Määrittelydokumentti

Aineopintojen harjoitustyö: Tietorakenteet ja algoritmit

Helsingin yliopisto, Tietojenkäsittelytieteen laitos

Tekijä:
Tomi Virtanen
tomi.virtanen@helsinki.fi
014140105

Määrittelydokumentti

Toteutettavat algoritmit ja tietorakenteet

Työssäni toteutan Bellman-Ford, dijkstran ja A* -algoritmit. Lisäksi Dijkstrassa ja A*:ssa tarvitaan minimi kekoa sekä käytyjen solmujen tallettamiseen A*:ssa, joko hajautustaulua tai linkitettyä listaa. Kyseisiä algoritmeja käytän sokkelosta tai labyrintista pakenemiseen ja vertailen näiden algoritmien tehokkuutta keskenään erikokoisilla syötteillä. Jokainen syötteenä oleva taulukko on myös tarkoitus bruteforce:ta jotta saadaan kontrastia tehottomien ja tehokkaiden ratkaisujen välille.

Algoritmien valinta oli oikeastaan aika yksinkertaista sillä Tietorakenteet kurssilla kiinnostuin Dijkstraa toteuttaessani A*:sta ja päätin tehdä harjoitustyön vertailemalla niitä. Lisäksi päädyin ottamaan Bellman-Fordin hieman erityyppisenä algoritmina mukaan. Dijkstrassa ja A*:ssa käytetyssä keossa ajattelin toteuttaa ainakin lähtökohtaisesti tavallisen perus keon. A*:ssa käytyjen solmujen muistamiseen olen alustavasti ajatellut käyttää linkitettyä listaa. Etsiminen on tällöin toki hitaampaa kuin hajautusrakenteilla, mutta käytyjen solmujen määrän ei suhteessa pitäisi nousta valtavaksi suhteessa koko verkon solmujen määrään, ellei huonoin tapaus satu kohdalle.

Syötteenä sokkeloita

Ohjelma saa syötteinään erikokoisia labyrintteja matriisin muodossa. Aluksi ohjelmia rakennellessa matriisit pysyvät melko pieninä ja myöhemmin niitä kasvatetaan. Matriisit on tarkoitus joko generoida looppaamalla tai kuvien avulla. Matriisit sisältävät lukuja 1 ja 0, joista 1 tarkoittaa seinää ja 0 tarkoittaa käytävää.

Tavoite aika- ja tilavaativuudet

Bellman-Fordin tavoiteaika on O(nm), joka on kaarien ja solmujen tulo ja tavoitetila on O(n). Bellman-Ford käy läpi jokaisen solmun ja aina jokaisen solmun kohdalla jokaisen kaaren. Dijkstran tavoiteaika on O((n+m)log n). Eli taulukosta riippuen joko O(n log n) tai O(m log n). Tätä voitaisiin vielä parantaa käyttämällä fibonacci kekoa. Tilaa Dijkstra vie O(n).

A*:n aikavaativuus riippuu käytetystä heuristiikasta. Muuten A* on melkolailla sama kuin Dijkstra. Jos oletetaan että heuristiikka on järkevästi toteutettu ei se paranna pahimman tapauksen aikavaativuutta, mutta parantaa keskimääräistä. A*:llä on pahimman tapauksen tilavaativuus erittäin huono. Pahimmassa tapauksessa joudutaan tallettamaan muistiin kaikki mahdolliset reitit jolloin tilavaativuudeksi muodostuisi O(n*n).

Minimikeon toteutuksista tarvitaan heap-del-min, heap-insert ja heap-decrease-key toimintoja joiden aikavaativuus on O(log n). Vaativin operaatio on heapify, jota käytetään kaikissa näissä. Sen aikavaativuus on O(log n). Muuten nämä käytetyt operaatiot ovat vakioaikaisia, mutta heapifyn käyttö kasvattaa niiden vaativuuksia.

A*:ssa tarvitaan myös tietorakennetta käytyjen solmujen tallentamiseen. Olen ajatellut käyttää yhteen suuntaan linkitettyä listaa. Operaatioina tässä tarvitaan insert ja search. Insert laittaa alkion suoraan listan perälle ja sen vaativuudet ovat vakioaikaisia. Etsinnässä taas joudutaan käymään koko lista läpi jolloin sen vaativuus on syötteen koosta riippuvainen eli O(n). Aikavaativuus etsinnässä on iteroimalla vakioaikainen.

Lähteet

Patrik Floreen. Tietorakenteet ja algoritmit kurssimoniste.
http://www.cs.helsinki.fi/u/floreen/tira2012/tira.pdf
Patrik Floreen. Tietorakenteet ja algoritmit - verkkoalgoritmien vertailumoniste.
http://www.cs.helsinki.fi/u/floreen/tira2012/lisalehti2.pdf