Universidad del Valle de Guatemala
Redes
Ciclo II, 2024
Fecha de entrega: 25 jul 2024 19:00

Adrián Ricardo Flores Trujillo	21666
Andrea Ximena Ramírez Recinos	21874

Laboratorio 2

Esquemas de Detección y Corrección.

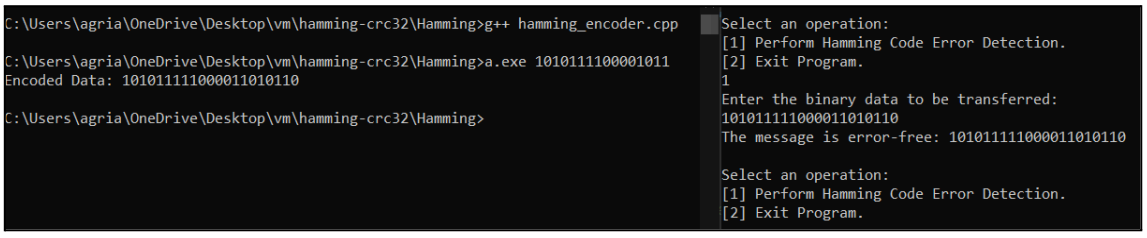
[Parte 1] Mensajes Libres de Errores

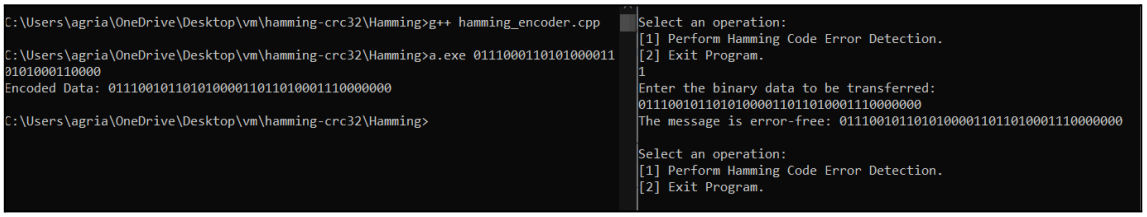
Enviar un mensaje al emisor, copiar el mensaje generado por este y proporcionarlo tal cual al receptor, el cual debe mostrar el mensajes originales. Realizar esto para ***tres mensajes distintos*** con distinta longitud.

¡Por favor note que se estarán realizando múltiples pruebas con las siguientes entradas y los resultados de sus codificaciones!

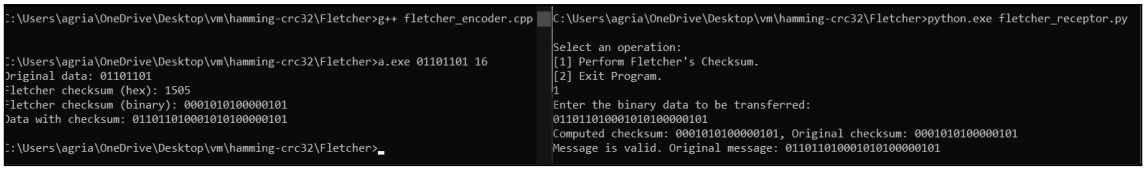
- **Mensajes Originales:**
  - 01101101 (8 bits)
  - 1010111100001011 (16 bits)
  - 01110001101010000110101000110000 (32 bits)

