LABORATORIO # 1

PRESENTADO POR:

MIGUEL ANGEL SALAMANCA

JUAN CAMILO BAZURTO ARIAS

PRESENTADO A:

SEBASTIAN CAMILO MARTINEZ REYES

ESCUELA COLOMBIANA DE INGENIERÍA JULIO GARAVITO

ALGORITMOS Y ESTRUCTURAS DE DATOS

PROGRAMA DE INGENIERÍA DE SISTEMAS

BOGOTÁ D.C.

2021 – 1

1. Contexto

A. 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.

B. 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

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

D. 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.

E. Programe un método recursivo que transforme un número entero positivo a notación binaria.
Ejemplo: 5 - 101

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

H. Para la implementación de Merge en el algoritmo merge_sort visto en clase, haga el análisis temporal por método de factores con función característica para el peor y el mejor de los casos.

2. Requisitos

2.1. Especificación

2.1.1. Entrada

A. Esta función recibe como parámetro de entrada una secuencia o arreglo de números enteros y un índice que me indica la posición a organizar.
B. Esta función recibe como parámetro de entrada una secuencia o arreglo de números enteros y dos índices que indican los limites de la secuencia o arreglo.
C. Esta función recibe como parámetro de entrada un numero entero.

D. Esta función recibe como parámetro de entrada dos enteros M y N donde MCD = M si N =0 MCD = MCD (N, M mod N) si N < > 0
E. Esta función recibe como parámetro de entrada un entero para transformar a binario.
F. Esta función recibe como parámetro una secuencia o arreglo y un índice entero de donde se parte para invertir el arreglo.

2.1.2. Salida

A. La función retorna el arreglo ordenado de menor a mayor.
B. La función retorna un entero el cual es la suma de un arreglo limitado por índices.
C. La función retorna un entero el cual es la suma de los pares desde n hasta 2.
D. La función retorna el valor de MCD para los enteros ingresados.
E. La función retorna la notación binaria de un entero positivo.
F. La función retorna un arreglo el cual es el invertido del que ingreso por parámetro.

3. Diseño

3.1. Estrategia

3.1.1. Descripción general

A. Se plantea el caso base cuando el índice es igual a la longitud del arreglo y este ya está ordenado, para el caso inductivo, se encuentra el menor de los números del recorte del arreglo desde el índice, si el numero menor de ese subarreglo es diferente al elemento del arreglo en la posición del índice entonces, se intercambian, para luego retornar el nuevo arreglo y aumentar el índice. Si no es diferente significa que el menor de los números está en la posición correcta y se retorna el arreglo, aumentando el índice.
B. Se plantea el caso base cuando el índice ***i*** es igual al índice ***j***, entonces el entero a retornar ya será la suma de los enteros entre ***i*** y ***j***. En el caso

inductivo se retorna la suma entre el arreglo en la posición que indica el índice ***i*** y el entero que retorna el llamado a la función sumSub, la cual tiene los mismos parámetros ingresados al inicio, con diferencia que se aumenta el índice ***i***.

C. Se plantea el caso base cuando n es igual a dos ya que la función requiere que se sumen los pares de n hasta dos, para entonces retornamos el entero dos. Para el caso inductivo, si n es un numero par retornamos la suma entre n y sumPares, función la cual tiene como parámetros la diferencia entre n y dos. Si n no es par, entonces, retornamos la suma entre la diferencia de n y uno, y sumPares, función la cual tiene como parámetros la diferencia entre n y tres, si n menos tres es mayor o igual a dos, de caso contrario a la diferencia de n y uno se le suma cero.
D. Se plantea el caso base cuando n es igual a 0 donde se retorna m, y cuando n es igual a 1 donde se retorna 1, para el caso inductivo se retorna la función MCD, que tiene como parámetros a n y al módulo entre m y n.
E. Se plantea el caso base cuando la división entre el entero n y dos, es igual a cero, por lo tanto, se retorna la concatenación de todos los elementos anteriores invertida. Para el caso inductivo a la variable ***bin*** se le concatena el casteo a string del modulo entre n y dos. Para luego retorna el llamado a la función int_to_bin, con los parámetros, n división entera entre dos y la variable ***bin***.
F. Se plantea el caso base cuando el índice es igual la mitad entera de la longitud del arreglo, se retorna el arreglo ya invertido. Para el caso inductivo, asignamos a la variable ***elemento*** lo que contiene el arreglo en la posición del signo opuesto de índice (***-index***) menos uno (esto con el fin de recorrer el arreglo de atrás para adelante), y al elemento que hay en el arreglo en la posición del signo opuesto de índice menos uno (***-index - 1***), lo que contiene el arreglo en la posición del índice. El arreglo en la posición del índice pasa a ser la variable ***elemento***, y aumentamos

el índice, para luego retornar invertir, con los parámetros, arreglo e índice.

