

Nama : Danang Wisnu Prabowo

NIM : 222111975

Kelas : 3SI2

## Tugas KSI Pertemaun 5

# Program Transposition Chipper

```
import math

def main():
    text = input("Masukkan Pesan : ")
    key = int(input("Masukkan Kunci [2-%s]: " % (len(text) - 1)))
    if (key < 2) or (key > len(text) - 1):
        print("\nWARNING : Silahkan masukkan kunci dalam rentang !")
    else :
        mode = input("Enkripsi/Dekripsi [1/2]: ")
        if(mode.lower() == "1"):
            result = encryptionMessage(key, text)
        elif (mode.lower() == "2"):
            result = decryptMessage(key, text)
        print ("\nHasil : \n%s" % (result + "|"))

def encryptionMessage(key, message):
    cipherText = [""] * key
    for col in range(key):
        pointer = col
        while pointer < len(message):
            cipherText[col] += message[pointer]
            pointer += key
        return "".join(cipherText)

def decryptMessage(key, message):
    numCols = math.ceil(len(message) / key)
    numRows = key
    numShadedBoxes = (numCols * numRows) - len(message)
    plainText = [""] * numCols
    col = 0
    row = 0
    for karakter in message:
        plainText[col] += karakter
        col += 1
        if(col == numCols or (col == numCols - 1) and (row >= numRows - numShadedBoxes)):
            col = 0
            row += 1
    return "".join(plainText)

main()
```

```
def encryptionMessage(key, message):
    cipherText = [""] * key
    for col in range(key):
        pointer = col
        while pointer < len(message):
            cipherText[col] += message[pointer]
            pointer += key
        return "".join(cipherText)

def decryptMessage(key, message):
    numCols = math.ceil(len(message) / key)
    numRows = key
    numShadedBoxes = (numCols * numRows) - len(message)
    plainText = [""] * numCols
    col = 0
    row = 0
    for karakter in message:
        plainText[col] += karakter
        col += 1
        if(col == numCols or (col == numCols - 1) and (row >= numRows - numShadedBoxes)):
            col = 0
            row += 1
    return "".join(plainText)

main()
```

Hasil pengujian :

a. Enkripsi

```
Masukkan Pesan : SELAMAT MALAM, JANGAN LUPA MAKAN
Masukkan Kunci [2-31]: 5
Enkripsi/Dekripsi [1/2]: 1

Hasil :
SALJNAAETAA  NL MNLMAM,GUAMA APK|
```

b. Dekripsi

```
Masukkan Pesan : SALJNAAETAA  NL MNLMAM,GUAMA APK
Masukkan Kunci [2-31]: 5
Enkripsi/Dekripsi [1/2]: 2

Hasil :
SELAMAT MALAM, JANGAN LUPA MAKAN|
```

Penjelasan:

Program di atas merupakan program enkripsi dan dekripsi untuk transposition (Matrix baris dan kolom). Adapun cara kerja dari algoritma di atas adalah sebagai berikut:

1. Algoritma Enkripsi

- Key akan menjadi jumlah kolom
- Menyiapkan array sebanyak kolom
- Menentukan start iterasi, lalu mengambil huruf dengan kelipatan jumlah kolom dan disimpan suatu array. Contoh: misalkan ada tiga kolom, array pertama menyimpan huruf index ke 0, 3, 6, dst; array kedua menyimpan huruf index ke 1, 4, 7, dst, array ketiga menyimpan huruf index ke 2, 5, 8, dst.
- Menggabungkan huruf yang tersimpan di masing-masing array dari index array terkecil

2. Algoritma dekripsi

- Berkebalikan dengan enkripsi, key dalam dekripsi akan dijadikan sebagai jumlah baris, sedangkan jumlah kolom adalah hasil pembulatan ke atas dari pembagian panjang pesan dan key.
- Menentukan jumlah sel yang akan kosong, yaitu dari seluruh jumlah sel yang terbentuk di kurangi panjang huruf
- Melakukan perulangan untuk setiap karakter, menyimpan karakter berurutan untuk setiap kolom. Baris akan berpindah apabila kolom dalam satu baris sudah

