

Universidad Tecnológica de Panamá
Facultad de Sistemas Computacionales
Asignatura: Desarrollo Lógico y Algoritmo

Taller Práctico1

Profesor: Napoleón Ibarra

Valor: 100 puntos

Estudiante:

Cédula:

Fecha Inicio: 27/10/2025 --> 4:10 PM

Fecha Entrega: 28/10/2025 -->3:20 PM

Procedimiento:

1. De manera individual o en grupo de trabajo de 2 personas, realizar la asignación. Utilizando la herramienta Internet, investigue, desarrolle los conceptos solicitados.
2. Entregar el trabajo en formato digital(Parte I, II, III, IV) en PDF en la plataforma.

Criterios de Evaluación:

| Criterios | Puntos (Mínimo 1, Máximo 5) | Porcentaje |
|--------------|-----------------------------|------------|
| Sustentación | 1 - 5 | 15 % |
| Puntualidad | 1 - 5 | 15 % |
| Desarrollo | 1 - 5 | 70 % |

I PARTE. Investigación. Valor 15 puntos

Temas:

- 1) *Procedimiento de búsqueda y ordenamiento de un arreglo.*

BÚSQUEDA

Con mucha frecuencia los programadores trabajan con grandes cantidades de datos almacenados en arreglos y registros, y por ello será necesario determinar si un arreglo contiene un valor que coincida con un cierto valor clave.

El proceso de encontrar un elemento específico de un arreglo se denomina búsqueda.

ORDENAMIENTO

El ordenamiento o clasificación de datos (sort, en inglés) es una operación consistente en disponer un conjunto —estructura— de datos en algún determinado orden con respecto a uno de los campos de elementos del conjunto.

Una colección de datos (estructura) puede ser almacenada en un archivo, un array (vector o matriz), una lista enlazada o un árbol. Cuando los datos están almacenados en un array, una lista enlazada o un árbol, se denomina ordenación interna. Si los datos están almacenados en un archivo, el proceso de ordenación se llama ordenación externa.

Los métodos de ordenación se suelen dividir en dos grandes grupos:

Directos: Burbuja, selección, inserción

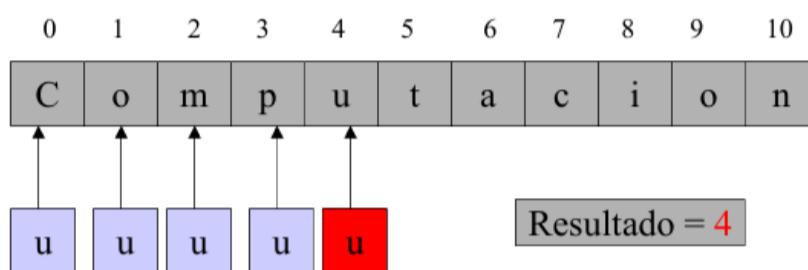
Indirectos (avanzados): Shell, QuickSort, ordenación por mezcla, Radixsort, en el caso de listas pequeñas, los métodos directos se muestran eficientes, sobre todo porque los algoritmos son sencillos; su uso es muy frecuente. Sin embargo, en listas grandes estos métodos se muestran ineficaces y es preciso recurrir a los métodos avanzados.

1.1. Búsqueda secuencial.

La búsqueda secuencial busca un elemento de una lista utilizando un valor destino llamado clave. En una búsqueda secuencial, los elementos de una lista o vector se exploran en secuencia, uno después de otro. La búsqueda secuencial es necesaria, por ejemplo, si se desea encontrar la persona cuyo número de teléfono es 64-234567 en la guía telefónica de Osorno. Las guías de teléfonos están organizadas alfabéticamente por el nombre del abonado en lugar de por números de teléfono, de modo que deben explorarse todos los números, uno después de otro, esperando encontrar el número requerido.

El algoritmo de búsqueda secuencial compara cada elemento del arreglo con la clave de búsqueda.

