



Testes baseados em grafos: fluxo de controle

Criado: abr/2005

Última atualização: mar/2013



TÓPICOS

- Terminologia
- Exemplos de grafos usados em testes
- Relações
- Uso de grafos em testes

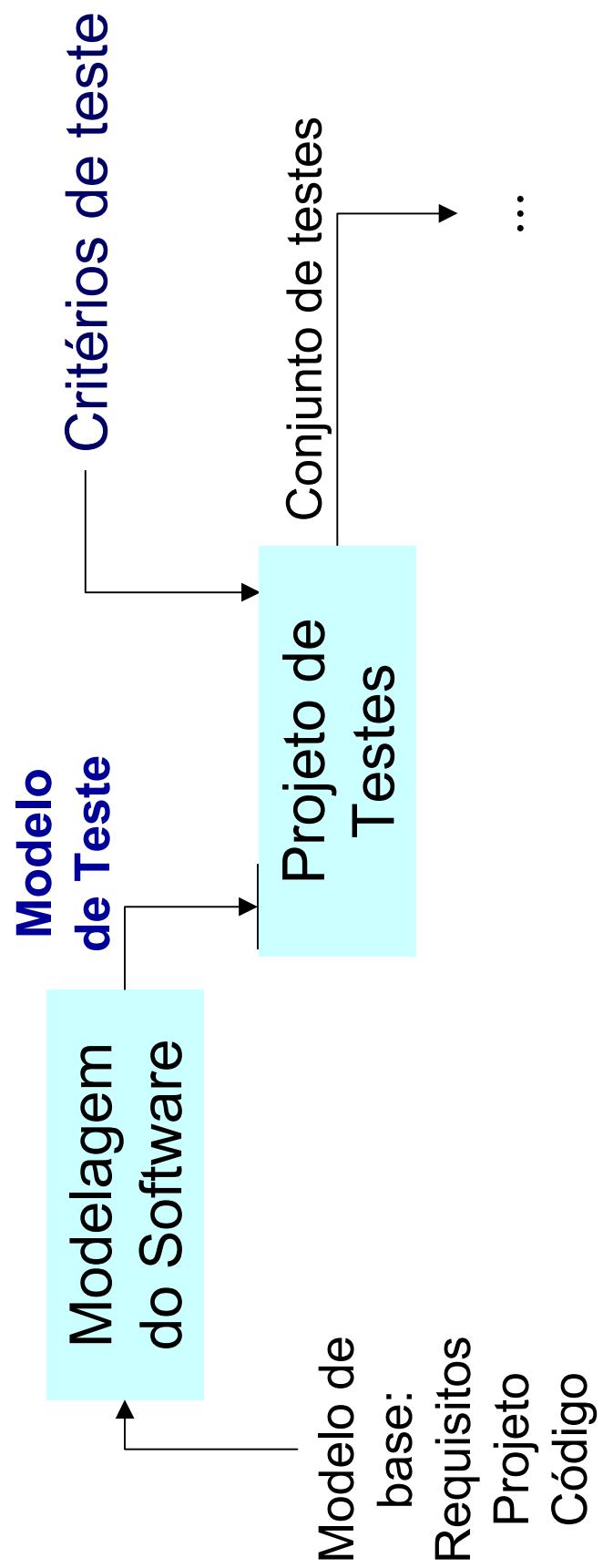


Referências

- P. Amman, J. Offutt. *Introduction to Software Testing*, 2008, cap.2.
- P.B.Menezes. Linguagens Formais e Autômatos, Editora Sagra Luzzato,
3a. Edição, 2000, c.1.
- B.Beizer. *Black-Box Testing*. John Wiley, 1995. cap. 2
- R.Binder. *Testing Object-Oriented Systems. Models, Patterns and Tools*.
Addison Wesley, 2000. c. 8.11

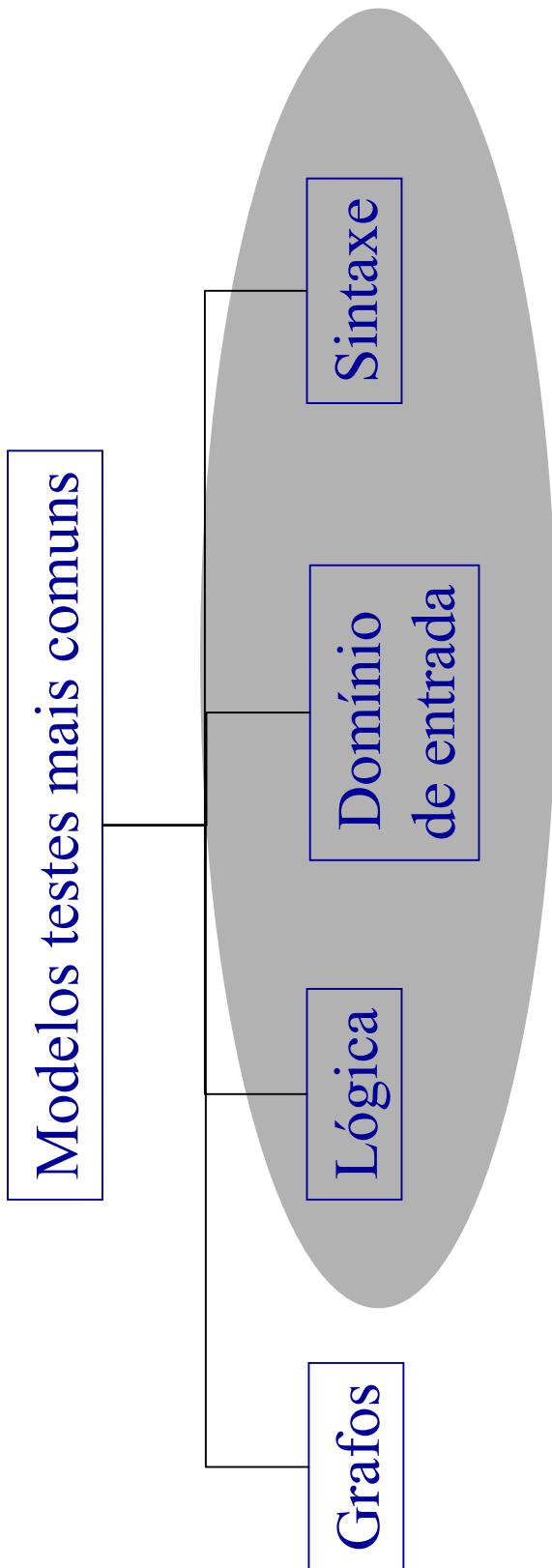


Projeto de testes





Modelos de Testes





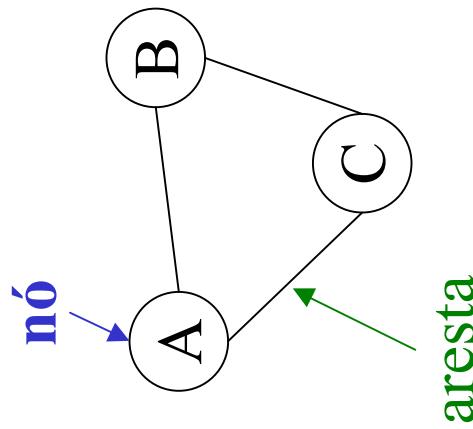
Grafos

- Estrutura muito comum em testes
- Pode ser obtido a partir de diferentes modelos de base:
 - Casos de uso
 - Diagramas da UML: estados, de atividades, de colaboração, de classes, de componentes
 - Máquinas finitas de estados (clássicas e variações)
 - Código
 - ...
- Geração de testes → percorrer o grafo
 - Existem inúmeros algoritmos na literatura



Terminologia - 1

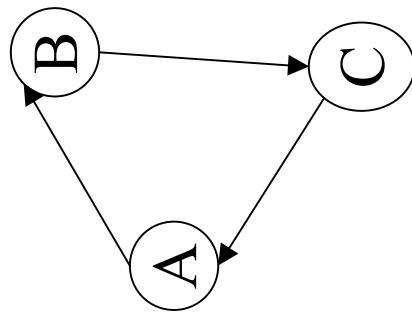
- Grafo
 - Representa relações entre entidades
 - Relação: associação de interesse entre entidades
 - **a R b** → a está relacionando com a
 - ex.:
 - **a** chama **b**
 - **a** calcula valor que é usado em **b**
 - a ação **a** é seguida da ação **b**
 - Elementos:
 - **Nós** (ou vértices): representam as entidades
 - **Arestas**: representam as relações entre as entidades





Terminologia -2

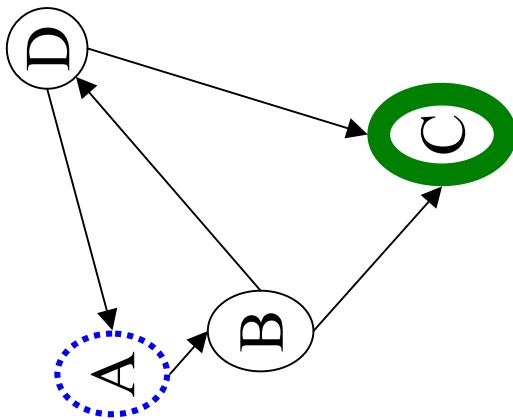
- As arestas de um grafo podem ser direcionadas, indicando que elas só podem ser atravessadas em uma única direção.
- Um grafo que só contém arestas direcionadas (também chamadas de **arcos**) é chamado de **grafo direcionado** ou **digrapho**.
- A maioria dos grafos usados em testes são digrafos.





Terminologia - 3

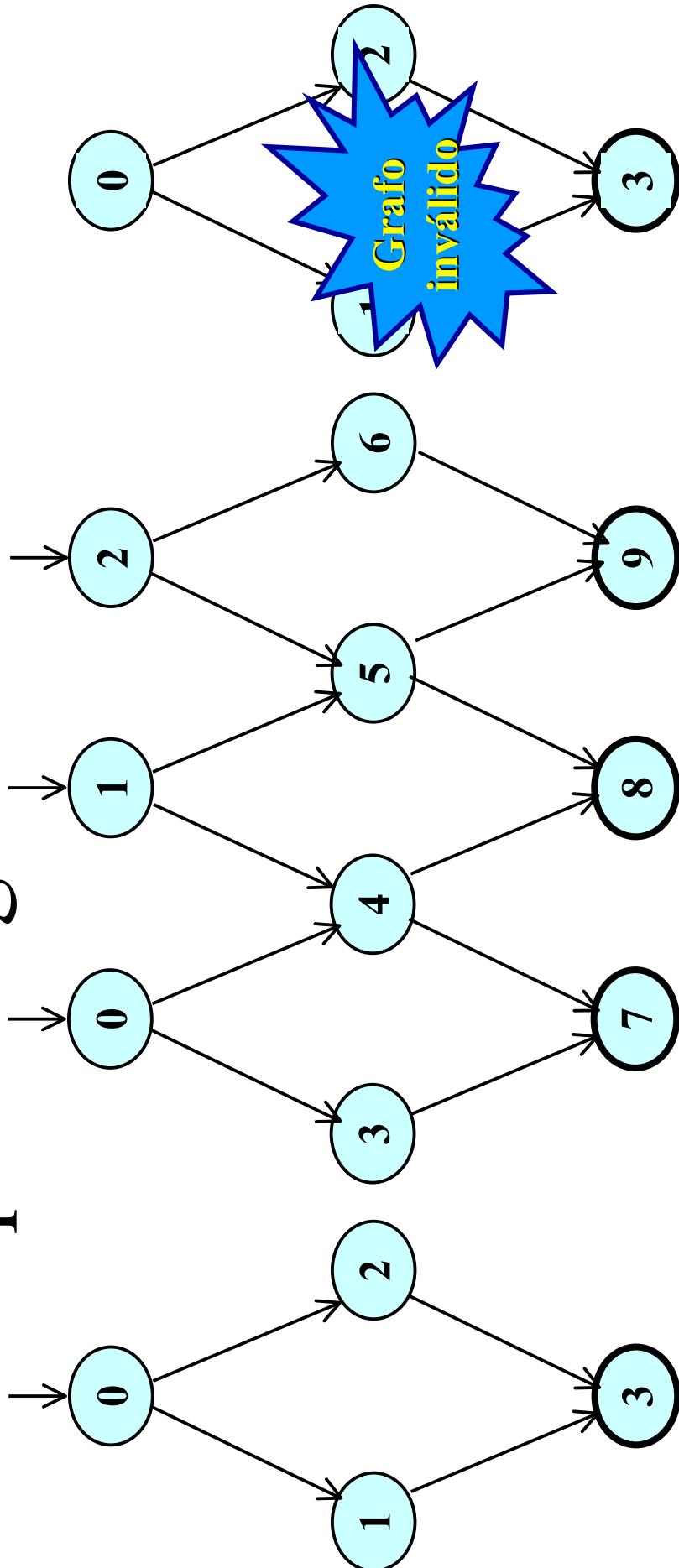
- Em testes, é interessante definir dois tipos especiais de nós:
 - **Nó de entrada (inicial)**: representa um ponto de entrada do programa ou sistema. Pode ter 1 ou mais nós de entrada.
 - **Nó de saída (terminal)**: um nó que não tem arcos saindo dele. Representa um ponto de saída de um programa ou sistema.
Um grafo pode conter zero ou mais nós de saída.





Exemplos de grafos

[Ammann, Offutt 2008]



$$N_0 = \{ 0 \}$$

$$N_f = \{ 3 \}$$

$$N_0 = \{ 0, 1, 2 \}$$

$$N_f = \{ 7, 8, 9 \}$$

$$N_0 = \{ \}$$

$$N_f = \{ 3 \}$$



Uso de grafos em testes

- Para usar grafos para projetar casos de teste deve-se proceder de acordo com os seguintes passos:
 1. Defina o grafo
 2. Defina o tipo da relação
 3. Percorra o grafo para atender a critérios selecionados (de preferência, usando um algoritmo de percurso em grafos)

ou seja, ao ver um grafo, o testador deve ...

cobri-lo

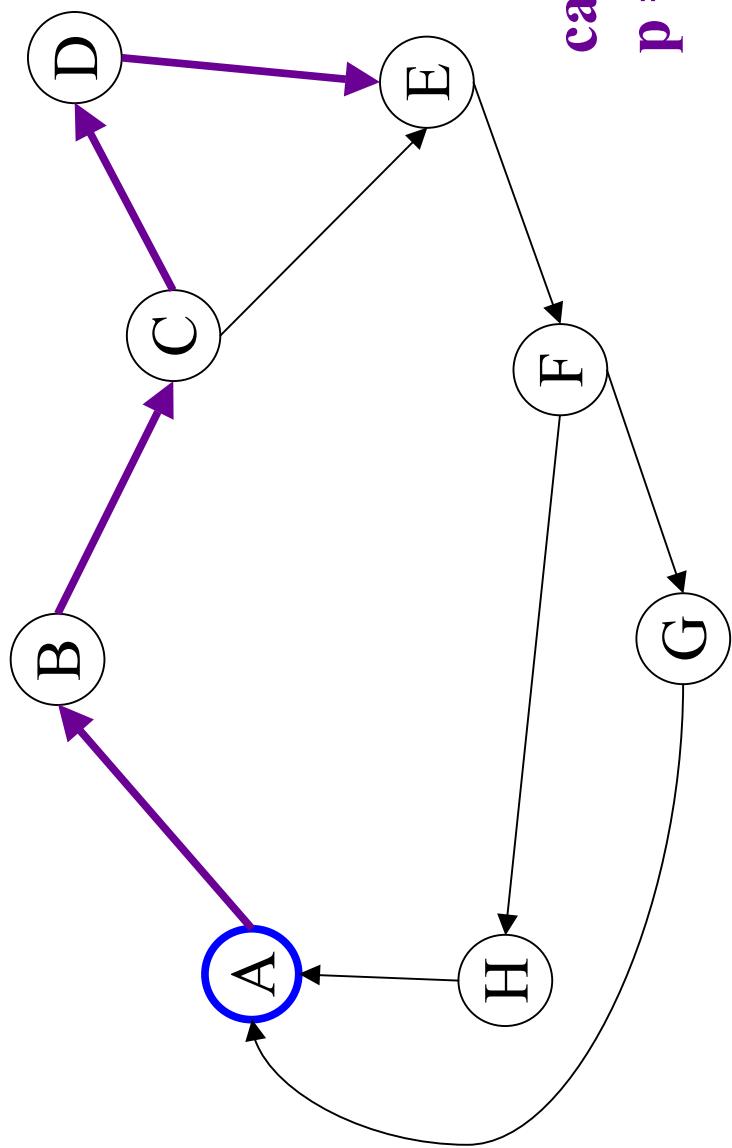


Definições úteis para os testes

- **Caminho:** sequência de arestas partindo de um nó origem a um nó destino
- **Caminho completo:** caminho de um nó de entrada a um nó de saída
- **Comprimento** de um caminho: número de nós ou número de arestas de um caminho
- **Ciclo** (ou laço): um caminho em que um nó (ou aresta) é visitado mais de uma vez
- **Caminho livre de ciclos:** um caminho em que nenhum nó (ou aresta) é visitado mais de uma vez



