Määrittelydokumentti

Aiheena on tiedon tiivistys alustavasti Huffman-koodauksella. Tämä kylläkin vaikuttaa yksinkertaiselta, joten ehkä jonkin LZ-algoritminkin voisi toteuttaa. Ohjelma tulee saamaan syötteenä tiivistettävän tai purettavan tiedoston sekä vivun, joka kertoo, kumpi on kyseessä. Ohjelman tarkoitus olisi muuntaa syöte tiiviimpään muotoon (ainakin useimmissa tapauksissa) niin, että alkuperäisen tiedoston kuitenkin saa taas tästä palautettua.

Tärkein tietorakenne tässä on Huffman-puu. Siihen liittyvät puun konstruointialgoritmi sekä algoritmit, joilla puun avulla koodataan ja puretaan merkkejä. Puun konstruoinnissa käytetään prioriteettijonoa, joka ja johon liittyvät algoritmit pitäisi myös toteuttaa.

Alustava O-analyysi:

1. Syötteenä on jono merkkejä, esimerkiksi 8 bitin tavuja. Merkkien lukumäärä (siis aakkoston koko) on kiinteä.
2. Ensin syötteestä lasketaan kunkin merkin esiintymislukumäärät. Koska merkkien lukumäärä on ennalta annettu, niin tilavaativuudeksi tulee $O(1)$; paitsi jos sallitaan oikeasti rajattoman kokoiset syötteet, jolloin se on kai ennemminkin luokkaa $O(\log n)$. Aikavaativuus on suunnilleen $O(n)$.
3. Tämän jälkeen muodostetaan esiintymislukumääristä Huffmanin puu. Puu ja sen luomisprosessi riippuvat vain esiintymislukumäärätaulukon alkioiden keskinäisestä järjestyksestä, joten kaikki on vakioaikaista ja -tilaista.
4. Lopuksi pakkauksessa ensin tulosteeseen kirjoitetaan esiintymismäärätaulukko. Vakioaika. Sitten jokaiselle syötteen merkille muodostetaan Huffman-koodi (vakioajassa) ja se kirjoitetaan tulosteeseen. Pakkaus on ajaltaan $O(n)$ ja tilaltaan $O(1)$.
5. Purkuvaiheessa ensin luetaan esiintymismäärätaulukko ($O(1)$), jonka jälkeen muodostetaan siitä Huffman-puu ($O(1)$) ja sitten puretaan koodi. Koska myös koodimerkkien lukumäärä on vakiolla ylhäältä rajoitettu, niin yhden merkin purkamiseen menee vakioaika ja siten koko purkuprosessi on $O(n)$.

Lähteet

Wikipedia (http://en.wikipedia.org/wiki/Huffman_coding)