Algorithmen und Programmierung 3, WS 2020/2021 — Klausur

Elektronische Abgabe bis Dienstag, 2. März 2021, 16:00 Uhr.

---

1. Baue ein zusammenhängendes Netz, 10 Punkte

   Wir wollen eine Menge T von Kanten eines zusammenhängenden ungerichteten Graphen G mit n Knoten V und m Kanten E auswählen, sodass man über die Kanten von T von jedem Knoten zu jedem anderen gelangen kann. Für jede Kante uv sind positive Kosten c_{uv} gegeben. Ein Teilmenge $E_0 \subseteq E$ von m_0 Kanten ist schon gebaut. Das heißt, diese Kanten müssen auf jeden Fall zu T gehören.

   Wie kann man eine Teilmenge T mit möglichst kleinen Gesamtkosten finden?

   Sie können einen eigenen Algorithmus entwickeln oder das Problem auf ein bekanntes Problem zurückführen und sich auf die Algorithmen aus der Vorlesung beziehen. Beschreiben Sie Ihre Lösung in Worten oder in Pseudocode. Analysieren Sie die Laufzeit und den Speicherbedarf Ihres Algorithmus in Abhängigkeit von n, m, und $m_0 = |E_0|$.

2. 2-3-Bäume, 10 Punkte

   Fügen Sie der Reihe nach die Schlüssel 1, 2, 3, 4, 5, 6, 7, 8 in einen leeren 2-3-Baum ein. Löschen Sie anschließend 3 und 6, und fügen Sie die Schlüssel 9 und 10 ein.

   Zeichnen Sie den 2-3-Baum *nach jeder Umstrukturierung* (nicht nur am Ende jeder Einfüge- oder Löschoperation), und markieren Sie die Knoten, wo die 2-3-Baum-Eigenschaft verletzt ist.

   Es reicht, wenn Sie die Gestalt des 2-3-Baums und die Schlüssel in den Blättern zeichnen; die Schlüssel in den inneren Knoten brauchen Sie nicht anzugeben.

3. Speichern von Mengen als Skiplisten, 10 Punkte

   Wir haben Skiplisten L_1 und L_2 für zwei Mengen M_1 und M_2 gegeben, mit Größe $|L_1|$ und $|L_2|$ und Höhe H_1 und H_2. Wir wissen, dass die erwartete Größe einer zufällig erstellten Skipliste mit n Elementen $O(n)$ ist.

   (a) Beschreiben Sie einen effizienten Algorithmus zum Erstellen einer Skipliste für die Menge $M_1 \cup M_2$. Analysieren Sie die Laufzeit Ihres Algorithmus.

   (b) Nehmen wir an, dass $x_1 < x_2$ für alle $x_1 \in M_1, x_2 \in M_2$ gilt. Das heißt, dass jedes Element von L_1 vor jedem Element von L_2 kommt. Beschreiben Sie einen Algorithmus, der die Listen L_1 und L_2 in Laufzeit $O(\max(H_1, H_2))$ vereinigt.

   Beschreiben Sie die Algorithmen in Worten oder in Pseudocode. Geben Sie jeweils an, ob es sich um die Laufzeit im schlimmsten Fall oder um die erwartete Laufzeit handelt.

4. Implementierung eines abstrakten Datentyps, 10 Punkte

   Wir wollen Mengen S von Zahlen speichern, sodass folgende Operationen möglich sind:

Operation	Vorbedingung	Ergebnis	Nachbedingung
einfügen(x)	keine	keines	$S = S^{\text{alt}} \cup \{x\}$
löscheMin$(\,)$	$S \neq \emptyset$	$\min S^{\text{alt}}$	$S = S^{\text{alt}} \setminus \{\min S^{\text{alt}}\}$
löscheMax$(\,)$	$S \neq \emptyset$	$\max S^{\text{alt}}$	$S = S^{\text{alt}} \setminus \{\max S^{\text{alt}}\}$

   Beschreiben Sie *in Worten oder in Pseudocode* für jede der folgenden Datenstrukturen, wie die beiden Operationen *löscheMin* und *löscheMax* möglichst effizient implementiert werden können, und geben Sie gute asymptotische obere Laufzeitschranken an.

   (a) Doppelt-verkettete sortierte Liste mit Zeiger zum ersten und letzten Element

   (b) AVL-Baum

   (c) Halde (eine binäre min-Halde, wo das kleinste Element in der Wurzel steht)