Exemplo

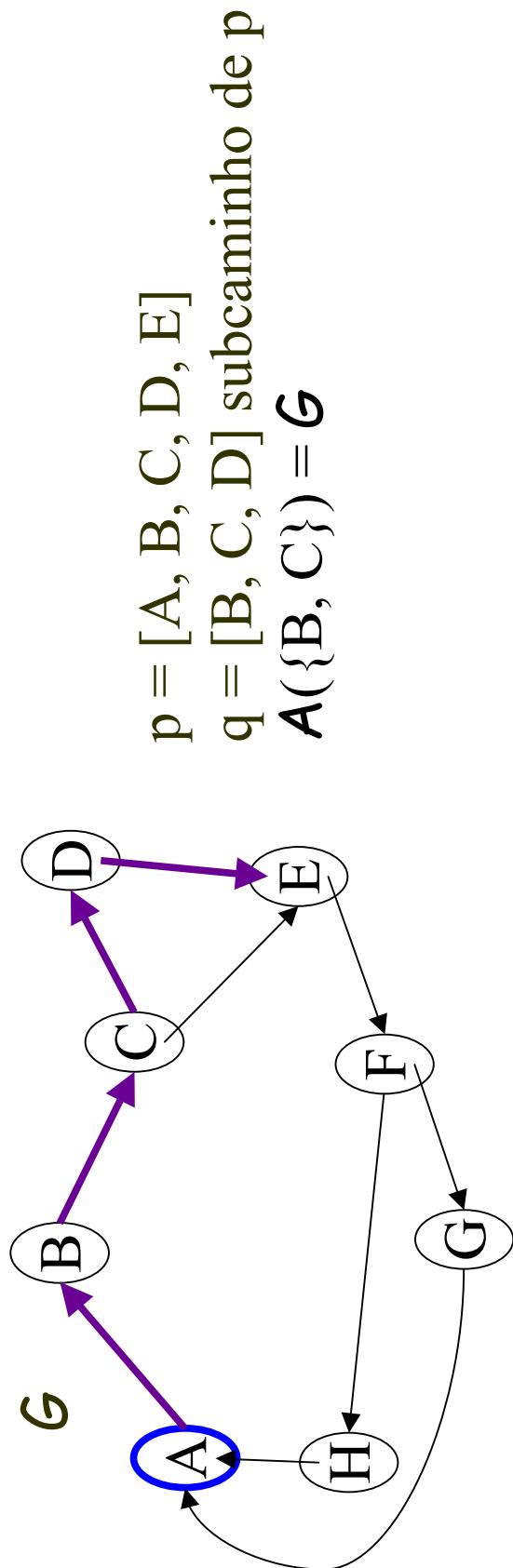


caminho de A a E:
 $p = [A, B, C, D, E]$



Mais definições

- **Subcaminho:** subsequência de nós em um caminho
- **Alcance** de um nó – $A(n)$: subgrafo que pode ser alcançado a partir de n





Caminho de teste

- **Caminho de teste:**
 - caminho que começa em um nó inicial e termina em um nó final
 - Representam a execução de um caso de teste:
 - Alguns caminhos de teste podem ser executados por mais de um caso de teste
 - Alguns caminhos de teste não podem ser executados por nenhum caso de teste
- **Grafos única entrada- única saída ou SESE**
(Single-Entry, Single-Exit):
 - Conjunto de nós iniciais e finais contém exatamente um nó cada.



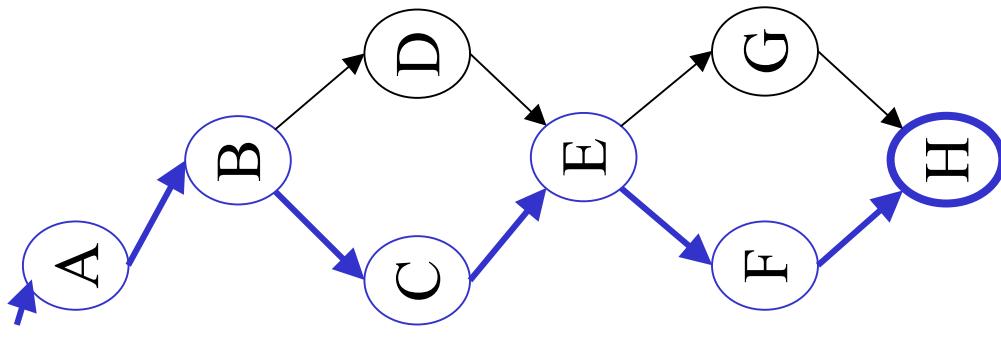
Visita e percurso

- Visita:
 - Um caminho de teste p **visita** um nó n se $n \in p$
 - Um caminho de teste p **visita** uma aresta e se $e \in p$
- Percurso (*tour*):
 - Um caminho de teste p **percorre** um subcaminho q se $q \in p$



Visita e percurso

- Visita:
 - Um caminho de teste p **visita** um nó n se $n \in p$
 - Um caminho de teste p **visita** uma aresta e se $e \in p$
- Percurso (*tour*):
 - Um caminho de teste p **percorre** um subcaminho q se $q \in p$



$P = [A, B, C, E, F, H]$
Visita nós A, B, C, E, F, H
Visita arestas $(A, B), (B, C), (C, E), (E, F), (F, H)$
Percorre subcaminhos $(A, B, C), (B, C, E), (C, E, F), (E, F, H), (B, C, E, F), \dots$



Testes e Caminhos de Testes

- $p(t)$: caminho de teste executado por um caso de teste t
- $p(T)$: conjunto de caminhos de testes executado pelo conjunto de testes T
- Cada teste t executa um e somente um caminho de teste
 - Software determinista: um caso de teste executa sempre o mesmo caminho de teste
 - Software indeterminista: um caso de teste pode executar diferentes caminhos a cada nova execução



Adequabilidade

- Sejam:
 - C : critério de teste
 - RT_C : requisitos de teste para o critério C
 - T : conjunto de testes
 - G um grafo
 - $p(T)$: conjunto de caminhos de teste em G executados por T
- Diz-se que T é **adequado** a C (ou **C -adequado**) se
$$\forall rt \in RT_C \exists q \in p(T) \mid q \text{ exercita } rt$$

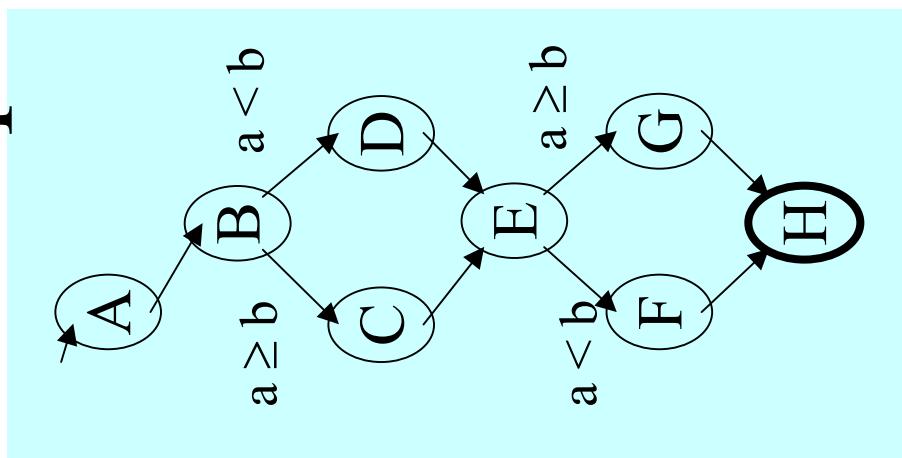


Alcançabilidade e caminho

- Um critério C pode requerer que um elemento do grafo (nó ou aresta) seja alcançado a partir de outro:
 - Um elemento x de G é **sintaticamente** alcançável a partir de um elemento y de G se existe um subcaminho entre x e y
 - Um elemento x de G é **semanticamente** alcançável a partir de um elemento y de G se existe um teste que pode executar o subcaminho de x a y .



Exemplo



C: cobrir arestas (B, D) e (G, H)
 $RT_C: \{(B, D), (G, H)\}$
 (G, H) é sintaticamente alcançável a partir de (B, D) :
Subcaminho: $(B, D) \rightarrow (D, E) \rightarrow (E, G) \rightarrow (G, H)$

(G, H) é semanticamente alcançável a partir de (B, D) ?



Critérios de cobertura de grafos

- Critérios baseados no fluxo de controle:
 - Considera somente nós e arestas
- Critérios baseados no fluxo de dados:
 - Requer que o grafo seja anotado com referências a variáveis



Exercício

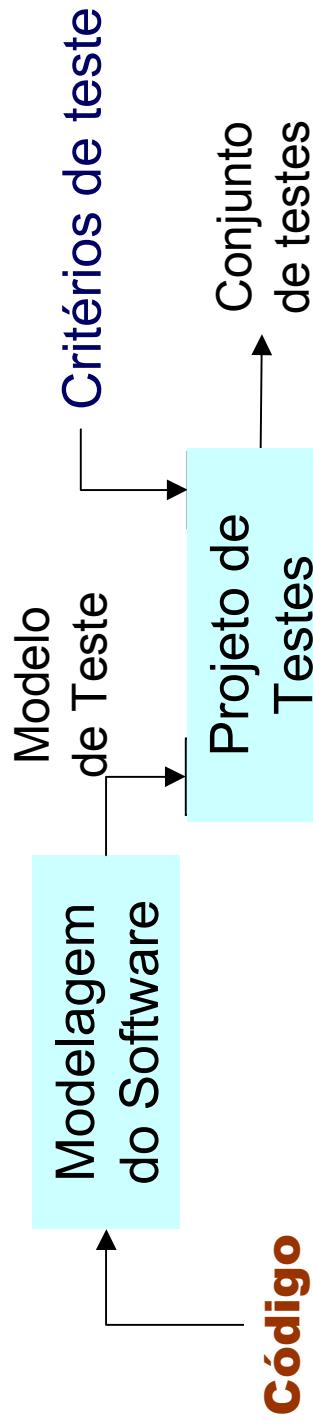
- Considere o seguinte grafo $G(N, E)$:
 - $N = \{a, b, c\}$
 - $N_0 = \{a\}$
 - $N_f = \emptyset$
 - $E = \{(a, b), (a, c), (b, c), (c, a)\}$
- Considere os seguintes caminhos candidatos:
 - $p_0 = [a, b, c, a]$
 - $p_1 = [a, c, a, b, c]$
 - $p_2 = [a, b, c, a, b, a, c]$
 - $p_3 = [b, c, a, c]$
 - $p_4 = [a, b, c, b, c]$
- Responda:
 - Quais dos caminhos dados são caminhos de teste?
 - Liste os oito requisitos de teste para o critério de cobertura de pares de arestas
 - Os caminhos de teste identificados anteriormente satisfazem ao critério?



Testes baseados na implementação (caixa branca)



Testes caixa branca

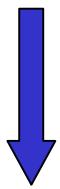


- Testes baseados na implementação ou testes caixa-branca:
 - visam exercitar estruturas de controle (instruções) e de dados de um programa



Modelos de Teste

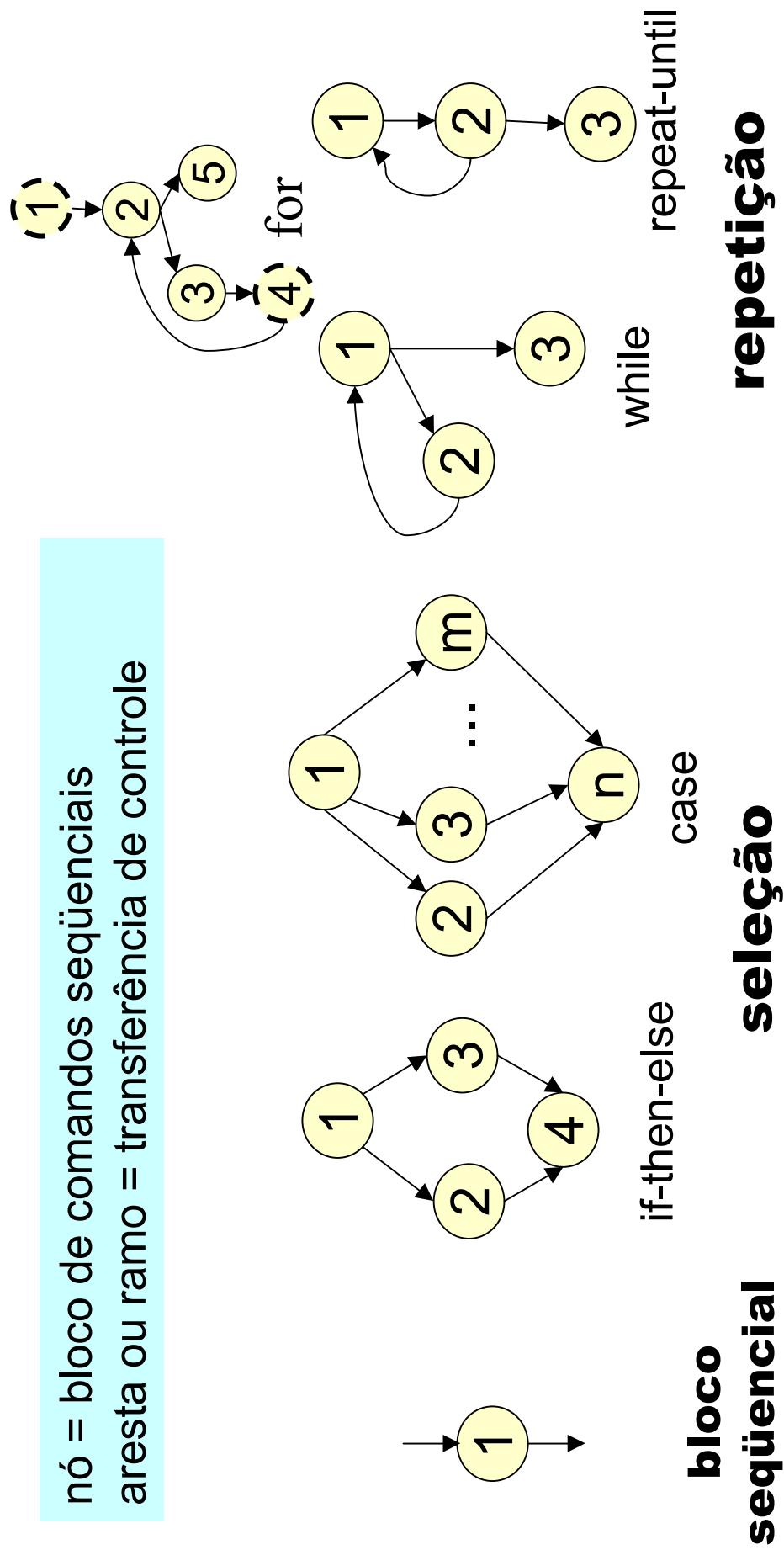
- Grafo de fluxo de controle
 - Representa a ordem na qual as instruções são executadas
- Grafo de fluxo de dados
 - Representa também as definições de variáveis e seus usos no programa





