

Reitinhaku verkossa

Testaus

Harjoitustyössä käytetään automatisoitua yksikkötestausta. Testit voi suorittaa komentomallalla pois pääohjelman rivi `#define NDEBUB` ja kääntämällä ohjelma uudelleen.

Yksikkötestaus koostuu kolmesta osasta: `FileOperatorTest`, `HeapTest` ja `SearcherTest`. Nämä testaavat tietostoissa `FileOperator`, `Heap` ja `Searcher` määriteltyjen operaatioiden toimintaa. Tarkastellaan seuraavaksi testien toimintaa hieman lähemmin.

`FileOperatorTest` saa syötteenään tiedoston `testmap.txt`, jonka sisältö on kuvion 1 mukainen. `Assert`-makrot tarkistavat, että tiedostosta luetun kartan koko on 8×13 ruutua, ja että eräiden ruutujen arvot täsmäävät tiedoston sisältöön: esim. kaikkien sarakkeissa 0-2 sijaitsevien ruutujen arvon tulee olla 0.

```
8 13

00010000000000
00011111000000
00000001000000
00000001111100
00000000001000
00001111111100
00001000000000
00001000000000
```

Kuvio 1: tiedoston `testmap.txt` sisältö

`HeapTest` tarkastaa aluksi, että sen luoma keko on tyhjä. Tämän jälkeen kekoon lisätään solmut 0-4, joiden avaimet (eli etäisyydet lähtösolmusta) ovat 8, 5, 17, 0 ja 1. Kuviossa 2 on esitetty keon rakenne silloin, kun lisäys on suoritettu onnistuneesti.

Kun lisäys on valmis, tarkastetaan keon alkioden lukumäärä, jonka tulee olla 5. Tämän jälkeen tarkastetaan, että operaatiot `getParent`, `getLeft` ja `getRight` palauttavat oikean taulukkoindeksin. Lopuksi tarkastetaan, että operaatio `delMin` palauttaa toistuvasti kutsuttaessa solmut järjestyksessä 3, 4, 1, 0 ja 2.

0 0 0 1 0 0 0 0 0 0 0 0 0 0	0 0 0 * 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1 1 1 1 0 0 0 0 0 0	0 0 0 * * * * * 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0 0 0	0 0 0 0 0 0 0 * 0 0 0 0 0
0 0 0 0 0 0 0 1 1 1 1 0 0 0	0 0 0 0 0 0 0 * * * * 0 0
0 0 0 0 0 0 0 0 0 0 0 1 0 0	0 0 0 0 0 0 0 0 0 0 0 * 0 0
0 0 0 0 1 1 1 1 1 1 1 0 0 0	0 0 0 0 * * * * * * * 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0	0 0 0 0 * 0 0 0 0 0 0 0 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0	0 0 0 0 * 0 0 0 0 0 0 0 0

Kuvio 4: A* sovellettuna tiedostoon testmap.txt

Toista tarkastelua varten molemmille algoritmeille annettiin syötteeksi tiedosto map2.txt, jossa määritellään jälleen 8 x 13 ruudun kokoinen kartta. Tällä kertaa kartalla on kuitenkin useita mahdollisia polkuja alkusijainnista (0, 7) ja loppusijaintiin (7, 12). Kuvioissa 5 ja 6 on esitetty vasemmalla alkuperäinen kartta ja oikealla algoritmien löytämät lyhimmat reitit. Jälleen nähdään, että molemmat algoritmit löytävät lyhimmän polun sen sijaan, että esimerkiksi oikaisisivat ruudun (4, 10) kautta.

9 1 1 1 6 6 5 0 4 3 3 2 2	9 * * * 6 6 5 * 4 3 3 2 2
1 1 7 1 7 1 1 1 0 0 0 0	* * 7 * 7 * * * 1 0 0 0
1 9 6 1 1 1 7 0 0 1 1 1	* 9 6 * * * 7 0 0 * * *
1 1 0 0 0 0 7 0 3 1 0 9 1	* * 0 0 0 0 7 0 3 * 0 9 *
7 1 1 1 1 3 7 3 3 1 8 1 1	7 * * * * 3 7 3 3 * 8 * *
5 5 4 5 1 1 1 3 3 1 0 1 9	5 5 4 5 * * * 3 3 * 0 * 9
2 3 4 4 3 3 1 1 1 0 1 1	2 3 4 4 3 3 * * * * 0 * *
3 3 3 3 3 3 2 2 3 3 0 3 1	3 3 3 3 3 3 2 2 3 3 0 3 *

Kuvio 5: Dijkstran algoritmin löytämä reitti tiedoston map2.txt kartalla

9 1 1 1 6 6 5 0 4 3 3 2 2	9 * * * 6 6 5 * 4 3 3 2 2
1 1 7 1 7 1 1 1 0 0 0 0	* * 7 * 7 * * * 1 0 0 0
1 9 6 1 1 1 7 0 0 1 1 1	* 9 6 * * * 7 0 0 * * *
1 1 0 0 0 0 7 0 3 1 0 9 1	* * 0 0 0 0 7 0 3 * 0 9 *
7 1 1 1 1 3 7 3 3 1 8 1 1	7 * * * * 3 7 3 3 * 8 * *
5 5 4 5 1 1 1 3 3 1 0 1 9	5 5 4 5 * * * 3 3 * 0 * 9
2 3 4 4 3 3 1 1 1 0 1 1	2 3 4 4 3 3 * * * * 0 * *
3 3 3 3 3 3 2 2 3 3 0 3 1	3 3 3 3 3 3 2 2 3 3 0 3 *

