Title will be here

Rodion Efremov

Master thesis
UNIVERSITY OF HELSINKI
Department of Computer Science

Helsinki, May 5, 2016

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

Tiedekunta — Fakultet — Faculty	Laitos — Institution — Department
Faculty of Science	Department of Computer Science

Tekijä — Författare — Author

Rodion Efremov

Työn nimi — Arbetets titel — Title

Title will be here

Oppiaine — Läroämne — Subject

Computer Science

Työn laji — Arbetets art — Level	Aika — Datum — Month and year	Sivumäärä — Sidoantal — Number of pages
Master thesis	May 5, 2016	1

Tiivistelmä — Referat — Abstract

Abstract goes here

Avainsanat — Nyckelord — Keywords

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

Contents

Chapt

Dummy section

0.1 Strongly connected components

Since our methods require the input graph to be strongly connected, we review here briefly how to algorithmically validate that the input graph exhibits the requirement. A directed graph is called *strongly connected* if and only if for every pair of nodes u, v of the graph u is reachable from v, and v is reachable from u. Let $u \overset{r}{\sim} v$ denote the aforementioned reachability relation. Now, it is easy to see that

(Reflexivity) $u \overset{r}{\sim} u$, for all $u \in V(G)$.

(Symmetry) if $u \overset{r}{\sim} v$, then $v \overset{r}{\sim} u$.

(Transitivity) If $u \overset{r}{\sim} v$ and $v \overset{r}{\sim} v'$, then $u \overset{r}{\sim} v'$.

The above three properties imply that $\overset{r}{\sim}$ is an equivalence relation, and as such, implies that the graph has a unique partition into strongly connected components.

Algorithms for finding all strongly connected components of a graph in linear time are known. We review three of them below.

0.1.1 Kosaraju's algorithm

Algorithm 1: KosarajuVisit(G, S, L, v)

if $v \notin S$ **then**
- $S \leftarrow S \cup \{v\}$
- **for** $(v, w) \in G(A)$ **do**
  - KosarajuVisit(G, S, L, w)
- $L \leftarrow \langle v \rangle \circ L$

Algorithm 2: KosarajuAssign(G, μ, u, r)

if $(u \mapsto r) \notin \mu$ **then**
- $\mu(u) \leftarrow r$
- For all parents of u
- **for** $(v, u) \in G(A)$ **do**
  - KosarajuAssign$(G\mu, v, r)$

Algorithm 3: KosarajuSCC(G)

$S \leftarrow \varnothing$
$L \leftarrow \langle\rangle$
$\mu \leftarrow \varnothing$
for $v \in V(G)$ **do**
    KosarajuVisit(G, S, L, v)
Iterate the list L in its natural order
for $v \in L$ **do**
    KosarajuAssign()