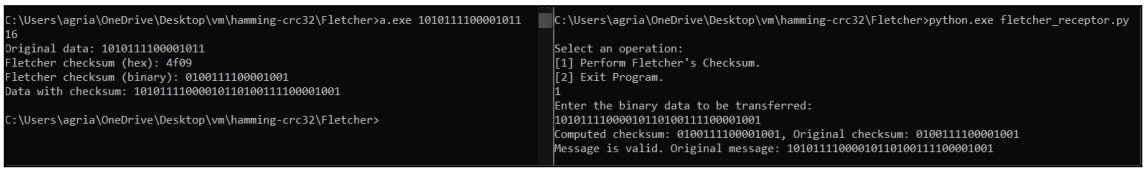
Hamming

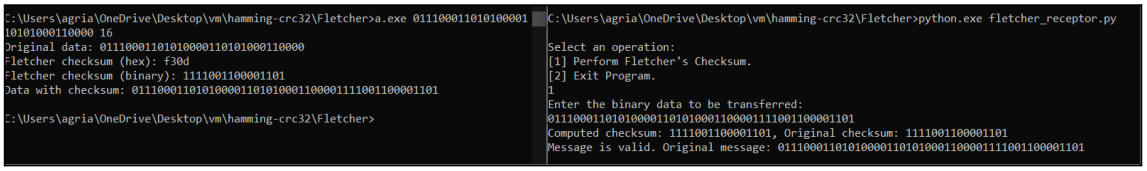




Fletcher's Checksum







[Parte 2] Mensajes Con Bit Alterado

Enviar un mensaje al emisor, copiar el mensaje generado por este y cambiar un bit cualquiera antes de proporcionarlo al receptor. Si el algoritmo es de detección debe mostrar que se detectó un error y que se descarta el mensaje. Si el algoritmo es de corrección debe corregir el bit, indicar su posición y mostrar el mensaje original.

Hamming

- **Mensajes Originales Codificados :**
  - 011001100111 (8 bits)
  - 1010111110000110101 10 (16 bits)
  - 01110010110101000011011010001110000000 (32 bits)

- **Mensajes Codificados Alterados :**
  - 01100110001 1 (8 bits)
  - 1010111110000100 10110 (16 bits)
  - 011000101101010000110110100011100000000 (32 bits)

```
Select an operation:
[1] Perform Hamming Code Error Detection.
[2] Exit Program.
1
Enter the binary data to be transferred:
011001100011
Error: The message has errors at position 3. Message discarded.
Correct message: 011001100111
```

```
Select an operation:
[1] Perform Hamming Code Error Detection.
[2] Exit Program.
1
Enter the binary data to be transferred:
1010111110000100 10110
Error: The message has errors at position 7. Message discarded.
Correct message: 1010111110000110101 10
```

```
Select an operation:
[1] Perform Hamming Code Error Detection.
[2] Exit Program.
1
Enter the binary data to be transferred:
01100010111010100001101101000111000000
Error: The message has errors at position 35. Message discarded.
Correct message: 01110010111010100001101101000111000000
```

Fletcher's Checksum

- **Mensajes Originales Codificados :**
  - 011011010001010100000101 (8 bits)
  - 1010111100001011010011111000001001 (16 bits)
  - 01110001101010000110101000110000111110011000001101 (32 bits)

- **Mensajes Codificados Alterados :**
  - 011011010001010100000001 (8 bits)
  - 1010111100001011010011111000000001 (16 bits)
  - 01110001101010000110101000100000111110011000001101 (32 bits)

```
C:\Users\agria\OneDrive\Desktop\vm\hamming-crc32\Fletcher>python.exe fletcher_receptor.py

Select an operation:
[1] Perform Fletcher's Checksum.
[2] Exit Program.
1
Enter the binary data to be transferred:
011011010001010100000001
Computed checksum: 0001010100000101, Original checksum: 0001010100000001
Error: This message has errors. Message discarded.
```

```
C:\Users\agria\OneDrive\Desktop\vm\hamming-crc32\Fletcher>python.exe fletcher_receptor.py

Select an operation:
[1] Perform Fletcher's Checksum.
[2] Exit Program.
1
Enter the binary data to be transferred:
1010111100001011010011110000001
Computed checksum: 0100111100001001, Original checksum: 0100111100000001
Error: This message has errors. Message discarded.
```

```
C:\Users\agria\OneDrive\Desktop\vm\hamming-crc32\Fletcher>python.exe fletcher_receptor.py

Select an operation:
[1] Perform Fletcher's Checksum.
[2] Exit Program.
1
Enter the binary data to be transferred:
01110001101010000110101000100000111100100001101
Computed checksum: 1110111000001100, Original checksum: 1111001100001101
Error: This message has errors. Message discarded.
```

[Parte 3] Mensajes Con 2 Bits Alterados

Enviar un mensaje al emisor, copiar el mensaje generado por este y cambiar dos o más bits cualesquiera antes de proporcionarlo al receptor. Si el algoritmo es de detección debe mostrar que se detectó un error y que se descarta el mensaje. Si el algoritmo es de corrección y puede corregir más de un error, debe corregir los bits, indicar su posición y mostrar el mensaje original.

