

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM  
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG  
KHOA CÔNG NGHỆ THÔNG TIN**



# **BÁO CÁO GIỮA KỲ MÔN XÁC SUẤT THỐNG KÊ**

*Người hướng dẫn:* **TS NGUYỄN QUỐC BÌNH**

*Người thực hiện:* **PHẠM TRÍ HÙNG**

**Lớp : 20050201**

**Khoá : 24**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM  
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG  
KHOA CÔNG NGHỆ THÔNG TIN**



# **BÁO CÁO GIỮA KỲ XÁC SUẤT THỐNG KÊ**

Người hướng dẫn: **TS NGUYỄN QUỐC BÌNH**

Người thực hiện: **PHẠM TRÍ HÙNG**

Lớp : **20050201**

Khoá : **24**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2022**

## **LỜI CẢM ƠN**

Xin gửi lời cảm ơn đến thầy vì đã cung cấp cho em đủ kiến thức trong quá trình học tập nhằm thực hiện được báo cáo này một cách tốt nhất.

## CÔNG TRÌNH ĐƯỢC HOÀN THÀNH TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của TS Nguyễn Quốc Bình;. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong luận văn còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung luận văn của mình.** Trường đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày tháng năm*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Phạm Trí Hùng*

## TÓM TẮT

Trong báo cáo này em sẽ dùng kiến thức mình đã học được trong học kỳ vừa rồi nhằm giải quyết các vấn đề liên quan đến môn học cấu trúc rời rạc.

## MỤC LỤC

LỜI CẢM ƠN .....	i
TÓM TẮT .....	iii
MỤC LỤC.....	1
DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT .....	2
DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ .....	3
PHẦN 1:.....	4
1.1 Introduction: .....	4
1.1.1 phần câu hỏi: .....	4
1.1.2 phần trả lời: .....	4
1.2 Monoalphabetic substitution cipher .....	6
1.2.1 phần câu hỏi: .....	6
1.2.2 phần trả lời: .....	6
1.3 Frequency Analysis:.....	16
1.3.1 phần câu hỏi: .....	16
1.3.2 phần trả lời: .....	16
1.4 Experiments: .....	18
1.4.1 phần câu hỏi: .....	18
1.4.2 phần trả lời: .....	18

## **DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT**

### **CÁC CHỮ VIẾT TẮT**

## DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

### DANH MỤC HÌNH

Hình 2.1 Thuật toán mã hóa atbash.....	7
Hình 2.2 Thuật toán mã hóa caesar.....	9
Hình 2.3 Thuật toán mã hóa affine.....	11
Hình 2.4 Thuật toán mã hóa mixed_alphabet.....	13
Hình 2.5 Thuật toán mã hóa rot13.....	15
Hình 3.1 Thuật toán frequency analysis.....	17
Hình 4.1 Thuật toán atbash kèm giải thích.....	19
Hình 4.2 Thuật toán caesar kèm giải thích.....	20
Hình 4.3 Thuật toán affine kèm giải thích.....	21
Hình 4.4 Thuật toán rot13 kèm giải thích.....	22
Hình 4.5 Thuật toán mixed_alphabet giải thích.....	23
Hình 4.6 Thuật toán frequency analysis kèm giải thích.....	25
Hình 4.7 Minh họa sử dụng thuật toán atbash để mã hóa chuỗi 50 từ.....	27
Hình 4.8 Minh họa sử dụng thuật toán caesar để mã hóa chuỗi 50 từ.....	28
Hình 4.9 Minh họa sử dụng thuật toán affine để mã hóa chuỗi 50 từ.....	29
Hình 4.10 Minh họa sử dụng thuật toán rot13 để mã hóa chuỗi 50 từ.....	30
Hình 4.11 Minh họa sử dụng thuật toán atbash để mã hóa chuỗi 100 từ.....	32
Hình 4.12 Minh họa sử dụng thuật toán affine để mã hóa chuỗi 1000 từ.....	43



## **PHẦN 1:**

### **1.1 Introduction:**

#### **1.1.1 phần câu hỏi:**

Giới thiệu các khái niệm về mã hóa và giải mã, khóa đối xứng và khóa bất đối xứng, ...

#### **1.1.2 phần trả lời:**

Nhằm phục vụ nhu cầu bảo mật thông tin trong lúc vận chuyển, nhiều người đã tạo ra nhiều phương pháp khác nhau nhằm mã hóa, làm thay đổi nội dung gốc theo một trật tự, một khuôn mẫu nhất định nhằm đảm bảo tính bảo mật cũng như khả năng dịch trở lại thành văn bản gốc tại phía người nhận, việc mã hóa thực chất là một nhóm các thao tác được thực hiện chính xác và tuần tự nhằm chuyển văn bản gốc thành văn bản mã hóa, các phương pháp mã hóa này đa phần đều hoạt động dựa vào khóa, độ phức tạp và quá trình mã hóa phụ thuộc vào loại khóa được chọn, khóa bắt buộc phải được chọn trước quá trình mã hóa và trong đa phần trường hợp nếu không có khóa sẽ rất khó thậm chí không thể chuyển từ văn bản mã hóa sang văn bản gốc.

Việc mã hóa có thể hiểu đơn giản là quá trình biến bản rõ (plaintext) trở thành bản mã (ciphertext), thông thường chỉ có những người được ủy quyền (có được key mã hóa) mới có thể chuyển ngược lại từ bản mã thành bản rõ được, các key được dùng để tiến hành mã hóa thường được tạo ra từ các thuật toán tạo mã ngẫu nhiên.

Hiện tại các khóa dùng để mã hóa chia thành 2 dạng khác nhau đối xứng và bất đối xứng, cụ thể như sau:

Với khóa đối xứng việc mã hóa và giải mã sẽ được thực hiện bằng duy nhất một khóa, văn bản gốc ban đầu còn gọi là bản rõ sẽ được biến đổi thành dạng văn bản mà con người không thể đọc được, còn gọi là bản mã, sau đó khi bản mã gửi đến người nhận có thể sử dụng cùng một loại khóa đã dùng để mã hóa nhằm giải mã và lấy được thông tin.

Cách hoạt động: bên gửi sẽ tiến hành mã hóa bản rõ thành bản mã thông qua một khóa đã được chọn lựa sẵn từ trước và được thông báo cho bên nhận, sau khi bên nhận có được thông tin sẽ tiến hành sử dụng chính xác loại khóa đã được bên gửi dùng để mã hóa nhằm tiến hành giải mã.

Ưu điểm:

Khóa đối xứng có ưu điểm là tốc độ tạo ra khóa nhanh, dễ dàng, dễ sử dụng.

Vì cả hai bên sử dụng cùng một loại khóa nên phương pháp mã hóa này có tốc độ truyền dữ liệu nhanh hơn.

Khuyết điểm:

Do sử dụng chung một loại khóa cho cả mã hóa và giải mã nên khi bên thứ 3 có được khóa thông tin sẽ ngay lập tức bị lộ.

Việc chia sẻ khóa giữa hai bên sẽ gặp khó khăn.

Cần tạo ra rất nhiều khóa nếu muốn giữ được tính bí mật của từng người khi trao đổi trong một nhóm người.

Không có tính ứng dụng cao trong các hệ thống có tính mở do khi đó sẽ có quá nhiều key cần phải được tạo ra nhằm đảm bảo được tính an toàn của thông tin.

Không thể sử dụng cho mục đích xác thực và chống thoái thoát

Trái lại với khóa đối xứng việc mã hóa và giải mã đối với khóa bất đối xứng sẽ cần 1 bộ chìa khóa, một bộ chìa khóa sẽ bao gồm 2 chiếc một chiếc khóa công khai (có thể cung cấp một cách công khai cho tất cả mọi người) và một chiếc khóa cá nhân(chỉ duy nhất một người giữ khóa này), vì các vấn đề bảo mật khóa bí mật khóa bí mật phải được đảm bảo bí mật với tất cả với tất cả mọi người.

Cách hoạt động:

Người gửi sẽ phải gửi thông tin đã được mã hóa bằng khóa công khai của người nhận.

Người nhận sẽ dùng khóa cá nhân của mình để giải mã thông tin.

Ưu điểm:

Khóa công khai không cần được chia sẻ một cách bí mật mà có thể được chia sẻ rộng rãi cho tất cả mọi người.

Khóa cá nhân chỉ có riêng một cá nhân được biết an toàn hơn và có khả năng xác thực được nguồn thông tin

Số lượng khóa cần tạo ra tuy vẫn nhiều nhưng ít hơn so với khóa đối xứng nếu muốn đảm bảo tính riêng tư của từng bên trong quá trình giao tiếp.

Cung cấp khả năng bảo mật dữ liệu cao hơn.

Nhược điểm:

Có khả năng bị tấn công thông qua phương pháp tấn công người đứng giữa.

Việc tạo ra và sử dụng tương đối khó.

## **1.2 Monoalphabetic substitution cipher**

### **1.2.1 phần câu hỏi:**

Giới thiệu về các vấn đề, các ràng buộc, phương pháp và thuật toán nhằm mã hóa.

### **1.2.2 phần trả lời:**

Phương pháp mã hóa monoalphabetic substitution cipher có các đặc điểm sau:

Substitution: các dạng mã hóa xếp vào loại substitution này sẽ tiến hành thay thế ký tự trong văn bản gốc thành các ký tự khác trong quá trình mã hóa và thông qua một key nhất định được người thực hiện lựa chọn từ trước và trong quá trình mã hóa sẽ không làm thay đổi vị trí các ký tự.

Monoalphabetic: dạng mã hóa có tính chất monoalphabetic là dạng mã hóa chỉ dùng duy nhất một loại dạng substitution duy nhất trên toàn bộ bản rõ để tạo ra được bản mã, một ký tự trong bản rõ sẽ luôn trở thành một ký tự trong bản mã.

Thuật toán mã hóa bằng phương pháp atbash:

A screenshot of a code editor with a dark background and light-colored text. The code is a Python function named `at_bash` that takes a `plaintext` string as input. It first creates a list of lowercase ASCII characters, copies it, and reverses it to form a key. The plaintext is converted to lowercase. The function then iterates over each character in the plaintext. If the character is an alphabetic character, it is replaced by the corresponding character in the reversed alphabet (e.g., 'a' becomes 'z', 'b' becomes 'y'). Non-alphabetic characters are left unchanged. Finally, the resulting ciphertext is converted to uppercase and returned. The code is numbered from 1 to 13.

```
1 def at_bash(plaintext):
2     alphabet = list(string.ascii_lowercase)
3     key = alphabet.copy()
4     key.reverse()
5     plaintext = plaintext.lower()
6     print(key)
7     ciphertext = ""
8     for i in plaintext:
9         if i.isalpha():
10             ciphertext += key[ord(i) - 97]
11         else:
12             ciphertext += i
13     return ciphertext.upper()
```

Hình 2.1 Thuật toán mã hóa atbash

Thuật toán atbash là một thuật toán mã hóa vô cùng đơn giản, việc mã hóa đơn giản này là một nhược điểm lớn và nên được cân nhắc khi quyết định sử dụng mã atbash.

Với thuật toán atbash ta sẽ chỉ có duy nhất một lựa chọn duy nhất cho khóa nhằm mã hóa bản rõ thành bản mã, đó là đảo ngược toàn bộ bảng chữ cái ( $a-z \Rightarrow z-a$ ) và do đó loại mật mã này khá rất dễ bị phá, trong trường hợp chỉ cần biết được văn bản được mã hóa bằng phương pháp atbash thì văn bản sẽ được phá ngay lập tức.

