Laboratorio 2 – Parte 1: Algoritmos de detección

Giovanni Santos – 22523

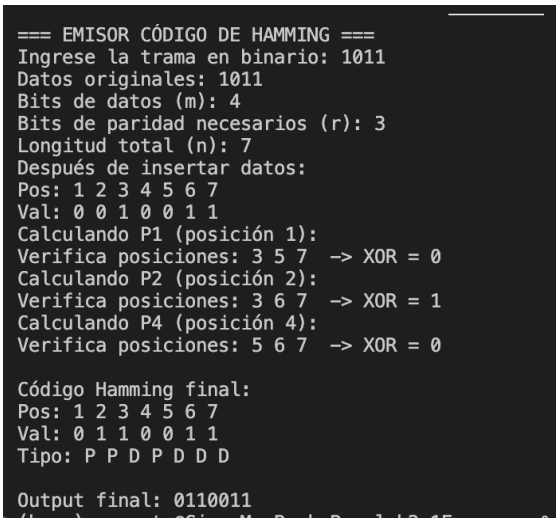
Julio Lemus

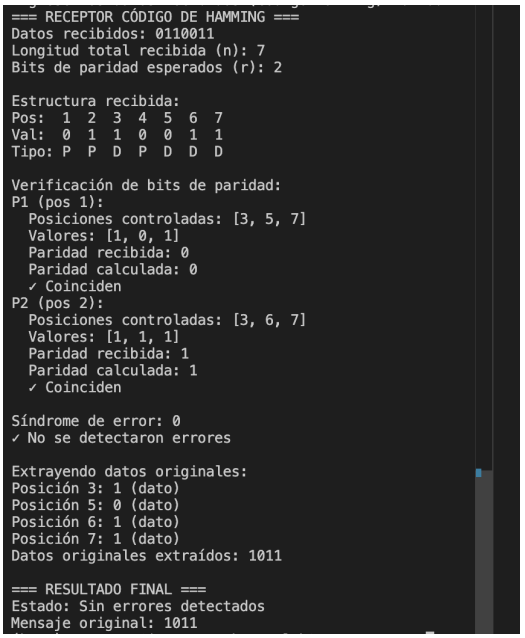
Repositorio: https://github.com/Giosantos23/lab2.1Esquemas.git
Algoritmo de Corrección de errores: Código de Hamming

Pruebas

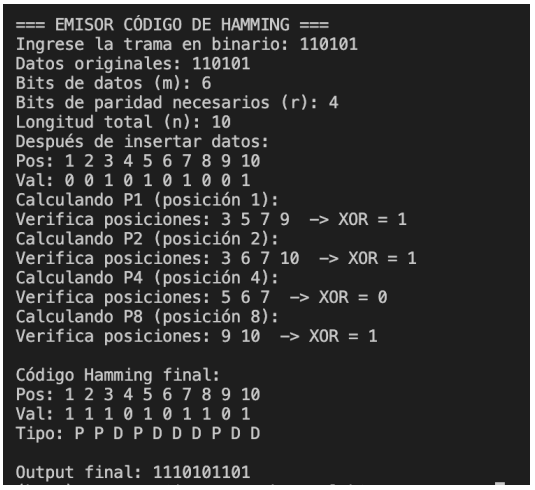
Sin errores:

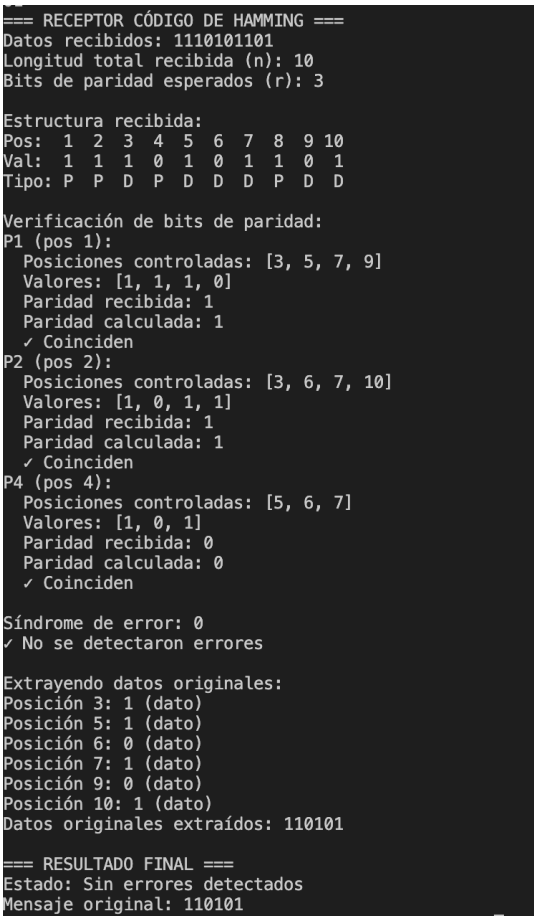
1. Prueba 1, trama: 1011



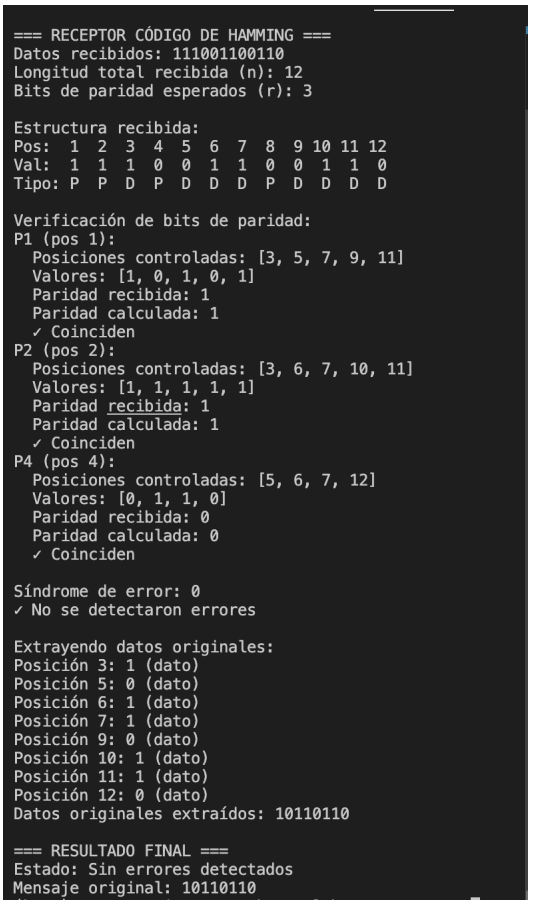


2. Prueba 2, trama: 110101



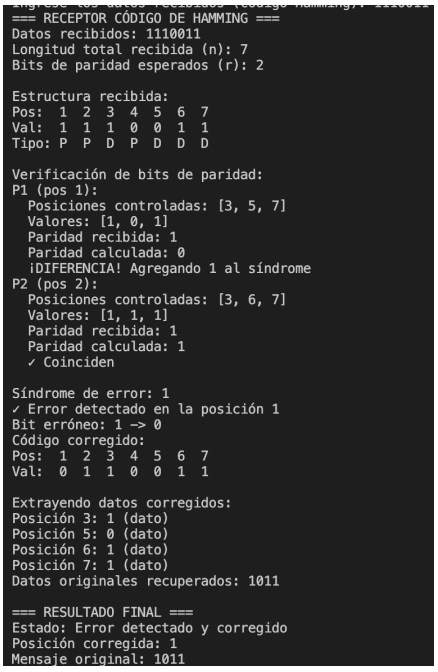


3. Prueba 3, trama: 10110110



Con un error:

1. Prueba 1, trama: 1011



2. Prueba 2, trama: 110101

```
01
=== RECEPTOR CÓDIGO DE HAMMING ===
Datos recibidos: 1010101101
Longitud total recibida (n): 10
Bits de paridad esperados (r): 3

Estructura recibida:
Pos:  1  2  3  4  5  6  7  8  9 10
Val:  1  0  1  0  1  0  1  1  0  1
Tipo: P  P  D  P  D  D  D  P  D  D

Verificación de bits de paridad:
P1 (pos 1):
  Posiciones controladas: [3, 5, 7, 9]
  Valores: [1, 1, 1, 0]
  Paridad recibida: 1
  Paridad calculada: 1
  ✓ Coinciden
P2 (pos 2):
  Posiciones controladas: [3, 6, 7, 10]
  Valores: [1, 0, 1, 1]
  Paridad recibida: 0
  Paridad calculada: 1
  ¡DIFERENCIA! Agregando 2 al síndrome
P4 (pos 4):
  Posiciones controladas: [5, 6, 7]
  Valores: [1, 0, 1]
  Paridad recibida: 0
  Paridad calculada: 0
  ✓ Coinciden

Síndrome de error: 2
✓ Error detectado en la posición 2
Bit erróneo: 0 -> 1
Código corregido:
Pos:  1  2  3  4  5  6  7  8  9 10
Val:  1  1  1  0  1  0  1  1  0  1

Extrayendo datos corregidos:
Posición 3: 1 (dato)
Posición 5: 1 (dato)
Posición 6: 0 (dato)
Posición 7: 1 (dato)
Posición 9: 0 (dato)
Posición 10: 1 (dato)
Datos originales recuperados: 110101

=== RESULTADO FINAL ===
Estado: Error detectado y corregido
Posición corregida: 2
Mensaje original: 110101
```

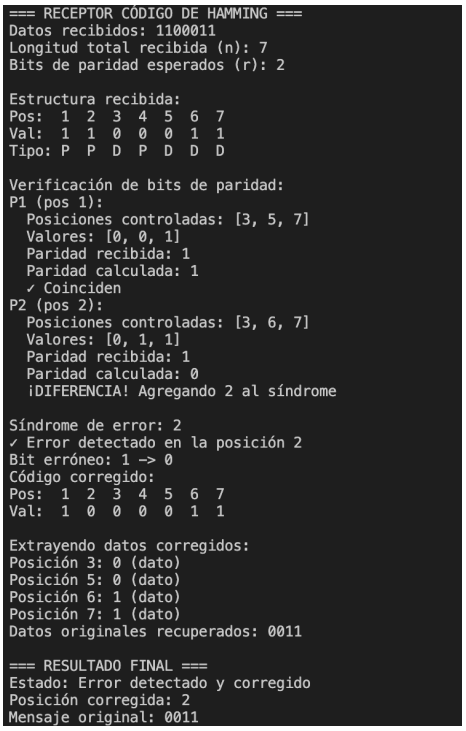
3. Prueba 3, trama: 10110110

```
=== RECEPTOR CÓDIGO DE HAMMING ===
Datos recibidos: 110001100110
Longitud total recibida (n): 12
Bits de paridad esperados (r): 3

Estructura recibida:
Pos:  1  2  3  4  5  6  7  8  9 10 11 12
Val:  1  1  0  0  0  1  1  0  0  1  1  0
Tipo: P  P  D  P  D  D  D  P  D  D  D  D

Verificación de bits de paridad:
P1 (pos 1):
  Posiciones controladas: [3, 5, 7, 9, 11]
  Valores: [0, 0, 1, 0, 1]
  Paridad recibida: 1
  Paridad calculada: 0
  ¡DIFERENCIA! Agregando 1 al síndrome
P2 (pos 2):
  Posiciones controladas: [3, 6, 7, 10, 11]
  Valores: [0, 1, 1, 1, 1]
  Paridad recibida: 1
  Paridad calculada: 0
  ¡DIFERENCIA! Agregando 2 al síndrome
P4 (pos 4):
  Posiciones controladas: [5, 6, 7, 12]
  Valores: [0, 1, 1, 0]
  Paridad recibida: 0
  Paridad calculada: 0
  ✓ Coinciden

Síndrome de error: 3
✓ Error detectado en la posición 3
Bit erróneo: 0 -> 1
Código corregido:
Pos:  1  2  3  4  5  6  7  8  9 10 11 12
Val:  1  1  1  0  0  1  1  0  0  1  1  0

Extrayendo datos corregidos:
Posición 3: 1 (dato)
Posición 5: 0 (dato)
Posición 6: 1 (dato)
Posición 7: 1 (dato)
Posición 9: 0 (dato)
Posición 10: 1 (dato)
Posición 11: 1 (dato)
Posición 12: 0 (dato)
Datos originales recuperados: 10110110

=== RESULTADO FINAL ===
Estado: Error detectado y corregido
Posición corregida: 3
Mensaje original: 10110110
```

