Übungen zu Algorithmen
WSI für Informatik
Kaufmann/Kuckuk/Förster

Wintersemester 2021/22
Universität Tübingen
1. Dezember 2021

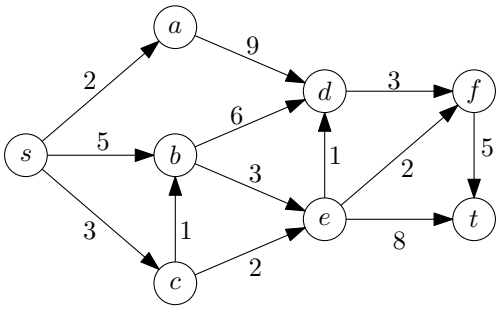
Aufgabensammlung Teil 6

Thema: Kürzeste Wege

Aufgabe 1: Dijkstra

Dies ist eine alte Klausuraufgabe!

Betrachten Sie den folgenden Graph:



a) Führen Sie den Dijkstra-Algorithmus mit Start in s aus. Verwenden Sie dafür die untenstehende Tabelle und geben Sie die Einträge in der folgenden Form an:

(Knoten, aktuelle Kosten, Vorgängerknoten)

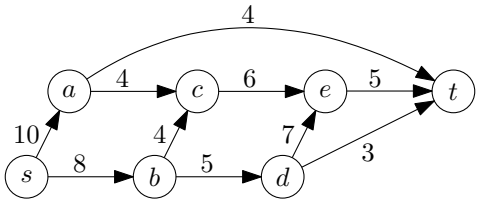
b) Sei $G = (V, E)$ ein (beliebiger) gerichteter Graph mit Kantenkosten $c : E \to \mathbb{N}$ und p ein kürzester Pfad in G von $s \in V$ nach $t \in V$. Zeigen oder widerlegen Sie: Werden alle Kantenkosten von G verdoppelt, so ist p immer noch ein kürzester Pfad von s nach t.

Schritt	S (fertig)	S' (Warteschlange)
1	(s,0,-)	
2		
3		
4		
5		
6		
7		
8		

Aufgabe 2: Maximum Bottleneck Path

Sei $G = (V, E, c)$ ein gerichteter Graph mit positiven Kantenkosten c. Sei P ein Pfad zwischen zwei Knoten u und v, wobei die billigste Kante in P Kosten k habe. P heißt *breitester Pfad* von u nach v, wenn es keinen anderen Pfad in G von u nach v gibt, dessen billigste Kante teurer als k ist.

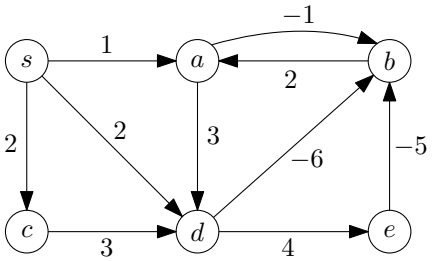
a) Zeichnen Sie in folgendem Graphen den breitesten Pfad von s nach t ein.



b) Modifizieren Sie Dijkstras Algorithmus so, dass er zu einem gegebenen Startknoten s die breitesten Pfade zu allen anderen Knoten findet. Begründen Sie die Korrektheit und die Laufzeit Ihres Algorithmus.

Aufgabe 3: Bellman-Ford-Algorithmus

Gegeben sei folgender gerichteter Graph $G = (V, E)$.



Lösen Sie mit Hilfe des Bellman-Ford-Algorithmus das *single-source shortest-paths* Problem für den Source-Knoten s. Verwenden Sie dabei die folgende Reihenfolge für die Kanten:

$$(s, a), (s, c), (s, d), (a, b), (a, d), (b, a), (c, d), (d, b), (d, e), (e, b).$$

Zeigen Sie alle Zwischenschritte!

Aufgabe 4: Floyd-Warshall-Algorithmus

Die in der Vorlesung präsentierte Variante des Floyd-Warshall-Algorithmus berechnet nur die Kürzeste-Pfad-Distanz zwischen allen Knoten, nicht aber kürzeste Pfade selbst (also Pfade, die die Kürzeste-Pfad-Distanz realisieren). Adaptieren Sie den Algorithmus so, dass er kürzeste Pfade repräsentiert durch eine Vorgängermatrix mitberechnet. Geben Sie den adaptierten Algorithmus in Pseudo-Code an und begründen Sie seine Korrektheit.

