Residência em TI
Gerência de Configuração e Teste de Software

Prof. Eiji Adachi

Objetivos

- Como testar?
    - Testes Estruturais (Caixa-branca)

Teste estrutural

Teste Estrutural

- Técnica de teste baseada na estrutura do código fonte do programa
  - Também conhecido como “Caixa branca”, ou “Caixa de vidro”

- Teste estrutural ainda checa a funcionalidade de um programa com respeito à sua especificação
  - Os critérios de avaliação são distintos dos critérios da técnica funcional

Teste Estrutural

- Teste Estrutural e Teste Funcional são complementares
  - Mais uma forma de aumentar a variedade de condições testadas
  - Teste estrutural pode incluir casos de teste que não são visíveis pela interface da função testada
  - Teste funcional é capaz de identificar falhas causadas por funcionalidades ausentes

Fundamentação teórica

- Critérios de teste estrutural são tipicamente baseados na representação de programas em forma de grafos

O que é um grafo?

$$G = (V, E)$$

$$V = \{n_1, n_2, n_3\} \text{ – Conjunto de nós}$$

$$E = \{e_1, e_2, e_3\} \text{ – Conjunto de arestas,}$$

em que $e_k = \{n_i, n_j\} \mid n_i, n_j \backslash in\ V$

Fundamentação teórica

- O grafo do programa é um grafo **dirigido** em que:
  - Os nós correspondem às instruções do programa
  - As arestas representam o fluxo de controle do programa
    - Se n1 e n2 são nós no Grafo do Programa, uma aresta {n1, n2} existe sse. a instrução correspondente a n1 pode ser executada imediatamente após a instrução correspondente a n2

Fundamentação teórica

- O grafo do programa também é chamado de **Grafo de Fluxo de Controle (GFC)**
  - Descreve a sequência em que as instruções são executadas e a forma que o controle de um programa flui durante a execução
  - Possui um único nó de entrada e um único nó de saída

Como desenhar um GFC?

- Numere todas as instruções do código fonte
  - Cada instrução numerada representa um nó no CFG
  - Instruções como declaração de variáveis e definição de tipos podem ser ignoradas

- Se a execução de uma instrução pode resultar na transferência de controle para outra instrução, então adicione uma aresta entre os nós que representam estas instruções

Como desenhar um GFC?

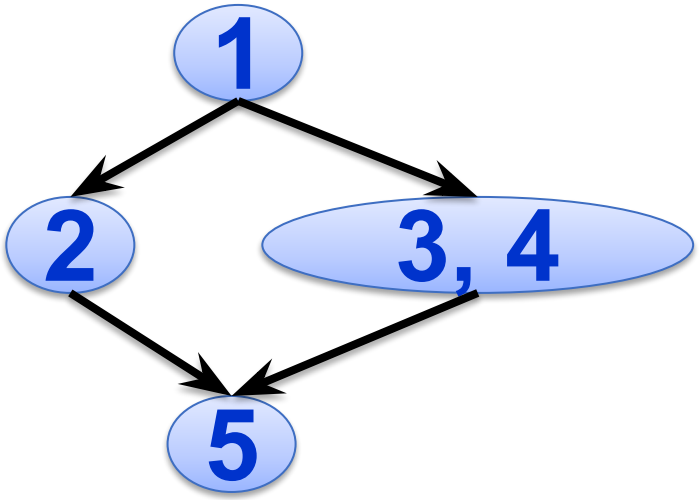
- Sequência de instruções

```
1aux = b;
2b = a;
3a = aux;
```

Como desenhar um GFC?

