Labyrintin ratkaisualgoritmit

Harjoitustyö

Joni Korpihalkola

Joonas Mankinen

Niko Poutanen

Tietorakenteet ja algoritmit

3/2017

Tieto- ja Viestintätekniikka

Tekniikan ala

Sisällys

[1 Johdanto 2](#_Toc478502309)

[2 Algoritmien esittely 2](#_Toc478502310)

[2.1 Wall Follower algoritmi 2](#_Toc478502311)

[2.2 Random Mouse algoritmi 5](#_Toc478502312)

[2.3 Rekursiivinen algoritmi 5](#_Toc478502313)

[3 Testiaineisto ja testaus 6](#_Toc478502314)

[3.1 Testiaineiston esittely 6](#_Toc478502315)

[3.2 Tulokset 7](#_Toc478502316)

[4 Yhteenveto 7](#_Toc478502317)

[Lähteet 10](#_Toc478502318)

[Liitteet 10](#_Toc478502319)

# Johdanto

Harjoitustyömme aiheena oli tutkia labyrintin ratkaisualgoritmeja ja vertailla itsetehtyjä Wall Follower- ja Random Mouse - algoritmeja sekä valmista rekursiivista ratkaisualgoritmia toisiinsa. Vertasimme algoritmien nopeutta ja toimintatapaa labyrinteissä, jotka teimme eri kokoisina kaksiulotteisina taulukkoina.

# Algoritmien esittely

## Wall Follower algoritmi

Suunnittelimme ja toteutimme Wall Follower algoritmin itse, nimensä mukaisesti seuraamaan oikean puoleista seinää erilaisin säännöin, kunnes löytää tiensä maaliin. Algoritmin toteutus löytyy WallFollowerAlgorith.cs tiedostosta (Liite 1).

Käytetyissä labyrinteissä arvo 1 vastaa tyhjää vapaata tilaa ja arvo 0 vastaa seinää, aloituspiste merkataan arvolla 3 ja lopetuspiste arvolla 4.

Ensimmäisenä Wall Follower algoritmi katsoo aloituspisteen indeksipaikan perusteella, minne suuntaan se aloittaa arvojen vertaamisen.

* Jos aloituspaikka on taulukon ylimmällä laidalla, verrataan ensimmäisenä alaspäin
  + {0,0,**3**,0,0},
  + {0,0,1,0,0},
  + {0,0,1,0,0},
  + {0,0,1,0,0},
  + {0,0,4,0,0},
* Jos aloituspaikka on taulukon alimmalla laidalla, verrataan ensimmäisenä ylöspäin
  + {0,0,4,0,0},
  + {0,0,1,0,0},
  + {0,0,1,0,0},
  + {0,0,1,0,0},
  + {0,0,**3**,0,0},
* Jos aloituspaikka on taulukon vasemmalla laidalla, verrataan ensimmäisenä oikealle
  + {0,0,0,0,0},
  + {0,0,0,0,0},
  + {**3**,1,1,1,4},
  + {0,0,0,0,0},
  + {0,0,0,0,0},
* Jos aloituspaikka on taulukon oikealla laidalla, verrataan ensimmäisenä vasemmalle.
  + {0,0,0,0,0},
  + {0,0,0,0,0},
  + {4,1,1,1,**3**},
  + {0,0,0,0,0},

Algoritmi toimii samalla periaatteella koko ratkaisun ajan, vertailemalla nykyistä indeksipaikkaa aikaisempaan ja selvittää liikkumansa suunnan, ja sitä kautta valitsee ensimmäisen vertaussuunnan sääntöjen perusteella, jonne katsoo kulkureitin löytääkseen.

