Toteutusdokumentti

Ohjelman yleisrakenne

Ohjelman pohjana on Graph-luokka, joka määrittää solmuista (Vertex) ja niiden välisistä kaarista (Edge) koostuvan verkon. Se käyttää aputietorakenteina mm. hajautustaulua (hashtable) ja linkitettyä listaa (linkedlist). Polunetsintäluokat (Dijkstra, Astar, BFS, BellmanFord) toteuttavat rajapinnan PathFinder, jolloin ne toteuttavat metodin, jolla saa lyhyimmän polun kahden solmun välillä. Graph-luokka käyttää näistä algoritmeista sitä, joka sopii sen hetkiseen verkkoon (painoton, vai vain positiivisia painoja?) ja muita algoritmiluokkia niille tarkoitettujen ongelmien ratkaisuun.

Lista pakkauksista ja luokista (metoidesta ja luokista tiedot javadoceissa):

- Algorithms pakkaus
  - Astar luokka. A* algoritmi
  - FloodFill luokka. Flood fill algoritmi
  - CycleDetector. Syklintunnistus algoritmi
  - BFS luokka. BFS algoritmi
  - Bellman-Ford luokka . Bellman-Fordin algoritmi
  - Dijkstra luokka. Dijkstran algoritmi
  - Heuristics rajapinta. Rajapinta A* algoritmia käyttävälle luokalle
  - Path luokka. Sisältää polun linkettynä lista sekä sen pituuden
  - PathFinder rajapinta. Rajapinta kaikille polunetsintäalgoritmeille (Dijkstra, Astar, BFS, BellmanFord)
- Structures pakkaus
  - Graph pakkaus
    - Edge luokka. Määrittää kaaren kahden solmun välillä verkossa
    - Graph luokka. Sisällyttää ja hallitsee solmuja ja kaaria verkossa
    - Vertex luokka. Solmu verkossa
  - Grid pakkaus
    - Coordinate. Määrittää koordinatin gridissä
    - Grid. Laajennus verkolle, jossa voi tehdä koordinaatiston kaltaisen verkon, jossa ”solmut” ovat koordinaatteja yhden yksikön päästä toisistaan. Tukee A* polunetsintäalgoritmia.
  - Hashtable pakkaus. Sisältää tietorakenteen hajautustaulu
  - Heap. Sisältää tietorakenteen minimikeko
  - LinkedList. Sisältää tietorakenteen linkitettylista
  - Queue. Sisältää tietorakenteen jono
  - Stack. Sisältää tietorakenteen pino
  - Utils pakkaus. Sisältää apuluokkia
- Examples pakkaus. Sisältää esimerkkejä

Saavutetut aika- ja tilavaativuudet

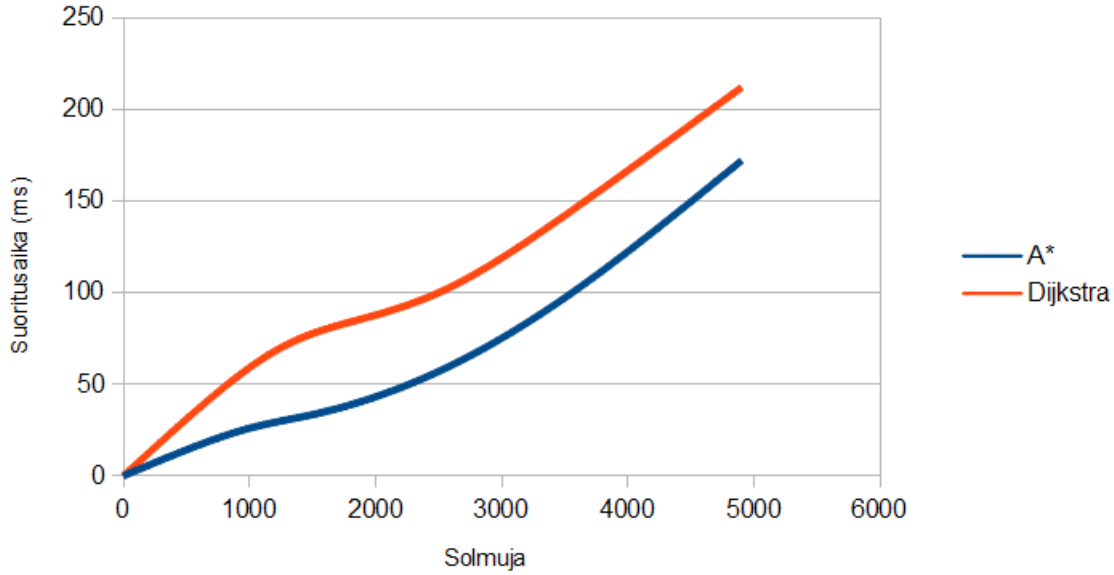
Algoritmi	Aikavaativuus	Tilavaativuus
A*	O((|V|+|E|)log(|V|)) (minimikekoa käyttäen)	O(|V|)
Flood fill	O(|V|+|E|)	O(|V|)

DFS syklien tarkastus	O(|V|+|E|)	O(|V|)
BFS	O(|V|+|E|)	O(|V|)
Bellman-Ford	O(|V||E|)	O(|V|)
Dijkstra	O((|V|+|E|)log(|V|)) (minimikekoa käyttäen)	O(|V|)

Lähde: en.wikipedia.org

Suorituskykyvertailu – A ja Dijkstra*

Tässä on A* ja Dijkstran suorituskykyvertailu. Vertailu suoritettiin vankilapako tehtävän kautta (PrisonBreak.java) laittamalla polunetsintäalgoritmit ratkaisemaan ongelmaa eri kokoisilla kartoilla. Kuvaajan x-akseli kuvaa verkossa olleiden solmujen määrää ja y-akseli algoritmin suoritusaikaa (parhaan pakoreitin löytämiseen kulunutta aikaa). Molemmat algorimit suorittivat polunetsinnän samalle kartalle ja kaaviossa näkyvät ajat kuvaavat viiden suoritusajan keskiarvoa kullakin eri verkolla.



Kaaviosta näkyy selvästi, että molempien algoritmien aikavaativuudet ovat samat, O((|V|+|E|)log(|V|)) (derivaatta on keskimäärin sama). A starin heuristiikka saa sen kuitenkin toimimaan käytännössä hieman Dijkstraa nopeammin.

Puutteet ja parannusehdotukset

A starin ja Dijkstran olisi voinut toteuttaa käyttämällä minimikeon sijasta fibonacci kekoa, jollen ne olisivat olleet hieman tehokkaampia.