Määrittelydokumentti

Aihe: Labyrintti seikkailu

Labyrintti koostuu erilaisista “tasoista” ja yksi taso sisältää aina yhden avaimen ja yhden oven, josta pääsee ulos avaimella. Avain on siis löydettävä ennen kuin ovesta pääsee ulos. Jos ovi on löydetty ennen avainta, tallennetaan oven paikka, jotta sitä ei tarvitse etsiä uudestaan.

Labyrinttiin voi lisätä myös yhden dynamiitin, jolla pääsee läpi mistä vain! Reitinselvittäjä yrittää löytää lyhimmän reitin ulos labyrintista, jolloin tulee optimoida milloin dynamiitti käytetään.

Aion käyttää etsintäalgoritmina Dijkstran algoritmia, johon toteutan itse sen vaatiman keko-tietorakenteen. Algoritmi pitää kirjaa siitä, missä ruuduissa on jo käyty ja siitä, onko ovi jo löydetty. Käytän Dijkstraa, sillä haluan löytää varmasti nopeimman reitin.

Luon kirjastoluokan, jolle syötetään parametrina ratkaistava labyrintti (alustavasti taulukkona). Tässä luokassa on itse Dijkstran algoritmin tarvitsemia apumetodeja ja tarvittaessa joitain muitakin labyrintin läpikäynti algoritmeja. Keon toteutan erillisenä luokkana. Aion luoda myös erillisen “koordinaatti”-luokan, joka pitää tietoa painoarvoista ja muista koordinaattiin liittyvistä asioista. Näin voin määritellä helposti keon vaatimat compare-operaatiot.

Dijkstran algoritmin aikavaativuus on yleisesti O((|E| + |V|)*log|V|) ja tilavaativuus O(|V|
). Avainten ja ovien kautta kulkeminen lisännee tätä vain vakiokertoimien verran, joten aikavaativuus ei muutu. Tämä kuitenkin selviää vielä tarkemmin työn edetessä. Dynamiitin lisääminen lisää aikavaativuutta, jos halutaan etsiä absoluuttisesti paras reitti. Tätä ongelmaa en ole vielä ratkaissut, joten en osaa sanoa kuinka paljon aikavaativuus kasvaa.

Lähteet:

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