Residência em TI
Gerência de Configuração e Teste de Software

Prof. Eiji Adachi

Objetivos

- Como testar?
  - Técnicas de Teste
    - Testes Funcionais (Caixa-preta)

Técnicas de testes

Teste de Software

```
public String defineType(int l1, int l2, int l3)    {
        if ((l1 > (l2 + l3)) ||
              (l2 > (l1 + l3)) ||
              (l3 > (l1 + l2)))
        {
              return NOT_TRIANGLE;
        }
        else if ((l1 == l2) && (l2 == l3))  {
              return EQUILATERAL;
        }
        else if ((l1 == l2) || (l1 == l3) || (l3 == l2))  {
              return ISOCELES;
        }
        else {
              return SCALENE;
        }
  }
```

Como definir os casos de teste?

Teste de Software

```
public String defineType(int I1, int I2, int I3)    {
        if ((I1 > (I2 + I3)) ||
              (I2 > (I1 + I3)) ||
              (I3 > (I1 + I2)))
        {
              return NOT_TRIANGLE;
        }
        else if ((I1 == I2) && (I2 == I3))  {
              return EQUILATERAL;
        }
        else if ((I1 == I2) || (I1 == I3) || (I3 == I2))  {
              return ISOCELES;
        }
        else {
              return SCALENE;
        }
  }
```

Como definir os casos de teste?

1) Definir casos de testes escolhendo as entradas aleatoriamente

Teste de Software

```
public String defineType(int l1, int l2, int l3)    {
        if ((l1 > (l2 + l3)) ||
              (l2 > (l1 + l3)) ||
              (l3 > (l1 + l2)))
        {
              return NOT_TRIANGLE;
        }
        else if ((l1 == l2) && (l2 == l3))  {
              return EQUILATERAL;
        }
        else if ((l1 == l2) || (l1 == l3) || (l3 == l2)) {
              return ISOCELES;
        }
        else{
              return SCALENE;
        }
  }
```

Como definir os casos de teste?

1) Definir casos de testes escolhendo as entradas aleatoriamente

2) Definir um caso de teste para cada possível combinação de entradas

Teste de Software

```
public String defineType(int l1, int l2, int l3)    {
        if ((l1 > (l2 + l3)) ||
                (l2 > (l1 + l3)) ||
                (l3 > (l1 + l2)))
        {
                return NOT_TRIANGLE;
        }
        else if ((l1 == l2) && (l2 == l3))  {
                return EQUILATERAL;
        }
        else if ((l1 == l2) || (l1 == l3) || (l3 == l2)) {
                return ISOCELES;
        }
        else{
                return SCALENE;
        }
}
```

Inviável querer testar todas possíveis combinações!

Número de possíveis combinações das entradas:

(assumindo *int* com 4 bytes)

$$2^{32} * 2^{32} * 2^{32}$$

$$= 2^{96}$$

Teste de Software

- Teste de Software consiste em verificar dinamicamente que um programa provê comportamentos esperados em
um conjunto finito de casos de teste
que são adequadamente selecionados de um domínio de execução tipicamente infinito [SWEBOK]

Teste de Software

- Uma das etapas fundamentais do processo de testes é o projeto e a criação do conjunto de casos de testes
  - Qual subconjunto de casos de teste escolher?
  - Qual subconjunto tem maior probabilidade de realçar falhas?

Técnica de Teste

- Sistematização usada para projetar casos de teste
- Objetivo é:
  - Limitar o número de casos de testes a uma quantidade viável de ser executada
  - Selecionar casos de testes que exponham o objeto de teste a condições diversas
    - “Testar algo diferente é mais valioso do que testar algo igual ao que já foi testado”

Técnica de Teste

- Técnicas de teste diferenciam-se em quais fontes de informações são usadas para projetar os casos de teste
  - Requisitos e especificações técnicas
  - Código fonte do programa
  - Experiência da equipe de testes
  - Histórico ou padrões de falhas recorrentes

Técnica de Teste

- Principais técnicas:
  - Técnica Funcional (caixa-preta)
    - Baseia-se nos requisitos e especificações técnicas do objeto de teste
  - Técnica Estrutural (caixa-branca)
    - Baseia-se no código fonte do objeto de teste

- São técnicas complementares
  - Tipicamente, aplica-se a Técnica Funcional e então complementa-se com a Técnica Estrutural

