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:

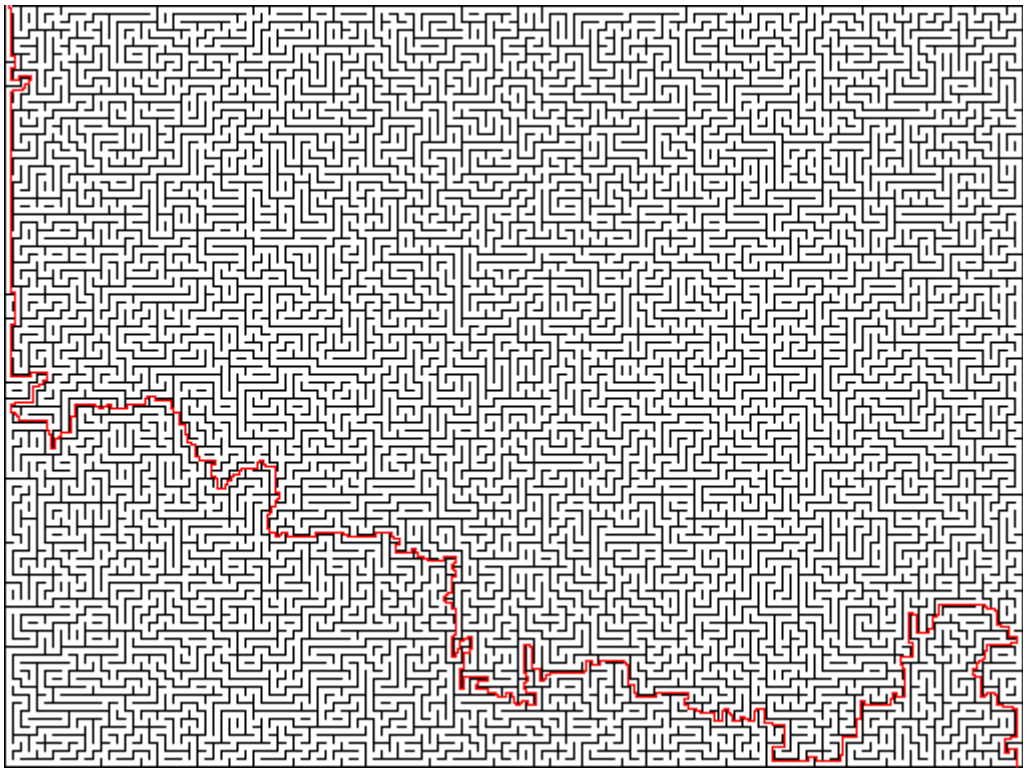
MazeMakerin drawPath piirtä oikeanlaisen reitin monimutkaisissa tapauksissa

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 projektikansiosta löytyvää maze.giffiä



Testi 3:

Eroaako oman Heap luokan ja javan PriorityQueuen tulokset

Tulos:

Ei (=testi läpi)

Toistettavuus:

Vaihda DijkstraSolver luokan Heap javan PriorityQueueksi ja käytä delMinin sijaan PriorityQueuen pop metodia

Testi 4:

Oman heapin ja PriorityQueuen 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 priorityqueuella on arrayn kopiointiin käytössä parempia työkaluja kun koko listan looppaaminen uudestaan.

Toistettavuus:

Sama kun testi 3.

Testi 5:

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