UNIVERSIDAD DEL VALLE DE GUATEMALA

Laboratorio 2 - Primera parte

Edwin de León – 22809
Abby Donis – 22440

Guatemala 18 de agosto, del 2025

Ejemplo de función de algoritmos

Emisor: haming

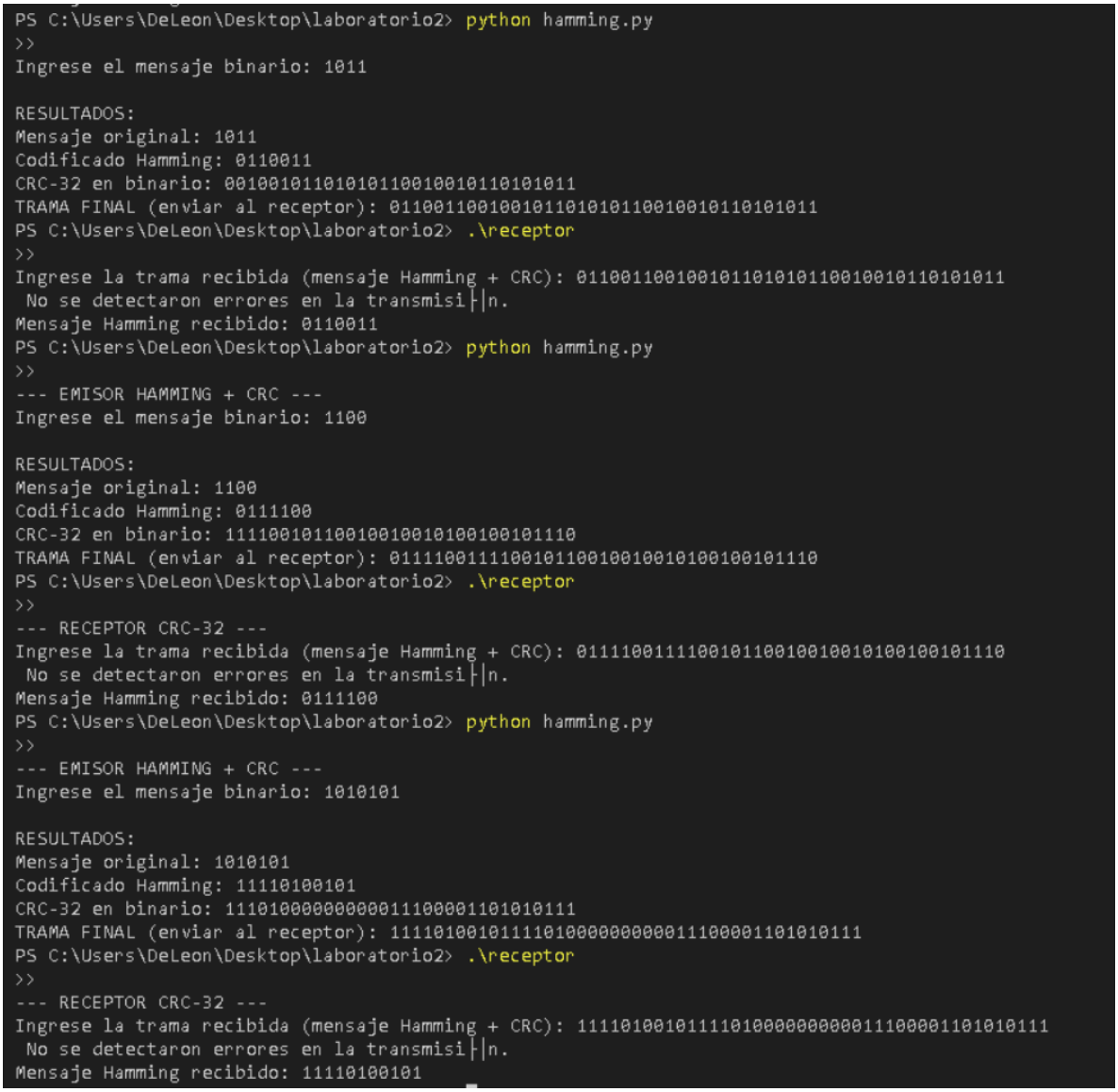
Receptor: CRC-32

```
PS C:\Users\DeLeon\Desktop\laboratorio2>
python hamming.py
>>
--- EMISOR HAMMING + CRC ---
Ingrese el mensaje binario: 1011

--- DETALLES DE Hamming ---
Bit 1: [P] = 0
Bit 2: [P] = 1
Bit 3: [M] = 1
Bit 4: [P] = 0
Bit 5: [M] = 0
Bit 6: [M] = 1
Bit 7: [M] = 1

--- RESULTADOS FINALES ---
Mensaje original: 1011
Mensaje codificado Hamming: 0110011
CRC-32 en binario: 00100101101010110010010110101011
TRAMA FINAL (enviar al receptor): 01100110010010110101011001001011010101 1
Bits de paridad añadidos: 3
PS C:\Users\DeLeon\Desktop\laboratorio2> .\receptor
>>
--- RECEPTOR CRC-32 ---
Ingrese la trama recibida (mensaje Hamming + CRC): 01100110010010110101011001001011010101 1
 No se detectaron errores en la transmisi├│n.
Mensaje Hamming recibido: 0110011
PS C:\Users\DeLeon\Desktop\laboratorio2>
```

