ANDERSSON DAVID SÁNCHEZ MÉNDEZ

DOCUMENTO TÉCNICO

Requisitos

Especificación1

Escriba una función recursiva que ordene de menor a mayor una lista de enteros basándose en la siguiente idea: coloque el elemento más pequeño en la primera ubicación, y luego ordene el resto del arreglo con una llamada recursiva.

Entrada

En la primera línea se digita la cantidad de números que se desea y en la segunda línea cada número de acuerdo a la cantidad puesta anteriormente, separado por espacio entre cada número. Se toman números enteros, pero la cantidad de números mayor a 0.

Salida

Se generan los números organizados ascendentemente en forma de lista.

Especificación2

Escribir una función recursiva que devuelva la suma de un subarreglo de N enteros, limitado por índices (i, j) en una lista de enteros L.

Entrada

En la primera línea se digita la cantidad de números que se desea, en la segunda línea cada número de acuerdo a la cantidad puesta anteriormente, separado por espacio entre cada número; y en la tercera línea se digitan el intervalo que se desea, también separado por espacio. Se toman números enteros, pero la cantidad de números mayor a 0.

Salida

Se genera la suma de los números que están en el intervalo.

Especificación3

Escribir una función y un programa que encuentre la suma de los enteros positivos pares desde N hasta 2.

Entrada

En la única línea se digita el número que sea entero positivo.

Salida

Se genera la suma de los números enteros positivos desde el número ingresado hasta 2.

Especificación4

Dada una función recursiva para MCD cómo

MCD = M si N =0
MCD = MCD (N, M mod N) si N <> 0

Escriba un programa que le permita al usuario ingresar los valores para M y N desde la consola. Una función recursiva es entonces llamada para calcular el MCD. El programa entonces imprime el valor para el MCD.

Entrada

En la primera línea se digitan los dos números a los cuales se va a calcular el MCD, separado por espacio.

Salida

Se muestra el máximo común divisor de los dos números.

Especificación5

Programe un método recursivo que transforme un número entero positivo a notación binaria.

Entrada

Se digita el número entero positivo que se quiere mostrar en binario en la primera línea.

Salida

Se muestra el número en notación binaria.

Especificación6

Programe un método recursivo que invierta los números de un arreglo de enteros.

Entrada

En la primera línea se digita la cantidad de números que se desea y en la segunda línea cada número de acuerdo a la cantidad puesta anteriormente, separado por espacio entre cada número. Se toman números enteros, pero la cantidad de números mayor a 0.

ANDERSSON DAVID SÁNCHEZ MÉNDEZ

Salida

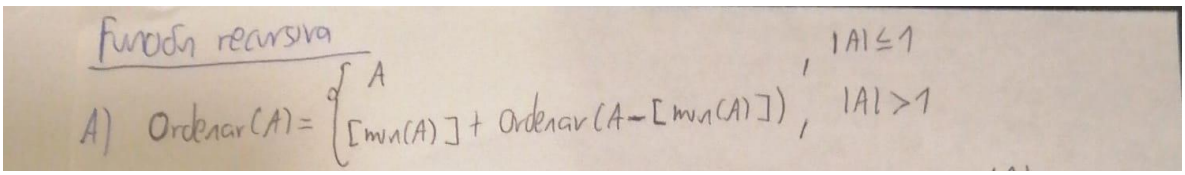
Se generan el arreglo invertido de los números en forma de lista.

Diseño

Estrategia1

La estrategia a implementar es encontrar una función que defina el problema en términos anteriores, es decir, se usa la recursión, ya después de encontrar la función, sólo queda plasmar eso en código fuente.

En este caso, se usa la recursión por pila, porque va desde encontrar el mínimo del arreglo y añadir lo que sobra, y así hasta que el arreglo tenga 1 elemento.

