COLOMBIA POTENCIA DE LA VIDA

TIC

ALGORITMOS Y COMPLEJIDAD

INTEGRADOR-Lección 1-Unidad 2

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira

Tiempo de ejecución: 4 horas

PLANTEAMIENTO DE LA SESIÓN	Materiales
En esta lección se profundizará en los conceptos de algoritmia y su complejidad asociada. Se expondrán los pseudocódigos y diagramas de flujo para tener una herramienta de representación de los algoritmos. La notación Big O servirá para definir de manera estandarizada la complejidad de un algoritmo y se van a analizar sus cotas superiores e inferiores.	Lápiz y papel y/o computador con conexión a internet

Desarrollo teórico de la sesión: 2 horas

Algoritmia

Un algoritmo es un conjunto de instrucciones definidas, no ambiguas, ordenadas y finitas de un sistema o proceso para realizar una actividad concreta, por ejemplo, calcular el resultado de una ecuación matemática, procesar y transformar datos, preparar una receta de comida, etc. Hay algoritmos que pueden contener bucles de acciones en algunos de sus pasos, pero estos bucles deben tener un final establecido.

Los algoritmos pueden ser descritos de varias maneras, las más comunes son el pseudocódigo y los diagramas de flujo.

Pseudocódigo:

El pseudocódigo es una forma de representar algoritmos, funciones y otros procesos, utilizando una combinación de lenguaje natural y elementos similares al lenguaje de programación. No está escrito en ningún lenguaje de programación específico, sino que utiliza un lenguaje natural y símbolos fácilmente comprensibles para las personas.

El pseudocódigo es una herramienta de programación en la que las instrucciones se escriben en palabras similares al español, que facilitan tanto la escritura como la lectura de programas, es legible por humanos, pero no por máquinas. Es un paso intermedio entre los diagramas de flujo, que se expresan mediante símbolos, y los lenguajes de programación, que están ligados a una sintaxis bien definida.

+ Imagen

Figura 1: Pseudocódigo de una suma.

En la **figura 1** se puede apreciar el pseudocódigo de la suma de dos números. Cuenta con una estructura de pasos ordenados, bien definidos, y cuenta con un final al dar el resultado de dicha suma.

Pseudocódigo:

```
INICIO
   Num1, Num2,Suma  : ENTERO
   ESCRIBA  "Diga dos números: "
   LEA Num1, Num2
   Suma ←   Num1 + Num2
   ESCRIBA "La Suma es:", Suma
FIN
```

Figura 1: Pseudocódigo de una suma.

Diagramas de flujo:

Un diagrama de flujo es un gráfico que muestra los pasos de un algoritmo de manera visual y fácil de entender. A diferencia del pseudocódigo, los diagramas de flujo cuentan con símbolos o figuras geométricas que indican utilidades, estas se utilizan para describir visualmente los diferentes pasos y decisiones del algoritmo.

+ Imagen

Figura 2: Símbolos de un diagrama de flujo.

En la figura 2 se muestran los símbolos principales para crear un diagrama de flujo y representar un algoritmo. Estas figuras son un estándar y se deben utilizar de manera adecuada para que otros puedan entender los procedimientos que se llevan a cabo, aunque a veces puede haber alguna variación.