Grafo de fluxo de controle

nó = bloco de comandos seqüenciais
aresta ou ramo = transferência de controle

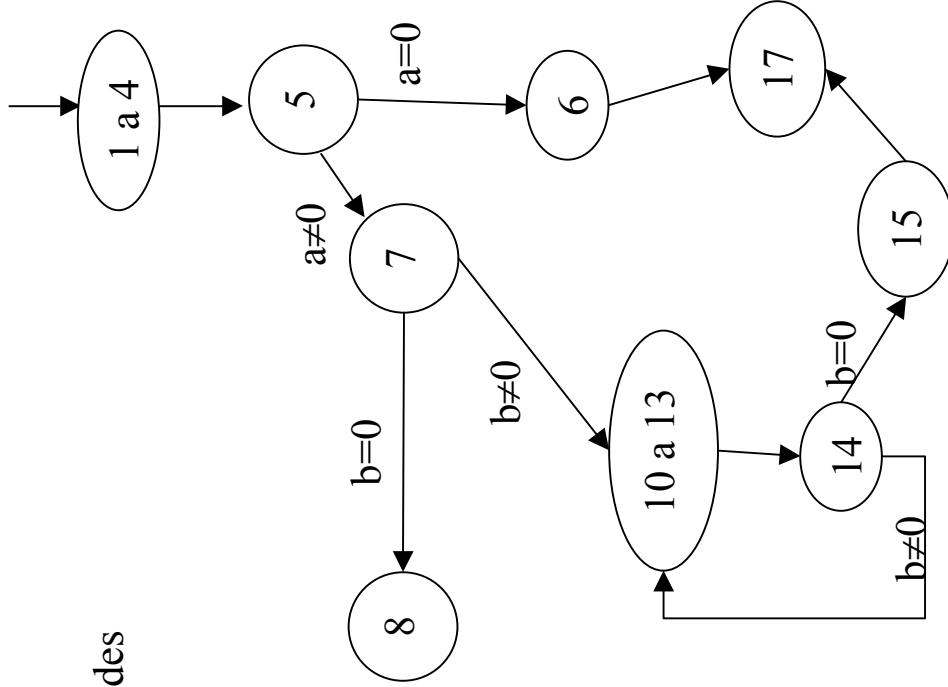




Exemplo

Cálculo do mdc (a, b) baseado no algoritmo de Euclides

```
1. function mdc (int a, int b) {  
    2.     int temp, value;  
    3.     a := abs(a);  
    4.     b := abs(b);  
    5.     if (a = 0) then  
    6.         value := b; // b é o MDC  
    7.     else if (b = 0) then  
    8.         exceção;  
    9.     else  
    10.        repeat  
    11.            temp := b;  
    12.            b := a mod b;  
    13.            a := temp;  
    14.        until (b = 0)  
    15.        value := a;  
    16.    end if;  
    17.    return value;  
    18. end mdc
```



Modelo de base



Modelo de teste



Critérios de cobertura

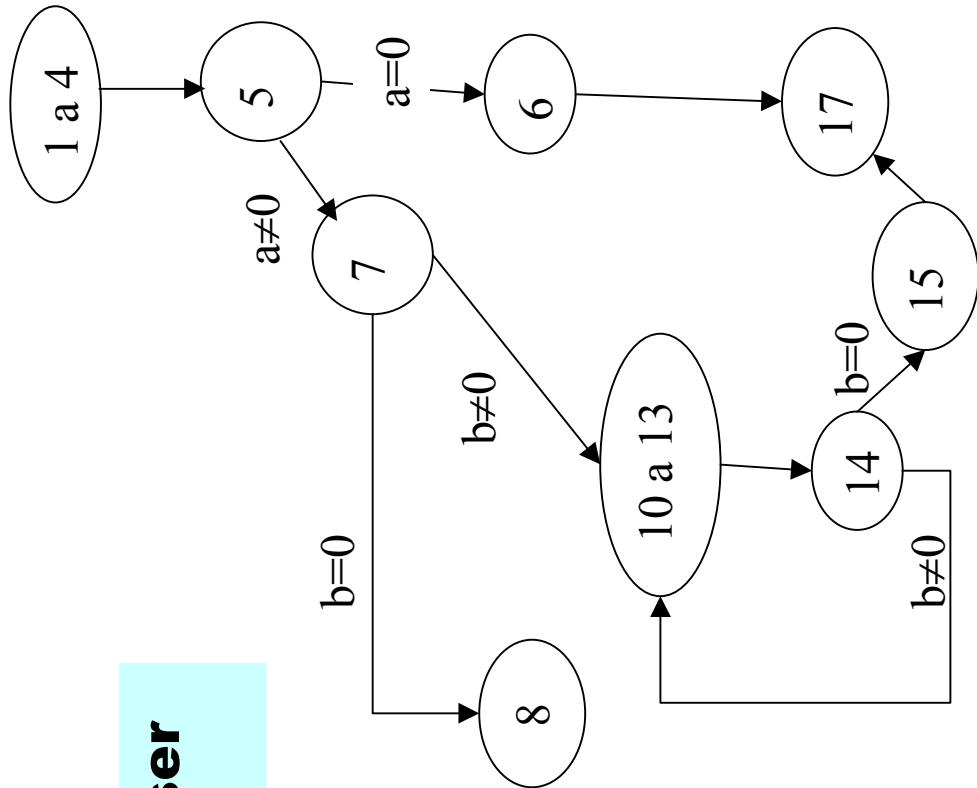
- Cobertura de nós
- Cobertura de ramos
 - Cobertura de pares de ramos
 - Cobertura de condições
- Cobertura de caminhos
 - Cobertura de caminhos relevantes
 - Cobertura de caminhos básicos
- Cobertura de laços



Teste de instruções

Critério: cada instrução deve ser executada pelo menos 1 vez

nós predicados dados

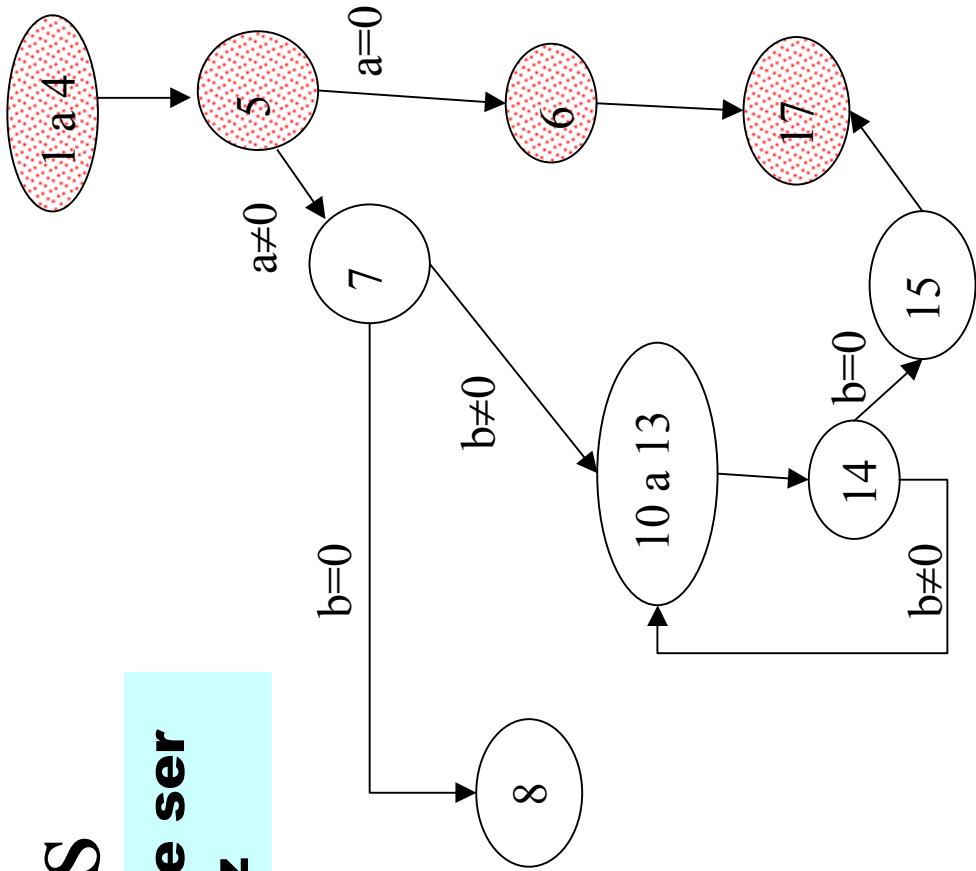




Teste de instruções

Critério: cada instrução deve ser executada pelo menos 1 vez

nós predicados dados
 $\{1, 5, 6, 16, 17\}$ $a = 0, \forall b$ $(0, 3)$

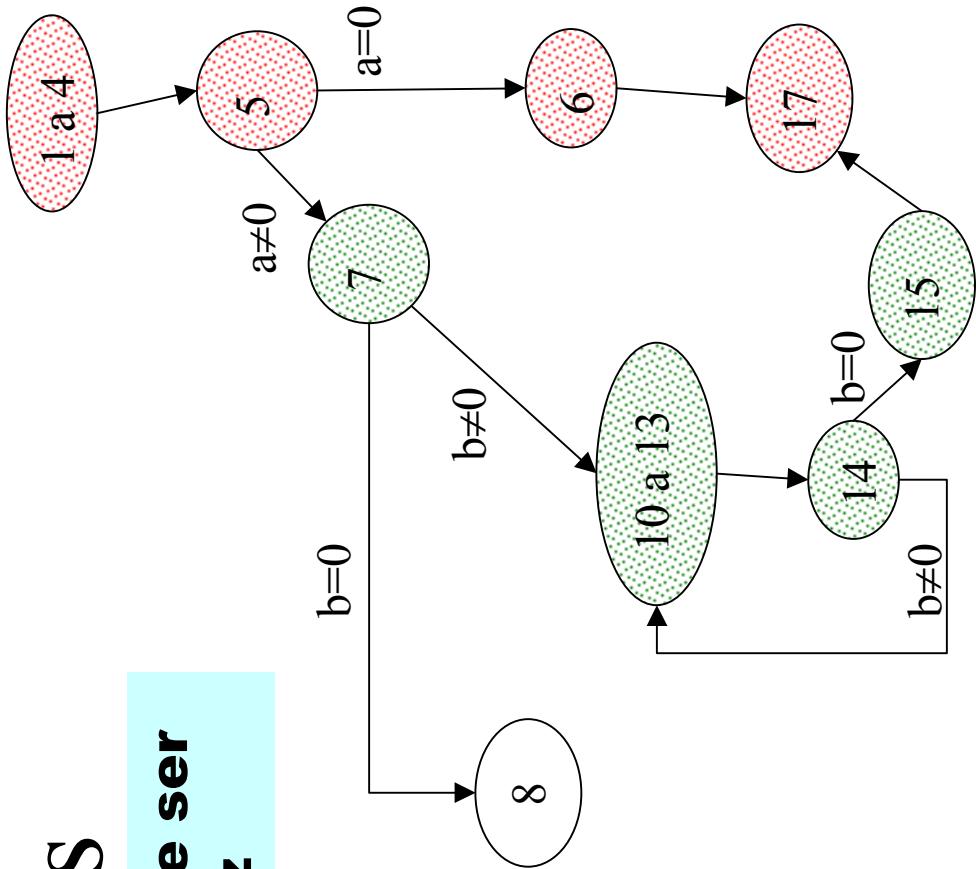




Teste de instruções

Critério: cada instrução deve ser executada pelo menos 1 vez

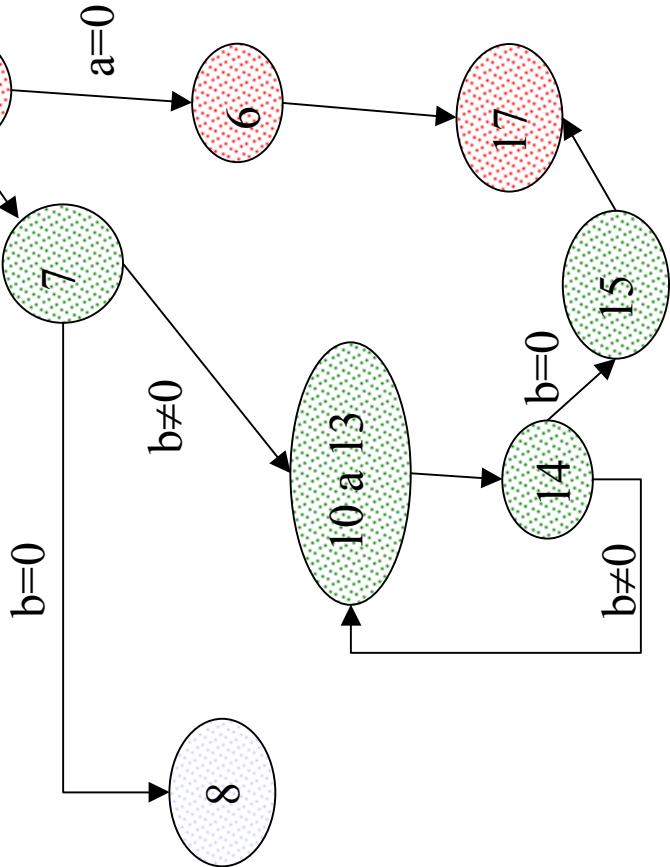
nós	predicados	dados
{1, 5, 6, 16, 17}	$a = 0, \forall b$	(0, 3)
{1, 5, 7, 9, 10, 14}	$a \neq 0, b \neq 0$	(4, -2)
{15, 16, 17}		