Kuvio 6: A*:n löytämä reitti tiedoston map2.txt kartalla

Kolmannessa tarkastelussa käytettiin tiedostoa map.txt, jossa määritelty 30 x 30 ruudun kokoinen kartta on esitetty kuviossa 7. Jälleen sekä Dijkstran algoritmi että A* löysivät saman reitin alkusijainnista (0, 0) loppusijaintiin (29, 29). Reitin todistaminen lyhimmäksi mahdolliseksi on jo hieman työläämpää ja on jätetty tässä tekemättä. Tärkeää kuitenkin on, että molemmat algoritmit ylipäättään löytävät reitin, ja että reitti on riippumaton käytetystä algoritmista. Kuvio 8 esittää löydettyä reittiä.

```

9 1 1 1 6 6 5 0 4 3 3 2 2 9 1 1 1 6 6 5 0 4 3 3 2 2 9 1 1 1
1 1 7 1 7 1 1 1 0 0 0 0 0 1 1 7 1 7 1 1 1 0 0 0 0 0 1 1 7 1
1 9 6 1 1 1 3 3 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 9 6 1
1 1 0 0 0 0 0 0 3 1 0 9 1 1 1 0 0 0 0 0 0 3 1 0 9 1 1 1 0 0
7 1 1 1 1 3 3 3 3 1 0 1 1 7 1 1 1 1 3 3 3 3 1 0 1 1 7 1 1 1
5 5 4 5 1 1 1 3 3 1 0 1 9 5 5 2 5 1 1 1 3 3 1 0 1 9 5 5 4 5
2 3 4 4 3 3 1 1 1 1 0 1 1 2 3 3 4 3 3 1 1 1 1 0 1 1 2 3 4 4
3 3 3 3 3 3 2 2 3 3 0 3 1 3 3 4 3 3 3 2 2 3 3 0 3 1 3 3 3 3
9 1 1 1 6 6 5 0 4 3 3 2 2 9 1 5 1 6 6 5 0 4 3 3 2 2 9 1 1 1
1 1 7 1 7 1 1 1 0 0 0 0 0 1 1 6 1 7 1 1 1 0 0 0 0 0 1 1 7 1
1 9 6 1 1 1 3 3 0 1 1 1 1 1 9 7 1 1 1 3 3 0 1 1 1 1 1 9 6 1
1 1 0 0 0 0 0 0 3 1 0 9 1 1 1 8 0 0 0 0 0 3 1 0 9 1 1 1 0 0
7 1 1 1 1 3 3 3 3 1 0 1 1 7 1 2 1 4 3 3 3 3 1 0 1 1 7 1 1 1
5 5 4 5 1 1 1 3 3 1 0 1 9 5 5 4 5 1 1 1 3 3 1 0 1 9 5 5 4 5
2 3 4 4 3 3 1 1 1 1 0 1 1 2 3 4 4 3 3 1 1 1 1 0 1 1 2 3 4 4
3 3 3 3 3 3 2 2 3 3 0 3 1 3 3 3 3 3 3 2 2 3 3 0 3 1 3 3 3 3
9 1 1 1 6 6 5 0 4 3 3 2 2 9 1 1 1 6 6 5 0 4 3 3 2 2 9 1 1 1
1 1 7 1 7 1 1 1 0 0 0 0 0 1 1 7 1 7 1 1 1 0 0 0 0 0 1 1 7 1
1 9 6 1 1 1 3 3 0 1 1 1 1 1 9 6 1 1 1 3 3 0 1 1 1 1 1 9 6 1
1 1 0 0 0 0 0 0 3 1 0 9 1 1 1 0 0 0 0 0 0 3 1 0 9 1 1 1 0 0
7 1 1 1 1 3 3 3 3 1 0 1 1 7 1 1 1 1 3 3 3 3 1 0 1 1 7 1 1 1
5 5 4 5 1 1 1 3 3 1 0 1 9 5 5 4 5 1 1 1 3 3 1 0 1 9 5 5 4 5
2 3 4 4 3 3 1 1 1 1 0 1 1 2 3 4 4 3 3 1 1 1 1 0 1 1 2 3 4 4
3 3 3 3 3 3 2 2 3 3 0 3 1 3 3 3 3 3 3 2 2 3 3 0 3 1 3 3 3 3
1 1 0 0 0 0 0 0 3 1 0 9 1 1 1 0 0 0 0 0 0 3 1 0 9 1 1 1 0 0
9 1 1 1 6 6 5 0 4 3 3 2 2 9 1 1 1 6 6 5 0 4 3 3 2 2 9 1 1 1
1 1 7 1 7 1 1 1 0 0 0 0 0 1 1 7 1 7 1 1 1 0 0 0 0 0 1 1 7 1
1 9 6 1 1 1 3 3 0 1 1 1 1 1 9 6 1 1 1 3 3 0 1 1 1 1 1 9 6 1
9 1 1 1 6 6 5 0 4 3 3 2 2 9 1 1 1 6 6 5 0 4 3 3 2 2 9 1 1 1
7 1 1 1 1 3 3 3 3 1 0 1 1 7 1 1 1 1 3 3 3 3 1 0 1 1 7 1 1 1

