Tietorakenteiden harjoitustyö – Toteutusdokumentti

Harjoitustyönä on toteutettu Määrittelydokumentin mukainen ohjelma, joka selvittää annetusta maantieteellisten paikkojen muodostamasta verkosta kahden paikan välisen lyhimmän reitin. Polunetsintä suoritetaan sekä A*- että Dijkstra-algoritmeillä, sillä tarkoituksena on myös vertailla näiden algoritmien suoritusaikoja.

Ohjelman rakenne

Ohjelma koostuu 16 luokasta, kolmesta rajapinnasta ja kahdesta enumista. Luokat on jaettu paketteihin vastuualueidensa perusteella.

Käyttöliittymään liittyvät luokat (UserInputHandler, joka käsittelee käyttäjän antamat syötteet, sekä Messenger, joka vastaa käyttäjille näytettävien viestien ja kehotteiden esittämisestä) ovat paketissa *tiralabra.ui*.

Ohjelma lukee polunetsinnässä käytettävän verkon tiedot sisään määrämuotoisesta tiedostosta. Lisäksi ohjelma voi suorittaa määrämuotoisessa skriptitiedostossa listatut polunetsintäkomennot. Näiden tiedostojen lukemisessa ja tulkitsemisessa tarvittavat luokat ovat paketissa *tiralabra.datainput*. Tiedostojen käsittelystä vastaa DataFileHandler-luokka ja luetun informaation tulkitsemisesta olioiksi vastaavat PlaceGraphMapper- ja ScriptMapper-luokat. Ensiksi mainittu tulkitsee verkkotiedoston tietoja ja jälkimmäinen skriptitiedostoa. Paketissa on määritelty lisäksi kaksi rajapintaa, IDataMapper sekä IGraphMapper; näistä IGraphMapper perii IDataMapperin. IDataMapper määrittelee muutaman metodin, joita ylipäänsä tiedostoja tulkitsevan olion on toteutettava ja IGraphMapper lisäksi yhden metodin, jota verkkotiedostoa tulkitsevalta oliolta odotetaan.

Varsinainen polunetsintälogiikka sisältyy *tiralabra.search* –paketin luokkiin. PathSearcher-luokka vastaa algoritmien ajosta ja PathAlgorithm-luokka etsintäalgoritmin toteuttamisesta. PathAlgorithm sisältää sekä A*- että Dijkstra-algoritmin mukaiset polunetsintälogiikat.

Ohjelman tunnistamia tietokohteita vastaavat luokat ovat paketissa *tiralabra.domain*. Yksittäistä maantieteellistä paikkaa kuvaa PlaceNode; tällaisen naapureita kuvaa puolestaan NeighbourNode. Yhden algoritmin tuottamien polunetsintätulosten tallentamiseen käytetään PathSearchResult-olioita; kaikkien (kahden) algoritmin tulokset yhdestä polunetsintäpyynnöstä tallennetaan PathSearchResultSet-olioon. Skriptitiedoston sisältämistä komentoriveistä puolestaan muodostetaan Command-olioita. Paketissa määritellään myös INamedObject-rajapinta, jota toteuttavan luokan oliot voidaan tallentaa NamedArrayList-tietorakenteeseen. Tätä rajapintaa toteuttavat kaikki domain-luokat NeighbourNodea lukuun ottamatta.

Ohjelman hyödyntämät tietorakenteet, MinHeap (minimikeon toteutus), PathStack (paikoista koostuvan polun tallentamiseen sopiva pinorakenne) sekä NamedArrayList (dynaamisesti kasvava taulukko olioille, joita voi etsiä nimellä) ovat paketissa *tiralabra.datastructures*.

Aika- ja tilavaativuudet

Algoritmien suorituskykyvertailu

Ohjelman jatkokehitys

Lähteet

Tietorakenteet ja algoritmit –kurssin luentokalvot (katsottu 31.7.2016)
https://www.cs.helsinki.fi/u/jkivinen/opetus/tira/k16/luennot.pdf

Rosetta Code –sivustolla esitetty Haversine-kaavan toteutustapa Javalla
https://rosettacode.org/wiki/Haversine_formula#Java