Teste de instruções

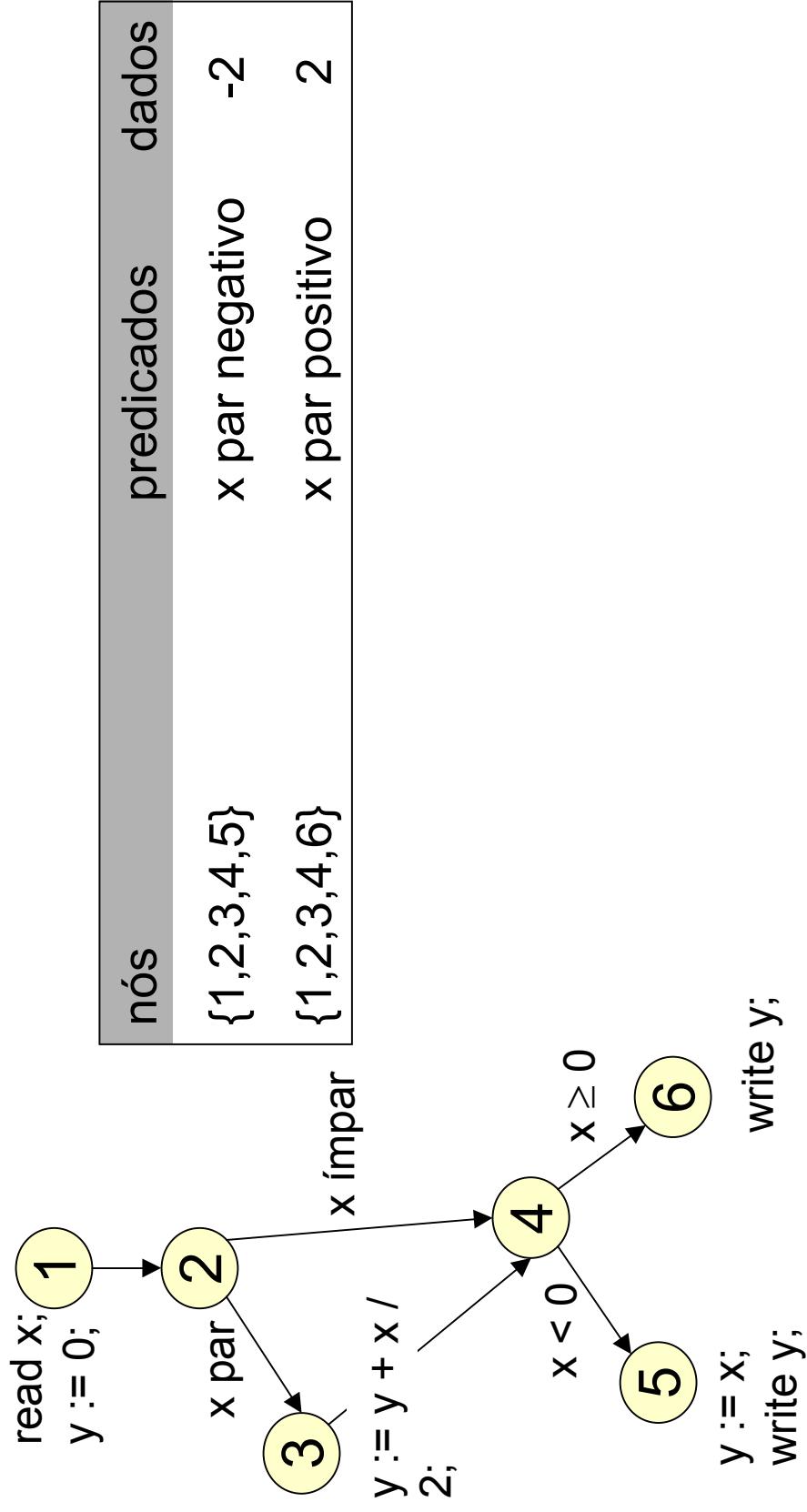
Critério: cada instrução deve ser executada pelo menos 1 vez



nós	predicados	dados
{1, 5, 6, 16, 17}	$a = 0, \forall b$	(0, 3)
{1, 5, 7, 9, 10, 14 15, 16, 17}	$a ≠ 0, b ≠ 0$	(4, -2)
{1, 5, 7, 8}	$a ≠ 0, b = 0$	(4, 0)

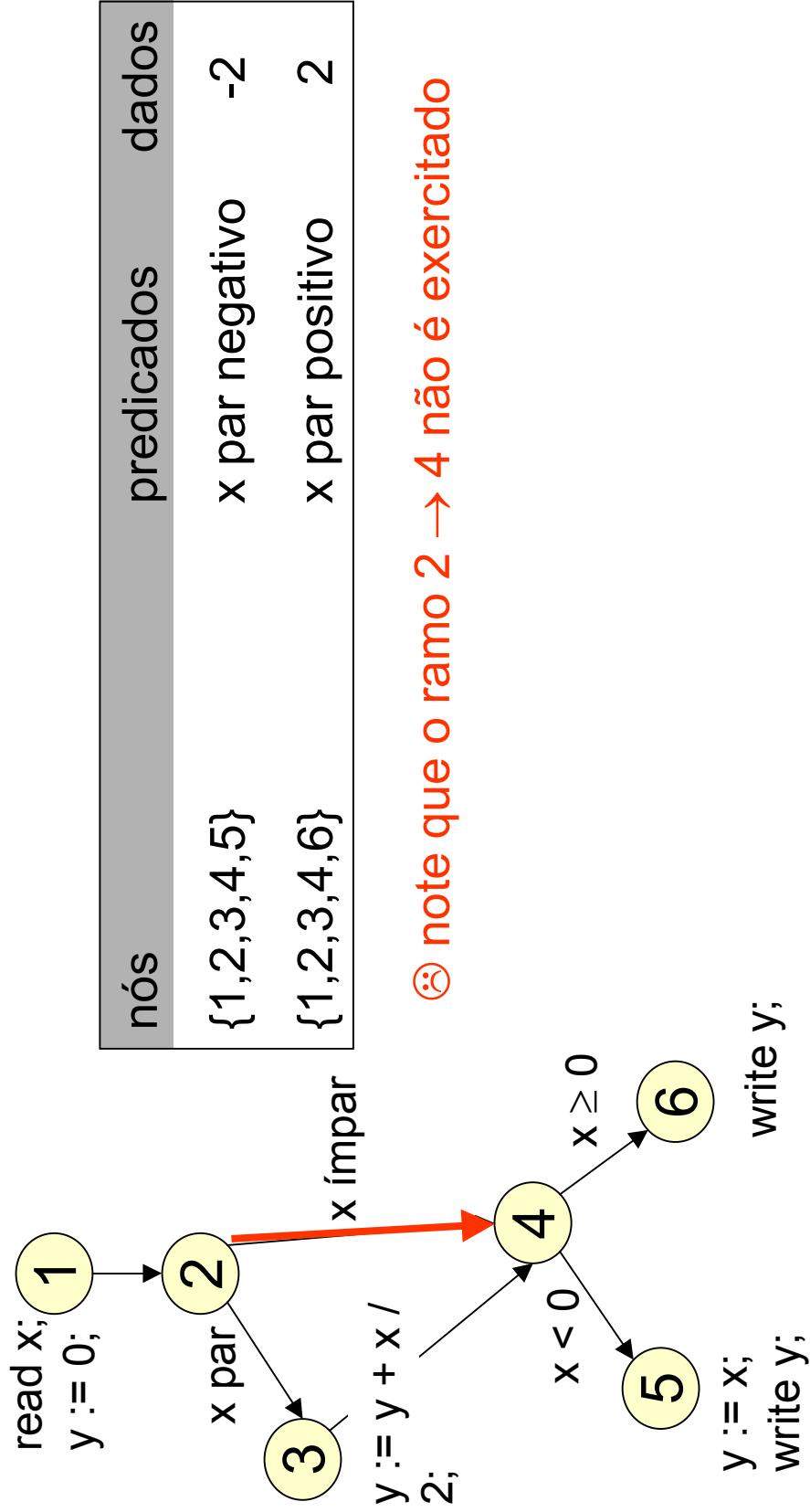


Cobertura de nós





Cobertura de nós

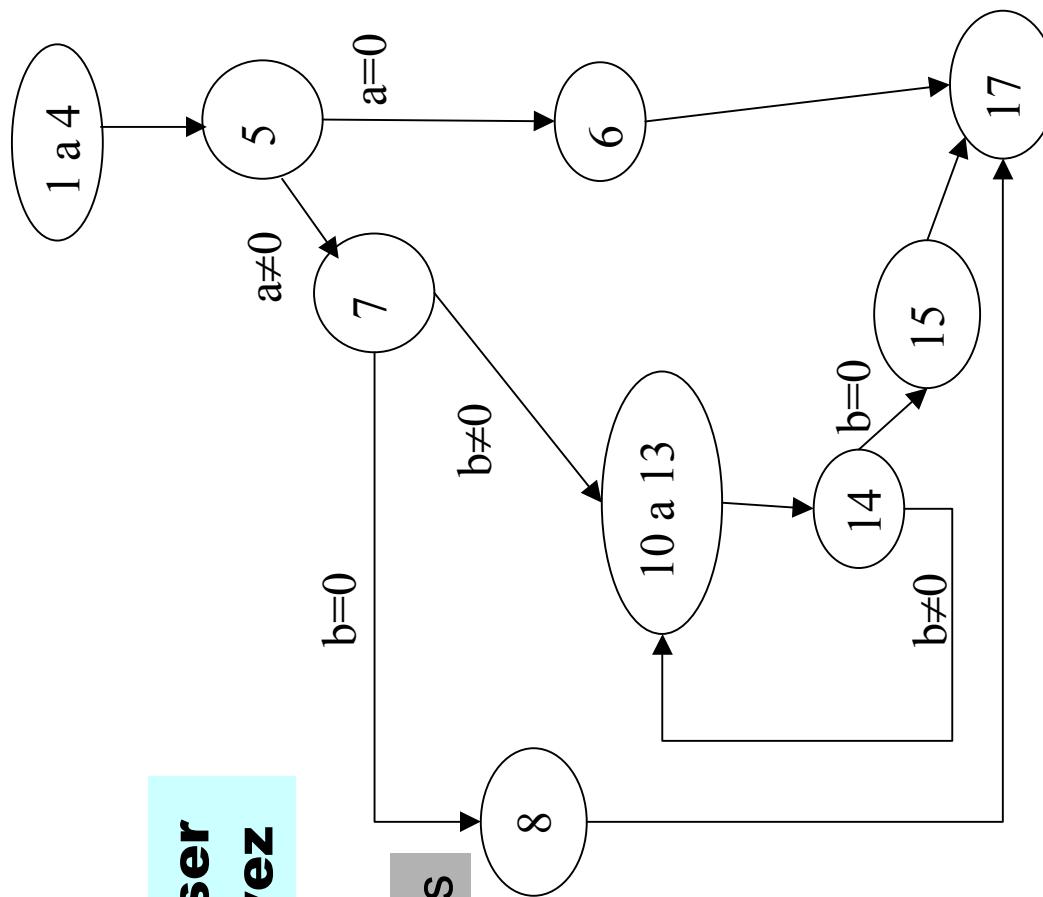




Teste de decisões

Critério: cada ramo deve ser percorrido pelo menos 1 vez

ramos predicados dados

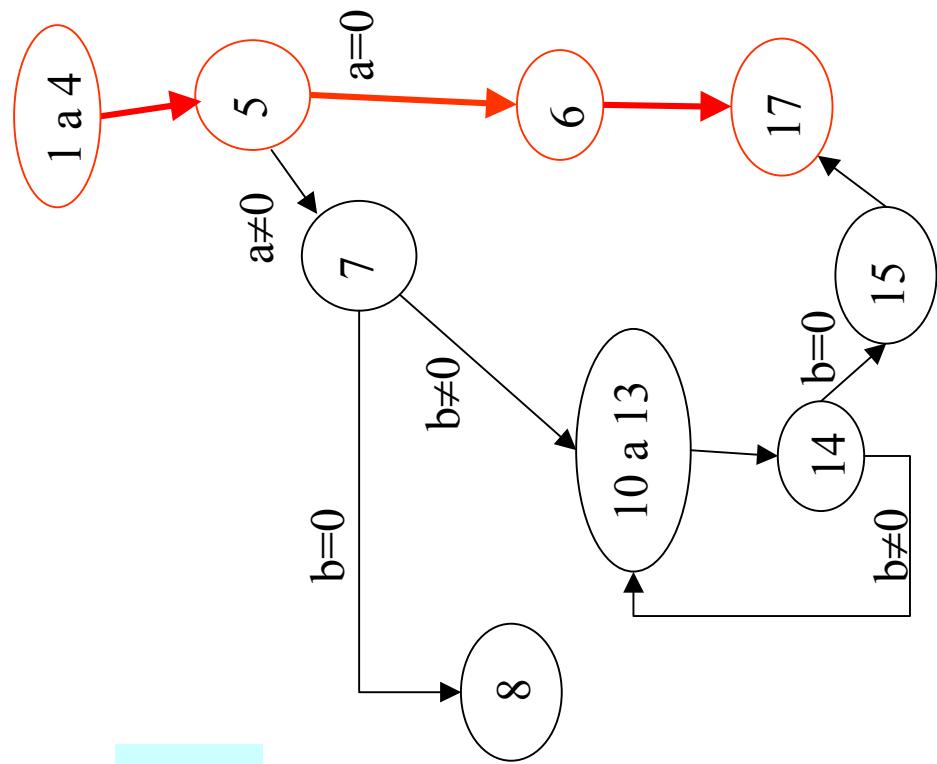




Teste de decisões

Critério: cada ramo deve ser percorrido pelo menos 1 vez

ramos
predicados dados
 $\{(1, 5), (5, 6), (6, 17)\}$ $a = 0, \forall b$ $(0, -1)$





Teste de decisões

Critério: cada ramo deve ser percorrido pelo menos 1 vez

ramos

$\{(1,5), (5,6), (6,17)\}$

predicados dados

$a = 0, \forall b$

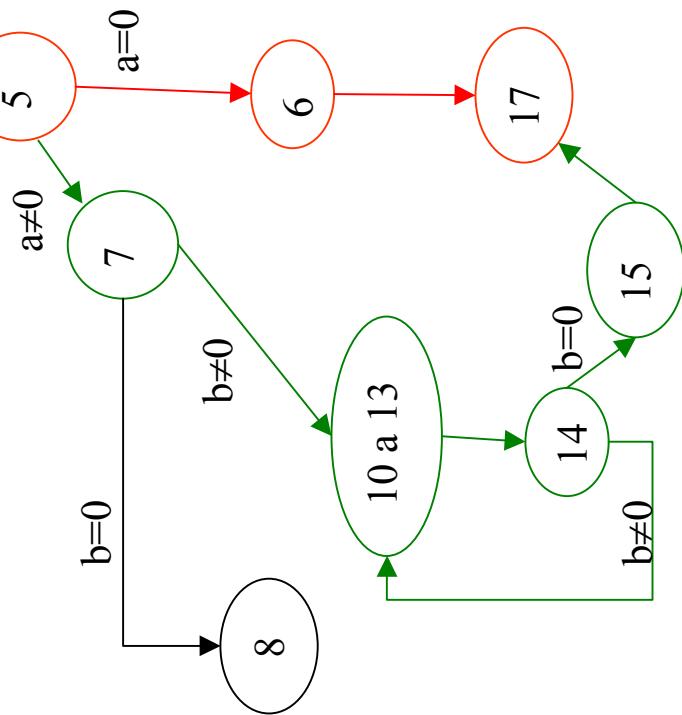
$(0, -1)$

$\{(1,5), (5,7), (7,10), (10,14), a \neq 0, b \neq 0\}$

$(14,10), (14,15), (15,17)\}$

$(4, 6)$

$(4, 6)$

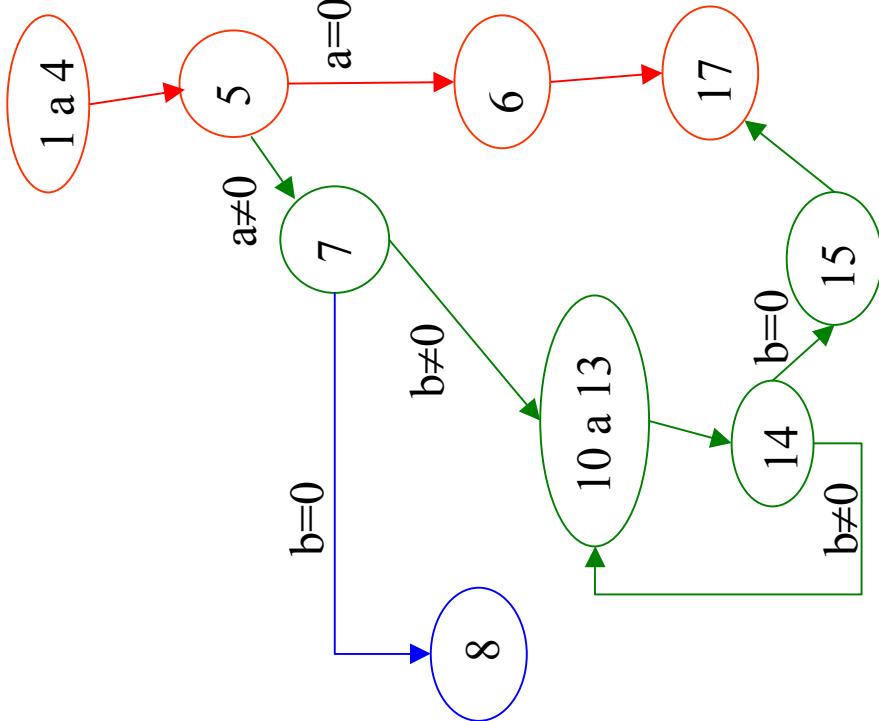




Teste de decisões

Critério: cada ramo deve ser percorrido pelo menos 1 vez

ramos	predicados	dados
$\{(1, 5), (5, 6), (6, 17),$	$a = 0, \forall b$	(0, -1)
$\{(1, 5), (5, 7), (7, 10), (10, 14), a \neq 0, b \neq 0$ $(14, 10), (10, 8), (8, 5), (5, 7)\}$	$a \neq 0, b > a$	(4, 6)
$\{(1, 5), (5, 7), (7, 8)\}$	$a \neq 0, b = 0$	(4, 0)





