

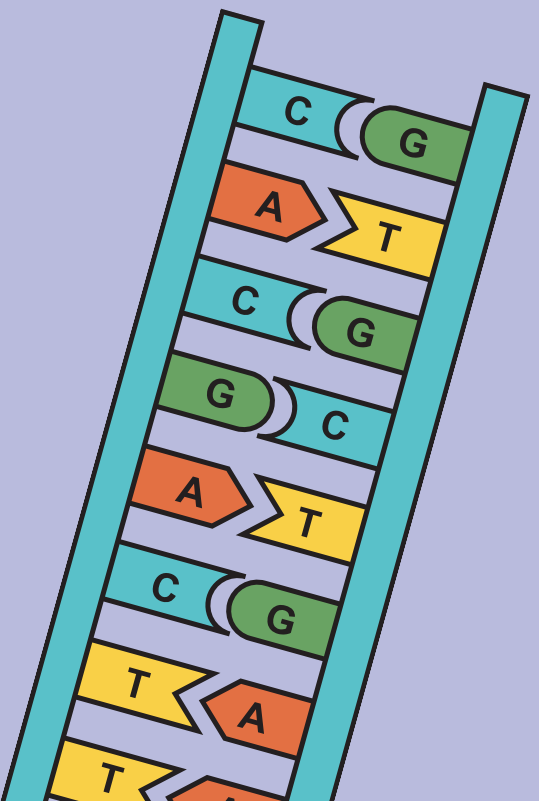
109550151 張昕蓁 110550052 楊沁瑜 110550091 吳承瑀 110550152 成文瀚 110550164 游建峰

# DNA Cryptography

Group 9：不重要吧



# 大綱



1

Introduction

2

流程圖

3

壓縮：Adaptive Huffman Code

4

Encryption & Decryption

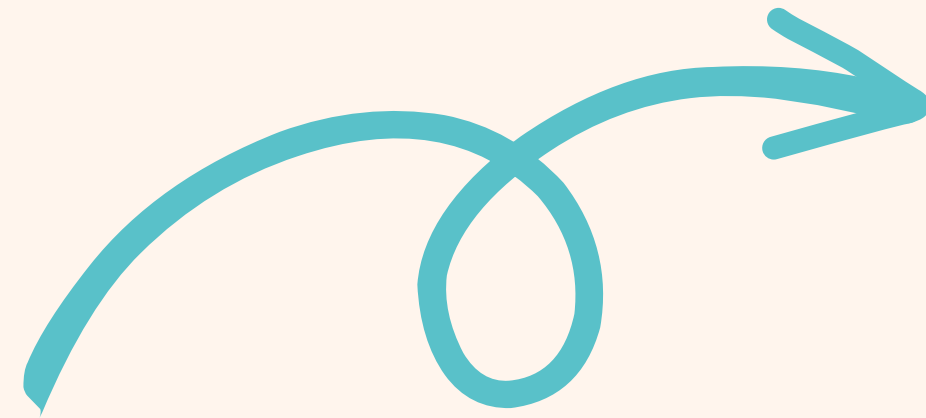
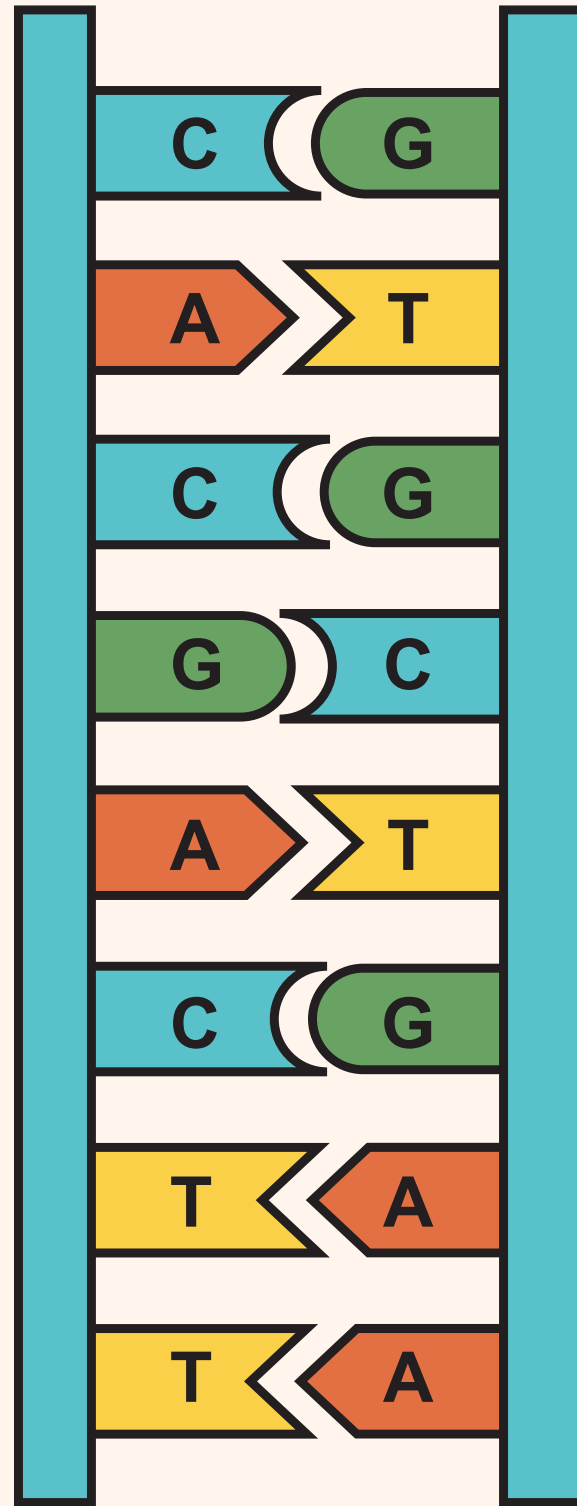
5