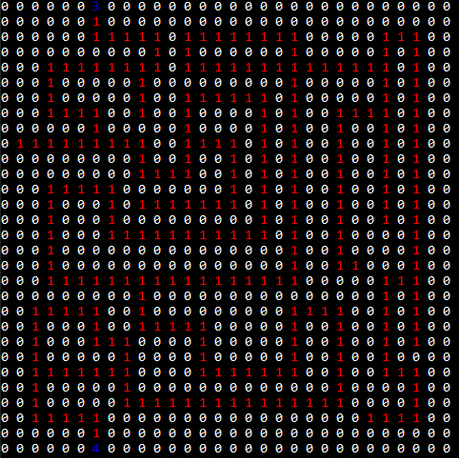
* Jos liikuttiin alaspäin, aloitetaan vertaaminen vasemmalta
* Jos liikuttiin ylöspäin, aloitetaan vertaaminen oikealta
* Jos liikuttiin oikealle, aloitetaan vertaaminen alhaalta
* Jos liikuttiin vasemmalle, aloitetaan vertaaminen ylhäältä.

Jos ensimmäinen vertaussuunta ei tuota tulosta, eli siellä on seinä, verrataan seuraavana suuntaa, joka on vastapäivään äskeisestä vertaussuunnasta.

* Verrataan vasemmalle ja jos siellä on seinä, verrataan alaspäin, jos siellä on seinä, verrataan oikealle ja niin edespäin.

Algoritmi etsii näin vertaillen arvoa yksi, ja kun se löytää arvon yksi se ”siirtyy” siihen indeksipaikkaan ja muuttaa ykkösen arvoksi kaksi. Tämä toistetaan, kunnes ulospääsy, eli arvo neljä, on löydetty.

Jos algoritmi joutuu umpikujaan, eli ympärillä olevat luvut ovat nollia ja kakkosia, algoritmi liikkuu arvoja kaksi pitkin takaisin päin, kunnes löytää taas arvon yksi ja jatkaa samaan tapaan vertaamista kuin aikaisemmin etsien seuraavaa arvoa yksi.



Kuva 1. Labyrintti ennen algoritmin ajoa



Kuva 2. Labyrintti algoritmin ajon jälkeen

Yllä esimerkkitulostus (Kuva 2), kun ajamme ohjelman Wall Follower- algoritmin kanssa, jossa kakkoset kuvastavat kuljettua reittiä ja ykköset vapaata tilaa, jossa ei olla käyty.

Kuten kuvasta huomataan, algoritmi toimii annetulla tavalla ja se jättää käymättä ylimääräisellä kiertoreitillä, koska se vertaa risteyskohdassa, josta kiertoreitti alkaa, ensimmäisenä alaspäin annetun säännön mukaan ja näin ollen löytää oikean suunnan ja siirtyy sinne.



Kuva 3. Labyrintti algoritmin ajon jälkeen, seurattu seinä värjätty

Yllä tulostus (Kuva 3), jossa verratut indeksit on värjätty keltaisella, jotta näemme mitä seiniä algoritmi on seurannut.

## Random Mouse algoritmi

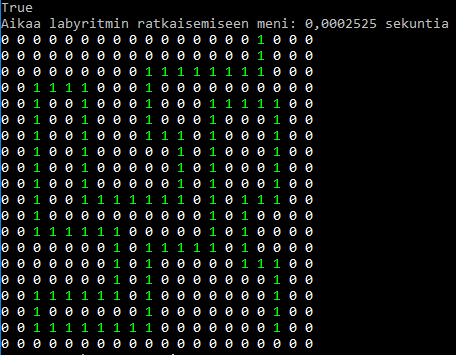
Random Mouse -algoritmi yrittää löytää tiensä ulos kääntymällä risteyksissä satunnaiseen suuntaan ja jatkaa kyseiseen suuntaan, kunnes tulee taas risteys vastaan. Algoritmi ei merkitse kulkemaansa reittiä, joten suoritusnopeus hidastuu huomattavasti, mitä vaikeampia ja isompia labyrintit ovat.

Random Mouse algoritmi aloittaa etenemisen samalla logiikalla, kuin Wall Follower algoritmin aloitus. Wall Follower - algoritmin tapaan Random Mouselle pitää kertoa aloituspisteen rivi- ja sarakekoordinaatti.