O que fazer com decisões compostas?

```
if (a > 0 && c < 3) {x = a + 1;}  
if (b == 2 || d ≥ 4) {y = d - 2;}
```

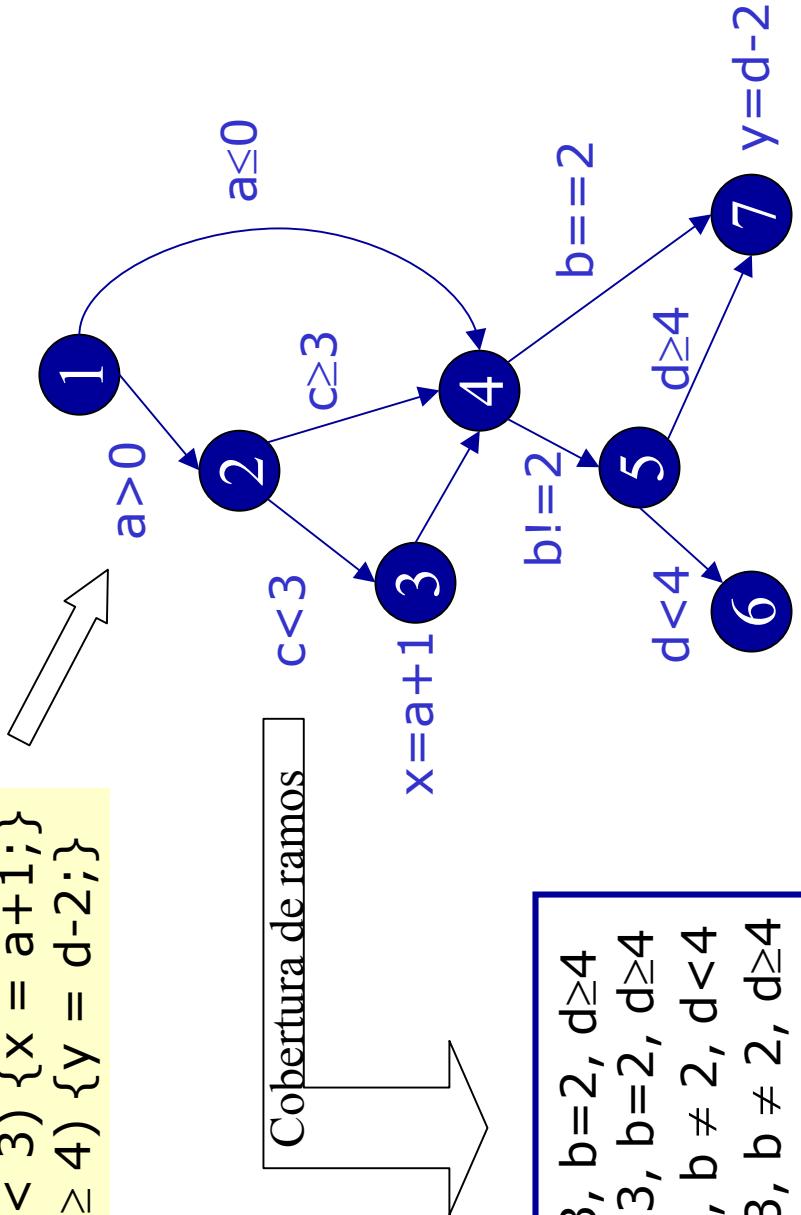
- Cobertura de condições:
 - Critério: **cada condição deve ser avaliada uma vez como V e F**
 - $a > 0, c < 3, b = 2, d \geq 4 \Leftrightarrow$ todas as condições = V
 - $a \leq 0, c \geq 3, b \neq 2, d < 4 \Leftrightarrow$ todas as condições = F
 - Mas o que acontece se:
 - $a \leq 0, c < 3, b \neq 2, d \geq 4$ ou
 - $a > 0, c \geq 3, b = 2, d < 4$ ou
 - ...

☞ **Como cobrir todas as combinações interessantes?**



Testes de múltiplas condições

```
if (a > 0 && c < 3) {x = a+1;}  
if (b == 2 || d ≥ 4) {y = d-2;}
```



$a > 0, c < 3, b = 2, d \geq 4$
 $a \leq 0, c \geq 3, b = 2, d \geq 4$
 $a > 0, c \geq 3, b \neq 2, d < 4$
 $a \leq 0, c \geq 3, b \neq 2, d \geq 4$

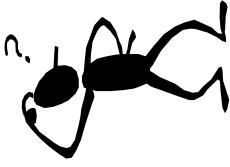


Cobertura de caminhos

- **Caminho:** seqüência de arestas partindo de um nó origem a um nó destino
- **Caminho completo:** caminho de um nó de entrada a um nó de saída
- **Ciclo** (ou laço): um caminho em que um nó (ou aresta) é visitado mais de uma vez
- **Caminho livre de ciclos:** um caminho em que nenhum nó (ou aresta) é visitado mais de uma vez

Critério: todos os caminhos possíveis do grafo devem ser percorridos pelo menos 1 vez

Esse critério é sempre factível?





Cobertura de caminhos

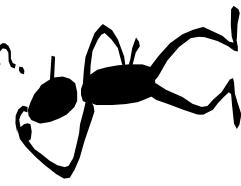
Critério: todos os caminhos possíveis do grafo devem ser percorridos pelo menos 1 vez

- Dificuldade 1: Grafos com ciclos
 - nº de caminhos pode ser indeterminado ou infinito
- ↓
- necessidade de critérios que permitam limitar o nº de caminhos:
 - Cobertura de caminhos relevantes
 - Cobertura de caminhos básicos
 - Cobertura de laços



Cobertura de caminhos básicos

- Critério proposto McCabe
 - Em vez de todos os caminhos, busca os caminhos linearmente independentes:
 - **Caminho linearmente independente (L.I.)**: dois caminhos são independentes quando nenhum contém nós internos do outro
- É possível calcular o n° de caminhos L.I?
 - ↓
 - Um caminho L.I. contém pelo menos 1 nova aresta não coberta pelos demais





Número de caminhos L.I.

- O n° de caminhos independentes é dado pela **complexidade ciclomática** ($V(G)$) :

$$V(G) = \text{nº de nós predicados} + 1 \quad V(G) = \text{nº de arestas} - \text{nº de nós} + 2$$

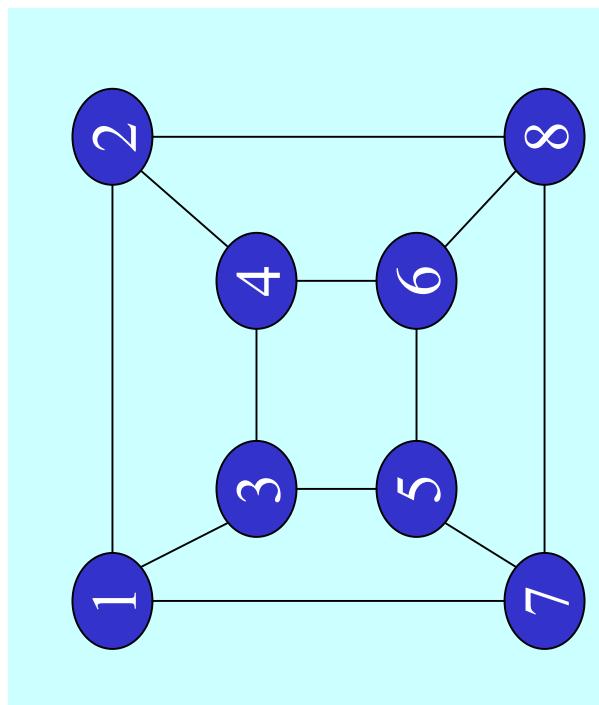
$$V(G) = \text{nº de regiões do grafo}$$

- É um **limite máximo** do n° de testes necessários para cobrir todas as instruções de um programa
- Uma vez que todos os outros caminhos do grafo são combinações dos caminhos independentes $\Rightarrow V(G)$ é o **limite inferior** do n° de testes de caminhos



De onde vem $V(G)$? (1)

- Euler (séc. XVIII) percebeu que um grafo simples, planar (desenhado em um plano) e conexo divide o plano em um certo n° de **regiões** totalmente fechadas + a região exterior, que é infinita



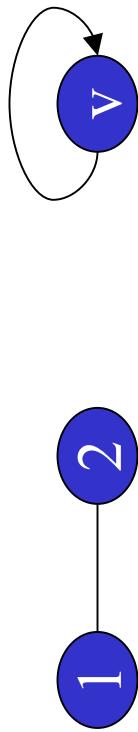
Só tem intersecção entre 2 arestas nos vértices inicial e final

Qtas regiões tem esse grafo?



De onde vem $V(G)? (2)$

- Foi estabelecida então uma relação entre o número de arestas (a), o número de vértices (v) e o número de regiões (r) para tais grafos que é dada pela fórmula:
 - **$r + n = a + 2 \rightarrow r = a - n + 2$** ← conhecida como fórmula de Euler para grafos planares
 - Se $n \geq 3$ então $a \leq 3n - 6$



☞ **r é o limite máximo de caminhos L.I.**



Combinação linear de vetores

Fonte: <http://www.fem.unicamp.br/~em421/semII-1999/textos/vetor.pdf>

- Um conjunto de vetores $\{v_i\} = \{v_1, v_2, \dots, v_n\}$ é linearmente independente se a combinação linear:

$$\sum_{i=1}^n \alpha_i v_i = 0$$

$\alpha_1 = \alpha_2 = \dots = \alpha_n = 0$

$$\alpha_i \neq 0$$

$\alpha_1 = \alpha_2 = \dots = \alpha_n = 0$

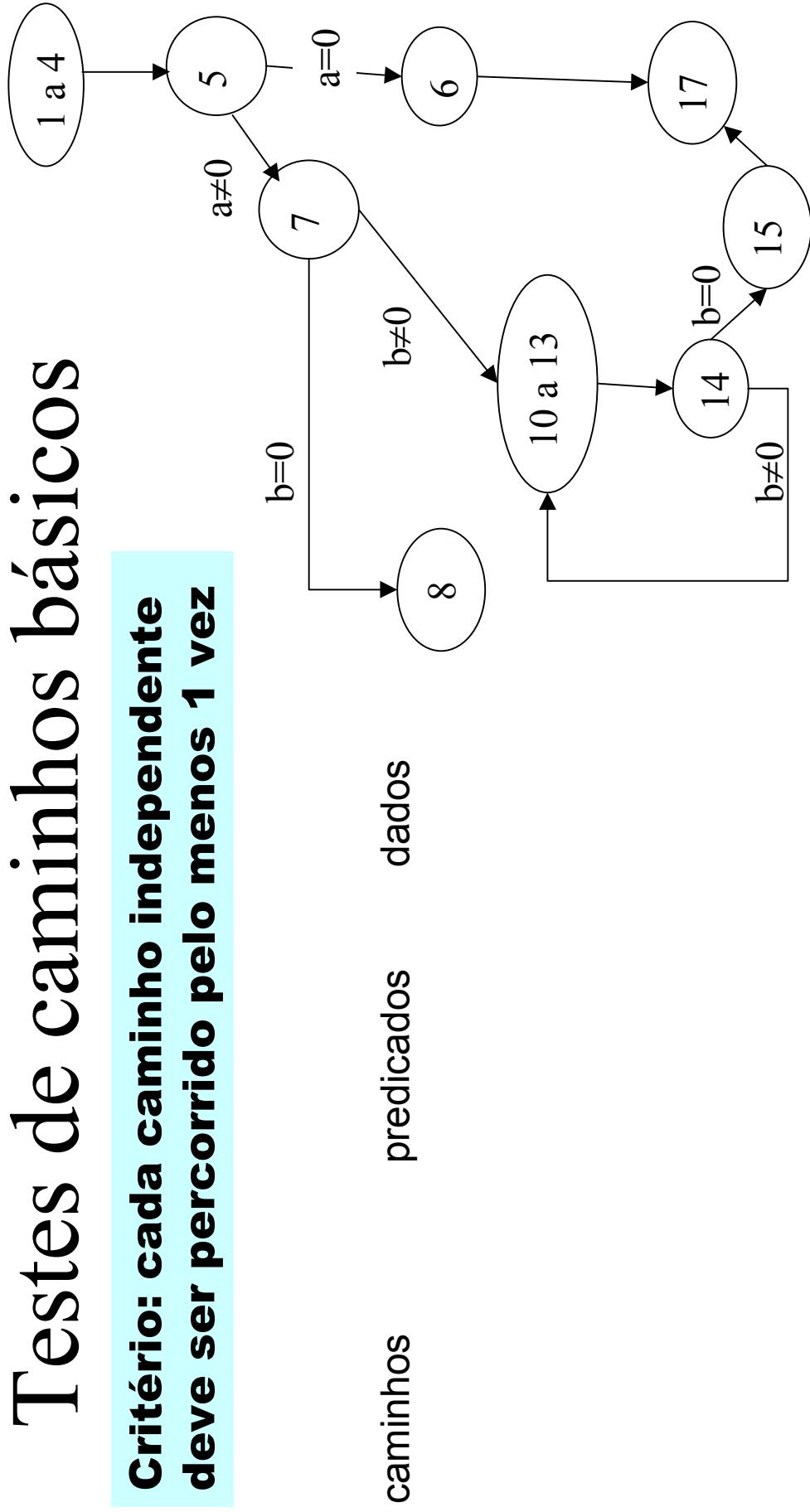
é válida somente se $\alpha_1 = \alpha_2 = \dots = \alpha_n = 0$

Senão, se existir algum $\alpha_i \neq 0$ então os vetores são linearmente dependentes



