UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGÓN

INGENIERÍA EN COMPUTACIÓN

Diseño y Análisis de Algoritmos

PROFESOR	:	JESUS HERNANDEZ CABRERA
ALUMNOS	:	LÓPEZ GARCÍA AARÓN DANIEL
GRUPO	:	1561
SEMESTRE	:	2025 – 1
ACTIVIDAD	:	TAREA 7.- 3 EJERCICIOS DE FUERZA BRUTA

Ejercicio 1

```python
A = [-9, 3, 5, -2, 9, -7, 4, 8, 6]

# Inicializar el producto máximo como un valor muy pequeño
producto_maximo = float('-inf')
num1, num2 = None, None  # Para almacenar los números con el producto máximo

# Iterar sobre todos los pares posibles en el arreglo
for i in range(len(A)):
    for j in range(i + 1, len(A)):  # Evitar repetir pares, comenzando desde i+1
        producto = A[i] * A[j]
        if producto > producto_maximo:
            producto_maximo = producto
            num1, num2 = A[i], A[j]

# Mostrar el resultado
print(f"El producto más alto es {producto_maximo}, obtenido con los números {num1} y {num2}.")
```

Big O

A = [-9, 3, 5, -2, 9, -7, 4, 8, 6]

producto_maximo = float('-inf')

num1, num2 = None, None

for i in range(len(A)): T(n)=n

  for j in range(i + 1, len(A)): T(n)=n

    producto = A[i] * A[j]

    if producto > producto_maximo:

      producto_maximo = producto

      num1, num2 = A[i], A[j]

print(f"El producto más alto es {producto_maximo}, obtenido con los números {num1} y {num2}.")

$T(n) = 4 + n^2$

$O(n^2)$

```
PS C:\Users\adany\Desktop\Tarea 7> & C:/Users/adany/AppData/
El producto más alto es 72, obtenido con los números 9 y 8.
PS C:\Users\adany\Desktop\Tarea 7> & C:/Users/adany/AppData/
El producto más alto es 72, obtenido con los números 9 y 8.
```

Ejercicio 5

```python
def cifrado_cesar(texto):
    alfabeto = 'abcdefghijklmnopqrstuvwxyz'
    texto = texto.lower()  # Convertir a minúsculas para facilitar el cifrado
    resultados = []

    # Probar todos los desplazamientos del 1 al 25
    for desplazamiento in range(1, 26):
        texto_cifrado = ''
        for char in texto:
            if char in alfabeto:
                # Encontrar la nueva posición en el alfabeto
                nueva_posicion = (alfabeto.index(char) + desplazamiento) % 26
                texto_cifrado += alfabeto[nueva_posicion]
            else:
                texto_cifrado += char  # No cifrar caracteres no alfabéticos

        resultados.append((desplazamiento, texto_cifrado))

    return resultados

# Texto cifrado
texto_cifrado = """
Cupdmzaplil Uikpwuis Icbwuwti lm Tmfpkw oi lmamtxmvilw cu xixms
xzwbiñwupkw mu si opabwzpi g mu si nwztikpwu lm ucmabzw xipa.
Sia bizmia acabiubpdia lm mabi puabpbckpwu xcjspki, icbwuwti g
sipki awu si lwkmukpi, si pudmabpñikpwu g si lpncapwu lm si
kcsbczi. Mu ms tculw ikilmtpkw ma zmkwuwkpli kwtw cui cupdmzaplil
lm mfkmsmukpi. Si CUIT zmaxwulm is xzmamubm g tpzi ms ncbczw
kwtw ms xzwgmkbw kcsbczis tia ptxwzbiubm lm Tmfpkw. Si CUIT ma
cu maxikpw lm spjmzbilma. Mu mssi am xzikbpki kwbplpiuitmubm
ms zmaxmbw, si bwsmziukpi g ms lpiswñw. Si xsczisplil lm plmia
g lm xmuaitpmubw ma ixzmkpili kwtw apñuw lm ac zpycmhi g ucuki
kwtw nikbwz lm lmjpsplil.
"""
```

