Examen Algoritmen en Datastructuren III

Naam : .....

Lees de hele oefening zorgvuldig voordat je begint ze op te lossen!
Als je niet goed verstaat wat de vraag of taak is, vraag het aan de lesgever!
Voor oefeningen die fout verstaan zijn, kunnen geen punten gegeven worden!

Schrijf leesbaar. Oplossingen die niet leesbaar zijn, kunnen ook niet beoordeeld worden.

Als technieken toegepast moeten worden, toon altijd voldoende tussenstappen om te kunnen zien wat er gebeurt en dat de technieken goed verstaan zijn.

Stellingen uit de les mogen natuurlijk altijd gebruikt worden zonder dat het bewijs opnieuw gegeven moet worden (behalve in gevallen waar het expliciet anders staat)!

1. **Hashing (2.5 pt)**

   - Voeg de sleutels met de volgende binaire hashwaarden in de gegeven volgorde toe aan een linear hashing tabel met laadfactor $0,6$ en maximaal 2 sleutels per emmer. In het begin heeft de tabel maar $n = 1$ emmer. Omdat het hier – net zoals in de les – voldoende is de hashwaarden in te vullen, worden de echte sleutels niet gegeven.

     De in te vullen hashwaarden zijn
     `10000`, `00100`, `11000`, `01111`, `01010`

- In welke gevallen moet de emmer met index 0 herverdeeld worden als een linear hashing tabel uitgebreid moet worden? Geef een redenering die aantoont dat jouw antwoord juist is.

- Je moet software schrijven die gebruikt wordt voor berekeningen als klanten geld uit de muur willen trekken. Daarvoor heb je ook een heel grote en uitbreidbare hashtabel nodig. Zou je voor een extensible hashing tabel of een linear hashing tabel kiezen? Geef redenen voor jouw keuze.

- Geef voor linear hashing de geamortiseerde kost van een reeks van n toevoegbewerkingen in het slechtste geval (dus als de hashfunctie niet goed is) en als de hashfunctie wel goed is (je mag bijvoorbeeld stellen dat het verschil tussen het minimaal en maximaal aantal sleutels in een emmerketting ten hoogste een factor van 3 is). Geef uitleg, maar een echt bewijs dat jouw antwoorden juist zijn is niet vereist.

2. **String matching (4 pt)**

- Stel dat een tekst t van lengte l en een zoekwoord z van lengte l' gegeven zijn. Wat is de complexiteit om z in t te zoeken voor de volgende algoritmen:
  - shift-AND
  - Knuth-Morris-Pratt
  - Rabin-Karp

- Pas het in de les geziene algoritme toe om de beste benaderende match van het woord `slagen` in de (rare) tekst `geslaagde_noot_-_slang_energiek` te vinden. Bouw de tabel op en geef voor sommige elementen in de tabel uitleg hoe die berekend zijn. Kies de elementen zo dat je toont dat het algoritme goed is verstaan.

- Het algoritme van Knuth-Morris-Pratt houdt alleen rekening met de informatie waar je een match of mismatch hebt, maar niet met het letterteken in de tekst op de mismatch. In hoe verre zou het helpen ook daarmee rekening te houden? Geef uitleg.

- Pas het algoritme van Knuth-Morris-Pratt toe om de string `gegevens` in `gegeven_gegevenheid_geeft_foute_gegevens` te zoeken. Toon daarbij niet alleen voor het zoeken maar ook voor het opstellen van de verschuivingstabel voldoende tussenstappen.

3. **Extern sorteren (1.5 pt)**

   Beschrijf het algoritme van extern sorteren en bepaal de asymptotische complexiteit voor het geval dat voor de delen die in het geheugen gesorteerd worden bubblesort gebruikt wordt.

4. **Compressie (2 pt)**

   Comprimeer de string `anansi_at_ananas`.

   - Comprimeer de string met LZ77 waarbij je een sliding window van grootte 8 gebruikt.

- Comprimeer dezelfde string door eerst de Burrows-Wheeler transformatie toe te passen en achteraf de move-to-front methode. Kies de lijst voor de codering als de verzameling van alle tekens die nodig zijn in een volgorde, waarbij eerst cijfers (in volgorde), dan speciale tekens zoals “_” en dan lettertekens (in alfabetische volgorde) komen.

5. **Metaheuristieken (4 pt)**

   In dit deel is het niet de bedoeling bijzonder efficiënte algoritmen te ontwikkelen. Het is voldoende alle parameters en constructies te beschrijven en het basisalgoritme te geven om aan te tonen dat de metaheuristieken goed verstaan zijn.

   Het probleem waarvoor heuristieken ontwikkeld moeten worden, is het volgende:

   Probleem: In een lager staan n goederen $g_1, \ldots, g_n$ te koop. Voor elk goed g_i is de prijs $p(g_i)$ die je zelf moet betalen gekend en ook de prijs $e(g_i)$ die je kan krijgen als je het aan een eindverbruiker doorverkoopt.

   Het doel is nu voor een gegeven getal M een verzameling van goederen te vinden zodat de som van de prijzen ten hoogste M is en jouw winst optimaal.

   - Beschrijf een op Variable Neighbourhood Search gebaseerd algoritme voor dit probleem. Drie buurfuncties zijn daarbij voldoende.

- Beschrijf een genetisch algoritme voor dit probleem.

6. **Parallelle algoritmen (2 pt)**

- Beschrijf een algoritme voor een CRCW-machine dat in constante tijd het maximale getal in een invoerarray $a[]$ van lengte n kan vinden.

  Geef de pseudocode voor de processoren.

  De volgende tips kunnen gebruikt worden maar moeten niet gebruikt worden:

  - Probeer niet te zuinig te zijn met processoren: $O(n^2)$ processoren mag.
  - Een element $a[i]$ is maximaal als en slechts als de vraag of $a[i] < a[j]$ voor alle $0 \leq j < n$ het antwoord *fout* geeft.
  - Of een gegeven waarde in een array staat kan je in constante tijd testen (zoals in de les gezien). Maar als je dat gebruikt geef ook hiervoor de pseudocode.

NOG NIET OMDRAAIEN !