Ví dụ về quá trình mã hóa sử dụng mã atbash:

Mã hóa bản rõ “atbash” theo phương thức mã hóa atbash:

Bản rõ: atbash

Bảng chữ cái bản rõ và bản mã:

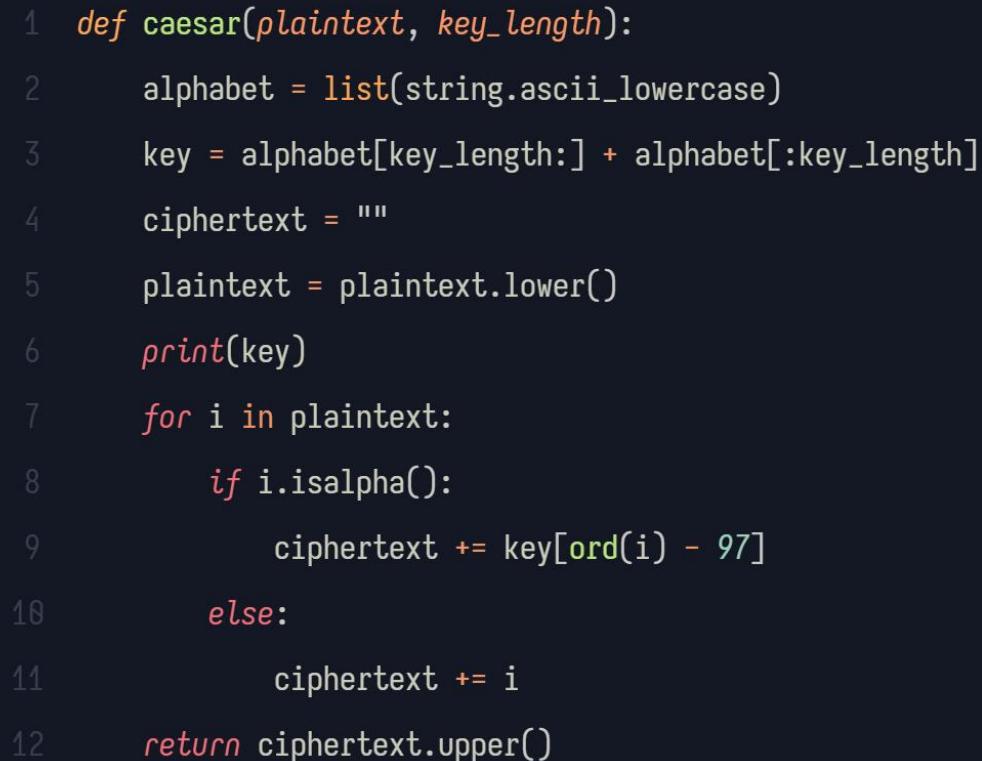
Plaintext Alphabet	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext Alphabet	Z	Y	X	W	V	U	T	S	R	Q	P	O	N	M	L	K	J	I	H	G	F	E	D	C	B	A

Bảng 2.1 bảng chữ cái bản rõ và bản mã của phương pháp mã hóa atbash

Từ bảng trên ta sẽ có được  $a = Z$ ,  $t = G$ ,  $b = Y$ ,  $a = Z$ ,  $s = H$ ,  $h = S$

Kết quả cuối cùng thu được sau quá trình mã hóa bằng phương pháp mã hóa atbash là: ZGYZHS

thuật toán mã hóa theo phương pháp caesar:



```

1  def caesar(plaintext, key_length):
2      alphabet = list(string.ascii_lowercase)
3      key = alphabet[key_length:] + alphabet[:key_length]
4      ciphertext = ""
5      plaintext = plaintext.lower()
6      print(key)
7      for i in plaintext:
8          if i.isalpha():
9              ciphertext += key[ord(i) - 97]
10         else:
11             ciphertext += i
12     return ciphertext.upper()

```

Hình 2.2 Thuật toán mã hóa caesar

Thuật toán mã hóa caesar đã được tạo ra và đặt tên theo Julius Caesar nhằm truyền thông tin trong quân đội, dù là độ phức tạp của thuật toán đã cao hơn atbash nhưng vẫn tương đối dễ phá đặc biệt là khi sử dụng các công nghệ hiện đại, việc tạo ra bản mã trong phương pháp mã hóa caesar tương đối dễ dàng chỉ cần một khóa là một số nguyên biểu thị độ dịch của bảng chữ cái từ đó tiến hành dịch phải đối với khóa dương và dịch trái đối với khóa âm với số ký tự tương ứng của khóa để tạo ra bản mã, và vì tính chất này mà ta biết được rằng ta chỉ có thể tạo ra được tối đa 26 dạng khóa khác nhau, số

lượng khóa có thể tạo ra không nhiều và do đó hoàn toàn có thể bị phá nếu sử dụng thuật toán vét cạn.

Trong ví dụ này ta sẽ dùng bảng mã caesar dịch 3 để tiến hành mã hóa văn bản “caesar”:

Bảng chữ cái bản rõ và bản mã:


Plaintext Alphabet	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Ciphertext Alphabet	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Bảng 2.2 Bảng chữ cái bản rõ và bản mã của phương pháp mã hóa caesar

Từ bảng trên ta sẽ có được c = F, a = D, e = H, s = V a = D và r = U.

Và kết quả thu được bản mã sau khi tiến hành mã hóa bằng mã hóa caesar: FDHVDU.

Thuật toán mã hóa bằng phương pháp affine:



```

1  def affine(plaintext, a, b):
2      plaintext = plaintext.lower()
3      alphabet = list(string.ascii_lowercase)
4      ciphertext = ""
5      key = [chr(((a * (ord(i) - 97) + b) % 26) + 97) for i in alphabet]
6      print(key)
7      for i in plaintext:
8          if i.isalpha():
9              ciphertext += (key[ord(i) - 97])
10         else:
11             ciphertext += i
12     return ciphertext.upper()

```

Hình 2.3 Thuật toán mã hóa affine

Quá trình tạo ra bảng mã của thuật toán mã hóa affine có chút khác biệt so với các dạng mã bên trên khi có sự xuất hiện của phép toán số học trong quá trình mã hóa, cụ thể khi muốn mã hóa văn bản gốc thành văn bản mã hóa ta cần sử dụng công thức  $(a * x + b) \% m$ , trong đó  $a$  và  $b$  chính là hai khóa để tiến hành mã hóa của dạng mã hóa này, theo mã hóa affine có thể tạo ra 312 loại key khác đã là một tiến bộ vượt bậc so với những gì mà atbash và caesar có thể.

Mã hóa văn bản “affine” theo phương thức mã hóa affine:

Văn bản gốc: affine

Do đặc thù loại mã này thao tác nhiều trên số học nên điều đầu tiên ta làm sẽ là chuyển từ các ký tự trong bảng chữ cái latin thành các số nguyên từ 0 đến 25:



Plaintext Alphabet	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
Plaintext Value	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25

Bảng 2.3 Bảng chuyển từ các ký tự trong bảng chữ cái sang số nguyên

Quy đổi các ký tự bên trong văn bản gốc thành các số nguyên dương tương đương và tiến hành thực hiện phép toán để mã hóa, sau đó tra lại vào bảng và tìm ký tự ứng với kết quả vừa tính được:

a sẽ được chuyển thành số 0 và sau khi thực hiện phép tính với  $a = 5$ ,  $b = 8$ , kết quả tính toán có được là 8 ứng với vị trí của chữ i trong bảng chữ cái.


Tiếp tục tiến hành cho các ký tự khác trong chuỗi cho đến khi toàn bộ các ký tự được chuyển đổi, ta có được kết quả như sau

Plaintext	a	f	f	i	n	e
x	0	5	5	8	13	4
$5x+8$	8	33	33	48	73	28
$(5x+8) \bmod 26$	8	7	7	22	21	2
Ciphertext	I	H	H	W	V	C

Bảng 2.4 Bảng chuyển từ chuỗi affine thành bản mã theo phương pháp mã hóa affine

IHHWVC

Thuật toán mã hóa bằng phương pháp mixed-alphabet:



```
1  def mixed_alphabet(plaintext):
2      alphabet = list(string.ascii_letters)
3      plaintext = plaintext.lower()
4      key = alphabet.copy()
5      random.shuffle(key)
6      ciphertext = ""
7      print(key)
8      for i in plaintext:
9          if i.isalpha():
10             ciphertext += key[ord(i) - 97]
11         else:
12             ciphertext += i
13     return ciphertext.upper()
```

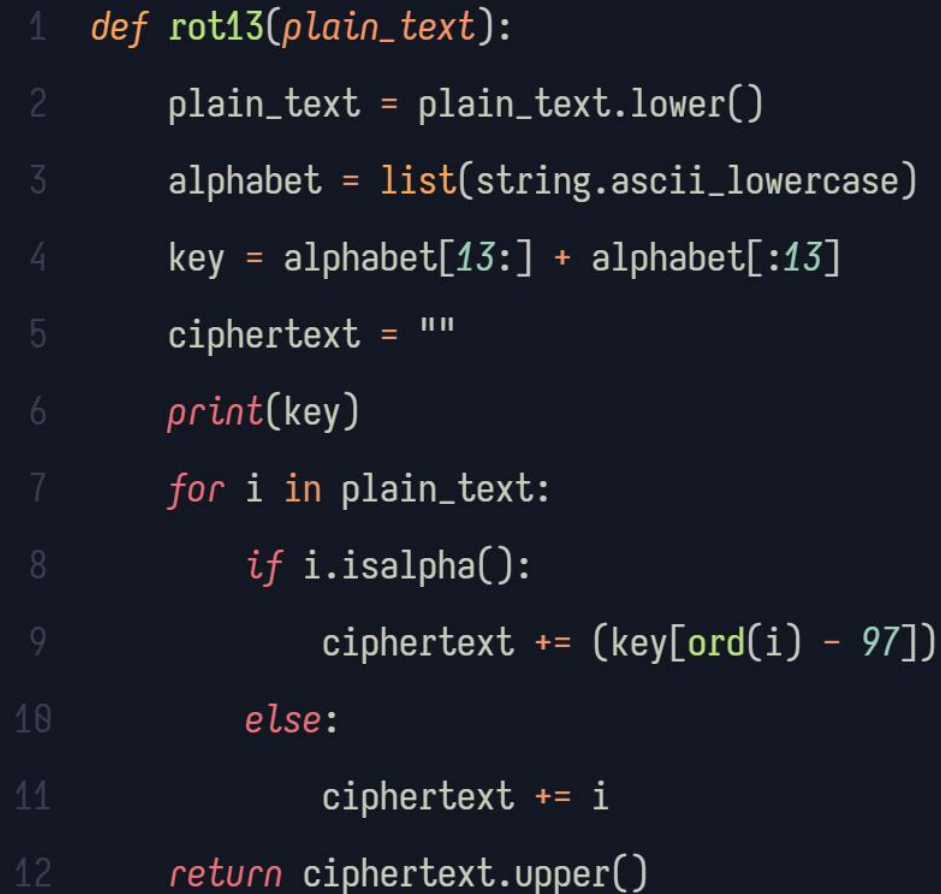
Hình 2.4 Thuật toán mã hóa mixed\_alphabet

Mixed-alphabet là một dạng mã hóa có khả năng tạo ra số lượng key rất lớn vì cách tạo ra key là trộn lẫn các chữ cái bên trong bảng chữ cái không theo bất cứ quy luật

nào để tạo ra được key hoặc có thể dùng một keyword nào đó để tạo bảng mã trong trường hợp keyword được cung cấp t sẽ xếp tất cả các chữ cái có trong keyword đó lên đầu theo thứ tự xuất hiện của từng chữ cái còn lại các chữ cái khác viết theo thứ tự như bình thường:

Do đó ta sẽ có  $26! = 403,291,461,126,605,635,584,000,000$  cách khác nhau để tạo key và với mỗi key đó ta sẽ có một văn bản mã hóa khác nhau, do đó nếu không biết được key chính xác mà chỉ biết được loại mã hóa thì cũng tương đối khó để tiến hành giải mã văn bản được mã hóa bằng loại mã hóa này.

