Dr. S. Lewandowski Universität Stuttgart, Fak. 5, FMI-FK, SS 2006

Wegeprobleme in Graphen **Übungsblatt 03**

Ausgabe: 12.05. Besprechung am 19.05, 8h45–9h30

Ablauf der Übungen: Die Übungen werden freitags, 8h45–9h30 (Raum 0.124) und zum Teil in der Vorlesung besprochen oder es werden Lösungshinweise gegeben. Sie haben die Möglichkeit, Abgaben zu machen, die dann korrigiert werden.

Eine allgemeine Empfehlung: Programmieren Sie die hier vorgestellten Algorithmen aus oder rechnen Sie zumindest unbedingt einige Beispiele durch, um die Eigenheiten der einzelnen Verfahren besser kennenzulernen.

---

1. (mittel) **Eine weitere Variante des Bellman-Ford-Algorithmus: der Algorithmus von Pallottino:** Wie bei der Variante von D'Esopo und Pape von Aufgabenblatt 2 wollen wir versuchen, falsch berechnete Teile des Kürzeste-Wege-Baums möglichst frühzeitig zu korrigieren, um unnötige Arbeit zu ersparen. Die beidseitige Schlange bei der Variante nach D'Esopo und Pape lässt sich als Kombination einer FIFO (für die erstmals betrachteten Knoten) und LIFO (für die Knoten, die vorher schon einen endlichen Wert $D(\cdot)$ hatten) auffassen – ist die LIFO nicht leer, so wird der nächste Knoten aus dieser entnommen, sonst aus der FIFO. Das Problem bei D'Esopo und Pape ist, dass die LIFO zu exponentieller Laufzeit im Worst-Case führen kann. Die Variante von Pallottino ersetzt die LIFO durch eine zweite FIFO: Knoten, die erstmals betrachtet werden, kommen in die FIFO1, die anderen in die FIFO2 – ist FIFO2 nicht leer, so wird der nächste Knoten aus dieser entnommen, sonst aus FIFO1.

   Spielen Sie diesen Algorithmus für die Worst-Case-Beispiele vom D'Esopo-Pape-Algorithmus und vom Bellman-Ford-Algorithmus durch. Überlegen Sie, wie ein Worst-Case-Beispiel für Pallottino aussehen könnte. Welche Worst-Case-Laufzeit kann garantiert werden?

   Anmerkung: In der Praxis haben beide Varianten im Mittel ein deutlich besseres Laufzeitverhalten als der Bellman-Ford-Algorithmus. Speziell für Graphen im Straßenverkehr sind sie sehr gut geeignet. Für einige wenige Graphklassen ist das Laufzeitverhalten jedoch recht schlecht, so dass man die beiden Algorithmen selten einsetzt, wenn über die Struktur des Graphen vorab wenig bekannt ist.

2. (leicht–mittel) **Kürzeste Wege in azyklischen Graphen:** Wir haben in der Vorlesung zwei Varianten für die Kürzeste-Wege-Berechnung in azyklischen Graphen vorgestellt und sind dabei davon ausgegangen, dass alle Knoten vom Startknoten v_0 erreichbar sind – dies lässt sich i.A. aber nicht garantieren (wählen Sie z.B. in einem azyklischen Graphen zwei beliebige Startknoten v_0 und v_0', so ist entweder v_0' von v_0 aus nicht erreichbar oder umgekehrt)

   Wie müssen Sie die beiden Varianten modifizieren, damit diese mit beliebigen Startknoten funktionieren? Ändern sich dadurch die Laufzeiten in O-Notation?

3. (leicht) **Ein Linearzeitalgorithmus:** Zeigen Sie: Das SSSP-Problem mit $\gamma(u,v) = 1$, $(u,v) \in E$, lässt sich in Linearzeit lösen.

4. (mittel) **Dijkstra mit negativen Kantengewichten:** In der Grundvorlesung haben wir gelernt, dass der Dijkstra-Algorithmus mit negativen Kantengewichten i.A. nicht funktioniert. Dies war nur die halbe Wahrheit. Lässt man die explizite Verwaltung der Baummenge weg und nimmt Knoten bei Verringerung der Werte $D(\cdot)$ ggf. auch erneut in R auf (dies kann aufgrund negativer Kantengewichte vorkommen), so arbeitet Dijkstras Algorithmus auch mit negativen Kantengewichten korrekt – nur die Laufzeitabschätzungen stimmen nicht mehr.

   - Beweisen Sie, dass Dijkstras Algorithmus mit negativen Kantengewichten korrekt arbeitet, wenn man zulässt, dass Knoten nach Auswahl aus R zu einem späteren Zeitpunkt wieder in R aufgenommen werden dürfen.
   - Finden Sie Beispiele, für die der so modifizierte Algorithmus möglichst viele Schritte benötigt.

   Hinweis: Überlegen Sie, welcher Ihnen bekannte Algorithmus dem modifizierten Dijkstra-Algorithmus möglichst ähnlich ist.
