Programação Modular: Lista 2

Entrega no dia xx de novembro de 2019 às 17h

Professor Flavio Bevilacqua

Antônio Vasconcellos Chaves
Engenharia da Computação
Pontifícia Universidade Católica
do Rio de Janeiro
Rio de Janeiro, RJ 22451-900
antoniovasconcelloschaves@gmail.com

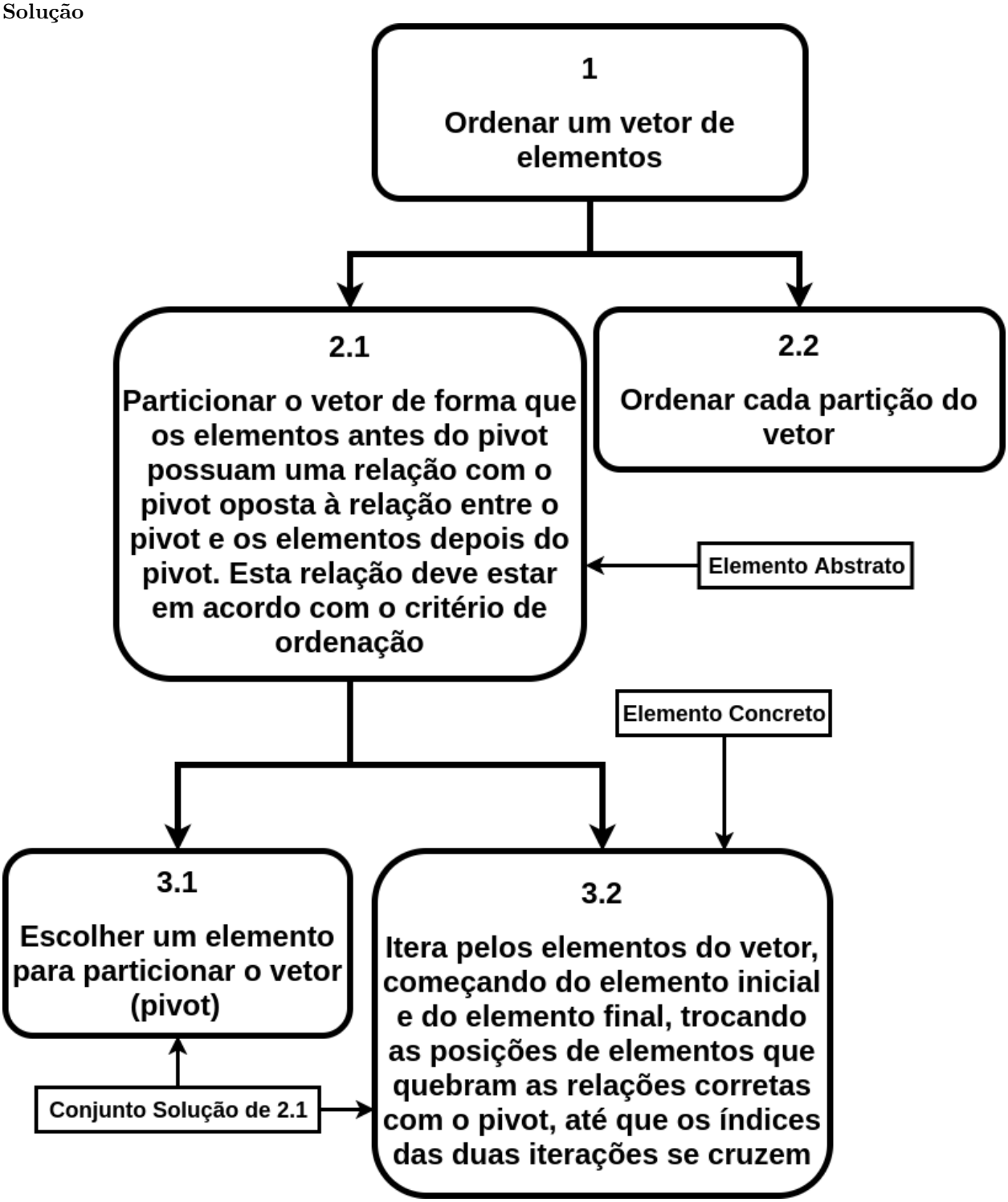
João Pedro Paiva
Ciência da Computação
Pontifícia Universidade Católica
do Rio de Janeiro
Rio de Janeiro, RJ 22451-900
joaopedrordepaiva@gmail.com