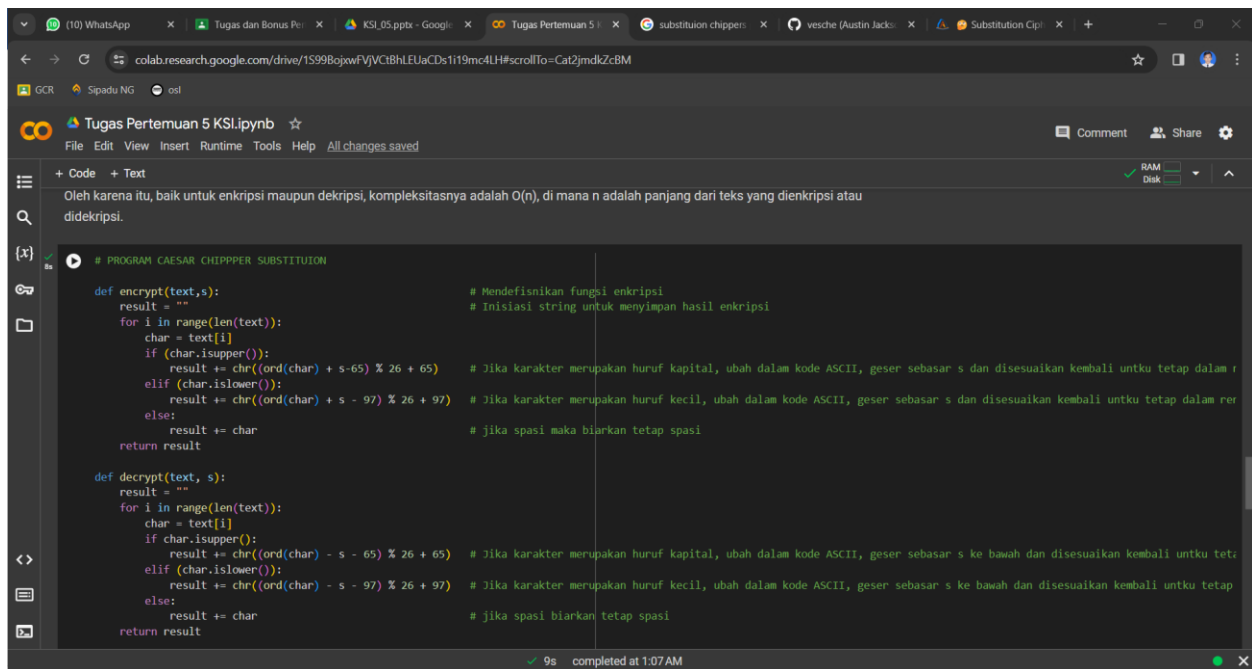
penuh, atau jika kolom masih tersisa satu, tapi baris lebih dari sama dengan jumlah sel yang kosong.

- o Setelah karakter habis, gabungkan array yang menyimpan huruf dalam satu kolom dari index terkecil.

Baik untuk algoritma enkripsi dan dekripsi di atas, memiliki kompleksitas  $O(n+k)$ . Hal ini disebabkan karena hal berikut.

- Iterasi melalui setiap karakter dalam pesan membutuhkan waktu  $O(n)$ , di mana  $n$  adalah panjang pesan.
- Dalam iterasi, kita mengumpulkan karakter-karakter dalam array kolom, yang memiliki ukuran tetap sesuai dengan jumlah kolom. Oleh karena itu, mengumpulkan karakter-karakter ini dalam array kolom membutuhkan waktu konstan
- Menggabungkan hasil dari array kolom menjadi satu string membutuhkan waktu  $O(k)$ , di mana  $k$  adalah kunci.

