

Ilkka Lehtikainen  
013787637

Aineopintojenharjoitustyö:  
Tietorakenteet ja algoritmit

## **Verkon vahvasti yhtenäiset komponentit**

### ***Toteutusdokumentti***

#### **Ohjelman yleisrakenne**

Ohjelman keskiössä on luokka VYK (vahvasti yhtenäiset komponentit), joka suorittaa varsinaisten algoritmien ajon erilaisilla verkoilla ja laskee näiden suoritukseen kuluvan ajan. Lisäksi ohjelmassa on kolme erillistä algoritmia (Kosaraju, Tarjan ja PathBased) verkon vahvasti yhtenäisten komponenttien löytämiseksi. Lisäksi ohjelmistoon kuuluvat apuluokat Solmu ja Pino, joilla on toteutettu algoritmien tarvitsema pino, sekä apuluokat PuuSolmu ja HakuPuu, joilla on toteutettu yksinkertainen rakenne tiedon tallennusta varten.

Ohjelma koostuu neljästä luokasta; VYK, Kosaraju, Tarjan ja PathBased. VYK-luokassa generoidaan suorituskäytöskäytännöissä tarvittavat verkot ja suoritetaan algoritmit samoille verkoille, jotta voidaan tehdä havaintoja niiden suorituskäytännöstä suhteessa toisiinsa. Muut luokat kukin käsittelevät yhden algoritmin, jolla etsitään verkosta vahvasti yhtenäiset komponentit.

#### **VYK**

Luokassa generoidaan suuria suunnattuja verkkoja muiden luokkien algoritmien suorituskäytöskäytännöjä varten.

Lisäksi suorituskäytöskäytännöistä suoritetaan tämän luokan kautta.

#### **Kosaraju**

Kosarajun algoritmi on tuttu jo luento monisteista [1] ja Cormanin kirjasta [2]. Siinä suoritetaan ensin verkon syvyysuuntainen läpikäynti sattumanvaraisesta solmusta, solmut talletetaan pinoon jälkijärjestyksessä eli kun solmu käsittelevä on tullut loppuun, lisätään solmu pinoon. Tämän jälkeen verkolle suoritetaan transpoosi, jossa verkon kaarien suunnat vaihdetaan päinvastaisiksi.

Lopuksi tälle transponoidulle verkolle suoritetaan syvyysuuntainen läpikäynti alkaen pinon pinnalta olevasta solmusta. Kun läpikäynti ei enää pääse uusiin solmuihin on löytynyt yksi vahvasti yhtenäinen komponentti. Löydettyihin komponentteihin kuuluvat solmut poistetaan pinosta ja syvyysuuntainen läpikäynti käynnistetään uudelleen jonon pinnalla olevasta solmusta. Ja tätä jatketaan niin kauan kuin pinossa on jäljellä käsittelemättömiä solmuja.

Kosarajun algoritmin aikavaativuus olisi vieruslistalla toteutettuna  $O(V+E)$ , mutta vierusmatriisiin läpikäynnin vuoksi algoritmin aikavaativuus on  $O(V^2)$ .