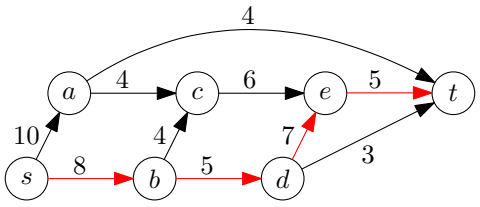
Lösungen:

1. a) Ergebnis:

Schritt	S (fertig)	S' (Warteschlange)
1	(s,0,-)	(a,2,s), (c,3,s), (b,5,s)
2	(a,2,s)	(c,3,s), (b,5,s), (d,11,a)
3	(c,3,s)	(b,4,c), (e,5,c), (d,11,a)
4	(b,4,c)	(e,5,c), (d,10,b)
5	(e,5,c)	(d,6,e), (f,7,e), (t,13,e)
6	(d,6,e)	(f,7,e), (t,13,e)
7	(f,7,e)	(t,12,f)
8	(t,12,f)	–

   b) Stimmt, betrachte Definition des kürzesten Pfades.

2. a) Breitester Pfad:



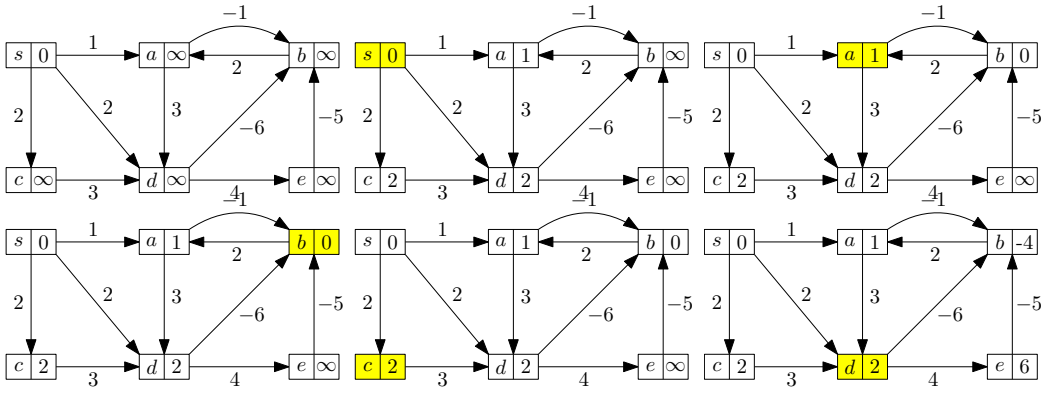
   b) Es gibt zwei Modifikationen:

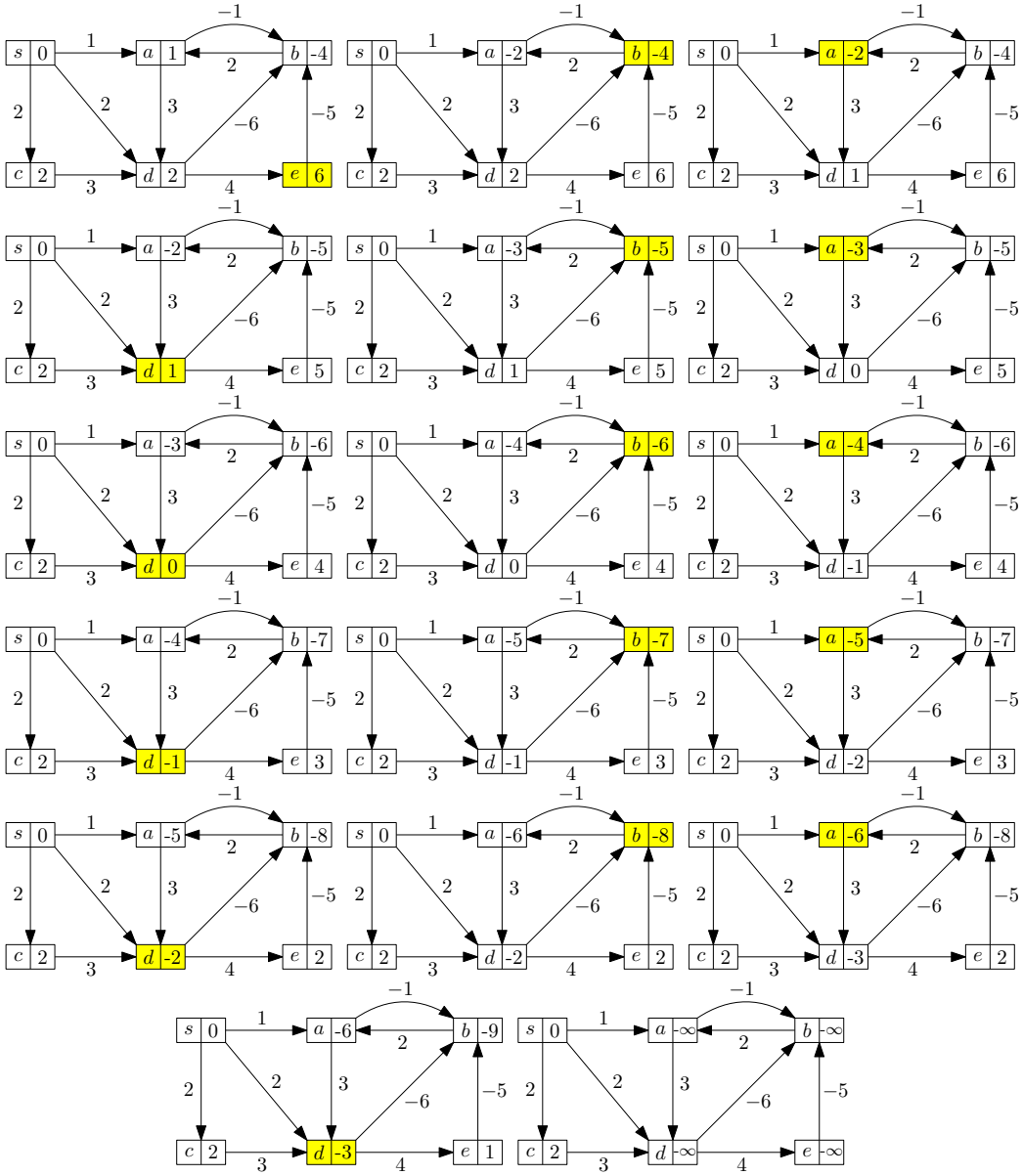
      i. Für noch nicht expandierte Knoten v in der Warteschlange wird die Breite des breitesten bereits gefundenen Pfades von s zu v gespeichert. Dies ersetzt die Speicherung der Länge des aktuell kürzesten Pfades.

      ii. Die Warteschlange S' wird nach der Breite des breitesten bereits gefundenen Pfades sortiert. Dementsprechend wird immer der Knoten aus S' expandiert, dessen aktuell gefundener breitester Pfad am breitesten ist.

   Die Laufzeit entspricht der des normalen Dijkstra-Algorithmus.

3. Ablauf (links nach rechts, dann oben nach unten):





4. Folgende Adaptionen sind nötig:

   i. Statt nur $\delta_k(i,j)$ zu berechnen, muss auch berechnet werden, was der letzte Knoten $\pi_k(i,j)$ auf dem Pfad zwischen $\delta_k(i,j)$ ist.

   ii. Initial gilt für alle Kanten (i,j), dass $\pi_0(i,j) = i$ (also der Vorgänger von j entlang der Kante (i,j) ist i).

   iii. Ansonsten gilt für alle i,j, die über keine Kanten verbunden sind, dass $\pi_0(i,j) = \varnothing$.

   iv. Wenn nun $\delta_k(i,j) \neq \delta_{k-1}(i,j)$, geht der neue günstigste Weg von i zu j über k, also ist $\pi_k(i,j) = \pi_{k-1}(k,j)$.

   Die Laufzeit ist identisch zum normalen Floyd-Warshall-Algorithmus. Die Korrektheit folgt analog.