- Decisão:

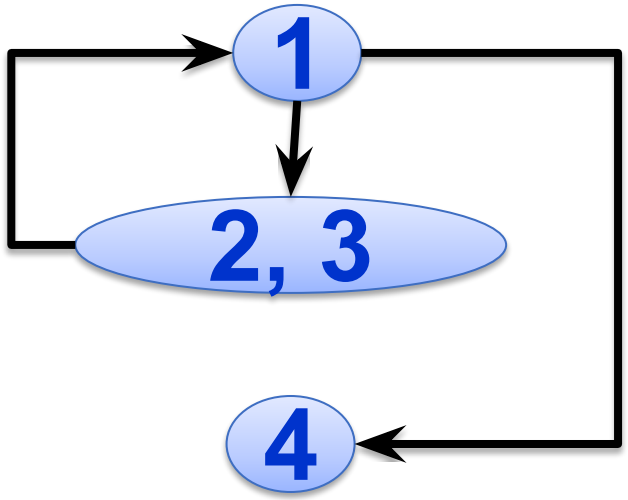
```
1  if(a>b) then{
2     c=3;
   }
   else{
3     c=5;
4     b=2;
   }
5  print c;
```



Como desenhar um GFC?

- Iteração:

```
1 while(a>b){
2   b=b*a;
3   b=b-1;
}
4  c=b+d;
```



Exemplo

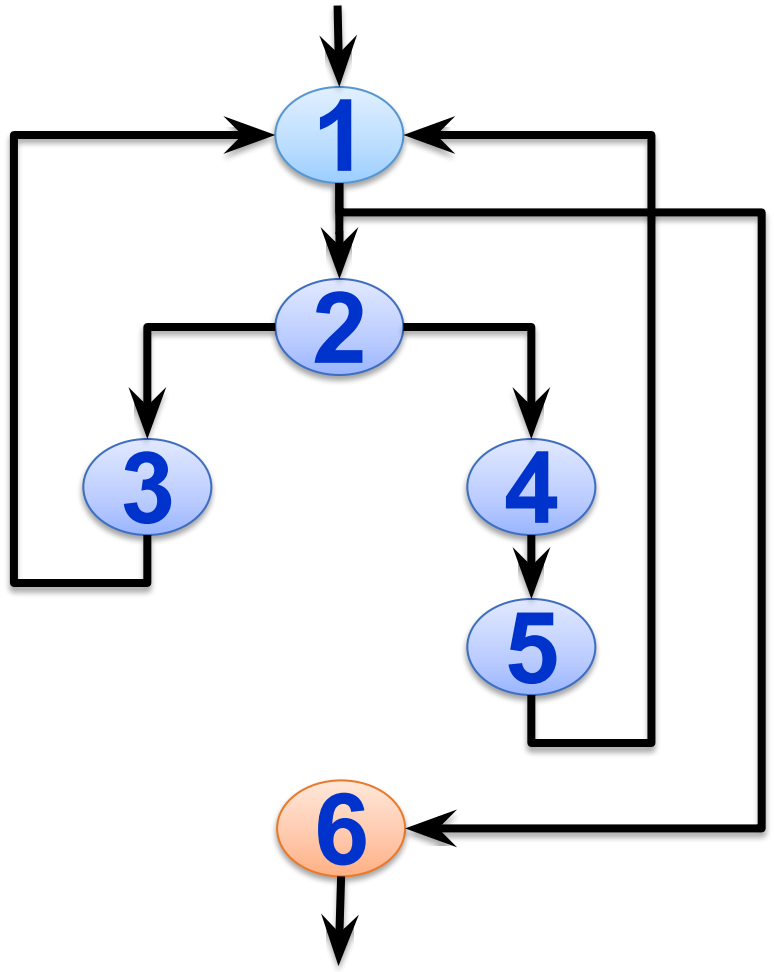
```
int f1(int x,int y){
   while (x != y){
     if (x>y)
       x=x-y;
     else
       y=y-x;
   }
   return x;
}
```

Exemplo

```
int f1(int x,int y){
1  while (x != y){
2    if (x>y)
3      x=x-y;
4    else
5      y=y-x;
   }
6  return x;
}
```

Exemplo

```
int f1(int x,int y){
1  while (x != y){
2    if (x>y)
3      x=x-y;
4    else
5      y=y-x;
   }
6  return x;
}
```



Critérios Baseados em Fluxo de controle

Motivação

- Uma das principais limitações dos critérios de teste funcional é a impossibilidade de verificar até que ponto os casos de teste selecionados são completos e/ou redundantes
  - Quantas classes de equivalência existem? Quantas eu não cobri?

