UNIVERSIDAD DE PANAMÁ

FACULTAD DE INFORMÁTICA, ELECTRÓNICA Y COMUNICACIÓN

ESCUELA DE INGENIERÍA DE INFORMÁTICA

COMPUTABILIDAD Y COMPLEJIDAD DE ALGORITMO

LABORATORIO #3

DIVIDE Y VENCERÁS

INTEGRANTES:

JESÚS DE GRACIA / 8-1086-1646

GISELA OJO / 8-904-2058

PROFESOR

AYAX MENDOZA

FECHA DE ENTREGA

13 DE NOVIEMBRE DE 2020

Laboratorio #3

Divide y Vencerás

Se realizó un script para cada método de divide y vencerás

Búsqueda binaria:

Este algoritmo consiste en encontrar la posición de un valor comparándolo con el elemento del medio del array. Si dicho valor no es igual, la mitad en la cual el valor no puede estar es eliminada y la búsqueda continúa con la mitad restante hasta que el valor se encuentre.

LAB3-CCA2020-main > LAB3-CCA2020 > BusquedaBinaria.py

```python

from random import random

N = 10
lista = []
for x in range(N):
    lista.append(int(random()*100))

lista.sort()

print("\n>>>>>>>>>>>>>>>>>>>>>>>>>> B U S Q U E D A   B I N A R I A <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<\n")
print("LISTA GENERADA")
print(lista)

num = int(input("\nEscriba el numero que desea saber su posición mediante busqueda binaria: "))

mini = 0
maxi = N-1

while mini <= maxi:
    med = (mini + maxi) //2
    if num < lista[med]:
        maxi = med-1
    elif num > lista[med]:
        mini = med+1
    else:
        print("\nEl numero está localizado en la posicion: ",med)
        break
else:
    print("\nEste numero no esta en la lista\n")

```

MergeSort:

En este algoritmo se utiliza la recursividad de un conjunto de elementos que se dividen entre dos ordenandos cada parte de forma separada y al final se combinan los dos resultados en el arreglo original.

```python
from random import random

#INPUT:-
print("\n>>>>>>>>>>>>>>>>>>>>>>>>>> M E R G E   S O R T <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<\n")
def mergeSort(lista):
    print("dividiendo elementos: ",lista)
    if len(lista)>1:
        med = len(lista)//2
        izqmitad = lista[:med]
        dermitad = lista[med:]

        mergeSort(izqmitad)
        mergeSort(dermitad)

        i=0
        j=0
        k=0
        while i < len(izqmitad) and j < len(dermitad):
            if izqmitad[i] <= dermitad[j]:
                lista[k]=izqmitad[i]
                i=i+1
            else:
                lista[k]=dermitad[j]
                j=j+1
            k=k+1

        while i < len(izqmitad):
            lista[k]=izqmitad[i]
            i=i+1
            k=k+1
```

```
        while j < len(dermitad):
            lista[k]=dermitad[j]
            j=j+1
            k=k+1
    print("fusionando elementos: ",lista)
lista = []
n=10
for i in range(n):
    lista.append(int(random()*100))
print("\n Arreglo generado aleatoriamente: ",lista)
mergeSort(lista)
print("\n Arreglo final ordenado por Merge Sort: ",lista)
print("\n")
```

Mínimo/ máximo de un array

El máximo consiste en encontrar el elemento mayor de un conjunto de valores y el mínimo el menor de los elementos de dicho conjunto.

```
from random import random


INF = float('inf')
# Solución Divide y venceras para encontrar el número mínimo y máximo en una lista
def MinMax(A, izq, der, min, max):

    # si la lista contiene solo un elemento

    if izq == der:         # Comparacion Comun

        if min > A[der]:      # comparacion 1
            min = A[der]

        if max < A[izq]:       # comparacion 2
            max = A[izq]

        return min, max

    # si la lista contiene solo dos elementos

    if der - izq == 1:     # Comparacion Comun

        if A[izq] < A[der]:  # comparacion 1
            if min > A[izq]:   # comparacion 2
                min = A[izq]

            if max < A[der]:  # comparacion 3
                max = A[der]

        else:
            if min > A[der]:  # comparacion 2
                min = A[der]

            if max < A[izq]:   # comparacion 3
                max = A[izq]

        return min, max

    # encontrando el elemento medio
    med = (izq + der) // 2

    # recurrir a la sublista izquierda
    min, max = MinMax(A, izq, med, min, max)

    # recurrir a la sublista derecha
    min, max = MinMax(A, med + 1, der, min, max)

    return min, max


if __name__ == '__main__':
    # proceso de creación de arreglo
    A = []
    n = 10
    for i in range(n):
        A.append(int(random()*100))
    print("\n>>>>>>>>>>>>>>>>>>>>>>>>>>> M A X I M O   Y   M I N I M O   E N   U N   A R R A Y <<<<<<<<<<<<<<<<")
    print("\nArreglo generado aleatoriamente", A)
    # inicializar el elemento mínimo por infinito y el elemento máximo por menos infinito
    (min, max) = (INF, -INF)
    (min, max) = MinMax(A, 0, len(A) - 1, min, max)
    print("\nEl elemento minimo del arreglo es: ", min)
    print("El elemento maximo del arreglo es: ", max)
```

```
    A.sort()
    print("\nArreglo finalmente ordenado: ", A)
    print("\n")
```