## Program Substitution Chipper – Cesar



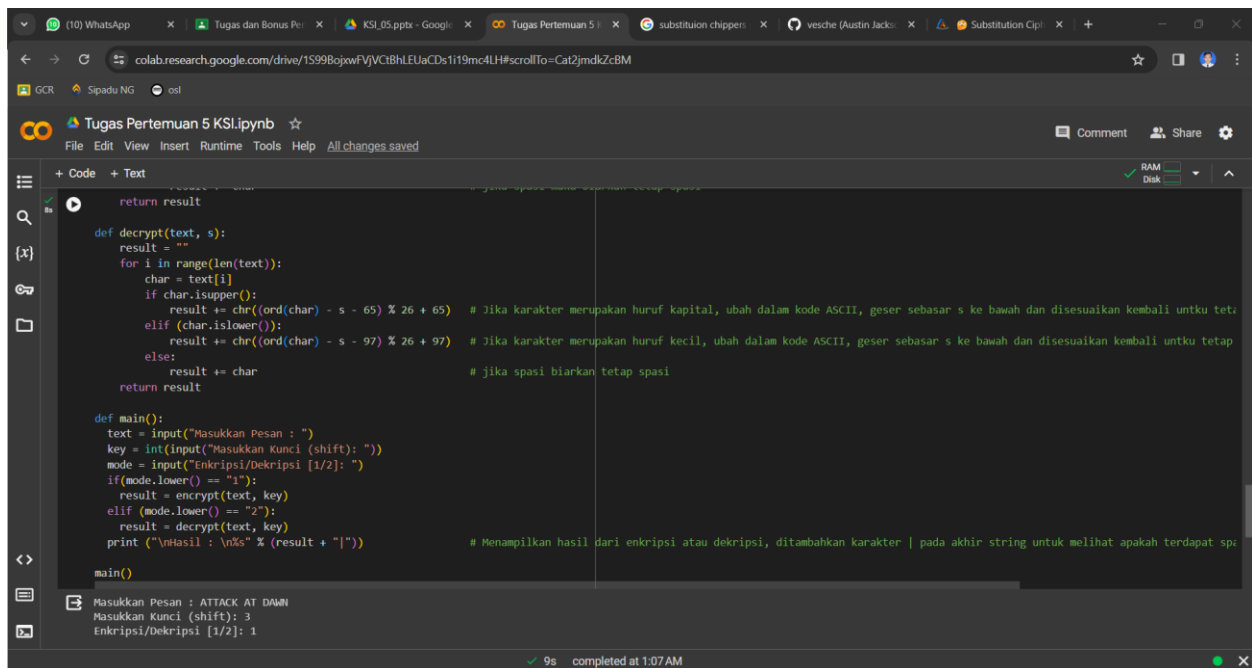
Oleh karena itu, baik untuk enkripsi maupun dekripsi, kompleksitasnya adalah  $O(n)$ , di mana  $n$  adalah panjang dari teks yang dienkripsi atau didekripsi.

```
# PROGRAM CAESAR CHIPPER SUBSTITUTION

def encrypt(text,s):
    result = ""
    # Mendefinisikan fungsi enkripsi
    # Inisiasi string untuk menyimpan hasil enkripsi
    for i in range(len(text)):
        char = text[i]
        if (char.isupper()):
            result += chr((ord(char) + s - 65) % 26 + 65) # Jika karakter merupakan huruf kapital, ubah dalam kode ASCII, geser sebesar s dan disesuaikan kembali untuk tetap dalam r
        elif (char.islower()):
            result += chr((ord(char) + s - 97) % 26 + 97) # Jika karakter merupakan huruf kecil, ubah dalam kode ASCII, geser sebesar s dan disesuaikan kembali untuk tetap dalam rer
        else:
            result += char # jika spasi maka biarkan tetap spasi
    return result

def decrypt(text, s):
    result = ""
    for i in range(len(text)):
        char = text[i]
        if char.isupper():
            result += chr((ord(char) - s - 65) % 26 + 65) # Jika karakter merupakan huruf kapital, ubah dalam kode ASCII, geser sebesar s ke bawah dan disesuaikan kembali untuk teti
        elif (char.islower()):
            result += chr((ord(char) - s - 97) % 26 + 97) # Jika karakter merupakan huruf kecil, ubah dalam kode ASCII, geser sebesar s ke bawah dan disesuaikan kembali untuk tetap
        else:
            result += char # jika spasi biarkan tetap spasi
    return result
```