Risteyksen tullessa vastaan, algoritmi kutsuu aliohjelmaa generateRandom, joka arpoo uuden suunnan. Arvottavista suunnista poistetaan se suunta, josta risteykseen on tultu, paitsi jos algoritmi ajautuu umpikujaan, jolloin liikutaan takaisinpäin, kunnes löydetään taas risteys. Algoritmin toteutus löytyy RandomMouseAlgorithm.cs tiedostosta (Liite 1).

## Rekursiivinen algoritmi

Rekursiivinen algoritmi liikkuu labyrintissä kutsumalla itseään aina uudella arvolla. Esimerkiksi jos vasemmalle pystyy liikkumaan, algoritmi kutsuu itseään recursiveSolve(x - 1, y) funktiolla. Algoritmi laittaa käydyt pisteet wasHere-nimiseen matriisiin, ja tallentaa oikean reitin correctPath-nimiseen matriisiin. Algoritmi tulostaa alla olevan kuvan mukaisen tulosteen (Kuvio 4).



Kuvio 4. Algoritmin ratkaisema reitti 20x20 labyrintissä, ulospääsy oikealla alhaalla

Algoritmi tarvitsee aloituspisteen ja lopetuspisteen koordinaatit. Algoritmin toteutus löytyy RecursiveAlgorithm.cs tiedostosta (Liite 1).

# Testiaineisto ja testaus

## Testiaineiston esittely

Testilabyrinttejä on yhteensä tehty seitsemän kappaletta, ja ne on määritelty Program.cs – tiedoston alussa. Kuvat labyrinteistä löytyvät liitteistä 2-7, joissa sinisellä merkitty kolmonen on alkupiste ja sinisellä merkitty nelonen loppupiste. Random mouse –algoritmiä testattiin vain matriiseilla 10x15, 20x20 ja 50x50, koska muissa labyrinteissä algoritmi jäi jumiin labyrinttiin. Myös rekursiivinen algoritmi kaatui index out of bounds – erroriin 10x15 labyrintissä, joten se jätettiin tuloksista pois.

Testejä suoritettiin viisi kappaletta, ja algoritmien suoritusajoista otettiin keskiarvo.

## Tulokset

Alla Random Mouse – algoritmin ajat sekunteina (Taulukko 1).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Koko** | **Keskiarvo** | **1** | **2** | **3** | **4** | **5** |
| 10x15 | 0,05639482 | 0,057375 | 0,055403 | 0,0837029 | 0,003602 | 0,081891 |
| 20x20 | 0,12950786 | 0,062816 | 0,065528 | 0,3035007 | 0,0385766 | 0,177118 |
| 50x50 | 24,75259418 | 5,244948 | 1,707933 | 72,85983 | 42,9918431 | 5,510091 |

Taulukko 1. Random Mouse - algoritmin tulokset.

Alla Wall Follower – algoritmin ajat millisekunteina (Taulukko 2).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Koko** | **Keskiarvo** | **1** | **2** | **3** | **4** | **5** |
| 10x10 | 0,88026 | 0,932 | 0,8043 | 0,9729 | 0,8878 | 0,8043 |
| 10x15 | 0,8694 | 0,7992 | 0,7881 | 0,877 | 0,7815 | 1,1012 |
| 20x20 | 0,87638 | 0,81 | 0,7767 | 1,1039 | 0,911 | 0,7803 |
| 30x30 | 0,8969 | 0,8034 | 0,8947 | 1,1694 | 0,8217 | 0,7953 |
| 40x40 | 0,9879 | 1,3518 | 0,81 | 1,1318 | 0,8572 | 0,7887 |
| 50x50 | 0,84038 | 0,8344 | 0,8623 | 0,8476 | 0,8407 | 0,8169 |
| 100x100 | 0,91254 | 0,9693 | 0,8743 | 0,8632 | 0,9657 | 0,8902 |

