Aineopintojen harjoitustyö: Tietorakenteet ja algoritmit, alkukesä 2015

Aihe: Tiedon tiivistys

Määrittelydokumentti

Toteutan harjoitustyössäni algoritmejä, joilla voidaan tiivistää eli pakata tietoa häviöttömästi. Pakkaamisella tarkoitetaan sitä, että ohjelma muokkaa syötteenä saamaansa tietoa siten, että se voidaan tallentaa alkuperäistä pienempään tilaan. Pakkamiselle käänteinen operaatio on pakatun tiedon purkaminen, joka tarkoittaa pakatun tiedon muuntamista takaisin alkuperäiseen muotoonsa. Häviöttömällä pakkauksella tarkoitetaan sitä, että kun tieto ensin pakataan ja sen jälkeen puretaan, on näin saatu tieto aina täsmälleen sama kuin alkuperäinen tieto.

Toteutettavat algoritmit ovat Huffmanin koodaus ja Lempel-Ziv –koodaus. Nämä algoritmit on valittu, koska ne edustavat kahta klassista lähestymistapaa tiedon pakkaamiseen.

Huffmanin koodaus käsittelee syötettä määrätyn kokoisissa yksiköissä, yleensä tavu kerrallaan. Menetelmän ideana on korvata usein esiintyvät tavut lyhyillä ja harvemmin esiintyvät tavut pidemmillä binääriesityksillä. Huffman-koodaus antaa siten kiinteän pituisille yksiköille vaihtelevan pituisen esitysmuodon.

Lempel-Ziv –koodaus käsittelee syötettä vaihtelevan kokoisina blokkeina. Syötettä läpikäytäessä muodostetaan ”sanakirjaa”, joka sisältää ne vaihtelevan mittaiset blokit, jotka ovat esiintyneet syötteessä. Kun vastaan tulee blokki, joka on jo sanakirjassa, se korvataan kyseisen blokkiin viittavalla sanankirjan indeksillä. Kun vastaan tulee blokki, jota ei vielä ole sanakirjassa, se lisätään sanakirjan loppuun, jolloin se saa indeksin, jolla se korvataan pakatussa tiedostossa. Tässä yhteydessä on ratkaistava kysymys, kuinka suurta sanakirjaa halutaan käyttää. Algoritmien tehokkuutta analysoitaessa on tarkoitus tutkia erilaisten sanakirjan kokojen vaikutusta algoritmin pakkaus- ja ajalliseen tehokkuuteen.

Ohjelmalle annetaan syötteenä käytettävän algoritmin tunnus ja tiedostonimi, jolle ohjelma suorittaa pakkausalgoritmin. Ohjelma lukee aluksi tiedoston muistiin, minkä jälkeen se pakkaa sen sisältämän tiedon, ja ilmoittaa pakkaamiseen kuluneen ajan. Tämän jälkeen ohjelma purkaa pakatun tiedon ja kertoo purkamiseen kuluneen ajan. Viimeiseksi ohjelma vertaa purettua tietoa syötteenä saatuun alkuperäiseen tietoon ja ilmoittaa, ovatko ne yhtenevät.

Tiedon pakkauksen osalta tavoitteena olevien aika- ja tilavaatimusten määrittäminen on ongelmallista, sillä algoritmin käyttökelpoisuuteen vaikuttaa aika- ja tilavaatimusten optimaalisuuden lisäksi se, kuinka tehokkaasti se kykenee pakkaamaan syötteenä annetun tiedon. Lisäksi pakkaustehokkuus riippuu merkittävästi syötteen ominaisuuksista, joten myöskään pakkauksentehokkuudelle ei voida asettaa yleisiä kriteereitä.

Huffmanin koodauksen perustoteutuksen aikavaativuus on O(n), sillä ensiksi lasketaan jokaisen tavun esiintymistiheys (aikavaatimus O(n)), sen jälkeen muodostetaan Huffman-puu esiintymistiheyksien perusteella (tämä onnistuu vakioajassa, sillä solmuja on korkeintaan 256). Lopuksi syötteen jokainen merkki korvataan sen koodiesityksellä, ja tämän aikavaatimus on myös O(n).

Lempel-Ziv –koodauksen perustoteutuksen aikavaatimus on myös O(n), sillä jokainen syötteen merkki luetaan ainoastaan kerran, ja jokaisen merkin suhteen suoritettavat operaatiot (haku sanakirjasta, sanakirjaan lisäys, jne.) ovat vakioaikaisia, eli ne eivät riipu syötteen pituudesta, kun sanakirjan koko on vakio.

Lähteet:

http://en.wikipedia.org/wiki/Data_compression

http://en.wikipedia.org/wiki/Huffman_coding

http://www.data-compression.com/lossless.shtml