**Entregable taller Ordenacion**

Juan Pablo Duque – 202110063010

Estructura de datos y Algoritmos 008

GitHub:

https://github.com/jpduquep/Taller-Ordenacion

**Taller**

1. (10)Se desea eliminar todos los números duplicados de una lista

Por ejemplo

Si se le ingresan los valores [4,7,11,4,9,5,11,7,3,5]

Se debe cambia a [4,7,11,9,5,3]

**1p.py**

1. (10)Elimine los elementos duplicados de un vector ordenado.

**2p.py**

¿Cuál es la eficiencia del método?

R/Eficiencia rápida con complejidad de O(n)

Compárela con la eficiencia del punto 1

R/ Este método tiene una mejor eficiencia ya que solo requiere de un ciclo for O(n), mientras que el anterior requiere de 2 ciclos for anidados O(n2), por lo cual habrá una diferencia notable en grandes listas.

1. (5)Dada la siguiente lista

[47,3,21,32.56,92]

Después de 2 “pasadas” de un algoritmo de ordenación, la lista ha quedado dispuesto asi

[3,21,47,32,56,92]

¿Qué algoritmo de ordenación se esta utilizando (selección, burbuja o inserción)?

Bubble Pasada 2 [3, 21, 32, 47, 56, 92]

Insercion Pasada 1 [3, 21, 32, 47, 56, 92]

Selectivo Pasada 2 [3, 21, 32, 47, 56, 92]

R/ No se puede saber con exactitud cual algoritmo de ordenamientos se esta utilizando ya que los 3 algoritmos en la 2da pasada llevan la lista de la misma manera

**5p.py**

1. (10) Utilizar el algoritmo de ordenación SHELL, encuentre las pasadas e intercambios que se realizan para la ordenación de la siguiente lista



**4p.py**

3 Pasadas del while principal

22 Pasadas del for anidado

1. (10) Dada una lista voto[0.......n-1], donde cada elemento de lista representa un voto en las elecciones. Suponga que cada voto se da como un número entero que representa el ID del candidato elegido. Desarrolle un algoritmo para determinar quién gana la elección. Determine la complejidad del algoritmo

**5p.py**

1. (10) Se cuenta con una lista de tuplas

futbolistasTup = [(1, "Casillas"), (15, "Ramos"), (3, "Pique"), (5, "Puyol"), (11, "Capdevila"), (14, "Xabi Alonso"), (16, "Busquets"), (8, "Xavi Hernandez"), (18, "Pedrito"), (6, "Iniesta"), (7, "Villa")]

si se aplica futbolistasTup.sort(key=lambda futbolista: futbolista[0])

1. Que resultado se obtiene al aplicar el método .sort
2. Que se esta especificando en los parámetro (key=lambda futbolista: futbolista[0])
3. Aplique este método a las listas de los punto 1,3, 4. Que conclusión puede obtener
4. Por favor según opinión realice una tupla con los mejores inventos del 2019 .

Donde usted califica el que mas le gusta o le parece importante.

**Anotación** la escala con la que usted cuenta es de 1 a 100 ( no tiene que asignar ninguno de los extremos si no lo desea)

1. (10) Diseñe e implemente una función para encontrar todos los valores negativos dentro de una lista dada. Tu función debería devolver una nueva lista que contiene los valores negativos.

**7p.py**

¿Cuándo ocurre el peor de los casos y cuál es el tiempo de ejecución para ese caso?

R/ El peor de los casos ocurre cuando toda la lista contiene números negativos, el tiempo de ejecución dependerá del largo de la lista, pero en ese caso será un O(n)

1. (5) Dada la siguiente lista de números:

[21, 1, 26, 45, 29, 28, 2, 9, 16, 49, 39, 27, 43, 34, 46, 40]

¿Cuál sería el resultado a la lista que después de 3 llamadas recursivas de ordenamiento por mezcla?

¿Por qué?

R/ Realmente es complejo de explicar como queda después de 3 llamadas recursivas ya que el algoritmo va dividiendo la lista en izquierda y derecha

Estrictamente hablando la lista que hay en la tercera llamada recursiva es: [21, 1, 26, 45]

**8p.py**

1. (5) Dado el siguiente código evalué su complejidad.

Determine que hace este esté algoritmo

Esplique cada uno de sus funciones y sus clases



1. (10) Construya un algoritmo para encontrar un valor específico en una matriz de valores ordenada por filas y columna.

El algoritmo toma como entrada una matriz de valores donde cada fila y cada columna están en orden, junto con un valor para ubicar en esa matriz. Devuelve si ese elemento existe en la matriz.

Por ejemplo, dado la siguiente matriz y buscar el 7, el algoritmo daría como resultado **sí**

Pero si se pide encontrar el número 0, el algoritmo daría como resultado **no**



**10p.py**

1. (10) se tiene una lista A con 100 elementos A[ a1……a100 ]

B de 60 elementos B[ b1……b60 ]

Se desean resolver las siguientes tareas

1. Ordenar cada lista aplicando el método Quicksort
2. Crear una lista C que sea la unión de la lista A y B
3. Ordenar la lista C y visualizarla

**11p.py**

1. (5) Investigue en que consisten las técnicas de ordenación Radixsort y Binsort, en lo posible haga un ejemplo

RadixSort =

Numero de pasadas: Va a ser igual a la cantidad de dígitos que tenga el numero con mas dígitos

Primero el algoritmo recorre la lista enlazada y se fija en el ultimo digito de cada numero y lo va colocando en una nueva lista enlazada de índice de dicho numero, de ahí vuelve y coloca todos los números en la lista enlazada en el orden en el cual hayan quedado.

LISTA ORIGINAL = [25]->[54]->[34]->[3]->[9]->[88]->NULL

PRIMERA ORGANIZACIÓN

0->

1->

2->

3->[3]->

4->[54]-> [34]->

5->[25]->

6->

7->

8->[88]->

9->[9]->

LISTA PRIMERA PASDA = [3]->[54]->[34]->[25]->[88]->[9]->NULL

Despues de aca vuelve y hace la pasada pero fijándose en el penúltimo digito, si el numero no cuenta con mas de 1 digito, se cuenta el 0 que tienen por defecto a la izquierda, y vuelva y hace la pasada

0->[03]-> [09]->

1->

2->[25]->

3->[34]->

4->

5->[54]->

6->

7->

8->[88]->

9->

LISTA SEGUNDA PASADA: [03]->[09]->[25]->[34]->[54]->[88]->NULL

En este caso como el numero con mas dígitos era de 2, en 2 pasada quedo organizado, sino simplemente se repite el proceso cada vez fijándose en el digito anterior y asi…

BinSort=

Mira el rango de los elementos, desde el elemento menor hasta el elemento mayor, y se crean unas “cubetas de ciertos rangos” que puedan incluir todos los elementos de la lista, de ahí cada “cubeta” es organizada con otro método, y ya se concatenan

LISTA ORIGINAL = [25]->[54]->[34]->[3]->[9]->[88]->NULL

Xmin: 9

Xmax: 88

“Cubetas”

3-33 ->[25]->[3]->[9]->

33-63 ->[54]->[34]->

64-88 ->[88]->

DESPUES DE OTROS METODOS DE ORGANIZACIÓN POR CUBETA

3-33 ->[3]->[9]->[25]->

33-63 ->[34]->[54]->

64-88 ->[88]->

CONCATENACION  
[03]->[09]->[25]->[34]->[54]->[88]->NULL