

Va a comparar cada numero por separado hasta ubicarlo en orden. El 45 se comparo con todos los números, después seguía el 2 a hacer lo mismo que el 45, después el 0, etc...

Su complejidad es O(n2)

En el código se ejecuta dos veces porque hay dos elementos mayores, simultáneamente los demás se agregan en la posición correcta. De ahí su complejidad.

En la primera iteración de las líneas de código que estan subrayadas hemos especificados el 9 sera el 2, y el 2 sera el 9, para que así se cambien los valores de la lista.

Debajo hay una representacion:

def bubble\_sort(elements):

    size = len(elements)

    for k in range(2):

        for i in range(size - 1):

            if elements[i] > elements[i+1]:

                tmp = elements[i]

                elements[i] = elements[i+1]

                elements[i+1] = tmp

elements = [5,9,2,1,67,34,88,34]

bubble\_sort(elements)

print(elements)

igual a 9

                tmp = elements[i]

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

9 es igual al 2

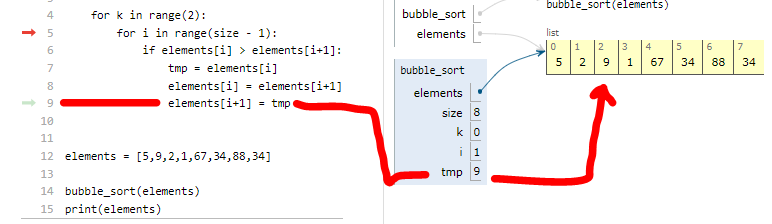
                elements[i] = elements[i+1]

Interfaz de usuario gráfica, Aplicación, Word

Descripción generada automáticamente

El 2 es igual al 9

                elements[i+1] = tmp



ASI ES COMO SE CAMBIAN LOS VALORES DENTRO DE UNA LSITA, CON UNA REGLA DE TRES DE VARIABLES.

Esto es una prueba de que la cantidad de ciclos del primer FOR es equivalente a la cantidad de números mayores:

En este ejemplo tenemos dos números mayores: [67, 88] en este el primer FOR es igual a 2.

    for k in range(2):



Si agregamos el 100, teniendo 2 iteraciones del primer FOR: (el 67 quedo al aire, y fuera del orden)



En este ejemplo tenemos tres números mayores: [67, 88, 100] mientras que en este el FOR debe ser igual a 3.

    for k in range(3):



La forma de solucionar esto es poniendo la cantidad de elementos de la lista, como un rango iterable:

La primera línea de código del FOR hará lo que hará es iterara, todos los números por separado x cantidad de veces, por ejemplo, el numero 9 se iterara, 7 veces, lo mismo con los demás.

El 5 no siguió la iteración del segundo FOR por que, no cumplía con el if, es decir, si no cumple con el IF, el segundo LOOP FOR cambia a otro número.

def bubble\_sort(elements):

    size = len(elements)

    for k in range(size - 1):

        for i in range(size - 1):

            if elements[i] > elements[i+1]:

                tmp = elements[i]

                elements[i] = elements[i+1]

                elements[i+1] = tmp

elements = [5,9,2,1,67,100,34,88,34]

bubble\_sort(elements)

print(elements)

una forma de mejorar la eficacia es restando el parámetro del primer FOR en el segundo FOR.

Te ahorras unos pasos extras, en el visualizador de código paso de 185 pasos a 129 pasos, debido a que como se resta un numero iterable, se disminuye la cantidad de iteraciones.

def bubble\_sort(elements):

    size = len(elements)

    for k in range(size - 1):

        for i in range(size - 1 - k):

            if elements[i] > elements[i+1]:

                tmp = elements[i]

                elements[i] = elements[i+1]

                elements[i+1] = tmp

elements = [5,9,2,1,67,100,34,88,34]

bubble\_sort(elements)

print(elements)

con la siguiente implementación disminuimos el numero de iteraciones a 120.

Esto para que cuando el numero que se esta iterando, al momento de llegar a su posición, el swapped pasa a ser false, y no seguirá iterando el loop FOR numero 2.

def bubble\_sort(elements):

    size = len(elements)

    for i in range(size-1):

        swapped = False

        for j in range(size-1-i):

            if elements[j] > elements[j+1]:

                tmp = elements[j]

                elements[j] = elements[j+1]

                elements[j+1] = tmp

                swapped = True

        if not swapped:

            break

elements = [5,9,2,1,67,100,34,88,34]

bubble\_sort(elements)

print(elements)

EJERCICIO:

def bubble\_sort(elements, clave):

    size = len(elements)

    for i in range(size-1):

        swapped = False

        for idx in range(size-1-i):

            if elements[idx][clave] > elements[idx+1][clave]:

                tmp = elements[idx]

                elements[idx] = elements[idx+1]

                elements[idx+1] = tmp

                swapped = True

        if not swapped:

            break

'''

def bubble\_sort(elements, clave):

    size = len(elements)

    for i in range(size-1):

        swapped = False

        for j in range(size-1-i):

            if elements[j] > elements[j+1]:

                tmp = elements[j]

                elements[j] = elements[j+1]

                elements[j+1] = tmp

                swapped = True

        if not swapped:

            break

'''

elements = [

         { 'nombre': 'mona', 'cantidad\_transacción': 1000, 'dispositivo': 'iphone-10'},

         { 'nombre': 'dhaval', 'cantidad\_transacción': 400, 'dispositivo': 'píxel de Google'},

         { 'nombre': 'kathy', 'cantidad\_transacción': 200, 'dispositivo': 'vivo'},

         { 'nombre': 'aamir', 'cantidad\_transacción': 800, 'dispositivo': 'iphone-8'}]

bubble\_sort(elements, clave='cantidad\_transacción')

print(elements)