## Tarjan

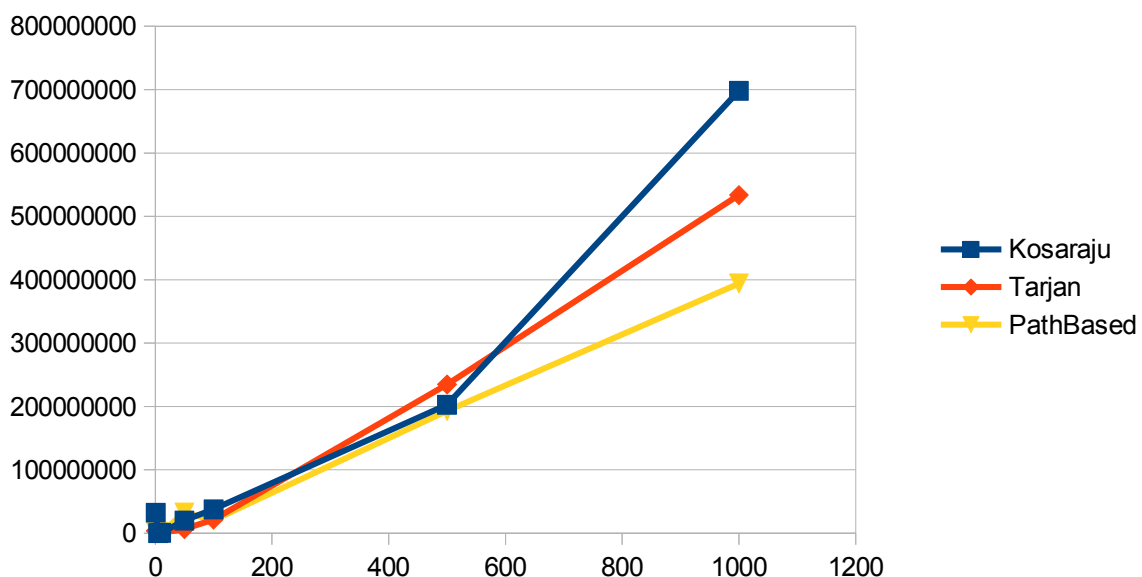
Tarjan-algoritmissa suoritetaan syvyysuuntaisia läpikäyntejä, niin kauan kuin verkossa on vielä käsittelemättömiä solmuja. Solmut laitetaan pinoon sitä mukaan kun ne tulevat käsittelyyn. Kun läpikäynti palaa niihin alipuusta, solmu otetaan pinosta ja tutkitaan, onko se jonkin vahvasti yhtenäisen komponentin juuri, jolloin sitä ennen pinosta poistetut solmut muodostavat vahvasti yhtenäisen komponentin.

## PathBased

Kuten edelliseltäkin niin myös PathBased-algoritmi perustuu syvyysuuntaiseen läpikäyntiin. Algoritmi käyttää pinoja ylläpitääkseen tietoa solmuista, joita ei ole vielä liitetty mihinkään vahvasti yhtenäiseen komponenttiin. Nämä solmut algoritmi liittää uuteen komponenttiin, kun viimeinen komponenttiin kuuluva solmu on käsitelty. Toisin kuin Tarjan-algoritmissa PathBased-algoritmi käyttää kahta pinoa. Ensimmäinen pino ylläpitää tietoa nykyisessä komponentissa olevista solmuista ja toinen pino pitää yllä listaa nykyisellä läpikäyntipolulla olevista solmuista.

Syvyysuuntainen läpikäynti, ylläpitäen kahta pinoa (S ja P). Pinossa S on kaikki ne solmut, joita ei ole vielä yhdistetty mihinkään vahvasti yhtenäiseen komponenttiin. Solmut ovat siinä järjestyksessä, jossa syvyysuuntainen läpikäynti ne saavuttaa (Esijärjestys). Pinossa P on ne solmut, joista ei ole vielä päätetty kuuluvatko ne samaan komponenttiin. Laskuriin C kertyy tieto siitä kuinka moneen solmuun läpikäynti on edennyt, jota käytetään esijärjestyksennumero.

## Saavutetut aika- ja tilavaativuudet (m.m. O-analyysi pseudokoodista)



Kuva 1. Esimerkki algoritmien nopeuserosta tiheydellä 0,1

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

### ***Puutteet ja parannusehdotukset***

HakuPuu pitäisi tasapainottaa lyhyempien hakuaikojen varmistamiseksi.

Algoritmiluokat voisi ohjelmoida myös vieruslistoille, jolloin päästäisiin matriisin läpikäynnistä solmujen ja kaarien läpikäyntiin, joka vähentäisi varsinkin harvassa verkossa matriisin tyhjien kaarien tarkistamista. Lisäksi voisi ohjelmoida muunnokset vierusmatriisista vieruslistaan ja päinvastoin.

Myöskin syvyys-suuntaisen läpikäynnin rekursion toteuttaminen iteraatiolla olisi mielenkiintoinen vertailu.

### ***Lähteet***