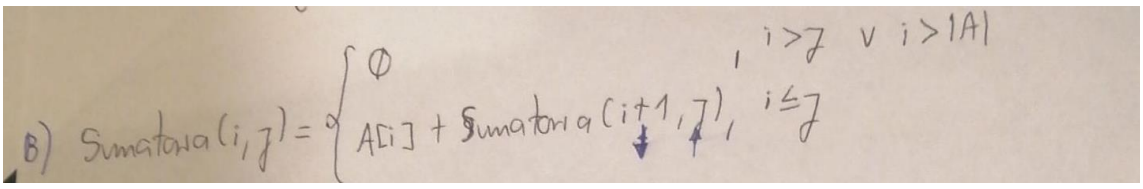


La estructura de datos a implementar será la lista, y se utilizará para guardar los inputs del usuario, y también para almacenar los números y la salida organizada ascendentemente.

Estrategia2

La estrategia a implementar es encontrar una función que defina el problema en términos anteriores, es decir, se usa la recursión, ya después de encontrar la función, sólo queda plasmar eso en código fuente.

En este caso, se usa la recursión por pila, porque va desde encontrar el primer intervalo del arreglo definido en la entrada hasta que llegue al intervalo final.

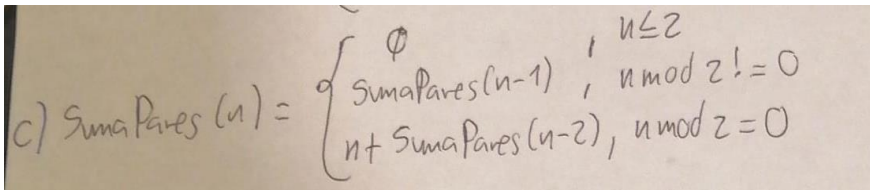


La estructura de datos a implementar será la lista, y se utilizará para guardar los inputs del usuario, y también para almacenar los números.

Estrategia3

La estrategia a implementar es encontrar una función que defina el problema en términos anteriores, es decir, se usa la recursión, ya después de encontrar la función, sólo queda plasmar eso en código fuente.

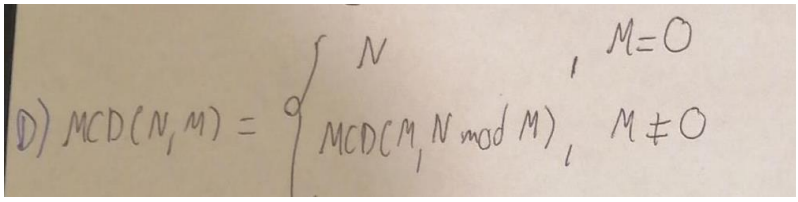
En este caso, se usa la recursión por cola, porque compara el número y va recorriendo en términos de posiciones anteriores.



Estrategia4

La estrategia a implementar es encontrar una función que defina el problema en términos anteriores, es decir, se usa la recursión, ya después de encontrar la función, sólo queda plasmar eso en código fuente.

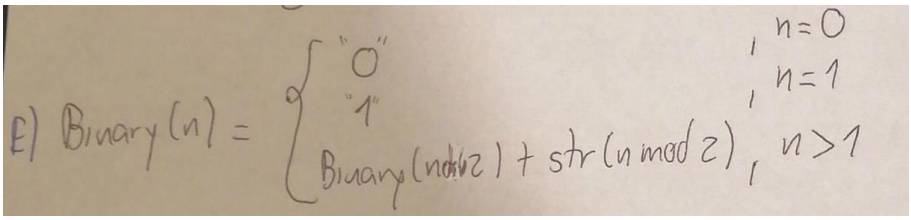
En este caso, se usa la recursión por cola, porque compara los dos números mediante un operador y va recorriendo en términos de posiciones anteriores.



Estrategia5

La estrategia a implementar es encontrar una función que defina el problema en términos anteriores, es decir, se usa la recursión, ya después de encontrar la función, sólo queda plasmar eso en código fuente.

En este caso, se usa la recursión por cola, porque compara el número cumpliendo la condición en términos anteriores para mostrar el número en notación binaria.



