Algorithmen und Programmieren III, WS 1999/2000 — 11. Übungsblatt

Abgabe bis Donnerstag, 10. Februar 2000

54. (6 Punkte) Triangulierungen eines konvexen n-Ecks; binäre Bäume mit $n-2$ Knoten.

    (a) (6 Punkte) Schreiben Sie ein Programm, das alle möglichen Gradlisten der Triangulierungen eines konvexen n-Ecks P_n erzeugt und abzählt. (Dies ist die Umkehrung zu Aufgabe 27d.) Dokumentieren Sie die Ausgabe für $n = 3, 4, 5, 6, 7, 8$. Bestimmen Sie für $n = 9$ nur die Anzahl.

    Ein möglicher Zugang ist, aus den Gradlisten für $(n-1)$-Ecke die Gradlisten für n-Ecke zu erzeugen. Achten Sie darauf, dass Sie Gradlisten nicht mehrfach erzeugen. Dabei kann es hilfreich sein, die Gradlisten in einen digitalen Suchbaum (*trie*) zu speichern.

    (b) (0 Punkte) Man kann Triangulierungen, die durch Drehungen ineinander übergeführt werden, als äquivalent betrachten. (Dies entspricht einer zyklischen Verschiebung der Gradliste.) Es soll nun aus jeder Äquivalenzklasse nur ein Repräsentant erzeugt werden. (Bei $n = 5$ gibt es also nur eine Lösung.)

    (c) (0 Punkte) Die *Gradfolge* ist die sortierte Gradliste. So hat zum Beispiel jede Triangulierung des P_5 die Gradfolge $(0, 0, 1, 1, 2)$. Erzeugen Sie alle möglichen Gradfolgen der Triangulierungen des n-Ecks.

    (d) (∞ Punkte) Schreiben Sie ein effizientes Programm, das testet, ob eine gegebene Folge $(b_1, \dots, b_n)$ die Gradfolge einer Triangulierung ist.

55. (5 Punkte) Ein Iterator für Mengen.

    (a) (3 Punkte) Implementieren Sie den *Iterator*, den Sie in Aufgabe 51 spezifiziert haben. Sie können von der in der Vorlesung besprochenen Implementierung für Mengen mit bis zu 100 Elementen[1] ausgehen.

    (b) (2 Punkte) Geben Sie die Abstraktionsfunktion an.

    (c) (5 Zusatzpunkte) Beweisen Sie die Korrektheit Ihrer Implementierung.

56. (0 Punkte) In einem Baum sei n_i die Anzahl der Knoten mit i Kindern, das heißt, n_0 ist die Anzahl der Blätter, usw. Die Gesamtzahl der Knoten ist $n = n_0 + n_1 + n_2 + n_3 + \cdots$. Beweisen Sie, dass $n \leq 2(n_0 + n_1) - 1$ ist.

57. (a) (0 Punkte) Konstruieren Sie einen optimalen binären Code-Baum (Huffman-Baum) für $n = 15$ Knoten mit den Häufigkeiten $p_i = 1/i$, $(i = 1, \dots, 15)$.

    (b) (5 Punkte) Wie kann man einen optimalen binären Code (Huffman-Code) in *linearer* Zeit konstruieren, wenn die Häufigkeiten p_i in *sortierter* Reihenfolge gegeben sind?

58. (0 Punkte) Welche Bedeutung hat die Flächenformel $\frac{1}{2} \cdot |\sum_i (x_i y_{i+1} - y_i x_{i+1})|$, wenn die Folge der Punkte $(x_i; y_i)$ gar kein Polygon beschreibt, weil sich zum Beispiel Kanten kreuzen?

59. (a) (3 Punkte) Berechnen Sie die Fläche des Fünfecks[2] mit den Ecken $(-1{,}3; 10000{,}12)$, $(-0{,}253; 10000{,}47)$, $(0{,}69; 10000{,}33)$, $(1{,}529; 10002{,}12)$, $(-0{,}783; 10001{,}05)$ mit der Formel aus der vorigen Aufgabe. Berechnen Sie auch den Flächeninhalt des um den Vektor $(0; -10000)$ verschobenen Fünfecks. Welches Ergebnis halten Sie für das genauere? (Bergünden Sie Ihre Antwort.)

    (b) (0 Punkte) Was passiert, wenn man das Fünfeck um den Vektor $(10000; -10000)$ verschiebt? Wie erklären Sie diese Ergebnisse?

