

Testaus

Raportoin tähän dokumenttiin käsin tehdyt testaukset. Testaan käsin asiat joita ei ole järkeä testata koodilla. Suurimman osan testaamisesta olen suorittanut Junit testeillä jotka löytyvät src/test kansion alta.

Testi 1:

MazeMakerin drawPath piirtää oikeanlaisen reitin yksinkertaisissa tapauksissa

Tulos:

Kyllä

Toistettavuus:

Anna kuva jossa on toisella puolella punainen piste ja toisella sininen.

Ohjelman pitäisi piirtää suora viiva pisteiden välille

Testi 2:

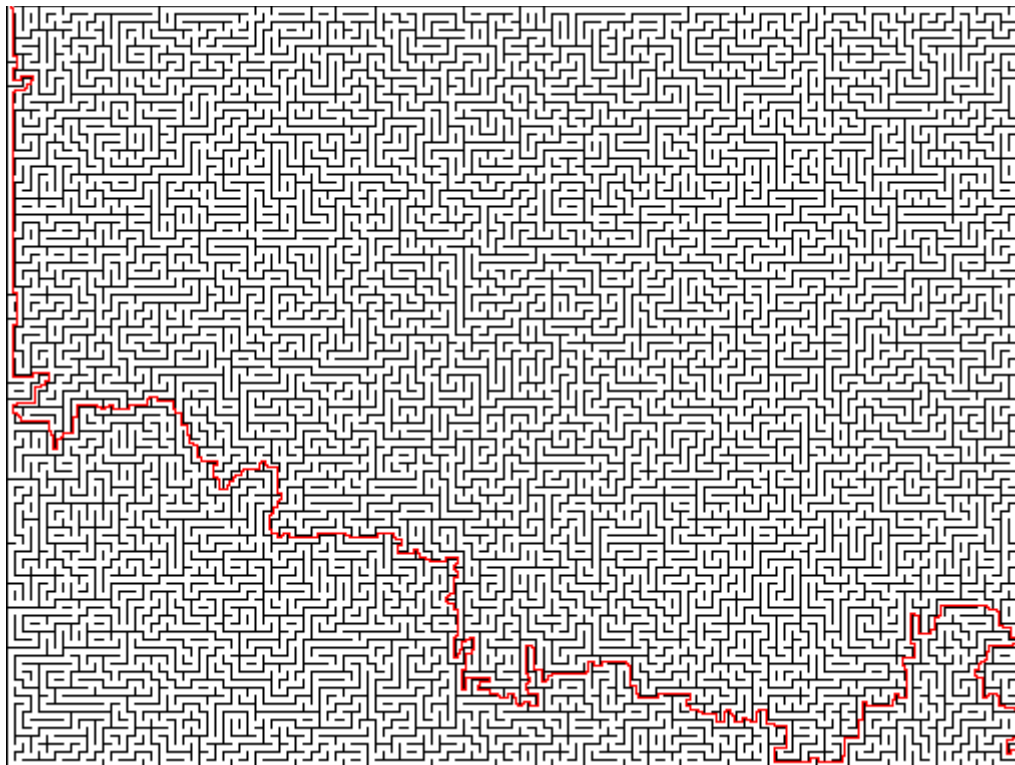
MazeMakerin drawPath piirtää oikeanlaisen reitin monimutkaisissa tapauksissa käyttäen Dijkstraa

Tulos:

Kyllä

Toistettavuus:

Etsi internetistä maze joka on toteutettu mustalla ja valkoisella, lisää siihen tarvittavat alku ja loppupiste, tarkista että reitti on lyhin mahdollinen, tai käytä projektikansiota löytyvää maze.giffiä



Testi 3:

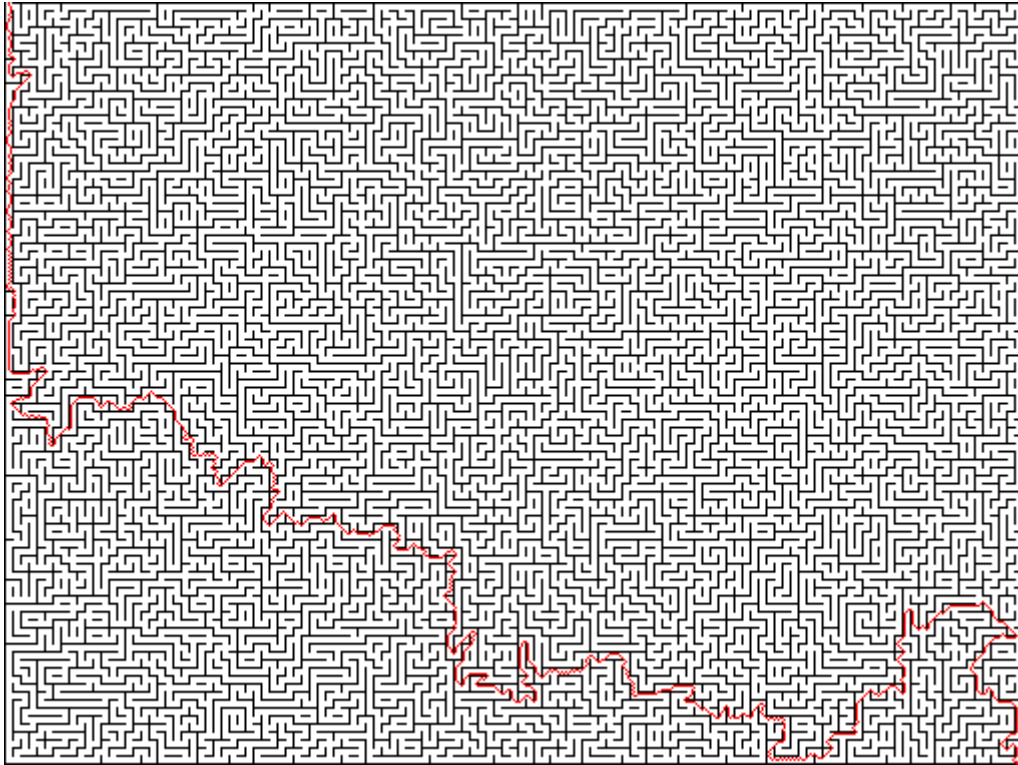
Sama kuin testi 2 mutta A starlil atoteutettuna

Tulos:

Kyllä

Toistettavuus:

Sama kuin ylempi

**Testi 4:**

Eroaako oman Heap luokan ja javan PriorityQueueen antama reittitulos

Tulos:

Ei (=testi läpi)

Toistettavuus:

Vaihda DijkstraSolver luokan Heap javan PriorityQueueeksi ja käytä delMinin sijaan PriorityQueueen pop metodia

Testi 5:

A* menee suoraa maalia kohti

Tulos:

Kyllä

Toistettavuus:

Lisää A* relax osuuteen että jokainen läpikäytävä node lisätään pathiin, jolloin ne piirtyy karttaan. Tee labyrintti jossa maalin ja alun välillä ei ole pisteitä. Reitin pitäisi mennä suoraan maaliin.

Testi 6:

Dijkstra tutkii isomman alueen kun A*

Tulos:

Kyllä

Toistettavuus:

Lisää molempien relaxiin että jokainen läpikäytävä node lisätään pathiin. Vertaa dijkstran ja A* antamia tuloksia.

Nopeustestit

Testi 1:

Oman heapin ja PriorityQueueen nopeuserot suurella yksinkertaisella syötteellä

Tulos:

Javan PriorityQueue 10 suorituskerran keskiarvo: 5.038s

Oman heappini 10 suorituskerran keskiarvo: 5.352s

Olen tyytyväinen saavuttamaani aikaan. Veikkaan että priorityqueueella on arrayn kopiointiin käytössä parempia työkaluja kun koko listan looppaaminen uudestaan.

Toistettavuus:

Sama kun testi 3.

Testi 2:

Oman NodeListin ja ArrayListin vertaus Dijkstran käytössä

Tulos:

Oma huomattavasti nopeampi koska listan taulukko on alunperin alustettu neljällä.

Javan priorityquella 10 suorituskerran keskiarvo: 5.038s (sama kuin edellinen)

Omalla NodeListalla 10 suorituskerran keskiarvo: 3.774s

Testi 3:

Dijkstra ja A* nopeusero yksinkertaisella isolla seinättömällä labyrintillä (bigSimple.png)

Tulos:

A* 10 suorituskerran keskiarvo: 3.327s

Dijkstra 10 suorituskerran keskiarvo= 7.146s

Testi 4:

Dijkstra ja A* nopeusero monimutkaisella 509px * 381px syötteellä

Tulos:

Dijkstra 10 suorituskerran keskiarvo: 1.404

A* 10 suorituskerran keskiarvo: 1.327