Petri Mäkinen
014027765

Tiralabra testausdokumentti

Toteutusta on testattu Junit yksikkö testeillä. Suurin huomio on ollut tietorakenteiden ja algoritmien oikeellisuuden testaamisessa. Tärkeitä operaatioita on testattu erilaisilla syötteillä, jotta on nähty että ne toimivat oikein. Esimerkiksi bfs ja a –tähti algoritmien toimintaa on testattu antamalla niille esimerkkiverkko ja katsomalla että ne palauttavat oikean polun. Tälläistä testausta on tehty myös graafisen käyttöliittymän kautta erilaisilla verkoilla jonkin verran. Testit testaavat esimerkkiverkolle myöskin että algoritmit käyvät niitä ajettaessa niissä solmuissa joissa niiden algoritmien perustoimintaidean mukaan pitääkin käydä.

Myös algoritimen nopeutta on testattu yksikkötesteillä antamalla algoritmeille suuria verkkoja, joista niiden pitää löytää lyhin polku. Algoritmeille annettiin ensin verkkoja jotka olivat ”täysiä”. Verkko oli siis n*n matriisi (n = matriisin sivun pituus), jossa jokaisella solmulla siis oli neljä naapuria (paitsi laidoilla vähemmän). Lähtöpiste asetettiin vasempaan yläkulmaan ja maalisolmu taas oli verkon viimeinen solmu, joka löytyy siis verkon oikeasta alakulmasta. Näillä verkoilla odotetusti BFS on huomattavasti A –tähteä hitaampi algoritmi. Alla tuloksia joista tämä nähdään selvästi:

				Läpikäytyjä solmuja	
	A*	**BFS**	**Solmujen määrä**	**A***	**BFS**
Aika	1ms.	31ms.	160k	799	160k
	1ms.	63ms.	n. 320k	1131	n. 320k
	2ms.	132ms.	640k	1599	640k
	4ms.	305ms.	n. 1280k	2261	n. 1280k
	8ms.	597ms.	n. 2560k	3197	n. 2560k

A-tähden ylivoimaisuus tällaisilla triviaaleilla verkoilla johtuu siitä että parhaaksi arvioitu solmu on aina myös oikeasti paras eikä turhia solmuja tarvitse käydä läpi. A tähti käykin solmuja läpi näillä verkoilla tarkalleen lyhimmässä polussa olevien solmujen määrän verran. Toisaalta koska lähtösolmu oli vastakkaisessa kulmassa kuin maalisolmu, joutuu BFS näillä verkoilla vierailemaan jokaisessa solmussa ennenkuin se vihdoin päätyy maalisolmuun.

Koska A –tähdellä oli tällaisissa verkoissa selvä etulyöntiasema, testattiin algoritmejä myös satunnaisesti generoiduilla verkoilla. Nämä generoitiin niin että jokaisen solmun kohdalla oli neljäsosan todennäköisyys että siihen kohtaan ei tulisi solmua. Laittamalla tällainen määrä ”seiniä” verkkoon niistä tulee usein niin monimutkaisia että a-tähti menettää etulyöntiasemaansa, mutta ”seiniä” ei kuitenkaan ole niin paljoa että yleensä jonkinlainen ratkaisu löytyy. Alla tuloksia testiajosta tällaisilla satunnaisilla verkoilla:

				Läpikäytyjä solmuja	
	A*	**BFS**	**Matriisin koko(myös seinät)**	**A***	**BFS**
Aika	15ms.	10ms.	n. 80k	n. 10k	n.59.5k
	13ms.	23ms.	160k	n. 9.7k	n. 119k
	20ms.	49ms.	n. 320k	n. 14.5k	n.238k

Vaikka BFS siis joutuu edelleen käymään kaikki solmut läpi (lähtösolmu ja maalisolmu olivat edelleen vastakkaisissa kulmissa), on sen nopeus huomattavasti lähempänä ja se on joskus jopa a –tähteä nopeampi. Tämä johtuu siitä, että vaikka a –tähti käy huomattavasti vähemmän solmuja läpi, se joutuu usein järjestämään kekoa, jossa solmut ovat.

Petri Mäkinen
014027765

Testasin toteutustani myös sellaista toteutusta vastaan, joka oli muutoin sama, mutta käytti hyväkseen javan valmiita tietorakenteita. Näissä kuitenkin kumpikin toteutus luo omat satunnaiset verkkonsa, joten nuo testit eivät ole vertailukelpoisia. Sensijaan täysillä verkoilla suoritetuista testeistä nähdään että oma toteutukseni on BFS:n tapauksessa hieman (noin 10-15%) javan työkaluja hitaampi. Ilmeisesti linkitetty lista ei ole aivan niin tehokas kuin se voisi olla. Toisaalta A -tähti toimii jopa javan kalustolla toteutettua nopeammin, mutta näillä täysillä verkoilla A –tähden yksi tärkeimmistä osista –minimikeko ei varsinaisesti joudu töihin. Pääpaino työssä oli BFS ja A-tähti algoritmien keskinäisissä vertailuissa, joten tässä javan valmiiseen kalustoon vertaamisessa päätettiin olla menemättä tämän syvemmälle.

	Oma toteutus:	**Javan tietorakenteet:**	**Oma:**	**Java:**	
	A*		**BFS**		**Solmujen määrä**
Aika	1ms.	0ms.	31ms.	27ms.	160k
	1ms.	1ms.	66ms.	57ms.	n. 320k
	2ms.	2ms.	144ms.	120ms.	640k
	4ms.	3ms.	300ms.	268ms.	n. 1280k
	7ms.	7ms.	633ms.	558ms.	n. 2560k
	17ms.	24ms.			9000k

Näistä ja edellisistä tuloksista näkyy myös hyvin että kun verkossa ei ole seiniä (eli kaarien määrä on suorassa suhteesssa solmujen määrään, eikä A –tähden tarvitse järjestellä solmuja), näyttää kummankin algoritmin aikavaativuus kasvavan lineaarisesti suhteessa solmujen määrään. Tuplakokoinen verkko tuottaa aina noin tuplasti kauemmin kestävän haun.

Yksikkötesteistä kannattaa huomata se että java -sanalla alkavat testit testaavat javan valmiilla tietorakenteilla tehtyä toteutusta ja muut testit omaa toteutustani. Omaa toteutustani testaavissa luokissa voi joissain tapauksissa olla enemmän testejä jos ne on tehty sen jälkeen kun testaus jaettiin kahtia.