3.2. Casos de Prueba

A.

Como casos de prueba se tomaron los siguientes casos:

- [80, 717, 384, 597, 277, 24, 245, 46]

Los resultados de estos casos fueron los siguientes:

- [24, 46, 80, 245, 277, 384, 597, 717]

B.

Como casos de prueba se tomaron los siguientes casos:

- [359, 599, 79, 73, 924]

Los resultados de estos casos fueron los siguientes:

- 1076

C.

Como casos de prueba se tomaron los siguientes casos:

- n = 12
- n = 19
- n = 5

Los resultados de estos casos fueron los siguientes:

- 42
- 90
- 6

D.

Como casos de prueba se tomaron los siguientes casos:

- 450 360

Los resultados de estos casos fueron los siguientes:

- 90

E.

Como casos de prueba se tomaron los siguientes casos:

- 60
- 13
- 99

Los resultados de estos casos fueron los siguientes:

- 111100
- 1101
- 1100011

F.

Como casos de prueba se tomaron los siguientes casos:

- [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

Los resultados de estos casos fueron los siguientes:

- [9, 8, 7, 6, 5, 4, 3, 2, 1, 0]

4. Análisis

4.1. Temporal

A.

```
def minM(s, index):                                  #Cost          #Time
    if index == len(s):                              # C1           n + 1
        return s                                     # C2           1
    else:                                            # C3           n
        minNum = min(s[index:])                      # C4           n
        if minNum != s[index]:                       # C5           n
            pos = s.index(minNum)                    # C6           n - 1
            s[index], s[pos] = s[pos], s[index]      # C7           n - 1

            return minM(s, index + 1)                # C8           n - 1
        else:                                        # C9           1

            return minM(s, index + 1)                # C10          1

                                                     # Toral = C1 + C2 + n(C1 + C3 + C4 + C5 + C6 + C7 + C8) + C9 + C10 - C6 - C7 - C8
                                                     # O(n): lineal
```

B.

```
def sumSub(list, i, j):                    # Time        Cost
    if i == j:                             # c1           n + 1
        return list[i]                     # c2           1
    else:                                  # c3           n
        return list[i] + sumSub(list, i + 1, j)   # c4    n

                                           # Total: n(c1 + c3 + c4) + c1 +c2
                                           # O(n): Lineal
```

C.

```
def sumPares(n):                           # Time      Cost
    if n == 2:                             # c1        n + 1
        return 2                           # c2        1
    else:                                  # c3        n
        if n % 2 == 0:                     # c4        n
            return n + sumPares(n - 2)     # c5        n - 1
        else:                              # c6        n - 1
            return n - 1 + (sumPares(n - 3) if n - 3 >= 2 else 0)   # c7   n - 2

                                           # Total: n(c1 + c3 + c4 + c5 + c6 + c7) + c1 + c2 - c5 - c7 - 2*c7
                                           # O(n): lineal
```

D.

```
def MCD(m, n):                             # Time      Cost
    if n == 0:                             # c1        n + 1
        return m                           # c2        1
    elif n == 1:                           # c3        n
        return 1                           # c4        1
    else:                                  # c5        n
        return MCD(n, m % n)               # c6        n

                                           # Total: n(c1 + c3 + + c5 + c6) + c2 + c4 + c1
                                           # O(n): lineal
```

E.

```python
# Programe un método recursivo que transforme un número entero
# positivo a notación binaria.
def int_to_bin(n, bin=""):
    print(n)
    if n//2 == 0:
        return (bin+str(n % 2))[::-1]
    else:
        bin += str(n % 2)
        return int_to_bin(n//2, bin)


def main():
    print("El binario de", 100, "es", int_to_bin(100))
```

```
int_to_bin()
labE
C:\Documentos\python\Arenas\venv\Scripts\python.exe C:/Users/Andrew/Deskt
100
50
25
12
6
3
1
El binario de 100 es 1100100
```

Por los resultados obtenidos tenemos que esta función tiene un costo de logaritmo en base dos de ***n*** (O(lg2(n)))

F.

```python
def invertir(s, index=0):                                   # Time      Cost
    if index == len(s)//2:                                  # c1        n + 1

        return s                                            # c2        1
    else:                                                   # c3        n
        elemento, s[-index-1] = s[-index-1], s[index]       # c4        n
        s[index], index = elemento, index+1                 # c5        n

        return invertir(s, index)                           # c6        n

                                                            # Total: n(c1 + c3 + c4 + c5 + c6) + c1 + c2
                                                            # O(n): lineal
```