Estratégias de testes de software

Prof. Dr. Rafael Galvão de Mesquita
rafael.mesquita@garanhuns.ifpe.edu.br

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO

Conteúdo da aula

- Estratégias de testes
  - Caminhos básicos
  - Particionamento em classes de equivalência
  - Análise de valor limite
  - Valores especiais

Caminhos básicos

- Teste caixa branca que busca acessar todos os possíveis caminhos do código do módulo testado
  - Em um conjunto de casos de teste, garante a execução de cada instrução pelo menos uma vez
  - Obs: um caso de teste é definido por um conjunto de valores de entradas e uma saída esperada

Como você testaria o codigo abaixo?

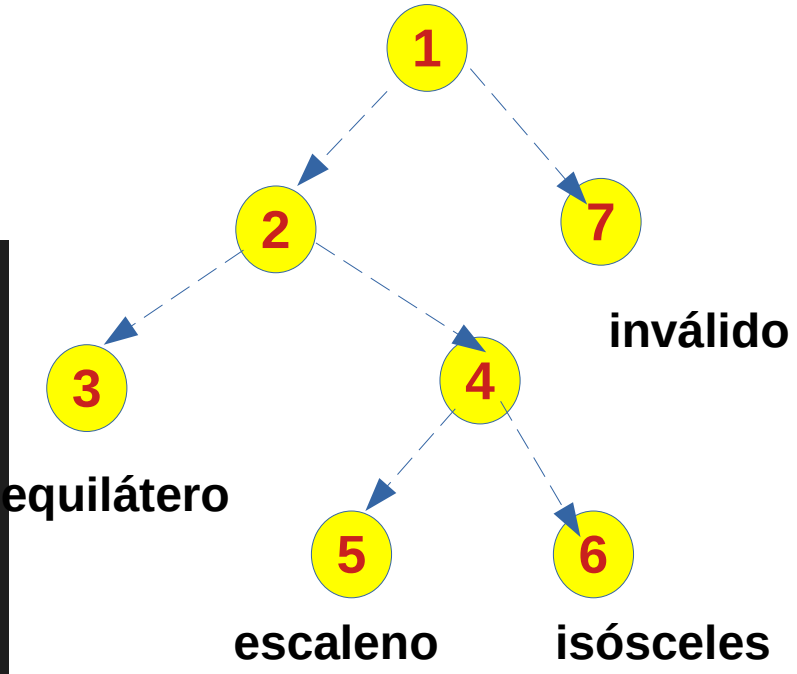
- Defina 4 casos de testes da melhor forma que puder

```
1   lado1 = int(input('tamanho do lado 1:'))
2   lado2 = int(input('tamanho do lado 2:'))
3   lado3 = int(input('tamanho do lado 3:'))
4
5   if((lado1 < lado2 + lado3) and (lado1 > abs(lado2-lado3)) and
6     (lado2 < lado1 + lado3) and (lado2 > abs(lado1-lado3)) and
7     (lado3 < lado2 + lado1) and (lado3 > abs(lado2-lado1))):
8
9       if(lado1 == lado2 and lado2 == lado3):
10          print('triângulo é equilátero')
11      else:
12          if (lado1 != lado2 and lado2 != lado3 and lado1 != lado3):
13              print('triângulo é escaleno')
14          else:
15              print('triângulo é isosceles')
16
17  else:
18      print('triângulo inválido')
```

Caminhos básicos

```
1   lado1 = int(input('tamanho do lado 1:'))
2   lado2 = int(input('tamanho do lado 2:'))
3   lado3 = int(input('tamanho do lado 3:'))
4
5   if((lado1 < lado2 + lado3) and (lado1 > abs(lado2-lado3)) and
6     (lado2 < lado1 + lado3) and (lado2 > abs(lado1-lado3)) and
7     (lado3 < lado2 + lado1) and (lado3 > abs(lado2-lado1))):
8
9       if(lado1 == lado2 and lado2 == lado3):
10          print('triângulo é equilátero')
11      else:
12          if (lado1 != lado2 and lado2 != lado3 and lado1 != lado3):
13              print('triângulo é escaleno')
14          else:
15              print('triângulo é isosceles')
16
17  else:
18      print('triângulo inválido')
```