Thuật toán mã hóa bằng phương pháp rot13:

A screenshot of a code editor with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The code is a Python function named `rot13` that takes `plain_text` as an argument. It converts the text to lowercase, creates a key by shifting the alphabet 13 positions, and then iterates through the plain text to build the ciphertext. Alphabetic characters are shifted using a formula involving their ASCII value and the key, while non-alphabetic characters are left unchanged. The final ciphertext is returned in uppercase.

```
1 def rot13(plain_text):
2     plain_text = plain_text.lower()
3     alphabet = list(string.ascii_lowercase)
4     key = alphabet[13:] + alphabet[:13]
5     ciphertext = ""
6     print(key)
7     for i in plain_text:
8         if i.isalpha():
9             ciphertext += (key[ord(i) - 97])
10        else:
11            ciphertext += i
12    return ciphertext.upper()
```

Hình 2.5 Thuật toán mã hóa rot13

Giống với loại mã hóa bằng phương pháp atbash, rot-13 chỉ có một lựa chọn key duy nhất đó là dịch từ bảng chữ cái gốc 13 ký tự từ trái sang phải, và cũng sẽ bị phá ngay lập tức nếu người giải mã nắm được loại mật mã được sử dụng là rot-13.

Mã hóa chuỗi “cipher” với thuật toán mã hóa rot-13:

Chuỗi gốc: “cipher”

Bảng chữ cái của bản rõ và bản mã:

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M

Bảng 2.5 Bảng chữ cái của bản rõ và bản mã của phương pháp mã hóa rot13

Từ bảng trên ta tiến hành mã hóa chuỗi: “cipher” như sau:

c = P, i = V, p = C, h = U, e = R, r = E

sau khi hoàn tất quá trình mã hóa từ bản rõ bằng phương pháp rot13 ta thu được bản mã như sau:

PVCURE

### 1.3 Frequency Analysis:

#### 1.3.1 phần câu hỏi:

Giới thiệu về các vấn đề, các ràng buộc, phương pháp và thuật toán phân tích tần số.

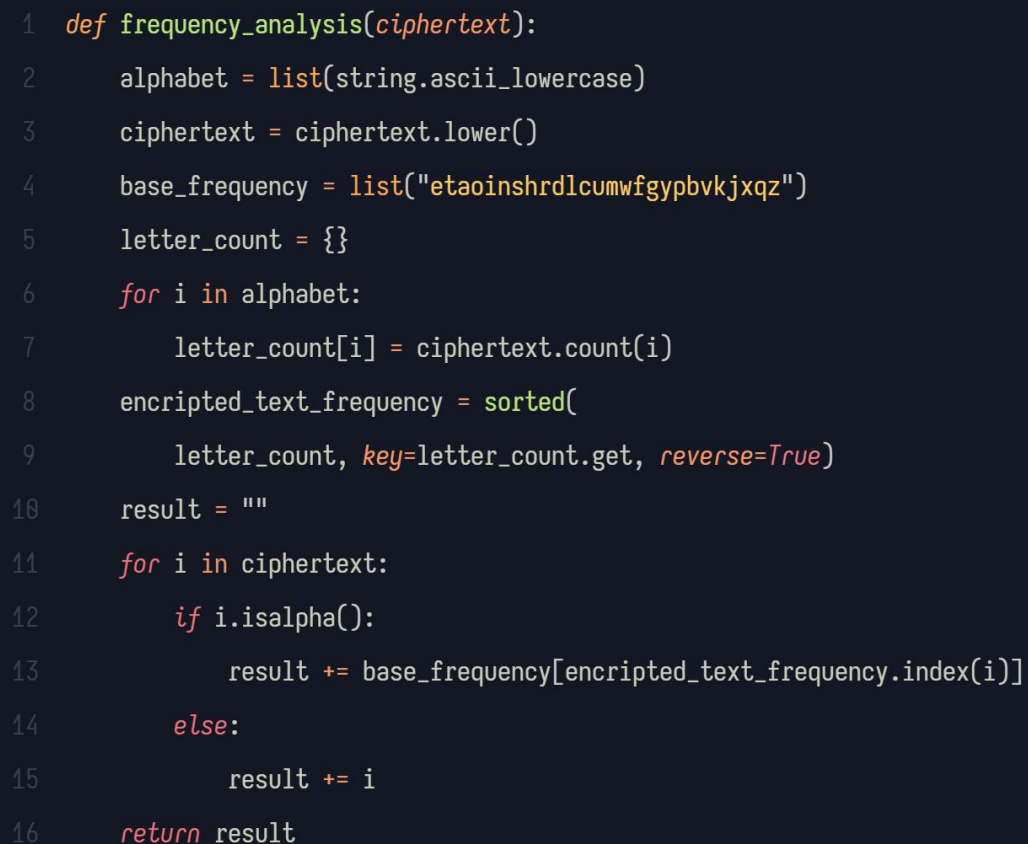
#### 1.3.2 phần trả lời:

Việc phá mã không phải lúc nào cũng cần có sự xuất hiện của khóa, có thể dùng các dạng thuật toán liên quan nhằm tìm được bản rõ hoặc gần với bản rõ mà không cần khóa, ví dụ: thuật toán vét cạn brute force có thể hoạt động nếu đó là một số loại mã nhất định, nhưng trong các thuật toán mã hóa hiện đại hơn số lượng khóa được tạo ra sẽ là quá nhiều và thuật toán vét cạn sẽ là không khả thi, do đó ta cần tìm một phương pháp tốt hơn để tiến hành phá mã, và một trong số đó là phân tích tần số.

Hoàn cảnh: trong tất cả các ngôn ngữ có sử dụng bảng chữ cái chắc chắn luôn luôn có các chữ cái sẽ có tần suất xuất hiện cao hơn các chữ cái còn lại, có thể lấy một ví dụ điển hình là bảng chữ cái tiếng anh, các chữ cái e, t, o, i, n được thống kê là những chữ cái có tần suất xuất hiện nhiều nhất, với tính chất của substitution cipher là chỉ thay đổi

từ một ký tự này sang một ký tự khác và không làm thay đổi vị trí, ta hoàn toàn có thể áp dụng việc phân tích tần số vào để phá mã, cụ thể như sau:

Bắt đầu bằng việc đếm tần suất xuất hiện của toàn bộ các ký tự trong bản mã, ta sẽ thu được một danh sách các chữ cái trong bản mã được sắp xếp theo tần suất xuất hiện, ta sử dụng danh sách ký tự sắp xếp theo tần suất xuất hiện của ngôn ngữ muốn giải mã là đã chuẩn bị từ trước và tiến hành thay thế từng cặp cho giữa hai danh sách này cho nhau nhằm thu được kết quả.



```

1  def frequency_analysis(ciphertext):
2      alphabet = list(string.ascii_lowercase)
3      ciphertext = ciphertext.lower()
4      base_frequency = list("etaoinshrdlcumwfgypbvjkxqz")
5      letter_count = {}
6      for i in alphabet:
7          letter_count[i] = ciphertext.count(i)
8      encrypted_text_frequency = sorted(
9          letter_count, key=letter_count.get, reverse=True)
10     result = ""
11     for i in ciphertext:
12         if i.isalpha():
13             result += base_frequency[encrypted_text_frequency.index(i)]
14         else:
15             result += i
16     return result

```

Hình 3.1 Thuật toán frequency analysis

Theo ý kiến riêng của cá nhân phương pháp này có tương đối nhiều khuyết điểm, đầu tiên việc phân tích tần số này chỉ có ý nghĩa với các đoạn bản mã đủ lớn, cách làm này sẽ hoàn toàn vô nghĩa đối với các mật mã có quá ít nội dung vì không có đủ dữ kiện nhằm đếm được tần xuất xuất hiện và đảm bảo được thay thế đúng ký tự cần thiết

## **1.4 Experiments:**

### **1.4.1 phần câu hỏi:**

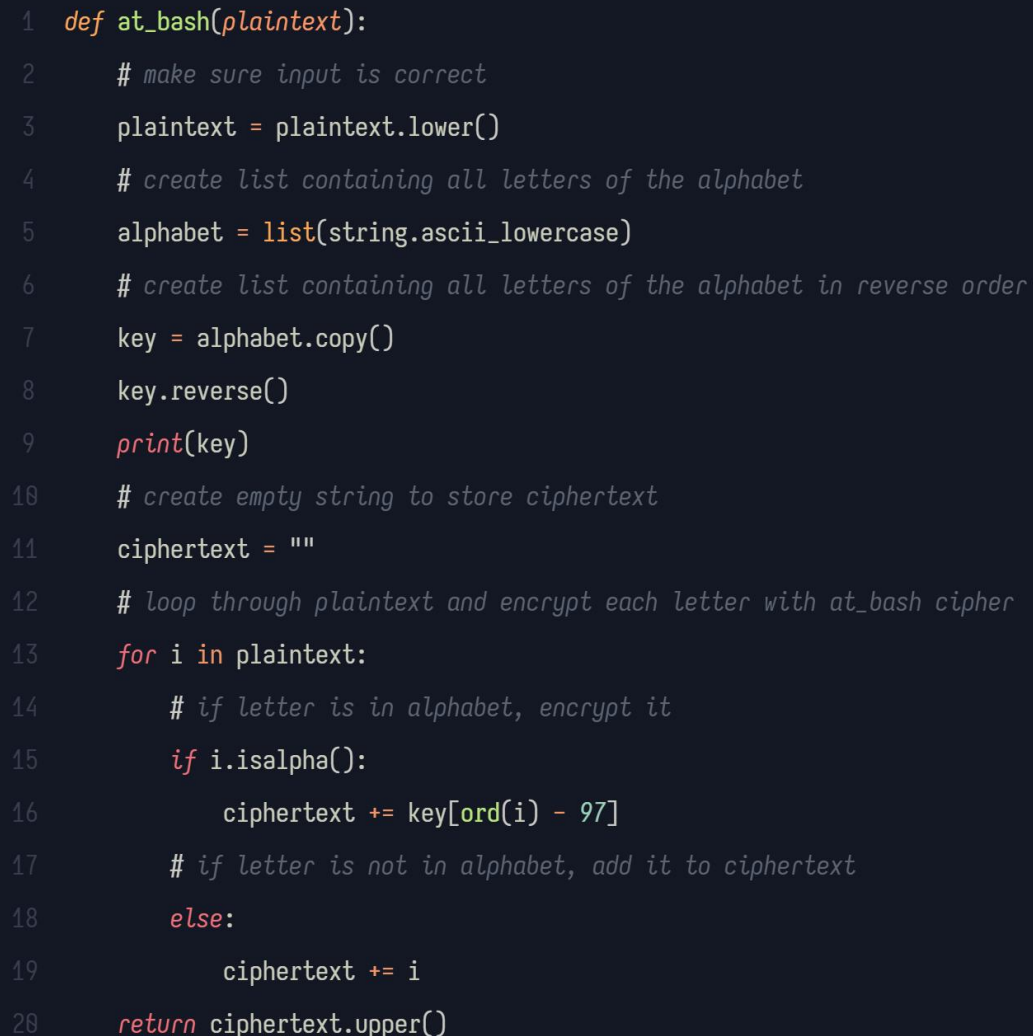
chạy thử các thuật toán trong các phần trước

