1 Source Code Complexity Metrics

Table of Contents

1.0 Introduction

ISTQB: complexity
See Also: cyclomatic complexity
The degree to which a component or system has a design and/or internal structure that is difficult to understand, maintain and verify.

This mini-lab will introduce how to calculate the code complexity with an analyzer tool named Lizard using Python. Documentation:

- https://pypi.org/project/lizard/
- http://www.lizard.ws/
- https://github.com/terryyin/lizard

It works similar to software as McCabe IQ: http://mccabe.com/ or the VS EE built-in tool Code Metrics https://docs.microsoft.com/en-us/visualstudio/code-quality.

If you want html output you need to install “pip install jinja2” as well. An example run against RomanNumeral.cs and CcmTest:
C:\> lizard -t 2 -l csharp --html --verbose > ./ccm_outp.html.
Here just the most complex method is listed.

Function name	Cyclomatic complexity (15)	LOC (1000)	Token count	Parameter count (100)
KeesTalksTech.Utilities.Latin.Numerals::RomanNumeral::Parse	10	44	219	1

- CCN or Cyclomatic complexity number, above 15 will get a warning.
    - Cyclomatic complexity is a software metric used to indicate the complexity of a program. It is a quantitative measure of the number of linearly independent paths through a program's source code. https://en.wikipedia.org/wiki/Cyclomatic_complexity
    - Going over 15 for CCN is not recommended in software development: https://en.wikipedia.org/wiki/Cyclomatic_complexity#Limiting_complexity_during_development

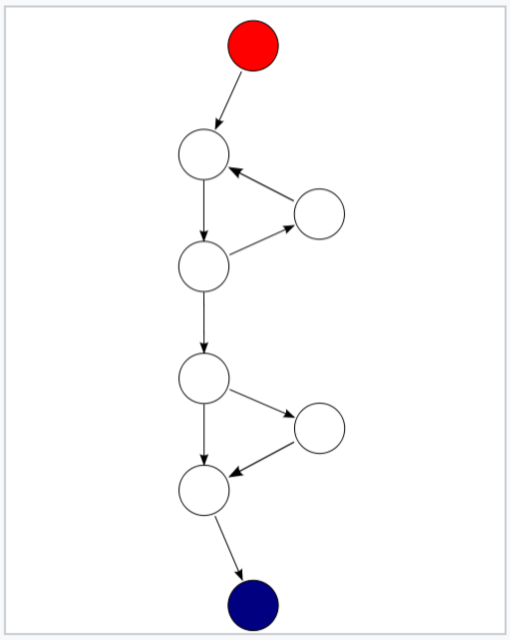
- NLOC/SLOC or just LOC is lines of code without comments, above 1000 will get a warning.
- Token count is the size of the vocabulary of a program, which consists of the number of unique tokens used to build a program. Note that Lizard do this calculation for every function/method instead.
    - Token count is defined as: $\eta = \eta 1 + \eta 2$, where
      η: vocabulary of a program
      η1: number of unique operators
      η2: number of unique operands
- Parameter count is the number of input parameters for functions/methods, above 100 will get a warning

Example 1

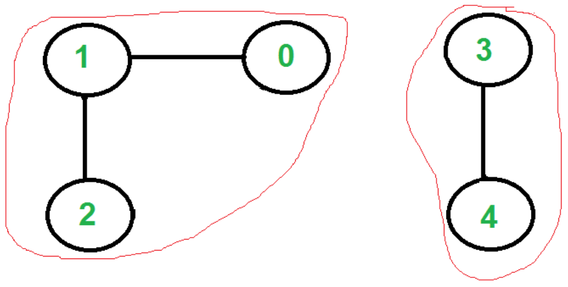
The complexity M is then defined as
$M = E - N + 2P$,
where

- E = the number of edges of the graph.
- N = the number of nodes of the graph.
- P = the number of connected components.

9 edges (arrows), 8 nodes and one component.
CCN = 9-8 + 2*1 => 3

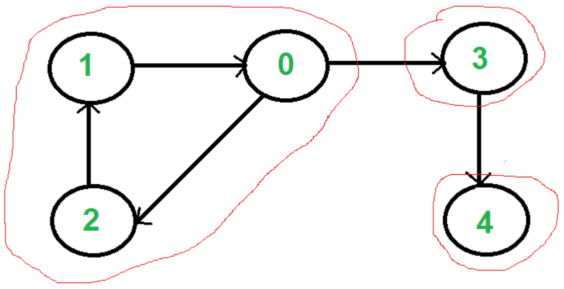

A control flow graph of a simple program. The program begins executing at the red node, then enters a loop (group of three nodes immediately below the red node). On exiting the loop, there is a conditional statement (group below the loop), and finally the program exits at the blue node. This graph has 9 edges, 8 nodes, and 1 connected component, so the cyclomatic complexity of the program is 9 - 8 + 2*1 = 3.

Example of 2 connected components.
https://www.geeksforgeeks.org/connected-components-in-an-undirected-graph/



There are two connected components in above undirected graph
0 1 2
3 4

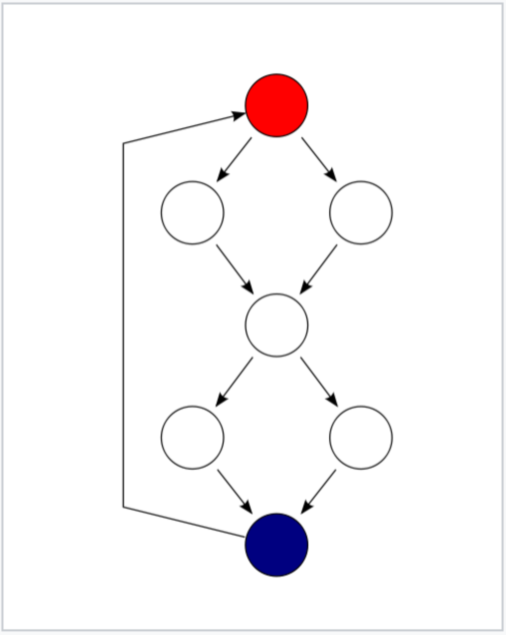
Example of 3 strongly connected components.
https://www.geeksforgeeks.org/strongly-connected-components/



Example 2

For **strongly** connected (in which each exit point is connected back to the entry point) the complexity M is then defined as: M = E − N + P

9 edges (arrows), 7 nodes and one component.
CCN = 9-7 + 1 => 3


The control flow graph of the source code above; the red circle is the entry point of the function, and the blue circle is the exit point. The exit has been connected to the entry to make the graph strongly connected.

Example 3

What is the CCN for the flow graph (M) to the right?
Using v(M) = E - N + 2

E = number of edges in the graph = 17
N = number of nodes in the graph = 13

v (M) = E - N + 2 = 17 - 13 + 2 = 6