Pedro Moreira Costa
Engenharia da Computação
Pontifícia Universidade Católica
do Rio de Janeiro
Rio de Janeiro, RJ 22451-900
pedromoreiramcosta@gmail.com

Questão 1

Apresente a estrutura de decomposição sucessiva do algoritmo de quicksort apontando um componente concreto, um componente abstrato e um conjunto solução.

Solução



Questão 2

Faça a argumentação de corretude completa de uma pesquisa binária em um vetor.

Solução

Algoritmo 1: Busca Binária em Vetor

```
AE ⟶
  INÍCIO
    COMEÇO ⟵ 1
    FINAL ⟵ LIMITE-LÓGICO
    ENQUANTO COMEÇO ≤ FINAL FAÇA
      ATUAL ⟵ (COMEÇO + FINAL) / 2
      SE VETOR[ATUAL] == PARÂMETRO-BUSCADO ENTÃO RETORNA ATUAL
      SE VETOR[ATUAL] < PARÂMETRO-BUSCADO ENTÃO COMEÇO ⟵ ATUAL +
       1
      SENÃO FINAL ⟵ ATUAL - 1
    FIM ENQUANTO
    RETORNA -1
  FIM
AS ⟶
```

Argumentação de Sequência 1
AE: Existe um número a ser buscado em um vetor ordenado.
AS: PARÂMETRO-BUSCADO está na posição retornada, ou não está no vetor e o valor retornado é -1.
AI 1: COMEÇO aponta para o primeiro elemento do vetor.
AI 2: FINAL aponta para o LIMITE-LÓGICO do vetor.
AI 3: PARÂMETRO-BUSCADO não está no vetor.
Argumentação de Repetição 1
AE: AI 2.
AS: PARÂMETRO-BUSCADO está na posição retornada, ou não está no vetor.
AINV:

- Existem dois conjuntos: pode conter PARÂMETRO-BUSCADO e não contém PARÂMETRO-BUSCADO.
- COMEÇO e FINAL apontam para os limites inferior e superior, respectivamente, do conjunto pode conter PARÂMETRO-BUSCADO.

(1) AE $\Longrightarrow$ AINV

- Pela AE, FINAL aponta para o LIMITE-LÓGICO do vetor. Todos os elementos estão no conjunto pode conter PARÂMETRO-BUSCADO e o conjunto não contém PARÂMETRO-BUSCADO está vazio. Logo, vale a AINV.

(2) AE && (Condição == False) $\Longrightarrow$ AS

- Pela AE, FINAL aponta para o LIMITE-LÓGICO do vetor. Para que (Condição == False), FINAL < COMEÇO. Já que COMEÇO == 1, FINAL == 0. Logo, LIMITE-LÓGICO == 0, ou seja, o vetor está vazio. Neste caso, vale a AS, já que o PARÂMETRO-BUSCADO não está no vetor.

(3) AE && (Condição == True) $\oplus$ B $\Longrightarrow$ AINV

- Pela AE, FINAL aponta para o LIMITE-LÓGICO do vetor. Para que (Condição == True), o vetor possui ao menos um elemento. Metade dos elementos do conjunto pode conter PARÂMETRO-BUSCADO passarão para o conjunto não contém PARÂMETRO-BUSCADO, e COMEÇO e FINAL serão reposicionados, apontando para os limites do novo conjunto pode conter PARÂMETRO-BUSCADO. Com isso, os dois conjuntos existem e COMEÇO e FINAL apontam para os limites do conjunto pode conter PARÂMETRO-BUSCADO. Logo, vale a AINV.s

(4) AINV && (Condição == True) $\oplus$ B $\Longrightarrow$ AINV

- Para que a AINV continue valendo, B deve garantir que metade dos elementos do conjunto pode conter PARÂMETRO-BUSCADO passem para o conjunto não contém PARÂMETRO-BUSCADO, e COMEÇO e FINAL sejam reposicionados, apontando para os limites do novo conjunto pode conter PARÂMETRO-BUSCADO.

