Haaga-Helia
University of Applied Sciences

Ohjelmistokehityksen teknologioita - *S*eminaarityö

Labyrintin generoiva algoritmi Pythonilla

Seminaari 3 – Tietorakenteet ja algoritmit

Tatu Paukkeri

3.5.2023

Sisältö

Tiivistelmä

Anna halutessasi **lyhyt** mutta kattava kuva tekemästäsi aiheesta. Noin 10-20 riviä tekstiä esim. kahdessa tai kolmessa kappaleessa. **Tavoite**/motivaatio työn takana, **käytetyt menetelmät**/otetut askeleet ja lopuksi tärkeimmät **tulokset**.

1 Johdanto

Idea tämän työn takana, oli halu tutustua paremmin algoritmeihin ja etenkin labyrinttejä generoiviin algoritmeihin. Valitsin toteutettavaksi algoritmiksi satunnaisen syvyyshaun (engl. *randomized depth-first search* tai *recursive backtracker*). Tavoitteena oli toteuttaa toimiva algoritmi ja myös visualisoida tuotokset jollain tavalla.

Työn vaiheet:

1. *Tutustuminen labyrinttien generointi algoritmeihin ja yhden valitseminen*
2. *Valitun algoritmin tarkempi ymmärtäminen ja opettelu*
3. *Toteuttaa algoritmi valitulla ohjelmointikielellä*
4. *Visualisoida labyrintti*

2 Käytetyt tekniikat

2.1 Algoritmi

Valitsin algoritmiksi satunnaisen syvyyshaun. Algoritmi toimii lyhyesti selitettynä näin:

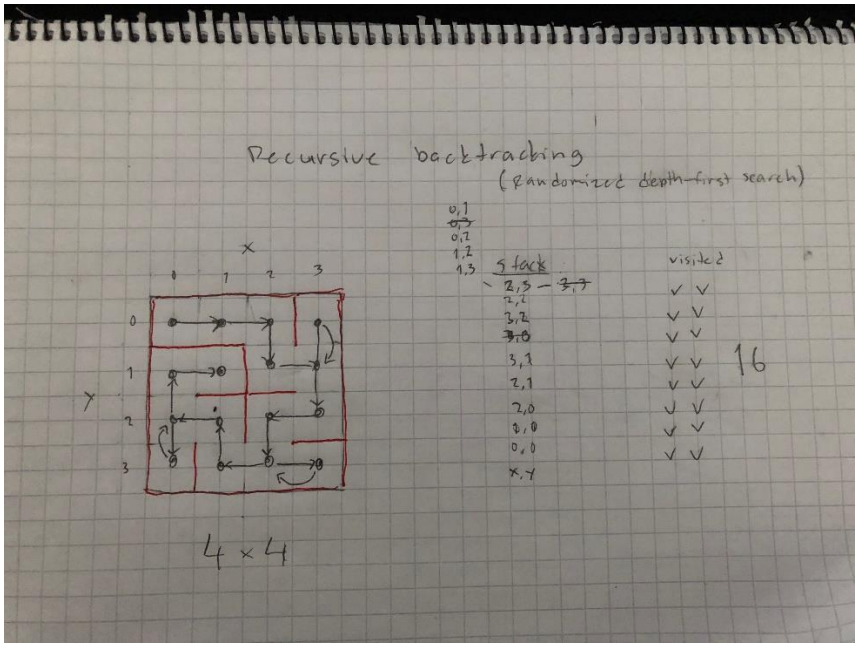
1. *Valitse aloituspiste tyhjästä ruudukosta.*
2. *Valitse satunnaisesti suunta (ilmansuunta), johon liikutaan ja liiku siihen, mutta vain jos siinä ei ole vielä käyty.*
3. *Jos kaikissa viereisissä pisteissä on käyty, palataan takaisin viimeisimpään pisteeseen, jolla on vielä vierailemattomia naapuripisteitä, ja toistetaan kohta 2.*
4. *Näitä kohtia toistetaan, kunnes kaikissa pisteissä on käyty.*

Algoritmista löytyy paljon erilaisia toteutuksia eri ohjelmointikielillä, ja tutustuin useisiin ennen omaa toteutusta. Yritin kuitenkin mahdollisimman paljon tehdä itse, mutta ongelmatilanteissa joutui välillä etsimään apua.

2.2 Kynä ja paperi

Valittuani algoritmiksi satunnaisen syvyyshaun, ryhdyin tutustumaan siihen tarkemmin. Luin aiheesta ja katsoin YouTube-videoita, mutta halusin vielä varmistaa ymmärtäväni algoritmin

kokeilemalla sitä käsin. Tein pienen labyrintin kynällä ja paperilla algoritmia noudattaen (Kuva 1).