Testes de caminhos básicos

Critério: cada caminho independente deve ser percorrido pelo menos 1 vez



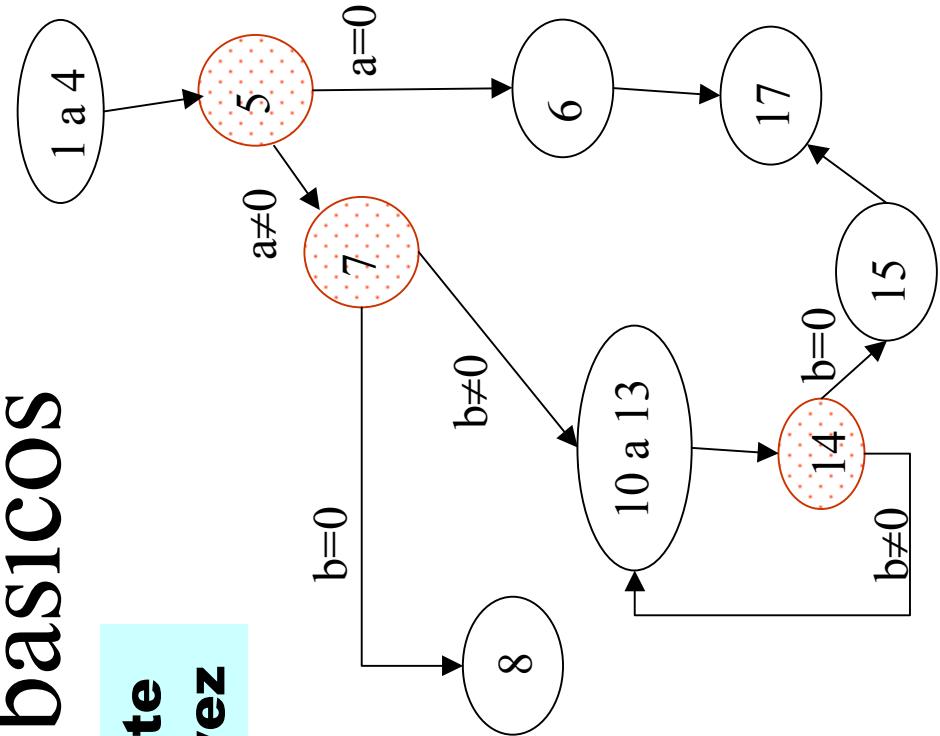


Testes de caminhos básicos

Critério: cada caminho independente deve ser percorrido pelo menos 1 vez

$$V(G) = \text{nº condições} + 1 = 4$$

caminhos predicados dados





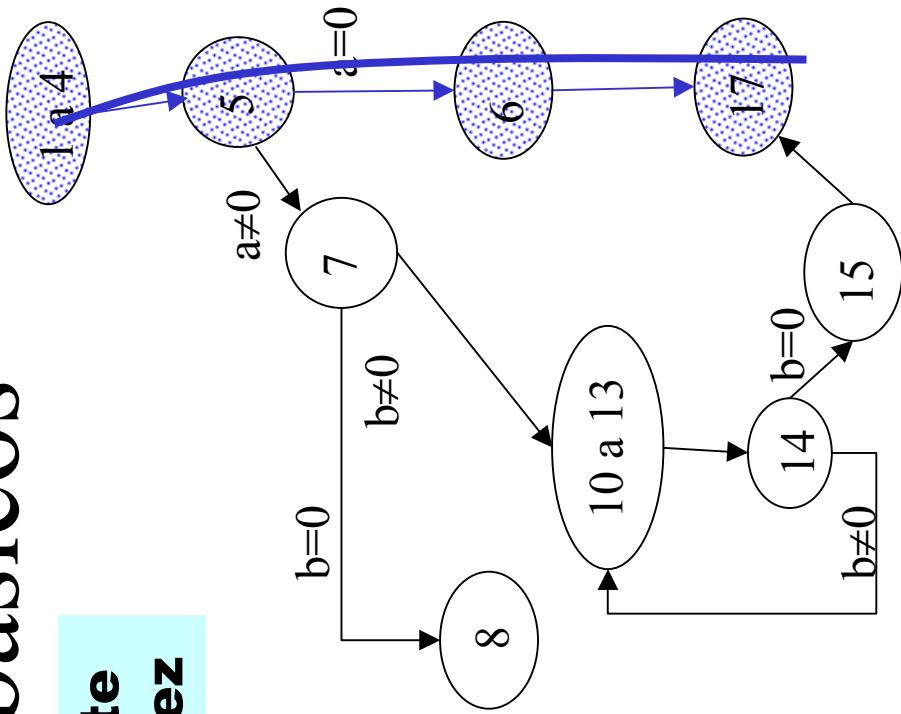
Testes de caminhos básicos

Critério: cada caminho independente deve ser percorrido pelo menos 1 vez

$$V(G) = \text{nº condições} + 1 = 4$$

caminhos predicados dados

$\{1-5-6-17\}$ $a = 0, \forall b$ $(0, -2)$



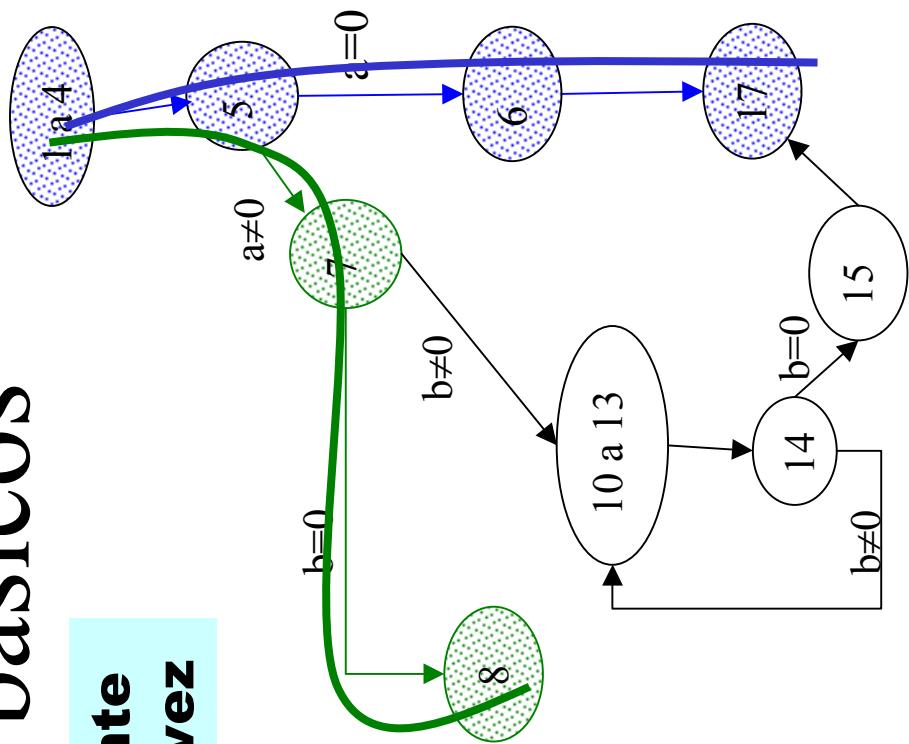


Testes de caminhos básicos

Critério: cada caminho independente deve ser percorrido pelo menos 1 vez

$$V(G) = \text{nº condições} + 1 = 4$$

caminhos	predicados	dados
$\{1-5-6-17\}$	$a = 0, \forall b$	$(0, -2)$
$\{1-\underline{5-7-8}\}$	$a \neq 0, b=0$	$(4, 0)$



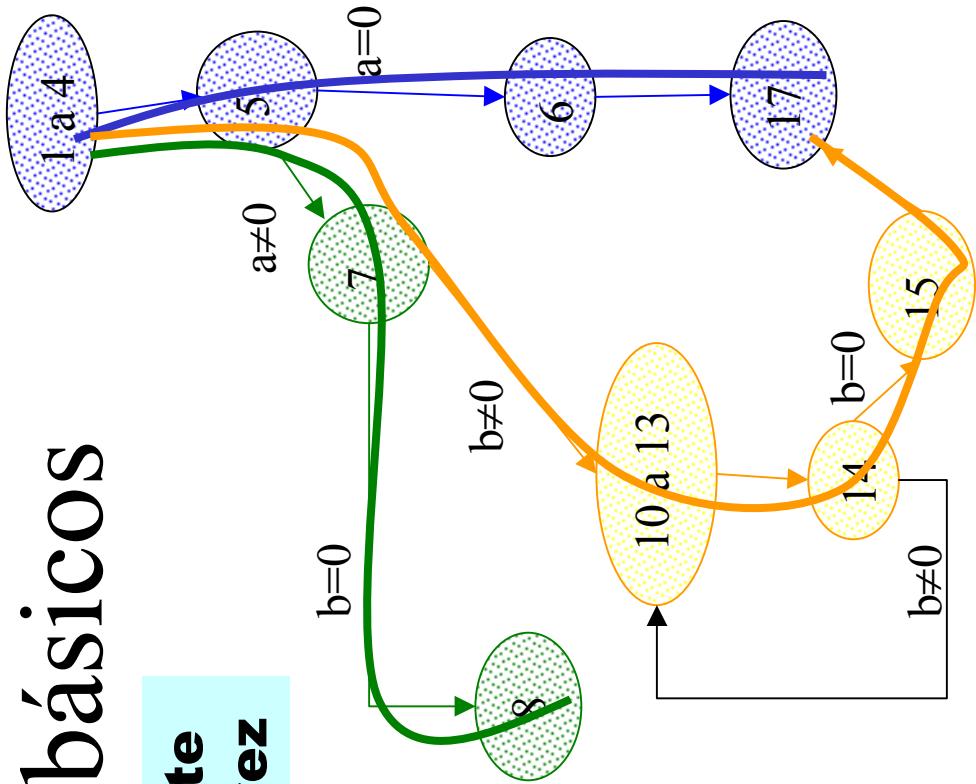


Testes de caminhos básicos

Critério: cada caminho independente deve ser percorrido pelo menos 1 vez

$$V(G) = \text{nº condições} + 1 = 4$$

caminhos	predicados	dados
{1-5-6-17}	$a = 0, \forall b$	(0, -2)
{1-5-7-8}	$a \neq 0, b=0$	(4, 0)
<u>{1-5-7-10-14}</u>	$a \neq 0, b \neq 0,$ $a \bmod b = 0$	(8, 4)
<u>15-17}</u>	$b \neq 0$	





Testes de caminhos básicos

Critério: cada caminho independente deve ser percorrido pelo menos 1 vez

$$V(G) = \text{nº condições} + 1 = 4$$

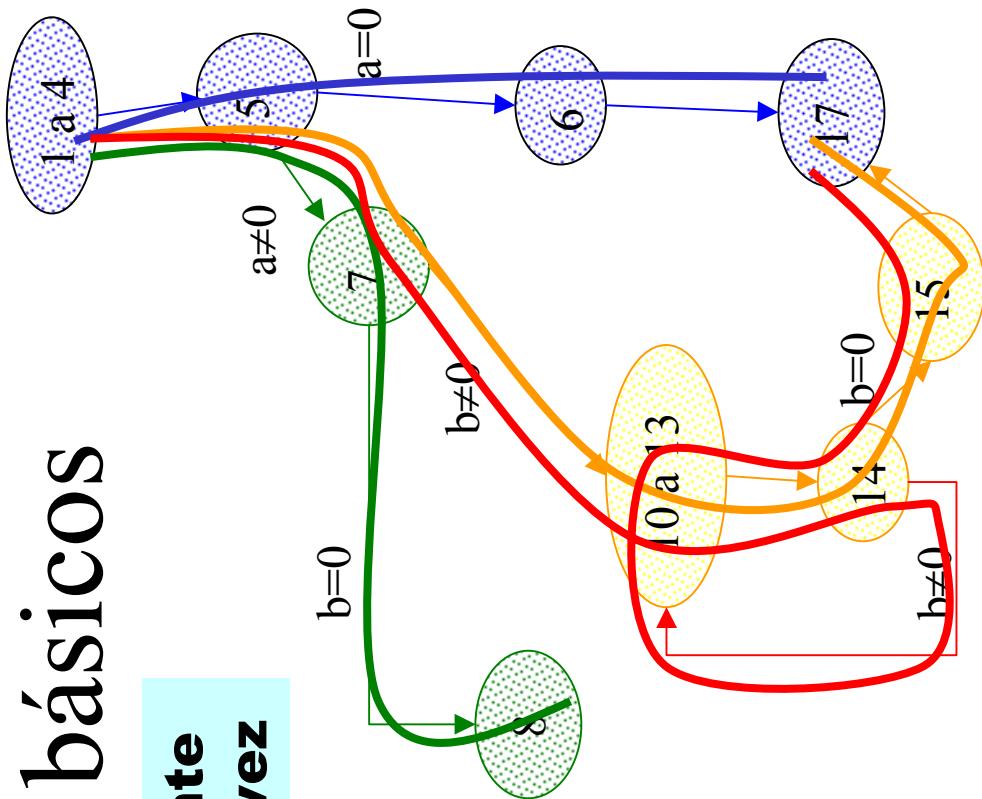
caminhos predicados dados

$\{1-5-6-17\}$ $a = 0, \forall b$ $(0, -2)$

$\{1-5-7-8\}$ $a \neq 0, b=0$ $(4, 0)$

$\frac{\{1-5-7-10-14-15-17\}}{15-17}$ $a \neq 0, b \neq 0$
 $a \bmod b = 0$ $(8, 4)$

$\frac{\{1-5-7-10-14-10-14-15-17\}}{14-15-17}$ $a \neq 0, b \neq 0$
 $a \bmod b \neq 0$ $(6, 4)$

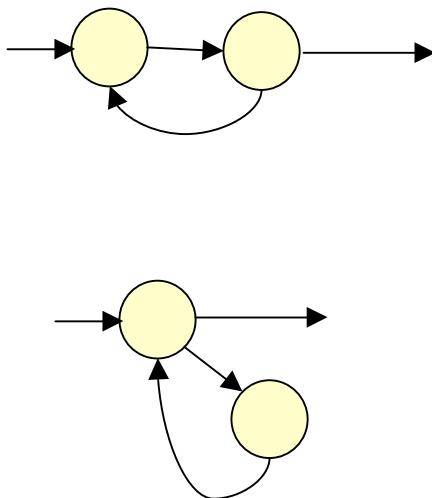




Cobertura de laços

laços simples • Procedimento:

- ① pule o laço
- ② passe pelo laço 1 vez
- ③ passe pelo laço m vezes, onde $m < \text{max}$
- ④ passe pelo laço max-1 vezes
- ⑤ passe pelo laço max vezes
- ⑥ tente passar pelo laço max+1 vezes





Cobertura de laços - exemplo

- ① Pule o laço

Predicado: $a=0$ ou

$$a \neq 0 \wedge b = 0$$

- ② Passe pelo laço 1 vez

Predicado: $a \neq 0 \wedge b \neq 0 \wedge a \bmod b = 0$

- ③ Passe pelo laço m vezes, onde $m < \max$

Predicado: $a \neq 0 \wedge b \neq 0 \wedge ?$

- ④ Passe pelo laço $\max - 1$ vezes

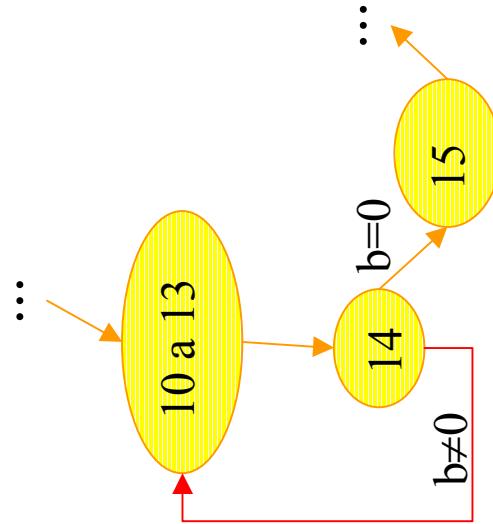
Predicado: ?

- ⑤ Passe pelo laço \max vezes

Predicado: ?

- ⑥ Tente passar pelo laço $\max + 1$ vezes

Predicado: ?