- A pesar de que tanto Hamming como Fletcher son capaces de reconocer más de un bit alterado, ninguno de los dos puede corregir los errores en el mensaje, especialmente el algoritmo tradicional de Hamming. Cabe destacar que existen ***variantes*** de Hamming que sí permiten esto. Por lo expuesto con anterioridad, en este apartado simplemente se colocarán los ejemplos pero no se espera ver una corrección apropiada del mensaje. También es importante mencionar que hay ocasiones en las que múltiples alteraciones de

la cadena pueden conllevar a una excepción, mientras que hay otras en las que corrige un bit de manera errónea, por favor tener esto en cuenta a la hora de leer este documento.

Hamming

- **Mensajes Originales Codificados :**
  - 011001100111 (8 bits)
  - 1010111110000110101l0 (16 bits)
  - 01111001011010100001101101000111000000000 (32 bits)

- **Mensajes Codificados Alterados :**
  - 010001100011 (8 bits)
  - 1010011111000010010110 (16 bits)
  - 011000101101010000110100100011110000000 (32 bits)

```
Select an operation:
[1] Perform Hamming Code Error Detection.
[2] Exit Program.
1
Enter the binary data to be transferred:
010001100011
Error: The message has errors at position 9. Message discarded.
Correct message: 010101100011
```

010101100011 != 011001100111

```
Select an operation:
[1] Perform Hamming Code Error Detection.
[2] Exit Program.
1
Enter the binary data to be transferred:
10100111110000010010110
Error: The message may have multiple errors.
```

```
Select an operation:
[1] Perform Hamming Code Error Detection.
[2] Exit Program.
1
Enter the binary data to be transferred:
011000101101010000110100100011110000000
Error: The message may have multiple errors.
```

Fletcher's Checksum

- **Mensajes Originales Codificados :**
  - 011011010001010100000101 (8 bits)
  - 1010111100001011010011110000l1001 (16 bits)
  - 011100011010100001101010001100001111001100001101 (32 bits)

- **Mensajes Codificados Alterados :**
  - 011011010000010100000001 (8 bits)
  - 1000111110000101101001111000000001 (16 bits)
  - 011000011010100001101010001000001111001100001101 (32 bits)

```
C:\Users\agria\OneDrive\Desktop\vm\hamming-crc32\Fletcher>python.exe fletcher_receptor.py

Select an operation:
[1] Perform Fletcher's Checksum.
[2] Exit Program.
1
Enter the binary data to be transferred:
0110110100000101000000001
Computed checksum: 0001010100000101, Original checksum: 0000010100000001
Error: This message has errors. Message discarded.
```

```
C:\Users\agria\OneDrive\Desktop\vm\hamming-crc32\Fletcher>python.exe fletcher_receptor.py

Select an operation:
[1] Perform Fletcher's Checksum.
[2] Exit Program.
1
Enter the binary data to be transferred:
1000111100001011010011100000001
Computed checksum: 0100000100001000, Original checksum: 0100111100000001
Error: This message has errors. Message discarded.
```

```
C:\Users\agria\OneDrive\Desktop\vm\hamming-crc32\Fletcher>python.exe fletcher_receptor.py

Select an operation:
[1] Perform Fletcher's Checksum.
[2] Exit Program.
1
Enter the binary data to be transferred:
01100001101010000110101000100000111100100001101
Computed checksum: 1101000100001011, Original checksum: 1111001100001101
Error: This message has errors. Message discarded.
```

[Parte 4] Sección de Preguntas

1. ¿Es posible manipular los bits de tal forma que el algoritmo seleccionado no sea capaz de detectar el error? ¿Por qué sí o por qué no? En caso afirmativo, demuestrelo con su implementación.