Taulukko 2. Wall Follower - algoritmin tulokset

Alla rekursiivisen algoritmin ajat millisekunteina (Taulukko 3).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Koko** | **Keskiarvo** | **1** | **2** | **3** | **4** | **5** |
| 10x10 | 0,2438 | 0,2178 | 0,271 | 0,2566 | 0,23 | 0,2436 |
| 20x20 | 0,23818 | 0,2184 | 0,2277 | 0,2686 | 0,2572 | 0,219 |
| 30x30 | 0,23872 | 0,225 | 0,2608 | 0,2581 | 0,2277 | 0,222 |
| 40x40 | 0,26518 | 0,2502 | 0,2325 | 0,3142 | 0,2433 | 0,2857 |
| 50x50 | 0,29044 | 0,2875 | 0,2875 | 0,2794 | 0,2857 | 0,3121 |
| 100x100 | 0,33858 | 0,3419 | 0,3242 | 0,3181 | 0,3251 | 0,3836 |

Taulukko 3. Rekursiivisen algoritmin tulokset

# Yhteenveto

Tuloksista luotiin kaaviot, joista näkyy, miten suoritusajan keskiarvo riippuu labyrintin koosta. Random Mouse – algoritmin kaaviosta nähdään, että suoritusaika nousee huimasti, kun labyrintit vaikeutuvat ja kasvavat isommaksi (Kuvio 5).

Kuvio 5. Random Mouse - algoritmin suoritusajan kaavio

Algoritmissa on myös erittäin suuri varianssi, sillä 50x50 matriisin pienin suoritusaika on n. 1,7 sekuntia, kun taas suurin aika on enemmän kuin yksi minuutti.

Wall Follower - algoritmin suoritusajan keskiarvossa taas tapahtui erikoinen ilmiö. 50x50 labyrintin suoritusajan keskiarvo on jonkin verran pienempi kuin edellisen testilabyrintin, 40x40:n (Kuvio 6).

Kuvio 6. Wall Follower - algoritmin suoritusajan kaavio

Tämä näyttäisi todistavan sen, että algoritmin suoritusaika riippuu enemmän labyrintin vaikeudesta, kuin sen koosta. Suoritusaikojen erot ovat kuitenkin aika pieniä, sillä suurin ero on vain noin 0,1 millisekuntia.

Rekursiivisen algoritmin suoritusajan keskiarvo pysyy ensimmäiset kolme testilabyrinttia melkein samana, kunnes aika alkaa pikkuhiljaa kasvaa (Kuvio 7).

Kuvio 7. Recursive algoritmin suoritusajan kaavio

Vaikka aika kasvaa jonkin verran labyrinttien kasvaessa koossa, on 10x10 ja 100x100 labyrinteillä suoritusajassa vain n. 0,1 millisekuntia eroa.

Algoritmeista nopein oli valmis rekursiivinen algoritmi, joka oli keskimäärin 0,6 millisekuntia nopeampi kaikissa labyrinteissä kuin meidän itse tehty Wall Follower - algoritmi. Itse tehdystä algoritmista löytyy aika monta vertailulausetta, joita luultavasti voisi karsia tai yhdistää. Wall Follower - algoritmi ei sentään ollut yhtä hidas kuin Random Mouse – algoritmi, muuten algoritmin toteutus olisi pitänyt aloittaa aivan alusta.

Lähteet

<https://en.wikipedia.org/wiki/Maze_solving_algorithm> Recursive Algorithm Sample Code 25.3.2017

Liitteet

Liite 1. LabyrinttiAlgoritmit.zip, johon pakattuna ohjelmointitoteutus C#:lla.

Liite 2. Labyrintti10x10.PNG

Liite 3. Labyrintti20x20.PNG

Liite 4. Labyrintti30x30.PNG

Liite 5. Labyrintti40x40.PNG

Liite 6. Labyrintti50x50.PNG

Liite 7. Labyrintti100x100.PNG