Bacharelado em Ciência da Computação

Fundamento de Computação e Lógica Digital -2021-1

Trabalho 03

|  |
| --- |
| Fundamento de computação e Lógica Digital |

|  |  |
| --- | --- |
| Professor da Disciplina: | Ildeberto de Genova Bugatti |
| Turma: | A-S |
| Identificação do grupo: | Grupo 3 |
| Nomes e Ras: | 614084 – André Genoti Dantas  604860 – João Victor Tudela Gabriel  617571 – Lucas Ferrari Lopes  614491 – Lucas Matheus de Souza Marques  614432 – Luís Felipe Ribeiro Campos |
| Título do Trabalho: | Códigos Numéricos |



1 - Responda os itens que seguem:

a) Defina distância entre palavras:

R: Distância entre palavras e a quantidade de bits que diferem entre duas palavras-código consecutivas.

b) O que é distância de um código?

R: Distância de um código é a menor distância entre duas palavras quaisquer de um código.

c) Quais as principais características de um código ponderado?

R: Nos códigos ponderados são necessários no mínimo 4 bits para representar toda a base 10 (dez), cada bit possui um peso associado à sua posição cujos quais somados devem ser maiores ou iguais a 9 (nove).

Para cada palavra-código é necessário existir pesos que somados geram os bits que representam essa palavra na base 10 (dez).

d) O que é bit de paridade?

R: Bit de paridade indica se a soma de todos os bits de valor 1 de uma palavra código é par ou ímpar.

e) Defina os tipos de paridade existentes.

R: Existem dois tipos de paridade:

Paridade par – define-se paridade par quando a soma de todos os bits de valor 1 de uma palavra código é par.

Paridade ímpar – define se paridade ímpar quando a soma de todos os bits de valor 1 de uma palavra código é ímpar.

f) Quais as principais características de um código detector de erros?

R: Códigos detectores de erros fazem uso de bit de paridade para detectar a ocorrência de um único erro (quando 0 se torna 1 ou vice e versa). Para conseguir detectar n erros, a distância do código deve ser n+1.

g) Quais as principais características de um código corretor de erros?

R: Para dotar um código com a capacidade de detectar e corrigir erros, a distância de código deve ser igual a (2\*n) + 1 (n representa a quantidade de erros existentes), para isso, usamos o código de Hamming para acrescentar bits de paridade em suas respectivas posições.

A quantidade de bits a ser adicionada no código respeita a inequação “2^k >= m + k + 1” onde m representa a quantidade de bits existentes em um código e k representa a quantidade de bits a ser acrescentada no código.

Ao final do processo e possível encontrar a palavra-código incorreta através do resultado da checagem da paridade.

h) Quais as principais características de um código cíclico?

R: Códigos cíclicos possuem distância entre palavras igual a 1 assim como a distância entre a primeira e a última palavra.

2 - Construa um código ponderado auto complementável que possua ao menos um peso negativo.

Qual a distância do código construído?

Inclua um bit de paridade ÍMPAR no código construído.

Qual é a distância do código ponderado com paridade ÍMPAR?

Por quê?

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Pesos | | | | Bit de paridade Impar | |
| Posições | -1 | 2 | 3 | 5 |
| 0 | 0 | 0 | 0 | 0 |  | 1 |
| 1 | 1 | 1 | 0 | 0 |  | 1 |
| 2 | 0 | 1 | 0 | 0 |  | 0 |
| 3 | 0 | 0 | 1 | 0 |  | 0 |
| 4 | 1 | 0 | 0 | 1 |  | 1 |
| 5 | 0 | 0 | 0 | 1 |  | 0 |
| 6 | 1 | 1 | 0 | 1 |  | 0 |
| 7 | 0 | 1 | 0 | 1 |  | 1 |
| 8 | 0 | 0 | 1 | 1 |  | 1 |
| 9 | 1 | 1 | 1 | 1 |  | 1 |

Antes do acréscimo do bit de paridade ímpar, a distância do código era 1(um) portanto não é possível detectar erros.

Como consequência do acréscimo de um bit de paridade, a distância do código passou a ser 2 (dois), sendo assim possível a detecção de um erro único.

3-Construa um código de Hamming para o código ponderado (4, 1, -2, 7).Teste o código gerado na questão anterior, simulando as seguintes condições:

a) Simule o envio e recepção de uma palavra sem ocorrência de erro. Comente como é verificada a não ocorrência de erro.

b) Simule o envio e recepção de uma palavra com ocorrência de erro único. Comente como é detectada a ocorrência do erro. Comente como é corrigido o erro.

2^k = m + k + 1

k = 3

m = 4

8 >= 8

A paridade exigida pelo exercício e ímpar!

‘k’ é a quantidade mínima de bits de paridade necessária para detecção e correção de erros (respeitando a inequação), sendo assim a quantidade mínima de bits necessária e 3(três).