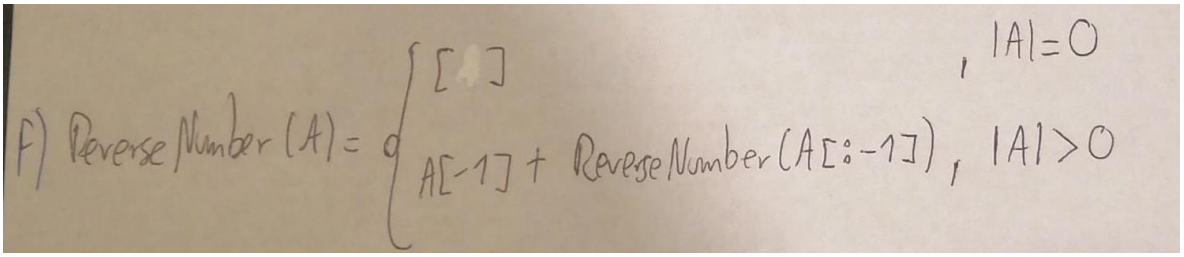
Da la salida como si la notación binaria fuera un string.

ANDERSSON DAVID SÁNCHEZ MÉNDEZ

Estrategia6

La estrategia a implementar es encontrar una función que defina el problema en términos anteriores, es decir, se usa la recursión, ya después de encontrar la función, sólo queda plasmar eso en código fuente.

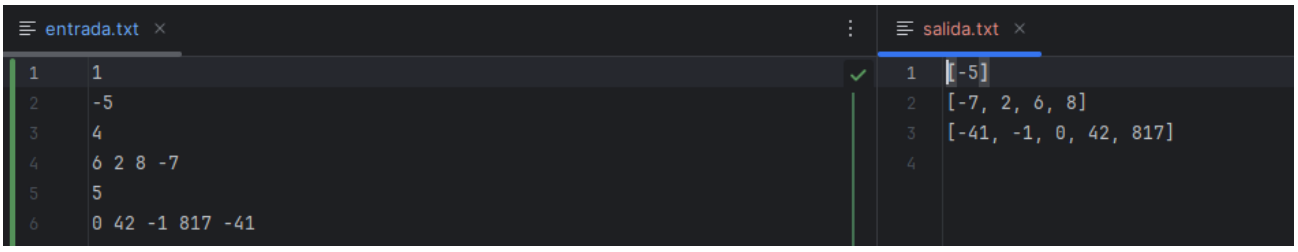
En este caso, se usa la recursión por cola, porque empieza tomando la última posición del arreglo y luego agrega lo que falta.



La estructura de datos a implementar será la lista, y se utilizará para guardar los inputs del usuario, y también para almacenar los números y la salida revertida.

Casos de prueba1

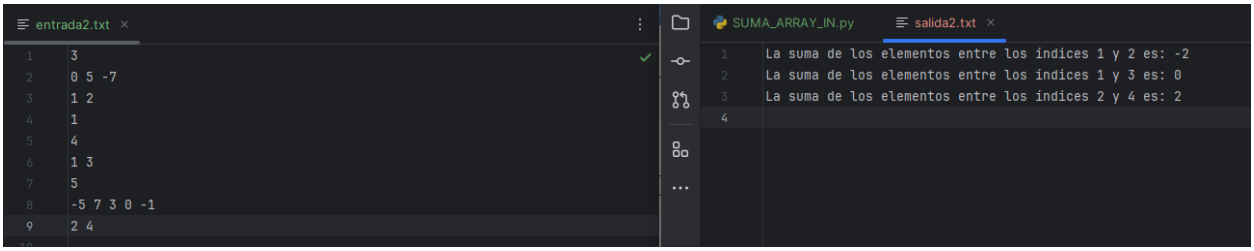
Entrada	Justificación	Salida
4 6 2 8 -7	Cualquier arreglo	[-7,2,6,8]
1 -5	Longitud 1	[-5]
5 0 42 -1 817 -41	Otro arreglo	[-41,-1,0,42,817]



