31.5.

Toteutin aluksi dijkstran-algoritmin käyttäen javan valmista priorityqueue:ta. Toteutin tänään oman keon ja korvaan priorityqueuen sillä lähipäivinä. Dijkstran-algoritmi palauttaa tällä hetkellä vain lyhimmän määrän askelia, millä lähdöstä pääsee maaliin. Tämä on kuitenkin helppo muokata sellaiseksi, että metodi palauttaa listan solmuista, joita pitkin lyhin reitti kulkee. Näin löydetyn reitin voi vaikka värittää labyrinttiin, jolloin ohjelman mielekkyys kasvaa.

6.6.-7.6.

Viimeistelin keon toiminnan ja varmistin aikavaativuudet. Toteutin keon sisäiseksi rakenteeksi oman luokan joka pitää sisällään tuplaavan taulukon. Tein bittikartta-luokan joka muodostaa char-labyrintteja paintilla piirretyistä kuvista. En ole alkanut vielä tekemään käyttöliittymää, vaan koko toteutuksen toimivuus on varmistettu ainoastaan JUnit testeillä.

Viimeinen viikko

Sain maanantaina valmiiksi käyttöliittymän ja parantelin sitä vielä tiistaina. Tiistaina lisäsin mahdollisuuden piirtää soita labyrinttiin, jolloin dijkstrasta on oikeasti jotain hyötyä ja työ eroaisi oleellisesti leveyssuuntaisesta verkon läpikäynnistä. Lisäsin myös mahdollisuuden käyttää “euklidista”-metriikkaa manhattan-metriikan sijasta. Näistä voi valita kumman tahansa. Euklidinen eroaa vain sillä tavalla, että siinä koordinaatilla on kahdeksan naapuria neljän sijasta. Diagonaalittain liikkumisen painoarvo on 1.42, joka on noin neliöjuuri 2. Toteutin myös metodin joka tallentaa parhaan reitin omaan arraylistiin. Tässä käytettiin ensin java.utilin arraylistia ja hashmappia, mutta korvasin arraylistin omalla tuplaavalla taulukolla ja loin maanantaina oman hashmapin (niiltä osin kuin sitä tarvitsee tässä projektissa).

Hashmap käyttää koordinaattien maksimimäärää hyödyksi, tämä tekniikka mainitaan lähteen sivulla 286 (Matti Luukkainen tira 2011). Toteutin hashmapin yksinkertaisella lineaarisella avoimella hajautuksella. Hashmap ei olisi ollut välttämätön ratkaisu, vaan parhaan reitin olisi voinut tallentaa myös ilman sitä. Tulipahan tehtyä harjoituksen vuoksi.

Oville ja avaimilla on lisätty värit ja koodaukset koordinaatteihin ja bittikarttaan, mutta niiden toimintaa ei ole vielä lisätty. Dijkstra ei siis osaa mennä vielä ovista läpi. Työ jää tähän.

Analyysit

Ohjelmaan liittyvien Dijkstran-algoritmin, keon, tuplaavan taulukon ja hashmapin aikavaativuudet ovat samat kuin materiaalissa (Matti Luukkainen, tira 2011). Toteutin omat tietorakenteen prujun pohjalta, joten pseudokoodin aikavaativuudet ovat samat kuin materiaalissa. Omien toteutusteni aikavaativuudet varmistin vielä JUnit testeillä

Puutteet ja parannusehdotukset

Olen pyrkinyt toteuttamaan työn helposti laajennettavaksi. Työhön voisi lisätä esimerkiksi erilaisia labyrintin läpikäyntialgoritmeja helposti. Myös alkuperäinen idea avaimista ja ovista on toteutuskelpoinen, vaikka jäikin tekemättä.

Itse valmis työ poikkeaa suunnitellusta. Se mahdollistaa omien labyrinttien piirtämisen piirto-ohjelmalla ja niiden ratkaisemisen Dijkstran-algoritmilla. Labyrintin läpikäyntiä voi simuloida näytöllä.

Lähteet:
Matti Luukkainen, tietorakenteet 2011: http://www.cs.helsinki.fi/group/java/k11/tira/tira2011.pdf