Teste de Software

Complexidade Ciclomática para Teste Estrutural

PUC Minas

Instituto de Ciências Exatas
e Informática

Prof. Lesandro Ponciano

Departamento de Engenharia de Software
e Sistemas de Informação (DES)

Objetivos da Aula

- Contextualizar Complexidade Ciclomática
- Introduzir formas de cálculo de complexidade
- Analisar o emprego da complexidade ciclomática na construção de casos de teste

Complexidade de Programas

- Como medir a complexidade de um programa a partir de sua estrutura interna?

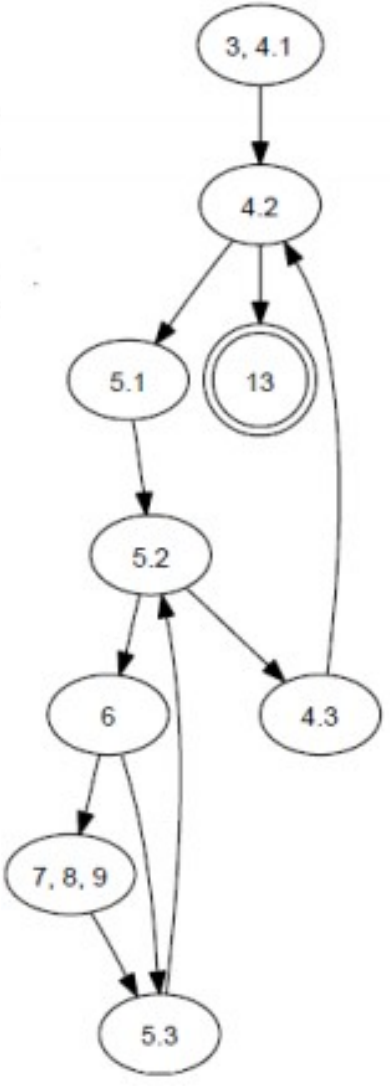
- A partir do grafo de fluxo de controle
  - Qual é o número de caminhos linearmente independentes?

- Caminho Linearmente Independente
  - é qualquer caminho que introduz pelo menos um novo conjunto de comandos ou uma nova condição
  - no grafo, significa incluir pelo menos uma aresta que não tenha sido atravessada antes de o caminho ser definido

Complexidade Ciclomática

- Dado um grafo de fluxo de controle (G), a complexidade ciclomática mede
  - Número de caminhos independentes
- $V(G) = E - N + 2$
  - E é o número de ramos do grafo
  - N é o numero de nós do grafo
- $V(G) = P + 1$
  - P é o número de nós predicados do grafo
  - Nó predicado é o que tem duas ou mais arestas saindo dele

```
2   public void bolha(int[] a, int size) {
3       int i, j, aux;
4       for (i = 0; i < size; i++) {
5           for (j = size - 1; j > i; j--) {
6               if (a[j - 1] > a[j]) {
7                   aux = a[j - 1];
8                   a[j - 1] = a[j];
9                   a[j] = aux;
10              }
11          }
12      }
13  }
```



- V(G) = E – N + 2
- V(G) = 11-9 + 2
- V(G) = 4

- V(G) = P + 1
- V(G) = 3 + 1
- V(G) = 4

Complexidade e Testabilidade

- Testabilidade diz respeito
  - à capacidade de ser testado
  - ao quão plausível é de se testar um programa e detectar seus eventuais defeitos
- Programa com baixa testabilidade
  - dificuldade de se construir casos de teste efetivos
- Complexidade ciclomática pode ser usada para medir a testabilidade
  - uma vez que ela mede a quantidade de caminhos linearmente independentes neste programa

Derivação de Casos de Teste

1) Desenhe o grafo de fluxo de controle correspondente

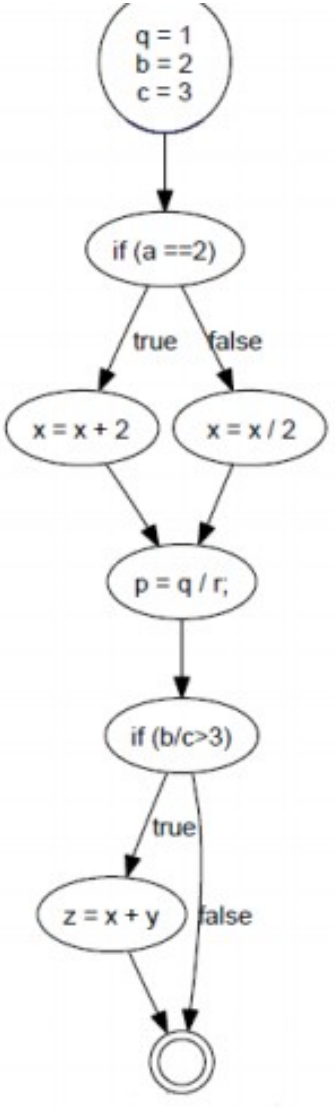
2) Determine a complexidade ciclomática do grafo

