

## Laboratorio 2 - Parte 1: Algoritmos de detección

Algoritmos de corrección de errores:

- Pruebas
  - Sin errores:

Entrada Original	Trama Codificada	Salida codificada por Viterbi
1011	11100001	1011
110	110101	110
01101	0011010100	01101

The screenshot shows a VS Code editor with a file named `receptor.py` open. The script implements a Viterbi decoder. The terminal output shows the execution of the script, where the user enters the encoded frame `11100001` and the program outputs the decoded frame `1011`.

```

24 def viterbi_decode(encoded_bits: str) -> str:
33     for bit_in in ['0', '1']:
34         next_state, expected_out = transitions[state][bit_in]
35         distance = hamming_distance(segment, expected_out)
36         total_cost = cost + distance
37
38         if next_state not in new_paths or total_cost < new_paths[next_state][0]:
39             new_paths[next_state] = (total_cost, path + bit_in)
40
41     paths = new_paths
42
43     # Buscar el camino con menor costo
44     best_state = min(paths, key=lambda s: paths[s][0])

```

```

PS C:\Users\WSI\Desktop\info\Escritorio\UG\8vo Semestre\Redes\Lab2\Lab2_Redes> .\Viterbi\emisorViterbi
Ingrese la trama binaria a codificar: 1011
Trama codificada: 11100001
PS C:\Users\WSI\Desktop\info\Escritorio\UG\8vo Semestre\Redes\Lab2\Lab2_Redes>

```

```

PS C:\Users\WSI\Desktop\info\Escritorio\UG\8vo Semestre\Redes\Lab2\Lab2_Redes\Viterbi> python receptor.py
Ingrese la trama codificada de 2 bits por simbolo: 11100001
Mejor estado final: 11
Costo total (errores corregidos): 0
Bits originales decodificados: 1011
PS C:\Users\WSI\Desktop\info\Escritorio\UG\8vo Semestre\Redes\Lab2\Lab2_Redes\Viterbi>

```

```
def viterbi_decode(encoded_bits: str) -> str:
    for bit_in in ['0', '1']:
        next_state, expected_out = transitions[state][bit_in]
        distance = hamming_distance(segment, expected_out)
        total_cost = cost + distance

        if next_state not in new_paths or total_cost < new_paths[next_state][0]:
            new_paths[next_state] = (total_cost, path + bit_in)

    paths = new_paths

    # Buscar el camino con menor costo
    best_state = min(paths, key=lambda s: paths[s][0])
```

Terminal output:

```
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes> ./Viterbi/emisorViterbi
Ingrese la trama binaria a codificar: 1011
Trama codificada: 11100001
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes> ./Viterbi/emisorViterbi
Ingrese la trama binaria a codificar: 110
Trama codificada: 110101
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes>
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes\Viterbi> python receptor.py
Ingrese la trama codificada de 2 bits por símbolo: 11100001
Mejor estado final: 11
Costo total (errores corregidos): 0
Bits originales decodificados: 1011
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes\Viterbi> python receptor.py
Ingrese la trama codificada de 2 bits por símbolo: 110101
Mejor estado final: 01
Costo total (errores corregidos): 0
Bits originales decodificados: 110
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes\Viterbi>
```

```
def viterbi_decode(encoded_bits: str) -> str:
    for bit_in in ['0', '1']:
        next_state, expected_out = transitions[state][bit_in]
        distance = hamming_distance(segment, expected_out)
        total_cost = cost + distance

        if next_state not in new_paths or total_cost < new_paths[next_state][0]:
            new_paths[next_state] = (total_cost, path + bit_in)

    paths = new_paths

    # Buscar el camino con menor costo
    best_state = min(paths, key=lambda s: paths[s][0])
```

