Määrittelydokumentti

Labyrintingeneroija ja –ratkoja

Tietorakenteet ja algoritmit harjoitustyö

Juri Kuronen

Kesä 2014

Sisältö

1. Johdanto

Labyrintin generoimiseen ja ratkaisemiseen on olemassa lukuisia algoritmeja. Tässä työssä toteutan näistä yleisimpiä ja vertailen niiden suoritusaikoja. Luon myös käyttöliittymän, jonka avulla labyrintin generoiva algoritmi, sekä muita asetuksia, voidaan valita. Tämän jälkeen voidaan valita ratkaisemiseen käytettävä algoritmi, ja käyttäjälle esitetään visuaalisesti algoritmin tekemä ”työ”, eli esimerkiksi mitkä labyrintin soluista tuli tutkittua ratkaisevan reitin löytämiseksi.

Algoritmit toteutetaan tavallisille N x M ristikoille, missä N ja M voidaan valita. Tavallinen ristikko tarkoittaa nelikulmaista ristikkoa, jonka solut ovat neliöitä, joista reitit muihin soluihin jatkuvat joko suoraan tai tekevät kohtisuoran käännöksen. Lähtö- ja maalisolu asetetaan ristikon vastakkaisille sivuille. Lisäksi oletetaan, että mistä tahansa labyrintin pisteestä A on vain ja ainoastaan yksi reitti mihin tahansa pisteeseen B. Tämä tarkoittaa, että labyrintti on syklitön ja yhtenäinen, ja labyrintin kaaret muodostavat virittävän puun. Lisäksi oletetaan, että ratkaisevalla algoritmilla on tieto maalisolun sijainnista.[1]

[1] Nämä ovat erittäin alustavia oletuksia. Työn edetessä oletuksia voitaneen muutella käyttäjän syötteellä.

2. Käytetyt algoritmit ja tietorakenteet

2.1. Labyrintin generointialgoritmit

Käyttäjä syöttää labyrintin koon N x M. Labyrintti alustetaan kaksiulotteisena N x M byte-arraynä, jossa ei ole vielä ollenkaan kaaria. Tämän jälkeen aloitetaan satunnaisesti luomaan polkuja lähtösolusta oletusten mukaisesti niin kauan, kunnes koko labyrintti on käyty läpi.

Rekursiivinen peruuttava haku. Käyttää pinoa ja laittaa aina edellisen solun pinon päällimmäiseksi. Lisäksi käytössä on N x M -kokoinen boolean-array vierailluista soluista.

Liikkuu labyrintissä ahneesti niin kauan, kunnes senhetkisellä solulla ei ole enää vierailemattomia naapureita. Tällöin algoritmi peruuttaa palaamalla pinon päällimmäiseen soluun niin kauan, kunnes löytyy solu, jolla on vierailemattomia naapureita. Jos tällaista solua ei löydy (eli pino on tyhjä) on labyrintti generoitu.

- Tilavaativuus O(|V|). Pahimmassa tapauksessa pinon koko on koko labyrintin koko ja lisäksi boolean-arrayn koko on koko labyrintin koko.
- Aikavaativuus O(|V|). Koko labyrintti käydään läpi, ja jokaisessa solussa vieraillaan keskimäärin 2-3 kertaa. Pinon operaatiot ovat O(1).

Primin algoritmi. Ylläpitää listaa soluista, jotka voidaan seuraavaksi liittää labyrinttiin. Lisäksi on käytössä N x M -kokoinen boolean-array labyrintin osana olevista soluista.

Aluksi merkataan lähtösolu osaksi labyrinttiä ja kaikki lähtösolun viereiset solut listaan soluista, jotka voidaan seuraavaksi liittää labyrinttiin. Tämän jälkeen valitaan satunnaisesti yksi solu listasta, liitetään se satunnaista reittiä kautta labyrinttiin, ja lisätään tämän solun naapurit listaan. Kun lista on tyhjä, on labyrintti generoitu.

- Tilavaativuus O(|V|). Listan koko on korkeintaan noin puolet labyrintin koosta, mutta boolean-arrayn koko on koko labyrintin koko.
- Aikavaativuus O(|V|). Koko labyrintti käydään läpi. Jos listaan varataan tarpeeksi tilaa, listan operaatiot ovat kaikki aina O(1).

Kruskalin algoritmi. Lyhyt kuvaus, O(n):t.

Lisää tulossa.

2.2. Labyrintin ratkoja-algoritmit

Satunnaisalgoritmi. Lyhyt kuvaus, O(n):t.

Oikean käden sääntö. Lyhyt kuvaus, O(n):t.

Syvyyssuuntainen haku. Lyhyt kuvaus, O(n):t.

Leveyssuuntainen haku. Lyhyt kuvaus, O(n):t.

A*-haku. Lyhyt kuvaus, O(n):t.

Lisää tulossa. Varsinkin monimutkaisempia algoritmeja, jotka ratkaisevat labyrintin kuin labyrintin ilman helpottavia oletuksia.