Dado que el arreglo no está en un orden prefijado, es probable que el elemento a buscar pueda ser el primer elemento, el último elemento o cualquier otro. De promedio, al menos el programa tendrá que comparar la clave de búsqueda con la mitad de los elementos del arreglo. El método de búsqueda secuencial funcionará bien con arreglos pequeños o no ordenados. La eficiencia de la búsqueda secuencial es pobre, tiene complejidad lineal, $O(n)$.



1.2. Push Down.

Algoritmo de pila (Stack): En la informática, una pila es una estructura de datos que opera bajo el principio "último en entrar, primero en salir" (LIFO). "Push down" podría referirse a la operación de insertar un elemento en la pila (push) o a una operación de pila específica.

Ordenamiento por inserción (Insertion sort): Este algoritmo "empuja" un elemento hacia abajo en un sub-arreglo ordenado hasta colocarlo en su posición correcta.

Búsqueda por división de la mitad (Binary search): A veces se puede referir a la forma en que el algoritmo de búsqueda binaria descarta la mitad del arreglo en cada paso, "empujando" la búsqueda hacia el rango restante.

2) *Base de Datos MySQL: Definición, ¿Qué se requiere para ser instalado?, ¿Qué se necesita para hacer una conexión PYTHON-MYSQL? Explique, ¿Qué es una replicación en una Base de Datos? Sustente su respuesta.*

Base de Datos MySQL.

Definición: MySQL es un sistema de gestión de bases de datos relacional (RDBMS) ha desempeñado un papel fundamental en la evolución de la tecnología de la información y ha sido un pilar esencial para aplicaciones web, empresas y proyectos de todo tipo.

¿Qué se requiere para ser instalado?

R/= La instalación de MySQL es un proceso relativamente sencillo. Los pasos exactos dependen del sistema operativo que estés utilizando.

En general, los pasos para instalar MySQL son los siguientes:

- Descarga el paquete de instalación de MySQL del sitio web oficial.
- Instala el paquete de instalación.
- Inicia el servidor MySQL.
- Crea un usuario y una contraseña para acceder a MySQL.

¿Qué se necesita para hacer una conexión PYTHON-MYSQL?

R/=

- Instalar el conector: Abre tu terminal o símbolo del sistema. Ejecuta el comando pip install mysql-connector-python para instalar el paquete oficial.
- Importar la biblioteca: En tu archivo de Python, importa la librería necesaria: import MySQL.connector.
- Establecer la conexión: Utiliza un bloque try-except para manejar posibles errores. Dentro del try, llama a MySQL.connector.connect() con los parámetros de tu base de datos:
- host: El nombre del servidor (por ejemplo, 'localhost').
- user: Tu nombre de usuario de MySQL.
- password: Tu contraseña.
- database: El nombre de la base de datos.
- port: El número de puerto (el predeterminado es 3306).
- Ejecutar consultas: Crea un objeto cursor a partir de la conexión: cursor = connection.cursor(). Usa cursor.execute() para ejecutar sentencias SQL.
- Confirmar cambios: Si realizas operaciones de modificación de datos (DML), confirma los cambios con connection.commit().
- Cerrar la conexión: Cuando termines, libera los recursos cerrando la conexión: connection.close().

Explique, ¿Qué es una replicación en una Base de Datos? Sustente su respuesta.

R/. La replicación en una base de datos es el proceso de copiar y mantener sincronizados los datos entre dos o más bases de datos que pueden estar en el mismo servidor o en diferentes ubicaciones físicas. En otras palabras, se trata de tener copias idénticas (réplicas) de una base de datos para mejorar la disponibilidad, rendimiento y seguridad de la información.