Técnica de Teste

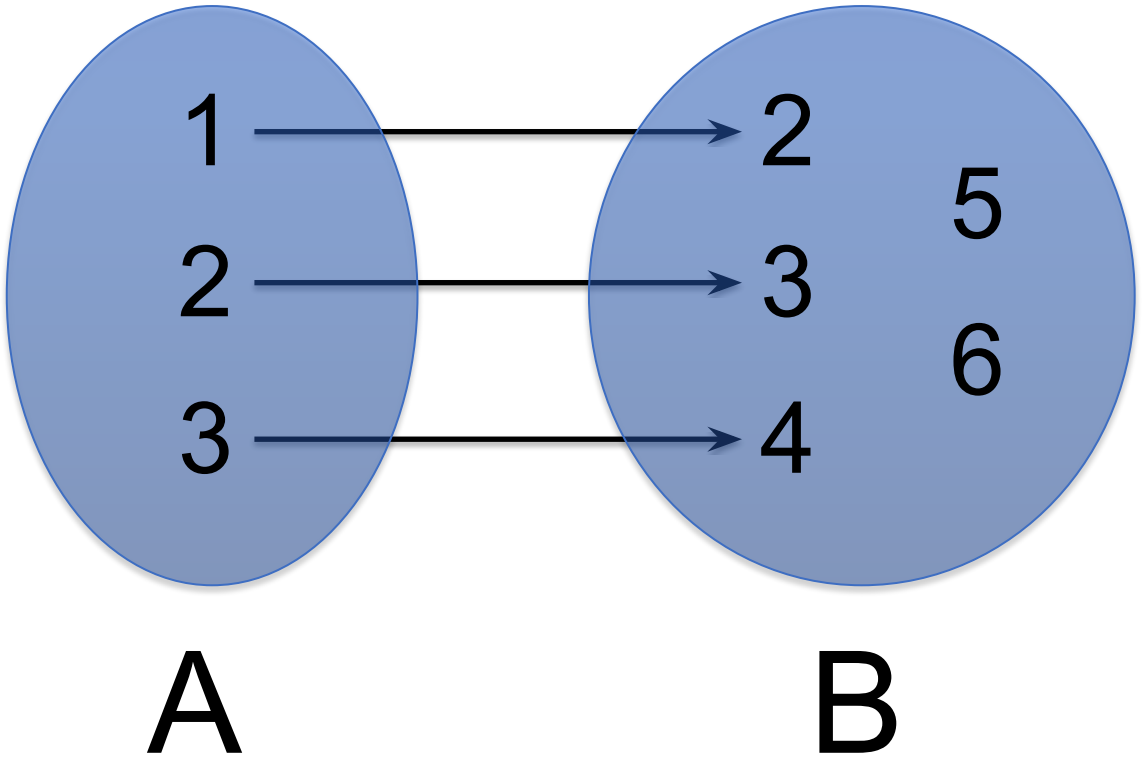
- Técnica Funcional (caixa-preta)
  - Particionamento em classes de equivalência
  - Análise de valor limite

- Técnica Estrutural (caixa-branca)
  - Cobertura de nós
  - Cobertura de arestas
  - Cobertura de condições

Testes funcionais

Função

$$f(x) = x + 1$$



- Domínio?

- Contra-domínio?

- Imagem?

Teste Funcional

- Programas são considerados funções
  - Ou “caixas pretas”

- Técnica funcional usam conhecimento sobre a natureza funcional de um programa para identificar casos de teste
  - Requer apenas conhecimento sobre requisitos e especificação do programa
  - Não requer conhecimento sobre como o programa é implementado
    - Permite a criação dos testes antes da implementação dos objetos de teste

Processo de Teste Funcional

- Passos básicos:
  - Analisar os requisitos do programa
  - Identificar condições de teste
  - Escolher as entradas para testar o programa
  - Determinar as saídas esperadas para as entradas escolhidas
  - Construir casos de teste
  - Executar conjunto de testes
  - Gerar relatório para avaliar o resultado dos testes

Vantagens e Desvantagens

Vantagens

- É compatível com todos os níveis de teste
- Independe do paradigma de programação
- Capaz de determinar falhas do tipo “funcionalidade ausente”

Desvantagens

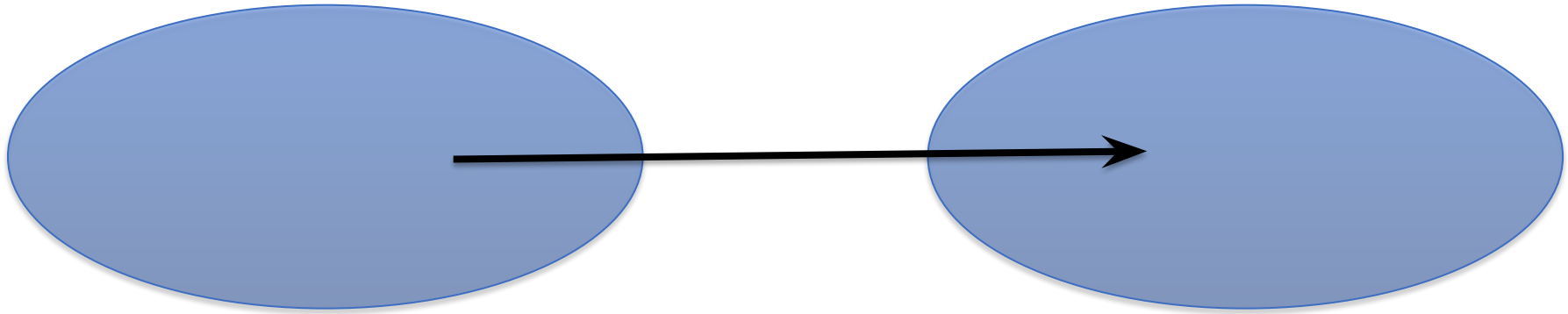
- Depende de uma boa especificação
  - Nem sempre existe
- Não garante que partes críticas do programa sejam executadas

Particionamento em Classes de equivalência

Motivação

- Domínio das entradas de um programa é geralmente muito grande para se testar exaustivamente

```
String defTipoTriangulo(int, int, int);
```



Classes de Equivalência

- Critério utilizado para reduzir a redundância e o número de casos de teste

- Objetivo:
  - Dividir o domínio das entradas em um conjunto finito de sub-domínios chamados de classe de equivalência
    - Todos os elementos de um sub-domínio são equivalentes do ponto de vista do teste de software
      - “Se uma entrada num sub-domínio revela um defeito, qualquer entrada no mesmo sub-domínio também revelaria”

Passos de Aplicação

- Identificar requisitos especificados para as entradas
- Definir classes de equivalência a partir dos requisitos identificados
- Para cada classe de equivalência identificada, criar um caso de teste

Programa Identificador

- O programa deve verificar se uma String é ou não um identificador válido. Um identificador válido deve começar com uma letra e conter apenas letras e dígitos. Além disso, deve conter no mínimo 2 e no máximo 6 caracteres de comprimento.

