Dijkstran algoritmi

Tietokannat ja algoritmit

Karttunen Veeti

Liuhanen Nicolas

Lyytinen Aaro

Harjoittelutyö

Marraskuu 2016

Tekniikan ja liikenteen ala

Tieto- ja viestintätekniikan tutkinto-ohjelma

Sisältö

[1 Johdanto 2](#_Toc467509378)

[2 Algoritmin tehokkuus ja erityispiirteet 2](#_Toc467509379)

[3 Toteutus 3](#_Toc467509380)

[3.1 Johdanto 3](#_Toc467509381)

[3.2 Pseudokoodi 3](#_Toc467509382)

[3.3 Satunnaisen verkon luonti 4](#_Toc467509383)

[3.4 Dijkstran algoritmi 5](#_Toc467509384)

[4 Testaus 6](#_Toc467509385)

[5 Yhteenveto 7](#_Toc467509386)

[6 Lähteet 8](#_Toc467509387)

# Johdanto

Tehtävänä oli luoda dijkstran algoritmi ja tutkia sen tehokkuutta sekä toimivuutta eri rasitetilanteissa. Dijkstran algoritmi valittiin sen mielenkiintoisuuden ja reaalimaailman korreloitumisen vuoksi.

# Algoritmin tehokkuus ja erityispiirteet

Dijkstran algoritmi on verkko algoritmi, jonka tarkoituksena on löytää lyhin polku lähtöpisteestä muihin verkon solmuihin. Kyseinen algoritmi on Edsger Dijkstran kehittämä, ja se on julkaistu vuonna 1959. Sitä voidaan hyödyntää esimerkiksi logistiikassa, navigointilaitteissa ja pelien tekoälyn ohjaamisessa.

Algoritmi ei ole heuristinen, joka etsii riittävän lyhyen polun mahdollisimman lyhyessä ajassa. Kyseessä on ahne algoritmi (greedy algorithm), joka etsii aina lyhimmän polun käyden läpi jokaisen verkon solmun. Ahne algoritmi ei ole aina nopein mahdollinen, mutta se antaa aina tarkan vastauksen lyhimmästä polusta.

Dijkstran algoritmin aikakompleksisuus:

Worst case scenario:

# Toteutus

## Johdanto

Algoritmi kirjoitettiin C# ohjelmointikielellä, visual studio kehitysympäristössä. Ohjelma sisältää itse algoritmin, satunnaisen verkon generointi funktion, sekä ajanmittaus osion. Eri suuruisia satunnaisia verkkoja luomalla pystyttiin kirjaamaan mittaustuloksia algoritmin tehokkuudesta.

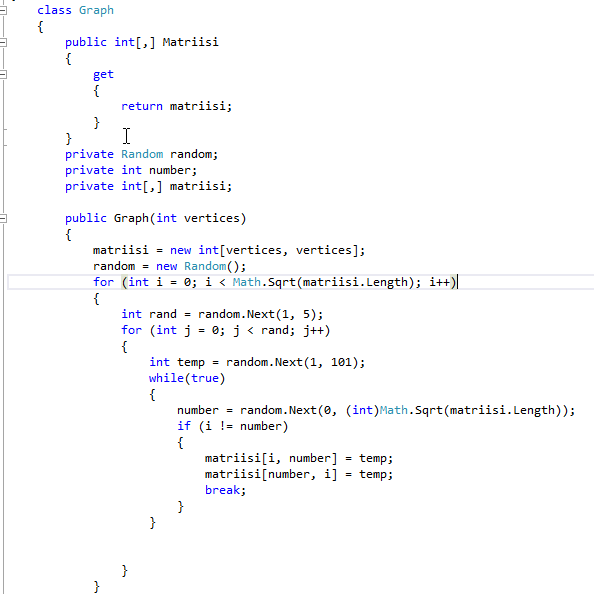
## Pseudokoodi

Algoritmin pseudokoodi:

