Tietorakenteiden harjoitustyö
Määrittely

Jukka Koskelin
013865465

Työn aihe: A* -reitinhakualgoritmia käyttävä ohjelma joka muodostaa kaksiulotteisen kartan ja etsii (lyhimmän) reitin kahden annetun pisteen välillä.

Käytetään reitinhakuun Manhattan metodia:

H = 10*(abs(currentX-targetX) + abs(currentY-targetY))

koska se vaikuttaa selkeimmältä ja suurin osa esimerkeistä käyttää sitä.

Toiminnalliset vaatimukset

Algoritmin tulee toimia oikein, eli löytää lyhin (ts. tehokkain) reitti kahden annetun pisteen välillä. Pisteet annetaan muodossa (x,y). **(valmis)**

Algoritmin tulee löytää reitti vaikka lyhimmällä polulla olisi esteitä. **(valmis)**

Mikäli lyhintä reittiä ei ole mahdollista löytää, ohjelma ei saa kaatua hallitsemattomasti. **(valmis)**

Ohjelma osaa muodostaa kaksiulotteisen kartan annetun testidatan tai kuvatiedoston pohjalta. Testidatassa muodostetaan X*Y -kokoinen kaksiulotteinen taulukko johon asetetaan esteitä merkitsemällä karttaan pisteitä joiden läpi ei voida liikkua. Kuvatiedostosta generoitaessa tulkitaan kuvasta vastaavat tiedot ja muodostetaan taulukko samoin periaattein. **(valmis, vaatii lisätestausta)**

Ohjelma käyttää itse toteutettua tietorakennetta (keko), joka korvaa Javan LinkedListin/PriorityQueuen käytön. **(valmis, vaatii lisätestausta)**

Ohjelmaa voidaan ajaa komentoriviltä tai GUI-käyttöliittymältä, ja sille voidaan syöttää omia (tietyt vaatimukset täyttäviä) kuvatiedostoja joista kartta muodostetaan.

Toiminnalliset lisävaatimukset

Vaihtoehtoinen toteutus toisella tietorakenteella keon asemesta ja toteutusten tehokkuuden vertailu.

Muut vaatimukset

Julkiset metodit ja muuttujat kommentoitava JavaDocilla.

Ohjelma kattavasti yksikkötestattu.

Ohjelma kattavasti testattu ja suorituskykytestattu.

Ohjelman suorituskykytestaus suoritetaan sekä Javan PriorityQueue:ta että itse toteutettua keko-tietorakennetta käyttäen, ja tuloksia vertaillaan.

Aikavaativuus: Dijkstran algoritmin aikavaativuus on O((|E| + |V|)log|V), missä E on solmujen ja V kaarien määrä. A*-algoritmin aikavaativuus johtuu etupäässä käytetystä heuristiikasta, olettaen että läpikäymättömien solmujen käsittely on tehokasta. Voitanee siis olettaa että pahimmassa tapauksessa A*:in aikavaativuus vastaa suunnilleen Dijkstran aikavaativuutta, vaikka yleensä se toimii (tai ainakin pitäisi toimia) nopeammin, koska

Käytetyt lähteet

Tietorakenteet -kurssi, kevät 2011. Luentokalvot.

Kurssin aihe -sivulla linkitetty artikkeli “Introduction to A*”.

Erilaiset hakukoneille löydetyt A*:n toteutukseen liittyvät kysymykset, vastaukset ja esimerkit. Varsin usein stackoverflow.com:in kautta löydetyt.