9s completed at 1:07 AM



```
def decrypt(text, s):
    result = ""
    for i in range(len(text)):
        char = text[i]
        if char.isupper():
            result += chr((ord(char) - s - 65) % 26 + 65) # Jika karakter merupakan huruf kapital, ubah dalam kode ASCII, geser sebesar s ke bawah dan disesuaikan kembali untuk tet
        elif char.islower():
            result += chr((ord(char) - s - 97) % 26 + 97) # Jika karakter merupakan huruf kecil, ubah dalam kode ASCII, geser sebesar s ke bawah dan disesuaikan kembali untuk tetap
        else:
            result += char # jika spasi biarkan tetap spasi
    return result

def main():
    text = input("Masukkan Pesan : ")
    key = int(input("Masukkan Kunci (shift): "))
    mode = input("Enkripsi/Dekripsi [1/2]: ")
    if mode.lower() == "1":
        result = encrypt(text, key)
    elif mode.lower() == "2":
        result = decrypt(text, key)
    print("\nHasil : \n%s" % (result + "|")) # Menampilkan hasil dari enkripsi atau dekripsi, ditambahkan karakter | pada akhir string untuk melihat apakah terdapat sp

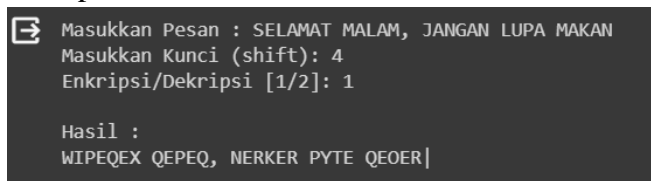
main()
```

Masukkan Pesan : ATTACK AT DAWN  
Masukkan Kunci (shift): 3  
Enkripsi/Dekripsi [1/2]: 1

9s completed at 1:07 AM

Hasil Pengujian :

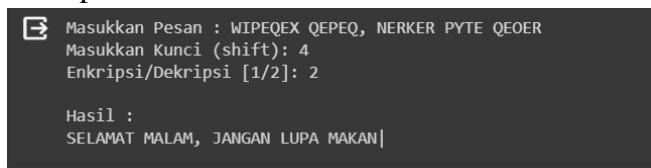
a. Enkripsi



```
Masukkan Pesan : SELAMAT MALAM, JANGAN LUPA MAKAN
Masukkan Kunci (shift): 4
Enkripsi/Dekripsi [1/2]: 1

Hasil :
WIPEQEX QEPEQ, NERKER PYTE QEOR|
```

b. Dekripsi



```
Masukkan Pesan : WIPEQEX QEPEQ, NERKER PYTE QEOR
Masukkan Kunci (shift): 4
Enkripsi/Dekripsi [1/2]: 2

Hasil :
SELAMAT MALAM, JANGAN LUPA MAKAN|
```

Penjelasan:

Program di atas merupakan program enkripsi dan dekripsi untuk algoritma caesar substitution. adapun cara kerja algoritma di atas adaalh sebagai berikut

1. Algoritma Enkripsi

- Perulangan dilakukan untuk setiap karakter
- Karakter akan di cek apakah merupakan huruf kapital, kecil, atau spasi. Jika huruf kapital, huruf akan digeser atau ditambah sebanyak shift (s) dan akan dikembalikan untuk tetap di rentang 0 - 26 dalam konteks huruf kapital (d disesuaikan kode ASCII nya dengan ditambah 65). Jika huruf kecil, huruf akan digeser atau ditambah sebanyak shift (s) dan akan dikembalikan untuk tetap di

rentang 0 - 26 dalam konteks huruf kecil (disesuaikan kode ASCII nya dengan ditambah 97). Jika spasi, akan dibarkan untuk tetap menjadi spasi.