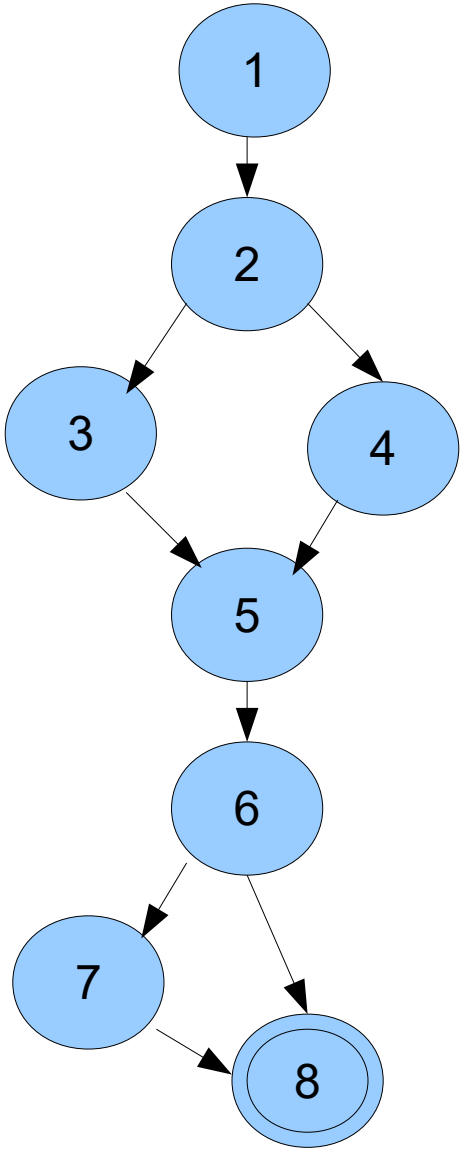
3) Determine um conjunto base de caminhos linearmente independentes

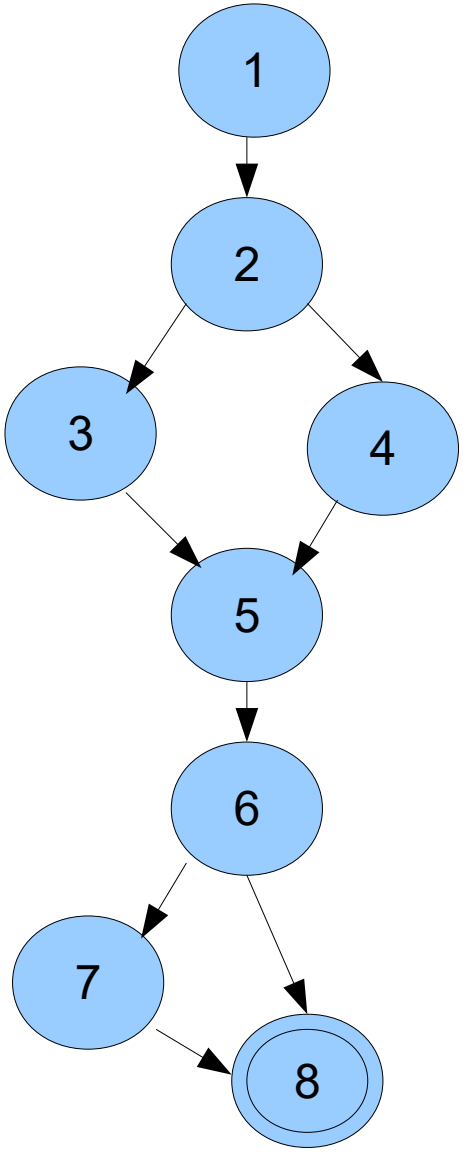
4) Prepare os casos de teste que vão forçar a execução de cada caminho do conjunto.