(5) AINV && (Condição == False) $\oplus$ B $\Longrightarrow$ AS

- Se (Condição == False), o limite inferior do conjunto pode conter PARÂMETRO-BUSCADO superou o limite superior, ou seja, todos os elementos passaram para o conjunto não contém PARÂMETRO-BUSCADO e o conjunto pode conter PARÂMETRO-BUSCADO está vazio. Como o PARÂMETRO-BUSCADO não está no vetor, vale a AS.

(6) Término

- Como a cada ciclo, metade dos elementos do conjunto pode conter PARÂMETRO-BUSCADO são retirados, e este conjunto possui um número finito de elementos, a repetição terminará em um número finito de passos.

Argumentação de Sequência 2
AE (seq2) = AS (seq2) = AINV.
AI 4: ATUAL aponta para o meio do conjunto pode conter PARÂMETRO-BUSCADO.
Argumentação de Seleção 1
AE: AI 4.
AS: AINV ou AS geral.

(1) AE && (Condição == True) $\oplus$ B1 $\Longrightarrow$ AS

Pela AE, ATUAL aponta para o meio do conjunto pode conter PARÂMETRO-BUSCADO. Como (Condição == True), ATUAL aponta para o PARÂMETRO-BUSCADO. Neste caso, executa B1 que retorna a posição de PARÂMETRO-BUSCADO, valendo a AS.

(2) AE && (Condição == False) $\oplus$ B2 $\Longrightarrow$ AS

Pela AE, ATUAL aponta para o meio do conjunto pode conter PARÂMETRO-BUSCADO. Como (Condição == False), ATUAL não aponta para o PARÂMETRO-BUSCADO. Neste caso, executa B2 que passa metade dos elementos do conjunto pode conter PARÂMETRO-BUSCADO para o conjunto não contém PARÂMETRO-BUSCADO. Vale a AS pois COMEÇO e FINAL apontam para os limites inferior e superior, respectivamente, do conjunto pode conter PARÂMETRO-BUSCADO.

Argumentação de Seleção 2
AE (sel2) = AE (sel1) e ATUAL não aponta para o PARÂMETRO-BUSCADO.
AS: AINV e metade inferior ou superior dos elementos do conjunto pode conter PARÂMETRO-BUSCADO foi passada para o conjunto não contém PARÂMETRO-BUSCADO.

Questão 2 continua na próxima página...

① AE && (Condição == True) ⊕ B1 ⟹ AS

Pela AE, ATUAL aponta para o meio do conjunto pode conter PARÂMETRO-BUSCADO mas não aponta para o PARÂMETRO-BUSCADO. Como (Condição == True), o elemento apontado por ATUAL é menor do que PARÂMETRO-BUSCADO. Neste caso, executa B1 que redefine COMEÇO, passando a metade inferior dos elementos do conjunto pode conter PARÂMETRO-BUSCADO para o conjunto não contém PARÂMETRO-BUSCADO e valendo a AS.

② AE && (Condição == False) ⊕ B2 ⟹ AS

Pela AE, ATUAL aponta para o meio do conjunto pode conter PARÂMETRO-BUSCADO mas não aponta para o PARÂMETRO-BUSCADO. Como (Condição == False), o elemento apontado por ATUAL é maior ou igual que PARÂMETRO-BUSCADO. Neste caso, executa B2 que redefine FINAL, passando a metade superior dos elementos do conjunto pode conter PARÂMETRO-BUSCADO para o conjunto não contém PARÂMETRO-BUSCADO e valendo a AS.

Questão 3

Escolha uma função do trabalho e distribua os controladores de cobertura pelo padrão cobertura de arestas.