‘m’ é a quantidade de bits que o código construído possui.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Pesos | | |  | Possibilidades com 3 bits de paridade Impar | | | | |
|  | m1 | m2 | m3 | m4 | 0 |  | 0 | 0 | 0 |
| Posições | 4 | 1 | -2 | 7 | 1 |  | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 0 | 2 |  | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 0 | 3 |  | 0 | 1 | 1 |
| 2 | 1 | 0 | 1 | 0 | 4 |  | 1 | 0 | 0 |
| 3 | 1 | 1 | 1 | 0 | 5 |  | 1 | 0 | 1 |
| 4 | 1 | 0 | 0 | 0 | 6 |  | 1 | 1 | 0 |
| 5 | 1 | 1 | 0 | 0 | 7 |  | 1 | 1 | 1 |
| 6 | 0 | 1 | 1 | 1 |  | | | | |
| 7 | 0 | 0 | 0 | 1 |
| 8 | 0 | 1 | 0 | 1 |
| 9 | 1 | 0 | 1 | 1 |

Dispor aleatoriamente cada uma das linhas das possibilidades com 3 bits de paridade ímpar.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Possibilidades com 3 bits de paridade Impar aleatoriamente dispostas  Atenção: A linha 000 deve obrigatoriamente ser a primeira!  Logo após, encontrar todos os bits de valor 1 em cada COLUNA da tabela, encontrar a respectiva linha cuja qual as outras colunas possuam valor 0.  P3 possui 1s nas linhas:  P3 = [2,3,5,6]  1 único na linha: 3    P2 possui 1s nas linhas:  P2 = [1,2,5,7]  1 único na linha: 7 | | | | |  |
| Depois | Antes | p1 | p2 | p3 |
| 0 | 0 | 0 | 0 | 0 |
| 1 | 6 | 1 | 1 | 0 |
| 2 | 3 | 0 | 1 | 1 |
| 3 | 1 | 0 | 0 | 1 |
| 4 | 4 | 1 | 0 | 0 |
| 5 | 7 | 1 | 1 | 1 |
| 6 | 5 | 1 | 0 | 1 |
| 7 | 2 | 0 | 1 | 0 |
|  |  |  |  |  |

P1 possui 1s nas linhas:

P1 = [1,4,5,6]

1 único na linha: 4

Cada posição das linhas com “1 único” acima citada será a coluna na qual os bits de paridade serão acrescentados no código, assim como cada linha das colunas nas quais foram encontradas bits de valor 1 serão as colunas cujas quais o bit de paridade será responsável pela checagem, ou seja:

Os bits de paridade p3 se encontram na coluna 3 do código e são responsáveis por checar as colunas 2, 5 e 6

os bits de paridade p2 se encontram na coluna 7 do código e são responsáveis por checar as colunas 1, 2 e 3

Os bits de paridade p1 se encontram na coluna 4 do código e são responsáveis por checar as colunas 1, 5 e 6.

Aplicando-se isso, gera o seguinte resultado:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | m + k bits | | | | | | | | |
| Posições |  |  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|  |  |  | m1 | m2 | p3 | p1 | m3 | m4 | p2 |
| 0 |  |  | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 |  |  | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 2 |  |  | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 3 |  |  | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 4 |  |  | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 5 |  |  | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 6 |  |  | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 7 |  |  | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 8 |  |  | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 9 |  |  | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
|  |  |  |  |  |  |  |  |  |  |

Essas são as palavras código do código de Hamming.

Para checar a integridade das palavras código faz-se o seguinte processo:

Obtém se a palavra-código incorreta:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Correta | 4 |  | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| Incorreta | 4 |  | 1 | 1 | 1 | 0 | 0 | 0 | 0 |

O bit em vermelho representa o bit no qual ocorre o erro.

O bit em azul representam os bits de paridade.

Faz-se a checagem de cada grupo de código representado pelos bits de paridade:

A palavra escolhida foi a palavra 4 do código.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Posições | 1 | 2 | 3 | 4 | 5 | 6 | 7 |  |
| Correta |  | 1 | 0 | 1 | 0 | 0 | 0 | 0 |  |
| Incorreta |  | 1 | 1 | 1 | 0 | 0 | 0 | 0 |  |
|  | Posições | 1 | 2 | 3 | 4 | 5 | 6 | 7 | P. Impar |
| P1 | P[1,4,5,6] | 1 |  |  | 0 | 0 | 0 |  | 0 |
| P2 | P[1,2,5,7] | 1 | 1 |  |  | 0 |  | 0 | 1 |
| P3 | P[2,3,5,6] |  | 1 | 1 |  | 0 | 0 |  | 1 |

Através dos bits de paridade obtidos, podemos encontrar o bit errado e corrigi-lo checando a tabela de códigos dispostos aleatoriamente, neste caso o bit com o erro é aquele cujos bits de paridade formam o código ‘011’.

Este código está localizado na linha 2 da tabela, logo o erro se encontra na coluna 2 do código de Hamming.

Para a correção basta complementar o bit errôneo.

Em casos de não ocorrência de erros, todos os bits de paridade são representados por ‘000’, neste caso não há necessidade de complementar nenhum bit da palavra.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Posições | 1 | 2 | 3 | 4 | 5 | 6 | 7 |  |
| Correta |  | 1 | 0 | 1 | 0 | 0 | 0 | 0 |  |
| Incorreta |  | 1 | 1 | 1 | 0 | 0 | 0 | 0 |  |
|  | Posições | 1 | 2 | 3 | 4 | 5 | 6 | 7 | P. Impar |
| P1 | P[1,4,5,6] | 1 |  |  | 0 | 0 | 0 |  | 0 |
| P2 | P[1,2,5,7] | 1 | 1 |  |  | 0 |  | 0 | 0 |
| P3 | P[2,3,5,6] |  | 1 | 1 |  | 0 | 0 |  | 0 |