### **1.4.2 phần trả lời:**

1.4.2a phần giải thích code:

Từng bước thực thi của các loại mã đã được ghi chú ngắn gọn thông qua comment:

Mã atbash:



```

1  def at_bash(plaintext):
2      # make sure input is correct
3      plaintext = plaintext.lower()
4      # create list containing all letters of the alphabet
5      alphabet = list(string.ascii_lowercase)
6      # create list containing all letters of the alphabet in reverse order
7      key = alphabet.copy()
8      key.reverse()
9      print(key)
10     # create empty string to store ciphertext
11     ciphertext = ""
12     # loop through plaintext and encrypt each letter with at_bash cipher
13     for i in plaintext:
14         # if letter is in alphabet, encrypt it
15         if i.isalpha():
16             ciphertext += key[ord(i) - 97]
17         # if letter is not in alphabet, add it to ciphertext
18         else:
19             ciphertext += i
20     return ciphertext.upper()

```

Hình 4.1 Thuật toán atbash kèm giải thích

Biến chuỗi nhận vào thành in thường nhằm đảm bảo tính đúng đắn của thuật toán, tạo ra khóa bằng cách đảo ngược bảng chữ cái latin, duyệt qua các phân tử trong bản rõ và tiến hành thay thế để thu được bản mã

Mã caesar:





```
1  def caesar(plaintext, key_length):
2      # make sure input is correct
3      plaintext = plaintext.lower()
4      # create list containing all letters of the alphabet
5      alphabet = list(string.ascii_lowercase)
6      # create list of shifted alphabet
7      key = alphabet[key_length:] + alphabet[:key_length]
8      # create string to store ciphertext
9      ciphertext = ""
10     print(key)
11     for i in plaintext:
12         # if letter is in alphabet, encrypt it
13         if i.isalpha():
14             ciphertext += key[ord(i) - 97]
15         # if letter is not in alphabet, add it to ciphertext
16         else:
17             ciphertext += i
18     return ciphertext.upper()
```

Hình 4.2 Thuật toán caesar kèm giải thích

Biến chuỗi nhận vào thành in thường nhằm đảm bảo tính đúng đắn của thuật toán, tạo ra khóa bằng cách dịch chuyển bảng chữ cái theo số ký tự đã truyền vào, duyệt qua các phần tử trong bản rõ và tiến hành thay thế để thu được bản mã

Mã affine:

```

1  def affine(plaintext, a, b):
2      # make sure input is correct
3      plaintext = plaintext.lower()
4      # create list containing all letters of the alphabet
5      alphabet = list(string.ascii_lowercase)
6      # create string to store ciphertext
7      ciphertext = ""
8      # create list containing all letters of the alphabet in after the encryption
9      key = [chr(((a * (ord(i) - 97) + b) % 26) + 97) for i in alphabet]
10     print(key)
11     # loop through plaintext and encrypt each letter with affine cipher
12     for i in plaintext:
13         # if letter is in alphabet, encrypt it
14         if i.isalpha():
15             ciphertext += (key[ord(i) - 97])
16         # if letter is not in alphabet, add it to ciphertext
17         else:
18             ciphertext += i
19     return ciphertext.upper()

```

Hình 4.3 Thuật toán affine kèm giải thích

Biến chuỗi nhận vào thành in thường nhằm đảm bảo tính đúng đắn của thuật toán, tạo ra khóa bằng cách lấy vị trí của chữ cái đó(có thể bắt đầu từ 0 hoặc 1) và làm theo

công thức  $(a \cdot x + b) \% 26$ , duyệt qua các phần tử trong bản rõ và tiến hành thay thế để thu được bản mã

Mã rot13:

```

1  def rot13(plain_text):
2      # make sure input is correct
3      plain_text = plain_text.lower()
4      # create list containing all letters of the alphabet
5      alphabet = list(string.ascii_lowercase)
6      # create list of shifted alphabet by 13 letters
7      key = alphabet[13:] + alphabet[:13]
8      # create string to store ciphertext
9      ciphertext = ""
10     print(key)
11     # loop through plaintext and encrypt each letter with rot13 cipher
12     for i in plain_text:
13         # if letter is in alphabet, encrypt it
14         if i.isalpha():
15             ciphertext += (key[ord(i) - 97])
16         # if letter is not in alphabet, add it to ciphertext
17         else:
18             ciphertext += i
19     return ciphertext.upper()

```

Hình 4.4 Thuật toán rot13 kèm giải thích

Biến chuỗi nhận vào thành in thường nhằm đảm bảo tính đúng đắn của thuật toán, tạo ra khóa bằng cách lấy 13 ký tự đầu bảng chữ cái chuyển xuống cuối duyệt qua các phần từ trong bản rõ và tiến hành thay thế để thu được bản mã

Mã `mixed_alphabet`:



```

1  def mixed_alphabet(plaintext):
2      # make sure input is correct
3      plaintext = plaintext.lower()
4      # create list containing all letters of the alphabet
5      alphabet = list(string.ascii_letters)
6      # create list of all letters but in different order
7      key = alphabet.copy()
8      random.shuffle(key)
9      # create empty string to store ciphertext
10     ciphertext = ""
11     print(key)
12     # loop through plaintext and encrypt each letter with mixed alphabet cipher
13     for i in plaintext:
14         # if letter is in alphabet, encrypt it
15         if i.isalpha():
16             ciphertext += key[ord(i) - 97]
17         # if letter is not in alphabet, add it to ciphertext
18         else:
19             ciphertext += i
20     return ciphertext.upper()

```

Hình 4.5 Thuật toán `mixed_alphabet` giải thích

Biến chuỗi nhận vào thành in thường nhằm đảm bảo tính đúng đắn của thuật toán, tạo ra khóa bằng cách trộn lẫn các chữ cái trong bảng chữ cái một cách ngẫu nhiên, duyệt qua các phần tử trong bản rõ và tiến hành thay thế để thu được bản mã

Phân tích tần số:



```

1  def frequency_analysis(ciphertext):
2      # make sure input is correct
3      ciphertext = ciphertext.lower()
4      # create list containing all letters of the alphabet
5      alphabet = list(string.ascii_lowercase)
6      # create list of all letters in alphabet order by frequency of appearance
7      base_frequency = list("etaoinshrdlcumwfgypbvkjxqz")
8      # create a dict to store the frequency of each letter
9      letter_count = {}
10     # loop through ciphertext and count the frequency of each letter
11     for i in alphabet:
12         letter_count[i] = ciphertext.count(i)
13     encrypted_text_frequency = sorted(
14         letter_count, key=letter_count.get, reverse=True)
15     result = ""
16     # loop through the ciphertext and decrypt each letter
17     for i in ciphertext:
18         if i.isalpha():
19             result += base_frequency[encrypted_text_frequency.index(i)]
20         else:
21             result += i
22     return result
23

```

Hình 4.6 Thuật toán frequency analysis kèm giải thích

Đếm tần suất xuất hiện của từng ký tự bên trong bản mã, sắp xếp theo thứ tự giảm dần, tiến hành thay thế theo danh sách có sẵn để thu được kết quả.

#### 1.4.2b Phần demo:

Nội dung file plaintext\_50:

Yesterday, after five subjects at school, I felt so tired and hungry. So, I wanted to go home quickly. Because I was in hurried, I rode my bike so fast. Suddenly, a traffic accident happened, it made all people and me frightened. Fortunately, It didn't happened to me and I felt so lucky.

Nội dung file ciphertext sau khi sử dụng thuật toán atbash để mã hóa:

BVHGVIWZB, ZUGVI UREV HFYQVXGH ZG HXSLLO, R UVOG HL  
GRIVW ZMW SFMTIB. HL, R DZMGVW GL TL SLNV JFRXPOB. YVXZFHV R  
DZH RM SFIIRVW, R ILWV NB YRPV HL UZHG. HFWWVMOB, Z GIZUURX  
ZXXRWVMG SZKKVMVW, RG NZWV ZOO KVLKOV ZMW NV  
UIRTSGVMVW. ULIGFMZGVOB, RG WRWM'G SZKKVMVW GL NV ZMW R  
UVOG HL OFXPB.

Nội dung file decryptedtext sau khi dùng phân tích tần số nhằm phá mã:

cehteriac, alter loke hubxemth at hmwnnd, o ledt hn torei asi wusprc. hn, o vastei  
tn pn wnge juomydc. bemaue o vah os wurroei, o rnie gc boye hn laht. huiiesdc, a  
trallom ammoiest waffesei, ot gaie add fenfde asi ge lropwtesei. Inrtusatedc, ot iois't  
waffesei tn ge asi o ledt hn dumyc.

```

18 import string
17
16
15 def at_bash(plaintext):
14     alphabet = list(string.ascii_lowercase)
13     key = alphabet.copy()
12     key.reverse()
11     plaintext = plaintext.lower()
10     print(key)
9     ciphertext = ""
8     for i in plaintext:
7         if i.isalpha():
6             ciphertext += key[ord(i) - 97]
5         else:
4             ciphertext += i
3     return ciphertext.upper()

[Running] python -u "d:\Code\study\statictis\midtern\atbash.py"
['z', 'y', 'x', 'w', 'v', 'u', 't', 's', 'r', 'q', 'p', 'o', 'n', 'm', 'l', 'k', 'j', 'i', 'h', 'g', 'f', 'e', 'd', 'c', 'b', 'a']

[Done] exited with code=0 in 0.287 seconds

```

Hình 4.7 Minh họa sử dụng thuật toán atbash để mã hóa chuỗi 50 từ

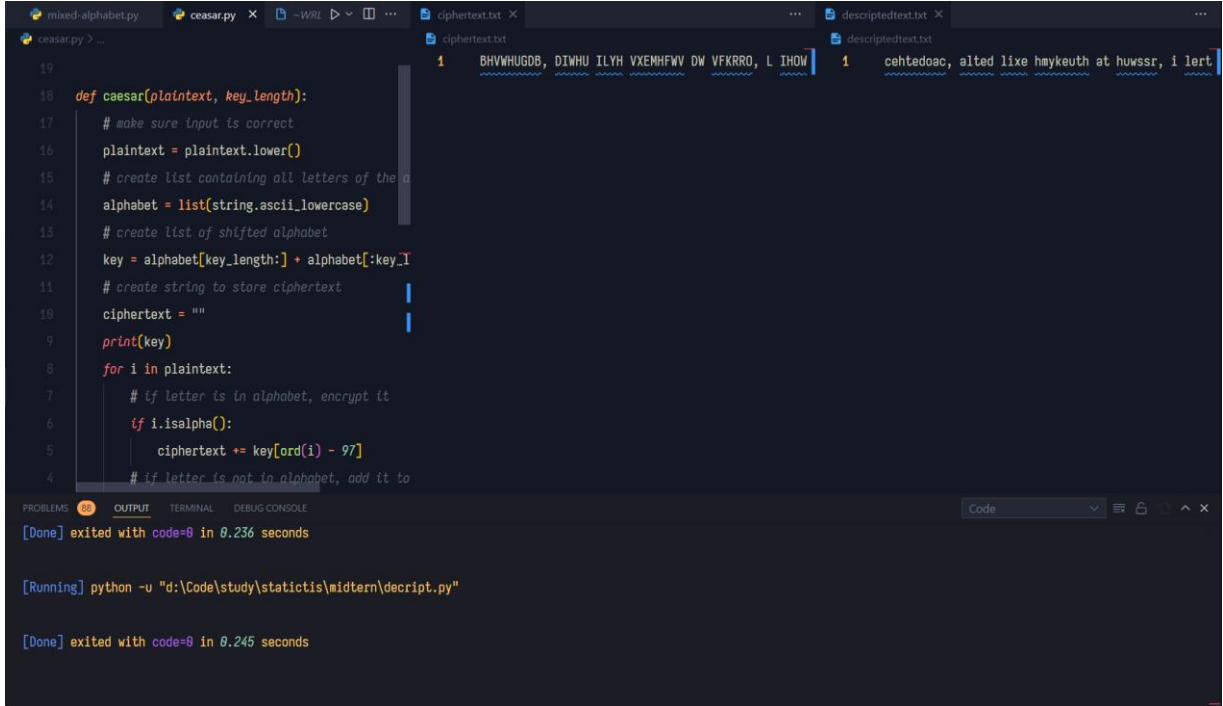
Nội dung file ciphertext sau khi sử dụng thuật toán caesar với độ dịch là 3 để mã hóa:

BHVWHUGDB, DIWHU ILYH VXEMHFWV DW VFKRRO, L IHOW VR WLUHG DQG KXQJUB. VR, L ZDQWHG WR JR KRPH TXLFNOB. EHFDXVH L ZDV LQ KXUULHG, L URGH PB ELNH VR IDVW. VXGGHQOB, D WUDIILF DFFLGHW KDSSHQHG, LW PDGH DOO SHRSOH DQG PH IULJKWHQHG. IRUWXQDWHOB, LW GLGQ'W KDSSHQHG WR PH DQG L IHOW VR OXFNB.

Nội dung file decryptedtext sau khi dùng phân tích tần số nhằm phá mã:

cehtedoac, altered lixe hmykeuth at huwssr, i lert hs tideo ano wmnpc. hs, i vanteo ts ps wsge jmiubrc. yeuamhe i vah in wmdideo, i dsoc gc yibe hs laht. hmooenrc, a tdalli uauioent waffeneo, it gaoe arr fesfre ano ge ldipwteneo. lsdtmnerc, it oion't waffeneo ts ge ano i lert hs rmubc.





```

19
20 def caesar(plaintext, key_length):
21     # make sure input is correct
22     plaintext = plaintext.lower()
23     # create list containing all letters of the alphabet
24     alphabet = list(string.ascii_lowercase)
25     # create list of shifted alphabet
26     key = alphabet[key_length:] + alphabet[:key_length]
27     # create string to store ciphertext
28     ciphertext = ""
29     print(key)
30     for i in plaintext:
31         # if letter is in alphabet, encrypt it
32         if i.isalpha():
33             ciphertext += key[ord(i) - ord('a')]
34         # if letter is not in alphabet, add it to

```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

[Done] exited with code=0 in 0.236 seconds

[Running] python -u "d:\Code\study\staticis\midterm\decrypt.py"

[Done] exited with code=0 in 0.245 seconds

Hình 4.8 Minh họa sử dụng thuật toán caesar để mã hóa chuỗi 50 từ

Nội dung file ciphertext sau khi sử dụng thuật toán affine với  $a = 5$  và  $b = 8$  để mã hóa:

YCUZCPXIY, IHZCP HWJC UENBCSZU IZ USRAAL, W HCLZ UA ZWPCX  
IVX REVMPY. UA, W OIVZCX ZA MA RAQC KEWSGLY. NCSIEUC W OIU WV  
REPPWCX, W PAXC QY NWGC UA HIUZ. UEXXCVLY, I ZPIHHWS ISSWXCXVZ  
RIFFCVCX, WZ QIXC ILL FCAFLC IVX QC HPWMRZCVCX. HAPZEVIZCLY,  
WZ XWXV'Z RIFFCVCX ZA QC IVX W HCLZ UA LESGY.

Nội dung file decryptedtext sau khi sử dụng phân tích tần số để phá mã:  
mehtediam, altd loje hcbkeuth at huwnnr, o lert hn todei asi wcpdpm. hn, o vastei tn pn  
wnge xcouyrm. beuache o vah os wcddeoi, o dnle gm boye hn laht. hciiesrm, a tdallou  
auuoiest waffesei, ot gaie arr fenfre asi ge ldopwtesei. Indtcsaterm, ot iois't waffesei tn  
ge asi o lert hn rcuym.

```

1  import string
2
3  def affine(plaintext, a, b):
4      plaintext = plaintext.lower()
5      alphabet = list(string.ascii_lowercase)
6      ciphertext = ""
7
8      key = [chr((a * (ord(i) - 97) + b) % 26) + 97)
9      print(key)
10
11  for i in plaintext:
12      if i.isalpha():
13          ciphertext += (key[ord(i) - 97])
14      else:
15          ciphertext += i
16
17  return ciphertext.upper()
18
19
20

```

OUTPUT

```

[Running] python -u "d:\Code\study\staticis\midtern\affine.py"
['i', 'n', 's', 'x', 'c', 'h', 'm', 'r', 'w', 'b', 'g', 'l', 'q', 'v', 'a', 'f', 'k', 'p', 'u', 'z', 'e', 'j', 'o', 't', 'y', 'd']
[Done] exited with code=0 in 0.216 seconds

```

Hình 4.9 Minh họa sử dụng thuật toán affine để mã hóa chuỗi 50 từ

Nội dung file ciphertext sau khi sử dụng thuật toán rot13 để mã hóa:

LRFGREQNL, NSGRE SVIR FHOWRPGF NG FPUBBY, V SRYG FB  
GVERQ NAQ UHATEL. FB, V JNAGRQ GB TB UBZR DHVPXYL. ORPNHFR V  
JNF VA UHEEVRQ, V EBQR ZL OVXR FB SNFG. FHQQRAYL, N GENSSVP  
NPPVQRAG UNCCRARQ, VG ZNQR NYY CRBCYR NAQ ZR SEVTUGRARQ.  
SBEGHANGRYL, VG QVQA'G UNCCRARQ GB ZR NAQ V SRYG FB YHPXL.

Nội dung file decryptedtext sau khi sử dụng phân tích tần số để phá mã:

uehteroau, alter lije hcyxemth at hmwssd, i ledt hs tireo ano wcnpru. hs, i vanteo  
ts ps wsge kcimbd. yemache i vah in wcrrieo, i rsoc gu yibe hs laht. hcooendu, a trallim  
ammioent waffeneo, it gaoe add fesfde ano ge lripwteneo. lsrtcnatedu, it oion't waffeneo  
ts ge ano i ledt hs dcmbu.

The screenshot shows a Python IDE with several open files: `rot13.py`, `decrypt.py`, `ciphertext.txt`, and `descriptedtext.txt`. The `rot13.py` file contains the following code:

```

13 print(key)
12 for i in plain_text:
11     if i.isalpha():
10         ciphertext += (key[ord(i) - 97])
9         else:
8             ciphertext += i
7     return ciphertext.upper()
6
5
4 # print(rot13("rot13"))
3 fread = open("plaintext_50.txt", "r")
2 fwrite = open("ciphertext.txt", "w")
1
22 fwrite.write(rot13(fread.read()))
1 fwrite.close()
2

```

The `OUTPUT` panel at the bottom shows the execution results:

```

[Running] python -u "d:\Code\study\statictis\midtern\rot13.py"
['n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z', 'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm']
[Done] exited with code=0 in 0.2 seconds

```

The `descriptedtext.txt` file shows the decrypted text:

```

1 uehteroau, alter lije hcyxemth at hmwssd, i ledt.

```

Hình 4.10 Minh họa sử dụng thuật toán rot13 để mã hóa chuỗi 50 từ

Nội dung file `plaintext_100`:

Abul Kalam Azad Jayanti' is celebrated on 11th November every year. It is the birth anniversary of Maulana Abul Kalam Azad. Abul Kalam Azad was born on 11th November, 1888. His real name was Abul Kalam Ghulam Muhiyuddin. His father's name was Maulana Khairuddin, who lived in Bengal. His mother's name was Alia, who was an Arabian. Maulana Azad was a journalist, author, poet and philosopher. He started to publish the newspaper 'Al-Hilal' in the year 1912. Al-Hilal played an important role in forging Hindu-Muslim unity. He wrote many works, reinterpreting the holy Quran. He played an active role in the freedom movement of India. He became the first Education Minister of independent India. Maulana Azad died on 22nd February, 1958. For his invaluable contribution to the nation, he was posthumously awarded India's highest civilian honour, Bharat Ratna in 1992. His birth anniversary 11th November is declared as 'National Education Day' by the Government of India.

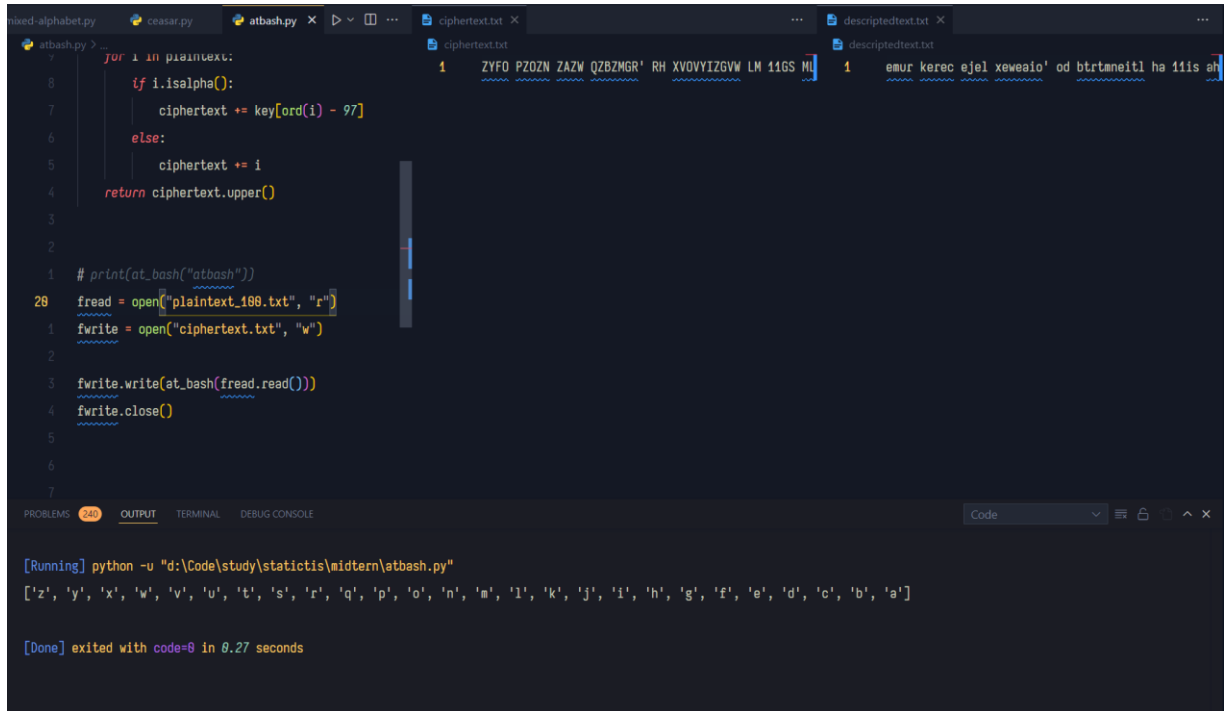
Nội dung file ciphertext sau khi sử dụng thuật toán atbash để mã hóa:

ZYFO PZOZN ZAZW QZBZMGR' RH XVOVYIZGVW LM 11GS  
MLEVNYVI VEVIB BVZI. RG RH GSV YRIGS ZMMREVIHZIB LU NZFOZMZ  
ZYFO PZOZN ZAZW. ZYFO PZOZN ZAZW DZH YLIM LM 11GS MLEVNYVI,  
1888. SRH IVZO MZNV DZH ZYFO PZOZN TSFOZN NFSRBFWWRM. SRH  
UZGSVI'H MZNV DZH NZFOZMZ PSZRIFWWRM, DSL OREVV RM YVMTZO.  
SRH NLGSVI'H MZNV DZH ZORZ, DSL DZH ZM ZIZYRZM. NZFOZMZ ZAZW  
DZH Z QLFIMZORHG, ZFGSLI, KLVG ZMW KSROLHKSVI. SV HGZIGVW GL  
KFYORHS GSV MVDHKZKVI 'ZO-SROZO' RM GSV BVZI 1912. ZO-SROZO  
KOZBVW ZM RNKLIGZMG ILOV RM ULITRMT SRMWF-NFHORN FMRGB. SV  
DILGV NZMB DLIPH, IVRMGVIVIGRMT GSV SLOB JFIZM. SV KOZBVW ZM  
ZXGREV ILOV RM GSV UIVVWLN NLEVNVMG LU RMWRZ. SV YVXZNV  
GSV URIHG VWFXZGRLM NRMRHGVI LU RMWVKVMWVMG RMWRZ.  
NZFOZMZ ZAZW WRVW LM 22MW UYVIFZIB, 1958. ULI SRH RMEZOFZYOV  
XLMGIRYFGRLM GL GSV MZGRLM, SV DZH KLHGSFNLFHOB ZDZIWW  
RMWRZ'H SRTSVHG XRERORZM SLMLFI, YSZIZG IZGMZ RM 1992. SRH  
YRIGS ZMMREVIHZIB 11GS MLEVNYVI RH WVXOZIVW ZH 'MZGRLMZO  
VWFXZGRLM WZB' YB GSV TLEVIMNVMG LU RMWRZ.

Nội dung file decryptedtext sau khi sử dụng phân tích tần số để phá mã:

emur kerec ejel xeweaio' od btrtmneitl ha 11is ahgtcmtn tgnw wten. oi od ist  
monis eaaogtndenw hp ceureae emur kerec ejel. emur kerec ejel fed mhna ha 11is  
ahgtcmtn, 1888. sod nter aect fed emur kerec vsurec cusowulloa. sod peistn'd aect fed  
ceureae kseonulloa, fsh rogtl oa mtaver. sod chistn'd aect fed eroe, fsh fed ea enemoea.  
ceureae ejel fed e xhunaerodi, euishn, yhti eal ysorhdystn. st dienitl ih yumrods ist  
atfdyeytn 'er-sorer' oa ist wten 1912. er-sorer yrewtl ea ocyhnieai nhrt oa phnvoav soalu-  
cudroc uauiw. st fnhit ceaw fhndk, ntoaitnyntioav ist shrw qunea. st yrewtl ea ebiogt nhrt  
oa ist pnttlhc chgtctai hp oaloe. st mtbect ist pondi tlubeioha coaditn hp oaltaltai oaloe.

ceureae ejel lotl ha 22al ptmnuenw, 1958. phn sod oageruemrt bhainomuioha ih ist aeioha, st fed yhdisuchudrw efenltl oaloe'd sovstdi bogoroea shahun, msenei neiae oa 1992. sod monis eaaogtndenw 11is ahgtcmtn od ltbrentl ed 'aeiohaer tlubeioha lew' mw ist vhgtnactai hp oaloe.



```

atbash.py
1  for i in plaintext:
2      if i.isalpha():
3          ciphertext += key[ord(i) - 97]
4      else:
5          ciphertext += i
6      return ciphertext.upper()
7
8  # print(at_bash("atbash"))
9  fread = open("plaintext_1000.txt", "r")
10 fwrite = open("ciphertext.txt", "w")
11
12 fwrite.write(at_bash(fread.read()))
13 fwrite.close()
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
```

diabetes is the product of obesity (Paxon 27). Obesity is a health condition that is caused by consuming or eating too much energy and fatty foods that contain more calories than what a body can use. Based on this, the extra calories are stored in the body in form of fat are said to be the factor that leads to overweight among people including children. Obesity, like any other disease, is treatable through either physical activities or medicinal pills. It has been noted that doctors use what is known as the body mass index (BMI) to determine whether someones health is at risk or not. This research paper was conducted with an aim of exploring the incidence of obesity in children in the U.S. (Paxon 45). Thesis statement: Obesity has been a burning issue among children in the U.S. for the last four decades. Weight management programs were the only method used by medical practitioners to fight against obesity and this has led to healthier lives among children living with obesity. Obesity is a serious medical condition that has affected the health of many children in the United States since the 1980s to date. Initially, people thought that gaining weight in children was a good indicator of being healthy without conceptualizing the dangers associated with being overweight in children. Studies have shown that obesity is a serious disease that can lead to other health problems that include arthritis, diabetes, heart disease and high blood pressure among obese victims (Paxon 61). Although studies have indicated that children are more prone to obesity than adults are, similar studies have shown that everybody is a victim of obesity because the disease is largely related to eating habits that are common to every individual in the world. Through numerous studies, it is evident that obesity in the U.S. is highly pronounced compared to other nations in the world (Hedley et al 2500). In conjunction with this, many obese victims are children as opposed to adults mainly because of the changes in lifestyles whereby technological advancement has led to the introduction of more luxurious activities. For instance, children between the age of 5 and 12 years are prone to obesity because they spend much of their time watching Television at the expense of physical activities. Numerous studies revealed that being a few pounds overweight is not an

implication of obesity but a person is considered obese when he or she has a body mass index that is higher than 30. On the other hand, many adults become overweight while using medicine to put on weight such as the use of corticosteroids and antidepressants among others (Waters Seidell and Swinburn 52). Recent studies have shown that in modern society, there are numerous incidences of obesity as opposed to the past because people lack physical activities that were practiced by early men and fatty and sugary foods have become more available than they were in the traditional society. Eating foods with lesser energy and indulging in physical practices are the only viable means for fighting obesity (Gortmaker and Dietz 47). The researcher embarked on the use of a survey as an appropriate method in researching obese children in the United States. The survey collected data using a questionnaire with questions related to obesity and weight management among the children being asked. A sample of 50 children was randomly selected with two doctors using the body mass index to determine the health of the sampled children whose age ranged between 5 and 15 years (Freedman et al 1759). From the survey, it was evident that 35 children out of 50 weighed much more than their health, which was an indicator that they were obese. For this survey, it was concluded that most children aged between 5 years and 12 years are affected by obesity early in their lives. Such children spend much of their time watching television programs rather than playing on the playground. They also feed on too fatty and sugary foods something that made them be more prevalent in obesity (Hedley et al 2500). The research accepted the null hypothesis that obesity is more pronounced among children aged between 5-12 years because of lack of physical exercise and consumption of too fatty and sugary foods. The category of children prone to obesity is the age group of children aged between 5 and 12 years. The implication is that they spent much of their time watching television rather than playing on the playground. They also have little thinking capability the fact is caused by unused energy in the body with the energy being stored as fat that amounts to obesity. If 35 children out of 50 are obese, it meant that in every 1000 children in the

United States 700 children suffer from obesity. The category of children prone to obesity is the age group of children aged between 5 and 12 years. The implication is that they spent much of their time watching television rather than playing on the playground. They also have little thinking capability the fact is caused by unused energy in the body with the energy being stored as fat that amounts to obesity. If 35 children out of 50 are obese, it meant that in every 1000 children in the United States 700 children suffer from obesity. The study is interesting in that it highlights the concept of obesity as applied to children in the United States. Watching Television and consumption of fatty and sugary foods are the main causes of obesity among children meaning that if children could be encouraged to participate in physical activities, obesity can easily be reduced gradually and eliminated in the end. Parents should urge children to play more than watch the television because this would enable them to manage their weight. In addition to improving their health, physical activities have been known to be positively associated with improved academic performance. At the national level, the government should introduce physical exercise programs in schools hence making it mandatory for all schools to have playgrounds for children. Therefore, participation in physical activities by American children would slightly reduce the health budget at the national level.

Nội dung file ciphertext sau khi sử dụng thuật toán atbash để mã hóa:

GSV HGFVB VCKOLIVH GSV XLMXVKG LU LYVHRGB ZH VERWVMG  
 ZNLMT XSROWIVM RM GSV F.H. ULI YVGGVI FMWVIHGZMWRMT, GSV  
 ZIGRXOV YVTZM DRGS GSV WVURMRGRML LU LYVHRGB, IVERVDVW  
 GSV VCRHGRMT ORGVIZGFIV, SRTSORTSGVW GSV NVGSLWLOLTB FHVW  
 GL XLMWFXG GSV IVHVZIXS, ZMW ZMZOBVW GSV IVHVZIXS  
 URMWRMTH. OZHGOB, IVXLNNVMWZGRLMH DVIV NZWV DRGS GSV  
 HGFVB VMWRMT RM Z XLMXOFHRLM. LYVHRGB RH Z WRHVZHV  
 XLNNLMOB ZHHLXRZGVW DRGS XSROWIVM RM NLHG XLFMGIRVH RM  
 GSV DLIOW. LYVHRGB NVZMH DVRTSRMT NFXS NLIV GSZM RH SVZOGSB



