Examen Algoritmen en Datastructuren III

Naam : .....

1. (2 pt)

   Linear Hashing

   - Stel dat je 2 sleutels per emmer mag plaatsen en met een bovengrens voor de laadfactor van 0.5 werkt. Stel dat er $n >> 2$ sleutels in de tabel zitten.
     - Hoe duur is 1 opzoekbewerking in het slechtste geval? Is dat een geval dat voor hashing bijzonder interessant is? Geef uitleg.

     - Stel dat je met maar 2 emmers begint en dan de uitbreidbewerkingen toepast. Hoeveel emmers zijn er ten minste en ten hoogste als je mogelijke overflow-emmers meetelt?

- Voeg sleutels met de volgende hash-waarden in de gegeven volgorde toe aan een linear hashing tabel waarbij je in het begin maar 1 (lege) emmer gebruikt. Er mogen twee sleutels in één emmer zitten. Het is hier voldoende de hash-waarden in de tabel te schrijven in plaats van de sleutels (die toch al niet gegeven zijn). Toon voldoende tussenstappen om te zien wat er gebeurt en schrijf altijd expliciet welke emmer herverdeeld moet worden. De hash-waarden zijn (binair):

  000000, 111100, 010110, 111111, 101010, 101100

2. (3 pt)

Compressie

- Wij hebben in de les een bewijs gezien dat je maar $1/255$ste van alle bestanden kan comprimeren. Stel nu dat elke computer ongelofelijk veel ruimte ter beschikking heeft en op voorhand al grote delen kan opslaan die in bestanden kunnen voorkomen (dus heel grote voorgëimplementeerde woordenboeken – b.v. alle bestanden t.e.m. 500 bytes) zodat je op de volgende manier kan comprimeren: je slaat gewoon de index van de onderdelen op en stuurt die door. De index van de woorden kan je natuurlijk nog eens Huffman-coderen. Bepaal voor deze manier van comprimeren een bovengrens voor het aantal bestanden dat je kan comprimeren. Resultaten en bewijzen uit de les mogen geciteerd worden en moeten niet herhaald worden.

- Comprimeer de volgende tekst 1 keer met Huffman en 1 keer met LZ77 (zonder achteraf Huffman toe te passen). Voor LZ77 gebruik een sliding window dat duidelijk groter is dan deze kleine tekst. Wij hebben 2 manieren gezien om de LZ77-code op te schrijven. Kies voor om het even welke.

  `abracadabra_simsalabim`

- Decodeer de volgende met LZW gecodeerde tekst. De nieuwe woorden in het woordenboek beginnen met nummer 256. Geef ook de woorden met code ten minste 256 in het woordenboek. Wees niet verrast als de tekst niet echt zinnig is...

  ```
  a 256 b c a b 257 260
  ```

3. (2 pt)

Prefixcodes

- Gegeven een alfabet met t tekens. Stel dat je een prefixcode met w woorden wil construeren die uit deze tekens bestaan. Beschrijf een algoritme om zo'n prefixcode te construeren zodat de som van de lengten van de w woorden minimaal is.

- Bewijs dat jouw algoritme juist is.

- Pas jouw algoritme toe op $t = 3$ (b.v. alfabet $\{0, 1, 2\}$) en $w = 15$.

4. (2 pt)

Extern sorteren

- Stel dat je de gesorteerde deelbestanden al hebt en nu de bestanden nog tot één groot bestand moet mergen. Hoe bepaal je de volgende sleutel die naar het finale bestand geschreven moet worden op een efficiënte manier?

- Stel dat je blokken van 8 kilobytes in één stap kan lezen en 300 MB in het geheugen ter beschikking hebt. Wat is de maximale grootte van een bestand die je dan met één keer mergen kan sorteren (in de les hebben wij dat *in één stap* genoemd)?

- Wij mergen altijd zoveel bestanden als mogelijk tegelijk in plaats van elke keer maar 2 samen te voegen tot een nieuw bestand omdat dat efficiënter is. Maar wat is **precies** efficiënter? Is de asymptotische complexiteit van het slechtste geval beter?

5. **PRAM algoritmen** (2 pt)

   Gegeven is een array $(a_1, \ldots, a_n)$ van getallen. Het probleem van de prefix-minima bestaat in het bepalen van het kleinste element uit $\{a_1, a_2, \ldots, a_i\}$, voor elke i, $1 \leq i \leq n$.

   Geef een PRAM algoritme dat dit probleem oplost in $\Theta(\log n)$ parallelle tijd met $\Theta(n)$ werk.

6. **Communicatie-algoritmen** (2 pt)

   (a) Geef een algoritme voor een all-to-all broadcast op een d-dimensionale hyperkubus. Bespreek de totale communicatietijd van het algoritme.

   (b) Geef een algoritme voor een single-node scatter op een 2-dimensionaal rooster (met wrap-around) met $n = \ell^2$ processoren. Bespreek de totale communicatietijd van het algoritme.

7. **String-matching** (3 pt)

   (a) Geef een voorbeeld waarbij de occurrence-heuristiek in het algoritme van Boyer-Moore het patroon P over $|P|-1$ posities naar links zou verschuiven (m.a.w. een negatieve verschuiving).

   (b) Kan de occurrence-heuristiek ook leiden tot een verschuiving over nul posities?

   (c) Bereken de verschuivingstabel voor de match-heuristiek van het algoritme van Boyer-Moore voor het patroon $P = \texttt{antenna}$. Gebruik het algoritme voor het zoeken naar het patroon P in de tekst $T = \texttt{anterrantennar}$. Hoeveel vergelijkingen maakt het algoritme?

NOG NIET OMDRAAIEN !