ANDERSSON DAVID SÁNCHEZ MÉNDEZ

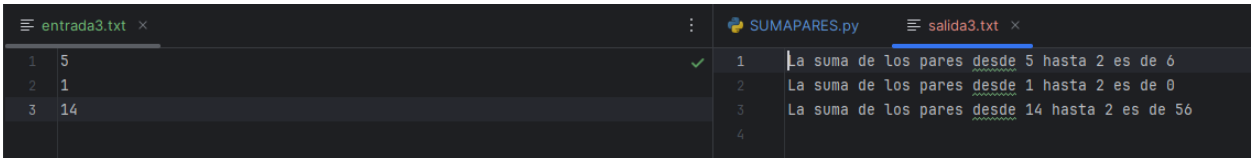
Casos de prueba2

Entrada	Justificación	Salida
3 0 5 -7 1 2	Cualquier arreglo	-2
1 4 1 3	Longitud 1	0
5 -5 7 3 0 -1 2 4	Otro arreglo	2



Casos de prueba3

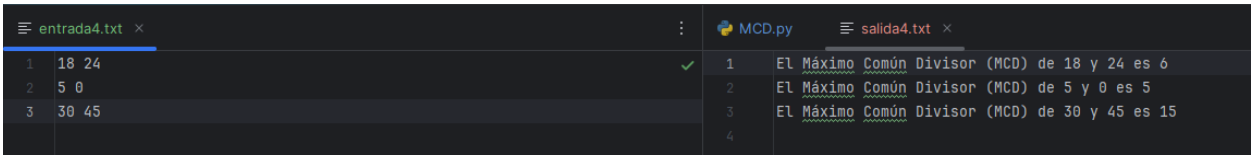
Entrada	Justificación	Salida
5	Cualquier número	6
1	Caso restrictivo	0
45	Otro número	56



ANDERSSON DAVID SÁNCHEZ MÉNDEZ

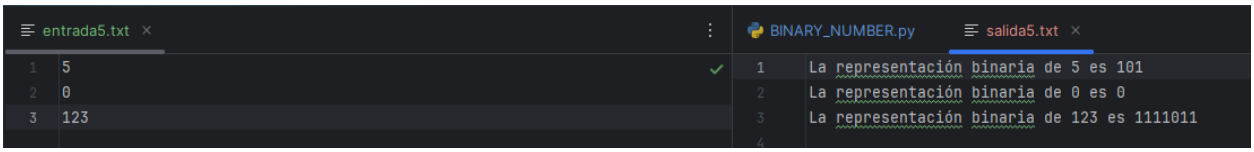
Casos de prueba4

Entrada	Justificación	Salida
18 24	Cualesquiera números	6
5 0	Caso restrictivo	5
30 45	Otros números	15



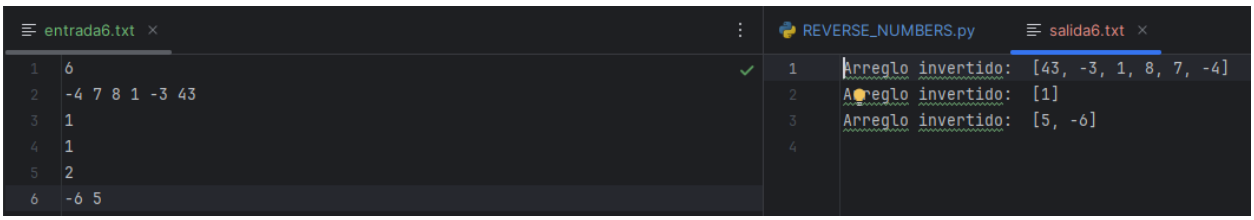
Casos de prueba5

Entrada	Justificación	Salida
5	Cualquier número	101
0	Caso restrictivo	0
123	Otro número	1111011



Casos de prueba6

Entrada	Justificación	Salida
6 -4 7 8 1 -3 43	Cualquier arreglo	[43,-3,1,8,7,-4]
1 1	Longitud arreglo 1	[1]
2 -6 5	Otro arreglo	[5,-6]



