Universidad Nacional Autónoma de México

# Tarea 3

**Introducción a Ciencias de la Computación**

Rodrigo André Decuir Fuentes

---

## Teoría

### Responde las siguientes preguntas:

1. **¿Qué son los arreglos y para qué son utiles?**
   - Los arreglos son conjuntos finitos y ordenados de elementos homogéneos, asimismo son objetos.
   - Son de mucha utilidad para manejar de forma sencilla y directa un conjunto de datos del mismo tipo.
2. **Se tiene el siguiente arreglo:**

| 2 | 35 | −22 | 0 | 56 | 78 | −84 | 8 | 43 | 1 |
|---|---|---|---|---|---|---|---|---|---|

Ordena el arreglo anterior por medio de Bubble Sort y Selection Sort

**Comportamiento Bubble Sort (ejecución):**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1. | 2 | 35 | −22 | 0 | 56 | 78 | −84 | 8 | 43 | 1 |
| 2. | 2 | −22 | 0 | 35 | 56 | −84 | 8 | 43 | 1 | 78 |
| 3. | −22 | 0 | 2 | 35 | −84 | 8 | 43 | 1 | 56 | 78 |
| 4. | −22 | 0 | 2 | −84 | 8 | 35 | 1 | 43 | 56 | 78 |
| 5. | −22 | 0 | −84 | 2 | 8 | 1 | 35 | 43 | 56 | 78 |
| 6. | −22 | −84 | 0 | 2 | 1 | 8 | 35 | 43 | 56 | 78 |
| 7. | −84 | −22 | 0 | 1 | 2 | 8 | 35 | 43 | 56 | 78 |

**Comportamiento Selection Sort (ejecución):**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1. | 2 | 35 | −22 | 0 | 56 | 78 | −84 | 8 | 43 | 1 |
| 2. | −84 | 35 | −22 | 0 | 56 | 78 | 2 | 8 | 43 | 1 |
| 3. | −84 | −22 | 35 | 0 | 56 | 78 | 2 | 8 | 43 | 1 |
| 4. | −84 | −22 | 0 | 35 | 56 | 78 | 2 | 8 | 43 | 1 |
| 5. | −84 | −22 | 0 | 1 | 56 | 78 | 2 | 8 | 43 | 35 |
| 6. | −84 | −22 | 0 | 1 | 2 | 78 | 56 | 8 | 43 | 35 |
| 7. | −84 | −22 | 0 | 1 | 2 | 8 | 56 | 78 | 43 | 35 |
| 8. | −84 | −22 | 0 | 1 | 2 | 8 | 35 | 78 | 43 | 56 |
| 9. | −84 | −22 | 0 | 1 | 2 | 8 | 35 | 43 | 78 | 56 |
| 10. | −84 | −22 | 0 | 1 | 2 | 8 | 35 | 43 | 56 | 78 |

3. **¿Qué es la recursión y en qué se aplica? ¿Qué componentes debe tener un método recursivo?**

   - La recursión es la forma de expresar la solución a un problema en términos de sí mismo, es aplicada para trabajar en cosas que tienen muchas ramificaciones posibles y son demasiado complejas para un enfoque iterativo. Un buen ejemplo de esto sería buscar a través de un sistema de archivos.
   - Toda función recursiva tiene dos componentes: un caso base y un paso recursivo.

4. **Se tienen los siguientes dos programas:**

```
1
2  public double mist (int n) {
3      double s = 0.0;
4      for (double i = 1; i <= n ; i ++) {
5              s = s + 1/ i ;
6      }
7      return s ;
8  }
9
```

Listing 1: mist

```
1
2  public double mistR (int n) {
3      if (n <= 1) {
4          return 1;
5      } else {
6          return ((double) 1/ n)  + mistR (n - 1) ;
7      }
8  }
9
```

Listing 2: mistR

¿Qué hacen mist y mistR?

- **mist:** recibe un entero, el cual es usado para determinar el número de veces que se va a realizar la operación de a un número decimal con valor inicial de 0.0 asignarle el valor de ese mismo número más 1 entre el índice de repetición. Es decir este método realiza un número determinado de veces la operación de sumarse a sí mismo con una mitad cada vez más pequeña de 1. Devuelve lo mismo que "$mistR$".
- **mistR:** recibe un entero que sirve para determinar el número de veces que se va a realizar la operación. Se tiene un caso base para detener el programa en algún momento, en el caso de la operación, esta realiza lo mismo "$mist$" pero de forma recursiva, es decir, se divide entre 1 entre un número n y a esto se le suma el antecesor de "$mistR$" usando "$mistR$". Devuelve lo mismo que "$mist$".

¿Cuál de las dos versiones es mejor y por qué?
La mejor versión es **mist**, ya que la iteración en este caso es más rápida que la recursión y dado que estamos trabajando con un double, tenemos que el resultado es cada vez más dificil de controlar, incluso si le pasamos un valor $'n'$ muy grande a ambos métodos, por poner un ejemplo 100000, el método iterativo va a terminar muy rápido, mientras que el recursivo probablemente entre en "$STACKOVERFLOW$", lo que significa que la iteración en este caso no permite manejar más números que la recursión.

5. **¿Por que es tan importante la Complejidad en Ciencias de la Computación?**
   Es importante porque nos permite determinar la eficacia de un algoritmo. La cual se expresa en función del tamaño de la entrada. Para poder representar la complejidad se hace uso de la notación O grandota la cual permite evaluar el ritmo de crecimiento en el tiempo de ejecución como una función dependiente de la cantidad de datos.

6. **Se tiene el siguiente programa:**

```
1
2   public int mist (int n) {
3       int r ;
4       for (r = 0; n != 0; n /= 10) {
5           r += n % 10;
6       }
7       return r ;
8   }
9
```

Listing 3: mist

```
1
2   public int mistR (int n)  {
3       if (n == 0) {
4           return 0;
5       } else {
6           return (n % 10) + mistR (n / 10) ;
7       }
8   }
9
```

Listing 4: mistR

¿Cuál es la complejidad de mist y mistR? Justifica tu respuesta.

- **mist:** $O(logn)$ Operación: O(1) + O(log n) + O(1) = O(1 + log n + 1) = O(log n)
  Desde mi punto de vista es logaritmica porque sin importar el tamaño de la entrada no es necesario recorrer todos los elementos.
- **mistR:** $O(logn)$ Operación: (O(1) * O(log n)) = O(1 log n) = O(log n)
  También siento que tiene complejidad logarítmica, pues a pesar de ejecutar más pasos que el método iterativo, sigue siendo innecesario recorrer todos los elementos.

¿Cuál método utiliza la regla de suma y cuál la regla de la multiplicación?

- "$mist$" utiliza la regla de la suma y "$mistR$" la regla de la multiplicación.