

# **ACH 2028**

## **Qualidade de Software**

### **Aula 08 - Teste de Software: Técnica Estrutural ou Caixa-Branca**

Prof. Marcelo Medeiros Eler  
[marceloeler@usp.br](mailto:marceloeler@usp.br)

# Objetivos

Apresentar a técnica de teste estrutural ou caixa-branca

Mostrar exemplos de casos de teste criados para satisfazer os principais critérios da técnica de teste estrutural

Propor exercícios para exercitar a técnica de teste estrutural

# Como encontrar todas as falhas de um software?

Um conjunto de casos de teste com **qualidade ótima** é capaz de revelar todas as falhas de um software

Para encontrar todas as falhas e garantir que o software está livre de defeitos é necessário **testá-lo em todas as condições** e com **todos os dados de entrada possíveis**

# Como encontrar todas as falhas de um software?

Entretanto, as condições, os cenários, os dados de entrada e suas combinações tendem a ser **infinitos ou muito grandes**

Portanto, criar um conjunto de casos de teste ótimo é geralmente **impossível ou impraticável**

# Uma pergunta importante neste cenário

Como criar um conjunto de casos de teste que seja:

- Finito e factível?
- Capaz de revelar o maior número de falhas perceptíveis?
- Capaz de revelar as falhas mais críticas ou relevantes do software?

Esta pergunta esconde algumas outras:

- Como seleccionar cenários/dados para criar bons casos de teste?
- Quantos testes são necessários para testar um software ou parte dele?
- Como saber se os testes criados são suficientes?

# Técnicas e critérios de teste

Um bom caso de teste é aquele que tem uma alta probabilidade de revelar uma falha ainda não descoberta

Portanto, técnicas e critérios de teste foram propostos para orientar o testador na tarefa de derivar os casos de teste com base nos artefatos do projeto (especificação, código, modelos, etc):

- Teste Adhoc
- Teste Exploratório
- Teste Funcional ou Caixa-Preta
- Teste Estrutural ou Caixa-Branca
- Teste Baseado em Erros

# Cr terios de teste

Definem requisitos de teste que devem ser satisfeitos pelos casos de teste

Um crit rio   satisfeito somente quando todos os requisitos que ele define s o satisfeitos

Ajudam a responder  s quest es:

- Como selecionar valores de entrada para criar bons casos de teste?
- Quantos casos de teste devem ser criados?
- Quando parar de testar?

Teste funcional ou caixa-preta (aula anterior)





# Teste funcional ou caixa-preta (aula anterior)

Tem o objetivo de verificar se uma funcionalidade, operação, função, método, classe, programa está de acordo com os requisitos especificados

Os casos de teste são definidos com base na especificação do software (descrições, casos de uso, requisitos, diagramas, etc)

Exemplos de critérios desta técnica:

- Classes de equivalência
- Análise de valor-limite
- Tabela de decisão

# Teste funcional ou caixa-preta (aula anterior)

No teste funcional, sabemos que o conjunto de casos de teste são suficientes (ou possuem qualidade adequada) quando eles:

- Exercitam todas as classes de equivalência conforme determina o critério
- Exercitam todos os valores-limite
- Exercitam todas as combinações expressas por meio de tabelas de decisão

# Teste funcional ou caixa-preta (aula anterior)

Entretanto, os casos de testes criados com a técnica funcional, ainda que satisfaçam todos os critérios, podem não ser o suficientes.

Uma possível limitação desta técnica é a seguinte:

- Eu não sei se existem estruturas de código que não foram executadas pelos casos de teste, pois só consigo garantir que eu testei aquilo que está especificado
- Se alguma regra ou funcionalidade foi implementada e não está completamente especificada, pode haver condições que precisam ser testadas ainda

# Teste estrutural ou teste caixa-branca

Também conhecido como teste caixa-branca

Tem o objetivo de verificar se os testes exercitam todas as estruturas de programação

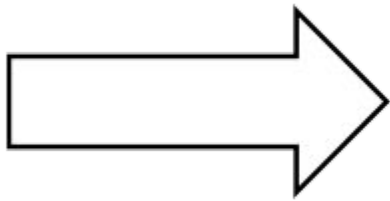
Seu principal objetivo é encontrar falhas no software por meio da análise de sua estrutura interna

# Teste estrutural ou teste caixa-branca

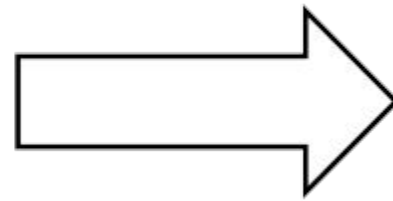
Também conhecido como teste caixa-branca

Tem o objetivo de verificar se os testes exercitam todas as estruturas de programação

Seu principal objetivo é encontrar falhas no software por meio da análise de sua estrutura interna



```
if (x<=0)
    throw new Exception("...");
else
    z = y/x;
return z;
```



# Cobertura de código

Na prática, a maioria das pessoas verifica duas métricas para saber o quanto de seus códigos foram exercitados pelos casos de teste:

- Linhas de código executadas
- Branches (desvios de fluxo)

Quando analisadas em termos de porcentagem de linhas de código ou branches executadas, essas métricas são comumente chamadas de “cobertura de código”

O termo cobertura de código, portanto, refere-se à porcentagem do código que foi executada (de acordo com cada métrica)

# Um pouco de teoria: critérios do teste estrutural

Os critérios da técnica de teste estrutural podem ser divididos em duas categorias:

- Critérios de fluxo de controle
- Critérios de fluxo de dados

O enfoque desta aula será critérios de fluxo de controle bem conhecidos:

- Todos os caminhos
- Todos os caminhos básicos
- Todos os nós
- Todas as arestas

# Um pouco de teoria: estruturas de representação

É comum que no teste caixa-branca o elemento de programação sob teste seja representado por abstrações que facilitam entender sua estrutura e derivar requisitos de teste

Uma estratégia comum neste cenário é utilizar Grafos de Fluxo de Controle (GFC)

- Os nós do grafo representam blocos de instruções e as arestas representam os possíveis desvios de fluxo
- Blocos de instruções devem ser identificados de tal forma que não existam desvios de fluxo entre sua primeira e sua última instrução, desconsiderando situações de exceção.



# Grafo de Fluxo de Controle

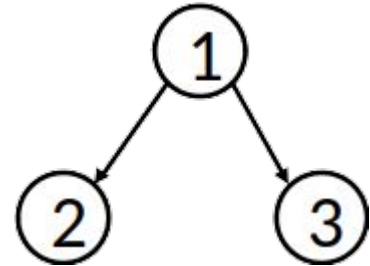
Exemplos de estruturas de programação e respectivo GFC

```
int p = Integer.valueOf(args[0]);  
int q = Integer.valueOf(args[1]);  
int n = p*q;  
if (n>18)  
    System.out.println("OK");  
else  
    System.out.println("ERRO");
```

# Grafo de Fluxo de Controle

Exemplos de estruturas de programação e respectivo GFC

```
1 { int p = Integer.valueOf(args[0]);  
   int q = Integer.valueOf(args[1]);  
   int n = p*q;  
2 { if (n>18)  
   System.out.println("OK");  
   else  
3 { System.out.println("ERRO");
```



# Grafo de Fluxo de Controle

Exemplos de estruturas de programação e respectivo GFC

```
int p = Integer.valueOf(args[0]);  
int q = Integer.valueOf(args[1]);  
if ((p>=0) && (q>=0))  
    System.out.println("OK");  
else {  
    System.out.println("ERRO");  
}
```

# Grafo de Fluxo de Controle

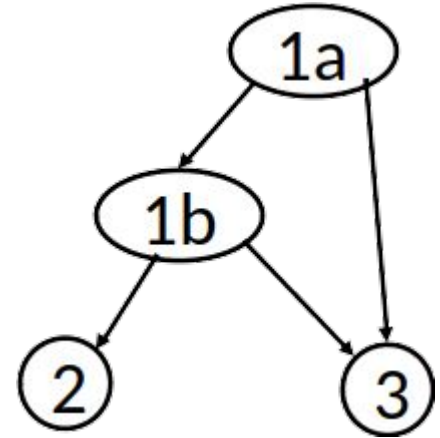
Exemplos de estruturas de programação e respectivo GFC

```
1a {  
    int p = Integer.valueOf(args[0]);  
    int q = Integer.valueOf(args[1]);  
    if ((p>=0) && (q>=0)) 1b  
2  {  
    System.out.println("OK");  
    else {  
3  {  
    System.out.println("ERRO");  
    }
```

# Grafo de Fluxo de Controle

Exemplos de estruturas de programação e respectivo GFC

```
1a {  
    int p = Integer.valueOf(args[0]);  
    int q = Integer.valueOf(args[1]);  
    if ((p>=0) && (q>=0)) 1b  
2 - System.out.println("OK");  
    else {  
3 - System.out.println("ERRO");  
    }
```



# Grafo de Fluxo de Controle

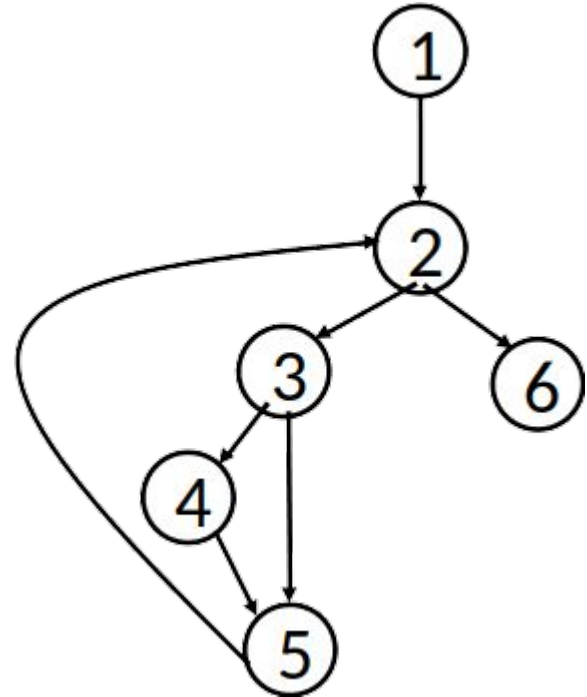
Exemplos de estruturas de programação e respectivo GFC

```
int cont=0;
int p = Integer.valueOf(args[0]);
int q = Integer.valueOf(args[1]);
int i = p;
while (i<=q){
    if (i % 2 == 0)
        cont++;
    i++;
}
System.out.println(cont + "pares");
```

# Grafo de Fluxo de Controle

Exemplos de estruturas de programação e respectivo GFC

```
1 { int cont=0;  
  int p = Integer.valueOf(args[0]);  
  int q = Integer.valueOf(args[1]);  
  int i = p;  
2 while (i<=q){  
3   if (i % 2 == 0)  
4     cont++;  
5   i++;  
  }  
6 System.out.println(cont + "pares");
```



# Exemplo 1 - Triângulo

Escreva casos de teste para testar um algoritmo que recebe como entrada três números inteiros positivos que representam os tamanhos dos três lados de um triângulo, e retorna o tipo de triângulo formado (Equilátero, isósceles ou escaleno).