ULI HLNVL MV. HLN KVLKOV XLMUFHV WRZYVGVH DRGS LYVHRGB.  
 GSVHV GDL HSLFOW MLG YV XLNKZIVW YVXZFHV WRZYVGVH RH GSV  
 KILWFXG LU LYVHRGB (KZCLM 27). LYVHRGB RH Z SVZOGS  
 XLMWRGRLM GSZG RH XZFHVW YB XLMHFNRMT LI VZGRMT GLL NFXS  
 VMVITB ZMW UZGGB ULLWH GSZG XLMGZRM NLIV XZOLIRVH GSZM  
 DSZG Z YLWB XZM FHV. YZHVW LM GSRH, GSV VCGIZ XZOLIRVH ZIV  
 HGLIVW RM GSV YLWB RM ULIN LU UZG ZIV HZRW GL YV GSV UZXGLI  
 GSZG OVZWH GL LEVIDVRTSG ZNLMT KVLKOV RMXOFWRMT  
 XSROWIVM. LYVHRGB, ORPV ZMB LGSVI WRHVZHV, RH GIVZGZYO  
 GSILFTS VRGSVI KSBHRXZO ZXGRERGRVH LI NVWRXRMZO KROOH. RG  
 SZH YVVM MLGVW GSZG WLXGLIH FHV DSZG RH PMLDM ZH GSV YLWB  
 NZHH RMWVC (YNR) GL WVGVINRMV DSVGSVI HLNVL MVH SVZOGS RH  
 ZG IRHP LI MLG. GSRH IVHVZIXS KZKVI DZH XLMWFXGVW DRGS ZM ZRN  
 LU VCKOLIRMT GSV RMXRWVMXV LU LYVHRGB RM XSROWIVM RM GSV  
 F.H. (KZCLM 45). GSVHRH HGZGVNVMG: LYVHRGB SZH YVVM Z YFIMRMT  
 RHHFV ZNLMT XSROWIVM RM GSV F.H. ULI GSV OZHG ULFI WVXZWVH.  
 DVRTSG NZMZTVNVMG KILTIZNH DVIV GSV LMOB NVGSLW FHVW YB  
 NVWRXZO KIZXGRGRLMVIH GL URTSG ZTZRMHG LYVHRGB ZMW GSRH  
 SZH OVW GL SVZOGSRVI OREVIH ZNLMT XSROWIVM ORERMT DRGS  
 LYVHRGB. LYVHRGB RH Z HVIRLFH NVWRXZO XLMWRGRLM GSZG SZH  
 ZUUVXGVW GSV SVZOGS LU NZMB XSROWIVM RM GSV FMRGVW  
 HGZGVH HRMXV GSV 1980H GL WZGV. RMRGRZOOB, KVLKOV GSLFTSG  
 GSZG TZRM RMT DVRTSG RM XSROWIVM DZH Z TLLW RMWRXZGLI LU  
 YVRMT SVZOGSB DRGSLFG XLMXVKGFZORARMT GSV WZMTVIH  
 ZHHLXRZGVW DRGS YVRMT LEVIDVRTSG RM XSROWIVM. HGFWRVH  
 SZEVS HSLDM GSZG LYVHRGB RH Z HVIRLFH WRHVZHV GSZG XZM OVZW  
 GL LGSVI SVZOGS KILYOVNH GSZG RMXOFWV ZIGSIRGRH, WRZYVGVH,

SVZIG WRHVZHV ZMW SRTS YOLLW KIVHHFIV ZNLMT LYVHV ERXGRNH (KZCLM 61). ZOGLSFTS HGFWRVH SZEZ RMWRXZGVW GSZG XSROWIVM ZIV NLIV KILMV GL LYVHRGB GSZM ZWFOGH ZIV, HRNROZI HGFWRVH SZEZ HSLDM GSZG VEVIBYLB RH Z ERXGRN LU LYVHRGB YVXZFHV GSV WRHVZHV RH OZITVOB IVOZGVW GL VZGRMT SZYRGH GSZG ZIV XLNNLM GL VEVIB RMWRERWFZO RM GSV DLIOW. GSILFTS MFNVILFH HGFWRVH, RG RH VERWVMG GSZG LYVHRGB RM GSV F.H. RH SRTSOB KILMLFMXVW XLNKZIVW GL LGSVI MZGRLMH RM GSV DLIOW (SVWOVB VG ZO 2500). RM XLMQFMXGRLM DRGS GSRH, NZMB LYVHV ERXGRNH ZIV XSROWIVM ZH LKKLHVW GL ZWFOGH NZRMOB YVXZFHV LU GSV XSZMTVH RM ORUVHGBOVH DSVIVYB GVXSMLOLTRXZO ZWEZMXVNVMG SZH OVW GL GSV RMGILWFXGRLM LU NLIV OFCFIRLFH ZXGRERGRVH. ULI RMHGZMXV, XSROWIVM YVGDVVM GSV ZTV LU 5 ZMW 12 BVZIH ZIV KILMV GL LYVHRGB YVXZFHV GSVB HKVMW NFXS LU GSVRI GRNV DZGXSRTM GVOVERHRLM ZG GSV VCKVMHV LU KSBHRXZO ZXGRERGRVH. MFNVILFH HGFWRVH IVEVZOVW GSZG YVRMT Z UVD KLFMWH LEVIDVRTSG RH MLG ZM RNKORXZGRLM LU LYVHRGB YFG Z KVIHLM RH XLMHRWVIVW LYVHV DSVM SV LI HSV SZH Z YLWB NZHH RMWVC GSZG RH SRTSVI GSZM 30. LM GSV LGSVI SZMW, NZMB ZWFOGH YVXLNV LEVIDVRTSG DSROV FHRMT NVWRXRMV GL KFG LM DVRTSG HFXS ZH GSV FHV LU XLIGRXLHGVLIRWH ZMW ZMGRWVKIVHHZMGH ZNLMT LGSVIH (DZGVIH HVRWVOO ZMW HDRMYFIM 52). IVXVMG HGFWRVH SZEZ HSLDM GSZG RM NLWVIM HLXRVGB, GSVIV ZIV MFNVILFH RMXRWVMXVH LU LYVHRGB ZH LKKLHVW GL GSV KZHG YVXZFHV KVLKOV OZXP KSBHRXZO ZXGRERGRVH GSZG DVIV KIZXGRXVW YB VZIOB NVM ZMW UZGGB ZMW HFTZIB ULLWH SZEZ YVXLNV NLIV ZEZROYOV GSZM GSVB DVIV RM

GSV GIZWRGRLMZO HLXRVGB. VZGRMT ULLWH DRGS OVHHVI VMVITB ZMW RMWFOTRMT RM KSBHRXZO KIZXGRXVH ZIV GSV LMOB ERZYO NVZMH ULI URTSGRMT LYVHRGB (TLIGNZPVI ZMW WRVGA 47). GSV IVHVZIXSVI VNYZIPVW LM GSV FHV LU Z HFIEVB ZH ZM ZKKILKIRZGV NVGSLW RM IVHVZIXSRMT LYVHV XSROWIVM RM GSV FMRGVW HGZGVH. GSV HFIEVB XLOOVXGVW WZGZ FHRMT Z JFVHGRLMMZRIV DRGS JFVHGRLMH IVOZGVW GL LYVHRGB ZMW DVRTSG NZMZTVNVMG ZNLMT GSV XSROWIVM YVRMT ZHPVW. Z HZNVKOV LU 50 XSROWIVM DZH IZMWLNOB HVOVXGVW DRGS GDL WLXGLIH FHRMT GSV YLWB NZHH RMWVC GL WVGVINRMV GSV SVZOGS LU GSV HZNVKOV XSROWIVM DSLHV ZTV IZMTVW YVGDVVM 5 ZMW 15 BVZIH (UIVVWNZM VG ZO 1759). UILN GSV HFIEVB, RG DZH VERWVMG GSZG 35 XSROWIVM LFG LU 50 DVRTSVW NFXS NLIV GSZM GSVRI SVZOGS, DSRXS DZH ZM RMWRXZGLI GSZG GSVB DVIV LYVHV. ULI GSRH HFIEVB, RG DZH XLMXOFVW GSZG NLHG XSROWIVM ZTVW YVGDVVM 5 BVZIH ZMW 12 BVZIH ZIV ZUUVXGVW YB LYVHRGB VZIOB RM GSVRI OREVB. HFXS XSROWIVM HKVMW NFXS LU GSVRI GRNV DZGXSRMT GVOVERHRLM KILTIZNH IZGSVI GSZM KOZBRMT LM GSV KOZBTILFMW. GSVB ZOHL UVVW LM GLL UZGGB ZMW HFTZIB ULLWH HLNVGSRMT GSZG NZWV GSVN YV NLIV KIVEZOVMG RM LYVHRGB (SVWOVB VG ZO 2500). GSV IVHVZIXS ZXXVKGW GSV MFOO SBKLGSVHRH GSZG LYVHRGB RH NLIV KILMLFMXVW ZNLMT XSROWIVM ZTVW YVGDVVM 5-12 BVZIH YVXZFHV LU OZXP LU KSBHRXZO VCVIXRHV ZMW XLMHFNKGRML LU GLL UZGGB ZMW HFTZIB ULLWH. GSV XZGVTLIB LU XSROWIVM KILMV GL LYVHRGB RH GSV ZTV TILFK LU XSROWIVM ZTVW YVGDVVM 5 ZMW 12 BVZIH. GSV RNKORXZGRML RH GSZG GSVB HKVMG NFXS LU GSVRI GRNV DZGXSRMT GVOVERHRLM IZGSVI GSZM KOZBRMT LM GSV

KOZBTILFMW. GSVB ZOHL SZEZ ORGGOV GSRMPRMT XZKZYRORGB GSV  
 UZXG RH XZFHVW YB FMFHVW VMVITB RM GSV YLWB DRGS GSV  
 VMVITB YVRMT HGLIVW ZH UZG GSZG ZNLFMGH GL LYVHRGB. RU 35  
 XSROWIVM LFG LU 50 ZIV LYVHV, RG NVZMG GSZG RM VEVIB 1000  
 XSROWIVM RM GSV FMRGVW HGZGVH 700 XSROWIVM HFUUVI UILN  
 LYVHRGB. GSV XZGVTLIB LU XSROWIVM KILMV GL LYVHRGB RH GSV  
 ZTV TILFK LU XSROWIVM ZTVW YVGDVVM 5 ZMW 12 BVZIH. GSV  
 RNKORXZGRLM RH GSZG GSVB HKVMG NFXS LU GSVRI GRNV DZGXSMT  
 GVOVERHRLM IZGSVI GSZM KOZBRMT LM GSV KOZBTILFMW. GSVB  
 ZOHL SZEZ ORGGOV GSRMPRMT XZKZYRORGB GSV UZXG RH XZFHVW  
 YB FMFHVW VMVITB RM GSV YLWB DRGS GSV VMVITB YVRMT HGLIVW  
 ZH UZG GSZG ZNLFMGH GL LYVHRGB. RU 35 XSROWIVM LFG LU 50 ZIV  
 LYVHV, RG NVZMG GSZG RM VEVIB 1000 XSROWIVM RM GSV FMRGVW  
 HGZGVH 700 XSROWIVM HFUUVI UILN LYVHRGB. GSV HGFVB RH  
 RMGVIVHGRMT RM GSZG RG SRTSORTSGH GSV XLMXVKG LU LYVHRGB  
 ZH ZKKORVW GL XSROWIVM RM GSV FMRGVW HGZGVH. DZGXSMT  
 GVOVERHRLM ZMW XLMHFNKGRLM LU UZGGB ZMW HFTZIB ULLWH ZIV  
 GSV NZRM XZFHVH LU LYVHRGB ZNLMT XSROWIVM NVZMRMT GSZG RU  
 XSROWIVM XLFOW YV VMXLFIZTVW GL KZIGRXRKZGV RM KSBHRXZO  
 ZXGRERGRVH, LYVHRGB XZM VZHROB YV IVWFXVW TIZWFZOOB ZMW  
 VORNRMZGVW RM GSV VMW. KZIVMGH HSLFOW FITV XSROWIVM GL  
 KOZB NLIV GSZM DZGXS GSV GVOVERHRLM YVXZFHV GSRH DLFOW  
 VMZYOV GSVN GL NZMZTV GSVRI DVRTSG. RM ZWWRGRLM GL  
 RNKILERMT GSVRI SVZOGS, KSBHRXZO ZXGRERGRVH SZEZ YVVM  
 PMLDM GL YV KLHRGREVOB ZHHLXRZGVW DRGS RNKILEVW  
 ZXZWNVRX KVIULINZMXV. ZG GSV MZGRLMZO OVEVO, GSV  
 TLEVIMNVMG HSLFOW RMGILWFXV KSBHRXZO VCVIXRHV KILTIZNH

RM HXSLLOH SVMXV NZPRMT RG NZMWZGLIB ULI ZOO HXSLLOH GL  
 SZEK KOZBTILFMWH ULI XSROWIVM. GSVIVULIV, KZIGRXXRKZGRLM RM  
 KSBHRXZO ZXGRERGRVH YB ZNVIRXZM XSROWIVM DLFOU HORTSGOB  
 IVWFXV GSV SVZOGS YFWTVG ZG GSV MZGRLMZO OVEVO.

