Tweede Toets Datastructuren

29 juni 2016, 13.30 – 15.30, Educ-Γ.

Motiveer je antwoorden *kort*! Stel geen vragen over deze toets; als je een vraag niet duidelijk vindt, schrijf dan op hoe je de vraag interpreteert en beantwoord de vraag zoals je hem begrijpt. Elke vraag levert evenveel punten, nl. 3, met een totaal van 18. T2 is punten plus 1, gedeeld door 1,8.

1. **Verwijderen uit gelinkte lijst:** In een *doubly linked list* heeft elke `Node` een `key`, `prev` en `next`. In een *singly linked list* ontbreekt de `prev`.

   (a) Geef een methode die `Node` y verwijdert uit een doubly linked list.
   (b) Geef een methode die `Node` y verwijdert uit een singly linked list.
   (c) Moet je y meegeven als `ref` parameter (zowel voor (a) als voor (b))? Leg uit!

2. **Master Theorem:** Bepaal de asymptotische oplossing van deze recurrenties met de Master Theorem.

   (a) $V(n) = 2V(n/3) + 2n$.
   (b) $W(n) = W(n/3) + (\frac{3}{2})^n$.
   (c) $X(n) = 3X(n/2) + O(n\sqrt{n})$.

3. **MinHeap:** Deze vraag gaat over een MinHeap in een 1-based array `A`. In `A` staan deze getallen: `(8, 48, 31, 20, 18, 33, 35, 30, 45, 21, 20, 35)`.

   (a) Welke veranderingen vinden plaats als `Heapify(2)` wordt aangeroepen?
   (b) Vervolgens wordt key `27` toegevoegd. Zeg wat er gebeurt en geef de heap na toevoeging.
   (c) Welke posities in een heap met n keys hebben geen kinderen, welke hebben 1 kind en welke hebben 2 kinderen?

4. **ContainsValue:** Christa slaat records op die een (spelers) *naam* en een *score* bevatten. Zij wil kunnen invoegen/verwijderen/zoeken op *naam*, en zij heeft een efficiente `ZoekSc(s)` nodig die de naam oplevert van een speler met score `s`.

   (a) Christa overweegt opslag in een hashtabel met de *naam* als key. Hoe kan zij een `ZoekSc(s)` uitvoeren?
   (b) Hoeveel tijd kost de `ZoekSc` bij deze opslag (noem de lengte van de hashtabel m en het aantal records n)?
   (c) Kan Christa haar `ZoekSc(s)` sneller krijgen door de records in een zoekboom op te slaan? Leg uit.

5. **Selectie uit Boom:** Gegeven is een binaire zoekboom, geaugmenteerd met het aantal keys per deelboom. Elke knoop bevat een `key`, deelbomen `left` en `right`, en een integer `size` die het aantal keys in de deelboom geeft.

   Geef een *recursieve* methode `KR(node b, int k)` die uit boom `b` het element met rang `k` oplevert (dwz., het op `k` na kleinste element; je mag aannemen dat alle keys verschillend zijn en dat $0 \leq k < b.size$ geldt). De methode moet recursief zijn en de tijd moet lineair zijn in de hoogte van de boom (dus $O(h)$).

6. **Recurrente betrekking:** Los op: $a_0 = 1$, $a_1 = 1$, $a_n = a_{n-1} + 6a_{n-2}$.