- o Untuk setiap karakter yang sudah di ditambah, masukan ke dalam variabel string result.

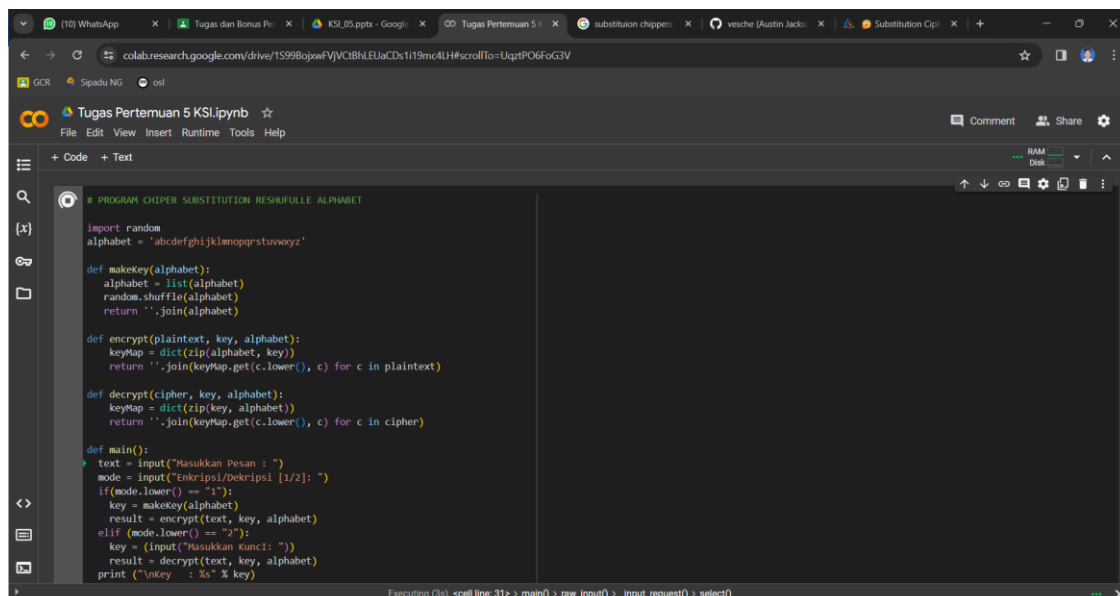
## 2. Algoritma Dekripsi

- o Perulangan dilakukan untuk setiap karakter
- o Karakter akan di cek apakah merupakan huruf kapital, kecil, atau spasi. Jika huruf kapital, huruf akan dikurangi sebanyak shift (s) dan akan dikembalikan untuk tetap di rentang 0 - 26 dalam konteks huruf kapital (disesuaikan kode ASCII nya dengan ditambah 65). Jika huruf kecil, huruf akan dikurangi sebanyak shift (s) dan akan dikembalikan untuk tetap di rentang 0 - 26 dalam konteks huruf kecil (disesuaikan kode ASCII nya dengan ditambah 97). Jika spasi, akan dibarkan untuk tetap menjadi spasi.
- o Untuk setiap karakter yang sudah di ditambah, masukan ke dalam variabel string result.

Baik fungsi encrypt maupun decrypt melalui setiap karakter dalam teks, dan operasi-operasi yang dilakukan di dalam loop tersebut (seperti ord, chr, dan operasi aritmatika) memerlukan waktu konstan. Jumlah iterasi dalam loop sesuai dengan panjang teks yang diberikan, sehingga kompleksitasnya bergantung pada panjang teks tersebut.

Oleh karena itu, baik untuk enkripsi maupun dekripsi, kompleksitasnya adalah  $O(n)$ , di mana  $n$  adalah panjang dari teks yang dienkripsi atau didekripsi.