Nội dung file decryptedtext sau khi dùng phân tích tần số để phá mã:

the stndu ekycires the linleyt ip igesatu os evadent ofinw lhacdren an the m.s. pir  
 getter mnderstondanw, the ortalce gewon bath the depanatain ip igesatu, revaebed the  
 ekastanw caterotmre, hawhcawhted the fethidiciwu msed ti lindmlt the reseorlh, ond  
 onocuxed the reseorlh pandanws. costcu, reliffendotains bere fode bath the stndu  
 endanw an o linlcmsain. igesatu as o daseose liffincu ossilaoted bath lhacdren an fist  
 limntraes an the bircd. igesatu feons beawhanw fmlh fire thon as heocthu pir sifeine. sife  
 yeiyce linpmse daogetes bath igesatu. these tbi shimcd nit ge lifyored gelomse daogetes  
 as the yridmlt ip igesatu (yokin 27). igesatu as o heocth lindatain thot as lomsed gu  
 linsmfanw ir eotanw tii fmlh enerwu ond pottu piids thot lintoan fire lociraes thon bhot  
 o gidu lon mse. gosed in thas, the ektro lociraes ore stired an the gidu an pirf ip pot ore  
 soad ti ge the poltir thot ceods ti iverbeawht ofinw yeiyce anlcmdanw lhacdren. igesatu,  
 caje onu ither daseose, as treotogce thrimwh eather yhusaloc oltavataes ir fedalanoc  
 yaccs. at hos geen nited thot diltirs mse bhot as jnibn os the gidu foss andek (gfa) ti  
 deterfane bhether sifeines heocth as ot rasj ir nit. thas reseorlh yoyer bos lindmlted bath  
 on oaf ip ekyciranw the anladenle ip igesatu an lhacdren an the m.s. (yokin 45). thesas  
 stotefent: igesatu hos geen o gmrnanw assme ofinw lhacdren an the m.s. pir the cost pimr  
 delodes. beawht fonowefent yriwrofs bere the incu fethid msed gu fedaloc yroltatainers  
 ti pawht owoanst igesatu ond thas hos ced ti heocthaer caves ofinw lhacdren cavanw bath  
 igesatu.igesatu as o seraims fedaloc lindatain thot hos oppelted the heocth ip fonu  
 lhacdren an the mnated stotes sanle the 1980s ti dote. anataoccu, yeiyce thimwht thot  
 woananw beawht an lhacdren bos o wiid andalotir ip geaww heocthu bathimt  
 linleytmocaxanw the donwers ossilaoted bath geaww iverbeawht an lhacdren. stmdaes

hove shibn thot igesatu as o seraims daseose thot lon ceod ti ither heoeth yrigcefs thot  
 anlcmde orthratas, daogetes, heort daseose ond hawh gciid yressmre ofinw igese valtafs  
 (yokin 61). octhimwh stmdaes hove andaloted thot lhacdren ore fire yrine ti igesatu thon  
 odmcts ore, safacor stmdaes hove shibn thot everugidu as o valtaf ip igesatu gelomse the  
 daseose as corwecu recoted ti eotanw hogats thot ore liffin ti everu andavadmoc an the  
 bircd. thrimwh nmferims stmdaes, at as evadent thot igesatu an the m.s. as hawhcu  
 yrinimnled lifyored ti ither notains an the bircd (hedceu et oc 2500). an linzmltain bath  
 thas, fonu igese valtafs ore lhacdren os iyyised ti odmcts foancu gelomse ip the lhonwes  
 an capestuces bheregu telhnicwaloc odvonlefent hos ced ti the antridmltain ip fire  
 cmkmraims oltavataes. pir anstonle, lhacdren getbeen the owe ip 5 ond 12 ueors ore yrine  
 ti igesatu gelomse theu syend fmlh ip thear tafe botlhanw tecevasain ot the ekyense ip  
 yhusaloc oltavataes. nmferims stmdaes reveoced thot geaww o peb yimnds iverbeawht  
 as nit on afycalotain ip igesatu gmt o yersin as linsadered igese bhen he ir she hos o gidu  
 foss andek thot as hawher thon 30. in the ither hond, fonu odmcts gelife iverbeawht bhace  
 msanw fedalane ti ymt in beawht smlh os the mse ip lirtalisteriads ond ontadeyressonts  
 ofinw ithers (boters seadecc ond sbangmrn 52). relent stmdaes hove shibn thot an fidern  
 silaetu, there ore nmferims anladenles ip igesatu os iyyised ti the yost gelomse yeiyce  
 colj yhusaloc oltavataes thot bere yroltaled gu eorcu fen ond pottu ond smworu piids  
 hove gelife fire ovoacogce thon theu bere an the trodatainoc silaetu. eotanw piids bath  
 cesser enerwu ond andmcwanw an yhusaloc yroltales ore the incu vaogce feons pir  
 pawhtanw igesatu (wirtfojer ond daetx 47). the reseorlher efgorjed in the mse ip o smrveu  
 os on oyyiyrate fethid an reseorlhanw igese lhacdren an the mnated stotes. the smrveu  
 licceted doto msanw o qmestainnoare bath qmestains recoted ti igesatu ond beawht  
 fonowefent ofinw the lhacdren geaww osjed. o sofyce ip 50 lhacdren bos rondifcu  
 secelted bath tbi diltirs msanw the gidu foss andek ti deterfane the heoeth ip the sofyced  
 lhacdren bhise owe ronwed getbeen 5 ond 15 ueors (preedfon et oc 1759). prif the  
 smrveu, at bos evadent thot 35 lhacdren imt ip 50 beawhed fmlh fire thon thear heoeth,

bhalh bos on andalotir thot theu bere igese. pir thas smrveu, at bos linlcmedd thot fist  
 lhacdren owed getbeen 5 ueors ond 12 ueors ore oppelted gu igesatu eorcu an thear caves.  
 smlh lhacdren syend fmlh ip thear tafe botlhanw tecevasain yriwrofs rother thon  
 ycouanw in the ycouwrimnd. theu ocsi peed in tii pottu ond smworu piids sifethanw thot  
 fode thef ge fire yrevocent an igesatu (hedceu et oc 2500). the reseorlh olleyted the nmcc  
 huyithesas thot igesatu as fire yrinimnded ofinw lhacdren owed getbeen 5-12 ueors  
 gelomse ip colj ip yhusaloc ekerlase ond linsmfytain ip tii pottu ond smworu piids. the  
 lotewiru ip lhacdren yrine ti igesatu as the owe wrimy ip lhacdren owed getbeen 5 ond  
 12 ueors. the afycalotain as thot theu syent fmlh ip thear tafe botlhanw tecevasain rother  
 thon ycouanw in the ycouwrimnd. theu ocsi hove cattce thanjanw loyogacatu the polt as  
 lomsed gu mnmsted enerwu an the gidu bath the enerwu geanw stired os pot thot ofimnts  
 ti igesatu. ap 35 lhacdren imt ip 50 ore igese, at feont thot an everu 1000 lhacdren an the  
 mnated stotes 700 lhacdren smpper prif igesatu. the lotewiru ip lhacdren yrine ti igesatu  
 as the owe wrimy ip lhacdren owed getbeen 5 ond 12 ueors. the afycalotain as thot theu  
 syent fmlh ip thear tafe botlhanw tecevasain rother thon ycouanw in the ycouwrimnd.  
 theu ocsi hove cattce thanjanw loyogacatu the polt as lomsed gu mnmsted enerwu an the  
 gidu bath the enerwu geanw stired os pot thot ofimnts ti igesatu. ap 35 lhacdren imt ip  
 50 ore igese, at feont thot an everu 1000 lhacdren an the mnated stotes 700 lhacdren  
 smpper prif igesatu. the stmdu as anterestanw an thot at hawhcawhts the linleyt ip igesatu  
 os oyycaed ti lhacdren an the mnated stotes. botlhanw tecevasain ond linsmfytain ip pottu  
 ond smworu piids ore the foan lomses ip igesatu ofinw lhacdren feonanw thot ap  
 lhacdren limcd ge enlimrowed ti yortalayote an yhusaloc oltavataes, igesatu lon eosacu  
 ge redmled wrodmoccu ond ecafanoted an the end. yorents shimcd mrwe lhacdren ti  
 ycou fire thon botlh the tecevasain gelomse thas bimcd enogce thef ti fonowe thear  
 beawht. an oddatain ti afyriwanw thear heocth, yhusaloc oltavataes hove geen jnibn ti ge  
 yisatavecu ossilaoted bath afyriwed olodefal yerpirfonle. ot the notainoc cevec, the  
 wivernfent shimcd antridmle yhusaloc ekerlase yriwrofs an slhiics henle fojanw at

fondotiru pir occ shiics ti hove ycouwrimnds pir lhacdren. therepire, yortalayotain an yhusaloc oltavataes gu oferalon lhacdren bimcd scawhtcu redmle the heoch gmdwet ot the notainoc cevec.

```

atbash.py > at_bash
8 import string
7
6
5 def at_bash(plaintext):
4     alphabet = list(string.ascii_lowercase)
3     key = alphabet.copy()
2     key.reverse()
1     plaintext = plaintext.lower()
9     print(key)
1     ciphertext = ""
2     for i in plaintext:
3         if i.isalpha():
4             ciphertext += key[ord(i) - 97]
5         else:
6             ciphertext += i
7     return ciphertext.upper()

PROBLEMS 262 OUTPUT TERMINAL DEBUG CONSOLE
[Running] python -u "d:\Code\study\statictis\midtern\atbash.py"
['z', 'y', 'x', 'w', 'v', 'u', 't', 's', 'r', 'q', 'p', 'o', 'n', 'm', 'l', 'k', 'j', 'i', 'h', 'g', 'f', 'e', 'd', 'c', 'b', 'a']
[Done] exited with code=0 in 0.485 seconds

```

Hình 4.12 Minh họa sử dụng thuật toán affine để mã hóa chuỗi 1000 từ



## TÀI LIỆU THAM KHẢO

Tiếng Anh

Encryption: <https://en.wikipedia.org/wiki/Encryption>

Decryption: <https://www.techopedia.com/definition/1773/decryption>

Asymmetric & symmetric key: <https://www.geeksforgeeks.org/difference-between-symmetric-and-asymmetric-key-encryption/>

Monoalphabetic Substitution Cipher:

<https://www.101computing.net/mono-alphabetic-substitution-cipher/>

[https://en.wikipedia.org/wiki/Substitution\\_cipher](https://en.wikipedia.org/wiki/Substitution_cipher)

<https://www.geeksforgeeks.org/difference-between-monoalphabetic-cipher-and-polyalphabetic-cipher/>

Thông tin về các loại mã được đề cập đến trong báo cáo:

[https://crypto.interactive-maths.com/frequency-analysis-breaking-the-code.html#google\\_vignette](https://crypto.interactive-maths.com/frequency-analysis-breaking-the-code.html#google_vignette)

Frequency Analysis: [https://en.wikipedia.org/wiki/Frequency\\_analysis](https://en.wikipedia.org/wiki/Frequency_analysis)

<https://www.101computing.net/frequency-analysis/>

<https://crypto.interactive-maths.com/frequency-analysis-breaking-the-code.html>