總結

6

Demo 影片

# Introduction



**00, 01, 10, 11**



# 流程圖

01

輸入

使用者可以輸入檔案或文字。

02

壓縮

使用 Adaptive Huffman Code。

03

加密

LFSR 生成 OTP key，再進行 XOR。

04

轉成 DNA 序列

將 Binary code 轉成 ATCG。

05

存檔

存成 txt 或 png。



# Adaptive Huffman Code

按照輸入資料流的順序逐個處理符號，並根據符號的頻率動態調整 Huffman tree。

跟 Huffman Code 有什麼不同？

- 不需預處理
- 動態更新
- 更高的壓縮效率
- 較低延遲
- 解壓縮時不需額外資訊

# Adaptive Huffman Code

Stream: aabbbacc

NYT: 為huffman tree 的初始狀態。空節點，新字符會接在NYT的位置上

初始狀態

僅有 NYT node  
weight = 0

NYT 0

初始編碼

a: 00

b: 01

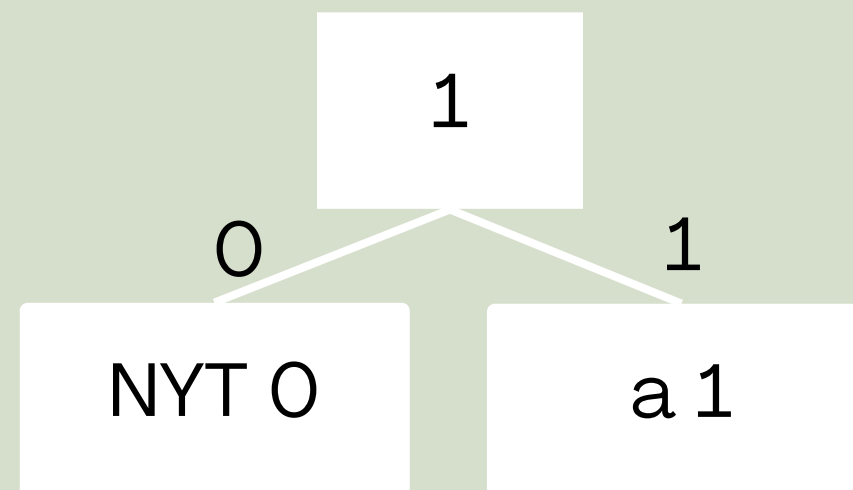
c: 10

輸入新字符a

0: NYT

00: a初始編碼