- Exemplo:
- ab12 (válido)
- ab! (inválido) – a (inválido) – abcde12345 (inválido)

Programa Identificador

- Extraindo informações dos requisitos:
  - Requisitos de um identificador válido:
    - Começa com uma letra: Sim
    - Tamanho do identificador: 2 <= T <= 6
    - Só contém letras e dígitos: Sim
  - Requisitos de um identificador inválido:
    - Começa com uma letra: Não
    - Tamanho do identificador: T < 2 || T > 6
    - Só contém letras e dígitos: Não
      - Contém ao menos um caractere que não é letra nem dígito

Programa Identificador

- Classes de Equivalência

Tamanho da Entrada	2 <= T <= 6	(1)
	T < 2	(2)
	T > 6	(3)

Programa Identificador

- Classes de Equivalência

Primeiro Caractere é Letra	Sim	(4)
	Não	(5)

Programa Identificador

- Classes de Equivalência

Só contém letras e dígitos	Sim	(6)
	Não	(7)

Programa Identificador

- Classes de Equivalência

Tamanho da Entrada	2 <= T <= 6	(1)
	T < 2	(2)
	T > 6	(3)
Primeiro Caractere é Letra	Sim	(4)
	Não	(5)
Só contém letras e dígitos	Sim	(6)
	Não	(7)

Programa Identificador

- Casos de Teste

Entrada	Saída Esperada	Classe Coberta
“**Ab12**”	Válido	1, 4, 6
“a”	Inválido	2, 4, 6
“**abcd1234**”	Inválido	3, 4, 6
“1234”	Inválido	1, 5, 6
“abc@”	Inválido	1, 4, 7
"**!**"	Inválido	2, 5, 7 (Não fazer assim)

Exercitar uma situação "excepcional" por vez

Identificação das Classes

- Se uma entrada possui uma restrição que é um intervalo [a, b] então:
  - Um classe válida dentro do intervalo
  - Duas classes inválidas fora do intervalo

- Se uma entrada possui uma restrição que é um conjunto de valores permitidos {a, b, c}, então:
  - Uma classe válida para cada valor permitido
  - Uma classe inválida para valor não permitido

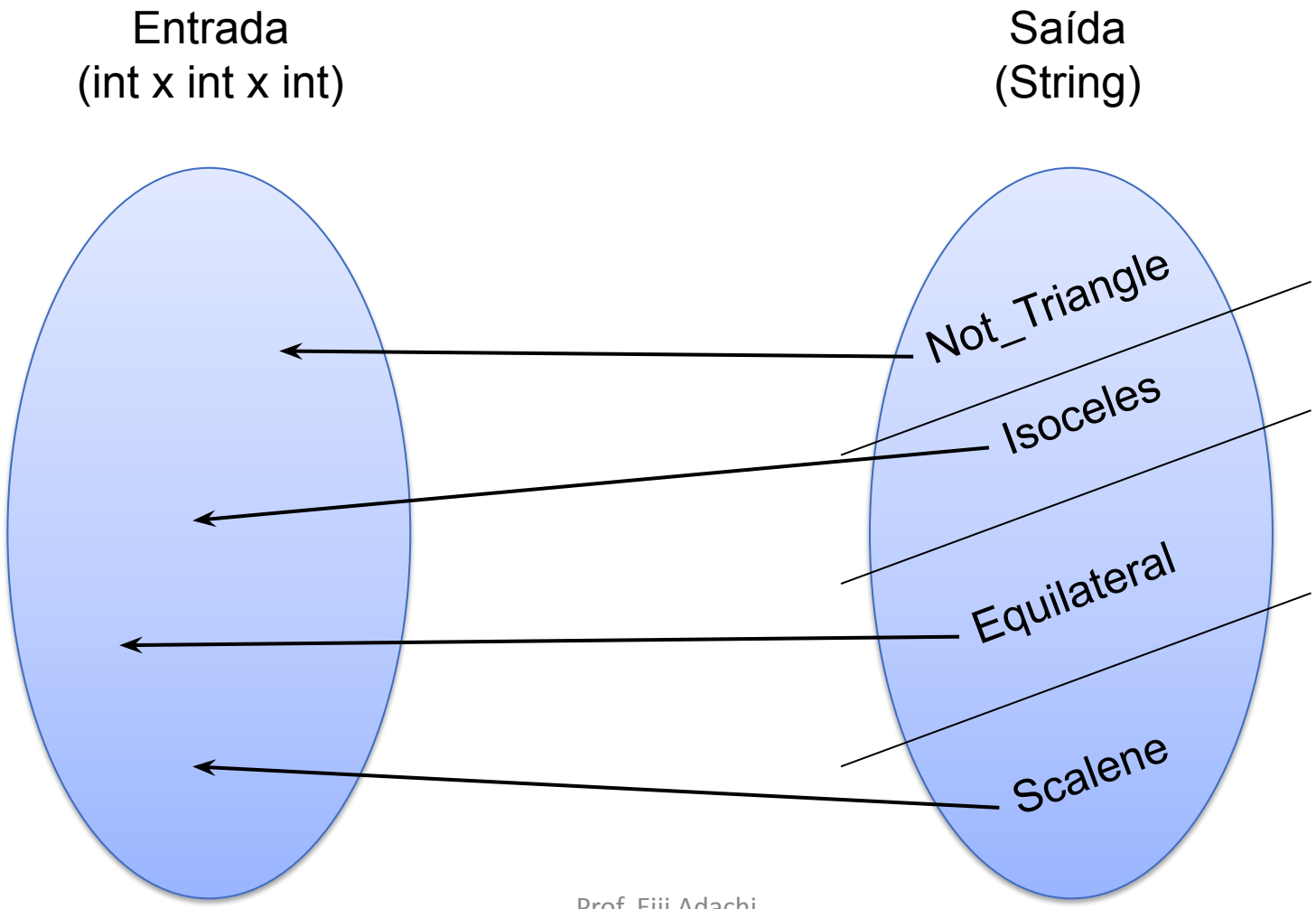
Exercício

Quais classes de equivalência para o programa do tipo de triângulo?