Passo 1

```
1   q = 1;
2   b = 2;
3   c = 3;
4   if (a ==2) {
5       x = x + 2;
6   } else {
7       x = x / 2;
8   }
9   p = q / r;
10  if (b/c>3) {
11      z = x + y;
12  }
```



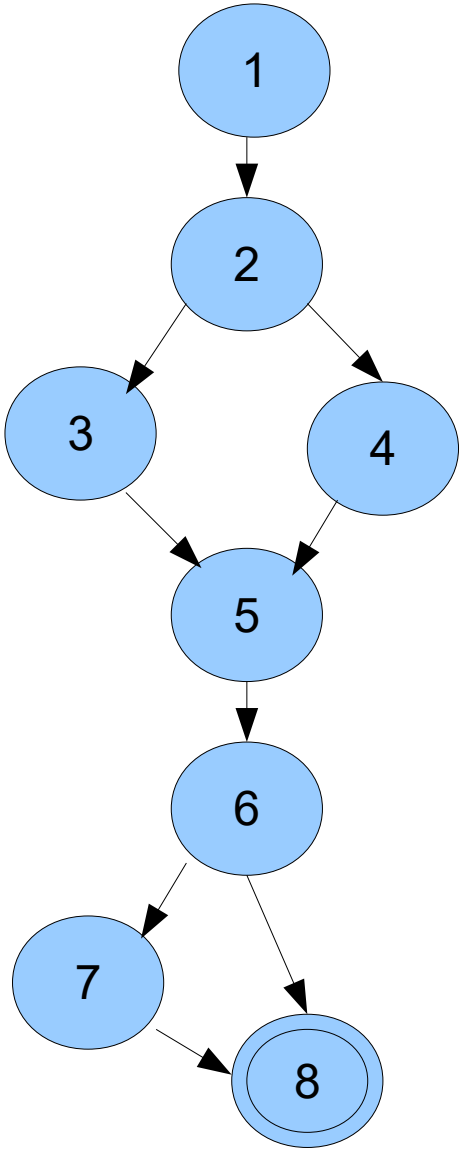




Passo 2

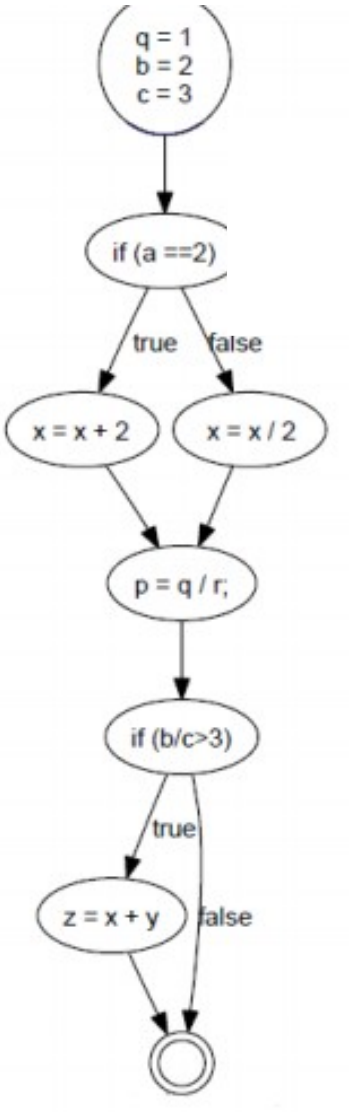
- V(G) = E – N + 2
- V(G) = 9 - 8 + 2
- V(G) = 3

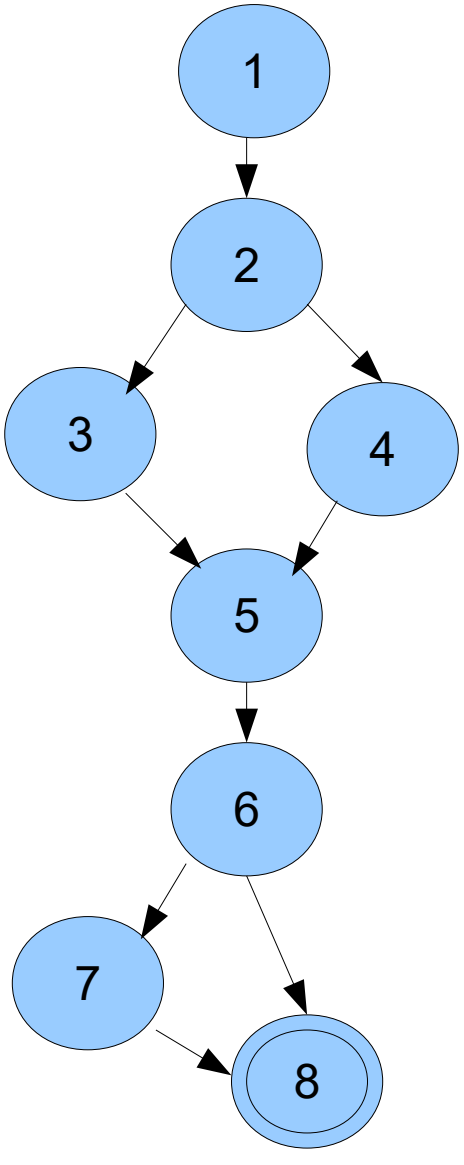
- V(G) = P + 1
- V(G) = 2 + 1
- V(G) = 3