ANDERSSON DAVID SÁNCHEZ MÉNDEZ

Análisis

Temporal1

```
def menor_mayor(lista):                          #Cost (O)    #Times (O)  #Cost (Ohm)    #Times (Ohm)
    if len(lista) <= 1:                          #c1          n           c1             n
        return lista                             #c2          1           c2             1
    else:
        menorNumero = min(lista)                 #c3          1           c3             1
        lista.remove(menorNumero)                #c4          n-1         c4             n

    return [menorNumero] + menor_mayor(lista)    #c5          1           c5             1
                                                    #T(n)=O(n)                  #T(n)=Ohm(n)
```

Temporal2

```
def sumaElementos(lista, i, j):                  #Cost (O)    #Times (O)  #Cost (Ohm)    #Times (Ohm)
    if i > j or i >= len(lista):                 #c1          n           c1             n
        return 0                                 #c2          1           c2             1
    else:
        return lista[i] + sumaElementos(lista, i+1, j)   #c3  n           c3             n-1
                                                    # T(n)=O(n)                 #T(n)=Ohm(n)
```

Temporal3

```
def sumaPares(n):                                #Cost (O)    #Times (O)  #Cost (Ohm)    #Times (Ohm)
    if n < 2:                                    #c1          n           c1             n
        return 0                                 #c2          1           c2             1
    elif n % 2 != 0:                             #c3          n           c3             n
        return sumaPares(n-1)                    #c4          n-1         c4             n-1
    else:
        return n + sumaPares(n-2)                #c5          1           c5             1
                                                    #T(n)= O(n)                #T(n)= Ohm(n)
```

Temporal4

```
def mcd(M, N):                                   #Cost (O)    #Times (O)  #Cost (Ohm)    #Times (Ohm)
    if N == 0:                                   #c1          N           c1             N
        return M                                 #c2          1           c2             1
    else:
        return mcd(N, M % N)                     #c3          1           c3             1
                                                    # T(n)= O(N)                #T(n)= Ohm(N)
```

ANDERSSON DAVID SÁNCHEZ MÉNDEZ

Temporal5

```
def a_binario(n):                                   #Cost (O)    #Times (O)  #Cost (Ohm)    #Times (Ohm)
    if n == 0:                                      #c1              n           c1              n
        return '0'                                  #c2              1           c2              1
    elif n == 1:                                    #c3              n           c3              n
        return '1'                                  #c4              1           c4              1
    else:
        return a_binario(n // 2) + str(n % 2)       #c5              n           c5              n/2
                                                        # T(n)= O(n)                #T(n)= Ohm(n)
1 usage  new *
```

Temporal6

```
2 usages  new *
def reverse_Array(list):                            #Cost (O)    #Times (O)  #Cost (Ohm)    #Times (Ohm)
    if len(list) == 0:                              #c1              n           c1              n
        return []                                   #c2              1           c2              1
    else:
        return [list[-1]] + reverse_Array(list[:-1])#c3              n-1         c3              n-1
                                                        # T(n)= O(n)                #T(n)= Ohm(n)
```

Código

Documentación1

```
"""Escriba una función recursiva que ordene de menor a mayor una lista de enteros
basándose en la siguiente idea: coloque el elemento más pequeño en la primera
ubicación, y luego ordene el resto del arreglo con una llamada recursiva. """

"""Librería para que los inputs se pongan en un archivo"""
import sys

new *
def carga_numeros():
    lista = []
    n = int(sys.stdin.readline().strip())
    numeros = sys.stdin.readline().strip().split()
                                                    #carga_numeros administra los datos(cantidad y números)
    for i in range(n):
        try:
            numero = int(numeros[i])
        except ValueError:
            continue

        lista.append(numero)

    return lista
```