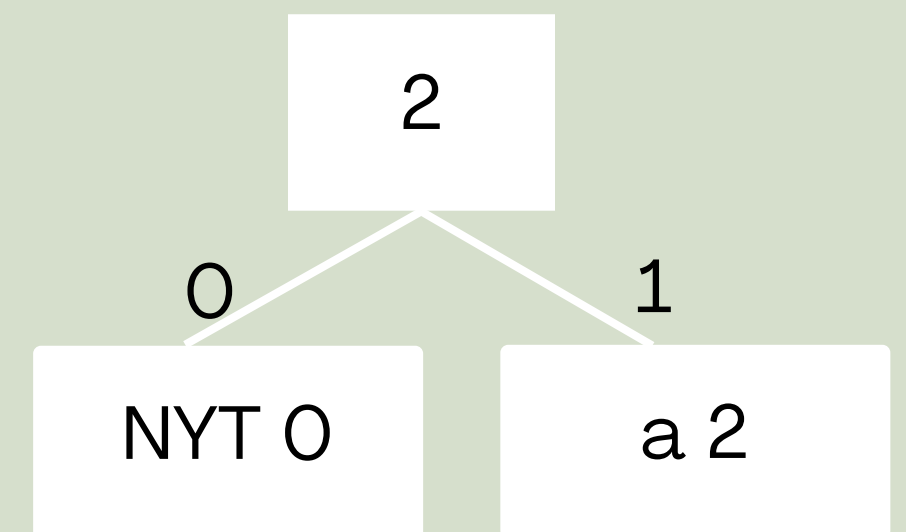
輸出: 000



code: 000

輸入字符a

輸出: 1



code: 0001

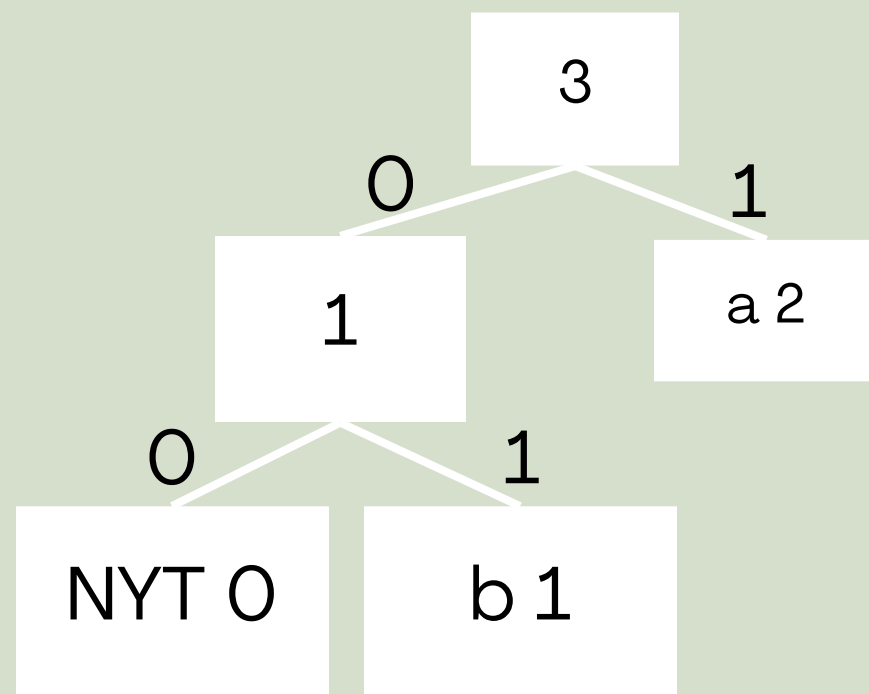
# Adaptive Huffman Code

輸入新字符b

0: NYT

01: b初始編碼

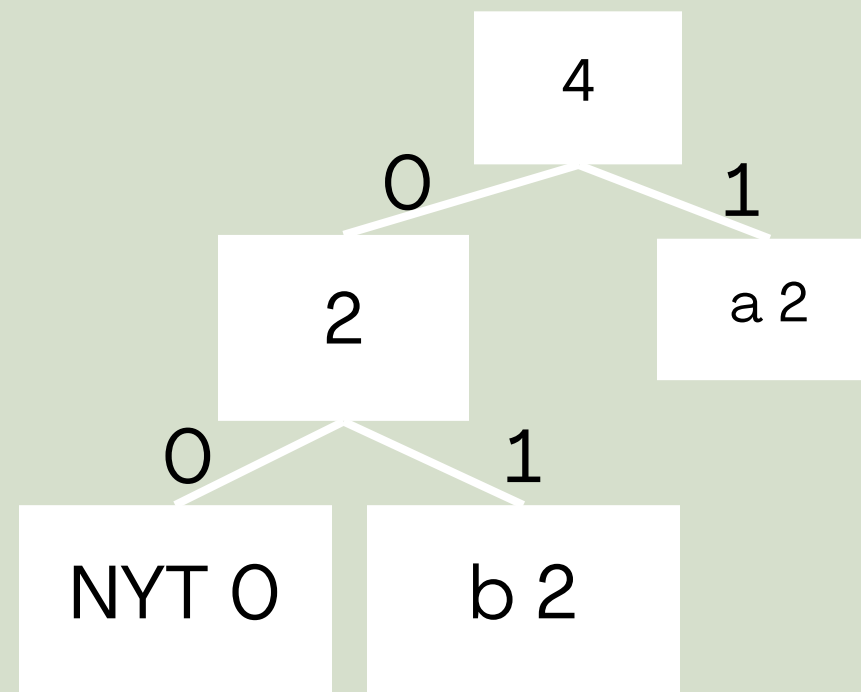
輸出：001



code: 0001001

輸入b

輸出：01

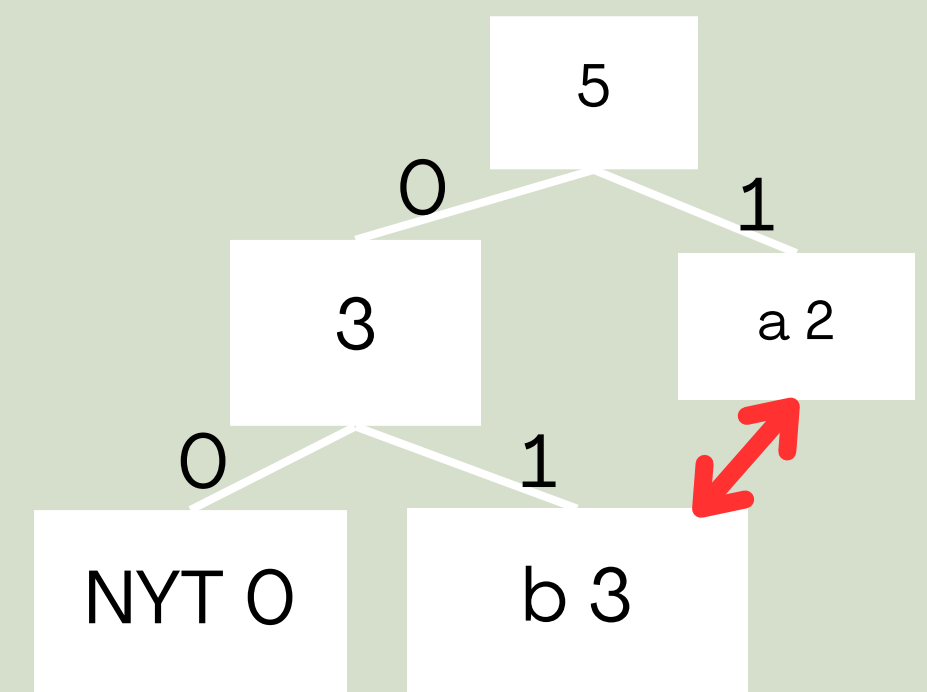