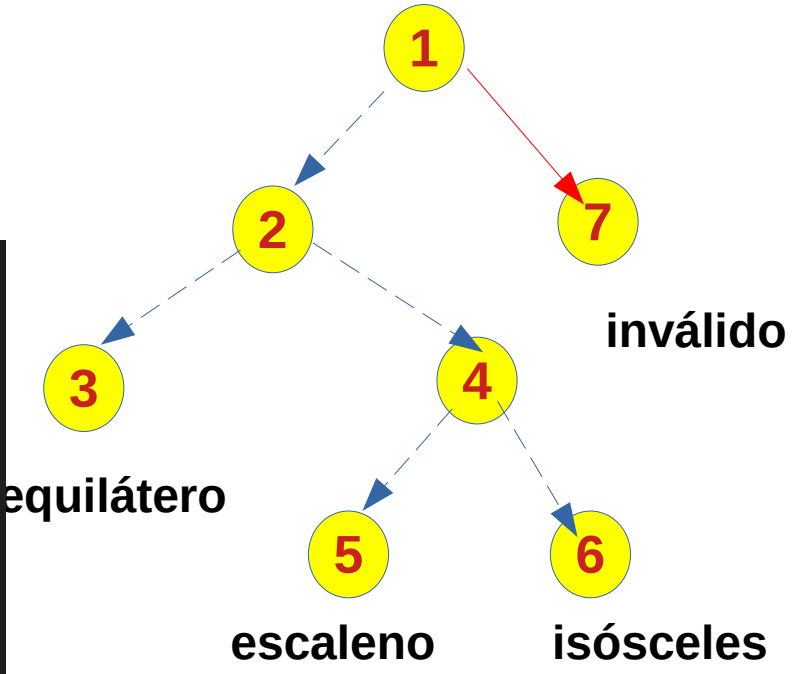
1 2 3 4 5 6 7



Caminhos básicos

```
1  lado1 = int(input('tamanho do lado 1:'))
2  lado2 = int(input('tamanho do lado 2:'))
3  lado3 = int(input('tamanho do lado 3:'))
4
5  if((lado1 < lado2 + lado3) and (lado1 > abs(lado2-lado3)) and
6    (lado2 < lado1 + lado3) and (lado2 > abs(lado1-lado3)) and
7    (lado3 < lado2 + lado1) and (lado3 > abs(lado2-lado1))):
8
9      if(lado1 == lado2 and lado2 == lado3):
10         print('triângulo é equilátero')
11     else:
12         if (lado1 != lado2 and lado2 != lado3 and lado1 != lado3):
13             print('triângulo é escaleno')
14         else:
15             print('triângulo é isosceles')
16
17 else:
18     print('triângulo inválido')
```

