Technische Universität
Kaiserslautern

Fachbereich Informatik
AG Algorithmen und Komplexität

Ausgabedatum: 09.01.2014
Version: 2014-01-10 12:34

11. Übungsblatt für Track ε zur Vorlesung Entwurf und Analyse von Algorithmen, WS 13/14

Abgabe: *Bis* ***Freitag****, 17.01.2014, 12:00 Uhr, Kasten im Treppenhaus 48-6.*

Wir wünschen ein frohes und erfolgreiches Jahr 2014!

Der Fehler der Woche

[Nach Betrachtung einer Prozedur.]
Alles nur konstante Laufzeit, da keine Schleifen.

Was ist falsch?

Basisaufgaben

B1/2: Engstellenoptimale Spannbäume

2+3 Punkte

Sei $G = (V, E, g)$ ein gewichteter Graph und $T = (V, E')$ ein Spannbaum von G. Wir bezeichnen die Kante

$$e^*(T) := \arg\max_{e \in E'} g(e)$$

als *Engstelle* von T und entsprechend

$$g^*(T) := \max_{e \in E'} g(e)$$

als *Engstellengewicht* von T. Unter allen Spannbäumen von G nennen wir die mit minimalen Engstellengewichten $g^*(\_)$ auch *engstellenoptimale Spannbäume*.

B1: Seien G ein (beliebiger) Graph und T ein Spannbaum von G. Zeigen oder widerlegen Sie:

$$T \text{ engstellenoptimaler Spannbaum} \implies T \text{ minimaler Spannbaum.}$$

B2: Seien G ein (beliebiger) Graph und T ein Spannbaum von G. Zeigen oder widerlegen Sie:

$$T \text{ minimaler Spannbaum} \implies T \text{ engstellenoptimaler Spannbaum.}$$

B3: Durchmesser von Graphen

3 Punkte

Zur Erinnerung: Der Durchmesser eines (Di)Graphen $G = (V, E)$ ist definiert als

$$d(G) := \max \{\text{dist}(u, v) \mid u, v \in V\} \ ,$$

wobei $\text{dist}(u, v)$ in unmarkierten (Di)Graphen die Anzahl Kanten auf einem (nach Kantenzahl) kürzesten Weg von u nach v ist.

Entwerfen Sie einen Algorithmus, der in Zeit $\mathcal{O}(|V|^3)$ den Durchmesser eines beliebigen (Di)Graphen findet. Beweisen Sie die Korrektheit und Laufzeitschätzung Ihres Entwurfs.

B4: Algorithmus von Prim

4 Punkte

Beschreiben Sie, wie Prims Algorithmus (Seite 199) möglichst effizient implementiert werden kann, und bestimmen Sie anschließend die Worst-Case Laufzeit (in $\mathcal{O}$-Notation) Ihrer Implementierung (als Funktion der Anzahl Knoten und Kanten des verarbeiteten Graphen).

Hinweis: Eine möglichst effiziente Implementierung muss darauf achten, dass die *billigste kreuzende Kante* schnell bestimmt werden kann.

Aufbauaufgaben

32. Aufgabe

2 Punkte

Gegeben sei ein markierter Digraph $G = (V, E, r)$ mit $r : E \to [0, 1] \subseteq \mathbb{R}$. Wir interpretieren die Kanten $e \in E$ als Leitungen eines Kommunikationsnetzwerks und deren Markierung als die *Zuverlässigkeit* der entsprechenden Verbindung (Kante), indem wir $r(e)$ als die Wahrscheinlichkeit dafür auffassen, dass eine Kommunikation über Verbindung e nicht scheitert. Dabei nehmen wir an, dass alle Wahrscheinlichkeiten unabhängig voneinander sind.

Entwerfen Sie einen Algorithmus, der in Zeit $\mathcal{O}(|V|^2)$ die verlässlichste Verbindung zwischen zwei gegebenen Knoten berechnet.