Escenario 1: Sin errores



```
PS C:\Users\DeLeon\Desktop\laboratorio2> python hamming.py
>>
Ingrese el mensaje binario: 1011

RESULTADOS:
Mensaje original: 1011
Codificado Hamming: 0110011
CRC-32 en binario: 00100101101010110010010110101011
TRAMA FINAL (enviar al receptor): 01100110010010110101011001001011010101 1
PS C:\Users\DeLeon\Desktop\laboratorio2> .\receptor
>>
Ingrese la trama recibida (mensaje Hamming + CRC): 01100110010010110101011001001011010101 1
 No se detectaron errores en la transmisi├│n.
Mensaje Hamming recibido: 0110011
PS C:\Users\DeLeon\Desktop\laboratorio2> python hamming.py
>>
--- EMISOR HAMMING + CRC ---
Ingrese el mensaje binario: 1100

RESULTADOS:
Mensaje original: 1100
Codificado Hamming: 0111100
CRC-32 en binario: 11110010110010010010100100101110
TRAMA FINAL (enviar al receptor): 01111001111001011001001001010010010111 0
PS C:\Users\DeLeon\Desktop\laboratorio2> .\receptor
>>
--- RECEPTOR CRC-32 ---
Ingrese la trama recibida (mensaje Hamming + CRC): 01111001111001011001001001010010010111 0
 No se detectaron errores en la transmisi├│n.
Mensaje Hamming recibido: 0111100
PS C:\Users\DeLeon\Desktop\laboratorio2> python hamming.py
>>
--- EMISOR HAMMING + CRC ---
Ingrese el mensaje binario: 1010101

RESULTADOS:
Mensaje original: 1010101
Codificado Hamming: 11110100101
CRC-32 en binario: 11101000000000011100001101010111
TRAMA FINAL (enviar al receptor): 11110100101111010000000001110000110101011 1
PS C:\Users\DeLeon\Desktop\laboratorio2> .\receptor
>>
--- RECEPTOR CRC-32 ---
Ingrese la trama recibida (mensaje Hamming + CRC): 11110100101111010000000001110000110101011 1
 No se detectaron errores en la transmisi├│n.
Mensaje Hamming recibido: 11110100101
```

1. **Ingresa un mensaje binario original, por ejemplo:**
   - **1011**
2. **Repite el procedimiento con dos mensajes más de distinta longitud.**

