TiRaLabra
Saku Säisä

Suorituskykymittauksia

Käytin työssä ensin Javan valmiita tietorakenteita ArrayList ja PriorityQueue. Korvasin ne sittemmin omilla toteutuksilla niin, että ArrayListin korvasi simppeli boolean 2D-taulukkoon pohjaava rakenne (joka toimii loistavasti juuri tässä käytössä) ja PriorityQueuen korvasi oma binäärikekoni.

Mittasin sitten aikoja, mitä eri rakenteilla meni suorittaa tiettyjä operaatioita. Ajoin testit kohtalaisen suurilla iteraatiomäärillä ja laskin niiden mittaustuloksista keskiarvot.

Ensimmäiset testit

Ensimmäinen testi testaa koko ruudukon sisällön lisäämistä ”closed set” -rakenteisiin. Toinen testi tarkistaa, onko closed setissä kahta tiettyä gridCelliä (joista toinen on setissä ja toista ei ole). Kolmas testi puolestaan lisää koko ruudukon sisällön ”open set” -rakenteeseen. Solujen arvot, joiden mukaan PQ/keko sijoittaa jokaisen solun oikeaan paikkaansa, on arvottu randomilla ohjelman suorituksen alussa. Arvot ovat molemmille toteutuksille samat.

```
Using a 1250*1250 grid (=1562500 cells).
Iterating each test 500 times to determine an average time.

*** "closed set" performance adding cells into a prebuilt structure ***
Average time taken over 500 iterations was 22.208ms.
*** "closed set" performance adding cells into a custom structure ***
Average time taken over 500 iterations was 20.494ms.

*** "closed set" performance checking if premade structure contains a
specific cell ***
Average time taken over 500 iterations was 3.0ms.
*** "closed set" performance checking if custom structure contains a specific
cell ***
Average time taken over 500 iterations was 0.0020ms.

*** "open set" performance adding cells into a prebuilt structure ***
Average time taken over 500 iterations was 89.376ms.
*** "open set" performance adding cells into a custom binary heap ***
Average time taken over 500 iterations was 89.184ms.
```

Testialustana toimi kohta neljä vuotta vanha 13” Macbook Pro.