1 2 3 4 5 6 7

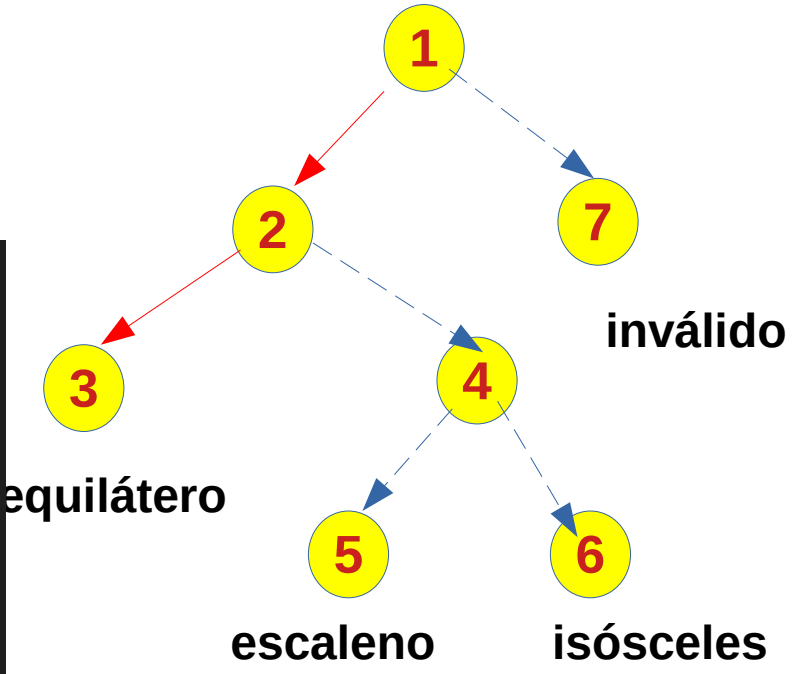


Entrada: 10 9 20
Saída: triângulo inválido

Caminhos básicos

```
1  lado1 = int(input('tamanho do lado 1:'))
2  lado2 = int(input('tamanho do lado 2:'))
3  lado3 = int(input('tamanho do lado 3:'))
4
5  if((lado1 < lado2 + lado3) and (lado1 > abs(lado2-lado3)) and
6    (lado2 < lado1 + lado3) and (lado2 > abs(lado1-lado3)) and
7    (lado3 < lado2 + lado1) and (lado3 > abs(lado2-lado1))):
8
9      if(lado1 == lado2 and lado2 == lado3):
10         print('triângulo é equilátero')
11     else:
12         if (lado1 != lado2 and lado2 != lado3 and lado1 != lado3):
13             print('triângulo é escaleno')
14         else:
15             print('triângulo é isosceles')
16
17 else:
18     print('triângulo inválido')
```

1 2 3 4 5 6 7



Entrada: 1 1 1
Saída: triângulo equilátero

Caminhos básicos

```
1  lado1 = int(input('tamanho do lado 1:'))
2  lado2 = int(input('tamanho do lado 2:'))
3  lado3 = int(input('tamanho do lado 3:'))
4
5  if((lado1 < lado2 + lado3) and (lado1 > abs(lado2-lado3)) and
6    (lado2 < lado1 + lado3) and (lado2 > abs(lado1-lado3)) and
7    (lado3 < lado2 + lado1) and (lado3 > abs(lado2-lado1))):
8
9      if(lado1 == lado2 and lado2 == lado3):
10         print('triângulo é equilátero')
11     else:
12         if (lado1 != lado2 and lado2 != lado3 and lado1 != lado3):
13             print('triângulo é escaleno')
14         else:
15             print('triângulo é isosceles')
16
17 else:
18     print('triângulo inválido')
```