Motivação

- Critérios de teste estrutural superam esta limitação ao definir métricas bem estabelecidas para se medir até que ponto um conjunto de casos de teste é completo

Critérios de Fluxo de Controle

- Utilizam apenas características do fluxo de controle da execução do programa para definir casos de teste

- Principais critérios:
  - Todos Nós (Instruções)
  - Todas Arestas (Decisões)
  - Todas Condições

Cobertura de Nós

- Critério: Cada instrução (ou nó do GFC) deve ser executado ao menos uma vez

- Justificativa: Um defeito numa instrução só pode ser revelado se tal instrução for executada

- Métrica de cobertura:
  # Nós cobertos / # Nós

Cobertura de Nós

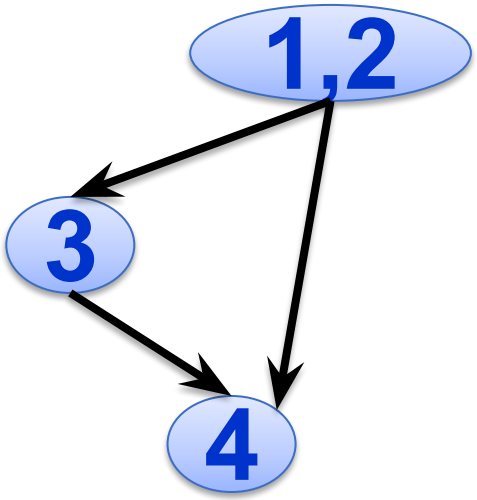
- % Cobertura X Número de casos de teste
  - Deve-se tentar obter um conjunto minimal de casos de teste que atenda o objetivo de % Cobertura
  - É mais fácil diagnosticar uma falha com um conjunto de casos de teste menor, além de ser menos custoso manter tal conjunto ao longo da evolução

- 100% de cobertura de nós não implica em cobertura total de arestas

Cobertura de nós

- Decisão:

```
1 var c;
2 if(a>b) then{
3    c= new Object();
  }
4 c.print();
```



Cobertura de Arestas

- Critério: Cada aresta do GFC deve ser executada ao menos uma vez
- Justificativa: Um defeito numa decisão só pode ser revelado se tal decisão for executada
- Métrica de cobertura:
  # Arestas cobertas / # Arestas

Cobertura de Arestas

- Ao cobrir todas as arestas, automaticamente cobrem-se todos os nós
  - Conjunto de testes que satisfazem o critério de cobertura de arestas para um programa, também satisfazem o critério de cobertura de nós

