# Toteutusdokumentti

Ohjelmassa on seuraavat luokat: Astar, Logic, MapMaker, Node, NodeComparator ja PriorityQueue.

Astar.java käynnistää ohjelman ja etsii halutun reitin.

Logic.java sisältää ohjelman toiminta logiikan. Kyseisessä luokassa etsitään reitti. Reitti etsitään siten, että tarkastellaan aina nykyisestä kohdasta joka puolella ja katsotaan mikä on edullisin suunta edetä.

MapMaker.java tekee automaattisesti kartan, jossa on #-merkkejä, jotka tarkoittavat seinää ja .-merkkejä jotka tarkoittavat vapaata reittiä kartalla. Luotu kartta on char-tyyppinen matriisi, jonka korkeus ja leveys annetaan konstruktorille samoin kuin seinien prosenttimäärä. Seinät generoidaan randomilla, jolloin on mahdollista, että seinät ovat päällekkäin. Eli 100% seiniä ei tarkoita kuitenkaan, että kartta olisi kokonaan täynnä seiniä.

Node.java on luokka, jossa alustetaan nodet ja muutetaan niiden tietoja. Node tietää paikkansa x- ja y-koordinaattien avulla. Lisäksi node tietää onko nodessa käyty ja onko node maalinode.

NodeComparator.java vertailee kahta nodea ja kertoo kummalla niistä on pienempi arvo.

PriorityQueue.java on itse toteutettu prioriteettijono. Jonosta otetaan aina pois node, jonka arvo on pienin. Prioriteettijonossa on seuraavat operaatiot: insert: joka asettaa jonoon uuden noden, sekä kutsuu tämän jälkeen heapify-operaatiota. Heapify: vertaa nodea sen vanhempaan ja jos noden arvo on pienempi siirtää nodea ylöspäin keossa. Eli käytännössä pitää keon kunnossa. ReturnMin: palauttaa miniminoden, mutta ei poista sitä keosta. ExtractMin: palauttaa ja poistaa keosta miniminoden, lisäksi kutsuu heapifyDown-metodia. HeapifyDown: Vertaa nodea sen lapsiin ja siirtää nodea alaspäin jos lapset ovat arvoltaan pienempiä kuin node.

A* on toteutettu manhattan etäisyydellä niin, että on mahdollista myös liikkua sivuilmansuuntiin.

Algoritmin tilavaativuus on O(n). Aikavaatimus on O(log n).

Tein suorituskykytestejä sekä niin että maalinode liikkui että myös niin ettei maalinode liikkunut. Suorituskykytestit tein niin, että otin ajat ylös ja laskin niistä keskiarvon. Pienemmistä kartoista on enemmän testidataa ja isommista vähemmän. Kaikissa suoritusaikatesteissä maalinode on aluksi vastakkaisessa kulmassa kuin etsijä. Suoritusaikatesteistä on myös graafi.

Maalinode liikkui joka viides siirto yhden liikkeen verran:

| Kartankoko: | Suoritukset: | Keskiarvo: |
|---|---|---|
| 10x10 | 100 | 0.000055345s |
| 100x100 | 100 | 0.079705005s |
| 250x250 | 10 | 5,190366773s |
| 500x500 | 10 | 36,80660202s |

Maalinode ei liikkunut ollenkaan:

| Kartankoko: | Suoritukset: | Keskiarvo: |
|---|---|---|
| 10x10 | 100 | 0.000099538s |
| 100x100 | 100 | 0.075614723s |
| 250x250 | 100 | 3.486151314s |
| 500x500 | 20 | 28.44920281s |

Parannettavaa olisi ohjelman käyttämisen puolella. Joku yksinkertainen graafinen käyttöliittymä voisi olla kiva. Tällä hetkellä ohjelma kysyy käyttäjältä vain kartan korkeuden ja leveyden. Ohjelma lisäksi kaatuu jos ei anna korkeutta tai leveyttä kokonaislukuna mikä on huonosti toteutettu.

Lisäksi maalinodea voisi lähteä karkuun etsijää. Siten, että se katsoisi mistä suunnasta etsijä on tulossa ja yrittäisi lähteä sitten aina mahdollisimman kauas etsijästä.