QuickSort

Este algoritmo consiste en utilizar un pivote para dividir de un lado los valores menores al pivote y del otro lado los valores mayores al pivote, y así sucesivamente hasta lograr su ordenamiento.

```
from random import randint


def particion(arreglo, inicio, final):
    pivote = arreglo[inicio]
    menor = inicio + 1
    mayor = final
    while True:
        #elementos que son más altos que pivotee
        while menor <= mayor and arreglo[mayor] >= pivote:
            mayor = mayor - 1
        #elems that are menorer than pivote
        while menor <= mayor and arreglo[menor] <= pivote:
            menor = menor + 1
        if menor <= mayor:
            arreglo[menor], arreglo[mayor] = arreglo[mayor], arreglo[menor]
        else:
            break
    arreglo[inicio], arreglo[mayor] = arreglo[mayor], arreglo[inicio]
    return mayor


def quick_sort(arreglo, inicio, final):
    if inicio >= final:
        return None
    p = particion(arreglo, inicio, final)
    quick_sort(arreglo, inicio, p-1)
    quick_sort(arreglo, p+1, final)
```

```
def generator():
    l1 = list()
    print("\n>>>>>>>>>>>>>>>>>>>>>>>>> M E T O D O   Q U I C K S O R T<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<\n")
    n = int(input("Ingresa el tamaño del arreglo a ordenar: \n"))
    for i in range(n):
        i += 1
        l1.append(randint(0, 50))
        if i > n:
            break
    return l1
arreglo = generator()
print("\nArreglo generado aleatoriamente:", arreglo)
quick_sort(arreglo, 0, len(arreglo) - 1)
print("\nArrgelo Ordenado por Quicksort: ", arreglo)
print("\n")
```

Contiene un Menú:

LAB3-CCA2020-main > LAB3-CCA2020 > Menu.py

```
import os
from random import randint
from random import random

def menu():
    """
    Función que limpia la pantalla y muestra nuevamente el menu
    """
    os.system('cls') # NOTA para windows tienes que cambiar clear por cls
    print ("\n >>>>>>>>>> L A B O R A T O R I O  3  - - - -  D I V I D E  Y  V E N C E R A S <<<<<<<<<<\n")
    print (">>>>>>>>>>>>>>>>>>>>>>>>> Elaborado por Jesus De Gracia & Gisela Ojo <<<<<<<<<<<<<<<<<<<<<<<<<\n")
    print ("Selecciona una opción")
    print ("\t1 - Busqueda Binaria")
    print ("\t2 - Quick Sort")
    print ("\t3 - Maximo y Minimo de un Array")
    print ("\t4 - Merge Sort")
    print ("\t9 - salir")

while True:
# Las 4 opciones del menu - Divide y Venceras
    def BusquedaBinaria():
        N = 10
        lista = []
        for x in range(N):
            lista.append(int(random()*100))

        lista.sort()
```

EJECUCIÓN

menú

```
C:\Windows\py.exe
 >>>>>>>>>> L A B O R A T O R I O  3  - - - -  D I V I D E  Y  V E N C E R A S <<<<<<<<<<

>>>>>>>>>>>>>>>>>>>>>>>>> Elaborado por Jesus De Gracia & Gisela Ojo <<<<<<<<<<<<<<<<<<<<<<<<<

Selecciona una opción
        1 - Busqueda Binaria
        2 - Quick Sort
        3 - Maximo y Minimo de un Array
        4 - Merge Sort
        9 - salir
inserta un numero valor >>
```

Búsqueda binaria:

```
>>>>>>>>>>>>>>>>>>>>>>>> B U S Q U E D A   B I N A R I A <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

Arreglo generado aleatoriamente:  [10, 25, 26, 26, 49, 64, 87, 93, 98, 99]

Escriba el numero que desea saber su posición mediante busqueda binaria: 64

El numero está localizado en la posicion:  5

pulsa una tecla para continuar
```

