Määrittelydokumentti

Johdatus

Tarkoituksena on tehdä kattava ja helppokäyttöinen verkkokirjasto, joka tarjoaa mahdollisuuden verkon luomiseen, muokkaamiseen, ja tarkasteluun sekä yleisimmät algoritmit ongelmien ratkaisuun.

Polunetsintä solmujen välillä on pääosassa, joten verkon täytyy itse tietää, millainen se on (painoton verkko käyttää BFS-hakua, pelkästään positiivisia painoja sisältävä Dijkstraa, negatiivisia sisältävä Bellman-Fordia ja heuristiikan omaava A staria). Näin ohjelmoijan ei tarvitse itse valita algoritmia polunetsinnälle. Verkon tulee tarjota myös muita yleisiä verkkoalgoritmeja, kuten Flood fill, syklin tunnistus, jne.

Tietorakenteet ja algoritmit

Lista tarvittavista algoritmeista ja niiden käyttämistä tietorakenteista:

- BFS
  - Aikavaativuudeksi tulisi tulla O(|V|+|E|) ja tilavaativuuden O(|V|). Aputietorakentaita: Hahstable, LinkedList, Stack, Queue
- Dijkstra
  - Minikeko, jossa on decrease-key ominaisuus, jonka aikavaativuus on O(log(n)). Näin Dijkstra saadaan toimimaan ajassa O((|V|+|E|)log(|V|)) ja tilavaativuudeksi tulisi O(|V|). Aputietorakentaita: Hahstable, LinkedList, Stack
- Bellman-Ford
  - Aikavaativuudeksi tulisi tulla O(|V||E|) ja tilavaativuudeksi O(|V|). Aputietorakentaita: Hahstable, LinkedList, Stack
- A*
  - Aika- ja tilavaativuudeksi tulisi tulla sama kuin Dijkstran, mutta käytännössä algoritmi toimisi hieman nopeammin heuristiikan takia. Aputietorakentaita: MinHeap, Hahstable, LinkedList, Stack
- Flood fill
  - Aikavaativuudeksi tulisi tulla O(|V|+|E|) ja tilavaativuuden O(|V|). Aputietorakentaita: Hahstable, LinkedList, Stack, Queue
- DFS (syklien tunnistus)
  - Aikavaativuudeksi tulisi tulla O(|V|+|E|) ja tilavaativuuden O(|V|). Aputietorakentaita: Hahstable, LinkedList

Aika- ja tilavaativuuksien lähteet: en.wikipedia.org

Syötteet

Verkkoluokkaa (Graph) käytetään yhdistämään solmuja (Vertex) verkkoon luomalla kaaria niiden välille. Koodina tämä toimisi seuraavasti:

```
Graph graph = new Graph();
Vertex a = new Vertex(”A”); // Solmu, jonka nimi on ”A” (vapaaehtoinen)
Vertex b = new Vertex(”B”);
Vertex c = new Vertex(”C”);
graph.connect(a,b,3);      // Alku- ja loppusolmu, sekä kaaren paino
graph.connect(b,c);        // ”Painoton kaari” (paino on 1)
```

Lyhyin polku solmusta a, solmuun c löytyisi seuraavasti:

```
graph.shortestPath(a,c);   // Palauttaa polun linkitettynä listana
```

Polunetsintäluokat ottavat konstruktorissa Graph-olion. Jokainen näistä algoritmeista toteuttaa rajapinnan PathFinder, johon liittyy metodi shortestPath(Vertex a, Vertex b);

Lopuista verkon toiminnallisuuksista ja niiden käytöstä javadocissa!

Toteutusdokumentti

Ohjelman yleisrakenne

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

- Algorithms pakkaus
  - Astar luokka. A* algoritmi
  - Algorithm luokka. Joukko pienempiä verkkoalgoritmeja
  - 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

Saavutetut aika- ja tilavaativuudet

Algoritmi	Aikavaativuus	Tilavaativuus
A*	O((|V|+|E|)log(|V|))	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|))	O(|V|)

Suorituskykyvertailu – A ja Dijkstra*

Käyttöohje

Esimerkit

Kirjaston esimerkit löytyvät Examples pakkauksesta. GLExamples.java sisältää main metodin, jossa on pohjat jokaisen esimerkin käynnistämiseen (poista vain kommentit). Lista esimerkeistä:

- SimpleGraph.java
    - Lyhyt opastus ensimmäisen verkon tekemiseen, olennaisimpiin metodeihin ja polunetsintään.
- Adventurer.java
    - Visuaalinen esitys Grid luokan toiminnasta. Seikkalija haluaa päästä takaisin linnaansa vältellen vaikeaa maastoa. Maasto vaihtelee (huonoimmasta parhaampaan) vuoristosta mereen, merestä metsään, metsästä niittyyn ja niitystä tiehen. Ohjelma piirtää punaisilla pisteillä parhaan reitin.
- PrisonBreak.java
    - Tuttu TiRan pajatehtävä, joka vertailee satunnaisella vankilan pohjapiirrustuksella A* algoritmin ja Dijkstran suorituskykyä.
- Pipes.java
    - Talossa tapahtuu putkirikko ja huone, jossa putki sijaitsi täyttyy vedellä.