Cobertura de Arestas

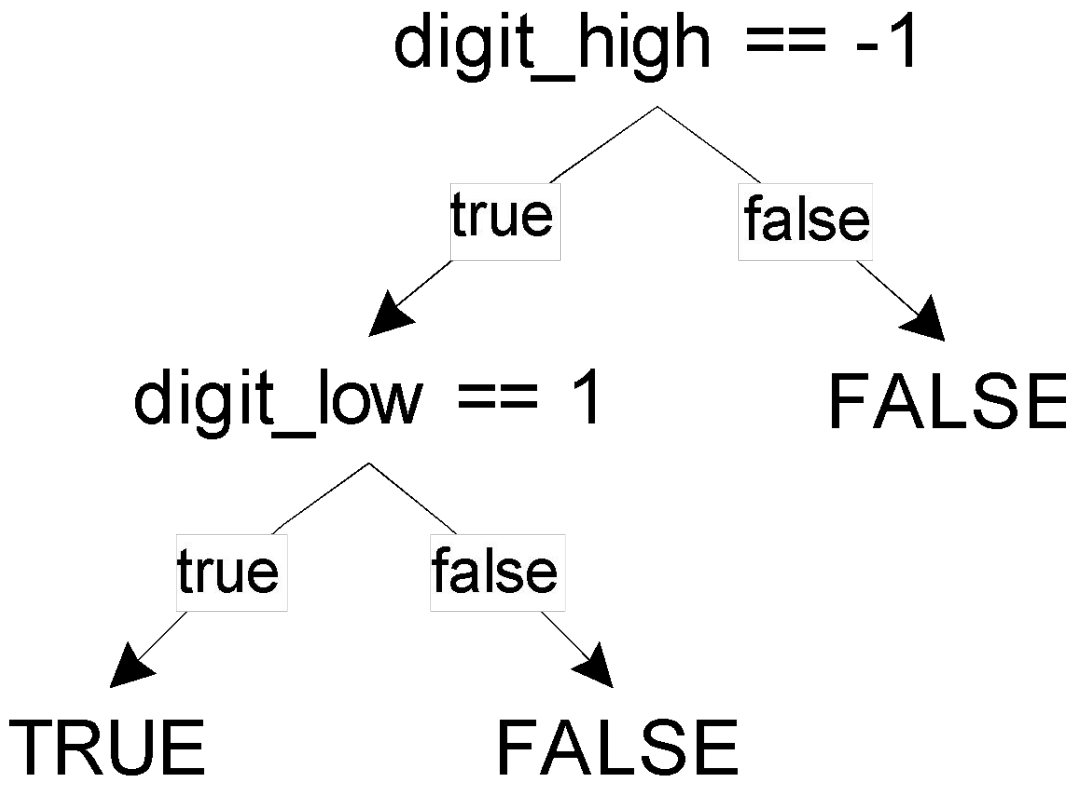
- Cobrir todas as arestas ainda pode deixar “escapar” algumas condições
- Ex.:
  if(digit_high == -1 && digit_low == 1) ... else ...
  - É possível testar as duas arestas da decisão acima apenas variando digit_high
    - O defeito pode estar numa condição não testada

Cobertura de Condições

- Estende cobertura de arestas fazendo uma análise mais detalhada das condições
  - Em especial condições compostas em decisões

Cobertura de Condições

- Critério de cobertura de condições compostas:
  - Cobrir todas possíveis combinações de condições compostas



Comentários e Críticas

- Exercitar todos os elementos de um GFC não garante encontrar todas as falhas
  - Executar uma instrução defeituosa nem sempre irá causar uma falha
    - Execução pode ter sido executada com o programa em um estado que não ocasiona falha

Comentários e Críticas

- Teste estrutural serve como um lembrete do que ainda não foi exercitado
  - Medir métricas de cobertura indicam o que ainda não foi coberto
  - Interpretar elementos não cobertos
    - Podem ser elementos que nunca serão exercitados na prática

Comentários e Críticas

- Teste estrutural é atrativo por prover métricas que indicam progresso (Ex.: % de linhas cobertas)
  - Métricas devem ser usadas com precaução, pois nem sempre indicam efetividade
    - Uma suíte com alta cobertura não é, necessariamente, uma suíte com alta efetivividade
    - Já uma suíte com baixa cobertura é, provavelmente, uma suíte com baixa efetividade

Exercício: Ferramenta de Apoio a Análise de Cobertura

- Instalem e usem no seu IDE alguma ferramenta de análise de cobertura de código
  - Eclipse - https://www.eclemma.org/
  - IntelliJ - https://www.jetbrains.com/help/idea/running-test-with-coverage.html
- Executem a suíte de testes criadas no exercício de Teste Funcional e analisem a cobertura alcançada
  - Alguma linha não coberta? Qual caso não foi coberto pela técnica de Teste Funcional?

Residência em TI
Gerência de Configuração e Teste de Software

Prof. Eiji Adachi