PIMB - Programación Imperativa Básica
Taller 6 – Ordenamiento y Búsqueda

Nombres: _____ **Fecha:** 12– Abril – 2018 **Grupo:** 01

Instrucciones

- La entrega se realizará por Moodle el 19 de abril. Debe subir un archivo comprimido con los ejercicios del taller.
- Cualquier intento de copia tendrá como consecuencia la anulación del taller y el inicio del proceso académico correspondiente (Para ambas partes).

1. Un algoritmo de ordenamiento burbuja puede modificarse para que "*burbujee*" en ambas direcciones. La primera pasada mueve la lista "*hacia arriba*" y la segunda pasada mueve la lista "*hacia abajo*". Este patrón alternante continúa hasta que no son necesarias más pasadas. Implemente esta variación.

2. Modifique el algoritmo de ordenamiento por inserción para que funcione de forma bidireccional, es decir, manteniendo dos sublistas ordenadas: una en las posiciones inferiores y otra en las posiciones superiores. En cada pasada su algoritmo deberá insertar un elemento en la lista inferior y un elemento en la lista superior. Implemente esta variación. ¿Qué ventajas tendría esto sobre el algoritmo original?

3. Modifique el algoritmo de ordenamiento *merge* para que no divida la lista en mitades sino en tres partes iguales. Implemente esta variación. ¿Qué ventajas tendría esto sobre el algoritmo original?

4. Implemente el algoritmo de ordenamiento *quick* y explique su funcionamiento.

5. Se conoce que en una lista de *n-1* enteros sus valores van desde *1* hasta *n.* También se sabe que en ella no hay elementos duplicados y hay un elemento faltante. Escriba un programa en Python que permita encontrar dicho elemento.

Ejemplo:

ENTRADA: [1, 2, 4, 6, 3, 7, 8] SALIDA: 5

6. Dadas dos listas ordenadas de tamaño *n* cada una, escriba un programa en Python que permita encontrar la mediana de la lista obtenida después de unir ordenadamente las dos listas.

Ejemplo:

ENTRADA: [1, 12, 15, 26, 38] SALIDA: 16
[2, 13, 17, 30, 45]

Explicación:

Después de unir las dos listas se obtiene: [1, 2, 12, 13, 15, 17, 26, 30, 38, 45]. Los elementos en el medio son 15 y 17, por lo tanto se promedian estos valores y se obtiene 16.

7. Escriba un programa en Python que permita encontrar el elemento más pequeño y el segundo más pequeño de una lista.

Ejemplo:

ENTRADA: [12, 13, 1, 10, 34, 1] SALIDA: 1 y 10

8. Dada una lista de enteros, escriba un programa en Python que permita encontrar los dos elementos cuya suma sea más cercana a cero.

Ejemplo:

ENTRADA: [1, 60, -10, 70, -80, 85] SALIDA: -80 y 85

9. Dada una lista de enteros y un número *n*, escriba un programa en Python que encuentre, si existe, una pareja

de elementos cuya diferencia sea *n.*

Ejemplo:

ENTRADA: [5, 20, 3, 2, 50, 80] SALIDA: 2 y 80
n = 78

ENTRADA: [90, 70, 20, 80, 50] SALIDA: Pareja no encontrada
n = 45

10. Dada una lista ordenada de enteros y un valor *x*, escriba un programa en Python que encuentre los *k* elementos más cercanos a *x.*

Ejemplo:

ENTRADA: [12, 16, 22, 30, 35, 39, 42, 45, 48, 50, 53, 55, 56]
x = 35
k = 4

SALIDA: 30 39 42 45

Nota:

Cuando el elemento está presente en la lista, este no deberá ser tenido en cuenta en la salida.