```

Kuvio 7: Kartta tiedostosta map.txt

```

* 1 1 1 6 6 5 0 4 3 3 2 2 9 1 1 1 6 6 5 0 4 3 3 2 2 9 1 1 1
* 1 7 1 7 1 1 1 0 0 0 0 0 1 1 7 1 7 1 1 1 0 0 0 0 0 1 1 7 1
* 9 6 1 1 1 3 3 0 * * * * * * * * * * * * * * 1 9 6 1
* * 0 0 0 0 0 0 3 * 0 9 1 1 1 0 0 0 0 0 0 3 1 0 9 * 1 1 0 0
7 * * * * 3 3 3 3 * 0 1 1 7 1 1 1 1 3 3 3 3 1 0 * * 7 1 1 1
5 5 4 5 * * * 3 3 * 0 1 9 5 5 2 5 1 1 1 3 3 1 0 * 9 5 5 4 5
2 3 4 4 3 3 * * * * 0 1 1 2 3 3 4 3 3 1 1 1 1 0 * * 2 3 4 4
3 3 3 3 3 3 2 2 3 3 0 3 1 3 3 4 3 3 3 2 2 3 3 0 3 * * * 3 3
9 1 1 1 6 6 5 0 4 3 3 2 2 9 1 5 1 6 6 5 0 4 3 3 2 2 9 * 1 1
1 1 7 1 7 1 1 1 0 0 0 0 0 1 1 6 1 7 1 1 1 0 0 0 0 0 * * 7 1
1 9 6 1 1 1 3 3 0 1 1 1 1 1 9 7 1 1 1 3 3 0 1 1 1 1 * 9 6 1
1 1 0 0 0 0 0 0 3 1 0 9 1 1 1 8 0 0 0 0 0 0 3 1 0 9 * * 1 0 0
7 1 1 1 1 3 3 3 3 1 0 1 1 7 1 2 1 4 3 3 3 3 1 0 * * 7 1 1 1
5 5 4 5 1 1 1 3 3 1 0 1 9 5 5 4 5 1 1 1 3 3 1 0 * 9 5 5 4 5
2 3 4 4 3 3 1 1 1 1 0 1 1 2 3 4 4 3 3 1 1 1 1 0 * * 2 3 4 4
3 3 3 3 3 3 2 2 3 3 0 3 1 3 3 3 3 3 3 2 2 3 3 0 3 * * * 3 3
9 1 1 1 6 6 5 0 4 3 3 2 2 9 1 1 1 6 6 5 0 4 3 3 2 2 9 * 1 1
1 1 7 1 7 1 1 1 0 0 0 0 0 1 1 7 1 7 1 1 1 0 0 0 0 0 * * 7 1
1 9 6 1 1 1 3 3 0 1 1 1 1 1 9 6 1 1 1 3 3 0 1 1 1 1 * 9 6 1
1 1 0 0 0 0 0 0 3 1 0 9 1 1 1 0 0 0 0 0 0 0 3 1 0 9 * * 1 0 0
7 1 1 1 1 3 3 3 3 1 0 1 1 7 1 1 1 1 3 3 3 3 1 0 * * 7 1 1 1
5 5 4 5 1 1 1 3 3 1 0 1 9 5 5 4 5 1 1 1 3 3 1 0 * 9 5 5 4 5
2 3 4 4 3 3 1 1 1 1 0 1 1 2 3 4 4 3 3 1 1 1 1 0 * * 2 3 4 4
3 3 3 3 3 3 2 2 3 3 0 3 1 3 3 3 3 3 3 2 2 3 3 0 3 * 3 3 3 3
1 1 0 0 0 0 0 0 3 1 0 9 1 1 1 0 0 0 0 0 0 0 3 1 0 9 * * * 0 0
9 1 1 1 6 6 5 0 4 3 3 2 2 9 1 1 1 6 6 5 0 4 3 3 2 2 9 * * *
1 1 7 1 7 1 1 1 0 0 0 0 0 1 1 7 1 7 1 1 1 0 0 0 0 0 1 1 7 *
1 9 6 1 1 1 3 3 0 1 1 1 1 1 9 6 1 1 1 3 3 0 1 1 1 1 1 9 6 *
9 1 1 1 6 6 5 0 4 3 3 2 2 9 1 1 1 6 6 5 0 4 3 3 2 2 9 1 1 *
7 1 1 1 1 3 3 3 3 1 0 1 1 7 1 1 1 1 3 3 3 3 1 0 1 1 7 1 1 *

```

Kuvio 8: A*:n ja Dijkstran algoritmin löytämä reitti kuvion 7 esittämällä kartalla