Solução

```
int LIS_verificador(LIS_tppCabecaLista pCabecaDaLista)
{

    int numFalhasObservadas, numNos, numSucessosObservados;

    LIS_tpNoLista *pNo, *ultimoNoDaVolta;

    if (!pCabecaDaLista)
        return LIS_CondRetListaNaoExiste;

    CNT_InicializarContadores("EXEC/contagemacumulada.txt");
    CNT_IniciarContagem();
    CNT_LerContadores("EXEC/contadores.txt");

    numFalhasObservadas = 0;
    numSucessosObservados = 0;

    if (!pCabecaDaLista->ExcluirValor)

    {
        ++numFalhasObservadas;
        CNT_Contar("13s", 0);
    }
    else
    {
        ++numSucessosObservados;
        CNT_Contar("13n", 0);
    }

    if (pCabecaDaLista->numNos && !pCabecaDaLista->pNoCorrente)

    {
        ++numFalhasObservadas;
        CNT_Contar("9s", 0);
```

Questão 3 continua na próxima página...

```
    }
    else
    {
        ++numSucessosObservados;
        CNT_Contar("9n", 0);

        if (pCabecaDaLista->numNos && pCabecaDaLista->pNoCorrente->pCabeca != pCabecaDaLista
            )

        {
            ++numFalhasObservadas;
            CNT_Contar("11s", 0);
        }
        else
        {
            ++numSucessosObservados;
            CNT_Contar("11n", 0);
        }
    }

    if (pCabecaDaLista->numNos && !pCabecaDaLista->pNoPrimeiro)

    {
        ++numFalhasObservadas;
        CNT_Contar("10s", 0);

        pCabecaDaLista->pNoPrimeiro = pCabecaDaLista->pNoUltimo;

        while (pCabecaDaLista->pNoPrimeiro->pNoAnterior)
            pCabecaDaLista->pNoPrimeiro = pCabecaDaLista->pNoPrimeiro->pNoAnterior;
    }
    else
    {
        ++numSucessosObservados;
        CNT_Contar("10n", 0);
        if (pCabecaDaLista->numNos && pCabecaDaLista->pNoPrimeiro->pCabeca != pCabecaDaLista
            )

        {
            ++numFalhasObservadas;
            CNT_Contar("12s", 0);
            pCabecaDaLista->pNoPrimeiro = pCabecaDaLista->pNoUltimo;

            while (pCabecaDaLista->pNoPrimeiro->pNoAnterior)
                pCabecaDaLista->pNoPrimeiro = pCabecaDaLista->pNoPrimeiro->pNoAnterior;
        }
        else
        {
            ++numSucessosObservados;
            CNT_Contar("12n", 0);
        }
    }


    pNo = pCabecaDaLista->pNoPrimeiro;

    numNos = pCabecaDaLista->numNos;

    ultimoNoDaVolta = pCabecaDaLista->pNoPrimeiro;

    while (1)
    {
```

```
        if (!pNo)
            break;

        if (pCabecaDaLista->tipoEstrutura != pNo->tipoEstrutura)

        {
            ++numFalhasObservadas;
            CNT_Contar("7s", 0);
        }

        else
        {
            ++numSucessosObservados;
            CNT_Contar("7n", 0);
        }

        if (pCabecaDaLista->tamNo != pNo->tamNo)

        {
            ++numFalhasObservadas;
            CNT_Contar("14s", 0);
        }

        else
        {
            ++numSucessosObservados;
            CNT_Contar("14n", 0);
        }

        if (!pNo->pConteudo)
        {
            ++numFalhasObservadas;
            CNT_Contar("6s", 0);
        }

        else
        {
            ++numSucessosObservados;
            CNT_Contar("6n", 0);
        }

        if (!pNo->pNoProximo && numNos == 1)

        {
            break;
        }

        if (!pNo->pNoProximo && numNos != 1)

        {
            ++numFalhasObservadas;
            CNT_Contar("2s", 0);
            ultimoNoDaVolta = pNo;
            break;
        }

        else
        {
            ++numSucessosObservados;
            CNT_Contar("2n", 0);
        }
```

```
        if (pNo->pNoProximo && numNos == 0)

        {
            ++numFalhasObservadas;
            CNT_Contar("4s", 0);