- Sí, en ambos algoritmos esto es posible. En el caso del algoritmo de Fletcher, esto podría llegar a suceder si los errores son tales que las sumas parciales resultantes se cancelan mutuamente, ya que recordemos que todo el algoritmo se ve limitado por los resultados de las operaciones de adición. En otras palabras, esto quiere decir que si se altera una n cantidad de bits procurando no afectar el resultado de la suma final, también conocida como el checksum, los errores pasarán por desapercibido.

  Lamentablemente, hallar dos cadenas que produzcan el mismo checksum es una tarea compleja, y en nuestra pequeña investigación fuimos capaces de hallar dos de ellas pero se producen mediante Fletcher 8 y no 16. Sin embargo, sigue siendo una muestra contundente de que errores como estos pueden llegar a pasar totalmente desapercibidos en este algoritmo.

```
C:\Users\agria\OneDrive\Desktop\vm\hamming-crc32\Fletcher>a.exe 10100010100001101101101111100110 8
Original data: 10100010100001101101101111100110
Fletcher checksum (hex): a2
Fletcher checksum (binary): 10100010
Data with checksum: 1010001010000110110110111110011010100010

C:\Users\agria\OneDrive\Desktop\vm\hamming-crc32\Fletcher>a.exe 00100010100001111101101111100110 8
Original data: 00100010100001111101101111100110
Fletcher checksum (hex): a2
Fletcher checksum (binary): 10100010
Data with checksum: 0010001010000111110110111110011010100010
```

- **Mensaje + Checksum (Sin alteraciones) =** 1010001010000110110110111110011010100010
- **Mensaje + Checksum (Bits alterados) =** 0010001010000111110110111110011010100010

- Por otro lado, en cuanto a Hamming, como ya se mencionó con anterioridad, este algoritmo se ve fuertemente limitado por el hecho de que no puede corregir más de un error. En dado caso más de un bit es alterado, tiende a malinterpretar el error de tal manera que corrige un

bit erróneo, pasando otro error por desapercibido, o simplemente no poder corregir los errores. La evidencia de esto, se encuentra en las imágenes que ya se han colocado en el apartado anterior.

2. **En base a las pruebas que realizó, ¿qué ventajas y desventajas posee cada algoritmo con respecto a los otros dos?**

Hamming

- Dentro de sus ventajas se encuentra que es capaz de detectar y corregir errores de un solo bit, claramente asegurando la integridad de los datos y con tan solo una mínima redundancia a los datos originales. Además, es un algoritmo bastante fácil de implementar. Por otro lado, en cuanto a sus limitaciones, se incluye que es incapaz de corregir errores de múltiples bits, como ya se evidenció en las pruebas realizadas anteriormente. Finalmente, aunque puede detectar algunos errores de múltiples bits, puede no detectar todos, y los bits redundantes añadidos aumentan el tamaño total de los datos, lo que puede ser problemático en situaciones con limitaciones de ancho de banda (GeeksforGeeks, 2024).

Fletcher

- De manera similar al algoritmo de Hamming, una de las principales fortalezas del algoritmo de Fletcher es su simpleza, su implementación es bastante sencilla. Dentro de sus desventajas, es importante mencionar que los errores en el cálculo del checksum o en la transmisión pueden llevar a falsos positivos o falsos negativos en la verificación de la integridad de los datos. También presenta el mismo error de que múltiples errores pueden llegar a pasar desapercibidos y aunque sí llega a detectar errores es incapaz de modificar el mensaje para corregirlo.

[Parte 5] Referencias

GeeksforGeeks. (2024, 17 junio). *Hamming Code in Computer Network*. GeeksforGeeks. https://www.geeksforgeeks.org/hamming-code-in-computer-network/

[Parte 6] Anexos

- Enlace a repositorio → https://github.com/Andrea-gt/hamming-fletcher