Se algum dos lados for negativo, o algoritmo deve informar que houve um erro porque um dos lados não tem valor válido

Se um dos lados tiver tamanho maior do que a soma dos outros três lados, então o algoritmo deve informar que esses valores não são suficientes para formar um triângulo



# Exemplo 1 - Triângulo

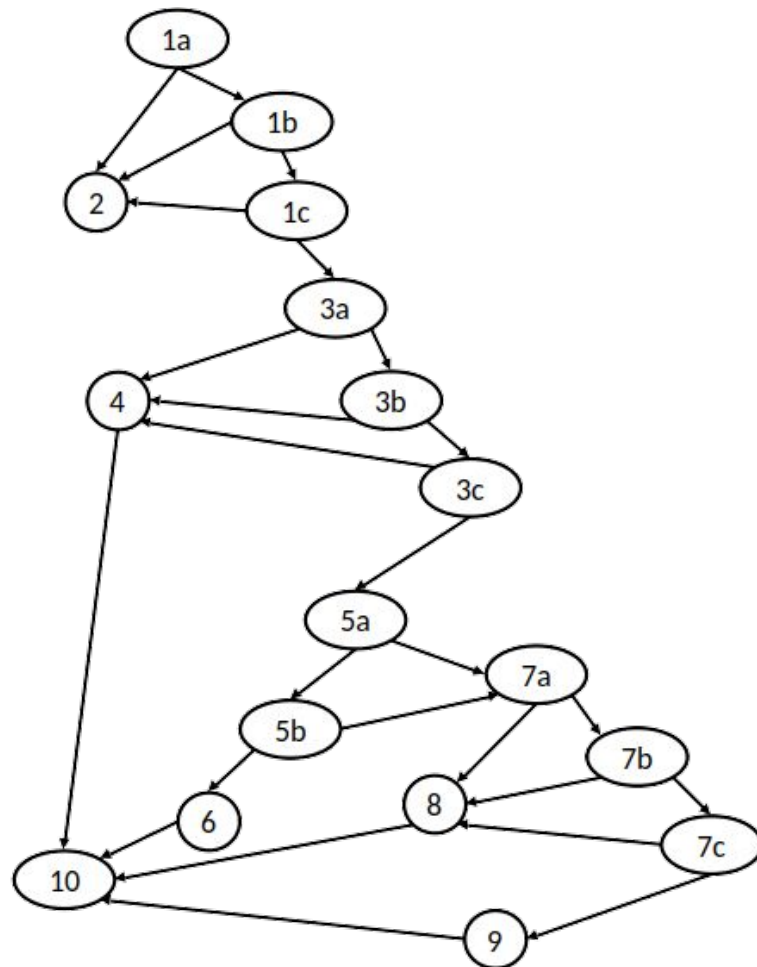
```
public static String classificaTriangulo(int LA, int LB, int LC) throws LadoInvalidoException{
    String resposta="";
    if (LA<=0 || LB <=0 || LC <=0)
        throw new LadoInvalidoException("lado inválido");

    if ( (LA >= LB + LC) || (LB >= LA + LC) || (LC >= LA + LB))
        resposta = "NAO FORMA TRIANGULO";
    else {
        if (LA==LB && LB==LC)
            resposta = "EQUILATERO";
        else {
            if (LA==LB || LB==LC || LA==LC)
                resposta = "ISOSCELES";
            else
                resposta = "ESCALENO";
        }
    }
    return resposta;
}
```

# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

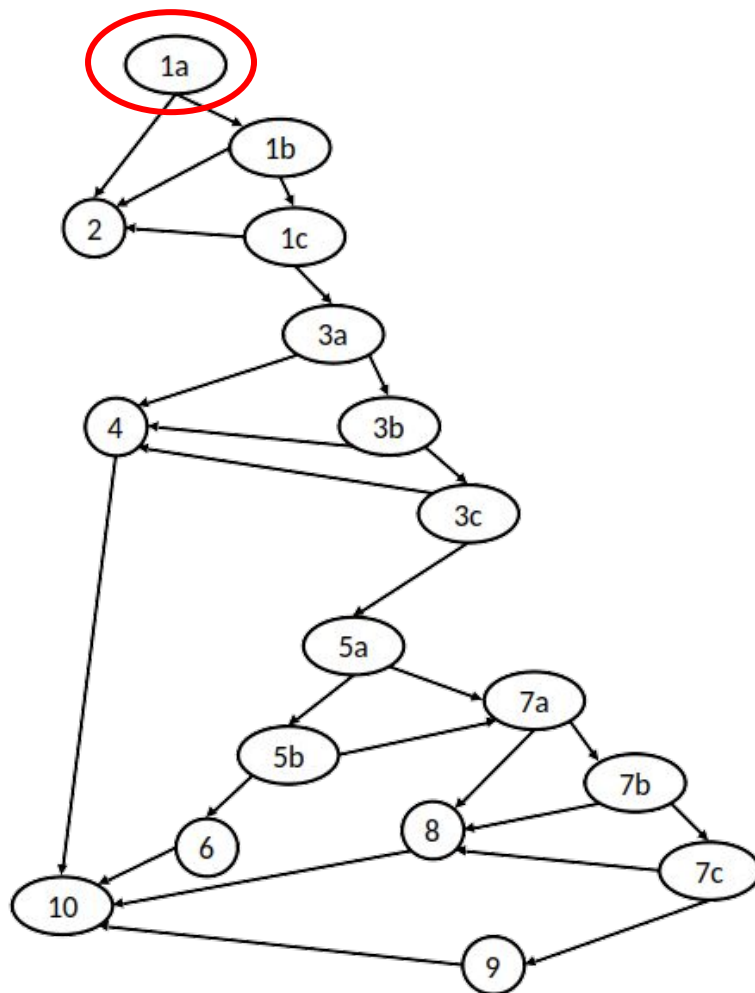
if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```



# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

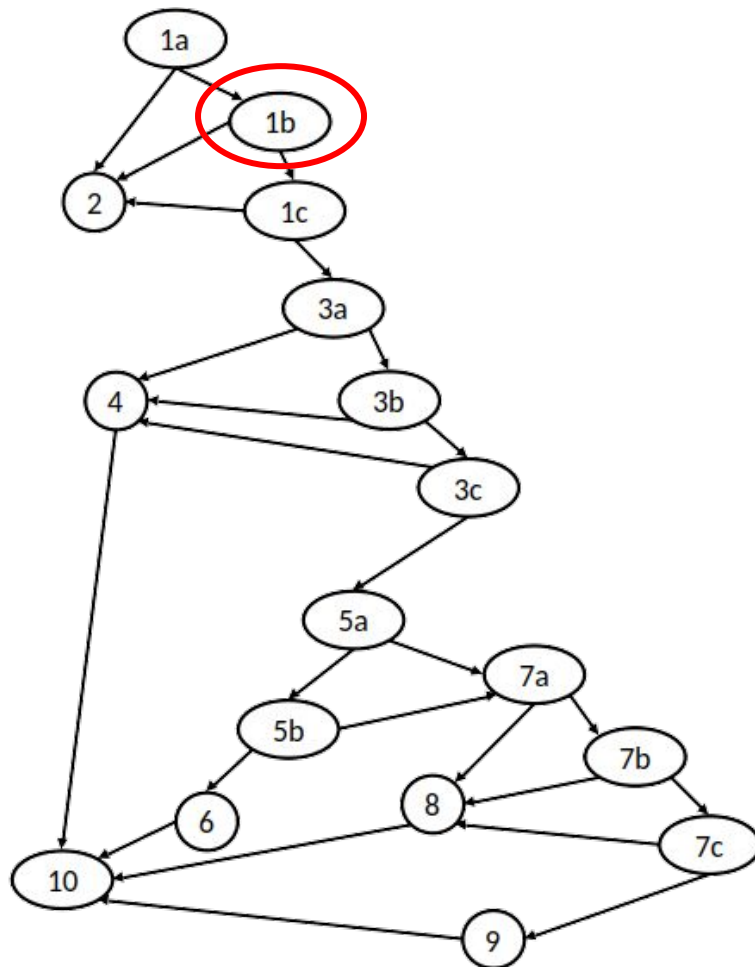
if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```



# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB<=0 || LC<=0)
    throw new LadoInvalidoException("inválido");

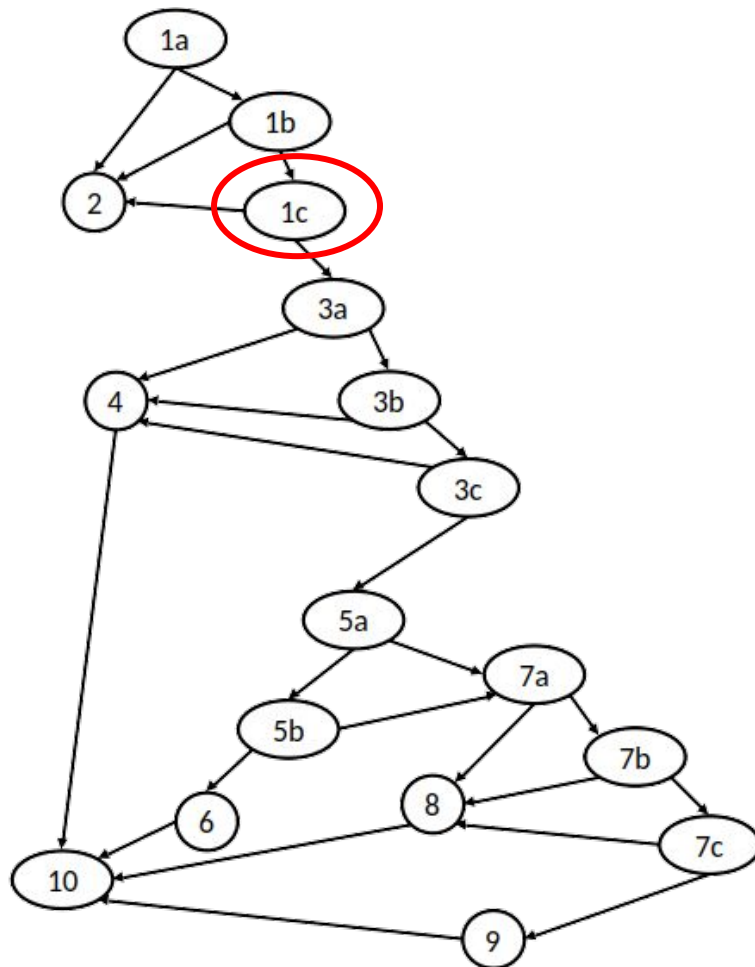
if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB) )
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```



# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

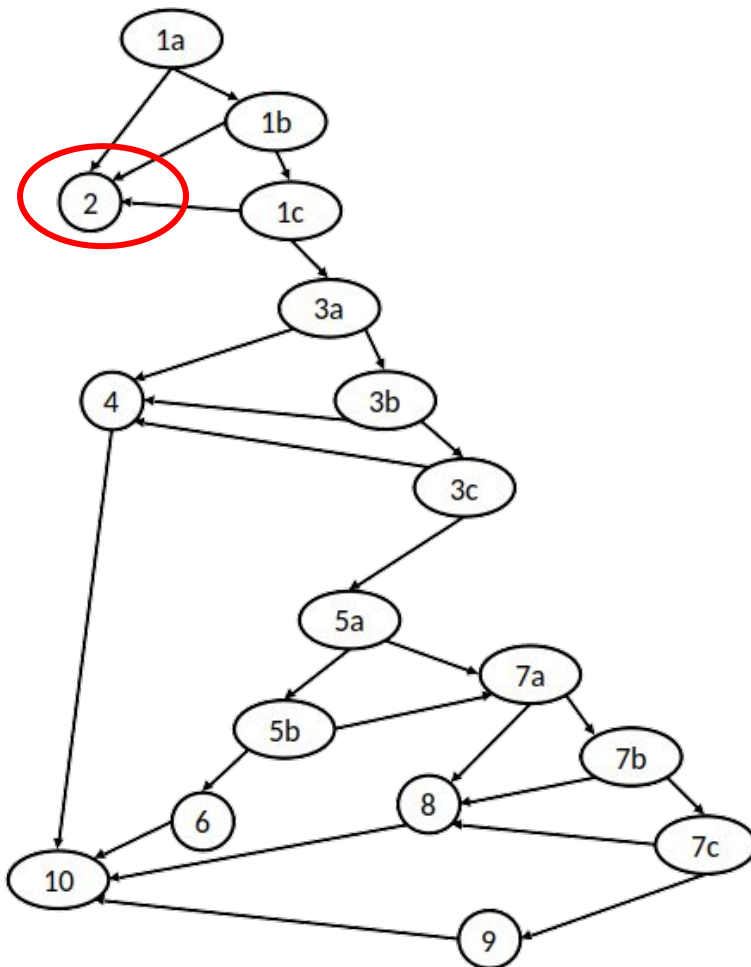
if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```



# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("invalido");

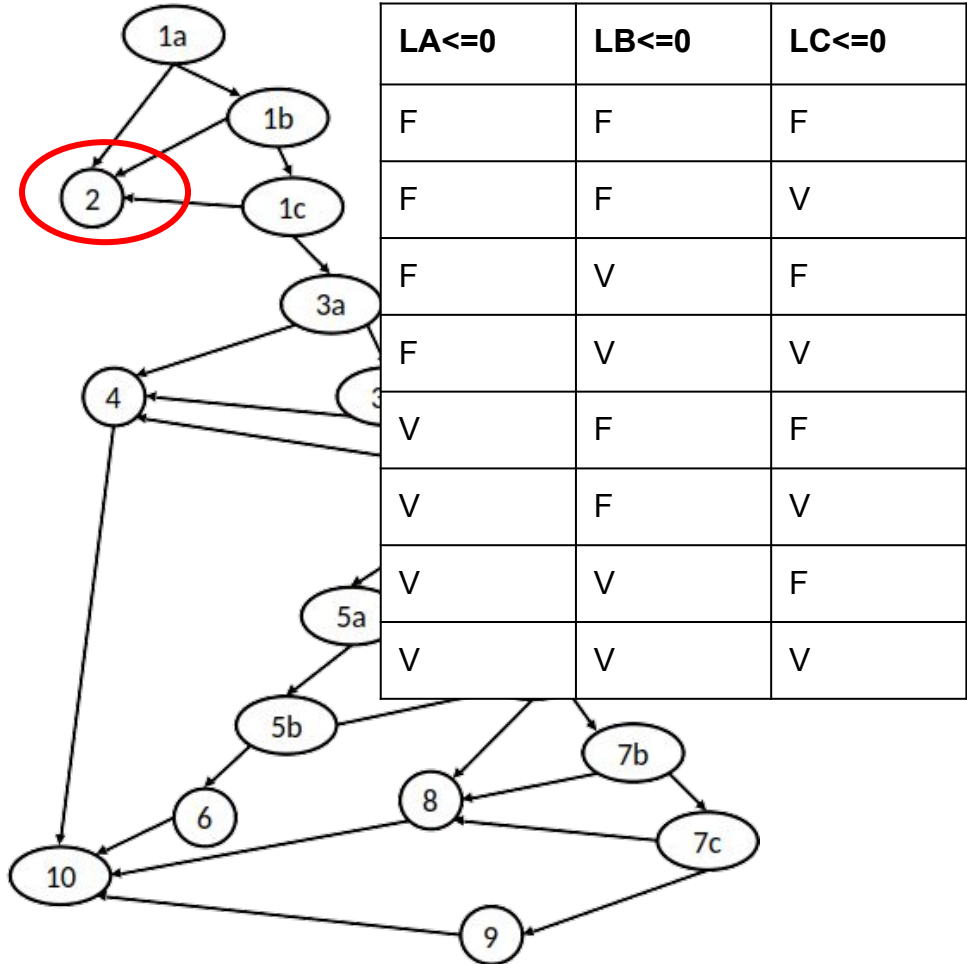
if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```



# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("invalido");

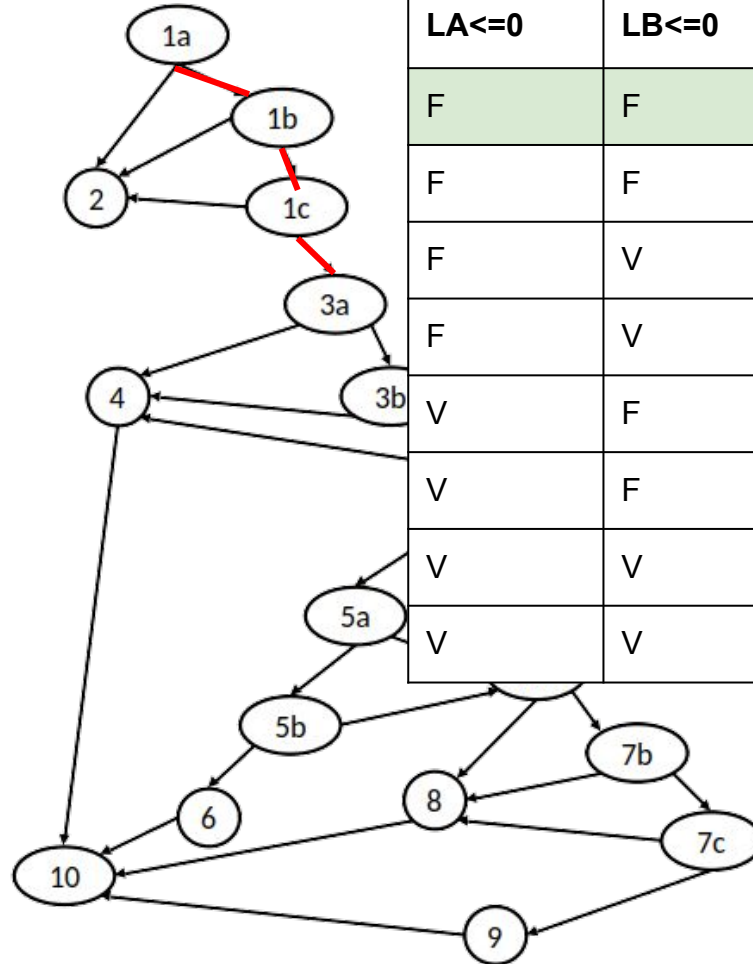
if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```





# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("invalido");
if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```



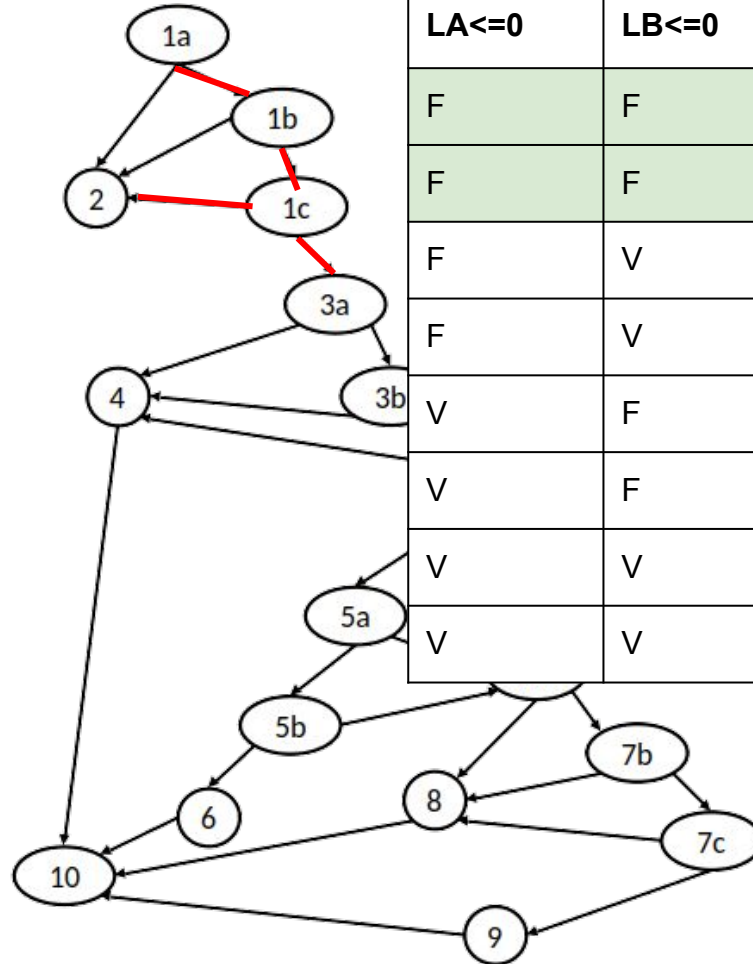
LA<=0	LB<=0	LC<=0
F	F	F
F	F	V
F	V	F
F	V	V
V	F	F
V	F	V
V	V	F
V	V	V



# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("invalido");

if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```

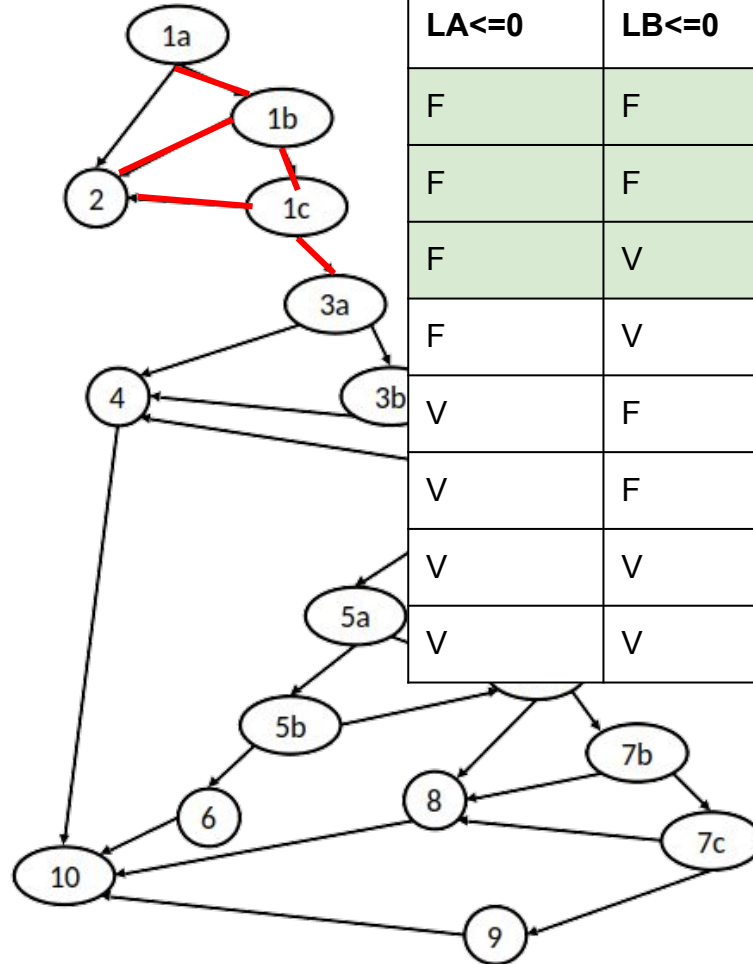


LA<=0	LB<=0	LC<=0
F	F	F
F	F	V
F	V	F
F	V	V
V	F	F
V	F	V
V	V	F
V	V	V

# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("invalido");

if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```

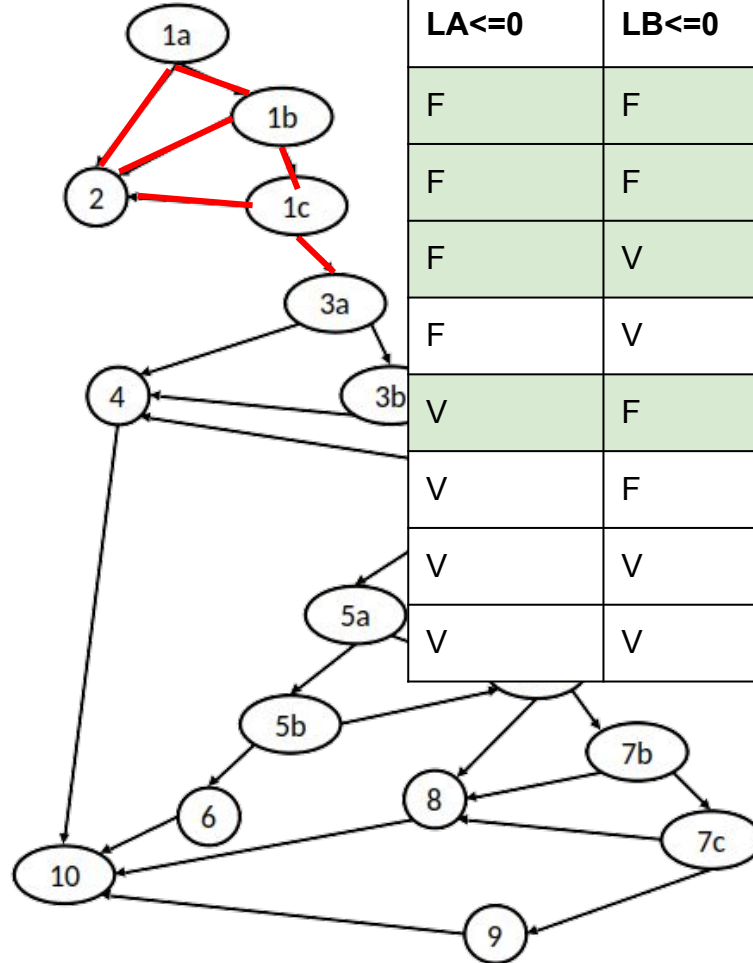


LA<=0	LB<=0	LC<=0
F	F	F
F	F	V
F	V	F
F	V	V
V	F	F
V	F	V
V	V	F
V	V	V

# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("invalido");

