Testausdokumentti

Algoritmien perustoiminnallisuutta on testattu yksinkertaisesti eri syötteillä ja varmistamalla, että koodi toimii oikein. Molempia reitinhakualgoritmeja on myös testattu suurella määrällä satunnaisia muuttujia suorituskykytestiä ja jotta varmistetaan, että algoritmit toimivat kaikissa mahdollisissa tapauksissa. Koska molemmat algoritmit testataan erikseen, vertailu ei ole aivan tarkkaa reittien satunnaisuuden takia. Isoilla testimäärillä molempiin testeihin kuitenkin mahtuu kaikenkokoisia reittejä ja tarkkuus paranee.

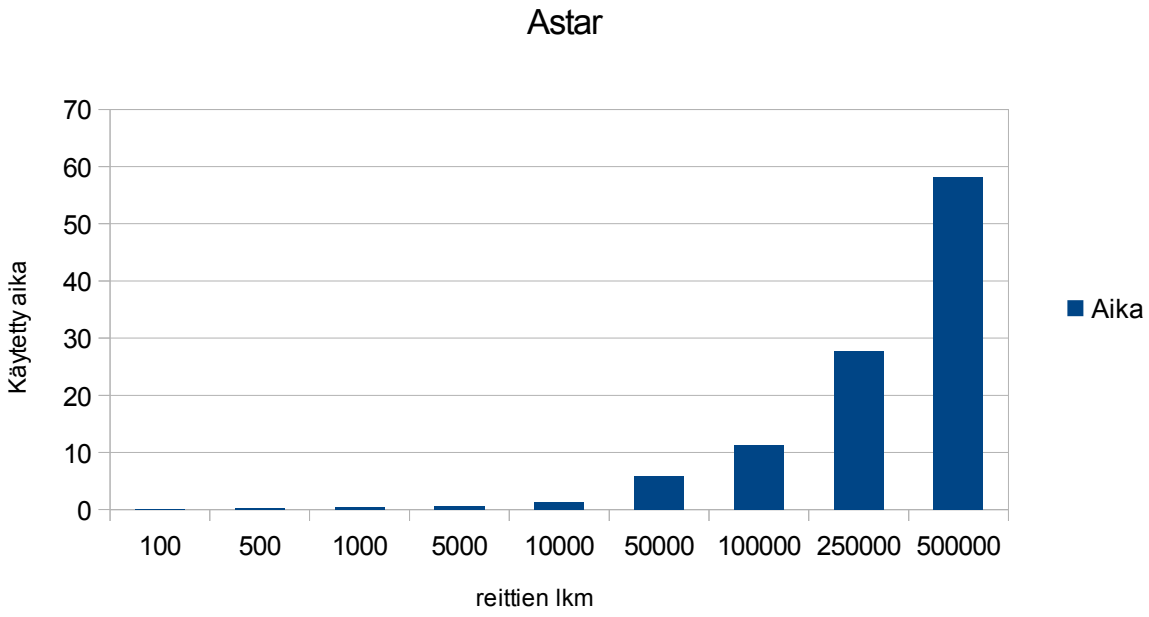
Kuten allaolevista kuvaajista näkee, Dijkstra on jonkin verran A* hitaampi.

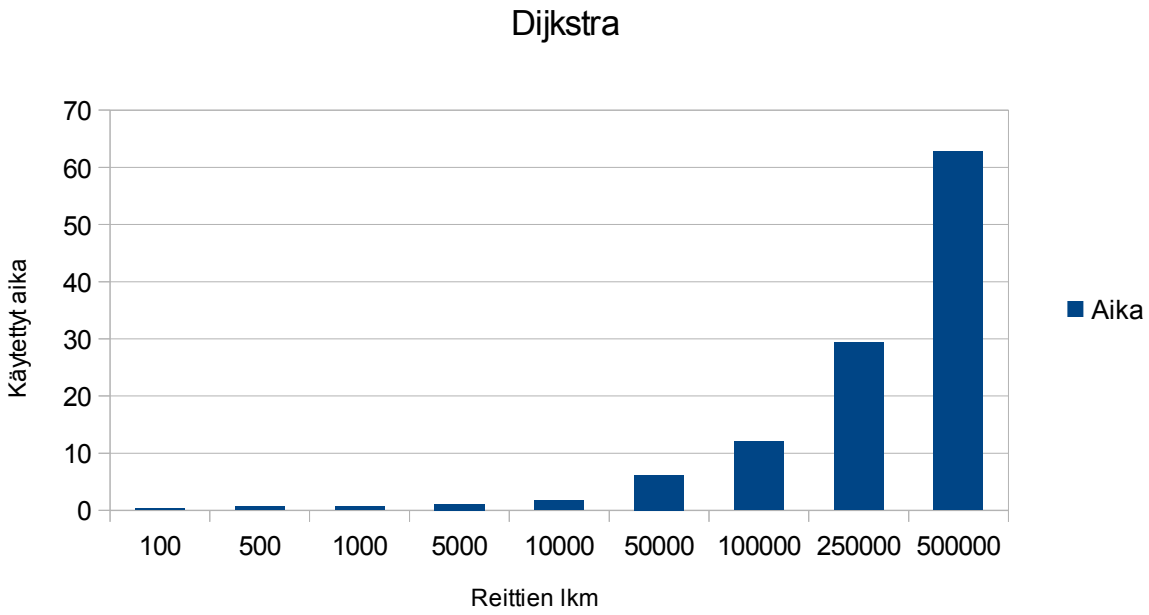
Astar

Reittien lkm	Aika
100	0.13
500	0.247
1000	0.414
5000	0.729
10000	1.262
50000	5.888
100000	11.254
250000	27.803
500000	58.178

Dijkstra

Reittien lkm	Aika
100	0.325
500	0.677
1000	0.726
5000	1.133
10000	1.757
50000	6.222
100000	12.04
250000	29.346
500000	62.741





Käytetty aika myös kasvaa exponentiaalisesti useampia reitinhakuja tehdessä, mikä tukee aikavaativuusarviota O(n+nlog(n)).

Tietorakenteiden testaamisessa yksinkertaisimmat setterit ja getterit on jätetty huomioimatta: en näe järkeä testata metodeita, jotka tallentavat tai palauttavat yhden muuttujan. Koordinaatti-luokan monimutkaisempia laskutoimituksia vaativat setterit on testattu. Muiden tietorakenteiden testejä varten näistä luokista löytyy metodeja, joilla voi muokata suoraan niiden sisältöä. Näin on voitu mm. Testata heapify siten, että keolle annetaan insert-metodi ohittaen satunnaisessa järjestyksessä oleva taulukko, ja buildheap-metodilla rakennetaan heapifyn avulla tästä toimiva keko.

Käyttöliittymälle tai Mainille ei ole kirjoitettu testejä.