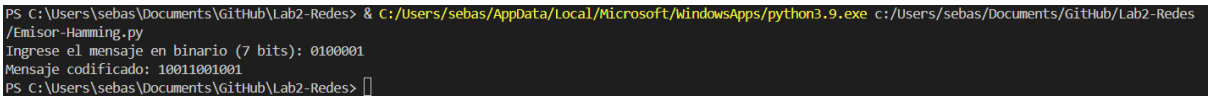
Guatemala 25 de julio de 2024
Sebastian Juarez - 21471
Sebastian Solorzano - 21826

Redes

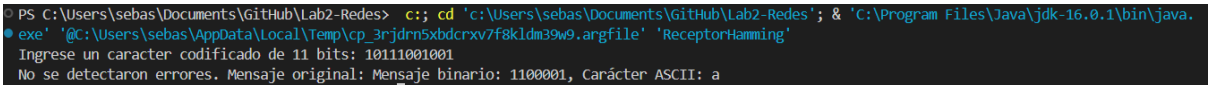
Laboratorio No. 2

Algoritmo de corrección de errores:
Código de Hamming

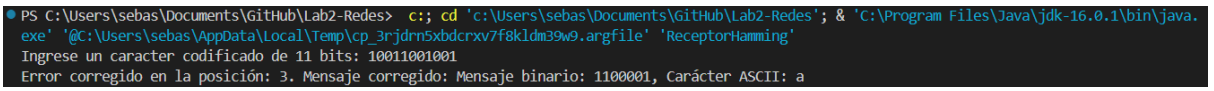
Codificación: Se codificó la letra “a”



- Sin errores



- Un error: Se cambia el tercer bit del mensaje anterior de 1 a 0



- Dos o más errores: El código de Hamming no está diseñado para poder corregir más de un error a la vez sin modificar mucho la estructura del código original. Este, en cambio, sí puede lograr detectar más de un error dependiendo de la complejidad del mensaje.

- ¿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.

  Si es posible manipular los bits para que no se logre detectar el error. Esto, como se había mencionado anteriormente, puede hacerse modificando más de un bit en lugares específicos dentro del mensaje. Esto podría hacer que el corrector no detecte ningún error y mande como correcto. Haciendo una demostración, se puede ver que sucede al introducir los errores.

Se codifica la letra “k” usando el emisor: 1101011 -> 10101010011

Luego se le cambian los valores a los bits 1, 2 y 4 (01111010011) y se pasa al corrector:

```
PS C:\Users\sebas\Documents\GitHub\Lab2-Redes>  c:; cd 'c:\Users\sebas\Documents\GitHub\Lab2-Redes'
 ' '@C:\Users\sebas\AppData\Local\Temp\cp_3rjdrn5xbdcrxv7f8kldm39w9.argfile' 'ReceptorHamming'
Ingrese un caracter codificado de 11 bits: 01111010011
Error corregido en la posición: 6. Mensaje corregido: Mensaje binario: 1111011, Carácter ASCII: {
```

Vemos que al intentar corregirlo, hace algunos cambios y detecta un carácter pero no es el que codificamos principalmente.

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

  Principalmente, la ventaja de usar Hamming es que detecta los errores de una manera rápida. Además es fácil revisar si es que el código tiene algún error al hacer manualmente la corrección del error. La principal desventaja es el condicionamiento a solo 1 bit erróneo y que es muy fácil hacer que el código (si es que no está bien reforzado) pueda dar mensajes erróneos y se pierda algún tipo de información.