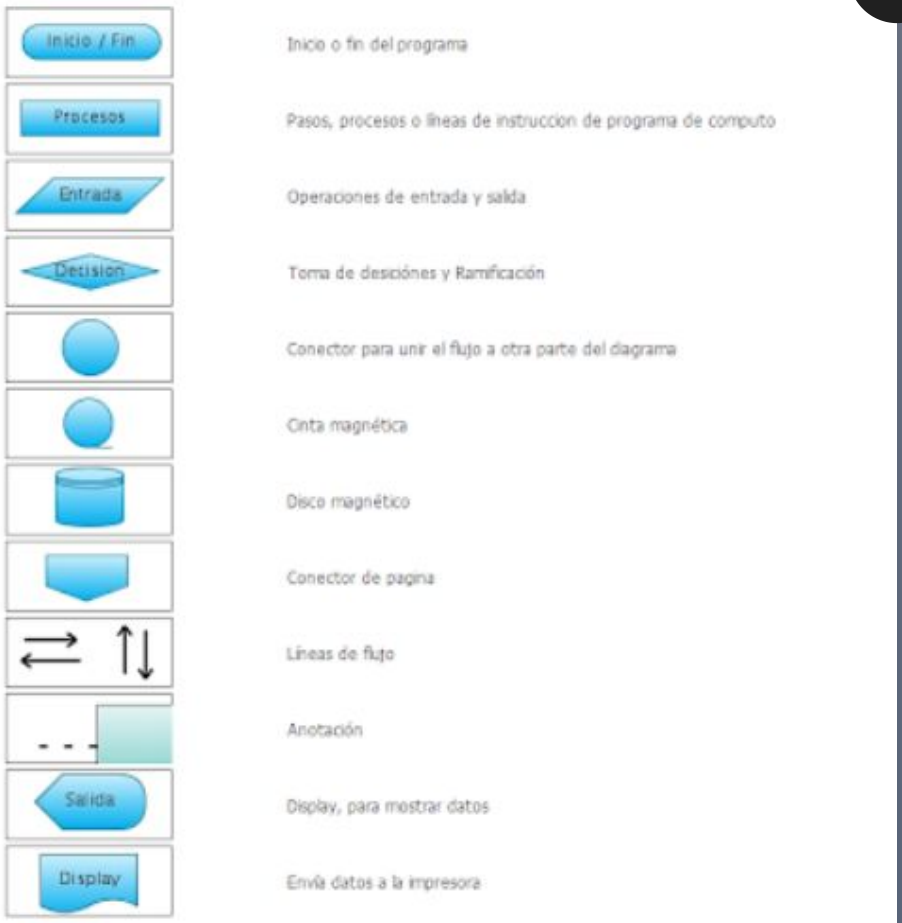


Figura 2: Símbolos de un diagrama de flujo.

Tomando el ejemplo del algoritmo de suma de dos números, ahora se va a realizar el mismo proceso con el diagrama de flujo.

Figura 2.1: Diagrama de flujo de una suma.

+ Imagen

En la **figura 2.1** se muestra el algoritmo para sumar dos números, haciendo uso de los símbolos para cada parte del proceso. Lo primero a destacar es que el algoritmo debe contar siempre con un inicio y un final, sin alguna de estas dos partes no se puede considerar que está bien hecho, el símbolo usado es un óvalo. Los procesos se definen usando un rectángulo, para la suma de números el programa define el tipo de datos de entrada, en este caso dos números enteros, y posteriormente hace la suma. El símbolo usado para el mensaje "DIGA DOS NUMEROS", como se muestra en la figura 2, es un símbolo que indica que el programa le muestra un mensaje al usuario con instrucciones o información. Para que el algoritmo indique que recibe una entrada se usa el paralelogramo, en este caso recibe el Num1 y Num2.

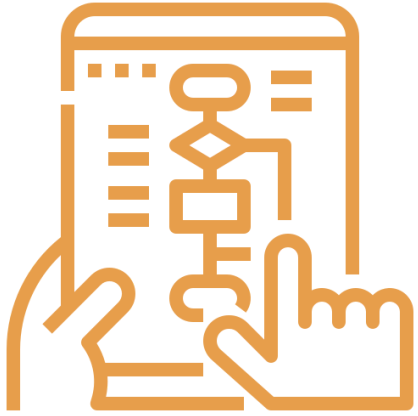
Pseudocódigo:

```
INICIO
   Num1, Num2,Suma  : ENTERO
   ESCRIBA  "Diga dos números: "
   LEA Num1, Num2
   Suma ←   Num1 + Num2
   ESCRIBA "La Suma es:", Suma
FIN
```

Figura 1: Pseudocódigo de una suma.

Complejidad algorítmica

Se centra en medir y analizar la eficiencia de los algoritmos en función del tamaño de una entrada. Se mide en unidades de tiempo y los recursos necesarios para la ejecución, la complejidad crece en la medida en que se necesite más tiempo y recursos del sistema para ejecutar un algoritmo. Para medir esta complejidad se usa la notación Big O, la cual la representa de manera general y asintótica.



Notación Big O.

La notación Big-O es una herramienta que ayuda a entender cómo se comportan los algoritmos en relación con el tamaño de los datos que manejan. En términos generales, proporciona una indicación de la eficiencia y escalabilidad de un algoritmo.

