Testausraportti

Otto-Ville Raitolahti

Lyhimpiä polkuja voi etsiä kolmessa erilaisen tyyppisessä verkossa. Yksinkertaisin verkko on painoton. Painottomana verkkona voi kuvata esimerkiksi labyrintin. Painolliset verkot jakaantuvat verkkoihin, jossa ei ole negatiivisia kaaripainoja ja verkkoihin, joissa on negatiivisia kaaripainoja. Mikä algoritmi kannattaa valita kuhunkin tilanteeseen?

Testaamme eri algoritmien toimintaa bmp-tyyppisillä kuvilla. Mustavalkoiset kuvat ovat painottomia verkkoja, värilliset painollisia.

Käytämme testaamisessa Suorituskykytesti-luokan testaaKentta-metodia. Metodi hakee annetun kuvatiedoston ja muodostaa siitä verkon. Verkossa haetaan polku vasemmasta yläkulmasta vasempaan alakulmaan annetulla algoritmilla. Jokainen algoritmi lopettaa löydettyään polun. Metodi mittaa polunhakemiseen kuluvan ajan ja tulostaa sen. Tämän jälkeen algoritmin löytämä ratkaisu ja läpikäydyt solmut voidaan piirtää uuteen kuvaan. Testit ovat toistettavissa milloin vain. Täytyy kutsua testaaKentta-metodia. Javan roskienkeruu on ohjattu tehtäväksi juuri ennen ajanottoa. Testit toistetaan kymmenen kertaa ja tuloksista otetaan keskiarvo.

Testit suoritetaan Intel® Core™ i5-3470 CPU @ 3.20GHz × 4 -suorittimella.

Painottomien verkkojen testaaminen

Kannattaako painottomissa verkoissa käyttää painollisissa verkoissa toimivia algoritmeja, kuten Dijkstra tai A* -algoritmeja? Painottomassa verkossa tavallinen leveyshaku löytää lyhimmät polut jokaiseen solmuun ajassa O(|V| + |E|). On odotettavaa, että leveyshaku toimii nopeammin. Miten Dijkstra ja A* suoriutuisivat binaarikekoa käyttäen? Entä miten ne suoriutuvat Fibonacci-kekoa käyttäen?

Algoritmeihin on mahdollista tehdä erilaisia optimointeja. Nämä optimoinnit usein parantavat algoritmin suorituskykyä vakiokertoimella, mutta aikavaativuusluokka pysyy samana. Bidirectional - algoritmit aloittavat alkusolmusta ja loppusolmusta ja yhdistävät polut, kun ne kohtaavat. Teoriassa aikaa pitäisi kulua vähemmän. Useista polunhakemisalgoritmeista voi tehdä Bidirectional -tyyppisen.
Vertaamme Bidirectional-leveyshakua tavalliseen leveyshakuun.

Testaamme algoritmien toimintaa painottomissa verkoissa erikokoisilla labyrinteilla. Labyrintissa valkoisiin ruutuihin voi kulkea ja mustiin ei ollenkaan. Aloitamme 100x100 kokoisella labyrintilla. Suurin labyrintti on noin 2000x4000 pikseliä.

Testeistä saatiin seuraavat tulokset:

Solmujen määrä	Kaarien määrä	Dijkstra (Binaarikeolla)	A* (Binaarikeolla	Dijkstra (Fibonacci-keolla)	A* (Fibonacci-keolla)	BFS	Bidirectional
10000	20241	4	4	8	3	2	3
63001	109549	10	13	21	17	3	3
90000	175057	15	20	34	26	5	4
126002	219098	19	25	41	27	6	6
250000	438006	43	60	83	72	16	13
500000	875992	73	110	128	108	20	22
999000	1751966	159	265	287	248	38	35
1998000	3503906	263	492	548	492	55	66
3996000	7007842	712	1238	1851	1742	129	183
7992000	14015658	1393	2590	2655	2836	243	289

Tulokset ovat millisekunteja.

Kaikki optimoidut algoritmit ovat käytännössä hitaampia. Mistä tämä johtuu? A*-algoritmin heuristiikasta ei näytä olevan hyötyä labyrintissa. On syytä testata algoritmien toimintaa myös helpommassa verkossa.