            break;
        }

        else
        {
            ++numSucessosObservados;
            CNT_Contar("4n", 0);
        }

        if (!pNo->pNoProximo->pNoAnterior)

        {
            ++numFalhasObservadas;
            CNT_Contar("3s", 0);
        }

        else
        {
            ++numSucessosObservados;
            CNT_Contar("3n", 0);

            if (pNo->pNoProximo->pNoAnterior != pNo)

            {
                ++numFalhasObservadas;
                CNT_Contar("5s", 0);
            }

            else
            {
                ++numSucessosObservados;
                CNT_Contar("5n", 0);
            }
        }

        pNo = pNo->pNoProximo;
        --numNos;
    }

    if (pNo != pCabecaDaLista->pNoUltimo)
    {


        pNo = pCabecaDaLista->pNoUltimo;

        numNos = pCabecaDaLista->numNos;

        while (pNo != ultimoNoDaVolta)
        {

            if (pCabecaDaLista->tipoEstrutura != pNo->tipoEstrutura)

            {
                ++numFalhasObservadas;
                CNT_Contar("7s", 0);
            }
```

```
            else
            {
                ++numSucessosObservados;
                CNT_Contar("7n", 0);
            }

            if (pCabecaDaLista->tamNo != pNo->tamNo)

            {
                ++numFalhasObservadas;
                CNT_Contar("14s", 0);
            }

            else
            {
                ++numSucessosObservados;
                CNT_Contar("14n", 0);
            }

            if (!pNo->pConteudo)
            {
                ++numFalhasObservadas;
                CNT_Contar("6s", 0);
            }

            else
            {
                ++numSucessosObservados;
                CNT_Contar("6n", 0);
            }

            if (!pNo->pNoAnterior && numNos == 1)

            {
                break;
            }

            if (!pNo->pNoAnterior && numNos != 1)

            {
                ++numFalhasObservadas;
                CNT_Contar("3s", 0);
                break;
            }

            else
            {
                ++numSucessosObservados;
                CNT_Contar("3n", 0);
            }

            if (pNo->pNoAnterior && numNos == 0)

            {
                ++numFalhasObservadas;
                CNT_Contar("5s", 0);
                break;
            }

            else
            {
```

```
                ++numSucessosObservados;
                CNT_Contar("5n", 0);
            }

            if (pNo->pNoAnterior != ultimoNoDaVolta && !pNo->pNoAnterior->pNoProximo)

            {
                ++numFalhasObservadas;
                CNT_Contar("2s", 0);
            }

            else
            {
                ++numSucessosObservados;
                CNT_Contar("2n", 0);

                if (pNo->pNoAnterior != ultimoNoDaVolta && pNo->pNoAnterior->pNoProximo !=
                    pNo)

                {
                    ++numFalhasObservadas;
                    CNT_Contar("4s", 0);
                }

                else
                {
                    ++numSucessosObservados;
                    CNT_Contar("4n", 0);
                }
            }

            pNo = pNo->pNoAnterior;
            --numNos;
        }
    }

    CNT_GravarContadores("EXEC/contagemacumulada.txt");
    CNT_Parar Contagem();
    CNT_TerminarContadores();

    return numFalhasObservadas;
}
```

Questão 4

Transforme uma estrutura de matriz tridimensional criada com listas em autoverificável.

Solução

Questão 5

Elabore um verficador para a estrutura da questão quatro com três verificações de campos redundantes.

