Universidade Federal de Minas Gerais
Instituto de ciências exatas
Departamento de ciência da computação
Projeto e análise de algoritmos

# Practical Work 2

September 21, 2014

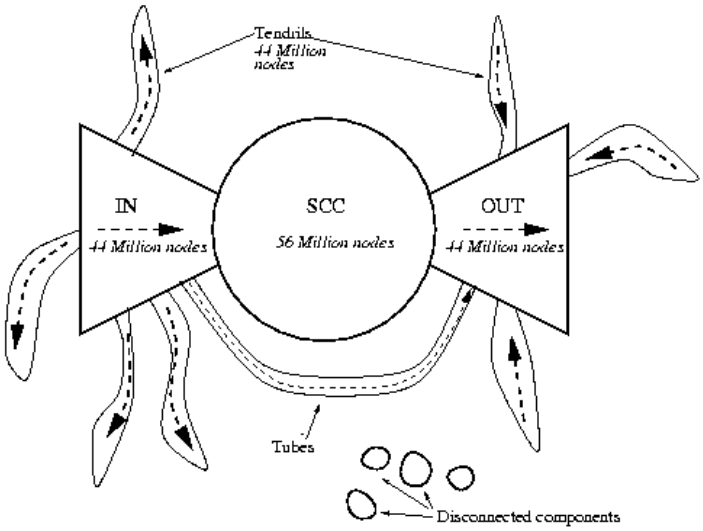
## 1 The Bow-Tie Structure of the Web

In 1999, after the Web had been growing for the better part of a decade, Andrei Broder and his colleagues set out to build a global map of the Web, using strongly connected components as the basic building blocks [1]. They used the index of pages and links from one of the largest commercial search engines at the time, AltaVista. Their influential study has since been replicated on other, even larger snapshots of the Web, including an early index of Google's search engine and large research collections of Web pages [2].

Broder et al [1] modeled the web by dividing it in larger pieces, and show in a stylized way how these pieces fit together. They proposed the 5 types of structures below and created a nice bow-tie structure as shown in Figure 1.1.

- A giant strongly connected component (SCC): These nodes are the largest strongly connected component of the graph.
- IN nodes: These nodes can reach the giant SCC but cannot be reached from it - i.e., nodes that are "upstream" of it.
- OUT nodes: These nodes can be reached from the giant SCC but cannot reach it - i.e., nodes are "downstream" of it.
- Tendrils: They consist of (a) the nodes reachable from IN that cannot reach the giant SCC, and (b) the nodes that can reach OUT but cannot be reached from the giant SCC, and (c) the nodes that satisfy both (a)

Figure 1.1: The Bow-Tie Structure of the Web [1]




and (b), and they consist of a "tube" that travels from IN to OUT without touching the giant SCC.

- Disconnected: These nodes do not have a path to the giant SCC even if we completely ignore the directions of the edges.

Taken as a whole, then, the bow-tie picture of the Web provides a high-level view of the Web's structure, based on its reachability properties and how its strongly connected components fit together.

**The goal of this work** is to implement a program that correctly classify the 7 types of structures above inside a network of websites (those include the types (a),(b) and (c) of tendrils). The project SNAP [1] provides four interesting datasets of web graphs that might be used to test the program. In the following section we provide the instructions about the implementation of the program.

**It is forbidden to use external libraries or off the shelf software to manipulate or compute the graph. Only standard resources of the language are allowed.**

## 2 Instructions about Implementation

We will accept only implementations in C, C++ and Java. Be sure that your code can be compiled and executed in GNU/Linux machines. The created program must receive a text file as input and return 7 text files as result. The

[1] `http://snap.stanford.edu/data/index.html#web`

content of the input and output files are specified in the next section and ONLY that format will be accepted.

In addition to the code that solves the proposed problem, you must write shell scripts (.sh) to compile and run your code. These scripts must be named *compile.sh* and *execute.sh* respectively. It follows below some examples of these scripts:

- **compile.sh**

```
#!/bin/bash
gcc main.c -o main
```

- **execute.sh infile**

```
#!/bin/bash
./main $1
```

If you write your code in Java, you can use the command *java -jar program.jar*, where *program.jar* is the executable file of your project generated on *compile.sh*.

The implementations must be tested on computers of the Department of Computer Science where the postgraduate students have free access. This test is to ensure that the implementation will be compiled and executed in an environment known by the student. For example, two available computers are:

- *cipo.grad.dcc.ufmg.br*
- *claro.grad.dcc.ufmg.br*

**The performance of the solution (time and space) will be taken into consideration in the evaluation.**

## Input and Output

The program must solve one instance of the problem per execution. Each instance of the problem contains a list of nodes with the following format:

```
1 2
2 3
3 1
3 2
3 4
...
```

Every line represent pairs of nodes and consists of 2 integers separated by spaces, where the first number is the start node and the second number is the goal node. Remember that the program must receive as input parameter the name of the file with the node list (e.g., *./execute.sh nodelist.txt*).