1

2

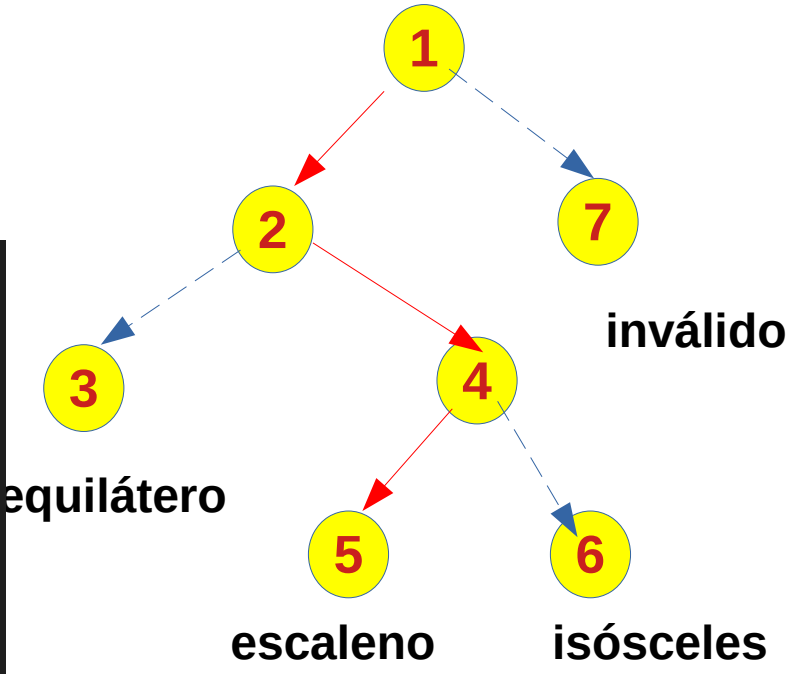
3

4

5

6

7

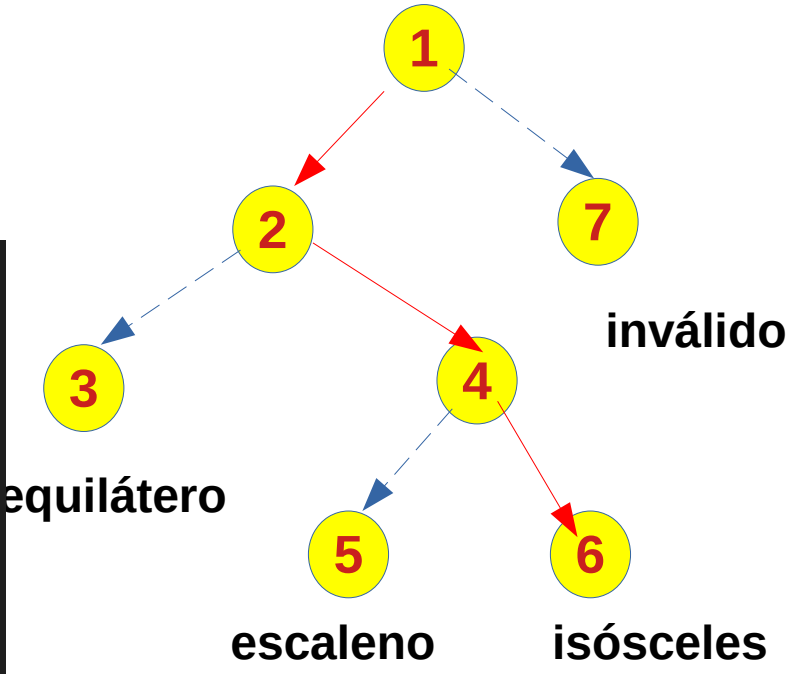


Entrada: 4 2 3
Saída: triângulo escaleno

Caminhos básicos

```
1  lado1 = int(input('tamanho do lado 1:'))
2  lado2 = int(input('tamanho do lado 2:'))
3  lado3 = int(input('tamanho do lado 3:'))
4
5  if((lado1 < lado2 + lado3) and (lado1 > abs(lado2-lado3)) and
6    (lado2 < lado1 + lado3) and (lado2 > abs(lado1-lado3)) and
7    (lado3 < lado2 + lado1) and (lado3 > abs(lado2-lado1))):
8
9      if(lado1 == lado2 and lado2 == lado3):
10         print('triângulo é equilátero')
11     else:
12         if (lado1 != lado2 and lado2 != lado3 and lado1 != lado3):
13             print('triângulo é escaleno')
14         else:
15             print('triângulo é isosceles')
16
17 else:
18     print('triângulo inválido')
```