Solução

Questão 6

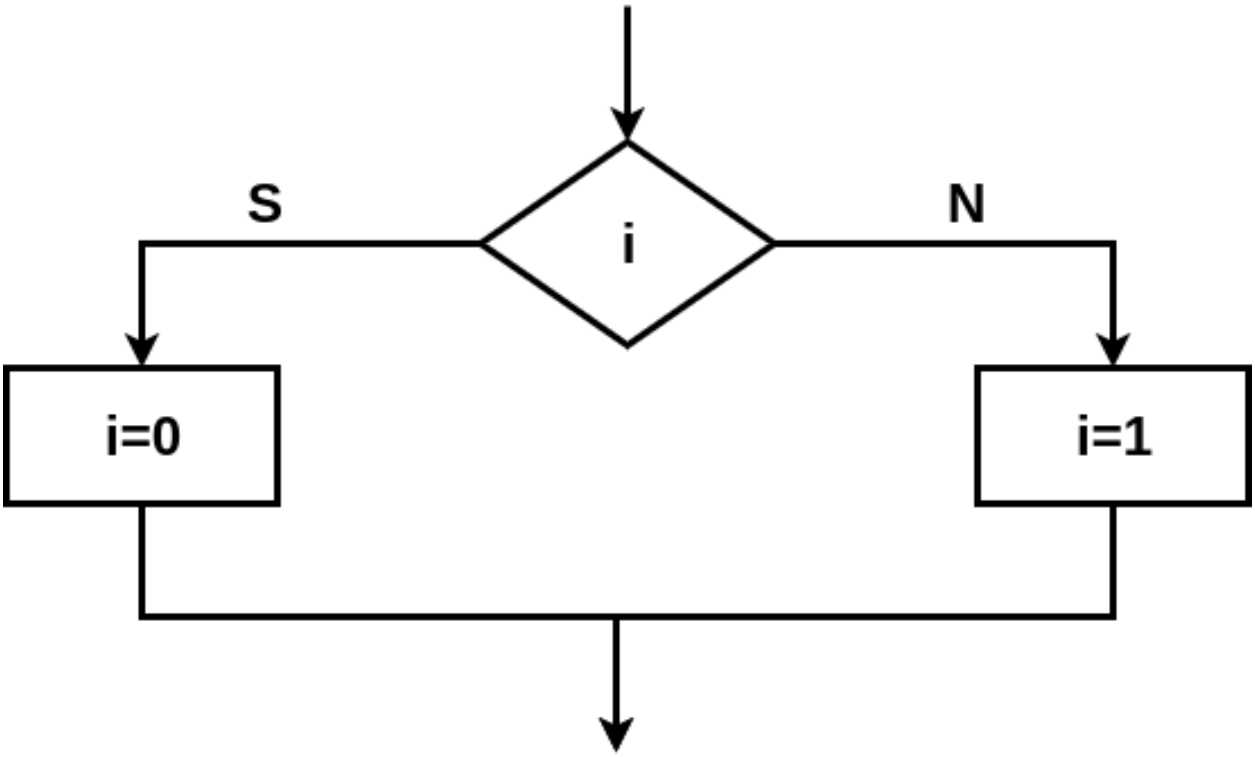
Apresente um deturpador utlizado para testar o verficador da questão cinco.

Solução

Questão 7

Apresente uma situação em que os casos de teste pelo padrão cobertura de comandos, decisões e arestas são exatamente iguais.

Solução



Questão 8

Qual é o arrasto de uma repetição existente em uma pesquisa binária recursiva?

Solução
2.

Questão 9

Um controle de qualidade garante a corretude de um código quando seu teste é completo e não apresenta erros. Certo/Errado/Justifique.

Solução
Errado. O teste não apresentar erros significa que a implementação possui alta qualidade observada, mas não garante que a qualidade atestada seja igualmente alta caso a análise sobre o teste seja incorreta. Mesmo que o teste seja completo e a análise sobre o teste correta, não significa que o critério de seleção de casos de teste seja válido, o teste pode não acusar falhas quando há, ou confiável, o teste pode não paresentar erros por uma escolha específica ou ação feito em um dos casos de teste.

Questão 10

Apresente todos os casos de teste do deturpador do trabalho 4 pelo padrão cobertura de arestas.

