TiRaLabra, testausdokumentti

Arto Kaikkonen

Algoritmit

Testasin algoritmien toimintaa jonkin verran JUnit-testeillä. Vertasin algoritmien toimintaa toisiinsa muutamilla satunnaisesti generoiduilla verkoilla sekä varmistin, että ne palauttavat Hamiltonin piirin. Kävin myös läpi dataa algoritmien vertailuvaiheessa ja varmistin, että tulokset ovat loogisia.

Tein ongelman visualisointia ja testausta varten pienen koodinpätkän, joka arpoo verkon, ja antaa sen syötteenä eri kauppamatkustajan ongelmaa ratkoville algoritmeille. Ohjelma tulostaa näytölle verkon pisteet ja piirtää lyhimmän reitin viivoina pisteiden välille. Ainakin osa algoritmeista palauttaa virheilmoituksen kun verkon koko on 2 tai alle, mutta en koe sitä tässä kohtaa ongelmana.

Solmujen määrä on koodattu sisään ohjelmaan, mutta sitä on helppo muuttaa. Samoin käytettäviä algoritmeja on helppo vaihtaa tarvittaessa muuttamalla paria kohtaa koodista.

Ajoin ohjelman lukuisia kertoja eri kokoisilla verkoilla (5-1000 solmua) ja päättelin algoritmien toiminnasta seuraavaa:

Brute force:

Brute force toimii järkevässä ajassa kun solmuja alle 15. Ajoin brute forcen kymmeniä kertoja muiden eri algoritmien kanssa ja tuloksena saatu reitti oli aina yhtä pitkä tai lyhyempi kuin toisella algoritmilla saatu tulos. Reitti kävi myös joka kerta jokaisessa solmussa, joten brute forcen pitäisi toimia oikein

Branch-and-bound:

Ensimmäinen versio branch-and-boundista tuotti niinikään aina hamiltonin kierroksen. Kierros oli aina sama kuin brute forcella, joten oletettavasti reitti oli aina myös lyhin. Branch-and-bound toimi hieman nopeammin kuin brute force, mutta käytännössä sillä sai laskettua vain 1-2 solmua isompia verkkoja kuin brute forcella. Tulokset olivat odotettuja, joten olettaisin koodin toimivan kuten pitääkin.

Aina lähimpään:

Aina lähimmän käymättömän naapurin valitseva algoritmi tuotti aina Hamiltonin kierroksen. Pienemmillä syötteillä kierros oli usein sama kuin tarkan vastauksen antavilla algoritmeilla, mutta isommilla syötteillä (15-30) tuli jo eroja. Algoritmi toimi vielä yli tuhannen kokoisilla syötteillä kohtuullisen nopeasti, ja näyttäisi testien perusteella toimivan oikein.

Karp-Held –heuristiikka:

Karp-Held toimii kuten sen kuuluukin, eli se joko löytää lyhimmän reitin tai tietää, ettei lyhintä reittiä löytynyt. Karp-Heldin löytämä reitti oli aina sama kuin branch-and-boundin tai brute forcen, tosin näitä pystyttiin testaamaan vain noin 15 solmun kokoisilla verkoilla. 15-30 solmun kokoisilla verkoilla Karp-Held löysi usein lyhyemmän reitin kuin aina lähimmän naapurin valitseva algoritmi. Tätä suuremmilla verkoilla algoritmi osasi kyllä arvioida lyhimmän reitin pituuden, mutta arvion oikeellisuutta on hankala empiirisesti arvioida, koska lyhin reitti ei ole tiedossa. Joka tapauksessa Karp-Held ei kertaakaan palauttanut pidempää reittiä kuin muut algoritmit, ja toisaalta se palautti aina joko hamiltonin kierroksen tai ei mitään. Näin ollen olettaisin sen toimivan oikein.

Primin algoritmi:

Käytin Primin algoritmin muunnelmaa apuna Karp-Heldin algoritmin toteutuksessa. Primin algoritmi löytää siis pienimmän virittävän puun, ja sen muunnos pienimmän virittävän l-puun. Testasin l-puun löytävää metodia samalla metodilla kuin muita algoritmeja, eli tulostin sen tuottaman virittävän l-puun näytölle joitakin kymmeniä kertoja ja varmistin, että tulos oli aina virittävä l-puu, joka ainakin näytti pienimmältä mahdolliselta.

Tietorakenteet

Kaikkia käyttämiäni tietorakenteita olen testannut JUnit-testeillä melko kattavasti.