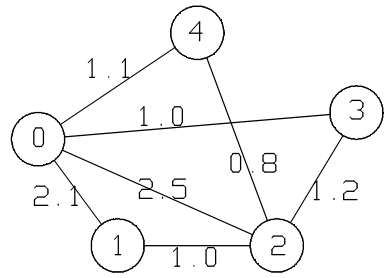
Mi., 8.12.2004 www.r-krell.de Informatik-Lk 13M (Kr)

Arbeitsauftrag/Hausaufgabe: Wegsuchen in Graphen



1 Bekanntlich kann man Graphen in einer Ajazenzmatrix (=2-dim. Tabelle bzw. Reihung) oder mit Adjazenzlisten speichern.

a) Zeichne von Hand die Matrix bzw. die Listen für den Graphen auf!

b) Wie sehen die Objekte aus, die als *inhalt* in jedem (Listen-)Knoten der äußeren Liste (Liste aller Graphen-Knoten, die Nachbarn haben) stehen? Und was wird in die Knoten jeder Nachbar-Liste gepackt?

c) Beschreibe, wie bei Adjazenzlisten der nächste unbesuchte Nachbar eines angegebenen Knotens gefunden wird. Beginne am Kopf der äußeren Liste und verwende die üblichen Listenoperationen!

2 Tiefensuche

a) Beschreibe das Verfahren der Tiefensuche in eigenen Worten

b) Schreibe die Tiefensuche rekursiv in Java

c) Erkläre den Sinn der Schleife, die notfalls alle Nachbarn eines jeden Knotens durchgeht.

d) Welchen Weg findet die Tiefensuche im abgebildeten Graphen bei start=1 und ziel=4? Kosten?

e) Wir hatten beim Labyrinth eine Tiefensuche mit Keller programmiert (z.B. http://home.arcor.de/rkrell/if-java-e.htm#Tiefen). Wie/wo muss das alte Struktogramm abgewandelt werden, um für unseren in einer Matrix gespeicherten Graphen zu gelten?

3 Breitensuche

a) Beschreibe das Verfahren der Breitensuche in eigenen Worten

b) Schreibe die Breitensuche unter Verwendung einer Schlange in Java

c) Erkläre den Sinn der Schleife, die alle Nachbarn eines jeden Knotens durchgeht und ggf. in die Schlange packt!

d) Welchen Weg findet die Breitensuche im abgebildeten Graphen bei start=1 und ziel=4? Kosten?

e) Gilt das Struktogramm von http://home.arcor.de/rkrell/if-java-e.htm#Breiten auch hier?

f) Weder beim Labyrinth noch jetzt hatten wir einen Versuch unternommen, die Breitensuche rekursiv zu programmieren. Wäre das möglich? Welche Probleme entstünden vielleicht?

4 Qualität der Wege und Aufwand

a) Im Unterricht hatten wir die Suchen meist an Graphen mit ungerichteten Kanten getestet (d.h. Kanten, die in beiden Richtungen durchlaufen werden konnten). Eignen sich Tiefen- und/oder Breitensuche auch für gerichtete Graphen, wo Kanten Einbahnstraßen darstellen (können)?

b) Welches der beiden Suchverfahren, Tiefen- oder Breitensuche, findet den besseren Weg in einem (ungerichteten) Graphen mit Kantengewichtung (unterschiedlichen Kosten der Kanten)? Wird unbedingt der günstigste Weg gefunden?

c) Wie groß ist der maximale Aufwand einer Tiefen- oder Breitensuche? Hängt er von der Zahl *n* der Knoten oder von der Zahl *k* der Kanten ab? Wie?

d) Wie oft und wie müssten die Suchen durchgeführt werden, um sicher den günstigsten Weg zu finden? Gesamtaufwand im schlimmsten Fall?

5 Wegsuche aller kürzesten Wege von einem Startpunkt aus (Verfahren von Dijkstra).

Dijkstra ermittelt, wie man von einem festgelegten Startpunkt am günstigsten zu jedem anderen Knoten kommen kann – wobei in einem kantengewichteten Graphen nicht immer der direkte Weg der günstigste sein muss. In drei Tabellen hält er für alle [Ziel-]Knoten jeweils die Gesamtkosten des bisher besten gefunden Weges (anfangs alle ∞), Besucht (anfangs alles false) und den Vorgänger (anfangs überall -1) fest. Beim Startknoten werden jetzt die Kosten 0, true und der Startknoten als eigener Vorgänger in die Tabellen eingetragen. Ausgehend von diesem Startknoten prüft man jetzt alle seine bisher unbesuchten direkten Nachbarn und (*) untersucht für jeden [Ziel-]Knoten, ob der Weg vom Start über den Nachbarn und von dort zum Zielknoten billiger ist, als der bisher gefundene Weg (laut Kostentabelle). Wenn ja, werden die niedrigeren Kosten eingetragen und der Vorgänger des Zielknotens mit dem Nachbarn gefüllt. Gibt es aber keine direkte Verbindung vom Nachbarn zum Ziel, so ist natürlich auch kein günstigerer Weg möglich und die Kosten bleiben hoch, evtl. bei ∞. Wurden alle Nachbarn vom Start überprüft, wird als nächstes der Knoten ausgewählt, der die niedrigsten Kosten in der Kostentabelle stehen hat. Er wird als besucht markiert und jetzt wird für alle seine unbesuchten Nachbarn das Verfahren ab (*) wiederholt.

a) Führe das Dijkstra-Verfahren von Hand für den Graphen aus Aufgabe 1 und Start=1 durch!

b) Schreibe das Struktogramm für Dijkstra!