1 2 3 4 5 6 7



Entrada: 2 3 2
Saída: triângulo isósceles

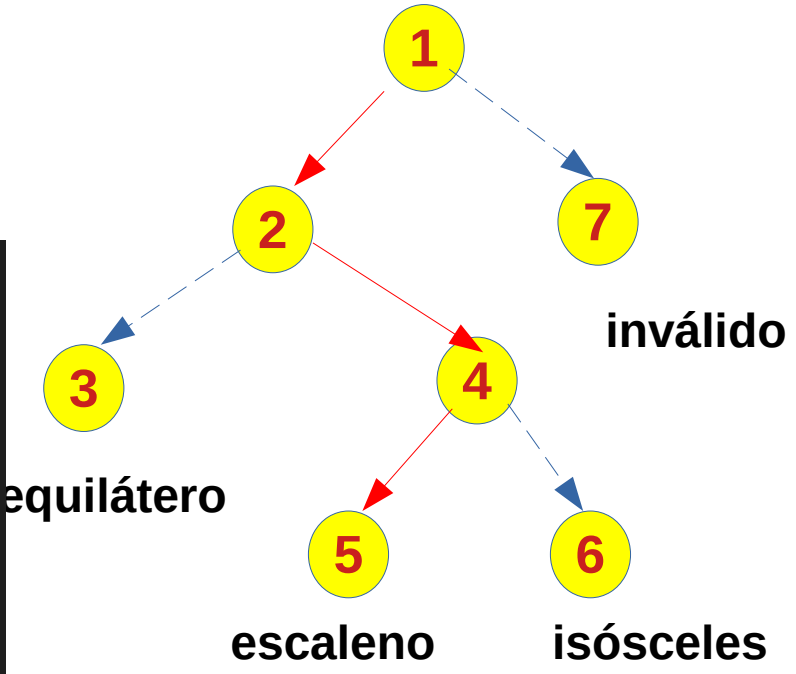
Caminhos básicos

- Cobrir todas as instruções quer dizer que eliminamos todas as possibilidades de erros?
  - Não
  - É possível que uma instrução errada seja executada sem que um defeito se manifeste
  - Exemplo no próximo slide
    - Com uma pequena alteração no código

Caminhos básicos

```
1  lado1 = int(input('tamanho do lado 1:'))
2  lado2 = int(input('tamanho do lado 2:'))
3  lado3 = int(input('tamanho do lado 3:'))
4
5  if((lado1 < lado2 + lado3) and (lado1 > abs(lado2-lado3)) and
6    (lado2 < lado1 + lado3) and (lado2 > abs(lado1-lado3)) and
7    (lado3 < lado2 + lado1) and (lado3 > abs(lado2-lado1))):
8
9      if(lado1 == lado2 and lado2 == lado3):
10         print('triângulo é equilátero')
11     else:
12         if (lado1 != lado2 and lado2 != lado3 and lado1 != lado3):
13             print('triângulo é escaleno')
14         else:
15             print('triângulo é isosceles')
16
17 else:
18     print('triângulo inválido')
```

1 2 3 4 5 6 7



Entrada: 1 3 3
Saída: triângulo isósceles

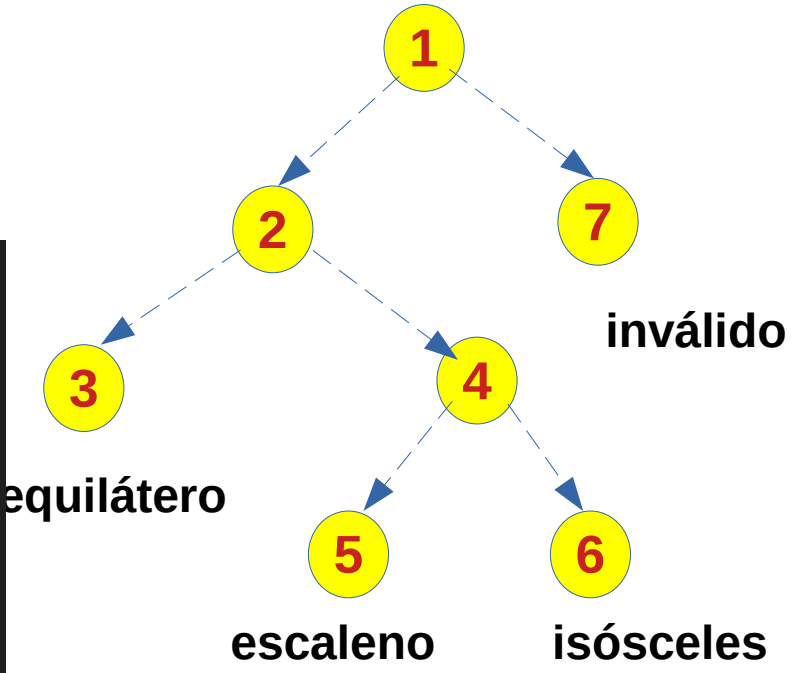
Apesar de a saída acima estar correta, o código possui um erro

Detecção de erro com outro caso de teste
Entrada: 2 3 2
Saída: triângulo escaleno