code: 000100101

輸入b

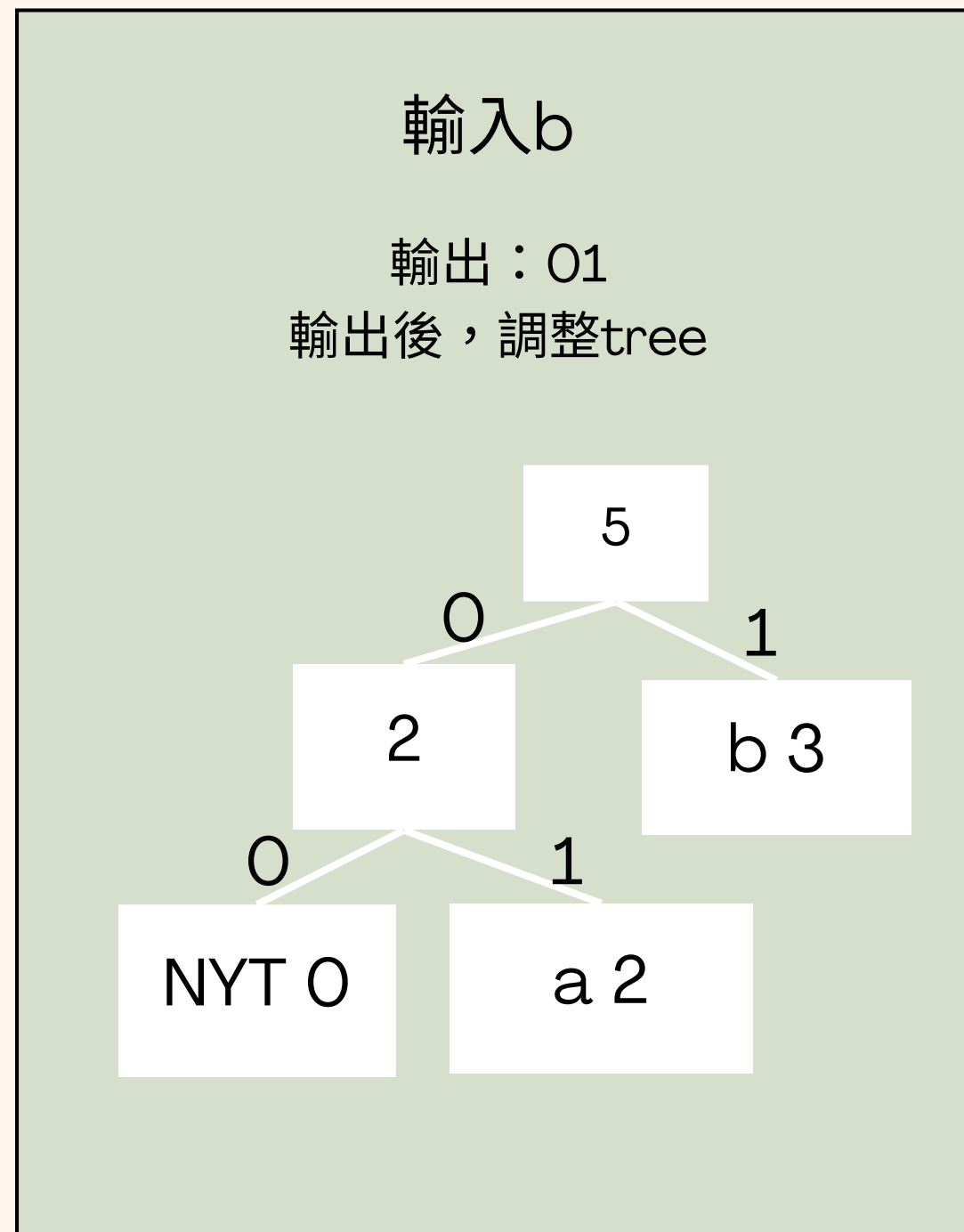
輸出：01

輸出後，調整tree



code: 00010010101

# Adaptive Huffman Code



code: 00010010101

新字符：

輸出 NYT 0 + 字符初始編碼



舊字符：

1. 輸出 root node 到字符路徑上的 0 和 1
2. 更新權重
3. 更新 tree

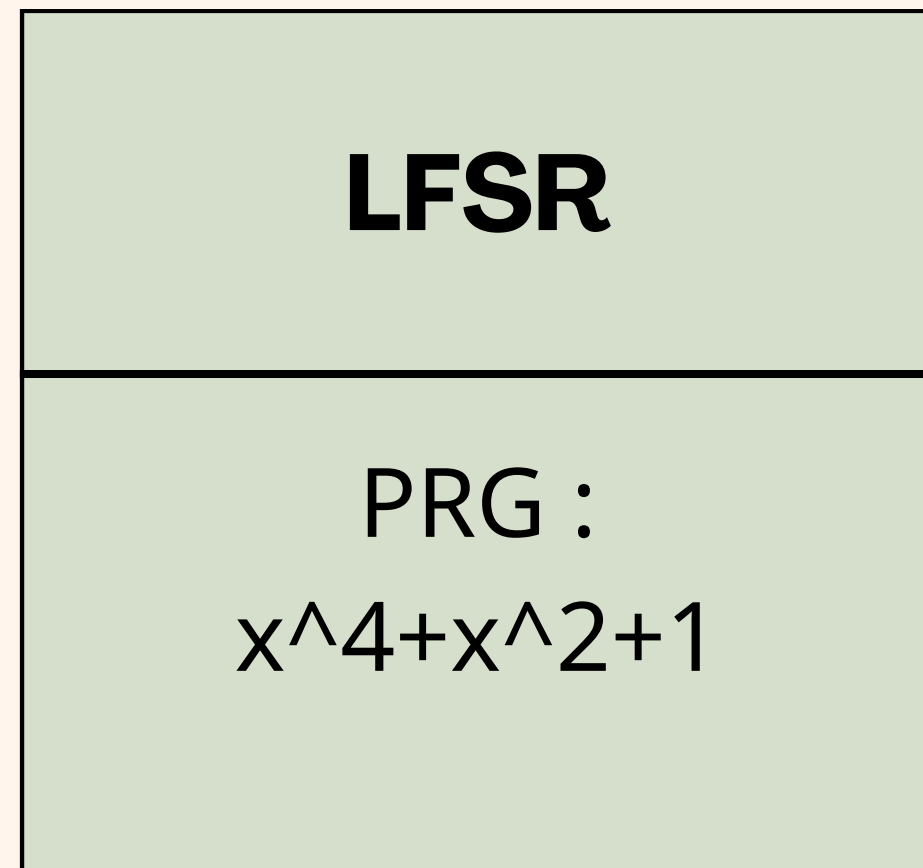
Stream: aabbbacc

Huffman code:

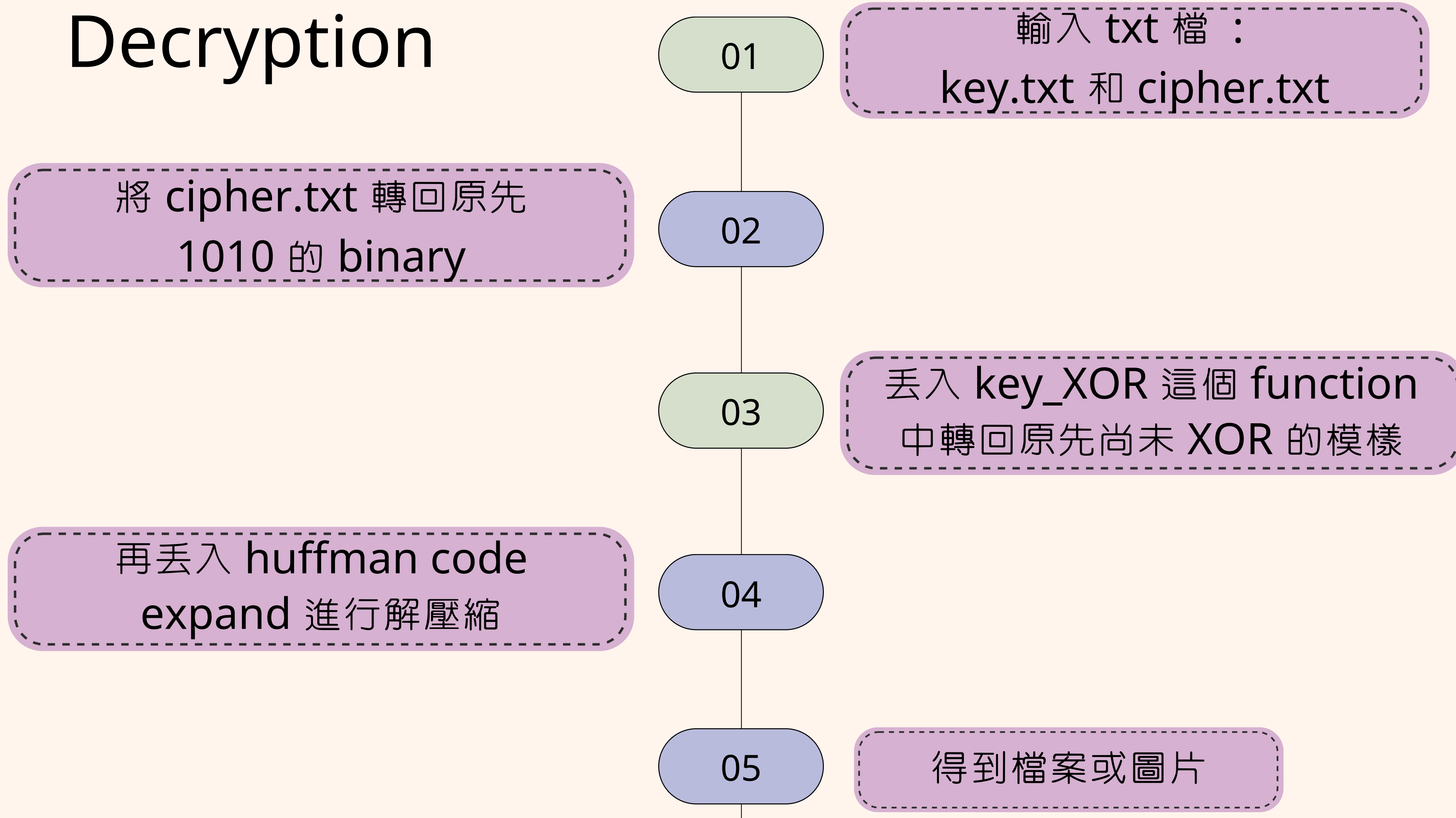
000100101010010101



# Encryption



# Decryption



# Experiment

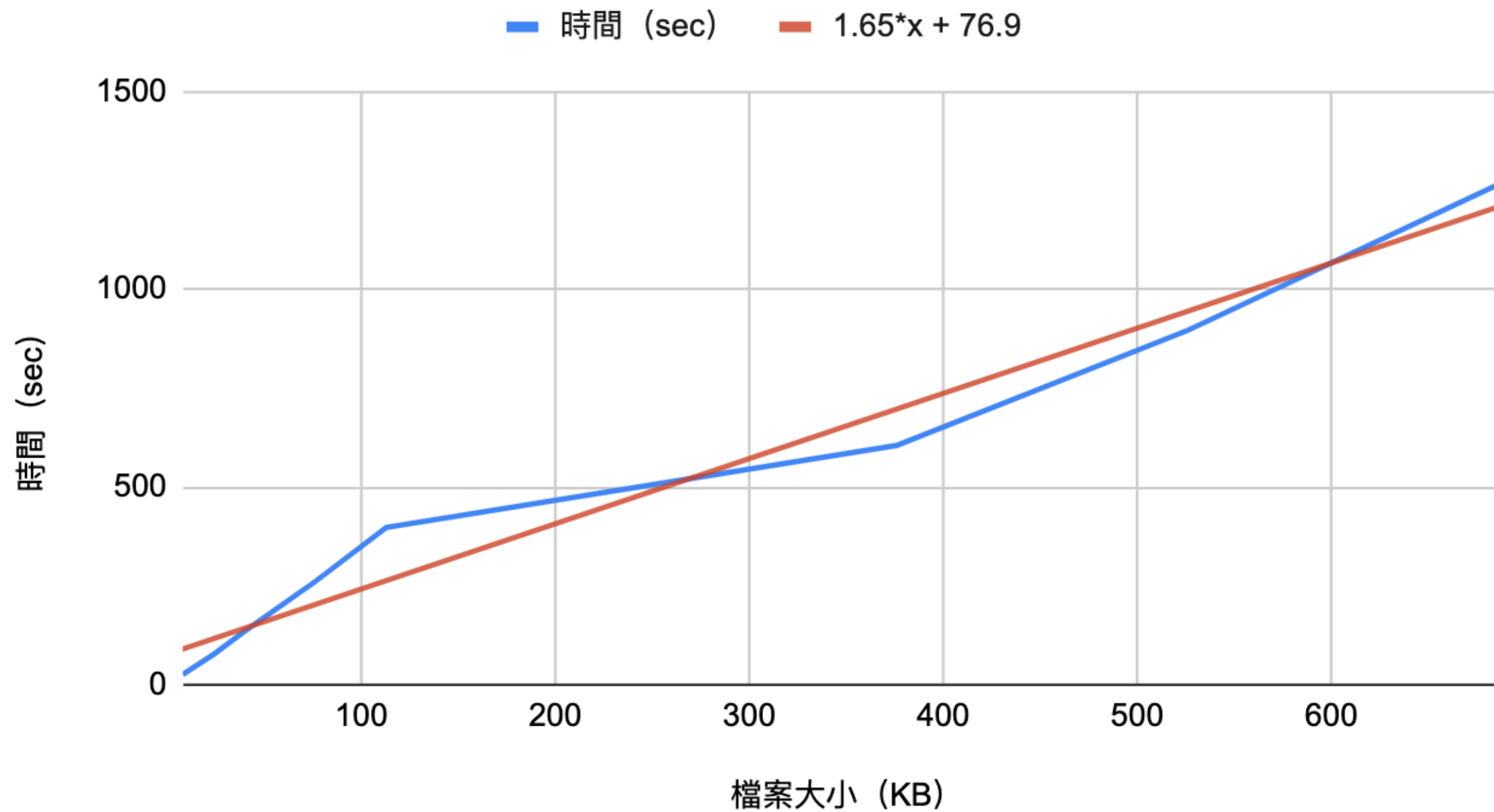
## Analysis

圖片大小 (單位:kb)	加密時間 (單位:秒)	解密時間 (單位:秒)	總共時間 (單位:秒)
8 kb	13	13	26
24 kb	38	39	77
39 kb	64	68	132
76 kb	128	132	260
113 kb	193	205	398
376kb	286	319	605
526kb	394	502	896
693 kb	474	607	1281

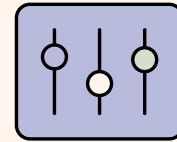
# Experiment

Analysis

時間 (sec) vs. 檔案大小 (KB)

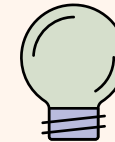


# 總結



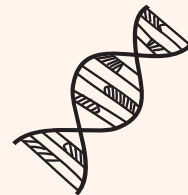
Performance

Linear



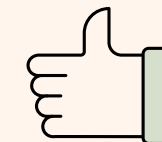
Integral Image

速度變慢



LFSR 生成 OTP key

高安全性、隨機性



Adaptive Huffman

更高的壓縮效率

# Demo

The image shows a VS Code editor window with a Python script named `main.py` open. The script is part of a project named "CRYPTOGRAPHY ENGINEERING FINAL PROJECT". The script imports `DNA` and `utils` modules and contains a `main` function. The `main` function prompts the user to "Encrypt or decrypt a message?" and takes input from the user. If the user enters "1", it prompts for a message to encrypt. If the user enters "2", it prompts for a file to encrypt. The script then prints the encrypted message to the console.