if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```

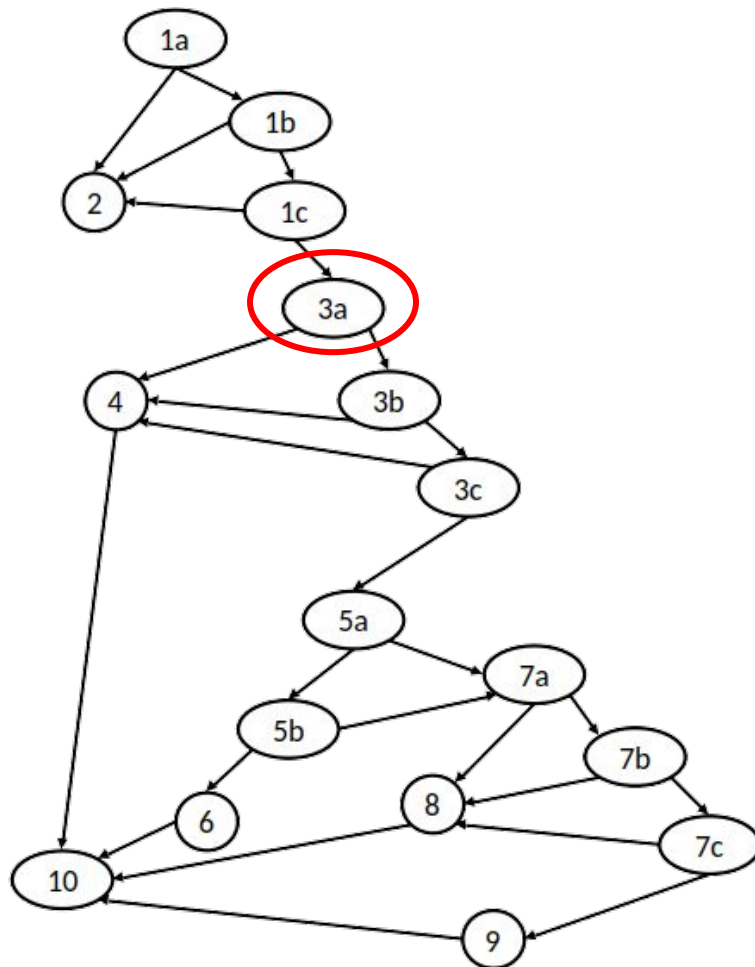


LA<=0	LB<=0	LC<=0
F	F	F
F	F	V
F	V	F
F	V	V
V	F	F
V	F	V
V	V	F
V	V	V

# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

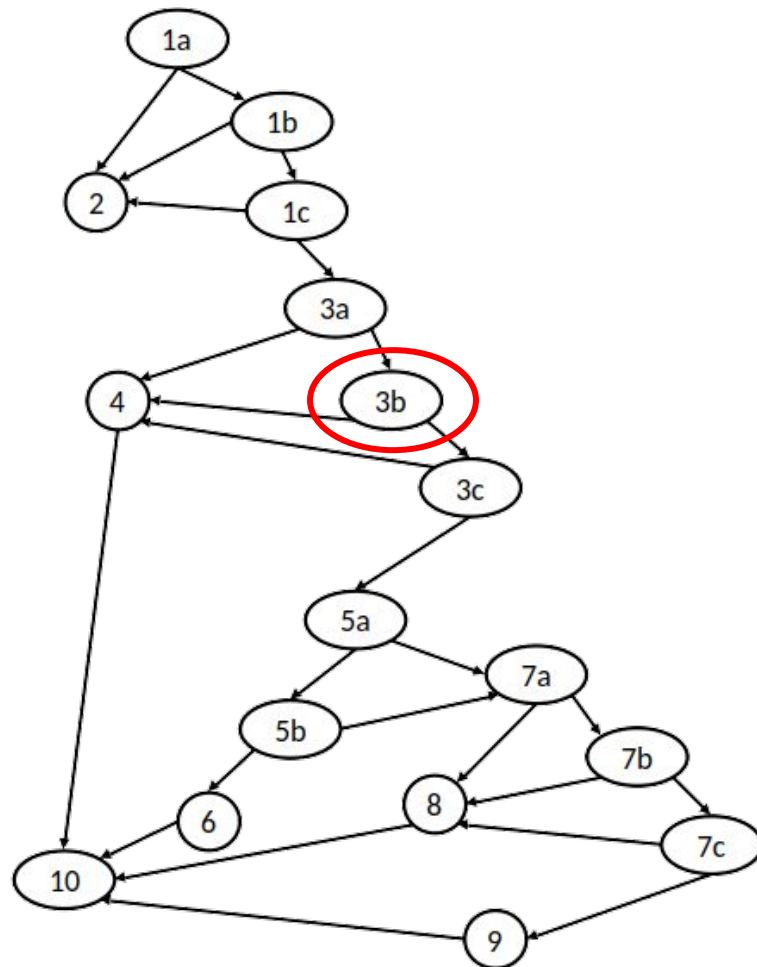
if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```



# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

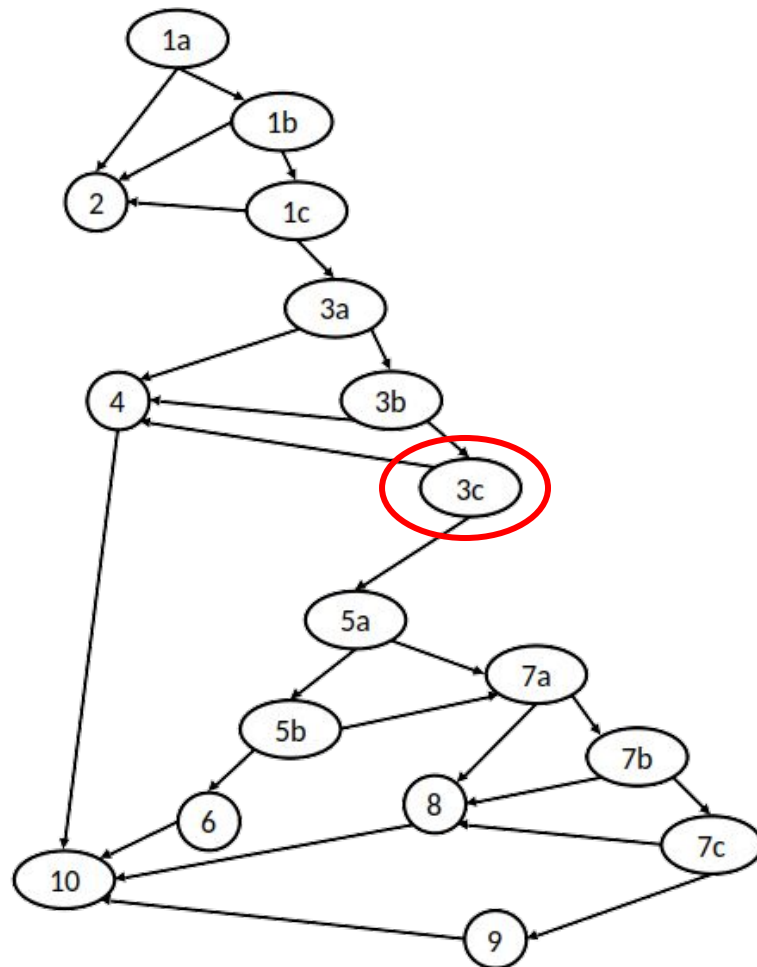
if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```



# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

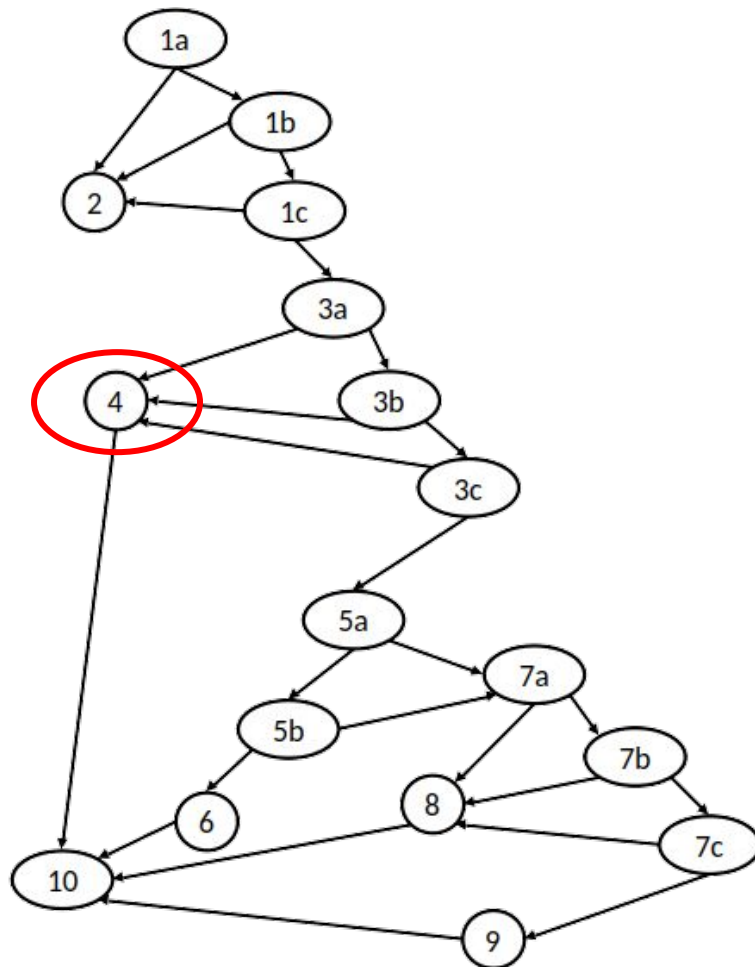
if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```



# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```

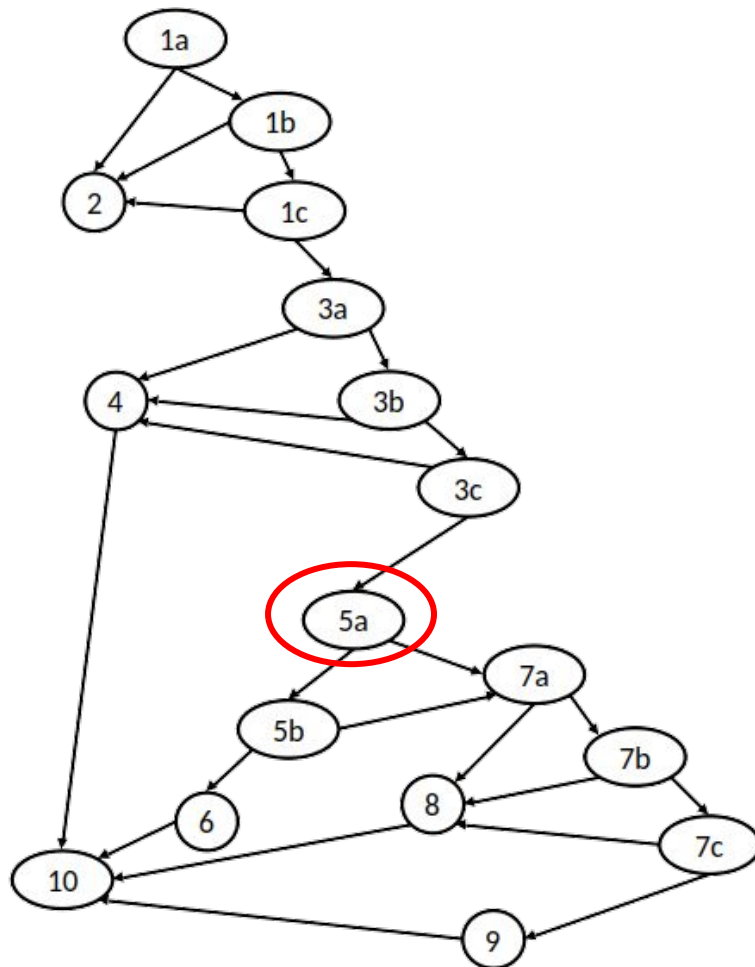




# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```

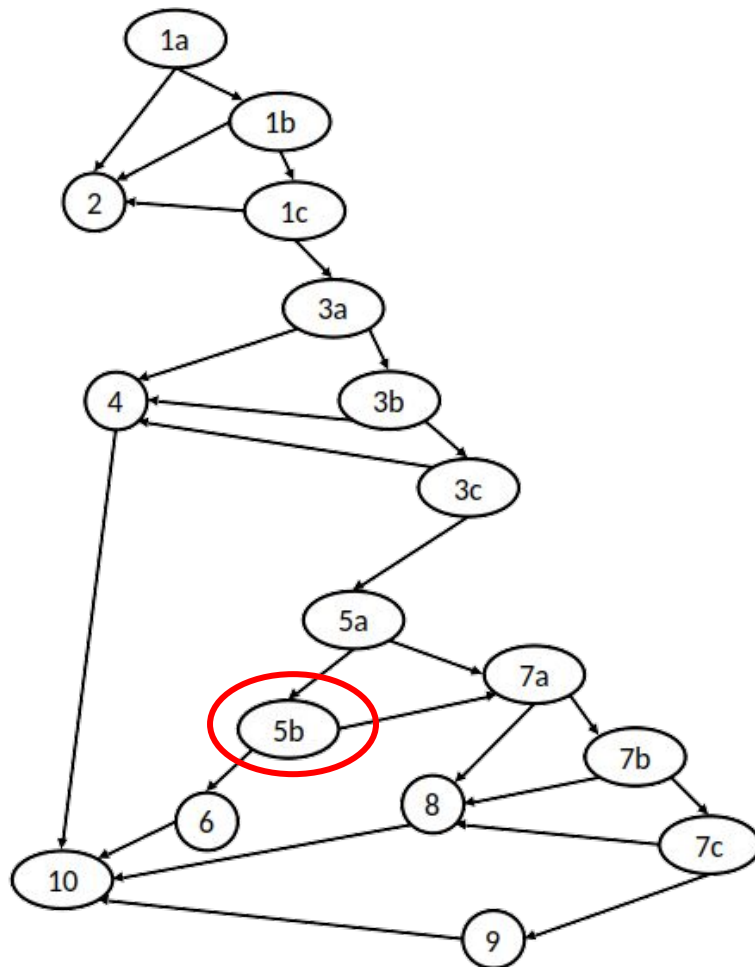




# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

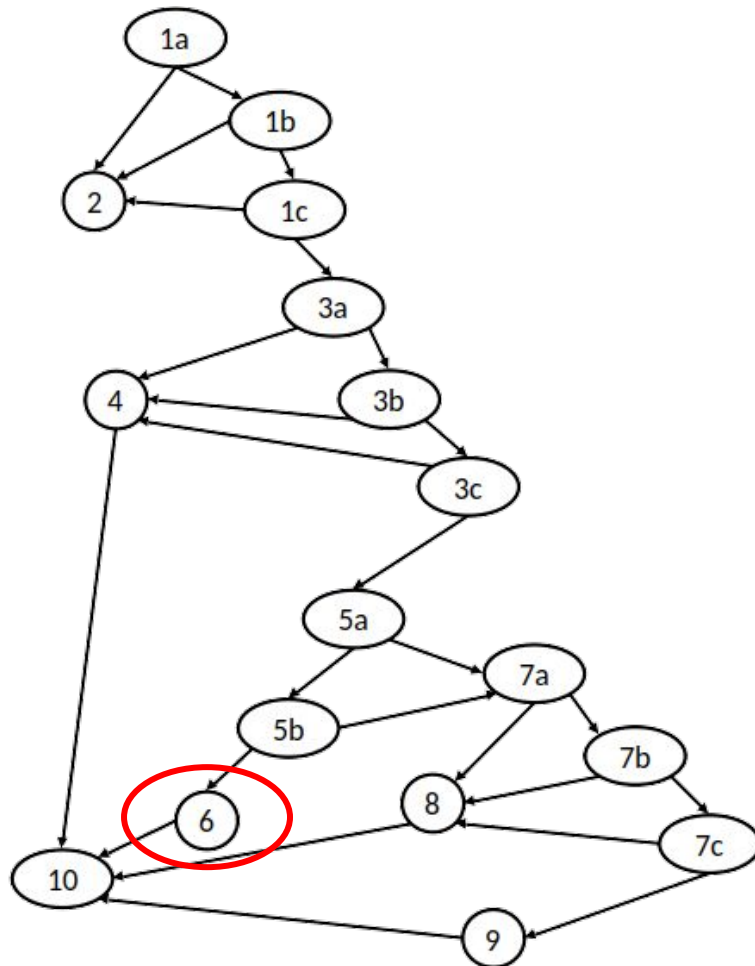
if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```



# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

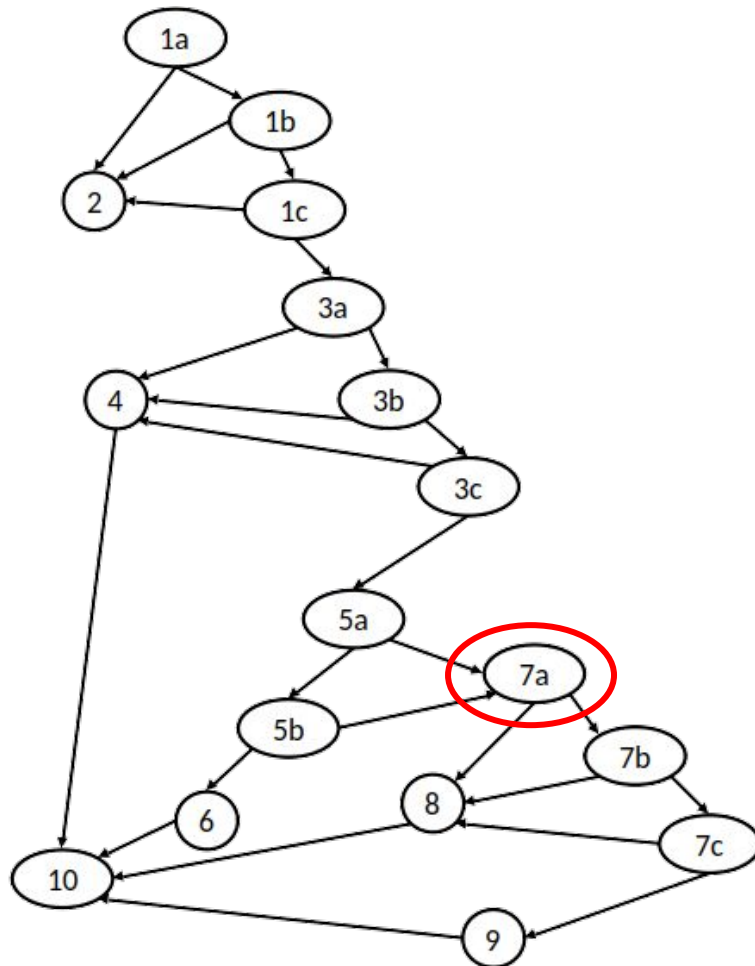
if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```



# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

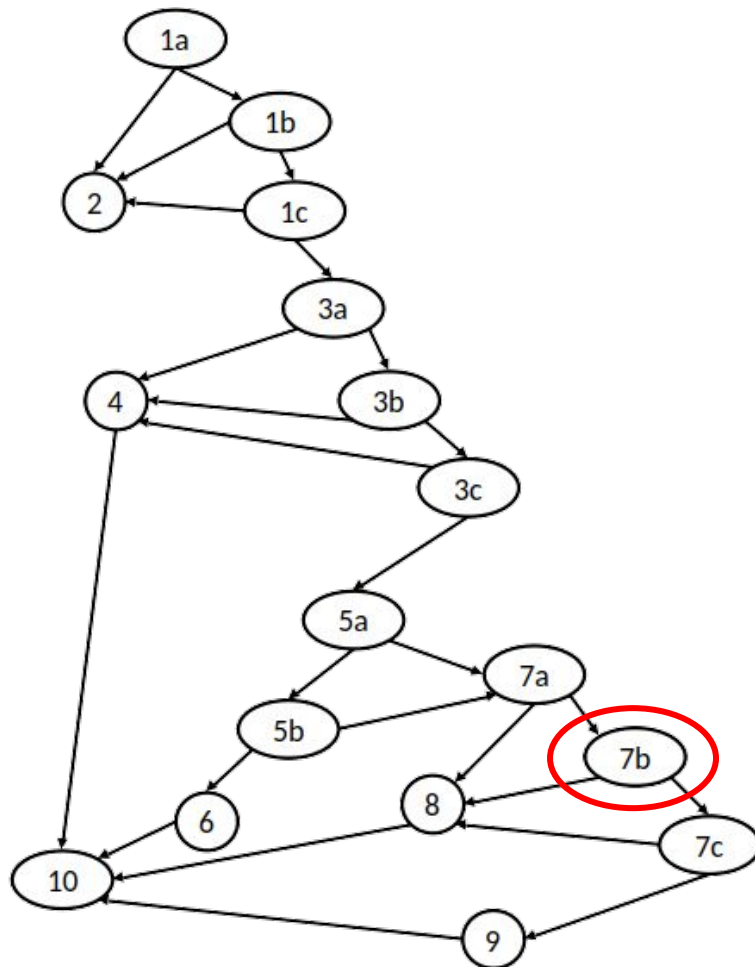
if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```



# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

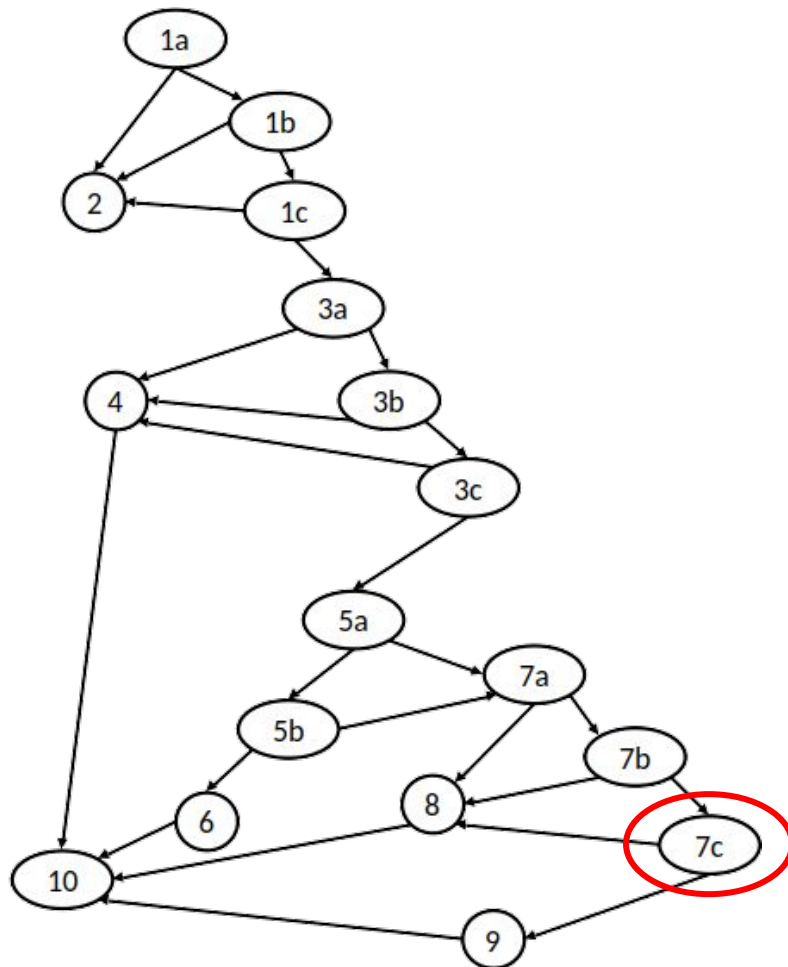
if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```



# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

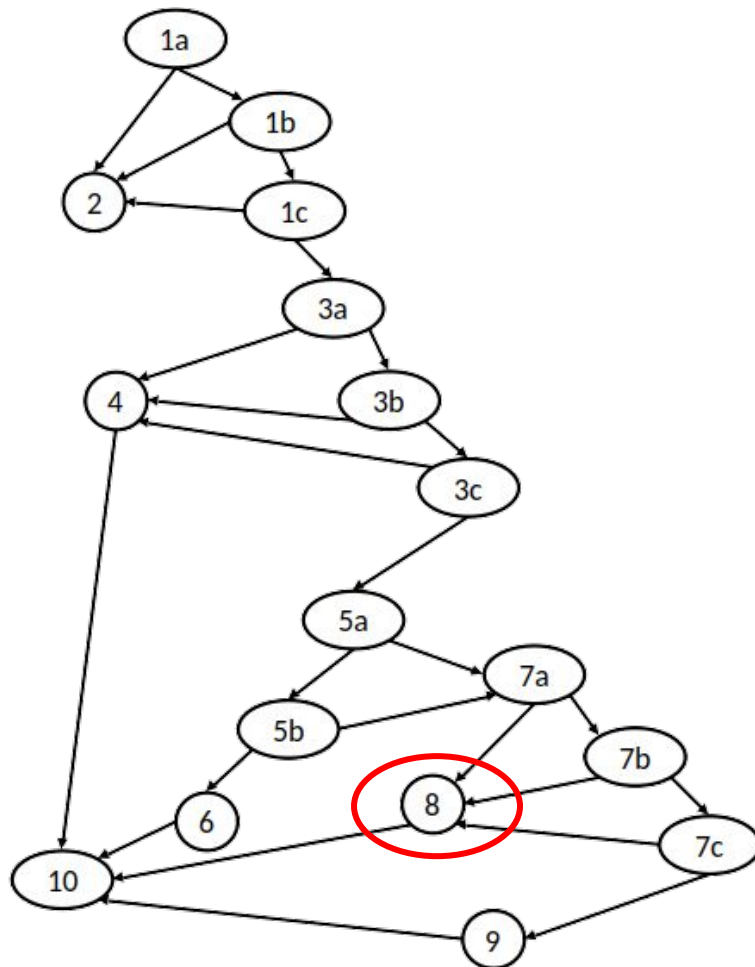
if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```



# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

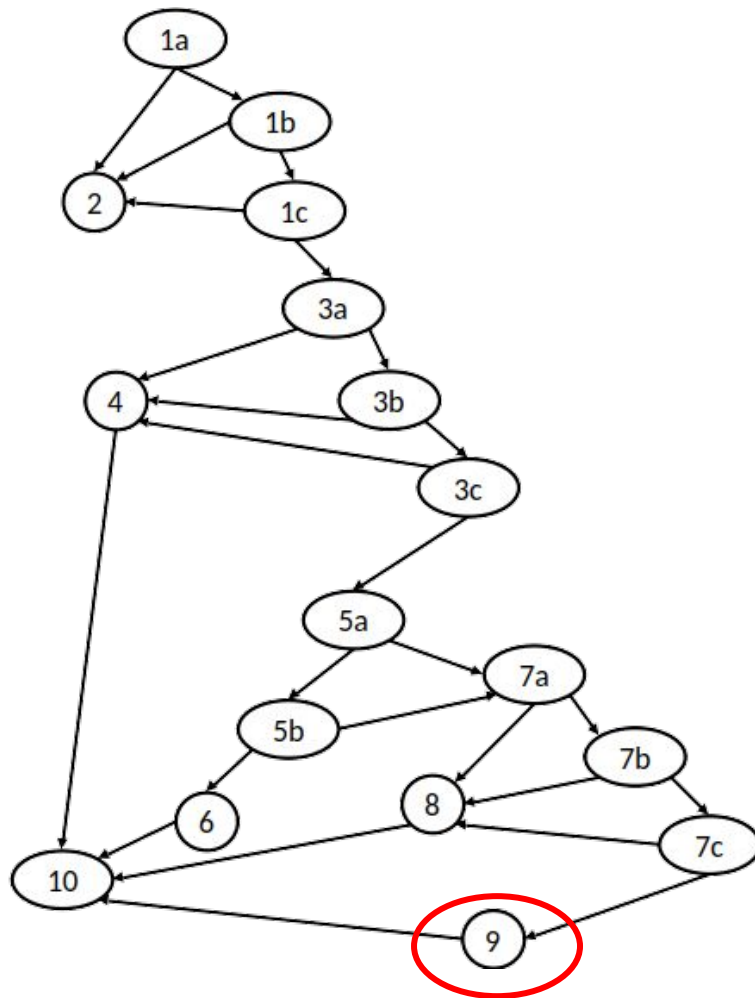
if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```



# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```



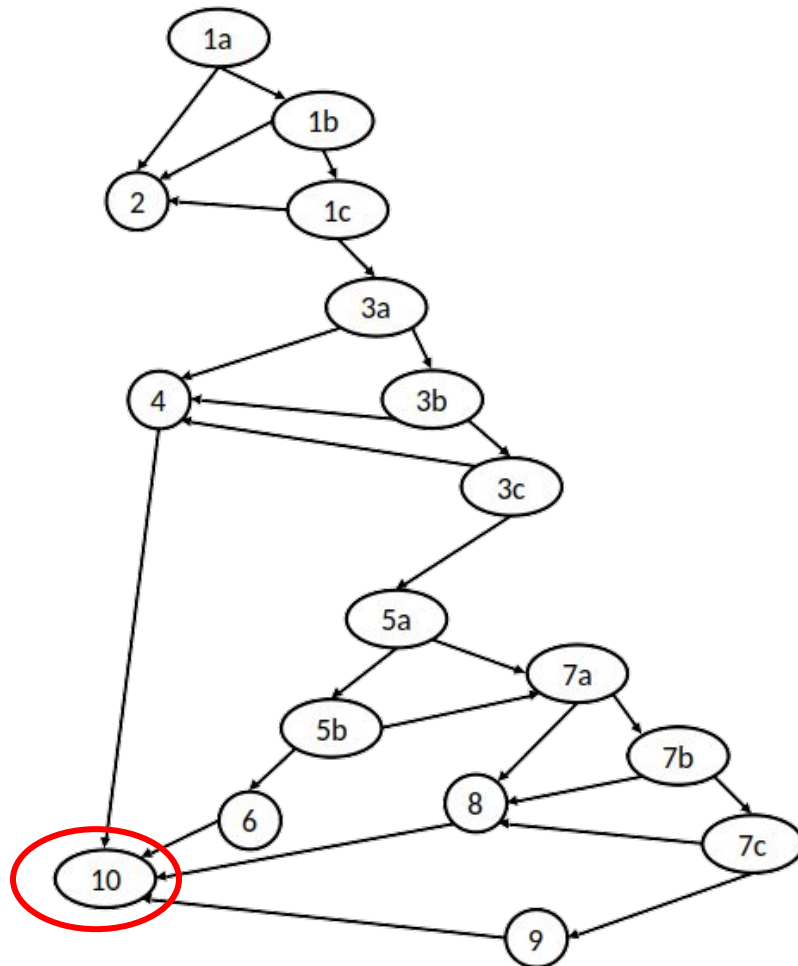


# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}

return resposta;
```

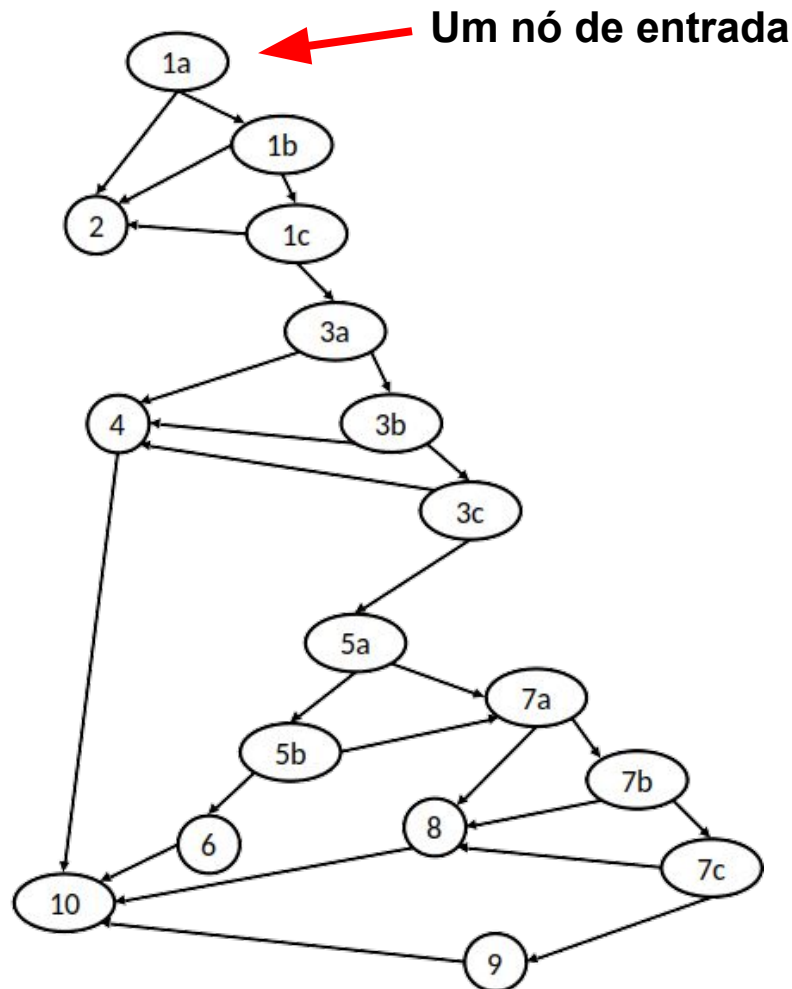




# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```

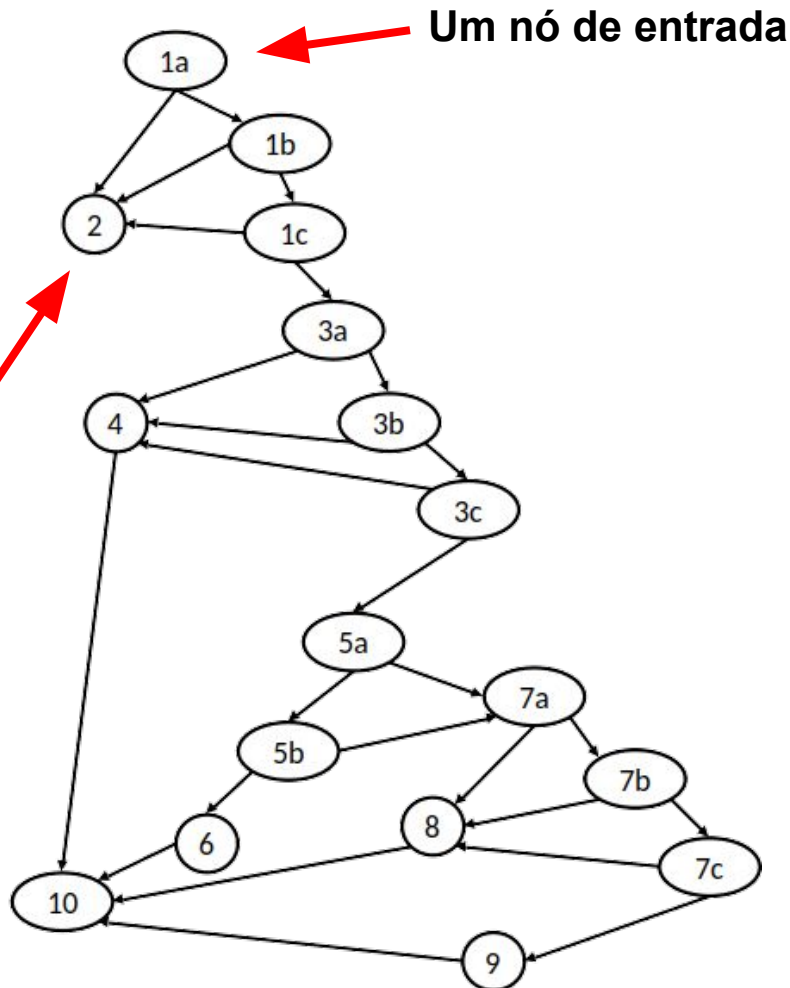


# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```

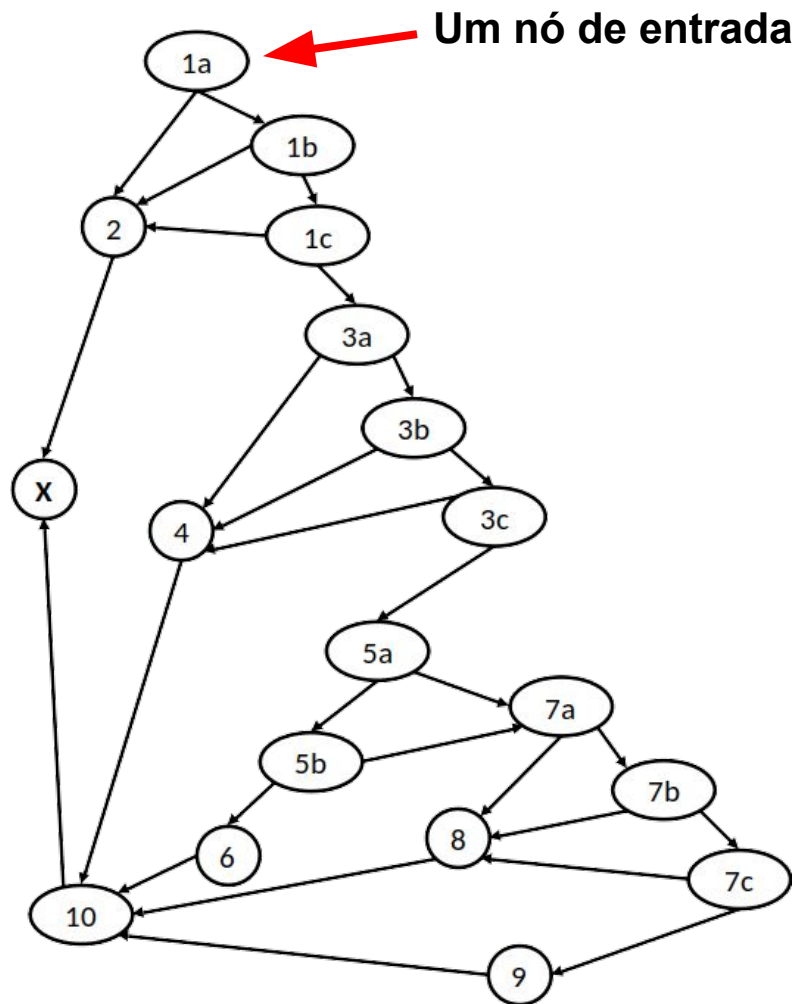
Dois nós de saída



# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```

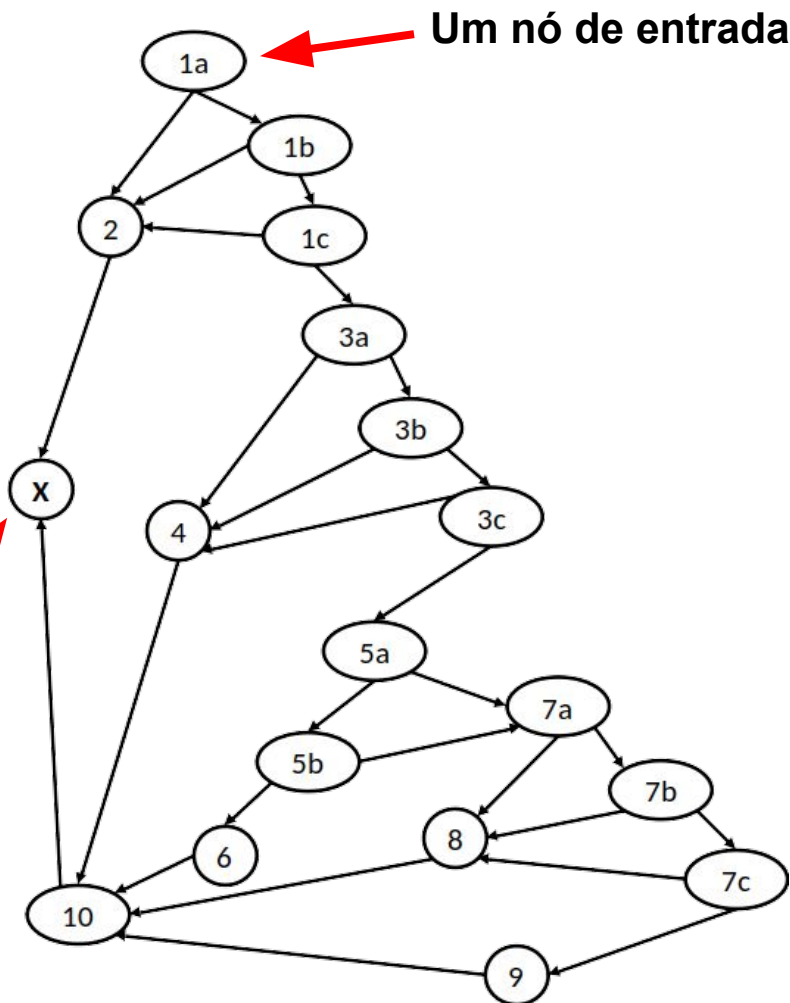


# Exemplo 1 - Triângulo