33. Aufgabe

2 Punkte

Sei $G = (V, E)$ mit $V = [1..n]$ ein zusammenhängender Graph mit Markierungsfunktion $g : E \to \mathbb{R}$ – man beachte den kleinen, aber wesentlichen Unterschied zu den bisher betrachteten Markierungsfunktionen – und $s \in V$ fest.

Der notationellen Einfachheit definieren wir eine auf beliebige Knotenpaare erweiterte Gewichtsfunktion $h : V \times V \to \mathbb{R} \cup \{\infty\}$ so:

$$h(i,j) := \begin{cases} 0 & , i = j \\ g(\{i,j\}) & , i \neq j \wedge \{i,j\} \in E \\ \infty & , \text{ sonst} \end{cases} . \tag{1}$$

Nun sei für $i, j \in [1..n]$ folgende Rekursionsgleichung gegeben:

$$B_s(i,j) = \begin{cases} h(s,j) & , i = 1 \\ \min_{k \in [1..n]} B_s(i-1,k) + h(k,j) & , \text{ sonst} \end{cases} . \tag{2}$$

Zeigen Sie: Wenn G keine Kreise mit negativem Gesamtgewicht hat, so löst B_s das SSSPP für G und s, das heißt $B_s(n,t)$ gibt die Länge eines kürzesten Weges von s nach t an.

34. Aufgabe

4 Punkte

Wir betrachten folgenden Algorithmus zur Berechnung eines minimalen Spannbaumes: Wie in Kruskals Algorithmus konstruieren wir den Baum, indem wir Kanten in einen

Wald von Teil-Spannbäumen einfügen. Dabei gehen wir in Phasen vor, wobei wir in jeder Phase mehrere Kanten einfügen. Im Detail suchen wir in jeder Phase für jeden vorliegenden Teil-Spannbaum die kürzeste Kante, die ihn mit einem anderen Teil-Spannbaum des Waldes verbindet. Dann fügen wir all diese Kanten in den zu konstruierenden Baum ein (dieses Vorgehen ist als Boruvkas Algorithmus bekannt).

Zeigen Sie, dass Boruvkas Algorithmus einen minimalen Spannbaum konstruiert, wenn alle Kantengewichte paarweise verschieden sind. Beschreiben Sie detailliert, wie Boruvkas Algorithmus möglichst effizient implementiert werden kann und bestimmen Sie anschließend die Worst-Case Laufzeit (in $\mathcal{O}$-Notation) Ihrer Implementierung (als Funktion der Anzahl Knoten und Kanten des verarbeiteten Graphen).

35. Aufgabe

2 + 3 Punkte

Erinnern Sie sich noch an die Enten aus Aufgaben B4.4 und 10 (Blatt 4)? In dieser Aufgabe wollen wir untersuchen, wie wir die damals auf einer Modellierung mit Arrays basierenden Algorithmen mit seitdem gewonnen Erkenntnissen ausstechen können.

a) Modellieren Sie das Szenario mittels (Di)Graphen; berücksichtigen Sie dabei die weiteren Teilaufgaben!

   Wie lösen Sie Aufgabe B4.4 in diesem Modell?

b) Entwerfen Sie einen Algorithmus im Modell von a), der das Problem aus 10 a) löst. Begründen Sie die Korrektheit des Algorithmus und analysieren Sie seine Laufzeit!

36. Aufgabe

7 Punkte

Sei $G = (V, E, g)$ ein gewichteter Graph und seien $T_1 = (V, E_1)$ und $T_2 = (V, E_2)$ minimale Spannbäume von G. Wir bezeichnen mit

$$\text{count}(A, c) = |\{e \in A \mid g(e) = c\}|$$

die Anzahl Kanten eines bestimmten Gewichts c in einer Kantenmenge $A \subseteq E$.

Zeigen oder widerlegen Sie: Für alle $c \in \mathbb{R}$ gilt

$$\text{count}(E_1, c) = \text{count}(E_2, c) \ .$$