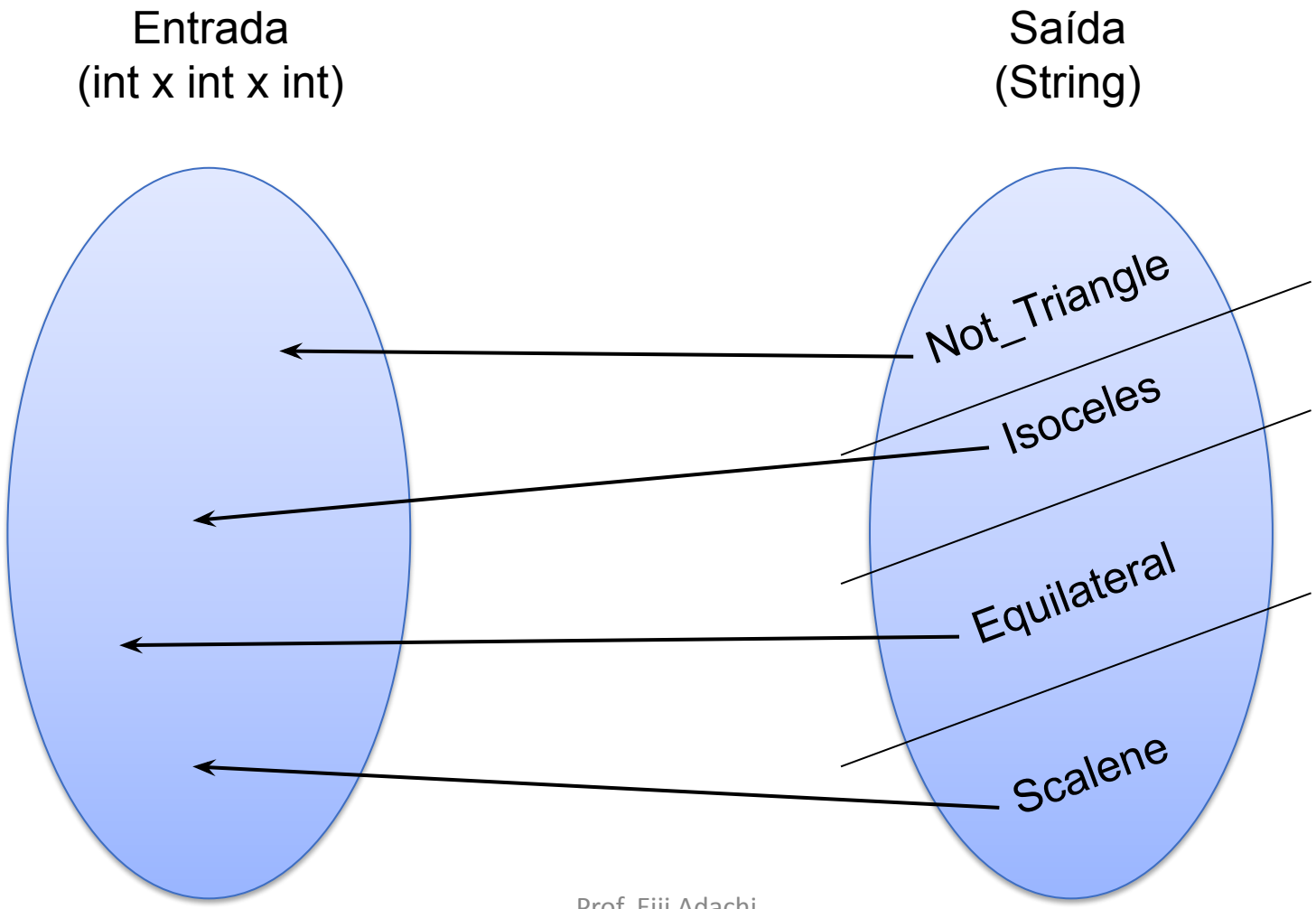
Exercício

- Nem sempre é fácil encontrar as classes de equivalência analisando apenas as entradas
- As vezes ajuda analisar e particionar as saídas em classes para saber quais entradas escolher
  - Para cada classe identificada na saída, escolher uma combinação de entrada que leve a esta saída

Tipo de Triângulo



Tipo de Triângulo



Análise de Valor limite

Motivação

- Muitos defeitos ocorrem nas fronteiras entre as classes de equivalência

```
for(int i = 0; i < l1; i++) ?
```

ou

```
for(int i = 0; i <= l1; i++) ?
```

Motivação

- Complementar o Critério de Particionamento em Classes de Equivalência:
  - Critério de Classes de Equivalência requer que qualquer elemento de uma classe seja selecionado
  - Critério do Valor Limite requer que elementos nas fronteiras das classes de equivalência sejam selecionados

Valor Limite

- São selecionados valores:
  - Nos limites Min e Max
  - Acima e Abaixo de Min e Max
  - Dentro do Intervalo Min-Max

