Tentamen Algoritmen en Datastructuren

vrijdag 20 april 2007, 9 - 12 uur

Het tentamencijfer T is (p/10) + 1, waarbij p het totaal aantal behaalde punten is.

Met de zinsnede 'geef een algoritme' in een opgave wordt bedoeld:

beschrijf een algoritme in pseudocode (dus niet in Java),
licht de werking ervan toe,
beargumenteer de correctheid.

1. (30 punt) Deze opgave gaat over het sorteren van lijsten van gehele getallen. Vergelijken van twee getallen gaat in $O(1)$ tijd. n is het aantal getallen in de lijst. Alle getallen in de lijst zijn verschillend.

   (a) Geef een algoritme voor merge-sort. Laat zien dat de tijdscomplexiteit $O(n \log n)$ is.

   (b) Laat zien dat de tijdscomplexiteit van een sorteeralgoritme dat op het vergelijken van getallen gebaseerd is, nooit beter kan zijn dan $O(n \log n)$.

   (c) Nu is tevens gegeven dat alle getallen van de lijst liggen in het interval $[0, 100 * n - 1]$. Geef een algoritme dat de lijst sorteert in $O(n)$ tijd. Hoeveel geheugen gebruikt het algoritme?

2. (30 punt) Gegeven is een ongerichte gewogen graaf $G = \langle V, E \rangle$, waarin alle gewichten verschillend zijn. Verder is gegeven dat G samenhangend en enkelvoudig is, dus geen self-loops en parallelle kanten bevat.

   (a) Wat is een opspannende boom (spanning tree) van G? En een minimale opspannende boom?

   (b) Nu is V_1, V_2 een partitie van de verzameling V van knopen van G, dus $V_1 \cap V_2 = \emptyset$ en $V_1 \cup V_2 = V$. Beschouw alle kanten van G die een eindpunt in zowel V_1 als V_2 hebben, en laat e van die kanten het laagste gewicht hebben. Laat zien: elke minimale opspannende boom van G bevat e.

   (c) Geef een algoritme dat een minimale opspannende boom van G vindt, met tijdscomplexiteit $O(m \log n)$ (m is het aantal kanten van G, n het aantal knopen). Je mag gebruik maken van een efficiënte priority queue.

3. (30 punt) Deze opgave gaat over complexiteitsklassen van beslissingsproblemen (decision problems).

   (a) Geef definities van de complexiteitsklassen P (polynomiaal) en NP (nondeterministisch polynomiaal).

   (b) Wanneer is een probleem NP-volledig (NP-*complete*)?

   (c) Formuleer een NP-volledig probleem en laat zien dat het in de klasse NP zit (je hoeft dus niet de NP-*volledigheid* te bewijzen).