Viikkoraportti

Viikko 0

Ohjelman kehitys lähti hyvin käyntiin. Sain implementoitua nopeasti A*-algoritmin ja testattua sitä muutamalla kuvalla, jotka löytyvät Tiralabra/test-images -kansiosta. Algoritmi näyttäisi toimivan oikein ja löytävän aina optimaalisen reitin, jos sellainen on olemassa.

Testien tuloksia:

- Pieni kuva (leveys ja korkeus kymmeniä pikseleitä) ratkeaa muutamassa millisekunnissa
- Keskikokoinen kuva (muutamia satoja pikseleitä) ratkeaa muutamassa kymmenessä tai sadassa millisekunnissa riippuen reitin pituudesta
- Erittäin iso kuva (10 000 x 10 000 pikseliä, hyvin monimutkainen sokkelo) ratkeaa noin puolessatoista minuutissa. (Luotu verkko sisältää 50 miljoonaa avointa solmua, joista algoritmi joutuu tarkistamaan lähes jokaisen).

Minulla ei ollut ennestään tarkkaa käsitystä A*:n toiminnasta, joten opin perinpohjaisesti sen toimintaperiaatteen. Testatessani edellä mainittua isoa sokkeloa opin myös, miten huomattavasti mitättömiltäkin vaikuttavat optimoinnit vaikuttavat suoritusaikaan, kun syöte on hyvin iso.

Seuraavaksi lisään uusia reitinetsintäalgoritmeja ja ohjelmalogiikkaa, jolla eri algoritmeja voi testata helposti eri syötteillä.

Viikko 1

Tällä viikolla lisäsin valikoimaan Dijkstran algoritmin sekä aloitin jump point searchin toteuttamisen, mutta se osoittautuikin oletettua haastavammaksi, joten en saanut sitä tehtyä tällä viikolla. Lisäsin myös yksinkertaisen tekstipohjaisen käyttöliittymän, joka tulostaa eri algoritmien testaustuloksia ja tallentaa löydetyt polut kuvatiedostoon. Kirjoitin myös joitakin testejä. Jätin niitä kirjoittamatta sellaisille metodeille ja luokille, joiden testaaminen ei mielestäni ollut kovin mielekästä.

Dijkstran algoritmi oli yllättävän hidas, mutta se saattaa osittain johtua siitä, että javassa ei ole valmiina kekoa, joka sisältäisi toiminnot alkion arvon muuttamiseen. Ainoa löytämäni vaihtoehto sille on käyttää PriorityQueueta ja arvon päivittämiseen kutsua queue.remove(o) ja queue.offer(o), mitkä molemmat ovat hitaita operaatioita varsinkin suurilla syötteillä.

Seuraavaksi teen jump point searchin loppuun, lisään ehkä breadth-first searchin ja aloitan toteuttamaan omia tietorakenteita.

Viikko 2

Ohjelma edistynee aikataulussa. Tällä viikolla koodasin jump point searchin loppuun, lisäsin breadth-first searchin ja aloitin toteuttamaan tietorakenteita. Tietorakenteista sain tehtyä ainakin alustavasti toimivaksi ArrayListin sekä HashSetin.

Kirjoitin muistaakseni kaikelle uudelle lisätylle testit, paitsi hakualgoritmille. Millekään hakualgoritmille ei ole tällä hetkellä omia automaattisia testejä, mutta niiden toiminnallisuuden voi tarkistaa manuaalisesti MainTestin tuottamasta tulosteesta.

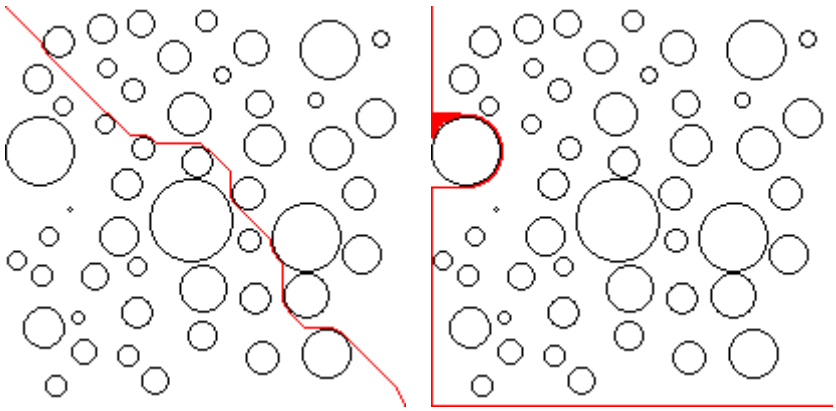
Yllättävästi breadth-first search oli monilla syötteillä nopeampi kuin A*, vaikka A* käsittelee teoriassa paljon pienemmän joukon solmuja löytääkseen maalin. Ilmeisesti A*:n tekemät keko-operaatiot ovat vain huomattavasti hitaampia kuin BFS:n nopeat jono-operaatiot.

Ensi viikolla toteutan lisää tietorakenteita sekä ehkä vielä jonkun hakualgoritmin.

Viikko 3

Tällä viikolla lisäsin omien tietorakenteiden listaan keon, linkitetyn listan, jonon ja pinon, ja muokkasin kaikki algoritmit käyttämään niitä Javan valmiiden tietorakenteiden sijasta. Yksikkötestien tulosten perusteella kaikki tietorakenteet toimivan kuin pitääkin, mutta yksi hakualgoritmi antoi hieman eri reitin kuin Javan tietorakenteilla suoritettuna. Reitti oli kuitenkin yhä optimaalinen, joten en nähnyt tässä ongelmaa. Oma tietorakenteeni todennäköisesti vain käsitteli samanarvoisia alkioita hieman eri järjestyksessä.

Lisäsin myös ohjelmaan uuden hakualgoritmin, depth-first searchin. Sen ei ole tarkoituskaan löytää lyhintä reittiä, vaan se on mukana suoritusnopeuksien mittauksessa. DFS on useilla syötteillä huomattavan nopea, esimerkiksi alla olevan kuvan syötteellä polun löytymiseen vasemmasta yläkulmasta oikeaan alakulmaan kului A*:lla 40 millisekuntia, kun DFS:llä kului vain 3 millisekuntia.



A:n tulos (40 ms)* *DFS:n tulos (3 ms)*

Tässä vaiheessa olen toteuttanut algoritmit, jotka alun perin suunnittelin toteuttavani, sekä kaikki niiden tarvitsemat tietorakenteet. Ensi viikolla teen lisää erilaisia syötekuvia ja alan dokumentoida algoritmien suoritusaikoja niillä. Tietorakenteiden osalta saatan harkita Fibonacci heapin toteuttamista, sillä Dijkstran algoritmi vaatii sen toimiakseen tehokkaimmin. Aloitan ehkä myös

yksinkertaisen graafisen käyttöliittymän tekemisen, joka helpottaisi syötteen antamista ja tulosten tutkimista.