Algoritmo de detección de errores:
CRC-32

```
PS C:\Users\50242\Documents\Universidad\CuartoAño\Redes\Lab2-Redes> python -u "c:\Users\50242\Documents\Universidad\CuartoAño\Redes\Lab2
misor-CRC32.py"
Ingrese el mensaje a enviar: Hola redes
Mensaje original: Hola redes
Mensaje en binario: 0100100001101111011011000110000100100000011100100110010101100100011001010111001
CRC32: 01010000110100110111111011000010
Mensaje con CRC32: 0100100001101111011011000110000100100000011100100110010101100100011001010111001101010001101001101111111011000010
```

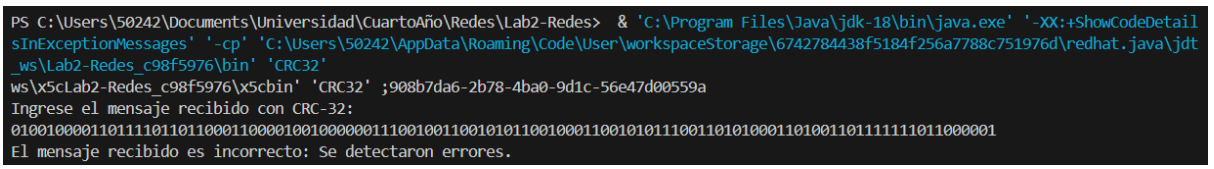
- Sin errores

```
PS C:\Users\50242\Documents\Universidad\CuartoAño\Redes\Lab2-Redes>  & 'C:\Program Files\Java\jdk-18\bin\java.exe' '-XX:+ShowCodeDetail
sInExceptionMessages' '-cp' 'C:\Users\50242\AppData\Roaming\Code\User\workspaceStorage\6742784438f5184f256a7788c751976d\redhat.java\jdt
_ws\Lab2-Redes_c98f5976\bin' 'CRC32'
ws\x5cLab2-Redes_c98f5976\x5cbin' 'CRC32' ;908b7da6-2b78-4ba0-9d1c-56e47d00559a
Ingrese el mensaje recibido con CRC-32:
0100100001101111011011000110000100100000011100100110010101100100011001010111001101010001101001101111111011000010
El mensaje recibido es correcto. Mensaje: Hola redes
```

- Un error: Último bit se cambia 1 por 0

```
PS C:\Users\50242\Documents\Universidad\CuartoAño\Redes\Lab2-Redes>  & 'C:\Program Files\Java\jdk-18\bin\java.exe' '-XX:+ShowCodeDetail
sInExceptionMessages' '-cp' 'C:\Users\50242\AppData\Roaming\Code\User\workspaceStorage\6742784438f5184f256a7788c751976d\redhat.java\jdt
_ws\Lab2-Redes_c98f5976\bin' 'CRC32'
ws\x5cLab2-Redes_c98f5976\x5cbin' 'CRC32' ;908b7da6-2b78-4ba0-9d1c-56e47d00559a
Ingrese el mensaje recibido con CRC-32:
0100100001101111011011000110000100100000011100100110010101100100011001010111001101010001101001101111111011000011
El mensaje recibido es incorrecto: Se detectaron errores.
```

- Dos o más errores: Último bit se cambia 1 por 0 y penúltimo bit se cambia 0 por 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?
  Con el algoritmo CRC-32 hay situaciones en las que ciertos tipos de errores pueden pasar desapercibidos. CRC-32 es muy eficaz para detectar cambios en uno o pocos bits. La probabilidad de que un error aleatorio no sea detectado es extremadamente baja. Sin embargo, si se conoce la estructura del CRC, se podría manipular los bits de tal manera que se mantenga el mismo valor CRC, haciendo que el error no sea detectado, aunque es extremadamente complejo.

- En base a las pruebas que realizó, ¿qué ventajas y desventajas posee cada algoritmo con respecto a los otros dos?
  En la implementación del algoritmo CRC-32, observé que este algoritmo es de alta eficiencia y tiene la capacidad de detección de errores comunes en la transmisión de datos. Una ventaja significativa es su rapidez en el cálculo y el impacto mínimo en el rendimiento, lo cual es importante en aplicaciones donde la velocidad es una prioridad. Además, en muchos protocolos de red garantiza compatibilidad y fiabilidad. Sin embargo, una desventaja notable es que CRC-32 solo puede detectar errores y no corregirlos.

Repositorio:
https://github.com/Sebas021210/Lab2-Redes