Toteutusdokumentti

Yleisrakenne:

Ohjelmaa suorittaessa luodaan ensin menu-ikkuna, johon käyttäjä syöttää haluamansa labyrintin koon. Ohjelma toimii suurillakin syötteillä, mutta ulkoasun ja käytännöllisyyden vuoksi syötteelle on annettu rajaksi [2-40]. Kun annettu koko täyttää ehdon, luodaan solmut (myöh. "maapalat"). Koska *maapalat* luodaan jo suorituksen alkuvaiheessa (ennen haun käynnistymistä), itse reitin haku toteutuu nopeammin.

Graafiseen käyttöliittymään luodaan "nappulat" edustamaan maapaloja, joita hiirellä painamalla asetetaan tiettyihin toiminnallisuuksiin. Nämä "nappulat" on talletettu 2D- taulukkoon, joten niiden haku koordinaattien perusteella toteutuu vakioajassa. Hiiren vasemmalla painikkeella painaessa *maapala* muuttuu joko seinäksi tai läpäistäväksi riippuen siitä, missä tilassa solmu on ennen painallusta. Hiiren oikealla painikkeella painaessa *maapala* muuttuu alku- tai loppupisteeksi riippuen sen tilasta ennen painallusta.

Kun yksi alku- ja loppupiste on alustettu, voidaan lyhimmän reitin etsintä käynnistää. Etsintä on toteutettu A*- algoritmilla käyttäen kahta linkitettyä listaa "avoinLista" ja "suljettuLista." Algoritmi käy avoimen listan läpi ja etsii sieltä *maapalan*, jolla on pienin kokonaisarvo (kokonaisarvo = heuristinen arvo + liikkumiskustannus(10) + vanhemman kokonaisarvo). Kun haettu *maapala* on löydetty, asetetaan sen naapureille oikeat kokonaisarvot, tieto vanhemmasta (parent- solmu) ja siirretään ne avoimelle listalle. Kun kaikki naapurit (ei diagonaaliset) on asetettu avoimelle listalle, siirretään jo käsitelty yksilö "suljetulle listalle."

Edellä mainittua tapahtumasarjaa suoritetaan niin kauan, kunnes avoin lista on tyhjä eli uusia solmuja ei ole käsiteltävänä tai kun loppupiste on saavutettu. Jos loppupiste on löytynyt saadaan lyhin reitti esiin kutsumalla while- loopissa saavutetun pisteen vanhempaa, kunnes vanhemmuus on arvossa null eli on saavutettu alkupiste.

Seuraavaksi avaan tuottamani algoritmin tehokkuus- ja tilavaatimuksien toteutumista ohjelman logiikan kannalta. Tämä kattaa labyrintin alustuksen sekä itse reitin etsimisen.

Ohjelma luo maapalat ajassa O(n), sillä jokainen *maapala* täytyy käydä lävitse luontiprosessissa.

```
for i = 0 : i < koko; i++;
        for j = 0: j < koko; j++;
                labyrintti[j][i] = new Maapala(j, i);
```

Näin ollen vaadittu aika on riippuvainen syötteen eli labyrintin koosta: O(n)
Tilavaativuus on myöskin riippuvainen syötteen koosta, joten sekin on: O(n)
Koodissa oleva muuttuja koko on labyrintin sivun pituus, joten syöte on todellisuudessa

koko x koko, mistä johtuu vaativuus O(n) eikä O(n^2). Jokainen paikka käydään läpi kerran eikä kahta kertaa.

Kun maapalat on luotu, rakennetaan seinät ja asetetaan lähtö- ja loppupisteet.
Nämä operaatiot tapahtuvat vakio ajassa, sillä maapaloilla on omat muuttujat sille, ovatko ne seiniä, vai ei. Tämän lisäksi Maapalarekisterille syötetään tiedot lähdön ja lopun koordinaateista, joten sekin tapahtuu vakio ajassa O(1).

```
maapala.asetaSeinaksi()
        this.seina = true;

maapalarekisteri.asetaAlkuX(int luku)
        this.alkuX = luku;
```

Sijoitus operaatiot ovat vakioaikaisia, joten operaatiot ovat kokonaisuudessaan vakioaikaisia.

Tämän jälkeen *maapaloille* asetetaan heuristiset arvot sen mukaan, miten kaukana ne ovat loppupisteestä x-koordinaatin ja y-koordinaatin suhteen. Jokainen alkio käydään läpi, joten aikavaatimuksena on O(n), kuten ensimmäisessä kohdassa on selitetty. Heuristisen arvon laskeminen sekä sijoitus kyseiselle maapalalle ovat vakioaikaisia operaatioita, joten metodin vaativuudeksi tulee O(n).

Lopuksi itse lyhimmän reitin etsimisessä käytetyn algoritmin aika- ja tilavaativuudet.
Jokainen alkio (maapala) asetetaan enintään kerran avoimelle listalle. Tämän lisäksi jokaista alkiota avoimella listalla tarkistellaan enintään kerran ennen kuin loppupiste olisi löydetty.
Tästä johtuen pahin tapaus, mikä voisi tapahtua vaatisi, että kaikki alkiot käytäisiin läpi ennen lopun löytymistä eli aikavaativuudeksi tulisi O(n).

Näin ollen:
ohjelman aikavaativuus O(n)
ohjelman tilavaativuus O(n)