```
1 import DNA
2 import utils
3
4 if __name__ == '__main__':
5
6     EncrypOrDecrypt = input("Encrypt or decrypt a message?\nPress 1 for encryption, press 2 for
7
8     if EncrypOrDecrypt == "1":
9         ans = input("What do you want to encrypt? \nPress 1 for std input, press 2 for file: ")
10        if ans == "1":
11            DNA.TextEncryption()
12        elif ans == "2":
13            file = input("Enter your file name. Ex: 'test.txt' or 'test.jpg' \n")
14            DNA.FileEncryption(file)
15        print("You can find your cipher in 'Cipher.txt' and your key in 'Key.txt")
16
17    if EncrypOrDecrypt == "2":
18        ans = input("What do you want to decrypt? \nPress 1 for .txt file, press 2 for img file: ")
19        if ans == "1":
20            DNA.Decryption("Cipher.txt", "Key.txt", type="text")
21        print("You can find your decoded message in 'PlainTextResult.txt")
```

The output window at the bottom shows the following messages:

```
File "C:\Users\bee00\AppData\Local\Programs\Python\Python39\lib\threading.py", line 361, in notify
if not self._is_owned():
File "C:\Users\bee00\AppData\Local\Programs\Python\Python39\lib\threading.py", line 274, in _is_owned
if self._lock.acquire(False):
KeyboardInterrupt
54%
```



```
19 def FileEncryption(path):
20     Compressor = adaptiveHuffman.AdaptiveHuffman()
21     Compressor.compress(path, "Compression.txt")
22     print("Saving files...")
23
24     message = ""
25     with open("CompressionBinary.txt", "r") as f:
26         message = f.read()
27     key = utils.lfsr(len(message))
28     xor = utils.keyXor(keys=key, text=message)
29     print(xor)
30     output = ''.join(text for text in Encode(xor))
31
32     with open("Key.txt", "w") as f:
33         f.write(key)
34     with open("Cipher.txt", "w") as f:
35         f.write(output)
36
37     os.remove("Compression.txt")
38     os.remove("CompressionBinary.txt")
```



Thank you!

# Reference

[Convert Text to a DNA Sequence with Python](#)

[GitHub Repo](#)