Algorithmen und Programmierung 3, WS 2019/2020 — 8. Übungsblatt

Abgabe bis Freitag, 13. Dezember 2019, 12:00 Uhr, in die Fächer der Tutoren

---

55. Topologisches Sortieren, 11 Punkte

    Betrachten wir die Knoten u eines Graphen in der Reihenfolge, wie der rekursive Aufruf $T(u)$ bei der Tiefensuche *beendet* wird. Das heißt, wir fügen *am Ende* des Programms $T(u)$ aus der Vorlesung[1] folgende Anweisungen ein:

    ```
    num2 = num2+1; T2Nummer[u] = num2;
    ```

    (a) (0 Punkte) Wenn der Vorgänger $f[v]$ von v gleich u ist, welche Beziehung besteht dann zwischen $T2Nummer[u]$ und $T2Nummer[v]$?

    (b) (4 Punkte) Beweisen Sie: Wenn es eine Kante (u, v) mit

    $$T2Nummer[v] > T2Nummer[u]$$

    gibt, dann enthält der Graph einen Kreis.

    (c) (4 Punkte) Wie kann man diesen Kreis bestimmen? Schreiben Sie ein Programmstück in Pseudocode, das den Kreis aus u und v, den Vorgängerzeigern im Tiefensuchbaum und der T2Nummerierung bestimmt.

    (d) (3 Punkte) Verwenden Sie Aufgabe (b), um aus der T2Nummerierung eines kreisfreien Graphen eine topologische Sortierung zu konstruieren. Beschreiben Sie den Algorithmus in Worten.

56. Algorithms von Dijkstra, 5 Punkte

    Wenden Sie den Algorithms von Dijkstra zur Bestimmung der kürzesten Wege im folgenden gerichteten Graphen an: Die Knotenmenge $V = \{1, 2, \ldots, 6\}$, und der Graph ist vollständig, das heißt, von jedem Knoten gibt es eine Kante zu jedem anderen Knoten. Die Kantenlängen sind $c_{ij} = 3^{j-i}$ für $i < j$ und $c_{ij} = i - j$ für $i > j$. Der Startknoten ist Knoten 2.

    Geben Sie nach jeder Iteration die Werte der Zeiger und Markierungen an. (Die internen Daten der Prioritätswarteschlange brauchen Sie nicht anzugeben.)

57. Negative Kantenlängen, 4 Punkte

    Konstruieren Sie einen Graphen, der auch Kanten mit negativer Länge enthält, und bei dem der Algorithmus von Dijkstra die kürzesten Wege nicht richtig bestimmt. Der Graph darf keine Kreise negativer Länge enthalten.

58. Ausgeglichene Bäume, 0 Punkte

    Frau C. B. Daflich hat die Schwäche des Verfahrens von Herrn Dom (Aufgabe 31 vom 4. Übungsblatt) erkannt: Die entstehenden Bäume sind nicht ausgeglichen! Sie hat daher eine Methode entwickelt, um in linearer Zeit einen „Suchbaum“ im Sinn von Aufgabe 31 herzustellen, der nur Höhe $h = O(\log n)$ hat. Können Sie auch einen solchen Algorithmus finden? Geht es sogar in der optimalen Höhe $h = \lfloor \log_2 n \rfloor$?

---

[1] `https://mycampus.imp.fu-berlin.de/access/content/group/fdc1e04e-407d-4d35-ac90-d6f03f853bc2/Programme/Tiefensuche.py`