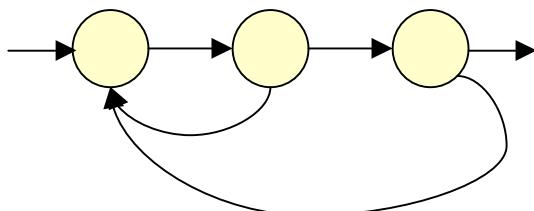




Cobertura de laços

laços aninhados • Procedimento:

- ① comece pelo laço mais interno.
Fixe os outros nos valores mínimos
- ② realize testes para laços simples
- ③ caminhe para fora, realizando testes no laço seguinte e mantendo os demais nos valores mínimos
- ④ continue até que todos tenham sido testados





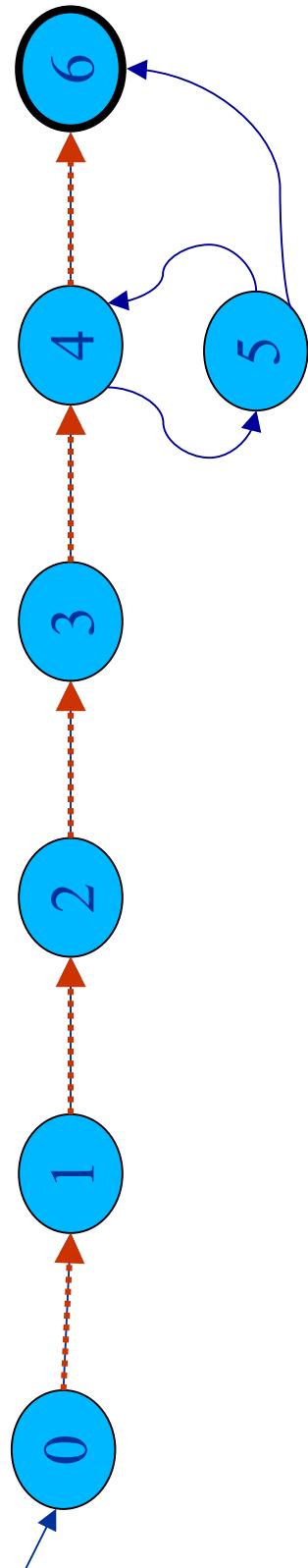
Testes de caminhos

- Dificuldade 2: alguns requisitos de teste não podem ser satisfeitos por nenhum caso de teste → são chamados **requisitos inviáveis**
 - Requisito de teste está em subgrafo que não é semanticamente alcançável
 - Trecho de código « morto »
 - Subcaminho com predicados contraditórios (ex.: $a > b$ e $a \leq b$)
 - Determinar se requisitos de teste são inviáveis é um problema **indecidível**
- Ammann e Offutt propõem o uso de percursos com desvios e contornos como forma de satisfazer critérios com requisitos de teste inviáveis:
 - Ao invés de cobertura de caminhos básicos → cobertura de caminhos com desvios ou alternativas
 - ⌚ Critério de teste fica mais fraco



Caminho requerido

Percorso original:

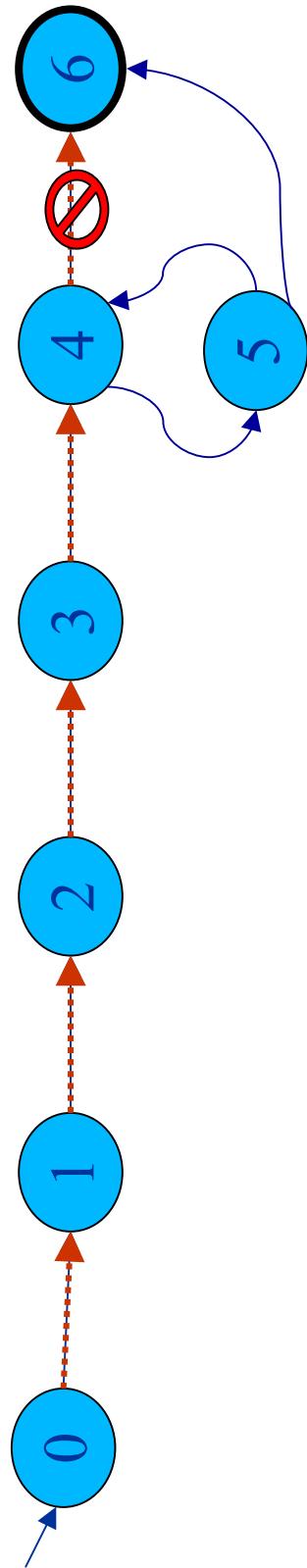




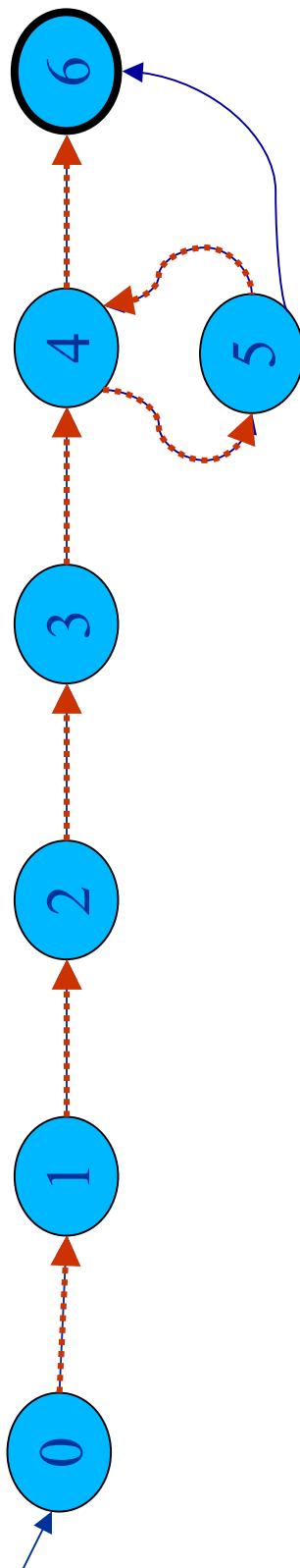
Caminho com percurso alternativo

alternativo

Percorso original:



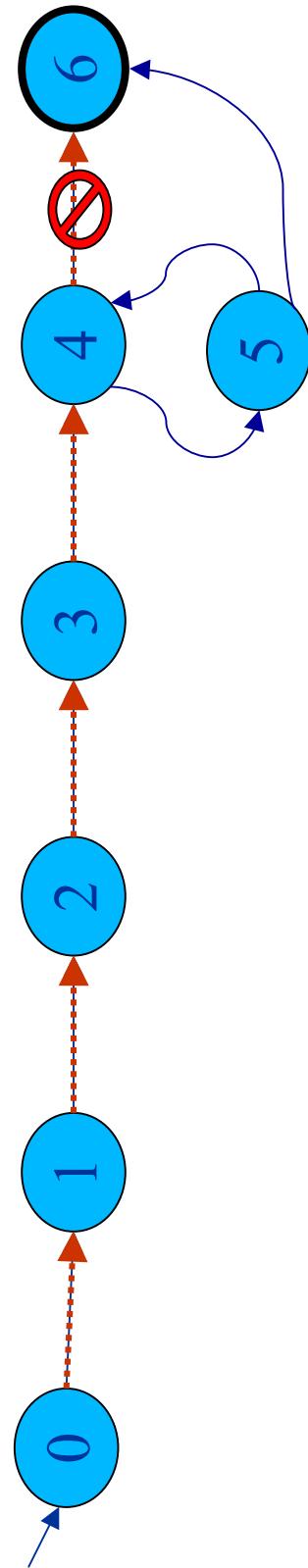
Percorso alternativo:



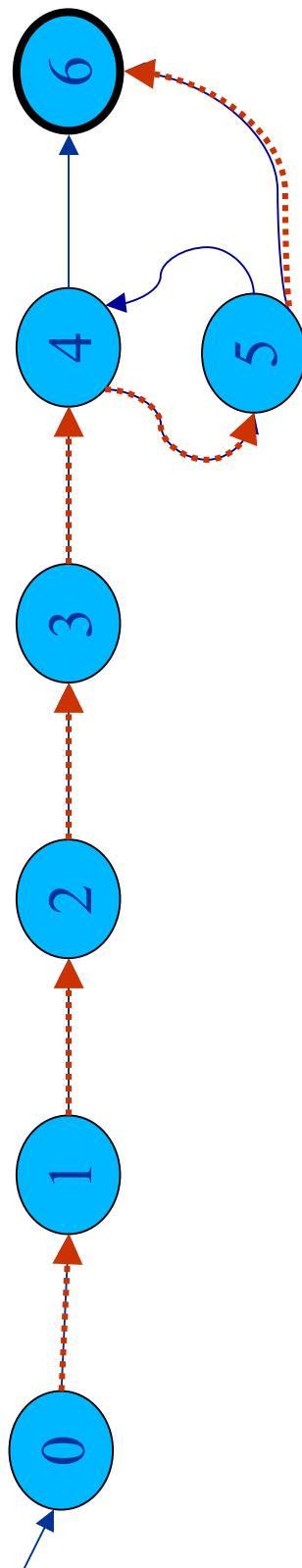


Caminho com desvio

Percorso original:



Percorso com desvio:





Em suma ...

- Grafos
 - Representam fluxo de controle (ou de dados):
 - Comportamental (testes de caixa preta)
 - Estrutural (testes de caixa branca)
 - Critérios baseados em grafos
- Os mesmos para testes caixa branca ou caixa preta
 - O que muda são os requisitos de teste: casos de uso, instruções do código, ...
- Cobertura de elementos do grafo:
 - Nós
 - Arestas
 - ...



Automação

- Para usar grafos para projetar casos de teste deve-se proceder de acordo com os seguintes passos:
 1. Defina o grafo
 2. Defina o tipo da relação
 3. Percorra o grafo para atender a critérios selecionados
- Algoritmos:
 - Busca em profundidade
 - Busca em largura
- Limitações:
 - Loops (ex.: busca em profundidade k-limitada)
 - Leva em conta aspectos estruturais, mas não semânticos → caminhos inviáveis
- Alternativas?





Analisar a cobertura

- Não gera testes automaticamente, mas analisa a cobertura dos casos de teste gerados
- Existem inúmeros analisadores de cobertura, tanto comerciais quanto livres:
 - Coverlipse
 - <http://coverlipse.sourceforge.net/>
 - PureCoverage (Rational/IBM)
 - EMMA/ECLEMMΑ
 - <http://emma.sourceforge.net/>
 - Outras ferramentas livres:
 - <http://java-source.net/open-source/code-coverage>



Coveripse

The screenshot shows the Eclipse IDE interface with the following details:

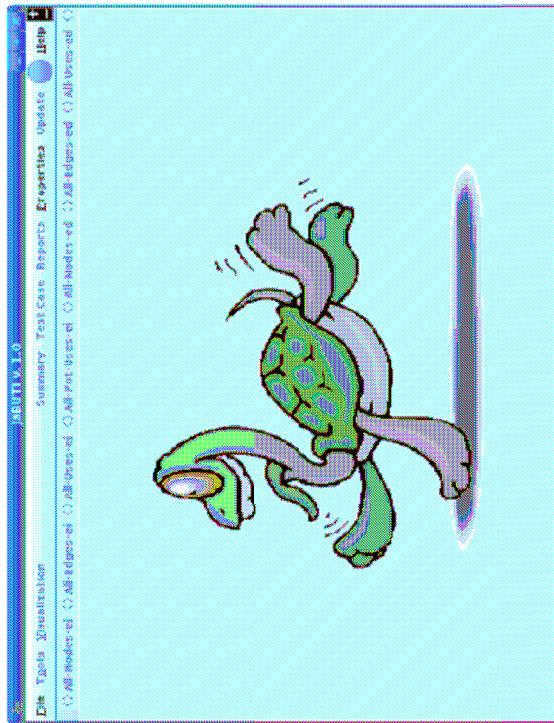
- Top Bar:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Toolbar:** Standard icons for file operations like Open, Save, Print, etc.
- Editor Area:** Displays Java code for a class named `Bolts`. The code includes imports for `java.util.List`, `java.util.Map`, and `java.util.Set`. It defines a class `Bolts` with a constructor taking a `Map<String, Set<String>>` parameter. The class has a private field `bolts` of type `Map<String, Set<String>>`. The `add` method adds entries to this map. The `getBolts` method returns the map.
- Outline View (Left):** Shows the package structure: `com.bolts` contains `Bolts`.
- Outline View (Bottom):** Shows the outline of the `Bolts` class, including its fields (`bolts`), methods (`add`, `getBolts`), and imports.
- Coverage Details View (Bottom Right):** A table titled "Content Description of Coverage Metrics View" showing coverage statistics for methods. It includes columns for "Metric", "line", "counted lines", and "uncovered lines". The table shows four rows: "The line is fully covered" (line 21), "The line is fully covered" (line 22), "The line is fully covered" (line 23), and "The line is fully covered" (line 24).

Grafos e Testes



Jabuti

- JaBUTi (Java Bytecode Understanding and Testing)
 - Ferramenta que automatiza critérios de teste caixa branca
 - Foi apresentada no XVII Simpósio Brasileiro de Engenharia de Software (2003)
 - Cobre tanto critérios de fluxo de controle quanto de dados





Preparação para os testes

The screenshot shows the JaBUTI Coverage Test tool interface. The main window displays a Java code editor with the following code:`package motoPark;
public class MotoPark {
 public int calcMotoParks(int[] ipo) {
 int z = ipo[0];
 int r = ipo[1];
 while (r != 0) {
 if (z % r == 0) {
 ipo[2] += 1;
 }
 r = ipo[3];
 }
 return ipo[2];
 }
}`

Below the code editor, a bar chart visualizes the coverage status for each line of code. The legend indicates:

