Toteutusdokumentti

Ohjelman yleisrakenne

Ohjelma on jaettu selvästi viiteen eri pakkaukseen. Pakkaukset ovat nimetty niiden toiminnallisuuksien mukaan (algorithms, datastructures, domain, gui ja logic). Algorithms pakkauksesta löytyy erilliset luokat Dijkstran ja A Starin algoritmeille. Algoritmit toimivat itse ohjelmoimillani tietorakenteilla jotka löytyvät pakkauksesta datastructures. Tietorakenteita ovat lista (List), minimikeko (MinHeap) sekä pino (Stack).

Loin ohjelmaa varten myös luokat Maze sekä Pixel, ja nämä luokat löytyvät pakkauksesta domain. Pixel luokka on tehty sitä varten, että kun ohjelma lukee alussa annettua labyrinttikuvaa, luodaan jokainen kuvan pikseli omaksi PIxel olioksi jolle asetetaan naapurit, onko pikseli seinä vai käytävä yms. Maze luokan tein nopeuttaakseni ohjelman toimintaa, koska sinne asetetaan kaksiulotteinen taulukko joka sisältää pikselit sekä erikseen maali- sekä aloituspikselin sijainnit.

Gui pakkaus on pieni lisäys alkuperäisen ohjelman toimintaan, päätin luoda ulkoasun ohjelmaa varten, jotta ohjelman käyttö olisi selkeämpää ja ohjelma olisi ”hienompi”. Se sisältää luokat App sekä Solver. App luokassa luodaan ulkoasu, ja Solver luokka taas käsittelee *Solve* napin painalluksen, eli toisinsanoen aloittaa kuvan lukemisen ja ratkaisemisen.

Logic kansio sisältää Main sekä MazeMaker luokat, joista ensimmäinen vain käynnistää graafisen käyttöliittymän. MazeMaker on laajempi luokka, missä mm. luetaan kuva kaksiulotteiseen taulukkoon ja piirretään ratkaistu reitti kuvaan.

Suorituskykyvertailu

Minun on vaikea keksiä miten voisin testata aikavaativuuksia, kuitenkin testausdokumentista käy ilmi A Starin ja Dijkstran erot nopeudessa. Uskoisin saavuttaneeni lähestulkoon oikeat aikavaativuudet jotka määrittelydokumentissa ilmoitin (Dijkstra O(|V|+|E| log |V|) ja A Star O(|V| + |E|)

Työn mahdolliset puutteet ja parannusehdotukset

Kuten labtoolin palautteessa mainittiin, A Star algoritmin painotuksia muuttamalla olisin mahdollisesti voinut saada tehokkaamman ratkaisun A Star algoritmiin. Olen kuitenkin tyytyväinen nykyiseen ratkaisuun, koska nykyinen algoritmi tuottaa ratkaisun nopeammin kuin Dijkstra, joten algoritmi toimii niinkuin pitää.

Työhön olisi voinut lisätä muita algoritmeja, tai näkyville missä pikseleissä algoritmi on käynyt hakiessaan ratkaisua. Tästä saattaisi kuitenkin tulla jonkinnäköinen sekamelska ratkaistuun kuvaan.