S1-Complejidad1

Pregunta **1**

Sin responder aún

Puntúa como 1.00

Marcar pregunta

¿Cuál es la complejidad asintótica, en el peor de los casos, del algoritmo de la subsecuencia común más larga entre X (de longitud m) y Y (de longitud p) presentado a continuación? Considera que el tamaño del problema
n = m + p.

```
def lcsAUX(X, Y, m, p):
    if m == 0 or p == 0:
      return 0
    elif X[m-1] == Y[p-1]:
      return 1 + lcsAUX(X, Y, m-1, p-1)
    else:
      return max(lcsAUX(X, Y, m, p-1),
                 lcsAUX(X, Y, m-1, p))
def lcs(X,Y):
    return lcsAUX(X,Y,len(X),len(Y))
```

- a. O(n^2)
- b. O(n^3)
- c. O(n)
- d. O(log n)
- e. O(2^n)

QUITAR MI ELECCIÓN

Pregunta **2**

Sin responder aún

Puntúa como 1.00

Marcar pregunta

¿Cuál es la complejidad asintótica, en el peor de los casos, del algoritmo de la distancia de Levenshtein entre la cadena X (de longitud m) y la cadena Y (de longitud n), implementado a continuación?

```
def editDist(str1, str2, m, n):
    dp = [[0 for x in range(n + 1)]
          for x in range(m + 1)]
    for i in range(m + 1):
        for j in range(n + 1):
            if i == 0:
                dp[i][j] = j
            elif j == 0:
                dp[i][j] = i
            elif str1[i-1] == str2[j-1]:
                dp[i][j] = dp[i-1][j-1]
            else:
                dp[i][j] = 1 + min(dp[i][j-1],
                                   dp[i-1][j],
                                   dp[i-1][j-1])
    return dp[m][n]
```

```
def levenshteinDistance(X, Y):
    return editDist(X, Y, len(X), len(Y))
```

- a. O(n log m)
- b. O(n)
- c. O(n^2*m^2)
- d. O(2^(n+m))
- e. O(n*m)

QUITAR MI ELECCIÓN

Pregunta **3**

Sin responder aún

Puntúa como 1.00

Marcar pregunta

¿Cuál es la complejidad asintótica, para el peor de los casos, de determinar si es posible encontrar un subgrupo de la lista *set* (que tiene *n* elementos) que sume el valor *sum*, con el siguiente algoritmo?

```
def isSubsetSum(set, n, sum):
    subset =([[False for i in range(sum + 1)]
            for i in range(n + 1)])
    for i in range(n + 1):
        subset[i][0] = True
    for i in range(1, sum + 1):
         subset[0][i]= False
    for i in range(1, n + 1):
        for j in range(1, sum + 1):
            if j<set[i-1]:
                subset[i][j] = subset[i-1][j]
            if j>= set[i-1]:
                subset[i][j] = (subset[i-1][j] or
                                subset[i - 1][j-
                                      set[i-1]])
    return subset[n][sum]
```

```
def sumaGrupo(set, sum):
    return isSubsetSum(set, len(set), sum)
```

- a. O(n)
- b. O(sum log n)
- c. O(n*sum)
- d. O(2^(sum + n))
- e. O(n^2*sum*2)

QUITAR MI ELECCIÓN