PRÁCTICA 1

REPORTE

ANÁLISIS Y DISEÑO DE ALGORITMOS

3CV1

SOLARES VELASCO ARTURO MISAEL
SOLIS LUGO MAYRA

DESARROLLO

Función buscar_un_arreglo(A, t):

- **Descripción:** Esta función recorre un arreglo **A** en busca de un elemento **t**. Si encuentra el elemento, devuelve **True**; de lo contrario, devuelve **False**.
- **Complejidad:** La complejidad de esta función es O(n), donde n es la longitud del arreglo **A**, ya que recorre cada elemento del arreglo una vez.

```
def buscar_un_arreglo(A, t):
    for i in range(len(A)):
        if A[i] == t:
            return True
    return False
```

```
def buscar_dos_arreglos(A, B, t):
    for i in range(len(A)):
        if A[i] == t:
            return True
    for i in range(len(B)):
        if B[i] == t:
            return True
    return False
```

Función buscar_dos_arreglos(A, B, t):

- **Descripción:** Esta función busca un elemento **t** en dos arreglos **A** y **B**. Si encuentra el elemento en alguno de los arreglos, devuelve **True**; de lo contrario, devuelve **False**.
- **Complejidad:** La complejidad de esta función es O(n), donde n es la suma de las longitudes de los arreglos **A** y **B**.

Función verificar_elemento_comun(A, B):

- **Descripción:** Esta función verifica si hay al menos un elemento común entre los arreglos **A** y **B**. Utiliza dos bucles anidados para comparar cada elemento de **A** con cada elemento de **B**.
- **Complejidad:** La complejidad de esta función es O(n^2), donde n es la suma de las longitudes de los arreglos **A** y **B**.

```
def verificar_elemento_comun(A, B):
    for i in range(len(A)):
        for j in range(len(B)):
            if A[i] == B[j]:
                return True
    return False
```

```
def verificar_duplicados(A):
    for i in range(len(A)):
        for j in range(i+1, len(A)):
            if A[i] == A[j]:
                return True
    return False
```

Función verificar_duplicados(A):

- **Descripción:** Esta función verifica si hay algún elemento duplicado en el arreglo **A**. Utiliza dos bucles anidados para comparar cada par de elementos en **A**.
- **Complejidad:** La complejidad de esta función es O(n^2), donde n es la longitud del arreglo **A**.

DESARROLLO

main():

1. **Generar arreglos aleatorios:**
   - Se define una lista **valores_n** que contiene los diferentes tamaños de arreglos que se generarán aleatoriamente.
   - Se inicializa una lista vacía **arreglos** que almacenará los arreglos generados.
   - Se itera sobre cada tamaño **n** en **valores_n** y se genera un arreglo aleatorio de tamaño **n** utilizando una comprensión de lista con **random.randint(1, 100)**.
   - Cada arreglo generado se agrega a la lista **arreglos**.
2. **Iterar sobre cada tamaño de arreglo y medir el tiempo de ejecución:**
   - Se itera sobre los índices de la lista **valores_n**.
   - Se obtiene el tamaño **n** del arreglo actual y el arreglo correspondiente de la lista **arreglos**.
   - Se mide el tiempo de ejecución de cada función de búsqueda (**buscar_un_arreglo**, **buscar_dos_arreglos**, **verificar_elemento_comun**, **verificar_duplicados**) utilizando la función **time.time()** antes y después de llamar a cada función.
   - Los tiempos de ejecución se calculan restando el tiempo de finalización del tiempo de inicio para obtener el tiempo total de ejecución de cada función.
3. **Almacenar los tiempos de ejecución en una lista de resultados:**
   - Se inicializa una lista vacía **resultados** que almacenará los tiempos de ejecución de cada función para cada tamaño de arreglo.
   - Para cada tamaño de arreglo, se crea una lista que contiene los tiempos de ejecución de cada función de búsqueda.
   - Esta lista se agrega a la lista **resultados**.
4. **Crear un DataFrame de pandas con los resultados y guardar en un archivo CSV:**
   - Se crea un DataFrame de pandas utilizando la lista **resultados** como datos y una lista de nombres de columnas como columnas.
   - Se agrega una columna adicional al DataFrame que contiene los tamaños de los arreglos correspondientes.
   - Se utiliza el método **to_csv()** para guardar el DataFrame en un archivo CSV en la ubicación **/home/artur/ESCOM/ADA/Practica_lab_1/results.csv**.
   - El parámetro **index=False** se utiliza para evitar que se incluya el índice de fila en el archivo CSV.
5. **Imprimir un mensaje de éxito:**
   - Se imprime un mensaje indicando que los resultados se guardaron exitosamente en el archivo CSV.

```python
def main():
    # Generar arreglos aleatorios
    valores_n = [10, 100, 1000, 10000, 1000000]
    arreglos = []
    for n in valores_n:
        arreglo = [random.randint(1, 100) for _ in range(n)]
        arreglos.append(arreglo)

    # Medir el tiempo de ejecución para cada algoritmo y tamaño de entrada
    resultados = []
    for i in range(len(valores_n)):
        n = valores_n[i]
        arreglo = arreglos[i]

        tiempo_inicio = time.time()
        buscar_un_arreglo(arreglo, 42)
        tiempo_fin = time.time()
        tiempo_buscar_un_arreglo = tiempo_fin - tiempo_inicio
        tiempo_inicio = time.time()
        buscar_dos_arreglos(arreglo, arreglo, 42)
        tiempo_fin = time.time()
        tiempo_buscar_dos_arreglos = tiempo_fin - tiempo_inicio

        tiempo_inicio = time.time()
        verificar_elemento_comun(arreglo, arreglo)
        tiempo_fin = time.time()
        tiempo_verificar_elemento_comun = tiempo_fin - tiempo_inicio

        tiempo_inicio = time.time()
        verificar_duplicados(arreglo)
        tiempo_fin = time.time()
        tiempo_verificar_duplicados = tiempo_fin - tiempo_inicio

        resultados.append([tiempo_buscar_un_arreglo, tiempo_buscar_dos_arreglos, tiempo_verificar_elemento_comun, tiempo_verificar_duplicados])

            # Crear un DataFrame de pandas para presentar los resultados
            df = pd.DataFrame(resultados, columns=['buscar un arreglo', 'buscar dos arreglos', 'verificar elemento comun', 'verificar duplicados'])
            df.insert(0, 'n', valores_n)
            # Guardar los resultados en un archivo CSV
            df.to_csv('/home/artur/ESCOM/ADA/Practica_lab_1/results.csv', index=False)

            print('Resultados guardados en el archivo CSV de manera exitosa.')

        if __name__ == '__main__':
            main()
```