Universität Karlsruhe (TH)
Fakultät für Informatik
Lehrveranstaltung Informatik II

Sommersemester 2003

Prof. Dr. Peter C. Lockemann
(lockemann@ira.uka.de)
Gábor M. Surányi (suranyi@ira.uka.de)
Daniel Pfeifer (pfeifer@ira.uka.de)
Philipp Bender (bender@ira.uka.de)

Übungsblatt 11

Suchen, Graphen, Dijkstra, Kruskal und dynamisches Programmieren

Abgabe: bis 16.07.2003, 13:30 Uhr in den Einwurfkästen im Untergeschoß des neuen Infobaus

Erreichbare Punkte: 48 Theoriepunkte (48 T), 8 Praxispunkte (8 P)

Aufgabe 1: *Tiefensuche und Labyrinthe (8 P)*

Implementieren Sie den in Aufgabe 6 c) auf Übungsblatt 10 vorgeschlagenen Algorithmus zur Generierung von Labyrinthen mit Hilfe des Labyrinth-Frameworks, das bereits auf Übungsblatt 7 vorgestellt wurde.
Hinweise: Sie können sich dabei an Ihrem eigenen Pseudocode oder an dem der Musterlösung von Aufgabe 6 c) orientieren. Die Implementierung sollte jedoch den Zeitaufwand O(n*m) besitzen.
Zur Vereinfachung der Aufgabe ist dem Übungsblatt noch eine erweiterte (aber noch unvollständige) Version der Datei **`StudentMazeGenerator.java`** beigefügt. Sie enthält die Klasse **`NNode`**, mit Hilfe derer Sie besuchte Knoten bei der Tiefensuche repräsentieren können.

Aufgabe 2: *Graphen (6 T)*

a) Sei G = (V, E) ein gerichteter, ungewichteter Graph. Beweisen Sie: Die Anzahl der Knoten von G, für die die Summe des Ein- und Ausgangsgrads ungerade ist, ist gerade.
(2 T)
Hinweis: Der Ausgangsgrad a(v) eine Knotens $v \in V$ ist definiert als
$a(v) = |\{ (v, y) \mid y \in V, (v, y) \in E \}|$. Die Definition des Eingangsgrads e(v) ist analog:
$e(v) = |\{ (y, v) \mid y \in V, (y, v) \in E \}|$.

b) Beweisen Sie: Falls ein ungerichteter Graph G = (V, E) zusammenhängend ist, gilt $|E| \geq |V| - 1$.
(4 T)

Aufgabe 3: *Dijkstras Algorithmus (7 T)*

a) Berechnen Sie die kürzesten Pfade nach Dijkstra für den Graphen in Abbildung 1. Knoten x sei hierbei der Startknoten. Geben Sie die einzelnen Berechnungsschritte mit Hilfe einer Tabelle in folgender Form an:

Schritt	Ausgewählter Knoten	Besuchte Knoten	Knoten 1	…	Knoten n

(3 T)

b) Geben Sie an, wie man Dijkstras Algorithmus verändern muß, damit er neben der Länge auch die Anzahl der kürzesten Wege berechnet. Modifizieren Sie hierzu den Algorithmus von Folie 37 aus Kapitel 11 entsprechend! (2 T)

c) Erläutern Sie an einem Beispiel, warum Dijkstras Algorithmus zur Bestimmung kürzester Pfade versagen kann, wenn man mit einer Gewichtungsfunktion c wie in der Vorlesung arbeitet, die aber auf beliebige reelle Zahlen abbildet. Geben Sie hierzu folgendes an:
   - einen Beispielgraphen G, bei dem der Algorithmus versagt,
   - den Startknoten,
   - die Berechnungsschritte des Algorithmus,
   - den Knoten, für den ein falscher Wert berechnet wird.