- **1100**
- **1010101**

Escenario 2: Un error (primer bit)

```
PS C:\Users\DeLeon\Desktop\laboratorio2>
python hamming.py
>>
--- EMISOR HAMMING + CRC ---
Ingrese el mensaje binario: 1011

Mensaje original: 1011
Codificado Hamming: 0110011
CRC-32 en binario: 00100101101010110010010110101011
TRAMA FINAL (enviar al receptor): 01100110010010110101011001001011010101 1
PS C:\Users\DeLeon\Desktop\laboratorio2> .\receptor
>>
--- RECEPTOR CRC-32 ---
Ingrese la trama recibida (mensaje Hamming + CRC): 11100110010010110101011001001011010101 1
 Error detectado, la trama se descarta.
PS C:\Users\DeLeon\Desktop\laboratorio2> python hamming.py
>>
--- EMISOR HAMMING + CRC ---
Ingrese el mensaje binario: 1100

RESULTADOS:
Mensaje original: 1100
Codificado Hamming: 0111100
CRC-32 en binario: 11110010110010010010100100101110
TRAMA FINAL (enviar al receptor): 01111001111001011001001001010010010111 0
PS C:\Users\DeLeon\Desktop\laboratorio2> .\receptor
>>
--- RECEPTOR CRC-32 ---
Ingrese la trama recibida (mensaje Hamming + CRC): 11111001111001011001001001010010010111 0
 Error detectado, la trama se descarta.
PS C:\Users\DeLeon\Desktop\laboratorio2> python hamming.py
>>
--- EMISOR HAMMING + CRC ---
Ingrese el mensaje binario: 1010101

RESULTADOS:
Mensaje original: 1010101
Codificado Hamming: 11110100101
CRC-32 en binario: 11101000000000011100001101010111
TRAMA FINAL (enviar al receptor): 11110100101111010000000000111000011010101 11
PS C:\Users\DeLeon\Desktop\laboratorio2> .\receptor
>>
--- RECEPTOR CRC-32 ---
Ingrese la trama recibida (mensaje Hamming + CRC): 01110100101111010000000000111000011010101 11
 Error detectado, la trama se descarta.
PS C:\Users\DeLeon\Desktop\laboratorio2>
```

3. Ingresa un mensaje binario original, por ejemplo:
   - 1011

✓ Modifica un solo bit de la trama. Por ejemplo, cambia el primer bit 0 -> 1:

- 11100110010010110101011001001011010101 1

4. Repite el procedimiento con dos mensajes más de distinta longitud.

- 1100
- 1010101

Escenario 3: Dos o más errores (los primeros 4 bits)

```
python hamming.py
>>
--- EMISOR HAMMING + CRC ---
Ingrese el mensaje binario: 1011

RESULTADOS:
Mensaje original: 1011
Codificado Hamming: 0110011
CRC-32 en binario: 00100101101010110010010110101011
TRAMA FINAL (enviar al receptor): 01100110010010110101011001001011010101 1
PS C:\Users\DeLeon\Desktop\laboratorio2>
PS C:\Users\DeLeon\Desktop\laboratorio2> .\receptor
>>
--- RECEPTOR CRC-32 ---
Ingrese la trama recibida (mensaje Hamming + CRC): 10010110010010110101011001001011010101 1
 Error detectado, la trama se descarta.
PS C:\Users\DeLeon\Desktop\laboratorio2> python hamming.py
>>
--- EMISOR HAMMING + CRC ---
Ingrese el mensaje binario: 1100

RESULTADOS:
Mensaje original: 1100
Codificado Hamming: 0111100
CRC-32 en binario: 11110010110010010010100100101110
TRAMA FINAL (enviar al receptor): 01111001111001011001001001010010010111 0
PS C:\Users\DeLeon\Desktop\laboratorio2>
PS C:\Users\DeLeon\Desktop\laboratorio2> .\receptor
>>
--- RECEPTOR CRC-32 ---
Ingrese la trama recibida (mensaje Hamming + CRC): 10001001111001011001001001010010010111 0
 Error detectado, la trama se descarta.
PS C:\Users\DeLeon\Desktop\laboratorio2> python hamming.py
>>
--- EMISOR HAMMING + CRC ---
Ingrese el mensaje binario: 1010101

RESULTADOS:
Mensaje original: 1010101
Codificado Hamming: 11110100101
CRC-32 en binario: 11101000000000011100001101010111
TRAMA FINAL (enviar al receptor): 11110100101111010000000000111000011010101 11
PS C:\Users\DeLeon\Desktop\laboratorio2> .\receptor
>>
--- RECEPTOR CRC-32 ---
Ingrese la trama recibida (mensaje Hamming + CRC): 00000100101111010000000000111000011010101 11
 Error detectado, la trama se descarta.
PS C:\Users\DeLeon\Desktop\laboratorio2>
```