Solução

Deturpação/Caso	1	2	3	4	5	6	7	8	9	10	11	12	13	14
==1	S	N	N	N	N	N	N	N	N	N	N	N	N	N
==2	NT	S	N	N	N	N	N	N	N	N	N	N	N	N
==3	NT	NT	S	N	N	N	N	N	N	N	N	N	N	N
==4	NT	NT	NT	S	N	N	N	N	N	N	N	N	N	N
==5	NT	NT	NT	NT	S	N	N	N	N	N	N	N	N	N
==6	NT	NT	NT	NT	NT	S	N	N	N	N	N	N	N	N
==7	NT	NT	NT	NT	NT	NT	S	N	N	N	N	N	N	N
==8	NT	NT	NT	NT	NT	NT	NT	S	N	N	N	N	N	N
==9	NT	NT	NT	NT	NT	NT	NT	NT	S	N	N	N	N	N
==10	NT	NT	NT	NT	NT	NT	NT	NT	NT	S	N	N	N	N
==11	NT	NT	NT	NT	NT	NT	NT	NT	NT	NT	S	N	N	N
==12	NT	NT	NT	NT	NT	NT	NT	NT	NT	NT	NT	S	N	N
==13	NT	NT	NT	NT	NT	NT	NT	NT	NT	NT	NT	NT	S	N
==14	NT	NT	NT	NT	NT	NT	NT	NT	NT	NT	NT	NT	NT	S

S: sim
N: não
NT: não testa

Questão 11

Utilizando o método de partição em classes de equivalência, gere o script para uma exclusão de nó corrente de uma árvore binária.

Solução
Passo 1: Árvore vazia, árvore com um único nível, árvore com três níveis (possui nível intermendiário).
Resultados esperados: excluiu, não excluiu.

Questão 11 continua na próxima página...

Passo 2:

Caso	Estrutura	Elemento a excluir	Excluiu	Não Excluiu
S	Vazia	Primeiro	N	S
2	A	Raiz	S	N
3	(A,(B,C))	Raiz	S	N
4	(A,(B,C))	Filho à direita	S	N
5	(A,(B,C))	Filho à esquerda	S	0

Passo 3:

==caso1
=criaarvore 0
=exclui 0

==caso2
=insere A
=exclui 0

==caso3
=insere A
=insere B
=insere C
=irpai 0
=exclui 0
=liberaarvore 0

==caso4
=insere A
=insere B
=insere C
=exclui 0
=liberalista 0

==caso5
=insere A
=insere B
=insere C
=irpai 0
=iresq 0
=exclui 0
=liberalista 0

Questão 12

Apresente uma situação em que as qualidades efetiva e observada são diferentes.

Solução

Quando a implementação de um artefato é incorreta e o teste é incompleto, não abordando o aspecto incorreto da implementação, mas a análise dos resultados do teste é correta, a qualidade observada será alta, a qualidade atestada será baixa e a qualidade efetiva será baixa.

Questão 13

Existe relação entre a abordagem de teste orientado à destruição e a qualidade atestada não se aproximar da qualidade requerida. Certo/Errado/Justifique.

Solução
A qualidade requerida aumenta quando nos aproximamos daquilo que o cliente quer. A qualidade atestada se refere ao que é descrito por um controle de qualidade. Ao utlizarmos a abordagem de teste orientado à destruição, examinamos somente se a implementação de algum artefato está correta, mas não se aquela é a implementação que o cliente deseja. Utlizando esta abordagem, garantimos que a implementação possua boa qualidade, mas não garantimos qualidade satisfatória.

Questão 14

Explique porque é necessário utilizar campos redundantes ao instrumentar uma estrutura autoverificável.

Solução
Os dados redundantes ajudam na confiabilidade daquilo que foi implementado. Por exemplo, se colocarmos um campo que descreve o tipo da estrutura na cabeça da estrutura e nos nós, então podemos verificar se o nó pertence à estrutura pela combinação, ou não, do tipo apontado pelo nó e pela cabeça. A implementação de múltiplos campos com este mesmo objetivo, como um campo que descreve o tamanho do nó colocado na cabeça da estrutura e nos nós, enfatizará ainda mais a corretude da implementação.