ANDERSSON DAVID SÁNCHEZ MÉNDEZ

```
def menor_mayor(lista):                                 #Cost (O)    #Times (O)  #Cost (Ohm)    #Times (Ohm)
    if len(lista) <= 1:                                 #c1          n           c1             n
        return lista                                    #c2          1           c2             1
    else:
        menorNumero = min(lista)                        #c3          1           c3             1
        lista.remove(menorNumero)                       #c4          n-1         c4             n

    return [menorNumero] + menor_mayor(lista)           #c5          1           c5             1
                                                           #T(n)=O(n)                  #T(n)=Ohm(n)
```

```
def main():
    with open('entrada.txt', 'r') as f:  # Abre el archivo 'pruebas.txt' en modo lectura
        while True:
            try:
                line = next(f).strip()  # Lee la siguiente línea y elimina los espacios en blanco
                if not line:  # Si la línea está vacía, pasa a la siguiente línea
                    continue
                n = int(line)  # Convierte la línea en un número entero
                numeros = next(f).strip().split()  # Lee los números de la prueba
                lista = [int(num) for num in numeros]
                listaAscendente = menor_mayor(lista)
                print(listaAscendente)
            except StopIteration:  # Cuando no hay más pruebas, termina el bucle
                break
main()
```

Documentación2

```
"""Escribir una función recursiva que devuelva la suma de un subarreglo de N enteros,
limitado por indices (i,j)  en una lista de enteros L"""

import sys                              #Librería

new *
def número_Elementos():
    n=-1

    while n <= 0:
        try:
            n=int(sys.stdin.readline().strip())       #número_Elementos administra la cantidad de números
        except ValueError:
            continue
    return n
```

```
def carga_numeros(n):
    lista = []
    numeros = sys.stdin.readline().strip().split()

    for i in range(n):
        try:                                                #carga_numeros administra los números
            numero = int(numeros[i])
        except ValueError:
            continue

        lista.append(numero)

    return lista
```

```
def sumaElementos(lista, i, j):                          #Cost (O)    #Times (O)  #Cost (Ohm)    #Times (Ohm)
    if i > j or i >= len(lista):                         #c1            n           c1              n
        return 0                                         #c2            1           c2              1
    else:
        return lista[i] + sumaElementos(lista, i+1, j)   #c3            n           c3              n-1
                                                           # T(n)=O(n)                #T(n)=Ohm(n)
1 usage  new *
```

```
def main():
    with open('entrada2.txt', 'r') as f:  # Abre el archivo 'entradas.txt' en modo lectura
        while True:
            try:
                n = int(next(f))  # Lee la cantidad de números en la prueba
                numeros = next(f).strip().split()  # Lee los números de la prueba
                lista = [int(num) for num in numeros]
                i, j = map(int, next(f).strip().split())  # Lee los índices i y j
                suma = sumaElementos(lista, i, j)
                print("La suma de los elementos entre los indices", i, "y", j, "es:", suma)
            except StopIteration:  # Cuando no hay más pruebas, termina el bucle
                break
main()
```

Documentación3

```
"""Escribir una función y un programa que encuentre la suma de los enteros positivos pares desde N hasta 2."""

import sys                #Librería

3 usages  new *
def sumaPares(n):                          #Cost (O)    #Times (O)  #Cost (Ohm)    #Times (Ohm)
    if n < 2:                              #c1            n           c1              n
        return 0                           #c2            1           c2              1
    elif n % 2 != 0:                       #c3            n           c3              n
        return sumaPares(n-1)              #c4           n-1          c4              n-1
    else:
        return n + sumaPares(n-2)          #c5            1           c5              1
                                                #T(n)= O(n)                #T(n)= Ohm(n)
1 usage  new *
def main():
    for line in sys.stdin:
        try:
            n = int(line.strip())                    #Número a tener en cuenta
        except ValueError:
            continue
        sumaParesF = sumaPares(n)
        print("La suma de los pares desde",n,"hasta 2 es de",sumaParesF)

main()
```