5. Ingresa un mensaje binario original, por ejemplo:
   - 1011

✓ Modifica un solo bit de la trama. Por ejemplo, cambia el primer 4 bit de 0 -> 1:

- 10010110010010110101011001001011010101 1

6. Repite el procedimiento con dos mensajes más de distinta longitud.

- 1100
- 1010101

¿Es posible manipular los bits de tal forma que el algoritmo seleccionado no sea capaz de detectar el error?

Hamming:

- Sí, es posible. Hamming detecta y corrige errores de 1 bit, pero no siempre detecta errores de 2 o más bits.
- Por ejemplo, si se cambian 2 bits estratégicamente, pueden generar un patrón de paridad válido que pase desapercibido.
- Esto se debe a que el código de Hamming simple (sin paridad extendida) solo garantiza la corrección de un solo bit y la detección de hasta 2 bits, pero no de combinaciones específicas de errores múltiples.

CRC-32:

- Teóricamente, el CRC-32 puede detectar prácticamente todos los errores aleatorios, incluidos los de múltiples bits.
- La probabilidad de que un error aleatorio pase desapercibido es extremadamente baja (1 en 2^32), por lo que para efectos prácticos no es probable que se pueda manipular la trama para que el CRC no detecte errores sin un conocimiento avanzado de la función polinómica usada.

Demostración con tu implementación:

- En tu prueba, cambiaste el primer bit - CRC detecta error
- Cambiaste los primeros 4 bits - CRC detecta error
- Esto coincide con la teoría: CRC-32 es muy robusto frente a errores múltiples aleatorios.
- Para Hamming, si hubieras modificado 2 bits de forma específica dentro de un bloque, podrías generar un error no detectado (aunque tu ejemplo no lo muestra, es posible).

Ventajas y desventajas de cada algoritmo

Algoritmo	Ventajas	Desventajas
Bit de paridad	Muy simple y rápido; requiere mínima redundancia (1 bit extra).	Solo detecta errores impares de 1 bit; no corrige errores; no robusto frente a errores múltiples.
Hamming	Permite detectar y corregir 1 bit; implementación relativamente sencilla; sobrecarga moderada (varios bits de paridad).	No detecta todos los errores múltiples; mayor redundancia que un bit de paridad simple; puede fallar en errores estratégicos de 2 o más bits.
CRC-32	Muy robusto frente a errores múltiples; casi imposible de evadir errores aleatorios; ampliamente usado en comunicaciones y almacenamiento.	Más complejo de implementar; requiere más tiempo de cálculo; alta redundancia (32 bits extra).

Comentario basado en tus pruebas:

- En el escenario sin errores, todos los algoritmos funcionan correctamente.
- Con un error simple (primer bit cambiado), Hamming puede corregirlo si está implementada la corrección, y CRC detecta el error.
- Con errores múltiples (4 bits), Hamming podría fallar, mientras que CRC detecta casi siempre el error.
- Esto demuestra que Hamming es más rápido y ligero, pero menos seguro ante múltiples errores, mientras que CRC es más confiable, pero con mayor costo computacional y redundancia.