Ilkka Lehikoinen
013787637

Aineopintojenharjoitustyö:
Tietorakenteet ja algoritmit

Verkon vahvasti yhtenäiset komponentit

Testausdokumentti

Toiminnallisuuden testaus

Apuluokkien Solmu, Pino, PuuSolmu ja HakuPuu metodit ovat testatut JUnitilla niitä varten tehdyillä testitapauksilla, jotka ottavat huomioon monia erilaisia tapauksia.

Koska algoritmiluokkani (Kosaraju, Tarjan ja PathBased) eivät aluksi palauttaneet mitään, joten testasin niiden toiminnan manuaalisesti seuraavassa kuvatulla tavalla.

Olen eri lähtestä kopioinut kolme suunnattua kahdeksan solmun verkkoa, joiden vahvasti yhtenäiset komponentit on tunnettu.

{0,1,4},{2},{3,7},{5,6}
```
static int[][] esim1 = new int[][] {
    {0, 0, 0, 0, 1, 0, 0, 0},
    {1, 0, 0, 0, 0, 0, 0, 0},
    {0, 1, 0, 0, 0, 0, 1, 0},
    {0, 0, 1, 0, 0, 0, 0, 1},
    {1, 1, 0, 0, 0, 0, 0, 0},
    {0, 1, 0, 0, 1, 0, 1, 0},
    {0, 0, 0, 0, 0, 1, 0, 0},
    {0, 0, 0, 1, 0, 0, 1, 0}};
```

{0,1,4},{2,3},{5,6},{7}
```
static int[][] esim2 = new int[][] {
    {0, 1, 0, 0, 0, 0, 0, 0},
    {0, 0, 1, 0, 1, 0, 0, 0},
    {0, 0, 0, 1, 0, 0, 1, 0},
    {0, 0, 1, 0, 0, 0, 0, 1},
    {1, 0, 0, 0, 0, 1, 0, 0},
    {0, 0, 0, 0, 0, 0, 1, 0},
    {0, 0, 0, 0, 0, 1, 0, 1},
    {0, 0, 0, 0, 0, 0, 0, 1}};
```

{0,1,4},{2,3},{5,6},{7}
```
static int[][] esim3 = new int[][] {
    {0, 0, 0, 0, 1, 0, 0, 0},
    {1, 0, 0, 0, 0, 0, 0, 0},
    {0, 0, 0, 1, 0, 0, 0, 0},
    {0, 0, 1, 0, 0, 0, 0, 0},
    {0, 1, 0, 0, 0, 0, 0, 0},
    {0, 1, 0, 0, 0, 0, 1, 0},
    {0, 0, 1, 0, 0, 1, 0, 0},
    {0, 0, 0, 1, 0, 0, 1, 1}};
```

Algoritmiluokkien testauksessa käytettyjä vierusmatriiseja.

Näitä verkkoja olen käyttänyt syötteenä kullekin algoritmille ja todennut että ne antavat oikean vastauksen.

Koska algoritmit suorittavat samaan tehtävää voidaan niiden tuloksia verrata toisiinsa, jolloin havaitaan, jos jossakin algoritmissa on virhe. Tätä ominaisuutta olen käyttänyt hyväksi ja generoinut erikokoisia ja -tiheyksisiä satunnaisia suunnattuja verkkoja ja todennut, että kukin algoritmi antaa vahvasti yhtenäiset komponentit, joissa jokainen solmu esiintyy jossakin komponentissa ja kukin solmu esiintyy ainoastaan yhdessä komponentissa. Lisäksi olen verrannut algoritmien antamaa vastausta toisiinsa ja todennut, että kaikki kolme algoritmia antaa samalle verkolle samat vahvasti yhtenäiset komponentit.

