Tietorakenteet ja algoritmit -harjoitustyö, kevät 2018

Sami Kukkonen

20. tammikuuta 2018

1 Johdanto

Kurssin tavoite on toteuttaa komentorivityökalu, joka pakkaa tai purkaa syötteenä annetun binääritiedoston Lempel-Ziv-Welchin algoritmia (LZW) käyttäen. LZW perustuu syötteessä esiintyvien osabittijonojen indeksoimiseen taulurakennetta käyttäen, korvaten osajonon yksilöllisellä n-bittisellä (yleensä $n \leq 12$) arvolla syötteessä. Algoritmi siis pakkaa tehokkaasti syötteitä, jossa tietyt osajonot toistuvat useasti. [1][2] Projekti toteutetaan Scala-kielellä [3].

2 Järjestelmän käyttämät tietorakenteet

Projektin tarkoituksena on opetella tietorakenteiden ja algoritmien tekemistä, joten projektin käyttämät tietorakenteet rakennetaan itse kurssin aikana. Olkoon joukko S syötteessä käytettävä aakkosto ja S^* kaikkien S:stä koostuvien merkkijonojen joukko. Joukko $\mathcal{P}(s)$ sisältää kaikki s:n mahdolliset osamerkkijonot ja $f : \mathcal{P}(s) \to I$ on jokin bijektiivinen kuvaus indeksijoukkoon $I \subseteq \mathbb{N}$. Tällöin kuvaus $LZW : S^* \to I^*$ on pakkaus merkkijonosta s merkkijonoon s'. LZW:n toimintaperiaate on siis käydä syötettä s läpi kirjain kirjaimelta (esimerkiksi ASCII-merkkijonojen tapauksessa 8 bittiä vastaa yhtä kirjainta) ja muodostaa pareja $(\hat{s}, i), \hat{s} \in \mathcal{P}(s)$ vastaan tuleville osajonoille, joita ei ole vielä tullut vastaan, ja lisätä palautettavaan merkkijonoon s' arvo i. [2]

Koska merkkijonoon s' ei haluta lisättävän koko joukkoa $S \times I$ purkamisen mahdollistamiseksi, muodostetaan I:n arvot systemaattisesti siten, että kaikille $x \in S$ annetaan nollasta alkavat indeksit $(0, \ldots, |S| - 1)$ nousevassa järjestyksessä. Tämän jälkeen vastaan tuleville osamerkkijonoille annetaan aina pienin mahdollinen luku, jota ei ole annettu jo aikaisemmin jollekkin toiselle osamerkkijonolle. [2] Tarvitsemme parien tallentamiseen siis tietorakenteen, joka täyttää seuraavat ehdot:

- mahdollisuus tallettaa pareja $(\hat{s}, f(\hat{s}))$ tietorakenteeseen, missä $\hat{s} \in \mathcal{P}(s)$; alkioiden määrä on $O(\mathcal{P}(s)) = O(|S^n|)$.
- mahdollisuus hakea pari $(\hat{s}, f(\hat{s}))$ $\hat{s}$:n perusteella.

Tietorakenteita, jotka täyttävät nämä ehdot, on useita, selvinä esimerkkeinä hajautustaulu ja binäärihakupuu. Huomataan kuitenkin, että mille tahansa tietorakenteessa olevalle alkiolle $\hat{s} \in \mathcal{P}(s)$ pätee, että tietorakenteessa on myös kaikki joukon $\mathcal{P}(\hat{s})$ alkiot, eli siis kaikki merkkijonon $\hat{s}$ osamerkkijonot. Tällöin mielenkiintoinen valinta tietorakenteeksi on *trie* joka takaa ajassa $O(k)$ toimivat lisäys- ja hakuoperaatiot merkkijonon pituuden k suhteen. [4][1]

Trieillä ja hajautustauluilla huomataan olevan sama aikavaativuus, jos hajautusfunktio käy koko merkkijonon läpi. Trie on todennäköisesti helpompi toteuttaa kuin hajautustaulu, sillä uudelleenhajautusta ja hajautusfunktion valintaa ei tarvitse tehdä. Tarkoitus on siis käyttää trietä LZW:n toteutukseen ja selvittää samalla sen edut ja haitat hajautustauluun ja binäärihakupuuhun nähden. Trien apurakenteena tarvitaan todennäköisesti listarakennetoteutus. Pyrin toteuttamaan molemmat rakenteet funktionaalisesti siten, että rakenteen operaatiot palauttavat uuden olion olemassaolevan rakenteen muuttamisen sijaan.

2.1 Tavoitteena olevat aika- ja tilavaativuudet

Tavoitteena on toteuttaa ajassa $O(n)$ toimiva pakkaus- ja purkualgoritmi syötteen koon n suhteen. [1] Tavoitetilavaativuus molemmille algoritmeille on $O(s)$.

3 Ohjelman saamat syötteet

Ohjelman käyttöliittymä toteutetaan komentorivikäyttöliittymänä. Ohjelma saa komentorivivipuna tiedon siitä, pakataanko vai puretaanko syötetiedosto. Ohjelma ottaa syötteenä tiedoston nimen ja tulostaa pakatun tai puretun tiedoston sisällön standard output -virtaan.

Viitteet

[1] Gonzalo Navarro. “Indexing text using the Ziv-Lempel trie”. *J. Discrete Algorithms* 2 (2002), s. 87–114.

[2] T. A. Welch. “A Technique for High-Performance Data Compression”. *Computer* 17.6 (kesäkuu 1984), s. 8–19. ISSN: 0018-9162. DOI: 10.1109/MC.1984.1659158.

[3] Martin Odersky, Philippe Altherr, Vincent Cremet et al. “An Overview of the Scala Programming Language Second Edition”. Teoksessa: 2006.

[4] Donald E. Knuth. “The Art of Computer Programming, Volume III: Sorting and Searching”. Teoksessa: 1973.