14. Aufgabenblatt zur Vorlesung

Algorithmen und Programmierung III

Wolfgang Mulzer WiSe 2013/14

Abgabe am 6. Februar 2014 vor der Vorlesung in die jeweiligen Tutorenfächer
Dies ist das letzte Aufgabenblatt.

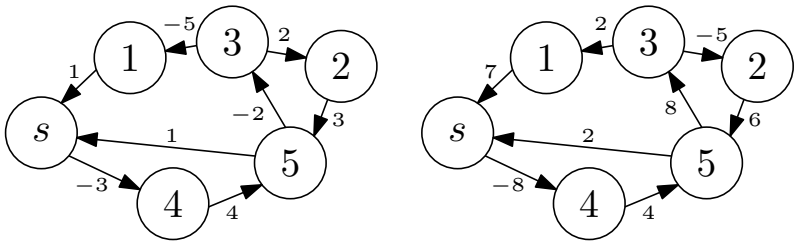
Aufgabe 1 Bellman-Ford und negative Kreise *10 Punkte*

Die folgende Variante des Algorithmus von Bellman-Ford findet negative Kreise.

```
improve(e = uv)
  if u.d + e.length < v.d then
    v.d <- u.d + e.length; v.pred <- u
    return true
  else
    return false

bellman-ford(G, s)
  for v in G.vertices() do
    v.d <- INFTY; v.pred <- NULL
  s.d <- 0
  for i := 1 to |V|-1 do
    for e in G.edges() do
      improve(e)
  for e in G.edges() do
    if improve(e) then
      return "Negative cycle found."
  return "There is no negative cycle."
```

(a) Führen Sie den Algorithmus auf den folgenden Graphen aus. Nehmen Sie dabei an, dass die Kanten in lexikographischer Reihenfolge durchlaufen werden (mit s als erstem Knoten). Zeigen Sie die einzelnen Schritte.



(b) Beweisen Sie, dass der Algorithmus korrekt ist.

Hinweis: Nehmen Sie an, dass es einen negativen Kreis C gibt und dass `improve` in der letzten Schleife für keine Kante von C `true` liefert. Leiten Sie einen Widerspruch ab.

(c) (freiwillig, 5 Zusatzpunkte) Modifizieren Sie den Algorithmus so, dass er auch die Knoten eines negativen Kreises liefert. Begründen Sie die Korrektheit und analysieren Sie die Laufzeit.

Aufgabe 2 Der Algorithmus von Floyd-Warshall *10 Punkte*

Sei G ein gerichteter gewichteter Graph ohne negative Kreise.

(a) Geben Sie detaillierten Pseudocode für den Algorithmus von Floyd-Warshall, um die *Längen* aller paarweisen kürzesten Wege in G zu bestimmen. Begründen Sie, dass Ihr Algorithmus $O(|V|^3)$ Zeit und $O(|V|^2)$ Platz benötigt.

(b) Ändern Sie Ihren Algorithmus so, dass er auch die kürzesten Wege selbst findet.

Aufgabe 3 Minimale aufspannende Bäume *10 Punkte*

(a) Zeigen Sie: Sei G ein ungerichteter gewichteter zusammenhängender Graph mit paarweise verschiedenen Kantengewichten. Dann ist der MST von G eindeutig.

(b) Sei G ein ungerichteter gewichteter zusammenhängender Graph und T ein MST für G. Sei e eine Kante von T, so dass $G \setminus e$ zusammenhängend ist. Geben Sie einen Algorithmus, der in Zeit $O(|V| + |E|)$ einen MST für $G \setminus e$ findet, wenn T bekannt ist. Beweisen Sie, dass Ihr Algorithmus korrekt ist.

Aufgabe 4 Das 8-Puzzle *freiwillig, 10 Zusatzpunkte*

Das *8-Puzzle* ist ein bekanntes Schiebespiel. Gegeben ist ein 3x3-Gitter, das acht verschiebbare Blöcke enthält, die mit den Zahlen von 1 bis 8 beschriftet sind. Eine Position des Gitters ist frei. Ziel ist es, die Blöcke mit so wenig Zügen wie möglich in die richtige Reihenfolge zu bringen. Dabei kann in jedem Zug ein Block, der zu der freien Position benachbart ist (horizontal oder vertikal), in diese Position geschoben werden. Hier ein Beispiel:

```
   1  3        1     3        1  2  3        1  2  3        1  2  3
4  2  5   =>   4  2  5   =>   4     5   =>   4  5     =>   4  5  6
7  8  6        7  8  6        7  8  6        7  8  6        7  8

 start          1 links         2 hoch         5 links          ziel
```

(a) Schreiben Sie ein Programm in Java, welches das 8-Puzzle für eine gegebene Eingabe mit Breitensuche löst. Bedenken Sie, dass der Graph nicht explizit gegeben ist. Welche Probleme entstehen dadurch, und wie kann man sie lösen? Testen Sie Ihr Programm. Wie gut eignet es sich, um das 8-Puzzle zu lösen?

(b) Verbessern Sie Ihr Programm mit A^*-Suche. Benutzen Sie dazu den *Hamming-Schätzer*, welcher den Abstand zum Ziel durch die Anzahl der Blöcke abschätzt, die nicht am richtigen Platz liegen. Genauer: für eine gegebene Konfiguration

des 8-Puzzles v definieren wir den Schätzer $h(v)$ als die Anzahl der Positionen im Gitter, an denen der aktuelle Block nicht der Zielkonfiguration entspricht (die freie Position wird hierbei nicht gezählt).

Begründen Sie, dass der Hamming-Schätzer konsistent ist. Implementieren Sie A^*-Suche mit dem Schätzer und testen Sie Ihr Programm.

Credits: Diese Aufgabe ist von Kevin Waynes COS 226 Vorlesung inspiriert.
`http://www.cs.princeton.edu/courses/archive/fall13/cos226/assignments/8puzzle.html`