Programa *Identificador*

- Classes de Equivalência

Tamanho da Entrada	2 <= T <= 6	(1)
	T < 2	(2)
	T > 6	(3)
Primeiro Caractere é Letra	Sim	(4)
	Não	(5)
Só contém letras e dígitos	Sim	(6)
	Não	(7)

Programa *Identificador*

- Casos de Teste

Entrada	Saída Esperada	Classe Coberta
“Ab12”	Válido	1(Normal), 4, 6
“A1”	Válido	1(Min), 4, 6
“ABC123”	Válido	1(Max), 4, 6
“A12”	Válido	1(Min+), 4, 6
“ABC12”	Válido	1(Max-), 4, 6
“A”	Inválido	2(Min-), 4, 6
“ABC1234”	Inválido	3(Max+), 4, 6

Análise de Valor Limite

- *Guideline*:
  - Se uma condição sobre uma entrada /saída impôr um limite, defina três casos de teste:
    - Um com valor logo abaixo do limite
    - Outro com valor logo acima do limite
    - Outro com valor exatamente sobre o valor limite

Exercício: Aprovação na UFRN

Sigla	Significado	Situação
REMF	Reprovado por Média e Faltas	Reprovado porque a média foi inferior a três e também por não atender os critérios de assiduidade.
REPF	Reprovado por Faltas	Reprovado por não atender apenas os critérios de assiduidade.
REP	Reprovado por Média	Reprovado apenas porque a média foi inferior a três.
REC	Em Recuperação	Aluno que fará a reposição.
APR	Aprovado por Média	Aluno aprovado com média maior ou igual a sete.
APRN	Aprovado por Notas	Aluno com média entre cinco e sete e que não tirou nenhuma nota inferior a três.

Exercício: Aprovação na UFRN

- Art. 105. É considerado aprovado, quanto à avaliação de aprendizagem, o estudante que satisfaz um dos seguintes critérios:
  - I – tem média parcial igual ou superior a 7,0 (sete); ou
  - II – tem média parcial igual ou superior a 5,0 (cinco), com rendimento acadêmico igual ou superior a 3,0 (três) em todas as unidades.
-
- Art. 106. O estudante que não atinge os critérios de aprovação definidos no artigo 105 tem direito à realização de uma avaliação de reposição se todas as seguintes condições forem atendidas:
  - I – O critério de aprovação por assiduidade é satisfeito; e
  - II – O estudante tem média parcial igual ou superior a 3,0 (três).
-
- Art. 113. Para ser aprovado em uma disciplina ou módulo presencial, o estudante deve comparecer a aulas que totalizem 75% (setenta e cinco por cento) ou mais da carga horária do componente curricular ou a 75% (setenta e cinco por cento) ou mais do total de aulas ministradas, o que for menor.

Exercício: Aprovação na UFRN

- Identifique as classes de equivalência
- Identifique os valores limites
- Crie casos de teste que cubram todas classes e valores limites identificados
- Automatize os casos de testes usando os conceitos do JUnit
  - Testes parametrizáveis
  - Testes de exceção

Testes baseados em Tabelas de decisão

Tabelas de Decisão – Definição

- Tabelas de Decisão são modelos para representação de regras que relacionam condições a determinadas ações

Tabelas de Decisão – Uso

- Podem ser usadas para:
  - Especificar relações lógicas
    - Requisitos de software
  - Auxiliar a criação de casos de teste

Tabelas de Decisão – Estrutura

- Estrutura dividida em 3 partes principais:
  - Condições
    - Cada condição corresponde a uma variável, relação ou predicado
  - Ações
    - Cada ação é um procedimento ou operação a realizar
  - Regras
    - Cada regra é uma combinação de condições que dispara uma ou mais ações

	Regra 1	Regra 2	Regra 3	Regra 4
Condição 1	V	V	V	F
Condição 2	V	V	F	F
Condição 3	V	F	F	V
Ação 1	X		X	X
Ação 2		X		

Tabelas de Decisão – Exemplo

- Tabela de decisão para auxiliar resolução do problema: “Impressora não imprime”

	Regra 1	Regra 2	Regra 3	Regra 4
Luz vermelha piscando	V	V	V	F
Impressora reconhecida	V	V	F	-
Impressora *online*	V	F	V	-
Verificar cabo de energia				X
Verificar cabo USB		X		
Verificar instalação do *driver*			X	
Verificar cartucho de tinta	X			
Verificar bandeja de papel	X			

Tabelas de Decisão – Estrutura

- Também é possível definir Tabelas de Decisão com condições que não são estritamente binárias
  - Boa prática: Deixar claro quais os possíveis valores que cada condição pode assumir

	Regra 1	Regra 2	Regra 3	Regra 4
Condição 1 - {V, F}	V	V	V	F
Condição 2 - { < 0, =0, >0}	<0	=0	>0	=0
Condição 3 - {A, B, C}	A	A	A	B
Ação 1	X		X	X
Ação 2		X		

Tabelas de Decisão – Exemplo

- Tabela de decisão para auxiliar a criação de casos de teste para o problema "Senha Forte-Fraca"

	Regra 1	Regra 2	Regra 3	Regra 4
Tamanho da cadeia {<8, =8, >8}	= 8	>8	< 8	=8
Possui caractere numérico? {V, F}	V	V	V	V
Possui caractere não-alfanumérico? {V, F}	V	V	V	V
Possui caractere em caixa alta? {V, F}	V	V	V	F
Retorne Senha Forte	X	X		
Retorne Senha Fraca			X	X