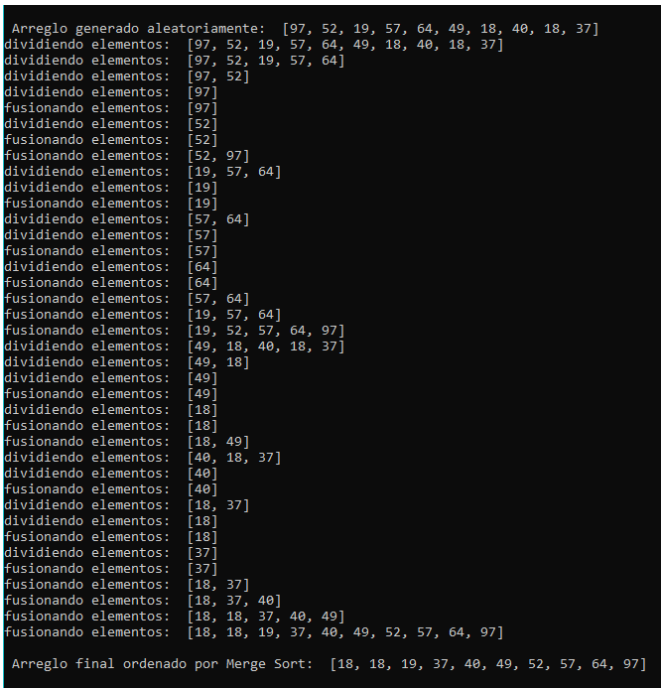
QuickSort

```
>>>>>>>>>>>>>>>>>>>>>>>> M E T O D O   Q U I C K S O R T<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

Ingresa el tamaño del arreglo a ordenar:
10

Arreglo generado aleatoriamente: [2, 43, 14, 27, 28, 1, 1, 1, 5, 37]

Arreglo Ordenado por Quicksort:  [1, 1, 1, 2, 5, 14, 27, 28, 37, 43]



pulsa una tecla para continuar
```

Mínimo/ máximo de un array

```
>>>>>>>>>>>>>>>>>>>>>>>> M I N I M O  Y  M A X I M O  E N  U N  A R R A Y <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

Arreglo generado aleatoriamente [65, 61, 91, 54, 15, 94, 60, 60, 21, 62]

El elemento minimo del arreglo es:  15
El elemento maximo del arreglo es:  94

Arreglo finalmente ordenado:  [15, 21, 54, 60, 60, 61, 62, 65, 91, 94]


pulsa una tecla para continuar
```

MergeSort:


Arreglo generado aleatoriamente: [97, 52, 19, 57, 64, 49, 18, 40, 18, 37]
dividiendo elementos: [97, 52, 19, 57, 64, 49, 18, 40, 18, 37]
dividiendo elementos: [97, 52, 19, 57, 64]
dividiendo elementos: [97, 52]
dividiendo elementos: [97]
fusionando elementos: [97]
dividiendo elementos: [52]
fusionando elementos: [52]
fusionando elementos: [52, 97]
dividiendo elementos: [19, 57, 64]
dividiendo elementos: [19]
fusionando elementos: [19]
dividiendo elementos: [57, 64]
dividiendo elementos: [57]
fusionando elementos: [57]
dividiendo elementos: [64]
fusionando elementos: [64]
fusionando elementos: [57, 64]
fusionando elementos: [19, 57, 64]
fusionando elementos: [19, 52, 57, 64, 97]
dividiendo elementos: [49, 18, 40, 18, 37]
dividiendo elementos: [49, 18]
dividiendo elementos: [49]
fusionando elementos: [49]
dividiendo elementos: [18]
fusionando elementos: [18]
fusionando elementos: [18, 49]
dividiendo elementos: [40, 18, 37]
dividiendo elementos: [40]
fusionando elementos: [40]
dividiendo elementos: [18, 37]
dividiendo elementos: [18]
fusionando elementos: [18]
dividiendo elementos: [37]
fusionando elementos: [37]
fusionando elementos: [18, 37]
fusionando elementos: [18, 37, 40]
fusionando elementos: [18, 18, 37, 40, 49]
fusionando elementos: [18, 18, 19, 37, 40, 49, 52, 57, 64, 97]
Arreglo final ordenado por Merge Sort: [18, 18, 19, 37, 40, 49, 52, 57, 64, 97]