Terminal output:

```
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes> ./Viterbi/emisorViterbi
Ingrese la trama binaria a codificar: 1011
Trama codificada: 11100001
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes> ./Viterbi/emisorViterbi
Ingrese la trama binaria a codificar: 110
Trama codificada: 110101
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes> ./Viterbi/emisorViterbi
Ingrese la trama binaria a codificar: 01101
Trama codificada: 0011010100
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes>
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes\Viterbi> python receptor.py
Ingrese la trama codificada de 2 bits por símbolo: 11100001
Mejor estado final: 11
Costo total (errores corregidos): 0
Bits originales decodificados: 1011
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes\Viterbi> python receptor.py
Ingrese la trama codificada de 2 bits por símbolo: 110101
Mejor estado final: 01
Costo total (errores corregidos): 0
Bits originales decodificados: 110
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes\Viterbi> python receptor.py
Ingrese la trama codificada de 2 bits por símbolo: 0011010100
Mejor estado final: 10
Costo total (errores corregidos): 0
Bits originales decodificados: 01101
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes\Viterbi>
```

- 1 Error

Entrada original	Codificado	Bit alterado	Trama con error	Salida decodificada	Costo
1011	11100001	bit 3 (1→0)	<b>11000001</b>	1011	1
110	110101	bit 3 (0→1)	<b>111101</b>	110	1
01101	0011010100	bit 3 (1→0)	0001010100	01101	1

- Mismos 3 entradas en el emisor, pero alteradas para input al receptor

```

def viterbi_decode(encoded_bits: str) -> str:
    for bit_in in ['0', '1']:
        next_state, expected_out = transitions[state][bit_in]
        distance = hamming_distance(segment, expected_out)
        total_cost = cost + distance

        if next_state not in new_paths or total_cost < new_paths[next_state][0]:
            new_paths[next_state] = (total_cost, path + bit_in)

    paths = new_paths

    # Buscar el camino con menor costo
    best_state = min(paths, key=lambda s: paths[s][0])

```

```

PS C:\Users\MSI\Desktop\info\Escritorio\UMG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes> .\Viterbi\emisorViterbi
Ingrese la trama binaria a codificar: 1011
Trama codificada: 11100001
PS C:\Users\MSI\Desktop\info\Escritorio\UMG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes> .\Viterbi\emisorViterbi
Ingrese la trama binaria a codificar: 110
Trama codificada: 110101
PS C:\Users\MSI\Desktop\info\Escritorio\UMG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes> .\Viterbi\emisorViterbi
Ingrese la trama binaria a codificar: 01101
Trama codificada: 0011010100
PS C:\Users\MSI\Desktop\info\Escritorio\UMG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes>

```

```

Lab2_Redes\Viterbi> python receptor.py
Ingrese la trama codificada de 2 bits por símbolo: 11000001
Mejor estado final: 11
Costo total (errores corregidos): 1
Bits originales decodificados: 1011
PS C:\Users\MSI\Desktop\info\Escritorio\UMG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes\Viterbi> python receptor.py
Ingrese la trama codificada de 2 bits por símbolo: 111101
Mejor estado final: 01
Costo total (errores corregidos): 1
Bits originales decodificados: 110
PS C:\Users\MSI\Desktop\info\Escritorio\UMG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes\Viterbi> python receptor.py
Ingrese la trama codificada de 2 bits por símbolo: 0001010100
Mejor estado final: 10
Costo total (errores corregidos): 1
Bits originales decodificados: 01101
PS C:\Users\MSI\Desktop\info\Escritorio\UMG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes\Viterbi>