Tabelas de Decisão – Definição

- Tabela de Decisão Limitada
  - Todas as condições são binárias

- Tabela de Decisão Estendidas
  - Condições podem assumir diversos valores

Como criar Tabela de Decisão

- Um possível método **exaustivo**:
  1. Identifique todas as condições e os respectivos valores possíveis
  2. Calcule o número de possíveis combinações
  3. Crie uma coluna para cada combinação
  4. Preencha as colunas com todas as possíveis combinações das condições
  5. Simplifique algumas combinações
  6. Identifique quais ações são disparadas por cada combinação

Como criar Tabela de Decisão

- Exemplo: Programa Tipo de Triângulo

  1. Identifique todas as condições e os respectivos valores possíveis
     - a < b + c? - {T, F}
     - b < a + c? - {T, F}
     - c < a + b? - {T, F}
     - a = b? - {T, F}
     - a = c? - {T, F}
     - b = c? - {T, F}

Como criar Tabela de Decisão

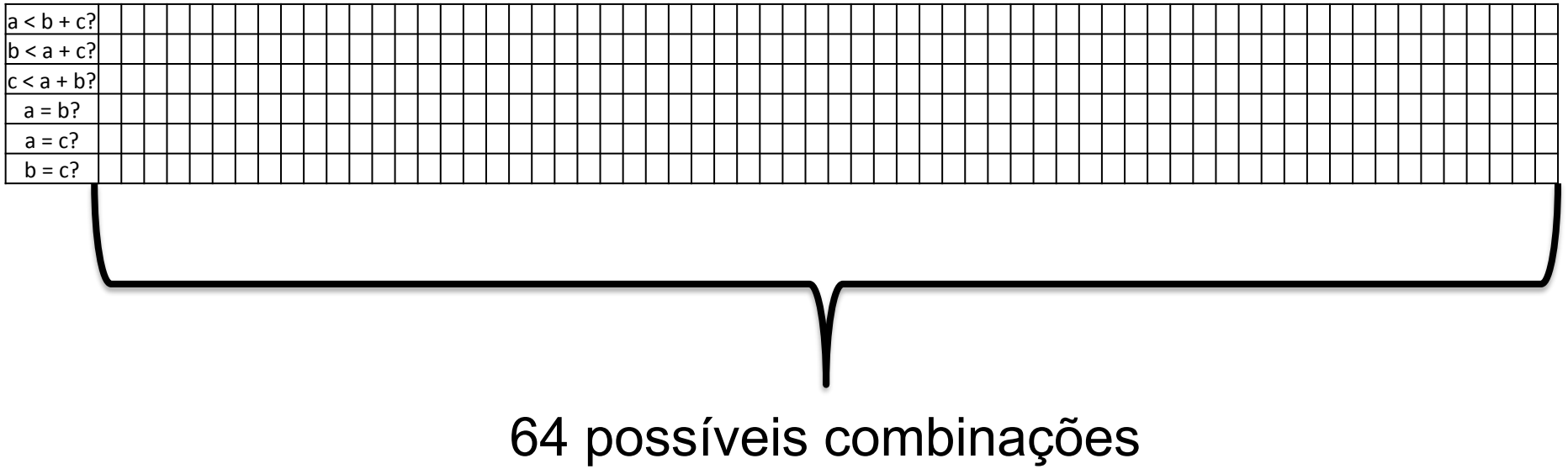
- Exemplo: Programa Tipo de Triângulo

  2. Calcule o número de possíveis combinações
     - 6 condições binárias = 2^6 combinações = 64

Como criar Tabela de Decisão

- Exemplo: Programa Tipo de Triângulo

  3. Crie uma coluna para cada combinação



Como criar Tabela de Decisão

- Exemplo: Programa Tipo de Triângulo

  4. Preencha as colunas com todas as possíveis combinações das condições

a < b + c?	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
b < a + c?	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
c < a + b?	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F
a = b?	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F
a = c?	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F
b = c?	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F

a < b + c?	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
b < a + c?	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
c < a + b?	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F
a = b?	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F
a = c?	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F
b = c?	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F

Como criar Tabela de Decisão

- Exemplo: Programa Tipo de Triângulo

  5. Simplifique algumas combinações

     - Identifique condições que são suficientes para disparar uma determinada ação

a < b + c?	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
b < a + c?	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
c < a + b?	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F
a = b?	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F
a = c?	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F
b = c?	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F

a < b + c?	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
b < a + c?	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
c < a + b?	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F
a = b?	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F
a = c?	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F
b = c?	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F

Como criar Tabela de Decisão

- Exemplo: Programa Tipo de Triângulo

  5. Simplifique algumas combinações

     - Identifique condições que são suficientes para disparar uma determinada ação

a < b + c?	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
b < a + c?	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
c < a + b?	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F
a = b?	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F
a = c?	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F
b = c?	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F

a < b + c?	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
b < a + c?	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F
c < a + b?	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F
a = b?	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F
a = c?	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F
b = c?	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F

a < b + c ? False □ Não é Triângulo

Como criar Tabela de Decisão

- Exemplo: Programa Tipo de Triângulo

5. Simplifique algumas combinações

- Identifique condições que são suficientes para disparar uma determinada ação

Entradas marcadas com “-” são chamadas de “Don´t care”

a < b + c?	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
b < a + c?	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	F	-
c < a + b?	T	T	T	T	T	T	T	F	F	F	F	F	F	F	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	-
a = b?	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	-
a = c?	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	-
b = c?	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	-