- Green: Line 0: 0% (uncovered)
- Yellow: Line 1: 1% (partially covered)
- Red: Lines 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 429, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 459, 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558, 559, 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 569, 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 579, 580, 581, 582, 583, 584, 585, 586, 587, 588, 589, 589, 590, 591, 592, 593, 594, 595, 596, 597, 598, 599, 599, 600, 601, 602, 603, 604, 605, 606, 607, 608, 609, 609, 610, 611, 612, 613, 614, 615, 616, 617, 618, 619, 619, 620, 621, 622, 623, 624, 625, 626, 627, 628, 629, 629, 630, 631, 632, 633, 634, 635, 636, 637, 638, 639, 639, 640, 641, 642, 643, 644, 645, 646, 647, 648, 649, 649, 650, 651, 652, 653, 654, 655, 656, 657, 658, 659, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679, 679, 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 689, 690, 691, 692, 693, 694, 695, 696, 697, 698, 699, 699, 700, 701, 702, 703, 704, 705, 706, 707, 708, 709, 709, 710, 711, 712, 713, 714, 715, 716, 717, 718, 719, 719, 720, 721, 722, 723, 724, 725, 726, 727, 728, 729, 729, 730, 731, 732, 733, 734, 735, 736, 737, 738, 739, 739, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749, 749, 750, 751, 752, 753, 754, 755, 756, 757, 758, 759, 759, 760, 761, 762, 763, 764, 765, 766, 767, 768, 769, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778, 779, 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 799, 800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818, 819, 819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829, 829, 830, 831, 832, 833, 834, 835, 836, 837, 838, 839, 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849, 849, 850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 859, 860, 861, 862, 863, 864, 865, 866, 867, 868, 869, 869, 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888, 889, 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 909, 910, 911, 912, 913, 914, 915, 916, 917, 918, 919, 919, 920, 921, 922, 923, 924, 925, 926, 927, 928, 929, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 939, 940, 941, 942, 943, 944, 945, 946, 947, 948, 949, 949, 950, 951, 952, 953, 954, 955, 956, 957, 958, 959, 959, 960, 961, 962, 963, 964, 965, 966, 967, 968, 969, 969, 970, 971, 972, 973, 974, 975, 976, 977, 978, 979, 979, 980, 981, 982, 983, 984, 985, 986, 987, 988, 989, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999, 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009, 1009, 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1019, 1020, 1021, 1022, 1023, 1024, 1025, 1026, 1027, 1028, 1029, 1029, 1030, 1031, 1032, 1033, 1034, 1035, 1036, 1037, 1038, 1039, 1039, 1040, 1041, 1042, 1043, 1044, 1045, 1046, 1047, 1048, 1049, 1049, 1050, 1051, 1052, 1053, 1054, 1055, 1056, 1057, 1058, 1059, 1059, 1060, 1061, 1062, 1063, 1064, 1065, 1066, 1067, 1068, 1069, 1069, 1070, 1071, 1072, 1073, 1074, 1075, 1076, 1077, 1078, 1079, 1079, 1080, 1081, 1082, 1083, 1084, 1085, 1086, 1087, 1088, 1089, 1089, 1090, 1091, 1092, 1093, 1094, 1095, 1096, 1097, 1098, 1099, 1099, 1100, 1101, 1102, 1103, 1104, 1105, 1106, 1107, 1108, 1109, 1109, 1110, 1111, 1112, 1113, 1114, 1115, 1116, 1117, 1118, 1119, 1119, 1120, 1121, 1122, 1123, 1124, 1125, 1126, 1127, 1128, 1129, 1129, 1130, 1131, 1132, 1133, 1134, 1135, 1136, 1137, 1138, 1139, 1139, 1140, 1141, 1142, 1143, 1144, 1145, 1146, 1147, 1148, 1149, 1149, 1150, 1151, 1152, 1153, 1154, 1155, 1156, 1157, 1158, 1159, 1159, 1160, 1161, 1162, 1163, 1164, 1165, 1166, 1167, 1168, 1169, 1169, 1170, 1171, 1172, 1173, 1174, 1175, 1176, 1177, 1178, 1179, 1179, 1180, 1181, 1182, 1183, 1184, 1185, 1186, 1187, 1188, 1189, 1189, 1190, 1191, 1192, 1193, 1194, 1195, 1196, 1197, 1198, 1199, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 1209, 1210, 1211, 1212, 1213, 1214, 1215, 1216, 1217, 1218, 1219, 1219, 1220, 1221, 1222, 1223, 1224, 1225, 1226, 1227, 1228, 1229, 1229, 1230, 1231, 1232, 1233, 1234, 1235, 1236, 1237, 1238, 1239, 1239, 1240, 1241, 1242, 1243, 1244, 1245, 1246, 1247, 1248, 1249, 1249, 1250, 1251, 1252, 1253, 1254, 1255, 1256, 1257, 1258, 1259, 1259, 1260, 1261, 1262, 1263, 1264, 1265, 1266, 1267, 1268, 1269, 1269, 1270, 1271, 1272, 1273, 1274, 1275, 1276, 1277, 1278, 1279, 1279, 1280, 1281, 1282, 1283, 1284, 1285, 1286, 1287, 1288, 1289, 1289, 1290, 1291, 1292, 1293, 1294, 1295, 1296, 1297, 1298, 1299, 1299, 1300, 1301, 1302, 1303, 1304, 1305, 1306, 1307, 1308, 1309, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317, 1318, 1319, 1319, 1320, 1321, 1322, 1323, 1324, 1325, 1326, 1327, 1328, 1329, 1329, 1330, 1331, 1332, 1333, 1334, 1335, 1336, 1337, 1338, 1339, 1339, 1340, 1341, 1342, 1343, 1344, 1345, 1346, 1347, 1348, 1349, 1349, 1350, 1351, 1352, 1353, 1354, 1355, 1356, 1357, 1358, 1359, 1359, 1360, 1361, 1362, 1363, 1364, 1365, 1366, 1367, 1368, 1369, 1369, 1370, 1371, 1372, 1373, 1374, 1375, 1376, 1377, 1378, 1379, 1379, 1380, 1381, 1382, 1383, 1384, 1385, 1386, 1387, 1388, 1389, 1389, 1390, 1391, 1392, 1393, 1394, 1395, 1396, 1397, 1398, 1399, 1399, 1400, 1401, 1402, 1403, 1404, 1405, 1406, 1407, 1408, 1409, 1409, 1410, 1411, 1412, 1413, 1414, 1415, 1416, 1417, 1418, 1419, 1419, 1420, 1421, 1422, 1423, 1424, 1425, 1426, 1427, 1428, 1429, 1429, 1430, 1431, 1432, 1433, 1434, 1435, 1436, 1437, 1438, 1439, 1439, 1440, 1441, 1442, 1443, 1444, 1445, 1446, 1447, 1448, 1449, 1449, 1450, 1451, 1452, 1453, 1454, 1455, 1456, 1457, 1458, 1459, 1459, 1460, 1461, 1462, 1463, 1464, 1465, 1466, 1467, 1468, 1469, 1469, 1470, 1471, 1472, 1473, 1474, 1475, 1476, 1477, 1478, 1479, 1479, 1480, 1481, 1482, 1483, 1484, 1485, 1486, 1487, 1488, 1489, 1489, 1490, 1491, 1492, 1493, 1494, 1495, 1496, 1497, 1498, 1499, 1499, 1500, 1501, 1502, 1503, 1504, 1505, 1506, 1507, 1508, 1509, 1509, 1510, 1511, 1512, 1513, 1514, 1515, 1516, 1517, 1518, 1519, 1519, 1520, 1521, 1522, 1523, 1524, 1525, 1526, 1527, 1528, 1529, 1529, 1530, 1531, 1532, 1533, 1534, 1535, 1536, 1537, 1538, 1539, 1539, 1540, 1541, 1542, 1543, 1544, 1545, 1546, 1547, 1548, 1549, 1549, 1550, 1551, 1552, 1553, 1554, 1555, 1556, 1557, 1558, 1559, 1559, 1560, 1561, 1562, 1563, 1564, 1565, 1566, 1567, 1568, 1569, 1569, 1570, 1571, 1572, 1573, 1574, 1575, 1576, 1577, 1578, 1579, 1579, 1580, 1581, 1582, 1583, 1584, 1585, 1586, 1587, 1588, 1589, 1589, 1590, 1591, 1592, 1593, 1594, 1595, 1596, 1597, 1598, 1599, 1599, 1600, 1601, 1602, 1603, 1604, 1605, 1606, 1607, 1608, 1609, 1609, 1610, 1611, 1612, 1613, 1614, 1615, 1616, 1617, 1618, 1619, 1619, 1620, 1621, 1622, 1623, 1624, 1625, 1626, 1627, 1628, 1629, 1629, 1630, 1631, 1632, 1633, 1634, 1635, 1636, 1637, 1638, 1639, 1639, 1640, 1641, 1642, 1643, 1644, 1645, 1646, 1647, 1648, 1649, 1649, 1650, 1651, 1652, 1653, 1654, 1655, 1656, 1657, 1658, 1659, 1659, 1660, 1661, 1662, 1663, 1664, 1665, 1666, 1667, 1668, 1669, 1669, 1670, 1671, 1672, 1673, 1674, 1675, 1676, 1677, 1678, 1679, 1679, 1680, 1681, 1682, 1683, 1684, 1685, 1686, 1687, 1688, 1689, 1689, 1690, 1691, 1692, 1693, 1694, 1695, 1696, 1697, 1698, 1699, 1699, 1700, 1701, 1702, 1703, 1704, 1705, 1706, 1707, 1708, 1709, 1709, 1710, 1711, 1712, 1713, 1714, 1715, 1716, 1717, 1718, 1719, 1719, 1720, 1721, 1722, 1723, 1724, 1725, 1726, 1727, 1728, 1729, 1729, 1730, 1731, 1732, 1733, 1734, 1735, 1736, 1737, 1738, 1739, 1739, 1740, 1741, 1742, 1743, 1744, 1745, 1746, 1747, 1748, 1749, 1749, 1750, 1751, 1752, 1753, 1754, 1755, 1756, 1757, 1758, 1759, 1759, 1760, 1761, 1762, 1763, 1764, 1765, 1766, 1767, 1768, 1769, 1769, 1770, 1771, 1772, 1773, 1774, 1775, 1776, 1777, 1778, 1779, 1779, 1780, 1781, 1782, 1783, 1784, 1785, 1786, 1787, 1788, 1789, 1789, 1790, 1791, 1792, 1793, 1794, 1795, 1796, 1797, 1798, 1799, 1799, 1800, 1801, 1802, 1803, 1804, 1805, 1806, 1807, 1808, 1809, 1809, 1810, 1811, 1812, 1813, 1814, 1815, 1816, 1817, 1818, 1819, 1819, 1820, 1821, 1822, 1823, 1824, 1825, 1826, 1827, 1828, 1829, 1829, 1830, 1831, 1832, 1833, 1834, 1835, 1836, 1837, 1838, 1839,



Exemplo de resultados

Requisitos cobertos e não cobertos

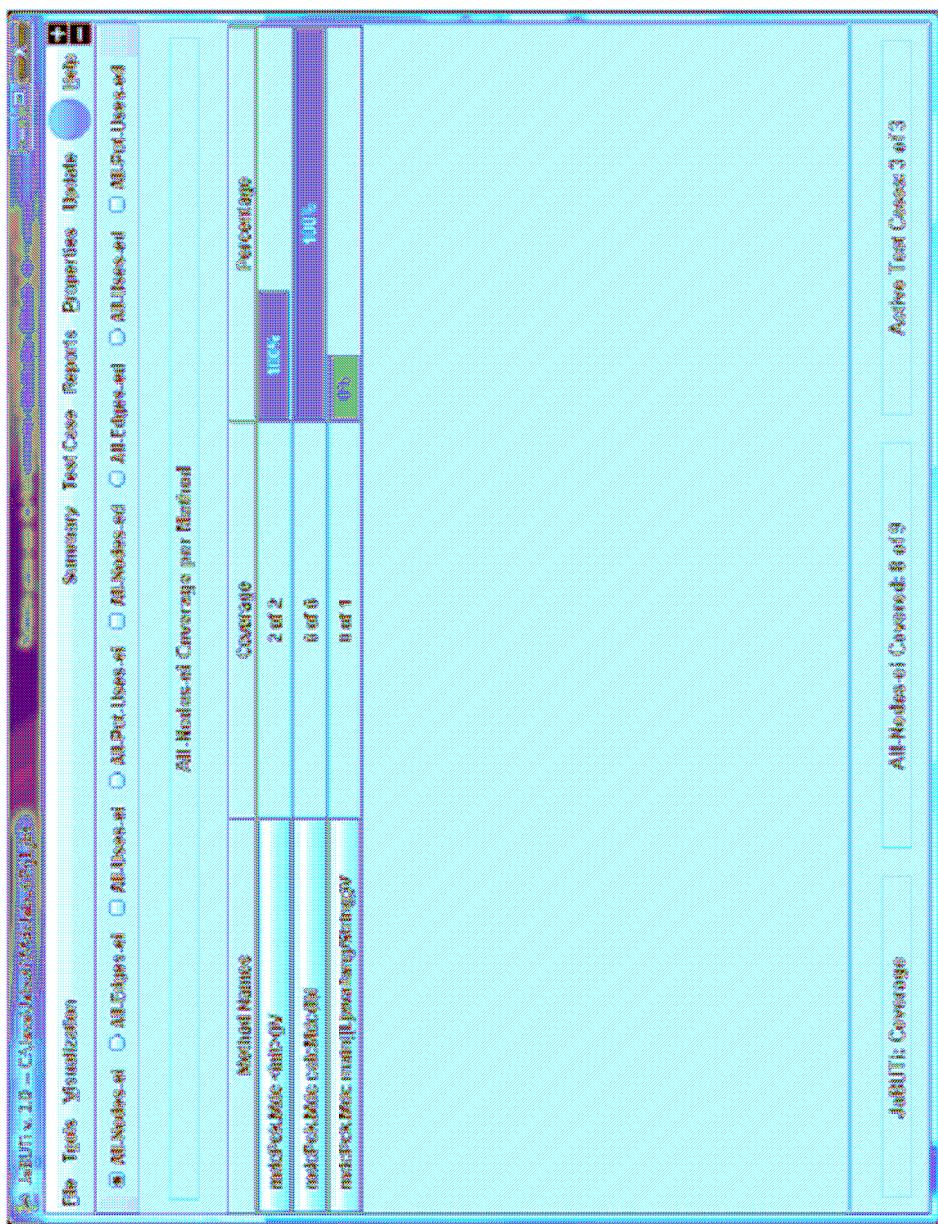
The screenshot shows a JUnit Coverage Tool interface with the following details:

- Toolbar:** Includes icons for New, Open, Save, Print, Copy, Paste, Find, and Exit.
- Navigation Bar:** Summary, Test Case, Reports, Properties, Update, Help.
- Filter Buttons:** All Nodes et, All Edges et, All Uses et, All Edges ed, All Nodes ed, All Uses ed.
- Code Area:** Displays the following Java code with red highlighting:

```
public class TestClass {
    public void testMethod() {
        int x = 10;
        int y = 20;
        if(x > 0) {
            while(y != 0) {
                if(x > y) {
                    x = x - y;
                } else {
                    y = y - x;
                }
            }
        }
        return x;
    }
}
```
- Legend:** Highlighting: All Prioritized
- Bottom Navigation:** JUnit: Coverage Tool, File: index.html, Line: 1 of 25, Coverage: All Nodes et.



Exemplo de resultados

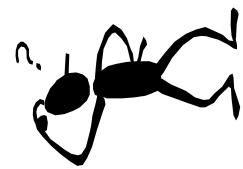


Grafos e Testes



Análise de Cobertura – considerações

- Objetivos
 - Definir os objetivos dos testes
 - Cobrir 100% dos requisitos de teste
 - Avaliar os resultados obtidos
 - Somente 70% dos requisitos de teste foram cobertos



O que fazer?



Análise de Cobertura – considerações

• Objetivos

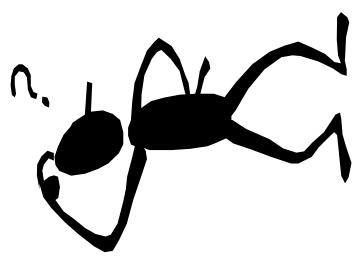
- Definir os objetivos dos testes
 - Cobrir 100% dos requisitos de teste
- Avaliar os resultados obtidos
 - Somente 70% dos requisitos de teste foram cobertos



Projetar mais testes até que o objetivo seja atingido
(grau de cobertura)



Análise de Cobertura – considerações sobre complexeza



É sempre possível ter
100% de cobertura?

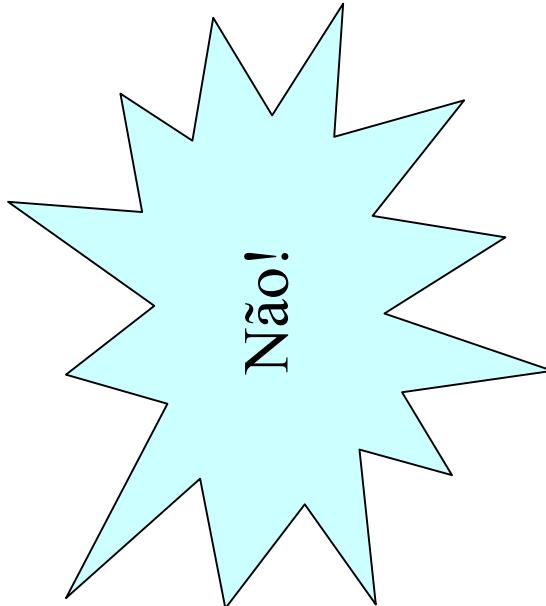


Análise de Cobertura –

considerações sobre Completeza

Natureza da unidade

- Simplicidade, baixa relevância → para que cobrir 100%?
- Há caminhos que não são viáveis:
 - Ex.: instruções inatingíveis
- Há partes difíceis de alcançar sem componentes auxiliares
 - Ex.: tratadores de exceção
- Falta de recursos
 - Pouco tempo disponível
 - Pouco pessoal treinado
 - Poucas ferramentas para auxiliar



```

/*
 * "Ler 3 números inteiros da entrada, imprimir o menor e o maior"
 * Autor: Jacques Sauvé
 */
public class MinMax4 {
    public static void main(String[] args) {
        final int NÚMEROS_A_LER = 3;
        Scanner sc = new Scanner(System.in);
        int mínimo = Integer.MAX_VALUE;
        int máximo = Integer.MIN_VALUE;
        for (int i = 0; i < NÚMEROS_A_LER; i++) {
            System.out.print("Entre com o proximo inteiro: ");
            int num = sc.nextInt();
            if (num < mínimo) {
                mínimo = num;
            }
            if (num > máximo) {
                máximo = num;
            }
        }
        System.out.println("O menor numero eh: " + mínimo);
        System.out.println("O maior numero eh: " + máximo);
    }
}

```

Exercício

- Obtenha o grafo de fluxo de controle para o método main.
- Crie um conjunto de testes que satisfaça ao critério de todas as instruções.
- Crie um conjunto de testes que satisfaça ao critério de todos os ramos.
- O número de casos de teste nos dois casos é o mesmo? Porquê?