Caminhos básicos

```
1  lado1 = int(input('tamanho do lado 1:'))
2  lado2 = int(input('tamanho do lado 2:'))
3  lado3 = int(input('tamanho do lado 3:'))
4
5  if((lado1 < lado2 + lado3) and (lado1 > abs(lado2-lado3)) and
6    (lado2 < lado1 + lado3) and (lado2 > abs(lado1-lado3)) and
7    (lado3 < lado2 + lado1) and (lado3 > abs(lado2-lado1))):
8
9      if(lado1 == lado2 and lado2 == lado3):
10         print('triângulo é equilátero')
11     else:
12         if (lado1 != lado2 and lado2 != lado3 and lado1 != lado3):
13             print('triângulo é escaleno')
14         else:
15             print('triângulo é isosceles')
16
17 else:
18     print('triângulo inválido')
```

1 2 3 4 5 6 7



Considere os seguintes casos de Teste:
1)10 9 20 → inválido
2)1 1 1 → equilátero
3)4 2 3 → escaleno
4)1 3 3 → isósceles

Os casos cobrem todos os caminhos básicos. Apesar disso, o erro gerado pela retirada do código destacado não é detectado!

Particionamento em classes de equivalência

- Reduz o número de casos de testes necessários particionando o universo de possíveis valores de entrada em grupos ou classes
- Conjunto de entradas válidas
  - Classe de equivalência válida
- Conjunto de entradas inválidas
  - Classe de equivalência inválida

Particionamento em classes de equivalência

- Exemplo
  - Entrada aceita valores entre 10 e 100
  - Classe de equivalência válida
    - Entre 10 e 100
  - Classes de equivalência inválida
    - < 10
    - > 100

Particionamento em classes de equivalência

- Exemplo
  - Entrada deve iniciar com uma letra e conter apenas letras ou dígitos. A cadeia deve conter até 6 caracteres
  -

Condições	Classes válidas	Classes inválidas
Inicia com uma letra	Sim (1)	Não (2)
Só contém letras ou dígitos	Sim (3)	Não (4)
Tamanho t da entrada	0 < t <= 6 (5)	T = 0 ou t > 6 (6)

Particionamento em classes de equivalência

- Exemplo
  - Entrada deve iniciar com uma letra e conter apenas letras ou dígitos. A cadeia deve conter até 6 caracteres
  - Conjunto de casos de teste
    - Teste1 (1,3,5), bla (1,3,5)
    - 1teste (2, 3, 5), test* (1, 4, 5), teste12 (1,3,6)

Análise do valor limite

- Complementa o particionamento de equivalência
  - Os limites de uma classe são propícias a erros
  - Limites: valores imediatamente acima ou abaixo dos extremos de cada classe
  - Exemplo: valor válido entre 10 e 100
  - Casos de teste:
    - 9 (inválido)
    - 10 (válido)
    - 100 (válido)
    - 101 (inválido)

Análise do valor limite

- Dada uma região de valores válidos [min,max], usados os seguintes valores limite
  - Min-1 (região inválidos < min)
  - Min (região válidos)
  - Max (região válidos)
  - Max + 1 (região inválidos > max)

Análise do valor limite

- Exemplo: valor esperado ‘n’
  - Casos de teste:
    - n-1 (inválido)
    - n (válido)
    - n+1 (inválido)

Análise do valor limite

- Imites com mais de um intervalo válidos
  - Exemplo: valor válido entre ‘c’ e ‘f’ ou entre ‘j’ e ‘m’
  - Nesse caso, devemos testar os limites dos dois intervalos
    - Primeiro intervalo
      - b (inválido), c (válido), f (válido), g (inválido)
    - Segundo intervalo
      - i (inválido), j (válido), m (válido), n (inválido)

Valores especiais

- Valores que frequentemente estão associados à erros
- Exemplos
  - Tipos inteiros
    - 0
  - Data
    - Número de dias em fevereiro em um ano bissexto
  - Hora
    - 0 e 24 devem ser sempre testados como valores limite inferior e superior, além dos valores de restrição da entrada

Duvidas?

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PERNAMBUCO