Não importam, pois a primeira condição já é suficiente para disparar a ação “Não é Triângulo”

a < b + c ? False □ Não é Triângulo

Como criar Tabela de Decisão

- Exemplo: Programa Tipo de Triângulo

5. Simplifique algumas combinações

- Identifique condições que são suficientes para disparar uma determinada ação

a < b + c?	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F
b < a + c?	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	F	F	F	F	F	F	-
c < a + b?	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	T	T	T	T	T	T	T	F	F	F	F	F	F	F	-
a = b?	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	-
a = c?	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	-
b = c?	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	-

b < a + c ? False □ Não é Triângulo

Como criar Tabela de Decisão

- Exemplo: Programa Tipo de Triângulo

5. Simplifique algumas combinações

- Identifique condições que são suficientes para disparar uma determinada ação

a < b + c?	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F
b < a + c?	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	-
c < a + b?	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	-	-
a = b?	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	-	-
a = c?	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	-	-
b = c?	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	-	-

c < a + b ? False □ Não é Triângulo

Como criar Tabela de Decisão

- Exemplo: Programa Tipo de Triângulo

5. Simplifique algumas combinações

- Identifique condições que são suficientes para disparar uma determinada ação

a < b + c?	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F
b < a + c?	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	F	-
c < a + b?	T	T	T	T	T	T	T	T	F	F	F	F	F	F	F	F	-	-
a = b?	T	T	T	T	F	F	F	F	T	T	T	T	F	F	F	F	-	-
a = c?	T	T	F	F	T	T	F	F	T	T	F	F	T	T	F	F	-	-
b = c?	T	F	T	F	T	F	T	F	T	F	T	F	T	F	T	F	-	-

c < a + c ? False □ Não é Triângulo

Como criar Tabela de Decisão

- Exemplo: Programa Tipo de Triângulo

  5. Simplifique algumas combinações

     - Identifique condições que são suficientes para disparar uma determinada ação

a < b + c?	T	T	T	T	T	T	T	T	T	T	F
b < a + c?	T	T	T	T	T	T	T	T	T	F	-
c < a + b?	T	T	T	T	T	T	T	T	F	-	-
a = b?	T	T	T	T	F	F	F	F	-	-	-
a = c?	T	T	F	F	T	T	F	F	-	-	-
b = c?	T	F	T	F	T	F	T	F	-	-	-

Como criar Tabela de Decisão

- Exemplo: Programa Tipo de Triângulo

  5. Simplifique algumas combinações

     - Identifique condições que são suficientes para disparar uma determinada ação

a < b + c?	T	T	T	T	T	T	T	T	T	T	F
b < a + c?	T	T	T	T	T	T	T	T	T	F	-
c < a + b?	T	T	T	T	T	T	T	T	F	-	-
a = b?	T	T	T	T	F	F	F	F	-	-	-
a = c?	T	T	F	F	T	T	F	F	-	-	-
b = c?	T	F	T	F	T	F	T	F	-	-	-

11 possíveis combinações

Como criar Tabela de Decisão

- Exemplo: Programa Tipo de Triângulo

  6. Identifique quais ações são disparadas por cada combinação

a < b + c?	T	T	T	T	T	T	T	T	T	T	F
b < a + c?	T	T	T	T	T	T	T	T	T	F	-
c < a + b?	T	T	T	T	T	T	T	T	F	-	-
a = b?	T	T	T	T	F	F	F	F	-	-	-
a = c?	T	T	F	F	T	T	F	F	-	-	-
b = c?	T	F	T	F	T	F	T	F	-	-	-

Como criar Tabela de Decisão

Condição											
a < b + c?	T	T	T	T	T	T	T	T	T	T	F
b < a + c?	T	T	T	T	T	T	T	T	T	F	-
c < a + b?	T	T	T	T	T	T	T	T	F	-	-
a = b?	T	T	T	T	F	F	F	F	-	-	-
a = c?	T	T	F	F	T	T	F	F	-	-	-
b = c?	T	F	T	F	T	F	T	F	-	-	-
Ação											
Não é Triângulo											
Escaleno											
Isóceles											
Equilátero											

Como criar Tabela de Decisão

Condição											
a < b + c?	T	T	T	T	T	T	T	T	T	T	F
b < a + c?	T	T	T	T	T	T	T	T	T	F	-
c < a + b?	T	T	T	T	T	T	T	T	F	-	-
a = b?	T	T	T	T	F	F	F	F	-	-	-
a = c?	T	T	F	F	T	T	F	F	-	-	-
b = c?	T	F	T	F	T	F	T	F	-	-	-
Ação											
Não é Triângulo											
Escaleno											
Isóceles											
Equilátero	x										

Como criar Tabela de Decisão

Algumas combinações são impossíveis de ocorrer no domínio da aplicação

Condição											
a < b + c?	T	T	T	T	T	T	T	T	T	T	F
b < a + c?	T	T	T	T	T	T	T	T	T	F	-
c < a + b?	T	T	T	T	T	T	T	T	F	-	-
a = b?	T	T	T	T	F	F	F	F	-	-	-
a = c?	T	T	F	F	T	T	F	F	-	-	-
b = c?	T	F	T	F	T	F	T	F	-	-	-
Ação											
Não é Triângulo											
Escaleno											
Isóceles											
Equilátero	x										

Como criar Tabela de Decisão

Por enquanto, vamos manter registro das combinações impossíveis

Condição											
a < b + c?	T	T	T	T	T	T	T	T	T	T	F
b < a + c?	T	T	T	T	T	T	T	T	T	F	-
c < a + b?	T	T	T	T	T	T	T	T	F	-	-
a = b?	T	T	T	T	F	F	F	F	-	-	-
a = c?	T	T	F	F	T	T	F	F	-	-	-
b = c?	T	F	T	F	T	F	T	F	-	-	-
Ação											
Não é Triângulo											
Escaleno											
Isóceles											
Equilátero	x										
Impossível		x									