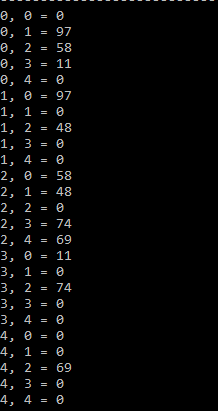
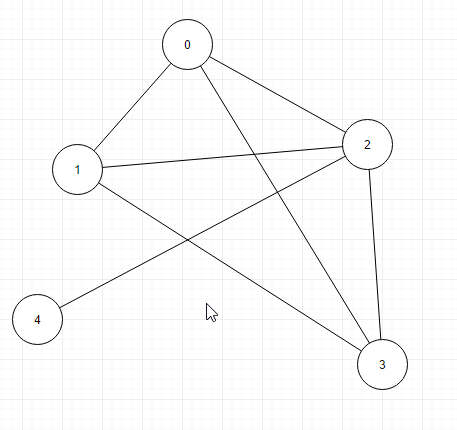
1. Aseta lähtöpiste ja sen etäisyys nollaksi
2. Aseta loput etäisyydet äärettömiksi
3. Mittaa etäisyydet naapurisolmuihin
4. Siirry lähimpään naapurisolmuun
5. Siirry kohtaan 3 jos jäljellä on vierailemattomia solmuja

## Satunnaisen verkon luonti

C# ohjelmakielellä luotiin funktio (kuva 1), joka luo satunnaisen matriisitaulukon (kuva 3). Matriisitaulukolle asetettiin ehdot: 1-N määrän verran alkioita, joista muodostuu yhtenäinen verkko. Verkon solmuille asetettiin 1-4 linkkiä. Matriisitaulukon havainnollistamiseksi ja funktion toimivuuden toteamiseksi (kuva 2), siitä piirrettiin verkko.



Kuva 1



Kuva 2

Kuva 3

## Dijkstran algoritmi

Dokumentaation epäselkeyden minimoimiseksi, sekä suurien kuvien välttämiseksi, alle on liitetty linkki dijkstran algoritmin lähdekoodista. Koodi on kommentoitu ja eri vaiheet ovat selvästi nähtävissä (Linkki 1)

# Testaus

Testaus suoritettiin tietokoneella, jossa oli Intel Core i5-4670K (3.7 GHz) suoritin ja 8GB DDR3 (1600 MHz) fyysistä muistia.

Algoritmin tehokkuuden tutkimiseksi, ohjelman annettiin laskea lyhimmät reitit yhdestä pisteestä. Tähän kulunut aika mitattiin millisekunteina microsoftin diagnostics -kirjaston stopwatch oliolla. Mittausnäytteitä otettiin kahdeksan (8) kappaletta jokaista erikokoista verkkoa kohden, joista laskettiin keskiarvot. Keskiarvot taulukoitiin alla olevaan taulukkoon (Taulukko 1).

Taulukko 1

Dijkstran algorimit aikakompleksisuus voidaan todeta paikkaansapitäväksi ylläolevan taulukon perusteella (Taulukko 1). Taulukko seuraa lineaarisesti logarimista kasvua.

# Yhteenveto

Yhteenvetona voidaan todeta, että kahden saman mittaisen särmän vertailu on haasteellista ohjelmoida. Dijkstran algoritmit toimivat nopeasti pienissä verkoissa ja takaavat parhaat mahdolliset lopputulokset.

Lähtösolmusta verkon jokaiseen solmuun dijkstran algoritmin nopeus on suurissakin verkoissa suhteellisen nopea. Kaikista verkon solmuista jokaiseen verkon solmuun ajan tarve kasvoo exponentiaalisesti verkon kokoon nähden.

Projekti valmistui ajallaan, eikä ylitsepääsemättömiä haasteita ilmennyt missään vaiheessa. Koodin rakentaminen vaati suurimman osan ajasta (n. 40 tuntia). Koodi kirjoittettiin alusta asti itse, eikä sitä kopiotu suoraan verkosta. Suoranaisia matemaattisia kaavoja oli vaikea hyödyntää, koska luomamme satunnaisverkon linkit olivat satunnaisia, joten laskut tehtiin keskiarvon mukaisesti.

# Lähteet

https://github.com/Wetski/Dijkstran\_algoritmi - Linkki 1