```

## Preguntas:

- ¿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, con su implementación

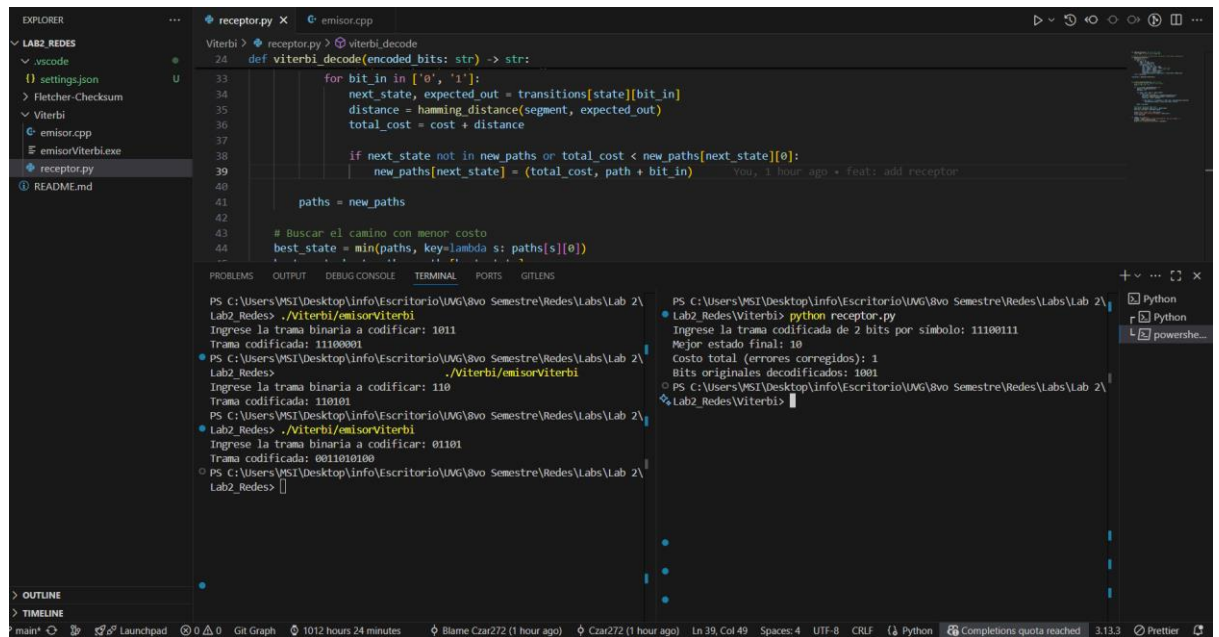
Sí, es posible cambiar algunos bits de forma que el algoritmo de Viterbi no note el error. Esto pasa porque este algoritmo no está hecho para comprobar si el mensaje está completamente bien, sino para tratar de adivinar cuál era el mensaje original más probable basándose en lo que recibió. Si los bits se cambian de cierta manera y el resultado aún "parece correcto" dentro de lo que el algoritmo espera, puede decodificar un mensaje equivocado sin darse cuenta. Esto suele ocurrir cuando el error no se nota fácilmente porque se parece mucho al mensaje original.

Por ejemplo:

Entrada original = 1011;

Entrada codificada = 11100001

Trama codificada errónea: 11100111



The screenshot shows a VS Code editor with a file explorer on the left containing files like `settings.json`, `Fletcher-Checksum`, `Viterbi`, `emisor.cpp`, `emisorViterbi.exe`, `receptor.py`, and `README.md`. The main editor displays the `receptor.py` file with a Viterbi decoding function. The terminal at the bottom shows the execution of the program, where the user enters the original binary sequence '1011', the encoded sequence '11100001', and then a corrupted version '11100111'. The program outputs the final state, total cost (1), and the decoded original bits (1001).

```
def viterbi_decode(encoded_bits: str) -> str:
    24
    33     for bit in ['0', '1']:
    34         next_state, expected_out = transitions[state][bit_in]
    35         distance = hamming_distance(segment, expected_out)
    36         total_cost = cost + distance
    37
    38         if next_state not in new_paths or total_cost < new_paths[next_state][0]:
    39             new_paths[next_state] = (total_cost, path + bit_in)
    40
    41     paths = new_paths
    42
    43     # Buscar el camino con menor costo
    44     best_state = min(paths, key=lambda s: paths[s][0])
```

```
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes> ./Viterbi/emisorViterbi
Ingrese la trama binaria a codificar: 1011
Trama codificada: 11100001
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes> ./Viterbi/emisorViterbi
Ingrese la trama binaria a codificar: 110
Trama codificada: 110101
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes> ./Viterbi/emisorViterbi
Ingrese la trama binaria a codificar: 01101
Trama codificada: 0011010100
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes>
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes\Viterbi> python receptor.py
Ingrese la trama codificada de 2 bits por simbolo: 11100111
Pejor estado final: 10
Costo total (errores corregidos): 1
Bits originales decodificados: 1001
PS C:\Users\MSI\Desktop\info\Escritorio\UAG\8vo Semestre\Redes\Labs\Lab 2\Lab2_Redes\Viterbi>
```

El receptor se equivoca y devuelve:

Costo total (errores corregidos): 1

Bits originales decodificados: 1001