Eerste serie sommen

1. Een array is gevuld met waarden 0 en 1.

   (a) (2 punten) Beschrijf een $O(n)$ algoritme die het array herschrijft zodat eerst alle 0en en daarna alle 1en komen.

   (b) (2 punten) Waarom kunt u geen sorteeralgoritme voor deze opgave gebruiken?

2. In een ongeordende lijst ℓ is de worst-case tijd die nodig is om een element op te zoeken $O(n)$, met n het aantal elementen van ℓ. Als je een aantal keer een element moet opzoeken kun je proberen de tijd te optimaliseren. Laat s een rij zoekopdrachten zijn. Het is duidelijk dat het aantal vergelijkingen minimaal is als de elementen in ℓ gesorteerd zijn naar aflopende frequentie in s.

   (a) (1 punt) Waarom?

   Omdat je vantevoren niet *weet* welke elementen hoe vaak gezocht gaan worden, kan de optimale volgorde niet gebruikt worden. Wel zijn er verschillende heuristieken bedacht om te proberen de ordening van ℓ "on the fly" aan te passen zodat het optimum benaderd wordt. Eén van die heuristieken hebben we ook al bij databases gezien, namelijk de move-to-front (MFT) strategie (in databases MRU). Hierbij wordt een element, als dat opgevraagd wordt, in ℓ meteen verwisseld met de eerste. We gaan bewijzen dat de MFT strategie voor *elke* rij searches s hoogstens twee keer zo duur is als het optimum. We tellen het aantal vergelijkingen van elementen uit s met elementen in ℓ. We nemen zbda aan dat alle elementen uit s voorkomen in ℓ. De algoritme begint steeds vooraan en vergelijkt een element uit s achtereenvolgens met elementen van ℓ totdat de waarden gelijk zijn.

   (b) (2 punten) Noem een vergelijking van s_i en ℓ_j "goed" als $s_i = \ell_j$ en anders "fout". Laat zien dat het totaal aantal goede searches onafhankelijk is van de gebruikte strategie.

   Merk vervolgens op dat als je s_i zoekt in een rij en vergelijkt met een element $\ell_j \neq s_i$, het aantal keren dat je dat doet *alleen* afhankelijk is van de relatieve positie van s_i en ℓ_j in ℓ en *niet* van de andere elementen in ℓ. Dit heet de paarsgewijze onafhankelijkheid van de foute vergelijkingen. Neem nu eerst aan dat ℓ uit twee elementen, A en B, bestaat. Laat A precies m keer voorkomen in s, en B precies n keer.

   (c) (2 punten) Laat zien dat het aantal foute vergelijkingen in de optimale strategie hoogstens gelijk is aan $\min\{m, n\}$.

   (d) (2 punten) Laat zien dat het aantal foute vergelijkingen in MFT *in dit geval* hoogstens twee keer zo groot is als de waarde in onderdeel c, misschien plus 1 afhankelijk van de beginsituatie.

   (e) (3 punten) Gebruik de paarsgewijze onafhankelijkheid om de conclusie te trekken.

3. (6 punten) Geef voor de volgende functies telkens een algoritme waarvoor deze functie een overeenkomstige grens op de (tijd)complexiteit is. Verklaar kort het antwoord.

   (a) $o(n^2)$ (b) $\Omega(n)$
   (c) $\Omega(n^2)$ (d) $\omega(2^n)$
   (e) $\theta(\log n)$ (f) $\sim n$

4. (2 punten) Voor welke k geldt $log^k n \in O(n^{1/k})$?

5. (2 punten) Als $f \in O(g)$, geldt dan $\frac{f}{g} \in O(1)$?