Toteutusdokumentti

Ohjelman rakenne

Ohjelma on rakenteeltaan kohtuullisen yksinkertainen. **Main** luokassa suoritetaan komentorivikyselyjä **CmdLine** luokan kautta, rakennetaan **FileIO** luokan lukeman tiedoston kautta **LabyrinthGeneratorilla** kaksiulotteinen taulukko solmuja (**Vertex**) ja solmuille kaaria (**Edge**), jotka kuvastavat verkkoa jota voidaan käydä läpi algoritmeilla.

Verkon luotua **Main** pistää **Solver** olion ratkaisemaan verkkoa kaikilla toteutetuilla algoritmeilla, jotka on sijoitettu omiin luokkiin (**Dijkstra**, **AStar**, **Bellman-Ford**). **Dijkstra** ja **AStar** käyttävät suorituksessa hyväkseen minimikekoa **MinHeap**, johon voi sijoittaa **KeySortable** rajapinnan toteuttavia olioita (**Vertex**) kapseloivan **HeapNode** olion avulla.

Solver luo tulosteita käyttämällä **FileIO**ta, joka tulostaa sekä konsoliin, että erilliseen *output.txt* tiedostoon.

Suorituskykyvertailu

Tässä perehdytään suorituskykytestien tuloksiin, jotka löytyvät testausdokumentista.

A* selviytyi selvästi parhaiten yleisellä tasolla.

Bellman-Ford algoritmi oli yksinkertaisuudessaan parempi, kun suoritettava labyrintti oli suora tunneli ja reitti kulki sen päästä päähän, mutta leveämmillä labyrinteilla sen suoritusaika nousi suuresti tehden siitä hitaimman algoritmin missään vähänkään isommassa labyrintissa.

Dijkstran molemmat versiot olivat jokaisessa testissä hitaampia kuin A*.

Dijkstran maaliin pysähtyvä versio oli omaa koko labyrintin läpikäyvää versiota hitaampi vain samassa tilanteessa, kuin missä Bellman-Ford oli nopein kaikista. Muissa tilanteissa se oli aina hieman nopeampi pysähtyessään maaliin, kuin käydessään läpi koko labyrinttia, mutta ei silti yltänyt kovin lähelle samoja tuloksia, kuin A*. Labyrinttien ollessa lähes esteettömiä A* oli erityisen paljon nopeampi kuin Dijkstra, mikä kertoo heuristiikan vahvuudesta polunetsinnässä.

Työn puutteet ja parannusehdotukset

Automaattinen labyrintin generointi olisi ollut mukava ominaisuus saada tähän työhön mukaan. Työssä olisi voinut myös yrittää käyttää enemmän tietorakenteita, kuten HashMappeja ja LinkedListejä saadakseen mielenkiintoisemman / haastavamman / selkeämmän algoritmitoteutuksen vaikkei suoranaista tarvetta ollutkaan.

Työn laatu olisi voinut olla parempi, jos olisin saanut kerättyä motivaatiota tehdä projektia järkeviin aikoihin, jotta olisin voinut ajoissa tutkia tiettyjä ongelmallisiksi koituneita asiota, kuten O analyysin suoritusta.

Lähteet

https://www.cs.helsinki.fi/courses/58131/2015/s/k/1

https://en.wikipedia.org/wiki/A*_algorithm

https://en.wikipedia.org/wiki/Dijkstra's_algorithm

https://en.wikipedia.org/wiki/Bellman–Ford_algorithm

http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html