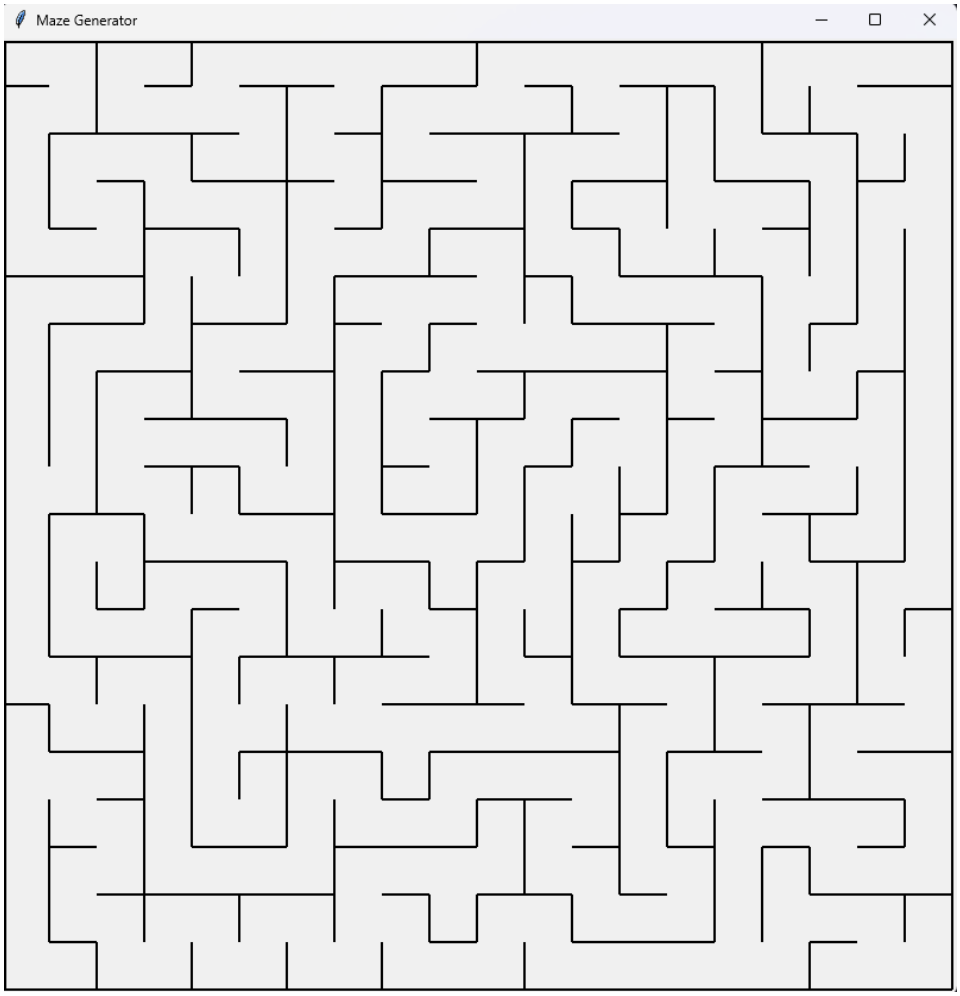
Kuva 1 Algoritmin kokeilu käsin.

2.3 Python

Päätin toteuttaa algoritmin Pythonilla. Kun aloin työn alussa tutkimaan erilaisia labyrintteihin liittyviä algoritmeja, en ollut aivan varma toteutanko ratkaisu vai generointi algoritmin. Kikkailin alkuun muun muassa hieman pyamaze-kirjastolla, ja mietin mahdollista rarkaisualgoritmin visualisointia.

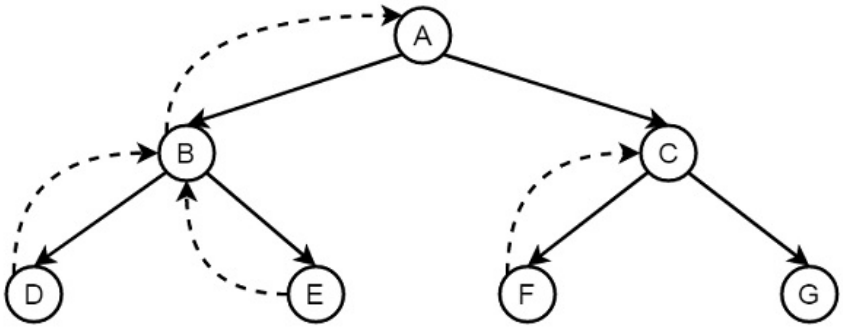
Päädyin kuitenkin valitsemaan aiheeksi labyrintin generoivan algoritmin ja pysyin myös Pythonissa. Python on jo ennestään tuttu Python-kurssilta, mutta oli kerennyt hieman ehkä unohtua.

Visualisoinnissa käytin Pythonin tkinter-kirjastoa. Tästä ei ollut aiempaa kokemusta ja jouduinkin katsomaan hieman mallia eri paikoista saadakseni visualisoinnin toimimaan. Lähdin myös ensin kokeilemaan visualisointia pygamella, mutta siinä oli hieman vaikeuksia.



Kuva 2 Kuvakaappaus valmiista labyrintista, videolla ei näkynyt kokonaan.

3 Algoritmin vuokaavio



Kuva 3 Backtracking algoritmi (Tutorialspoint).

4 Yhteenveto

Työn tekeminen oli mukava kokemus. Opin paljon uutta algoritmeista ja varsinkin niiden käytöstä labyrinttien luomisessa. Toteuttavaksi valitsemani algoritmi ei ollut ehkä kaikkein monimutkaisin, mutta oli siinäkin aivan riittävästi tekemistä.

Aikaa meni aluksi useita tunteja pelkästään aiheen opiskeluun, ennen kuin aloitin koodipuolen toteutuksen. Aikaa koko työhön kului luultavasti noin 18-22 tuntia.

Lopputulokseen olen tyytyväinen. Algoritmi toimii, ja myös labyrintin visualisointi onnistui. Vaikka koko työtä en aivan ilman mallia muiden tuotoksista onnistunut tekemään, tunnen silti oppineeni paljon. Algoritmien lisäksi myös Pyhtonista tuli lisää kokemusta.

Aiheesta saisi irti vielä paljon lisää. Käyttöliittymästä voisi tehdä hienomman näköisen enemmillä toiminnallisuuksilla. Myös labyrintin ratkaisu algoritmin lisääminen ja sen visualisointi olisi hyvä lisä. Luomisen jälkeen voisi siis ratkaista labyrintin haluamallaan algoritmilla.

Demovideo

Github

Lähdeluettelo

Tutorialspoint. Backtracking. Luettavissa: https://www.tutorialspoint.com/prolog/prolog_backtracking.htm.