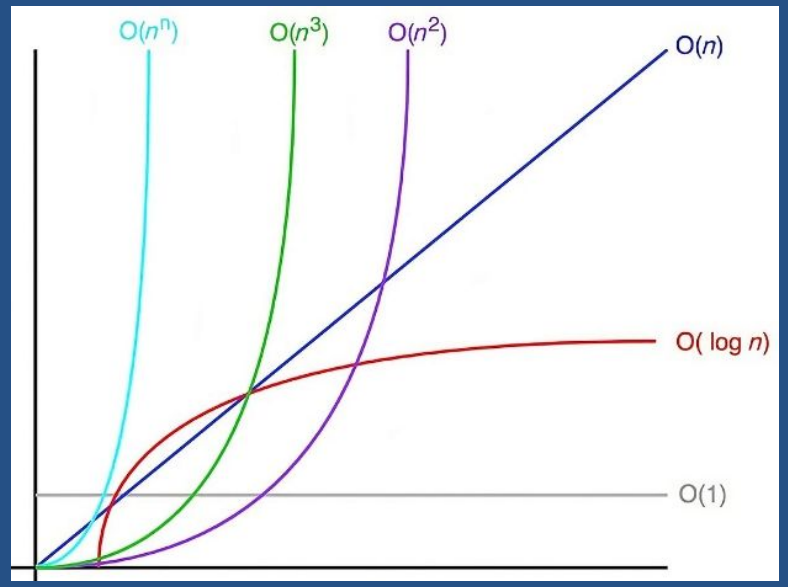


Figura 3: Complejidades típicas de un algoritmo.

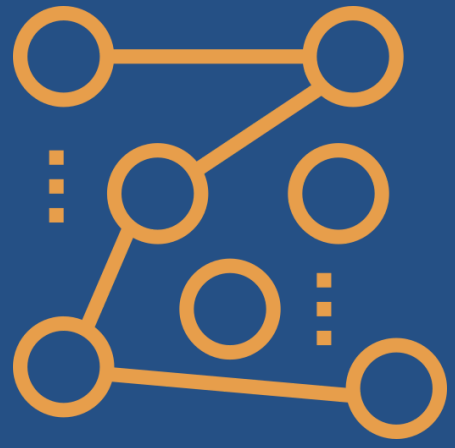
En la **figura 3** se pueden ver algunas de las complejidades típicas de los algoritmos con la notación Big O.

Entre más un algoritmo se acerque a la notación O(nn), menor va a ser su eficiencia con grandes entradas de datos, esto quiere decir que va a tomar más tiempo y capacidad en la memoria procesar las instrucciones de dicho algoritmo. Las funciones que aparecen en la figura 3 son las siguientes:

- O(1): constante, la operación no depende del tamaño de los datos.
- O(log n): logarítmica, asociada con algoritmos que dividen el problema en partes.
- O(n): lineal, el tiempo de ejecución es proporcional al tamaño de los datos.
- O(n2): cuadrática, común en algoritmos que requieren iteración por todos los elementos en cada uno de los elementos a procesar.
- O(n2): cuadrática, común en algoritmos que requieren iteración por todos los elementos en cada uno de los elementos a procesar.
- O(n3): cúbica, es típica en algoritmos que tienen 3 bucles anidados.
- O(nn): exponencial, se puede ver en algoritmos de fuerza bruta que buscan todos los subconjuntos posibles de un conjunto.

Cota superior (notación O):

La notación O, o cota superior, describe el límite superior asintótico del crecimiento de una función a medida que el tamaño de la entrada n se aproxima a infinito. Se usa para representar la peor situación posible en términos de eficiencia. Por ejemplo, si un algoritmo tiene una complejidad $O(n^2)$, se sabe que en el peor de los casos el tiempo de ejecución crece cuadráticamente en relación con el tamaño de la entrada. Su definición formal es: si una función fn, es O(gn), significa que existe una constante positiva c y un valor n0 tal que fn nunca crecerá más rápido que c*g(n) para todo n mayor o igual a n_0.



Cota Inferior (Ω-notación):

La cota inferior, representada por la notación (g(n)), describe el crecimiento mínimo asintótico de un algoritmo. Proporciona un límite inferior en la tasa de crecimiento a medida que n se aproxima a infinito. Se utiliza para describir la mejor situación posible del rendimiento de un algoritmo. Por ejemplo, si un algoritmo tiene una complejidad de (n), se sabe que, en el mejor caso, el tiempo de ejecución crecerá al menos linealmente con respecto al tamaño de la entrada. Su definición formal es: si una función f(n) es (g(n)), esto significa que existe una constante positiva c y un valor n0 tal que fn nunca crecerá más lentamente que c*g(n) para todo n mayor o igual a n_0.

En conjunto, las cotas superior e inferior proporcionan un marco para comprender cómo se comporta un algoritmo a medida que cambia el tamaño de la entrada. La cota superior indica el crecimiento máximo, mientras que la cota inferior indica el crecimiento mínimo, proporcionando así un rango en el cual se puede clasificar el comportamiento asintótico de un algoritmo.

COLOMBIA POTENCIA DE LA VIDA

TIC

TALENTO TECH

AZ | PROYECTOS EDUCATIVOS

UTP Universidad Tecnológica de Pereira