Procedimiento:

1. Utilice la técnica (a su criterio) para desarrollar el tema propuesto.
2. En su desarrollo (Ponencia) debe estar los siguientes puntos:
 - 2.1. *Un ejemplo (código sencillo funcional) de Arreglos. Su pseudocódigo y simulación.*
 - 2.2. *Concepto.*
 - 2.3. *Su sintaxis.*

pseudocodigo

INICIO

DEFINIR arreglo = [8, 3, 5, 1, 9]

MIENTRAS verdadero HACER

```
    MOSTRAR "===== MENÚ PRINCIPAL ====="  
    MOSTRAR "1. Mostrar arreglo actual"  
    MOSTRAR "2. Ordenar arreglo (Push Down)"  
    MOSTRAR "3. Buscar valor (Búsqueda Secuencial)"  
    MOSTRAR "4. Salir"  
    LEER opcion
```

SI opcion = 1 ENTONCES

 MOSTRAR "Arreglo actual: ", arreglo

FIN_SI

SI opcion = 2 ENTONCES

 PARA i DESDE 1 HASTA longitud(arreglo) - 1 HACER

 valor = arreglo[i]

 j = i - 1

 MIENTRAS j >= 0 Y arreglo[j] > valor HACER

 arreglo[j + 1] = arreglo[j] // Empujar hacia abajo

 j = j - 1

 FIN_MIENTRAS

 arreglo[j + 1] = valor

FIN_PARA

 MOSTRAR "Arreglo ordenado: ", arreglo

FIN_SI

SI opcion = 3 ENTONCES

 LEER valor

 encontrado = FALSO

 PARA i DESDE 0 HASTA longitud(arreglo) - 1 HACER

 SI arreglo[i] = valor ENTONCES

 MOSTRAR "Valor encontrado en la posición ", i

 encontrado = VERDADERO

 ROMPER

 FIN_SI

FIN_PARA

 SI encontrado = FALSO ENTONCES

 MOSTRAR "Valor no encontrado en el arreglo"

 FIN_SI

FIN_SI

SI opcion = 4 ENTONCES

 MOSTRAR "Saliendo del programa..."

 ROMPER

FIN_SI

SI opcion NO ES 1,2,3,4 ENTONCES

MOSTRAR "Opción inválida. Intente de nuevo."

FIN_SI

FIN_MIENTRAS

FIN

Código en Python

```
# ======  
# PROGRAMA: Búsqueda Secuencial y Ordenamiento Push Down  
# ======
```

```
def busqueda_secuencial(arreglo, valor):
```

"""

Busca un valor dentro de un arreglo recorriéndolo uno por uno.

Retorna la posición si lo encuentra, o -1 si no está.

"""

```
for i in range(len(arreglo)):
```

```
    if arreglo[i] == valor:
```

```
        return i
```

```
return -1
```

```
def push_down(arreglo):
```

"""

Ordena el arreglo de menor a mayor usando el método Push Down

(similar al método de inserción).

"""

```
for i in range(1, len(arreglo)):
```

```
    valor = arreglo[i]
```

```
    j = i - 1
```

```
    while j >= 0 and arreglo[j] > valor:
```

```
        arreglo[j + 1] = arreglo[j]
```

```
        j -= 1
```

```
arreglo[j + 1] = valor

# =====
# PROGRAMA PRINCIPAL
# =====

def main():

    arreglo = [8, 3, 5, 1, 9]

    while True:

        print("\n===== MENÚ PRINCIPAL =====")

        print("1. Mostrar arreglo actual")
        print("2. Ordenar arreglo (Push Down)")
        print("3. Buscar valor (Búsqueda Secuencial)")
        print("4. Salir")

        opcion = input("Seleccione una opción: ")

        if opcion == "1":

            print("Arreglo actual:", arreglo)

        elif opcion == "2":

            push_down(arreglo)

            print("✓ Arreglo ordenado:", arreglo)

        elif opcion == "3":

            valor = int(input("Ingrese el valor a buscar:"))

            posicion = busqueda_secuencial(arreglo, valor)

            if posicion != -1:

                print(f"✓ Valor encontrado en la posición {posicion}")

            else:

                print("✗ Valor no encontrado en el arreglo")

        elif opcion == "4":

            print("👋 Saliendo del programa...")

            break
```

```

else:
    print("⚠️ Opción inválida. Intente de nuevo.")

```

```

# Ejecutar el programa
if __name__ == "__main__":
    main()

```