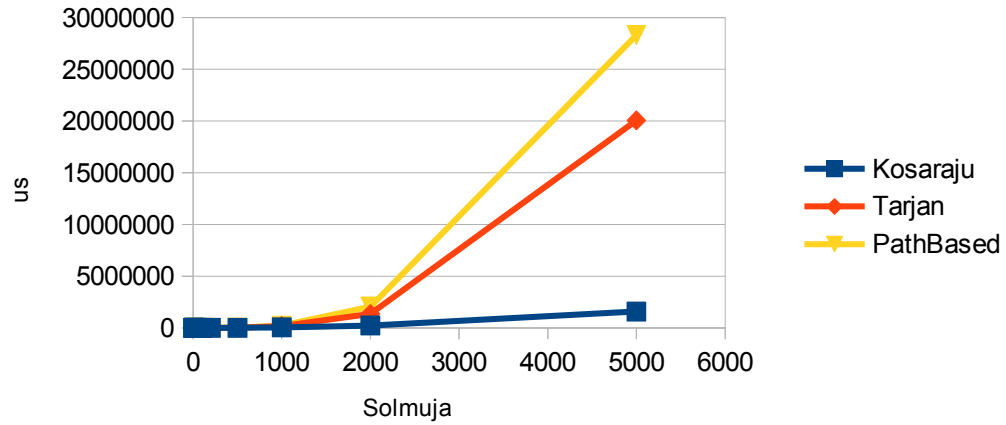
Tämän perusteella olen varma, että kaikki kolme algoritmia löytävät suunnatusta verkosta sen vahvasti yhtenäiset komponentit.

Nyt kun algoritmiluokat ja osa niiden metodeista palauttavat merkkijonoja (String), niin kaikille luokille on automaattisia JUnit-testejä.

Suorituskyvyn testaus

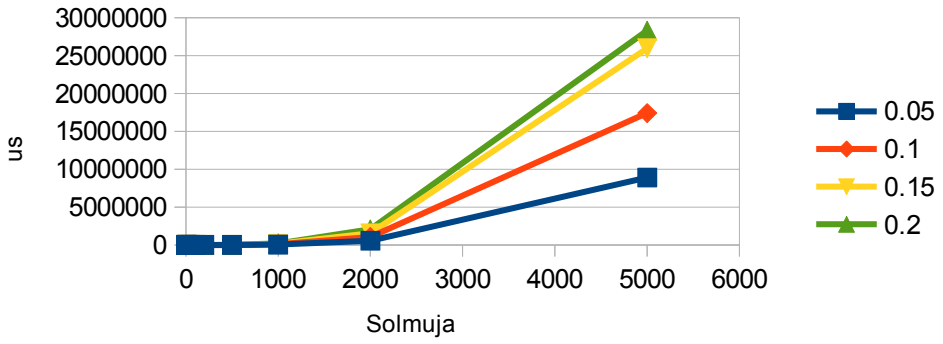
Algoritmeille syötettiin useita suunnattuja verkkoja, joissa solmujen määrät vaihteli yhdestä 5000:een ja tiheydet 0.05-0.020. Koska käyttöjärjestelmä tai muut ohjelmat saattavat vaatia prosessorilta toiminta-aikaa kesken algoritmien testauksen, on ollut tarpeen mitata samankaltaisilla verkoilla useita kertoja ja sitten keskiarvoistaa tulokset.

Tuloksista (Kuva 1.) nähdään että Kosaraju on testattuista algoritmeista nopein ja saavuttaa ennakolta odotettavan aikavaativuuden $O(n^2)$, kun se sijaan algoritmien Tarjan ja PathBased aikavaativuudet lähentelevät kuutiollista $O(n^3)$.



Kuva 1. Algoritmien vertailu verkon tiheydellä 0.2. (Arvot keskiarvoja, n=10)

Kosarajun algoritmi on suhteellisen immuuni kaarien lukumäärälle, mutta sen sijaan sekä Tarjan että PathBased algoritmien suoritusnopeudet ovat selvästi lineaarisesti riippuvia verkon kaarien määrästä.



Kuva 2. PathBased algoritmin suoritusnopeuden riippuvuus verkon kaarien lukumäärästä. (Arvot keskiarvoja, n=5)

Integrointi- ja regressiotestaus

VYK-luokassa manuaalisesti tehtyjä verkkoja ja generoituja satunnaisverkkoja on annettu kullekin algoritmiluokalle, ja nämä ovat kaikki palauttaneet samat vahvasti yhtenäiset komponentit. Koska algoritmit suorittavat samaa tehtävää, niin niiden tuloksia voidaan verrata toisiinsa. Näin, etenkin silloin jos johonkin algoritmiin tehdään muutoksia, voidaan testata että muutettu algoritmi edelleen palauttaa samat vahvasti yhtenäiset komponentit kuin aiemmin oikein toimivat algoritmit.

Lähteet:

[1] Introduction to Algorithms 3rd ed. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein. The MIT Press 2009.

[2] Luentomonisteet: http://www.cs.helsinki.fi/u/floreen/tira2013/tira.pdf

[3] Tarjan: http://en.wikipedia.org/wiki/Tarjan%27s_strongly_connected_components_algorithm

[4] PathBased: http://en.wikipedia.org/wiki/Path-based_strong_component_algorithm