Passo 3

- V(G) = 3
- Caminhos
  - 1-2-3-5-6-7-8
  - 1-2-4-5-6-7-8
  - 1-2-3-5-6-8





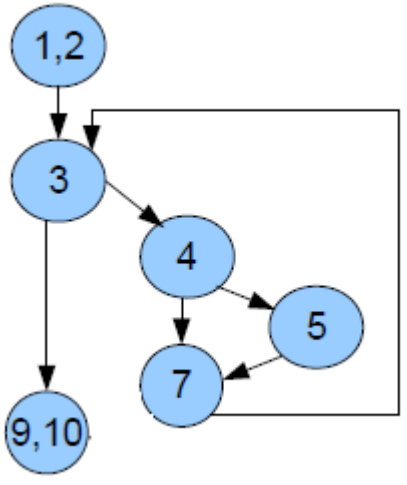
Passo 4

- Caminhos
  1) 1-2-3-5-6-7-8
  2) 1-2-4-5-6-7-8
  3) 1-2-3-5-6-8

- Casos de Teste
  1) a=2,b=12,c=2
  2) a=1,b=12,c=2
  3) a=2,b=4,c=2

Exemplo

```
1. int foo(int a, int b){
2.       int c=0;
3.       while(a<0){
4.           if (b<0){
5.               b=b+2;
6.           }
7.           a=a+1;
8.       }
9.       c=a+b;
10.      return c;
11. }
```



- V(G) = P + 1
  - V(G)=2+1
- Caminhos
  1) (1,2)-3-(9,10)
  2) (1,2)-3-4-7-3-(9,10)
  3) (1,2)-3-4-5-7-3-(9,10)
- Casos de Teste
  1) a=1, b=1
  2) a=-1, b=1
  3) a=-1, b=-1

Exemplo

my_foo.py

```python
class Foo:

    def __init__(self):
        pass

    def foo(self, a, b):
        c = 0
        while(a < 0):
            if (b < 0):
                b = b + 2
            a = a + 1

        c = a + b
        return c
```

testaFoo.py

```python
import unittest

from my_foo import Foo

class TestMethod(unittest.TestCase):

    m = Foo()

    def test_case1(self):
        self.assertEqual(self.m.foo(1,1), 2, "Precisa ser 2")

    def test_case2(self):
        self.assertEqual(self.m.foo(-1,1), 1, "Precisa ser 1")

    def test_case3(self):
        self.assertEqual(self.m.foo(-1,-1), 1,  "Precisa ser 1")

    def suite():
        suite = unittest.TestSuite()
        suite.addTest(WidgetTestCase('test_case1'))
        suite.addTest(WidgetTestCase('test_case2'))
        suite.addTest(WidgetTestCase('test_case3'))

if __name__ == '__main__':
    runner = unittest.TextTestRunner()
    runner.run(suite())
```

Referências

- Delamaro, M. E., Maldonado, J. C., & Jino, M. (2016). Introdução ao teste de software. Rio de Janeiro: Elsevier. (Capítulo 4)
- Myers, Glenford J. et al (2004) "The Art of Software Testing." 2ed. New York, NY, USA: John Wiley & Sons.
- Barbosa, Ellen Francine; de Souza, Simone do Rocio Senger. Técnica de Teste Estrutural. https://edisciplinas.usp.br/pluginfile.php/4648952/mod_resource/content/0/Aula05-TecnicaEstrutural-PartеI.pdf
- Pimentel, Andrey Ricardo. Teste de Software Estrutural ou “CaixaBranca”. http://www.inf.ufpr.br/andrey/ci221/apresentacaoTesteEstrutural.pdf
- https://www.sonarqube.org/