TESTAUSDOKUMENTTI 14.6.2013
Fifteen puzzle -tietorakenteiden harjoitustyö

@Tero Mäntylä

Mitä tästä dokumentista löytyy
Aluksi kerron lyhyesti käsin testauksesta ja yksikkötesteistä. Sen jälkeen käyn läpi projektiin liittyvät tehokkuustestit. Tehokkuustestaus jakaantuu seuraaviin kolmeen osaan: Eri heuristiikkojen ja toteutustapojen vaikutus IDA*-algoritmin tehokkuuteen, A*-algoritmin vertailu IDA*-algoritmiin, oman keko toteutuksen vertailu Javan PriorityQueue luokkaan. Kaikki testit on ajettu pajaluokan tietokoneella.

JUNIT ja käsin testaus
Yksikkötestauksen kattavuus on hyvä. Nyrkkituntumalta arvioiden se on päälle 90%. Testien määrä ei ehkä ole paras mittari niistä, koska yksi testi joissakin tapauksissa testaa joko koko tai melkein koko luokan toiminnallisuuden. Joten älkää katsoko testien määrää. Yksikkötestit tuli kirjoitettua vasta kun koodi oli jo toimivaa. Joten testeistä oli varsinaista hyötyä vasta työn loppuvaiheessa, jossa tuli kokeiltua erilaisten muutosten vaikutusta algoritmien tehokkuuteen. Testit ajamalla oli helppo varmentua etten rikkonut mitään.

MyMinKeko vs PriorityQueue
Vertailussa ajoin eri määriä solmuja pinoon jonka jälkeen ne poistettiin sieltä. Tähän kulutettu aika mitattiin.

taulukko 1.

	10.000	100.000	1.000.000	10.000.000
MyMinKeko	0,017 s	0,025 s	0,386 s	5,836 s
PriorityQueue	0,018 s	0,026 s	0,453 s	6,281 s

Tuloksia katsellessa huomaa itse tehdyn toteutuksen olevan isommilla syötteillä jonkin verran Javan omaa toteutusta nopeampi. Parannus on kuitenkin korkeintaan 20% nopeampi ajoaika.

A* ja IDA* algoritmien vertailu
Tässä vertailussa kokeilin molempien algoritmien ajonopeutta neljällä eri sekoituksella, joiden ratkaisu oli molemmilla algoritmeilla mahdollista. A* algoritmilla hankalampien sekoituksien ratkaisua etsittäessä loppuu muisti.
Tässä testissä käytettiin kummassakin samaa Manhattan Distance -heuristiikkaa.

taulukko 2.

	47 moves	49 moves	58 moves	56 moves
	9 8 12 7 4 14 11 2 1 6 13 5 3 15 10	2 12 4 15 11 1 8 4 10 9 3 6 13 7 14	8 9 10 14 5 13 6 15 3 12 1 11 4 2 7	15 4 12 9 3 8 10 14 2 5 1 7 6 11 13
IDA*	215 ms	290 ms	788 ms	1425 ms
A*	746 ms	2923 ms	3707 ms	20128 ms

Testeistä selkeästi näkee kuinka A* jää jälkeen IDA*:estä sekoitusten muuttuessa vaikeammin ratkaistaviksi. Suurin syy tähän on muistin käytön kasvu. Yhä useampi haku joudutaan tekemään keskusmuistiin välimuistin sijaan.

Eri toteutusten vaikutus IDA*:n nopeuteen

Viimeisenä testinä pureuduin erilaisten heuristiikkatoteutusten tehokkuuteen. Nämä heuristiikat arvioiva palapelin tilanteen kulloisenkin etäisyyden ratkaisuun. Eli kuinka monta siirtoa vähintään tarvitaan ratkaisuun.
Käytetyt arviointi menetelmät ovat:
- ei heuristiikkaa, joka luo vertailukohdan muille.
- MD naiivi manhattan etäisyys lasketaan kokonaisuudessaan jokaisen siirron jälkeen.
- MD manhattan etäisyys niin että lasketaan vain siirron tuoma muutos edelliseen.
- MD+LC yhdistetty Manhattan etäisyys ja lineaari konflikti laskettuna kuten MD.
  LC tarkoittaa omalla rivillä tai sarakkeella olevaa kahta numeroa, jotka ovat väärässä järjestyksessä.

taulukko 3.

	28 moves	49 moves	61 moves	48 moves
	5 2 3 7 1 15 4 9 6 10 11 13 8 14 12	8 11 3 2 4 6 12 5 1 9 7 15 13 14 10	14 4 6 8 11 15 9 12 3 5 1 2 10 7 13	14 15 11 10 2 3 6 13 12 5 4 7 8 9 1
ei heuristiikkaa	194 000 ms	--	--	--
MD naiivi	25 ms	23 612 ms	63 703 ms	78 275 ms
MD	5 ms	2 251 ms	5 524 ms	6 610 ms
MD + LC	8 ms	413 ms	1 297 ms	3 624 ms

Tuloksista näkee kuinka suuri vaikutus on sekä heuristiikan toteutuksella että sen kyvyllä arvioida etäisyyttä. Naiivi toteutus manhattan etäisyyden laskemisesta on kahdessa viimeisessä tapauksessa yli kymmen kertaa hitaampi kuin fiksusti tehty. Lineaaristen konfliktien huomioon ottaminen nopeuttaa laskentaa jopa viisinkertaiseksi, mutta on hyvin

tapaus kohtaista kuinka paljon siitä on hyötyä. Taulukosta 3. voidaan tehdä myöskin havainto että ratkaisuun tarvittavien siirtojen määrä ei välttämättä kerro kuinka nopeasti ratkaisu löytyy. Kyseinen 61 siirtoa tarvitseva sekoitus ratkeaa selkeästi nopeammin kuin viimeinen 48 siirtoa tarvitseva sekoitus. Käytetty heuristiikka onnistuu tekemään huomattavan paremmat etäisyys arviot 61 siirtoa tarvitsevan sekoituksen kanssa.

Lopuksi

Olen tyytyväinen miten toimivaksi sain IDA*-algoritmin. Keskimääräisen ratkaisuun kuluva aika nykyisellä parhaalla toteutuksella on n. 1,6 sekuntia. Huonoimman tapauksen (80 siirtoa) ajoaika on vielä selvittämättä, mutta se voi olla n. 50 tuntia. Arvio perustuu testiin jossa ajoin 80 siirtoa tarvittavaa sekoitusta ja kului hieman vajaa puoli tuntia selvittää että ratkaisu ei löydy 76 siirrolla. Seuraavan rajan 78 siirtoa selvittäminen vie noin kymmen kertaisen ajan. Onneksi näin huonot tapaukset ovat todella harvinaisia luokkaa yksi tapaus sadasta miljardista.