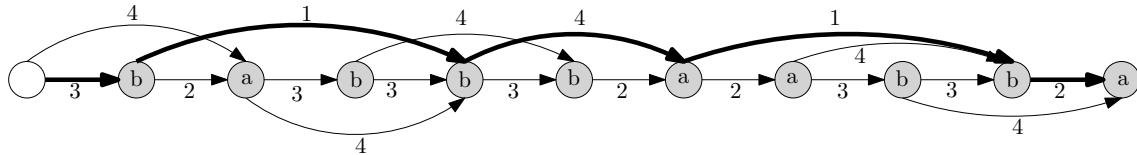
Jeder Weg von s (linker Knoten) nach 10 (rechter Knoten) legt eine Codierung fest. Somit erhält man eine Codierung mit minimaler Gesamtlänge durch einen kürzesten Weg von s nach 10.

2. Mit dem Algorithmus von Dijkstra erhalten wir

b abb ba abb a
 010 1 0110 1 00

als eine Codierung mit minimaler Gesamtlänge 11.

Veranschaulichung des kürzesten Weges:



Aufgabe 6. (Längste Wege)

(2 Punkte)

Es sei $T = (V, E)$ ein Baum, in dem kein Knoten Grad 2 hat. Zeigen Sie, dass für die Anzahl d von Kanten eines längsten Weges in T die Abschätzung $d \leq |V|/2$ gilt.

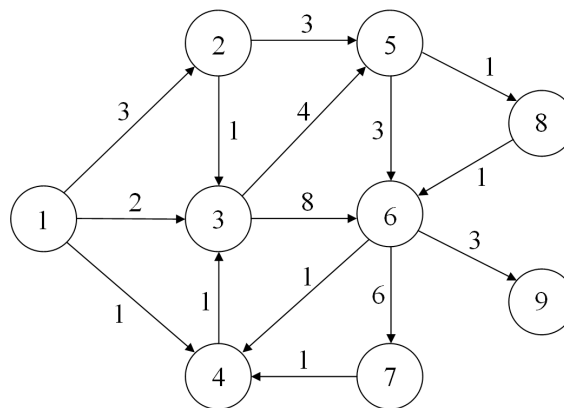
Lösung:

Ein Weg mit d Kanten enthält $d + 1$ Knoten. Jeder der $d - 1$ inneren Knoten des Weges muss noch zu mindestens einem weiteren Knoten außerhalb des Weges adjazent sein, da der Knotengrad nicht 2 ist. Da T kreisfrei ist, sind alle Knoten (die des Weges und die äußeren) paarweise verschieden. Also gilt: $(d + 1) + (d - 1) = 2d \leq |V|$, also $d \leq \frac{|V|}{2}$.

Aufgabe 7. (Dijkstra-Algorithmus und CATBox)

(1+1 Punkte)

Gegeben sei der folgende Graph G :



- Lösen Sie das Kürzeste-Wege-Problem vom Startknoten 1 zu allen anderen Knoten mithilfe des Algorithmus von Dijkstra und geben Sie den Kürzesten-Wege-Baum an. Benutzen Sie hierfür CATBox¹. Zuerst erstellen Sie mit **Gred** den Graphen und speichern diesen als z.B. **Aufgabe7Graph.cat** ab. Anschließend laden Sie den in CATBox vorhandenen Algorithmus **Dijkstra.alg** und den Graphen **Aufgabe7Graph.cat** mit **Gato**.
- Wie oft wurde das Distanzlabel am Knoten 6 aktualisiert? In welcher Reihenfolge werden die Knoten in die Menge S (vgl. Zeile 1 des Dijkstra-Algorithmus) aufgenommen? Setzen Sie hierfür in **Gato** an geeigneten Stellen im Algorithmus Breakpoints², um dies besser auslesen zu können.

Lösung:

- Der von CATBox berechnete Kürzeste-Wege-Baum ist Abb. 1 dargestellt.
- Das Distanzlabel am Knoten 6 wurde insgesamt 3 Mal aktualisiert. Der Vorgänger dieses Knotens war zuerst 3, dann 5 und zuletzt 8. Aufnahmereihenfolge in S : 1, 4, 3, 2, 5, 8, 6, 9, 7

¹Wie CATBox heruntergeladen und installiert werden kann, können Sie in unserer Installationsanleitung **Installationstutorial_CATBox.pdf** auf StudOn erfahren.

²Breakpoints können per Linksklick auf eine Zeile aktiviert bzw. deaktiviert werden

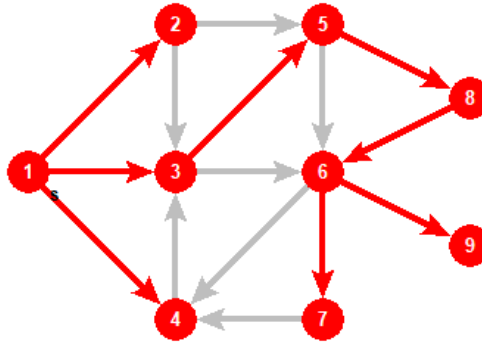


Abbildung 1: Von CATBox berechneter Kürzester-Wege-Baum

Aufgabe 8. (Programmieraufgabe: Dijkstra)

(3 Punkte)

Laden Sie die Datei `ProgAufgabe05.py` aus Studon herunter und fügen Sie sie in ihr PyCharm ein.

In dieser Programmieraufgabe implementieren Sie den aus der Vorlesung bekannten Dijkstra-Algorithmus zum Finden kürzester Wege, allerdings mit einer Adjazenzmatrix als Eingabe anstatt eines Graphen. Implementieren Sie hierzu die Methode `dijkstra(matrix,s)`, die folgende Signatur zu erfüllen hat:

- Der Parameter `matrix` ist die Adjazenzmatrix des Graphen. `matrix[i,j]` sind die Kosten, um von `i` nach `j` zu kommen.
Wichtig: `matrix[i,j] = ∞` (erhält man mittels `float('inf')`) bedeutet, dass keine Kante von `i` nach `j` führt!
- Der Parameter `s` ist der Startknoten ($0 \leq s < \text{len}(\text{matrix})$).
- Rückgabewert ist ein Tupel `(d,pred)`
 - `d` ist eine Liste, wobei der `i`-te Eintrag die Kosten des kürzesten Weges von `s` nach `i` sind.
 - `pred` ist eine Liste, wobei der `i`-te Eintrag der vorletzte Knoten im kürzesten Weg von `s` nach `i` ist.
 - Wichtig: Ist ein Knoten `i` überhaupt nicht von `s` aus erreichbar, dann muss `d[i] = ∞` (erhält man mittels `float('inf')`) und `pred[i] = i` sein.

Unterhalb der Methode `dijkstra` finden Sie ausführlichen Testcode. Zum Nachvollziehen empfiehlt es sich, den dort konstruierten Graphen aufzumalen.

Lösung:

Die Musterlösung der Programmieraufgabe finden Sie im StudOn in der Datei `ProgAufgabe05_Loesung.py`.

Gruppe 1: Abgabe der Hausaufgaben am 28.11.2017 in der Übung.
Gruppe 2+3: Abgabe der Hausaufgaben am 29.11.2017 in der jeweiligen Übung.