ANDERSSON DAVID SÁNCHEZ MÉNDEZ

Documentación4

```
"""Dada una función recursiva para MCD cómo

MCD = M si N =0
MCD = MCD (N, M mod N) si N <> 0

Escriba un programa que le permita al usuario ingresar los valores para M y N desde la consola.
Una función recursiva es entonces llamada para calcular el MCD. El programa entonces imprime el valor para el MCD."""

import sys          #Libreria

2 usages  new *
def mcd(M, N):                              #Cost (O)     #Times (O)  #Cost (Ohm)     #Times (Ohm)
    if N == 0:                              #c1             N           c1              N
        return M                            #c2             1           c2              1
    else:
        return mcd(N, M % N)                #c3             1           c3              1
                                                # T(n)= O(N)                #T(n)= Ohm(N)
1 usage  new *
def main():
    for line in sys.stdin:
        try:
            M, N = map(int, line.strip().split())           #Los números para calcular MCD
        except ValueError:
            continue
        result = mcd(M, N)
        print("El Máximo Común Divisor (MCD) de", M, "y", N, "es", result)
main()
```

Documentación5

```
"Programe un método recursivo que transforme un número entero positivo a notación binaria."

import sys              #Libreria

2 usages  new *
def a_binario(n):                                       #Cost (O)     #Times (O)  #Cost (Ohm)     #Times (Ohm)
    if n == 0:                                          #c1             n           c1              n
        return '0'                                      #c2             1           c2              1
    elif n == 1:                                        #c3             n           c3              n
        return '1'                                      #c4             1           c4              1
    else:
        return a_binario(n // 2) + str(n % 2)           #c5             n           c5              n/2
                                                            # T(n)= O(n)                #T(n)= Ohm(n)
1 usage  new *
def main():
    for line in sys.stdin:
        try:
            n = int(line.strip())                               #Número para pasar a notación binaria
        except ValueError:
            continue
        resultado = a_binario(n)
        print("La representación binaria de", n, "es", resultado)

main()
```

ANDERSSON DAVID SÁNCHEZ MÉNDEZ

Documentación6

```python
"Programe un método recursivo que invierta los números de un arreglo de enteros"

import sys                #Librería

new *
def lista():
    lista = []
    n = int(sys.stdin.readline().strip())                          #lista tiene cantidad y los números del arreglo
    numeros = sys.stdin.readline().strip().split()

    for i in range(n):
        try:
            numero = int(numeros[i])
        except ValueError:
            continue

        lista.append(numero)

    return lista
```

```python
2 usages  new *
def reverse_Array(list):                                #Cost (0)    #Times (0)  #Cost (Ohm)    #Times (Ohm)
    if len(list) == 0:                                  #c1            n           c1            n
        return []                                       #c2            1           c2            1
    else:
        return [list[-1]] + reverse_Array(list[:-1])    #c3            n-1         c3            n-1
                                                          # T(n)= O(n)              #T(n)= Ohm(n)
```

```python
1 usage  new *
def main():
    with open('entradaó.txt', 'r') as f:  # Abre el archivo 'entradas.txt' en modo lectura
        while True:
            try:
                line = next(f).strip()  # Lee la siguiente línea y elimina los espacios en blanco
                if not line:  # Si la línea está vacía, pasa a la siguiente línea
                    continue
                n = int(line)  # Convierte la línea en un número entero
                numeros = next(f).strip().split()  # Lee los números de la prueba
                lista = [int(num) for num in numeros]
                ArregloInvertido = reverse_Array(lista)
                print("Arreglo invertido: ", ArregloInvertido)
            except StopIteration:  # Cuando no hay más pruebas, termina el bucle
                break

main()
```

ANDERSSON DAVID SÁNCHEZ MÉNDEZ

Fuentes

/ MENOR-MAYOR.py	/entrada.txt	/salida.txt
/ SUMA_ARRAY_IN.py	/entrada2.txt	/salida2.txt
/ SUMAPARES.py	/entrada3.txt	/salida3.txt
/ MCD.py	/entrada4.txt	/salida4.txt
/ BINARY_NUMBER.py	/entrada5.txt	/salida5.txt
/ REVERSE_NUMBERS.py	/entrada6.txt	/salida6.txt