TiRaLabra, toteutusdokumentti

Arto Kaikkonen

# Ohjelman yleisrakenne

Ohjelma koostuu ReitinEtsija-luokkaa laajentavista luokista, joista kukin on yhden reitinhakualgoritmin toteutus. Jokaisella luokalla on konstruktori, joka saa parametrinaan XYVerkko-olion. Tämän jälkeen kutsutaan etsiLyhinReitti()-metodia, jolloin algoritmi suoritetaan. Tämän jälkeen voidaan hakea getterillä lyhin reitti, sen pituus sekä algoritmista riippuen myös muita arvoja.

XYVerkko koostuu taulukoista, joihon on tallennettu verkon solmut (XYKoordinaatti-oliot) sekä solmujen väliset etäisyydet. Tärkein konstruktori on konstruktori, jolle annetaan parametreina solmujen määrä ja alueen koko (x- ja y-koordinaattien maksimiarvot), ja joka arpoo verkon näiden parametrien mukaisesti.

XYKoordinaatti on luokka, jonka olioilla on muuttujinaan niiden kuvaaman pisteen x- ja y-koordinaatit.

# Saavutetut aika- ja tilavaativuudet

Tässä toteutuksessa on oletettu verkon olevan täydellinen. Koska kaikista solmuista lähtee kaari kaikkiin muihin solmuihin, on kaarien määrä O-analyysin näkökulmasta solmujen määrän neliö. Näin ollen en ole huomioinut kaarien määrää O-analyysiä tehdessäni.

## AntSystem

AntSystem-algoritmin aikavaatimus on n\*|V|^2, missä n on iteraatiokierrosten määrä ja V solmujen määrä. Algoritmin palauttaman tuloksen tarkkuus riippuu kuitenkin iteraatiokierrosten määrästä siten, että hyvän tuloksen vaatima kierrosmäärä on riippuvainen solmujen (tai ennemminkin kaarien) määrästä. Näin ollen aikavaativuus on ainakin V^3 (ehkä jopa V^4) mikäli halutaan, että tulokset säilyvät vertailukelpoisina.

## AinaLahimpaan

etsiLyhinReitti(){

for each v

aloitaEtsiminen(v)

aloitaEtsiminen(v)

while !ollaankoLopussa()

y = lahinKaymaton(v)

lisaaReittiin(y)

v=y

paivitaLyhinReitti

AinaLahimpaan-algoritmi toimii ajassa V^3. Metodissa etsiLyhinReitti kutsutaan V kertaa metodia aloitaEtsiminen. Metodissa aloitaEtsiminen kutsutaan V kertaa metodeja ollankoLopussa() ja lahinKaymaton(), jotka toimivat molemmat ajassa O(V). Lisäksi kutsutaan kerran ajassa V toimivaa paivitaLyhinReitti-algoritmia.

Mikäli metodi aloitaEtsiminen suoritettaisiin vain kerran, toimisi algoritmi ajassa O(V^2). Tällöin lopputulos olisi kuitenkin enemmän kiinni tuurista

## BruteForce

BruteForce-algoritmi käy läpi kaikki mahdolliset reitit, joten sen aikavaativuus on O(V !). Tilavaativuus on O(V), sillä jokaisen metodin tilavaativuus on O(1) ja rekursiopinon korkeus on korkeintaan V.

## BranchAndBound

BranchAndBoundin pahimman tapauksen aikavaativuus on sama kuin BruteForcella, eli O(V!).

## KarpHeld

KarpHeldin aikavaativuuden arviointiin eivät omat kykyni riitä, mutta aiheesta kirjoitettujen artikkeleiden mukaan se on O(V^2 \* 2^V).

# Suorituskyky- ja O-analyysivertailu (mikäli työ vertailupainotteinen)

# Työn mahdolliset puutteet ja parannusehdotukset

Vaikka algoritmeja tuskin saa hiottua nopeammiksi O-analyysimielessä, voi niiden toimintaa varmasti tehostaa paljonkin.

# Lähteet