Toteutusdokumentti

Yhteenveto

Kaikki alunperin määritellyt algoritmit onnistuttiin toteuttamaan toimivasti ja tehokkaasti. Ohjelman toimii tekstitilassa ja ohjelmaa käytetään komentoriviparametreillä.

Ohjelma tukee seuraavia ominaisuuksia:

- Algoritmien nopeusvertailu

Ohjelma generoi halutun kokoisen labyrintin, ja etsii siinä lyhimmän kuljettavan polun kahden satunnaisesti valitun pisteen välillä. Polun etsintää suoritetaan annettu määrä hakuja, ja löydetyn reitin pituus ja algoritmin käyttämä aika tulostetaan käyttäjälle. Mikäli hakualgoritmilla kestää yli 15 sekuntia suoriutua ratkaisusta, arvioidaan lopullinen aika jo suoritettujen hakujen avulla.

- Satunnaisen labyrintin generointi

Ohjelma generoi ja tulostaa satunnaisen labyrintin annetuilla parametreilla. Ohjelman tuloste voidaan ohjata esimerkiksi tiedostoon muokkausta varten.

- Lyhimmän polun etsiminen luetusta labyrintti-tiedostosta

Ohjelma lukee halutun tiedoston ja ratkaisee sen kuvaaman labyrintin. Ratkaisuun käytetään A* algoritmia.

- Satunnaisen labyrintin generointi ja lyhimmän polun etsintä

Kahden edellisen kohdan yhdistelmä; ohjelma generoi satunnaisen labyrintin ja etsii siitä lyhimmän polun.

Puutteet

- Huonot konfiguraatiomahdollisuudet ja osa toiminnallisuudesta on kovakoodattua.
- Käyttäjän syötteen oikeellisuuden virheentarkistus on puutteellista.

Algoritmien oikeellisuus

Bellman-Ford:

```
1.        public void findPath() {
2.              initializeNodes();
3.
4.              final LinkedList<Node> nodes = getGraph().getNodes();
5.
6.              for (int i = 0; i < nodes.size() - 1; ++i) {
7.                      for (Edge e : edges) {
8.                              relax(e.u, e.v, e.w);
9.                      }
10.             }
11.       }
12.
13.       private void relax(Node u, Node v, double w) {
14.             if (u.getDistance() == Double.MAX_VALUE) {
15.                     return;
16.             }
17.
18.             double uw = u.getDistance() + w;
19.
20.             if (v.getDistance() > uw) {
21.                     v.setDistance(uw);
22.                     v.setNearest(u);
23.             }
24.       }
```

Algoritmin lähteenä on käytetty Tietorakenteet ja algoritmit -kurssin kurssimateriaalia, ja koodin ulkoasu on tarkoituksella pyritty pitämään pseudokoodia vastaavana. Erilaista on ainoastaan 14. rivin ehto, joka estää turhan vertailun.

Algoritmin aikavaativuus on **O(|V||E|)**.