[1] `http://www.inf.fu-berlin.de/~rote/Lere/alp3/Menge.java`

[2] `http://www.inf.fu-berlin.de/~rote/Lere/alp3/5eck`

60. (0 Punkte) Verbesserung des Streckenschnittproblems.
    (a) Erläutern Sie, wie man den in der Vorlesung besprochenen plane-sweep-Algorithmus zum Ausgeben aller k Schnittpunkte von n Strecken so verbessern kann, dass er nur $O(n)$ Speicher benötigt. (Man muss dazu aus der Ereigniswarteschlange auch zwischendurch Ereignisse löschen.)
    (b) Wie muss man den Algorithmus modifizieren, dass er auch bei mehrfachen Schnitten (mehr als zwei Strecken gehen durch einen Punkt) funktioniert?
    (c) Erläutern Sie, welche zusätzlichen entarteten Fälle außer den eben erwähnten Mehrfachschnitten auftreten können.

61. (0 Punkte) Triangulierung eines ebenen Polygons.
    Triangulierungen eines konvexen n-Ecks sind in Aufgabe 27 definiert worden. Triangulierungen eines beliebigen[3] (auch nicht-konvexen) n-Ecks sind genauso definiert. Eine *Trapezoidierung* eines n-Ecks ist eine Zerlegung der Polygons in Dreiecke und Trapezoide,[4] die dadurch entsteht, dass man von jedem Knoten eine vertikale Strecke so weit nach oben und nach unten zieht, bis sie eine andere Polygonseite trifft. Die entstandenen Strecken, die außerhalb des Polygons liegen, werden gelöscht.
    (a) Zeigen Sie, dass auf diese Art wirklich eine Trapezoidierung entsteht.
    (b) Entwerfen Sie einen Algorithmus zur Bestimmung einer Trapezoidierung. (Sie können zum Beispiel dem plane-sweep-Muster folgen.)
    (c) (5 Zusatzpunkte) Beweisen Sie, dass jedes Polygon trianguliert werden kann.
    (d) (5 Zusatzpunkte, alternativ zu (c)) Wie kann man aus einer Trapezoidierung eine Triangulierung gewinnen?

62. (4 Punkte) Bestimmen Sie einen kürzesten spannenden Baum im vollständigen[5] Graphen mit den Knoten $V = \{5, 6, \ldots, 15\}$, wo die Länge der Kante zwischen den Knoten i und j gleich das kleinste gemeinsame Vielfache von i und j ist. Ist die Lösung eindeutig?

63. (0 Punkte) Zeigen Sie, dass der folgende Algorithmus einen kürzesten spannenden Baum T in einem zusammenhängenden Graphen (V, E) mit Kantengewichten w_{ij} berechnet:

    $T := \emptyset$; $S := V$; $u :=$ irgendein Knoten aus V;
    $d_u := 0$; $d_v := \infty$ für alle Knoten $v \neq u$, $v \in V$;
    while $S \neq \emptyset$ **do**
        wähle ein $i \in S$ mit kleinstem Wert d_i;
        if $i \neq u$ **then** $T := T \cup \{(i, f_i)\}$; **end if**;
        für alle Kanten (i, j), die von i ausgehen **do**
            if $w_{ij} < d_j$ **then** $d_j := w_{ij}$; $f_j := i$; **end if**;
        end für;
    end while;

    Worin unterscheidet sich dieser Algorithmus (der übrigens auch von E. W. Dijkstra stammt) vom Dijkstra-Algorithmus für kürzeste Wege?

64. (0 Punkte) Gegeben sei ein vollständiger[5] Graph $G = (V, E)$, dessen Knoten V Punkte in der Ebene sind, und wo die Kantenlängen den (Euklidischen) Abständen zwischen den Punkten entsprechen. Beweisen Sie:
    (a) In einem kürzesten spannenden Baum können sich nie zwei Kanten kreuzen.
    (b) (5 Zusatzpunkte) In einem kürzesten-Wege-Baum (mit einem beliebigen Startknoten) können sich nie zwei Kanten kreuzen.

    Bleiben diese Aussagen auch gültig, wenn der Graph nicht vollständig ist?

---

[3] Wir betrachten nur *einfache* Polygone; das sind solche, deren Rand eine einzelne geschlossene Kurve ohne Selbstüberschneidungen (eine Jordan-Kurve) ist.
[4] Ein Trapezoid ist ein Viereck mit zwei parallelen Seiten.
[5] Ein Graph ist *vollständig*, wenn zwischen allen Paaren von Knoten eine Kante verläuft.