Big O

```
def cifrado_cesar(texto):
    alfabeto = 'abcdefghijklmnopqrstuvwxyz'
    texto = texto.lower()  # Convertir a minúsculas para facilitar el cifrado
    resultados = []

    # Probar todos los desplazamientos del 1 al 25
    for desplazamiento in range(1, 26):
        texto_cifrado = ''
        for char in texto:
            if char in alfabeto:
                # Encontrar la nueva posición en el alfabeto
                nueva_posicion = (alfabeto.index(char) + desplazamiento) % 26
                texto_cifrado += alfabeto[nueva_posicion]
            else:
                texto_cifrado += char  # No cifrar caracteres no alfabéticos

        resultados.append((desplazamiento, texto_cifrado))

    return resultados

# Texto cifrado
texto_cifrado = """
Cupdmzaplil Uikpwuis Icbwuwti lm Tmfpkw oi lmamtxmvilw cu xixms
xzwbiñwupkw mu si opabwzpi g mu si nwztikpwu lm ucmabzw xipa.
Sia bizmia acabiubpdia lm mabi puabpbckpwu xcjspki, icbwuwti g
```

sipki awu si lwkmukpi, si pudmabpñikpwu g si lpncapwu lm si kcsbczi. Mu ms tculw ikilmtpkw ma zmkwuwkpli kwtw cui cupdmzaplil lm mfkmsmukpi. Si CUIT zmaxwulm is xzmamubm g tpzi ms ncbczw kwtw ms xzwgmkbw kcsbczis tia ptxwzbiubm lm Tmfpkw. Si CUIT ma cu maxikpw lm spjmzbilma. Mu mssi am xzikbpki kwbplpiuitmubm ms zmaxmbw, si bwsmziukpi g ms lpiswñw. Si xsczisplil lm plmia g lm xmuaitpmubw ma ixzmkpili kwtw apñuw lm ac zpycmhi g ucuki kwtw nikbwz lm lmjpsplil.

"""

```
resultados = cifrado_cesar(texto_cifrado)
for desplazamiento, texto_cifrado in resultados:
    print(f"Desplazamiento {desplazamiento}: {texto_cifrado}")
```

$T(n) = 25 \cdot n$

$O(n)$

```
Desplazamiento 25:
btoclyzokhk thjovthr hbavtvsh kl sleojv nh klzlswluhkv bt whwlr
wyvahñvtojv lt rh nozavyoh f lt rh mvyshjovt kl tblzayv whoz.
rhz ahylhz zbzahtaochz kl lzah otzaoabjovt wbirojh, hbavtvsh f
rhojh zvt rh kvjltjoh, rh otclzaoñhjovt f rh kombzovt kl rh
jbrabyh. lt lr sbtkv hjhklsojv lz yljvtvjokh jvsv bth btoclyzokhk
kl lejlrltjoh. rh bths ylzwvtkl hr wylzltal f soyh lr mbabyv
jvsv lr wyvfljav jbrabyhr shz oswvyahtal kl sleojv. rh bths lz
bt lzwhjov kl roilyahklz. lt lrrh zl wyhjaojh jvaokohthsltal
lr ylzwlav, rh avrlyhtjoh f lr kohrvñv. rh wrbyhrokhk kl oklhz
f kl wltzhsoltav lz hwyljohkh jvsv zoñtv kl zb yoxblgh f tbtjh
jvsv mhjavy kl kliorokhk.
```

Ejercicio 7

```python
pesos = [2, 3, 4, 5]
valores = [3, 4, 5, 8]
capacidad = 8
n = len(pesos)

# Crear una tabla DP con dimensiones (n+1) x (capacidad+1)
dp = [[0] * (capacidad + 1) for _ in range(n + 1)]

# Llenar la tabla DP
for i in range(1, n + 1):
    for w in range(1, capacidad + 1):
        if pesos[i - 1] <= w:
            dp[i][w] = max(dp[i - 1][w], dp[i - 1][w - pesos[i - 1]] + valores[i - 1])
        else:
            dp[i][w] = dp[i - 1][w]

# El valor máximo estará en dp[n][capacidad]
valor_maximo = dp[n][capacidad]

# Mostrar el valor máximo
print(f"El valor máximo que podemos llevar es: {valor_maximo}")
```

Big O

pesos = [2, 3, 4, 5]

valores = [3, 4, 5, 8]

capacidad = 8

n = len(pesos)

dp = [[0] * (capacidad + 1) for _ in range(n + 1)]

for i in range(1, n + 1):

  for w in range(1, capacidad + 1):

    if pesos[i - 1] <= w:

      dp[i][w] = max(dp[i - 1][w], dp[i - 1][w - pesos[i - 1]] + valores[i - 1])

    else:

      dp[i][w] = dp[i - 1][w]

valor_maximo = dp[n][capacidad]

print(f"El valor máximo que podemos llevar es: {valor_maximo}")

$T(n) = 6 + n \cdot n \cdot n$

$O(n^3)$