Määrittelydokumentti

Toteutan TIRA-labran työssäni kaksi reitinhakualgoritmia. Reitinhaku tapahtuu suunnatussa painotetussa verkossa, ja algoritmeiksi olen valinnut A*- ja Bellman-Ford-algoritmin. Pääpaino tulee olemaan A*:lla, ja Bellman-Ford toteutetaan vertailun vuoksi. Algoritmit on valittu seuraavin perustein: A* on hyvin laajalti käytetty ja yleisesti hyvänä pidetty reitinhakualgoritmi. A* on kelvollinen (? admissible) reitinhakualgoritmi, eli se löytää aina lyhyimmän reitin, ja se käy läpi vähemmän solmuja kuin mikään muu tunnettu kelvollinen reitinhakualgoritmi.

Reitinhaun voi työssäni ajatella tapahtuvan ruudukossa, jossa jokaisesta ruudusta (laitoja lukuunottamatta) voi liikua kahdeksaan suuntaan – sivuille, ylös-alas tai diagonaalisesti. Ruutujen(=solmujen) välillä on siis kaksisuuntainen yhteys, ja jokaisella ruudulla on paino. Toisinsanoen jokainen yhteen ruutuun *saapuva* yhteys saa saman painon. Bellman-Ford-algoritmin valitsin juuri siitä syystä, että sen aikavaativuus on $O(|V| \cdot |E|)$, jossa V on solmujen lukumäärä ja E on nuolten/lähtevien yhteyksien määrä. Sen pitäisi olla siis kyseisessä tehtävässä melko hidas verrattuna A*:iin, ja haluaisinkin nähdä selkeän eron näiden kahden algoritmin nopeuksissa. A* on aikavaativuudeltaan polynomiaalinen jos: hakuavaruus on puu, on olemassa yksi saavutettavissa oleva maali, ja heuristinen funktio *h* täyttää seuraavan ehdon: $|h(x) - h^*(x)| = O(\log h^*(x))$, missä *h** on optimaalinen heuristiikka – oikea hinta solmusta x maaliin – ja *h* on algoritmin laskema heuristinen arvio. Toisin sanoen: *h*:n virhe ei kasva nopeammin kuin logaritmi *h**:stä.

A* tarvitsee kaksi prioriteettilistaa, jotka toteutan keoilla. Wikipedian mukaan algoritmia voi nopeuttaa augmentoimalla minimikekoja hajautustaululla, jolloin solmun etsiminen (tarkistettaessa onko solmu jo listassa) nopeutuu *O(n)*:stä vakioaikaiseksi. Tavoitteena on, että molemmat algoritmit olisivat tilavaativuudeltaan luokkaa *O(n)*.

Mikäli jostain löytyy valmis grafiikkakirjasto ja labyrinttigeneraattori, käytän niitä reitinhaun visualisoimiseen ja mielenkiintoisempien labyrinttien luomiseen. Mikäli näin ei ole, teen itse labyrinttigeneraattorin. Ohjelmalle voi antaa syötteinä: labyrintin koon (annetaan sivujen pituudet, labyrintti muodoltaan suorakaide), lähtöpisteen, maalin ja solmujen painot. Kaikki muu paitsi labyrintin koko voidaan tarvittaessa generoida.

Lähteet:

*http://en.wikipedia.org/wiki/A*_search_algorithm*
http://en.wikipedia.org/wiki/Bellman%E2%80%93Ford_algorithm
http://en.wikipedia.org/wiki/Directed_graph
http://en.wikipedia.org/wiki/Maze_solving_algorithm