Tweede Toets Datastructuren

27 juni 2018, 13.30 – 15.30, Olympos Hal 2.

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. Vraag 1, 4 en 6 zijn 2pt, vraag 2, 3 en 5 zijn 3pt. Te behalen 15pt, T2 is totaal plus een half gedeeld door 1,5 (max 10). Maak 1 en 2 op de voorkant, 3 en 4 op 2 en 5 en 6 op 3.

1. **Utrechters in bier:** Bob koopt tien flesjes alcoholvrij bier waar als reclameactie foto's van bekende Nederlanders inzitten. Het is bekend dat voor elk flesje de kans op een Utrechter $\frac{1}{3}$ is en dat de plaatjes per flesje onafhankelijk zijn gekozen.
   (a) Wat is de kans dat Bob in zijn biertjes *exact vier* Utrechters aantreft?
   (b) Wat is de verwachting van het aantal Utrechters dat Bob aantreft?

2. **Heap inlezen en bouwen:** Dirk moet een heap maken van n getallen van de invoer. Hij maakt een `List<int> Heap` aan, voegt n maal een getal toe met `x = ...; Heap.Add(x);`, en doet eenmaal `Heap.BuildHeap()`.
   (a) Wat is de complexiteit van `BuildHeap` en hoe werkt die?
   (b) Dirk maakt zich zorgen over de herhaalde `Add(x)`, omdat telkens wanneer de array in `Heap` vol is, een langere wordt aangemaakt en alle waarden overgestapeld, wat lineaire tijd kost. Wat is de totale complexiteit van de n `Add`-operaties? Motiveer kort!
   (c) Ineke zegt dat hij beter elk ingelezen getal direct kan invoegen met `Heap.Insert(x)`. Wat is de complexiteit van Ineke's idee, en is het beter of slechter dan Dirks plan?

3. **De SomStack:** Steven heeft voor zijn algoritme een *SomStack* nodig. Behalve de standaard Stack-operaties heeft die ook de `SomKeys()`, die de som van getallen op de stack oplevert. Steven beschikt over de `Stack` klasse uit C#, maar weet niet of deze als array of gelinkte lijst gemaakt is.
   (a) Wat zijn de drie belangrijkste methoden van een Stack, en wat is hun complexiteit?
   (b) Geef een efficiente implementatie van `SomStack`, die de `Stack` gebruikt.

4. **Recurrentie:** Rij S_n is gedefinieerd door $S_0 = 3$, $S_1 = 1$ en $S_n = 2S_{n-1} + 3S_{n-2}$. Er bestaat een gesloten formule voor S van de vorm $S_n = C.\alpha^n + D.\beta^n$.
   (a) Hoe kun je de α en β bepalen? Geef de uitkomst.
   (b) Hoe kun je de C en D bepalen? Geef de uitkomst en de formule voor S.

5. **Dukkel:** In een lijst A van n verzamelingen is verzameling $A[i]$ een *dukkel* als hij deel is van een latere verzameling, dus als er een $j > i$ is met $A[i] \subseteq A[j]$. Door de binaire representatie is $X \subseteq Y$ in constante tijd te checken, maar voor elke i en j apart $A[i] \subseteq A[j]$ doen is duur (kwadratisch).
   We volgen deze aanpak: kijk *recursief* naar de eerste en tweede helft van A afzonderlijk. Als de eerste helft een dukkel bevat, lever die op en klaar. Als de tweede helft een dukkel bevat, lever die op en klaar. Alleen wanneer geen van de helften een dukkel bevat, vergelijk je elke set in de eerste helft met elke in de tweede helft.
   (a) Geef een recurrente betrekking voor de looptijd van dit algoritme.
   (b) Geef de oplossing van deze betrekking.
   (c) Los op: $G(n) = 2G(n/3) + \sqrt{n}$.

6. **Maximum in Hashtabel:** In een hashtabel (met chaining) `List<int>[m] A` van lengte m zijn n integer keys opgeslagen. Geef een methode die de *grootste key* bepaalt en zeg wat de complexiteit van deze methode is.