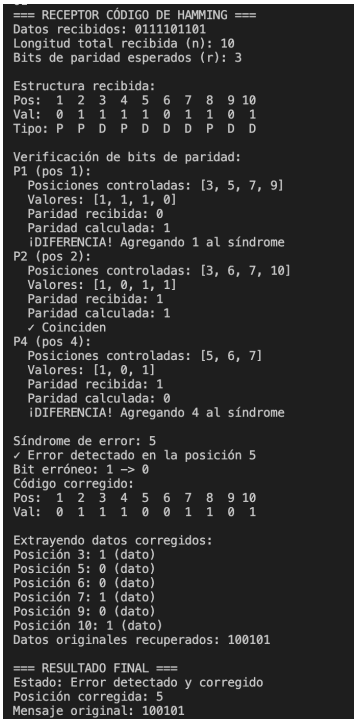
Con 2 o más errores:

Como es Hamming, solo se detectan errores de 1 bit, no de más.

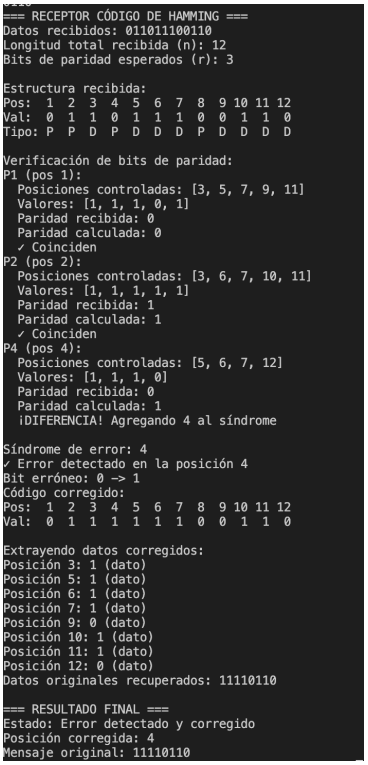
1. Prueba 1, trama: 1011



2. Prueba 2, trama: 110101



3. Prueba 3, trama: 10110110



¿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.

- Se puede manipular al momento de tener un mensaje con múltiples errores. Como el algoritmo de hamming solo maneja un error, muestra como corregido al momento de mandar al receptor.

¿Qué ventajas y desventajas posee cada algoritmo con respecto a los otros dos? Tome en cuenta complejidad, velocidad, redundancia (overhead), etc.

- Hamming tiene una serie de ventajas:
    - Es rápido y eficiente
    - Tiene bajo costo computacional.
    - Tiene una estructura fija
- Sin embargo, tiene una serie de desventajas:
    - Si hay dos o más errores, no puede corregirlos, y hasta puede hacerlo incorrectamente.
    - Tiene overhead significativo.

Conclusiones

- El código de Hamming es una herramienta eficaz para la corrección de errores simples.
- El código de Hamming tiene limitaciones importantes que pueden dificultar la corrección de resultados.