Como criar Tabela de Decisão

Completem o restante da tabela

Condição											
a < b + c?	T	T	T	T	T	T	T	T	T	T	F
b < a + c?	T	T	T	T	T	T	T	T	T	F	-
c < a + b?	T	T	T	T	T	T	T	T	F	-	-
a = b?	T	T	T	T	F	F	F	F	-	-	-
a = c?	T	T	F	F	T	T	F	F	-	-	-
b = c?	T	F	T	F	T	F	T	F	-	-	-
Ação											
Não é Triângulo											
Escaleno											
Isóceles											
Equilátero	x										
Impossível		x									

Como criar Tabela de Decisão

Condição											
a < b + c?	T	T	T	T	T	T	T	T	T	T	F
b < a + c?	T	T	T	T	T	T	T	T	T	F	-
c < a + b?	T	T	T	T	T	T	T	T	F	-	-
a = b?	T	T	T	T	F	F	F	F	-	-	-
a = c?	T	T	F	F	T	T	F	F	-	-	-
b = c?	T	F	T	F	T	F	T	F	-	-	-
Ação											
Não é Triângulo									X	X	X
Escaleno								X			
Isóceles				X		X	X				
Equilátero	X										
Impossível		X	X		X						

Como criar Tabela de Decisão

Para simplificar, removemos as combinações impossíveis

Condição											
a < b + c?	T	T	T	T	T	T	T	T	T	T	F
b < a + c?	T	T	T	T	T	T	T	T	T	F	-
c < a + b?	T	T	T	T	T	T	T	T	F	-	-
a = b?	T	T	T	T	F	F	F	F	-	-	-
a = c?	T	T	F	F	T	T	F	F	-	-	-
b = c?	T	F	T	F	T	F	T	F	-	-	-
Ação											
Não é Triângulo									X	X	X
Escaleno								X			
Isóceles				X		X	X				
Equilátero	X										
Impossível		X	X		X						

Como criar Tabela de Decisão

Para simplificar, removemos as combinações impossíveis

Condição								
a < b + c?	T	T	T	T	T	T	T	F
b < a + c?	T	T	T	T	T	T	F	-
c < a + b?	T	T	T	T	T	F	-	-
a = b?	T	T	F	F	F	-	-	-
a = c?	T	F	T	F	F	-	-	-
b = c?	T	F	F	T	F	-	-	-
Ação								
Não é Triângulo						X	X	X
Escaleno					X			
Isóceles		X	X	X				
Equilátero	X							

Como criar Casos de Teste

- Para criar Casos de Teste com base em Tabelas de Decisão, interprete:
  - As condições da Tabela de Decisão como as condições sobre as entradas do programa
    - Tipicamente, condições vão estar associadas a classes de equivalência
  - As ações da Tabela de Decisão como as saídas do programa
    - Tipicamente, ações vão estar associadas aos resultados esperados (valor retornado ou exceção lançada)
  - Interprete cada regra como um caso de teste

Como criar Casos de Teste

- Tabela de Decisão para o programa Tipo de Triângulo

Condição								
a < b + c?	T	T	T	T	T	T	T	F
b < a + c?	T	T	T	T	T	T	F	-
c < a + b?	T	T	T	T	T	F	-	-
a = b?	T	T	F	F	F	-	-	-
a = c?	T	F	T	F	F	-	-	-
b = c?	T	F	F	T	F	-	-	-
Ação								
Não é Triângulo						X	X	X
Escaleno					X			
Isóceles		X	X	X				
Equilátero	X							

Como criar Casos de Teste

- Casos de Teste para o programa Tipo de Triângulo

Id	a	b	c	Saída esperada
TC_1	4	4	4	Equilátero
TC_2	5	5	3	Isóceles
TC_3	2	1	2	Isóceles
TC_4	10	15	15	Isóceles
...				

Tabelas de Decisão – Considerações Finais

- Testes baseados em Tabelas de Decisão são apropriados para programas que:
  - Possuem relações lógicas entre as entradas
  - Possuem relações causa-efeito bem definidas entre entradas e saídas

- Criação exaustiva de Tabela de Decisão ajuda a verificar completude dos casos de teste

- Tabelas de Decisão não escalam bem
  - Cresce rapidamente com o número de condições e o número de possíveis valores de cada condição

- Tabelas de Decisão podem ser iterativamente refinadas

Tabelas de Decisão – Considerações Finais

- Ao invés de exaustivamente, Tabelas de Decisão podem ser criadas iterativamente:
  - Determine as condições e seus respectivos valores possíveis
  - Determine as regras possíveis
  - Determine as ações apropriadas para cada regra
  - Verifique a completude do conjunto de regras criadas
    - Faltou alguma regra?
    - Alguma ação identificada não foi disparada?
  - Verifique a consistência do conjunto de regras criadas
    - Existem regras inconsistentes (mesmas combinações disparando ações diferentes)?
  - Simplifique o conjunto de regras (se possível)

Testes Funcionais

- Sistemática de aplicação:
  - Compreenda e analise a especificação
  - Identifique as partições
  - Analise os limites
  - Projete os casos de testes
    - Use tabela de decisão para auxiliar
  - Implemente e automatize os casos de testes
  - Analise os resultados e expanda a suíte de testes

Exercício

- Aplicar sistemática anterior para:
  - O método de consolidação parcial da turma de graduação da UFRN
  - A funcionalidade de registrar um aluguel de Jogos

Residência em TI
Gerência de Configuração e Teste de Software

Prof. Eiji Adachi