Tietorakenteet ja algoritmit –harjoitustyö, C-toteutustapa
Mikko Rouru

1. Ongelman ratkaisu

Harjoitustyössä sai valita kahdesta vaihtoehdosta, C- tai Python-toteutuksen väliltä. Valitsin C:n, koska kieli on paremmin hallussa ja olen käyttänyt Pythonia varsin vähän. Molemmat työt vaikuttivat vaikeilta, erityisesti kun ohjelmointitaustaa ei ole hirmuisesti vielä kertynyt.

Ensi alkuun sai ruveta kertaamaan hash-taulujen toimintaa. Kurssilla ne eivät olleet kovin vaikea asia käsittää, mutta ohjelmointiin niiden yhdisteleminen tuntui työläältä. Paljon sai tehdä esitutkimusta Googlen avulla ennen varsinaisen työn aloitusta. Perusidean ymmärsin nopeasti, mutta vaikeaa oli esimerkiksi minkälaisen hash-funktion pitäisi olla jotta se on toimiva ja riittävän nopea.

Hash-taulut toimivat ohjelmassa yksinkertaisesti näin:

- avataan tiedosto ja pyydetään käyttäjää syöttämään tiedoston nimi
- luetaan tiedostosta kaikki sanat
- poistetaan ylimääräiset merkit sanoista
- luodaan hash-taulu jonka koko on sanojen lukumäärä
- sanat menevät hash-funktion läpi missä modulo otetaan hash-taulun koon mukaan
- etsitään sanoille funktioiden avulla paikka taulusta, jos varattu, mennään seuraavaan paikkaan
- tulostetaan quicksortin avulla lajiteltu sadan yleisimmän sanan listaus
- tuhotaan hash-taulu

Työ oli haastava ja tein sitä pieni pala kerrallaan. Kuten aiemmin mainitsin, paljon aikaa kului ohjelman osa-alueiden tutkimiseen ennen kuin varsinaista koodia sai edes aikaseksi. Välillä työ eteni huomattavasti, mutta välillä tietyt osat aiheuttivat pitkiäkin pohdiskelutaukoja työssä.

2. Suorituskyvyn analyysi

Mittasin algoritmin suorituskykyä kurssin toisen harjoituksen ajanmittaustyökaluilla (time.h-kirjasto). Tulokset olivat seuraavanlaiset:

Tiedosto	Sanojen lkm	Aika (s)	Aika (s, pyöristettynä)
small.txt	2616	0.006937	0.007
TheGun.txt	7614	0.014550	0.015
MetaMorph.txt	23843	0.031944	0.030
Species.txt	192875	0.260015	0.260
Ulysses.txt	245041	0.324657	0.320
WarPeace.txt	514911	0.699613	0.700

Kuulostaisi äkkiseltään että puolen miljoonan sanan käsittelyyn voisi kuluttaa vaikka kuinka paljon aikaa, mutta algoritmi suoriutuu tehtävästä alle sekunnissa. Algoritmi siis on

nopea ja sillä voisi käsitellä hyvinkin suuria sanamääriä ilman suurempia vaikeuksia. Algoritmi noudattaa myös melko lineaarista kasvua, esimerkiksi Ulyssesista WarPeaceen sanojen määrä tuplaantuu, samoin suoritusaika likimain. Jos siis miljoonan sanan kanssa menisi noin 1.4 sekuntia, minuutissa voitaisiin käsitellä noin 43 miljoonan sanan tiedosto. Tunnissa määrä kasvaisi jopa miltei 2.6 miljardiin sanaan.