**Resumen**

Una de las aplicaciones más frecuentes, en programación es la ordenación.

• Existen dos técnicas de ordenación fundamentales en gestión de datos: ordenación de listas y ordenación de archivos.

• Los datos se pueden ordenar en orden ascendente o en orden descendente.

• Cada recorrido de los datos durante el proceso de ordenación se conoce como pasada o iteración.

• Los algoritmos de ordenación básicos son:

• Selección.

• Inserción.

• Burbuja.

• Los algoritmos de ordenación más avanzados son:

• Shell.

• Mergesort.

• Quicksort.

• La eficiencia de los algoritmos de burbuja, inserción y selección es 0(n2).

• La eficiencia de los algoritmos heapsort, radixsort, mergesort y quicksort es 0(n Log n).

• La búsqueda es el proceso de encontrar la posición de un elemento destino dentro de una lista. • Existen dos métodos básicos de búsqueda en arrays: búsqueda secuencial y binaria.

• La búsqueda secuencial se utiliza normalmente cuando el array no está ordenado. Comienza en el principio del array y busca hasta que se encuentra el dato buscado y se llega al final de la lista.

• Si un array está ordenado, se puede utilizar un algoritmo más eficiente denominado búsqueda binaria.

• La eficiencia de una búsqueda secuencial es 0(n).

• La eficiencia de una búsqueda binaria es 0(log n).

**Taller**

**Integrantes: Simón Gaviria, Andrés Leonardo Rojas y Daniel Pineda**

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]

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

¿Cuál es la eficiencia del método?

Compárela con la eficiencia del punto 1

El punto 2 es mas demorado, pero no por mucho, ya que el punto uno se ejectua en 0,011 segundos a comparación del 2 con 0,014 segundos

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)?

Justifique su respuesta:

1. En el código no puede estar implementado el algoritmo de ordenación burbuja, ya que este empieza comparando el primer elemento con el siguiente y cambiándolo si el segundo elemento es mayor que el primero, y así sucesivamente con todos los números de la lista, y solo con la primera pasada ya la lista quedaría ordenada de menor a mayor así:

[3,21,32,47,56,92] y eso no es lo que buscamos

1. En este ejercicio se utilizo el algoritmo de ordenación por inserción, ya que en la primera pasada queda así:

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

Y en la segunda ya queda ordenado como no lo pide el ejercicio:

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

Y este fue nuestro código:

def ordenamiento\_por\_inserccion(lista):

 for indice in range(1,len(lista)):

     valor =lista[indice]

     i=indice - 1

     while i>= 0:

         if valor < lista[i]:

          lista[i+1] =lista [i]

          lista[i] = valor

          i = i - 1

          print (lista)

         else:

             break

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

print(d)

ordenamiento\_por\_inserccion(d)

print(d)

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



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
2. (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

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

1. Que se esta especificando en los parámetro (key=lambda futbolista: futbolista[0])

En el parámetro se esta especificando el valor por el cual queremos ordenar la lista, en el ejemplo el número “0” representa la primera posición de la tupla por lo tanto la lista se ordena según el numero de camiseta de los jugadores, si el numero fuera “1” estos se ordenarían según su nombre en el alfabeto.

1. Aplique este método a las listas de los punto 1,3, 4. Que conclusión puede obtener

Al aplicar el método .Sort en los puntos 1-3-4 podemos concluir que estas listas de números se ordenan de menor a mayor, adjunto ejemplo con el punto 4

L=[8,43,17,6,40,16,18,97,11,7]

orden= sorted(L)

print(orden)

output: [6, 7, 8, 11, 16, 17, 18, 40, 43, 97]

1. 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)

(10,'Silla Herman Miller Cosm'),(30, 'LighSail 2'),(50, 'Analogue Mega SG'),(60, 'Butterfly iQ'),(70, 'bzt'),(80, 'Mergadata'),(90, ' Helm personal server'),( 20, 'Loop')

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.

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

* El peor de los casos ocurre cuando todos los elementos son negativos, ya que la función tendrá que recorrer toda la lista y añadirla a una nueva haciendo la n iteraciones.
* En un caso aleatorio con la lista implementada la duración de la ejecución es de 0.008007287979125977 segundos, y en el peor de los casos su duración es de 0.008014440536499023

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?

[21] y [1]

Corresponde a [21] y [1] ya que primeramente dividimos la lista de 16 datos en dos listas de 8 datos cada una, en el segundo paso dividimos una de las lista dos grupos de 4 y en este punto tendríamos los números [21,1,26,45] a la hora de la tercera pasada volvemos a dividir la lista y tendríamos los números [21,1] después de esto la función de ordenamiento por mezcla continuará moviéndose recursivamente hacia el inicio de la lista hasta que se encuentre con el caso base.

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

Determine que hace este esté algoritmo

Explique cada uno de sus funciones y sus clases



La complejidad de este código es de O(n) ya que en el peor de los casos el código se ejecutaría n veces, gracias al ciclo while

- al llamar la clase se inicia una lista

- el método len me dice la longitud de la lista

- el método contains determina si un elemento está en la lista ingresada

- el método add sirve para añadir elementos a la lista, pero con la condición de que estos no sean repetidos

- el método remove se usa para eliminar un elemento con la condición de que el elemento este en la lista

- el método isSubsetOf compara la lista ingresada y mira si pertenece a un conjunto creado

- el método iter se utiliza para convertir la lista usada en set(), es decir en un conjunto

- el método findPosition me dice en que posición esta el elemento ingresado de la lista dad

En conclusion esta clase convierte en conjunto la lista ingresada asi que se ordena, ademas tiene la capacidad de añadir, eliminar o encontrar la posicion de cualquier elemento que este en la lista dada

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**



Complejidad O(n2)

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

Radixsort:Radixsort es un algoritmo de ordenación de enteros que ordena los datos con claves enteras agrupando las claves por dígitos individuales que comparten la misma posición y valor significativo (valor de posición). La ordenación radix utiliza la ordenación por recuento como subrutina para ordenar una matriz de números. Dado que los enteros pueden utilizarse para representar cadenas (mediante el hash de las cadenas a enteros), la ordenación radix funciona con otros tipos de datos además de los enteros.

Binsort:

Binsort en Python se define como un tipo de algoritmo de comparación que se utiliza para ordenar los datos o elementos que se distribuyen de manera uniforme con un cierto rango en el que los elementos dados de la matriz o lista se asignan en cubos o contenedores donde más tarde estos elementos en los cubos se ordenan en orden y tal disposición de los elementos dados se conoce como ordenación de cubos que también se conoce como ordenación de cubos. En general, el algoritmo de ordenación de cubos se define como la organización de los elementos dispersos que se asignan en los cubos se clasifican en un orden particular, y luego estos elementos dispersos se reúnen y se ordenan dando lugar a una nueva lista.

Traducción realizada con la versión gratuita del traductor www.DeepL.com/Translator