Määrittelydokumentti

Mitä algoritmeja ja tietorakenteita toteutat työssäsi

Käytän aluksi seuraavia collection-luokkia, jota lopulta toteutan itse.

- HashMap (viite ja arvoparit)
- Map.Entry (HahsMapien for-each-käyntiin)
- ArrayList (listauksiin)

Käytän myös seuraavia Javan valmiita metodeita.

- Math.hypot (etäisyyden laskeminen)
- Math.abs (itseisarvo)
- Math.random (satunnaislukujen tuottaminen)

Yleisesti luon oman algoritmin, joka satunnaisuutta hyväksikäyttäen luo uuden järjestyksen. Tämän järjestysen tuottama pistemäärä lasketaan, ja jos parempi kuin edellinen, niin tallennetaan.

Mitä ongelmaa ratkaiset ja miksi valitsit kyseiset algoritmit/tietorakenteet

Ongelma on sitsien plaseeraus ongelma, jossa esimerkiksi 80 henkilön sitseillä erilaisia järjestyksiä on olemassa 80:n kertoman verran. Tiedämme myös jo melko hyvin jokaisen sitsaajan mielipiteet muista sitsaajista, tai ilmoittautuessa he voivat nämä ilmoittaa.

Jos jokainen sitsaaja kertoo esimerkiksi kymmenen mielipidettä, tulee mielipiteitä yhteensä sitsaajien määrä kertaa kymmenen (esim. 80 x 10 = 800).

Ohjelma yrittää siis optimoida saamiensa tietojen perusteella parhaan järjestyksen, joka on mahdollista luoda.

Satunnaisuus sopii tehtävään melko hyvin, sillä näin pääsemme jo erittäin lähelle optimaalista järjestystä jo melko lyhyessä ajassa. Täydellisen järjestyksen löytyminen lienee ajallisesti mahdotonta jos melko pienillä juhlien osallistujamäärillä.

Mitä syötteitä ohjelma saa ja miten näitä käytetään

Ohjelmalle voi itse syöttää (lopulta) erillissä asetustiedostossa sitsaajien tiedot ja heidän antamat mieltymykset muista.

Testausta varten ohjelma voi kuitenkin luo itse satunnaiset nimet ja yhteydet, joita ohjelma lähtee optimoimaan.

Lopulta ei tarvitse muuta, kuin antaa asetustiedosto, ja ajaa ohjelmaa kunnes tuntee olevansa tyytyväinen, jolloin ohjelman sulkeutuessa tallentaa tiedostoon optimoimansa järjestyksen.

Tavoitteena olevat aika- ja tilavaativuudet (m.m. O-analyysi)

Tilavaativuus ei ole tule olemaan ongelma, sillä etsintäpuun edellisiä solmuja ei tallenneta viimeistä lukuunottamatta. Parasta löytynyttä solmua muokataan

yhden vaihdon ja osallistujien määrän verran, jolloin pyritään ensisijassa löytämään lähellä olevia parempia solmuja, mutta annetaan mahdollisuus myös täysin erilaisiin järjestyksiin toivon perässä.

Aika vaativuus on nyt se ongelma. Emme pysty käymään kaikkia mahdollisia järjestyksiä läpi ja kokeilemaan niitä pistemääriä, sillä esimerkiksi 80! on aivan jäätävän kokoinen luku.

Melko lähelle optimaalista pääsemme jo tietokoneen tehoista ja sitsaajien määrästä riippuen satunnaisuudella lyhyessäkin ajassa.