```
classificaTriangulo(int LA, int LB, int LC)
String resposta="";
if (LA<=0 || LB <=0 || LC <=0)
    throw new LadoInvalidoException("inválido");

if ( (LA>=LB+LC) || (LB>=LA+LC) || (LC>=LA+LB))
    resposta = "NAO FORMA TRIANGULO";
else {
    if (LA==LB && LB==LC)
        resposta = "EQUILATERO";
    else {
        if (LA==LB || LB==LC || LA==LC)
            resposta = "ISOSCELES";
        else
            resposta = "ESCALENO";
    }
}
return resposta;
}
```

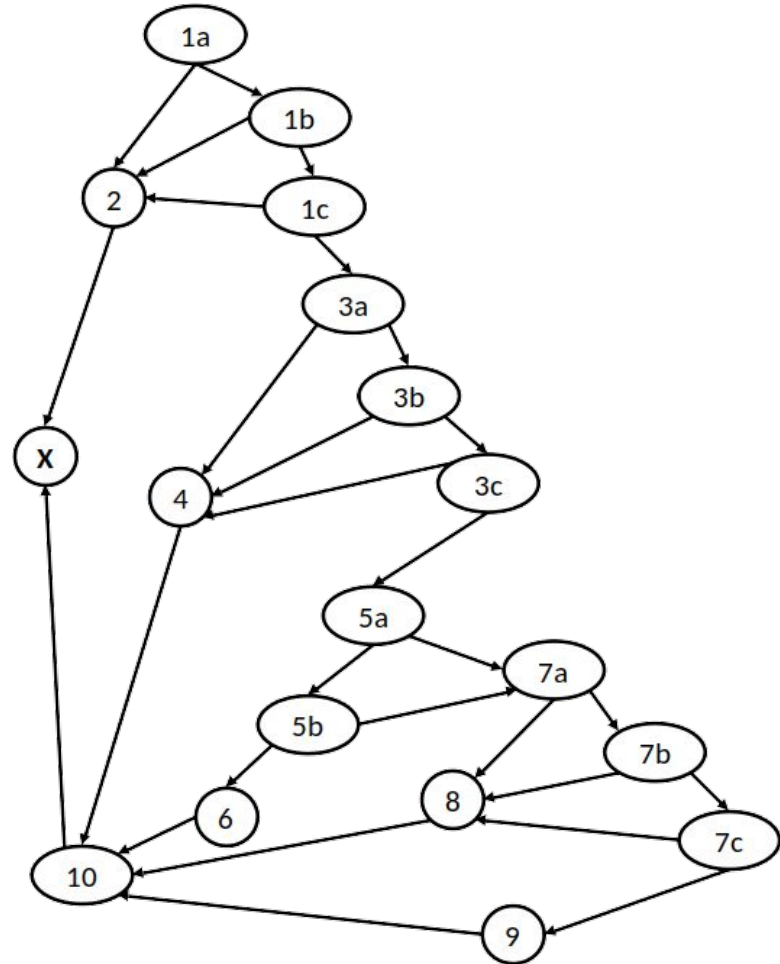
Um nó de saída



# Exemplo 1 - Triângulo

Critério todos os nós:

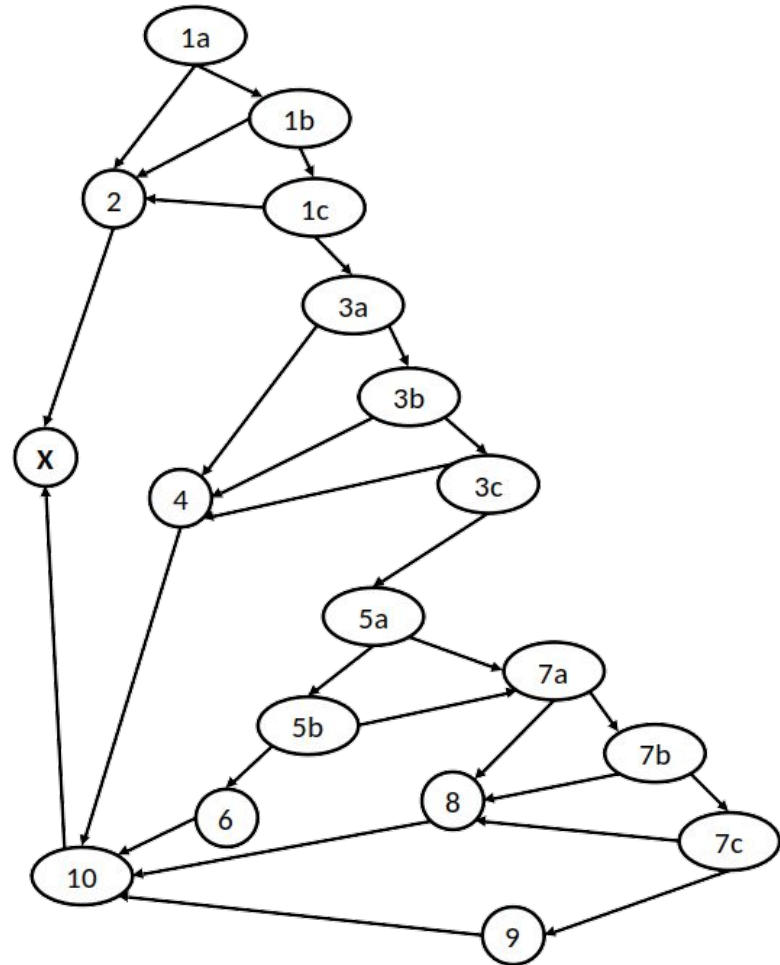
- 1a, 1b, 1c, 2,
- 3a, 3b, 3c, 4,
- 5a, 5b, 6,
- 7a, 7b, 7c,
- 8, 9, 10,
- X



# Exemplo 1 - Triângulo

Critério todas as arestas:

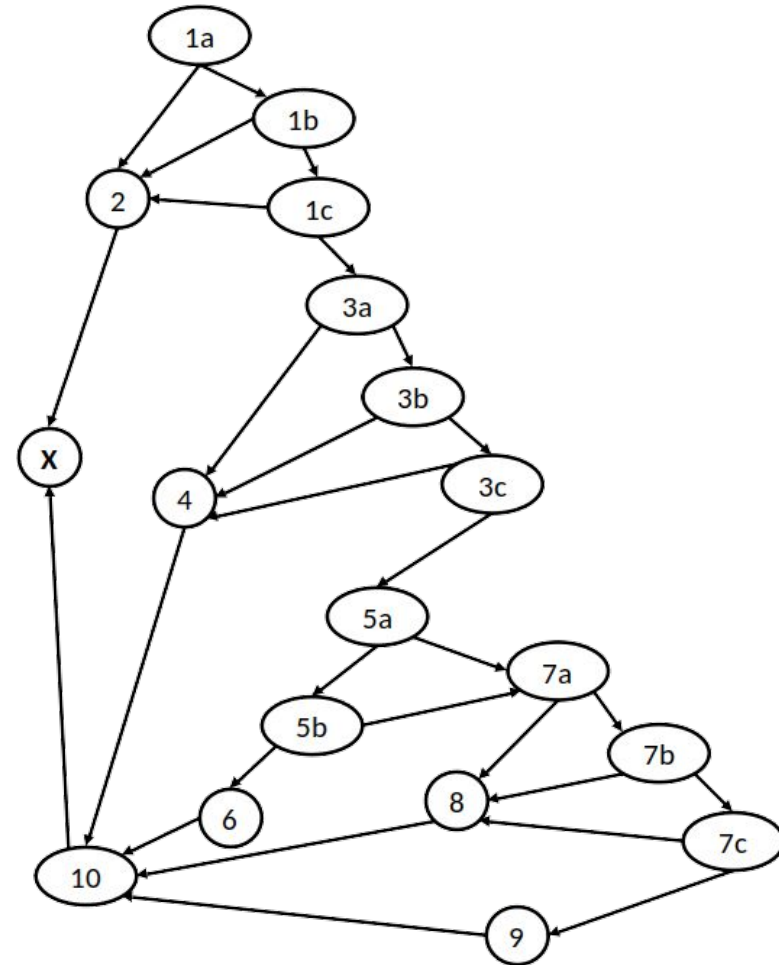
- (1a,1b), (1a,2), (1b,2),(1b,1c),
- (1c,2),(2,X),(1c,3a), (3a,3b),
- (3a,4),(3b,4),(3b,3c),(3c,4),
- (3c,5a),(4,10), (5a,7a),(5a,5b),
- (5b,7a),(5b,6),(6,10), (7a,7b),
- (7a,8),(7b,8),(7b,7c),(7c,8),
- (7c,9), (8,10), (9,10), (10,X)



# Exemplo 1 - Triângulo

Critério todos os caminhos básicos

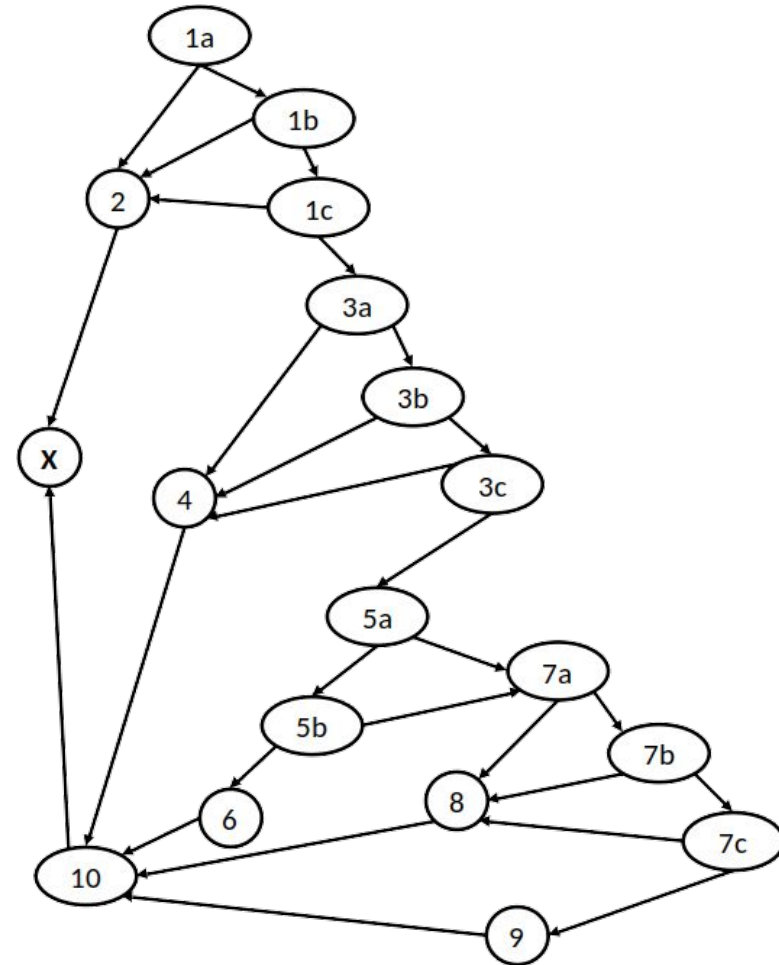
- Calcular complexidade ciclomática
- Complexidade ciclomática =  $A - N + 2$
- Complexidade ciclomática =  $P + 1$



# Exemplo 1 - Triângulo

Critério todos os caminhos básicos

- Calcular complexidade ciclomática
- Complexidade ciclomática =  $A - N + 2$ 
  - Arestas = 28
  - Nós = 18
  - $CC = 28 - 18 + 2 \Rightarrow 12$
- Complexidade ciclomática =  $P + 1$ 
  - Predicados = 11
  - $CC = 11 + 1$

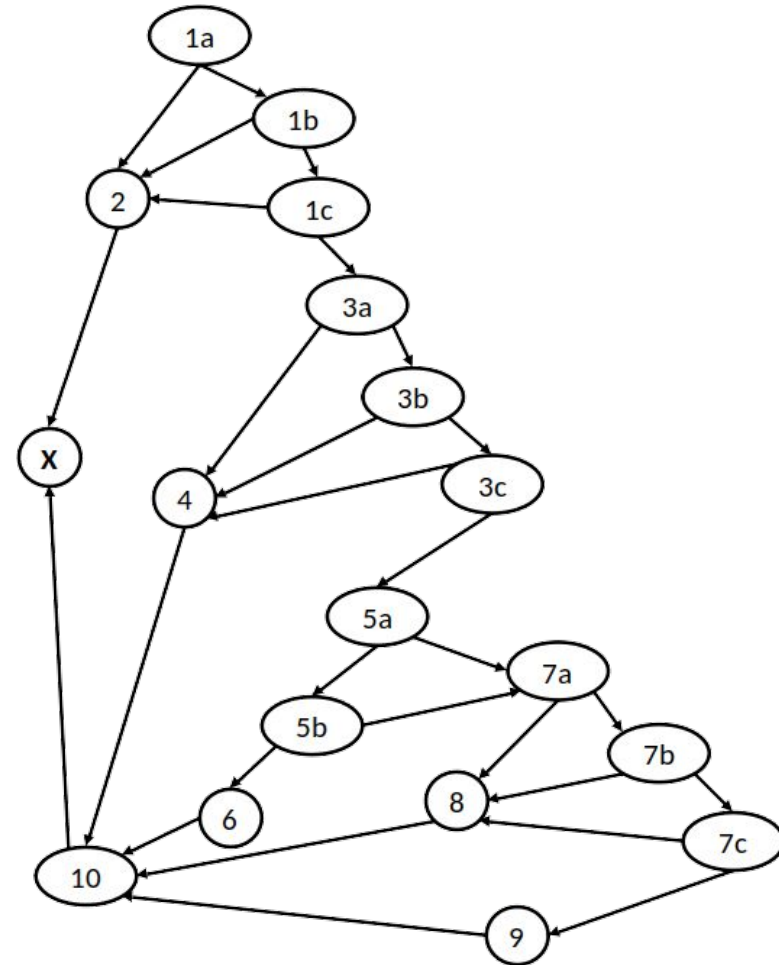




# Exemplo 1 - Triângulo

Critério todos os caminhos básicos

- 1a – 2 – X
- 1a – 1b – 2 – X
- 1a – 1b – 1c – 2 – X
- 1a – 1b – 1c – 3a – 4 – 10 – X
- 1a – 1b – 1c – 3a – 3b – 4 – 10 – X
- 1a – 1b – 1c – 3a – 3b – 3c – 4 – 10 – X
- 1a – 1b – 1c – 3a – 3b – 3c – 5a – 5b – 6 – 10 – X
- 1a – 1b – 1c – 3a – 3b – 3c – 5a – 7a – 8 – 10 – X
- 1a – 1b – 1c – 3a – 3b – 3c – 5a – 7a – 7b – 8 – 10 – X
- 1a – 1b – 1c – 3a – 3b – 3c – 5a – 7a – 7b – 7c – 8 – 10 – X
- 1a – 1b – 1c – 3a – 3b – 3c – 5a – 7a – 7b – 7c – 9 – 10 – X



# Derivando casos de teste

É preciso criar casos de teste para cobrir os requisitos de teste derivados de cada critério

Os casos de testes precisam passar por todos os nós

Os casos de testes precisam passar por todas as arestas

Os casos de testes precisam exercitar todos os caminhos básicos

# Exemplo do Triângulo

Ver arquivo:

- TesteEstrutural-ExemploTrianguloCobertura.pdf

Link:

- <https://drive.google.com/file/d/1IidjhH5RAgG-qwx2kL5ShGR3Q2wdMGMM/view?usp=sharing>

# Ferramentas

The screenshot displays the IntelliJ IDEA 7.0.4 interface with the 'Coverage' tool active. The main editor shows the `Money` class with various methods. The 'Coverage' window on the left provides a detailed breakdown of code coverage metrics for the project and the selected class.

**Coverage Data:**

Element	Coverage	Cplx	Lines	#Uncovered
com	94.7%	79	397	14
cenqua	94.7%	79	397	14
samples	94.7%	79	397	14
money	94.7%	79	397	14
IMoney	-	0	36	0
Money	89.4%	18	67	5
add	100%	1	3	0
addMoney	100%	2	5	0
amount	100%	1	3	0
appendTo	100%	1	3	0
currency	100%	1	3	0
equals	61.5%	4	11	5
hashCode	100%	1	3	0
isZero	100%	1	3	0
Money	100%	1	4	0
multiply	100%	1	3	0
negate	100%	1	3	0

**Coverage and Tests:**

- Methods: - / - -
- Statements: 5 / 7 71.4%
- Conditionals: 3 / 6 50%
- TOTAL: 61.5%**
- Tests passed: 16 / 16 100%

**Metrics:**

Lines of Code:	Conditionals:
-	6
NC Lines of Code:	Statements: 7
	Methods: -
Total Complexity:	Classes: -
Avg Complexity:	Files: -
Complexity Density:	Packages: -
Recorded Test Cases:	Test Methods: 0

The main editor shows the `Money` class with the following code:

```
22 public IMoney add(IMoney m) {
23     return m.addMoney(this);
24 }
25 public IMoney addMoney(Money m) {
26     if (m.currency().equals(currency())) {
27         return new Money(amount()+m.amount(), currency());
28     }
29     return MoneyBag.create(this, m);
30 }
31 public IMoney addMoneyBag(MoneyBag s) {
32     return s.addMoney(this);
33 }
34 public int amount() {
35     return fAmount;
36 }
37 public String currency() {
38     return fCurrency;
39 }
40 public boolean equals(Object anObject) {
41     if (this == anObject) {
42         return true;
43     }
44     if (anObject instanceof IMoney) {
45         IMoney m = (IMoney) anObject;
46         return m.equals(this);
47     }
48     return false;
49 }
50 public int hashCode() {
51     return fCurrency.hashCode()+fAmount;
52 }
53 public boolean isZero() {
54     return amount() == 0;
55 }
56 public IMoney multiply(int factor) {
57     return new Money(amount()*factor, currency());
58 }
59 public IMoney negate() {
60     return new Money(-amount(), currency());
61 }
62 public IMoney subtract(IMoney m) {
63     return add(m.negate());
64 }
65 public String toString() {
66     StringBuffer buffer = new StringBuffer();
67     buffer.append(""+amount()+"-"+currency()+"");
68     return buffer.toString();
69 }
70 public void appendTo(MoneyBag m) {
71     m.appendMoney(this);
72 }
73 }
```

# Ferramentas

The screenshot displays an IDE window with the following components:

- Toolbar:** Standard IDE navigation and development tools.
- Project Structure:** A tree view on the left showing the project hierarchy: tutorial > src > main > java > com > cenqua > samples > money > MoneyBag.
- Coverage Table:** A table listing coverage metrics for various elements in the project.
- Coverage and Tests Summary:** A section showing overall coverage statistics.
- Source Code:** The MoneyBag.java file is open, showing Java code with annotations indicating coverage status.

Element	Coverage	Cplx	Lines	#Uncovered
money	92,9%	52	285	12
IMoney	-	0	43	0
Money	88,9%	17	80	9
MoneyBag	94,3%	35	135	8
add(...)	100%	1	3	0
add(...)	100%	1	3	0
add(...)	100%	1	3	0
appe...	100%	2	4	0
appe...	100%	4	13	0
appe...	100%	1	3	0
cont...	100%	2	5	0
creat...	100%	1	6	0
equa...	72%	7	19	7
find...	100%	3	8	0
hash...	100%	2	8	0
isZer...	-	1	3	0
mult...	100%	3	10	0

**Coverage and Tests**

Methods: - / - -

Statements: 10 / 13 76,9%

Conditionals: 8 / 12 66,7%

**TOTAL: 72%**

Tests passed: 9 / 9 100%

**Metrics**

Lines of Code:	NC Lines of Code:	Total Complexity:	Avg Complexity:	Complexity Density:
-	-	7	-	0,54

**Source Code Annotations:**

- Method equals invoked 14 times
- Branch: true 0 times, false 14 times
- Statement 61:9 executed 14 times

```
private void addElement(Money m) {
    fMonies.addElement(m);
}

private boolean isEmpty() {
    return fMonies.isEmpty();
}

private boolean contains(Money m) {
    for (Enumeration e = fMonies.elements(); e.hasMoreElements(); ) {
        Money m2 = (Money) e.nextElement();
        if (m2.equals(m)) {
            return true;
        }
    }
    return false;
}

private boolean isZero() {
    if (fMonies.isEmpty()) {
        return true;
    }
    for (Enumeration e = fMonies.elements(); e.hasMoreElements(); ) {
        Money m = (Money) e.nextElement();
        if (!m.isZero()) {
            return false;
        }
    }
    return true;
}

private Money findMoney(String currency) {
    for (Enumeration e = fMonies.elements(); e.hasMoreElements(); ) {
        Money m = (Money) e.nextElement();
        if (m.currency().equals(currency)) {
            return m;
        }
    }
    return null;
}
```

# Ferramentas

The screenshot displays the Eclipse IDE interface. The main editor shows the `CursorableLinkedList.java` file with the following code:

```
public boolean addAll(int index, Collection c) {  
    if (c.isEmpty()) {  
        return false;  
    } else if (size == index || size == 0) {  
        return addAll(c);  
    } else {  
        Listable succ = getListableAt(index);  
        Listable pred = (null == succ) ? null : succ.prev();  
        Iterator it = c.iterator();  
        while (it.hasNext()) {  
            pred = insertListable(pred, succ, it.next());  
        }  
        return true;  
    }  
}
```

The left sidebar shows the 'Hierarchy' view with a tree structure of test suites and utilities. The bottom panel displays the 'Coverage' view for the 'TestAllPackages' run, showing a table of coverage statistics.

Element	Coverage	Covered Lines	Total Lines
java - commons-collections	79,5 %	10927	13738
org.apache.commons.collections	74,1 %	3842	5183
ArrayStack.java	86,5 %	32	37
BagUtils.java	86,7 %	13	15
BeanMap.java	72,4 %	155	214
BinaryHeap.java	87,6 %	127	145
BoundedFifoBuffer.java	93,2 %	82	88
BufferOverflowException.java	55,6 %	5	9
BufferUnderflowException.java	88,9 %	0	9
BufferUtils.java	30,8 %	4	13
ClosureUtils.java	93,9 %	31	33
CollectionUtils.java	92,4 %	293	317
ComparatorUtils.java	8,6 %	3	35
CursorableLinkedList.java	85,4 %	444	520

# Ferramentas

The screenshot shows the Eclipse IDE interface. The top menu bar includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, and Help. The main editor displays the code for `CursorableLinkedList.java`. The code defines a `public boolean addAll(int index, Collection c)` method. The code is as follows:

```
public boolean addAll(int index, Collection c) {  
    if (c.isEmpty()) {  
        return false;  
    } else if (size == index || size == 0) {  
        return addAll(c);  
    } else {  
        Listable succ = getListableAt(index);  
        Listable pred = (null == succ) ? null : succ.prev();  
        Iterator it = c.iterator();  
        while (it.hasNext()) {  
            pred = insertListable(pred, succ, it.next());  
        }  
        return true;  
    }  
}
```

The left sidebar shows the Project Hierarchy, listing various test suites and utilities. The bottom panel shows the Coverage report for the `TestAllPackages` suite, dated 31.10.2006 15:04:14. The table below summarizes the coverage data:

Element	Coverage	Covered Lines	Total Lines
java - commons-collections	79,5 %	10927	13738
org.apache.commons.collections	74,1 %	3842	5183
ArrayStack.java	86,5 %	32	37
BagUtils.java	86,7 %	13	15
BeanMap.java	72,4 %	155	214
BinaryHeap.java	87,6 %	127	145
BoundedFifoBuffer.java	93,2 %	82	88
BufferOverflowException.java	55,6 %	5	9
BufferUnderflowException.java	88,9 %	0	9
BufferUtils.java	30,8 %	4	13
ClosureUtils.java	93,9 %	31	33
CollectionUtils.java	92,4 %	293	317
ComparatorUtils.java	8,6 %	3	35
CursorableLinkedList.java	85,4 %	444	520

The bottom status bar indicates the current state: Winable, Smart Insert, 149 / 28.

# Uma demonstração

## Exemplo do triângulo

[Vídeo: cobertura de código com IntelliJ + Gradle + Jacoco](#)

```
public static TipoTriangulo classificaTriangulo (int A, int B, int C){  
    if (temLadoInvalido(A,B,C))  
        return TipoTriangulo.LADOINVALIDO;  
    if (!formaTriangulo(A,B,C))  
        return TipoTriangulo.NAOTRIANGULO;  
    if (A==B && B==C)  
        return TipoTriangulo.EQUILATERO;  
    if (A==B || B==C || A==C)  
        return TipoTriangulo.ISOSCELES;  
    return TipoTriangulo.ESCALENO;  
}
```



# **ACH 2028**

## **Qualidade de Software**

### **Aula 08 - Teste de Software: Técnica Estrutural ou Caixa-Branca**

Prof. Marcelo Medeiros Eler  
[marceloeler@usp.br](mailto:marceloeler@usp.br)