## Program Substitution Chipper – key reshuffle alphabet



```
# PROGRAM CHIPPER SUBSTITUTION RESHUFFLE ALPHABET
import random
alphabet = 'abcdefghijklmnopqrstuvwxyz'

def makekey(alphabet):
    alphabet = list(alphabet)
    random.shuffle(alphabet)
    return ''.join(alphabet)

def encrypt(plaintext, key, alphabet):
    keymap = dict(zip(alphabet, key))
    return ''.join(keymap.get(c.lower(), c) for c in plaintext)

def decrypt(cipher, key, alphabet):
    keymap = dict(zip(key, alphabet))
    return ''.join(keymap.get(c.lower(), c) for c in cipher)

def main():
    text = input("Masukkan Pesan : ")
    mode = input("Enkripsi/Denkripsi [1/2]: ")
    if mode.lower() == "1":
        key = makekey(alphabet)
        result = encrypt(text, key, alphabet)
    elif mode.lower() == "2":
        key = input("Masukkan Kunci: ")
        result = decrypt(text, key, alphabet)
    print("Key : %s" % key)

if __name__ == '__main__':
    main()
```

## Hasil Pengujian :

### a. Enkripsi

```
➡ Masukkan Pesan : Selamat malam, jangan lupa makan !  
Enkripsi/Dekripsi [1/2]: 1  
  
Key   : qdllozpauhcrsjbefgwtymxknv  
  
Hasil :  
worqsqt sqrqs, hqjppj ryeq sqcqj !|
```

### b. Dekripsi

```
➡ Masukkan Pesan : worqsqt sqrqs, hqjppj ryeq sqcqj !  
Enkripsi/Dekripsi [1/2]: 2  
Masukkan Kunci: qdllozpauhcrsjbefgwtymxknv  
  
Key   : qdllozpauhcrsjbefgwtymxknv  
  
Hasil :  
selamat malam, jangan lupa makan !|
```

## Penjelasan:

Program di atas merupakan program enkripsi dan dekripsi untuk algoritma substitution dimana key yang diberikan adalah huruf pengganti dari setiap alphabet. Algoritma dari program di atas adalah

### 1. Algoritma Enkripsi

- Membuat peta kunci dengan menggunakan fungsi `zip()` untuk menggabungkan setiap huruf dalam alphabet dengan huruf yang sesuai dalam key. Fungsi `zip()` membuat pasangan nilai dari alphabet dan key, dan fungsi `dict()` digunakan untuk mengonversi pasangan nilai ini menjadi dictionary. Dictionary ini berisi pasangan nilai dengan huruf dari alphabet sebagai kunci dan huruf dari key sebagai nilai.
- Lalu menggunakan enkripsi generator untuk menghasilkan karakter-karakter hasil enkripsi. Dimana setiap karakter plain text diambil satu persatu, lalu mengambil nilai dari dictionary berdasarkan key (karakter dari plain text), jika karakter bukan huruf maka akan dipertahankan.
- Menggabungkan setiap hasil enkripsi karakter dengan menggunakan `join`.

### 2. Algoritma Dekripsi

- Algoritma sama dengan enkripsi, hanya saja untuk membuat dictionary peta kunci, yang menjadi key adalah key, sedangkan value nya adalah alphabet asli.

Kedua algoritma di atas melakukan iterasi melalui setiap karakter dalam teks yang diberikan, dan operasi-operasi yang dilakukan di dalam loop tersebut (seperti `dict(zip(...))`, `get()`, `lower()`, `join()`, dan operasi untuk mengakses elemen dalam kamus) memerlukan waktu konstan. Jumlah iterasi dalam loop sesuai dengan panjang teks yang diberikan, sehingga kompleksitasnya bergantung pada panjang teks tersebut.

Oleh karena itu, baik untuk enkripsi maupun dekripsi, kompleksitasnya adalah  $O(n)$ , di mana  $n$  adalah panjang dari teks yang dienkripsi atau didekripsi.