   (2 T)

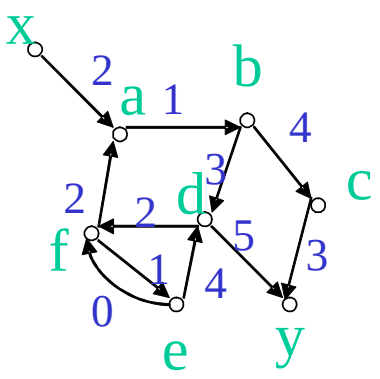

Abbildung 1: Gerichteter Graph mit positiven Kantengewichten

Aufgabe 4: *Das Flaschenhals-Problem (8 T)*
Gegeben seien n Städte die durch Straßen miteinander verbunden sein können. Über die Straßen führen Brücken, so daß die maximale Höhe eines Lastwagens, der eine Straße benutzen will, beschränkt ist. Die Aufgabe besteht darin, die maximale Höhe eines Lastwagens zu bestimmen, der von einer festen Stadt s zu eine beliebigen anderen Stadt fahren kann.

a) Formalisieren Sie das Problem mit Hilfe der Graphentheorie! (4 T)
b) Beschreiben Sie einen Algorithmus, der das Problem mit dem Aufwand $O(m + n*\log n)$, wobei m die Anzahl der direkten Städteverbindungen und n die Anzahl der Städte repräsentiert. (4 T)

Aufgabe 5: *Kruskals Algorithmus (6 T)*

a) Berechnen Sie den Spannbaum nach Kruskal für den Graphen in Abbildung 2.
Geben Sie dabei die einzelnen Berechnungsschritte mit Hilfe einer Tabelle in folgender Form an:

nächste Kante (Startknoten, Endknoten)	Wert der Kante	Teil des Baumes (j/n)

(4 T)

b) Beschreiben Sie, wie Kruskals Algorithmus zur Generierung von Labyrinthen verwendet werden kann! (Das entsprechende Labyrinthproblem wurde in Aufgabe 1 auf Übungsblatt 7 eingeführt.) (2 T)
c) **Sonderaufgabe (für diese Teilaufgabe gibt es keine Punkte):** Wie beurteilen Sie die Qualität der verschiedenen Labyrinth-Generierungsalgorithmen, die auf den Übungsblättern behandelt wurden? Stellen entsprechende Kriterien auf und ordnen Sie die verschiedenen Algorithmen danach ein!

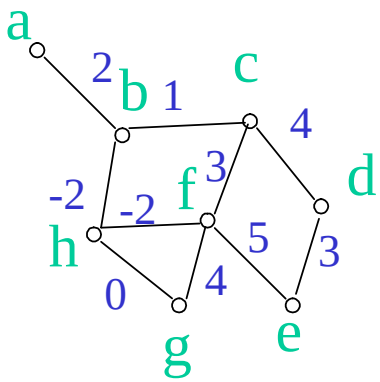

Abbildung 2: Ungerichteter, gewichteter Graph

Aufgabe 6: *Die Pert-Methode (12 T)*

Mit der Pert-Methode wird der kürzeste Pfad zwischen zwei festgelegten Knoten x, y in einem gerichteten, gewichteten Graph G = (V, E, ρ) bestimmt. Dabei seien die Kantengewichte reelle Zahlen und $\rho: E \rightarrow R$ die entsprechende Kantengewichtungsfunktion.
Der Algorithmus ist der folgende:

1) Gibt es keinen Pfad von x nach y, dann terminiert der Algorithmus.
2) Sei λ eine (weitere) Funktion, die den Knoten im Graphen reelle Zahlen zuweist und wie folgt definiert ist:
$$\lambda(p) = \begin{cases} 0, \text{falls } p = x \\ \infty, \text{falls } p \neq x \end{cases}.$$
3) Berechne die Zahl λ(q)-λ(p) für alle Kanten (p, q) ∈ E. Solange es eine Kante (p, q) ∈ E gibt, für die λ(q)-λ(p) > ρ((p, q)), dann wähle ein solches (p, q) aus, berechne die Funktion
$$\lambda'(r) = \begin{cases} \lambda(p) + \rho((p,q)), \text{falls } r = q \\ \lambda(r), \text{falls } r \neq q \end{cases}$$, und ersetze dann λ mit λ'.
Wenn es keine solche Kante gibt, gehe zu Schritt 4).
4) Markiere eine Kante (p, y) ∈ E, für die λ(y)-λ(p) = ρ((p, y)) gilt. Wenn p ≠ x ist, dann spielt p die Rolle von y und Schritt 4) wird wiederholt. Andernfalls gehe zu Schritt 5).
5) Die markierten Kanten von G formen einen minimalen Pfad von x nach y, und der minimale Wert des Pfades ist λ(y).

a) Berechnen Sie den kürzesten Pfad von x nach y nach Pert für den Graphen in Abbildung 3. Geben Sie dabei die einzelnen Berechnungsschritte in Schritt 3 des Algorithmus mit Hilfe einer Tabelle in folgender Form an:

	0	1: ρ(?→?) =?	2: ρ(?→?) =?	3: ρ(?→?) =?	…	?: ρ(?→?) =?
λ(x)					…	
λ(a)					…	
λ(b)					…	
λ(c)					…	
λ(d)					…	
λ(e)					…	
λ(f)					…	
λ(y)					…	

(4 T)

b) Was für eine Rolle spielt die am Anfang verwendete Funktion λ aus Schritt 2? (1 T)
c) Schlagen Sie 2 verschiedene Algorithmen vor, die man in Schritt 1) verwenden könnte! (1 T)
d) Was für einen Aufwand hat der ganze Algorithmus, wenn das beste Verfahren aus c) bei Schritt 1) verwendet wird? Begründen Sie Ihre Antwort! (3 T)
e) Der Algorithmus funktioniert nicht, wenn ein gerichteter Zyklus mit negativer Kantengewichtsumme in G existiert. Was passiert dann? Zeigen, daß der Algorithmus mit einem solchem Zyklus im Allgemeinen nicht korrekt arbeitet! (3 T)

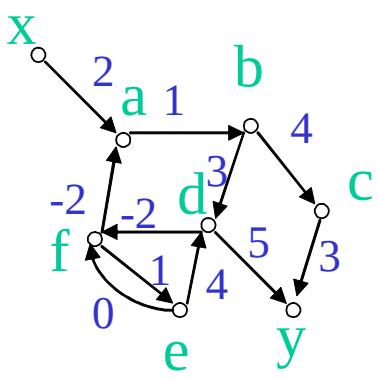

Abbildung 3: Gerichteter, gewichteter Graph

Aufgabe 7: *Dynamisches Programmieren (9 T)*
Gegeben sei eine Menge natürlicher Zahlen S = { c_1, c_2, …, c_m } und eine natürliche Zahl d. Es soll nun eine Teilmenge I von { 1, …, m } berechnet werden, so daß $\sum_{i \in I} c_i = d$ gilt. Der folgende Teile-und-Herrsche-Algorithmus (in Java-Pseudocode) löst dieses Problem über den Methodenaufruf **`subsetSum(S, 1, d)`**:

```
boolean subsetSum(Set s, int k, int h) {
  if (s.isEmpty()) return h == 0;

  boolean res = subsetSum(s \ { ck }, k + 1, h);
  if (res) return true;

  return subsetSum(s \ { ck }, k + 1, h – ck);
}
```

a) Begründen Sie (informell) die Korrektheit des Algorithmus! (2 T)
b) Berechnen Sie den Zeitaufwand des Algorithmus im O-Kalkül! (1 T)
c) Entwerfen Sie (auf Basis des oben angegebenen Teile-und-Herrsche-Algorithmus) einen neuen Algorithmus, der mit dynamischer Programmierung arbeitet und den Zeitaufwand O(m * d) aufweist. (6 T)