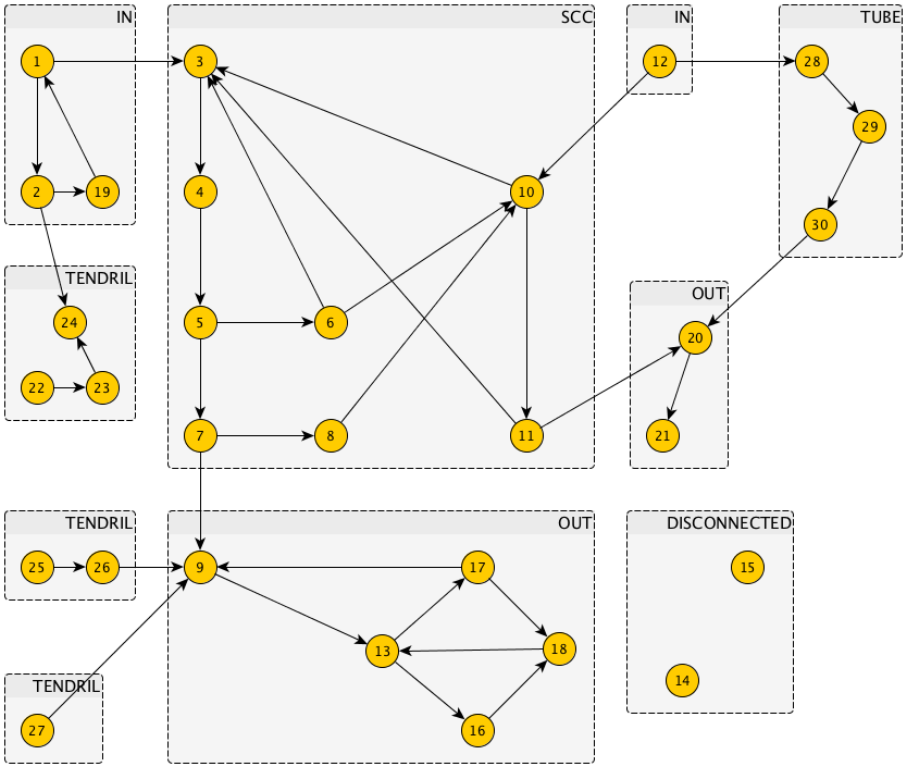
The output of the program must be 7 text files written on the same directory of program execution. These files contain the list of nodes for every classification type organized in ascending order $(1, 2, 3, ..., n)$. The text files must receive the following names:

- **disconnected.txt** - disconnected nodes.
- **in.txt** - IN nodes.
- **scc.txt** - nodes in the largest strongly connected component.
- **out.txt** - OUT nodes.
- **tendrils_a.txt** - tendrils nodes of types (a).
- **tendrils_b.txt** - tendrils nodes of types (b).
- **tendrils_c.txt** - tendrils nodes of types (c).

It follows below an example (Fig 2.1) that might be used as a first test of your program.

## Example

Figure 2.1: Example graph



**input.txt**

```
1 2
1 3
2 24
2 19
3 4
4 5
5 6
5 7
6 3
7 8
7 9
8 10
9 13
10 11
10 3
11 3
11 20
12 10
12 28
13 17
13 16
14 14
15 15
16 18
17 18
17 9
18 13
19 1
20 21
22 23
23 24
25 26
26 9
27 9
28 29
29 30
30 20
```

**Output:**
**disconnected.txt**:

```
14
15
```

**in.txt**:

```
1
2
12
19
```

**scc.txt**:

```
3
4
5
6
7
8
10
11
```

**out.txt**:

```
9
13
16
17
18
20
21
```

**tendrils_a.txt**:

```
22
23
24
```

**tendrils_b.txt**:

```
25
26
27
```

**tendrils_c.txt**:

```
28
29
30
```

This example can also be downloaded from `http://pastebin.com/SE2CndHB`
.

# 3 Documentation

Start your documentation with a brief description of the problem. In the following you can present your solution for the problem. It is important to mention in the documentation:

- The description of the problem, the graph that was used to model the problem, and how the graph was generated.
- Complexity analysis of time and memory for each solution.
- Analysis of the obtained results.
- The documentation must not exceed 8 pages.
- The documentation will be submitted via *minha.ufmg*
- The deadline: **12/10**

# 4 Parameters of evaluation

The evaluation parameters are divided into the following topics:

- Problem Modeling (20%)
- Theoretical Analysis of Asymptotic Cost (10%)
- Execution performance and Analysis of Experiments (35%)
- Correct Operation of the Code (35%)

**Good luck!**

# References


[1] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. In *Proc. of the 9th International World Wide Web Conference on Computer Networks*, pages 309–320, Amsterdam, The Netherlands, 2000.

[2] D. Easley and J. Kleinberg. *Networks, Crowds, and Markets: Reasoning about a Highly Connected World*. Cambridge University Press, 2009.