

Lógica de programación I

Juan Pablo Restrepo Uribe

Ing. Biomedico

MSc. Automatización y Control Industrial

jprestrepo@correo.iue.edu.co

2024

Institución Universitaria de Envigado

Presentación del curso

“...promueve en el estudiante el desarrollo de los skills necesarios para analizar e interpretar problemas y proponer soluciones mediante el diseño de algoritmos.”

“...gestionar los sistemas de Información en el ámbito del desarrollo de software con incorporación de inteligencia artificial y la analítica de datos, considerando mejores prácticas y metodologías para el desarrollo de proyectos de ingeniería informática...”

Propósito de formación de la asignatura

Desarrollar algoritmos para solucionar problemas reales, utilizando un lenguaje de programación y realizando validaciones mediante la prueba de escritorio como técnica para identificar los posibles errores.

Unidades de aprendizaje

Unidad	Tema	Semana
Unidad 1: Definiciones Básicas	Definición de algoritmo, características de los algoritmos, pasos para la solución de problemas empleando algoritmos	1
	Tipos de instrucción, tipos de datos, variables y constantes, operadores aritméticos.	2
	Formas de representar algoritmos: pseudocódigo, diagrama flujo y diagrama de bloques	3
Unidad 2: Estructuras selectivas, condicionales o de decisión. repetitivas, cíclicas o iterativas	Estructuras selectivas simples y dobles	4
	Estructuras selectivas múltiples y anidadas	5
	Definición de variables contador y acumulador. Ciclo para, desde o for	6
	Ciclo mientras o while. Variable centinela y bandera	7
	Ciclo haga mientras o do..while y ciclos anidados	8
Parcial	Examen parcial	9
Unidad 3: Introducción al lenguaje de programación Subprogramas - Arreglos	Definición de programa informático y lenguajes de programación (niveles de abstracción: lenguaje de máquina, lenguaje de bajo nivel, lenguaje de alto nivel).	10
	Estructuras selectivas y repetitivas en el lenguaje de programación	11
	Subprogramas: Diseño e invocación de Funciones y Procedimientos.	12
	Comunicación con subprogramas: paso de parámetros, Funciones y procedimientos como parámetros	13
	Arreglos unidimensionales: Vectores, Operaciones con vectores: Asignación, lectoescritura, acceso y actualización.	14
	Arreglos unidimensionales: Algoritmos de búsqueda. Algoritmo de ordenamiento.	15
	Arreglos bidimensionales: Matrices (declaración, creación, inicio)	16
Final	Operaciones con matrices. Ejercicios Prácticos con vectores y Matrices.	16
	Evaluación final	17

Evaluación

Evento evaluativo	Porcentaje	Fecha
Parcial I	20 %	11 de septiembre
Parcial II	20 %	13 de noviembre
Proyecto de aula 1	15 %	5 de agosto
Proyecto de aula 2	15 %	4 de septiembre
Proyecto de aula 3	15 %	9 de octubre
Proyecto de aula 4	15 %	30 de octubre

Fechas importantes

- Iniciación de Periodo Académico: El 22 de Julio de 2024
- Terminación de Periodo Académico: El 17 de Noviembre de 2024
- Iniciación de Clases: El 22 de Julio de 2024
- Terminación de Clases: El 9 de Noviembre de 2024
- Actividad Evaluativa Parcial 20%: Desde el 9 hasta el 14 de Septiembre de 2024
- Actividad de Evaluación Final 20%: Desde el 10 hasta el 17 de Noviembre de 2024
- Registro en el sistema del 60%: Hasta el 9 de Noviembre de 2024

Evaluación

Trabajo con acompañamiento directo del docente

- Clase magistral
- Socialización por parte de los estudiantes de consultas, proyectos y trabajos realizados.
- Desarrollo de ejercicios prácticos.
- Solución de inquietudes y retroalimentación en la elaboración de talleres individuales y en grupos
- Trabajo guiado de laboratorio utilizando entornos de desarrollo integrado (IDE)
- Trabajo guiado utilizando diferentes herramientas

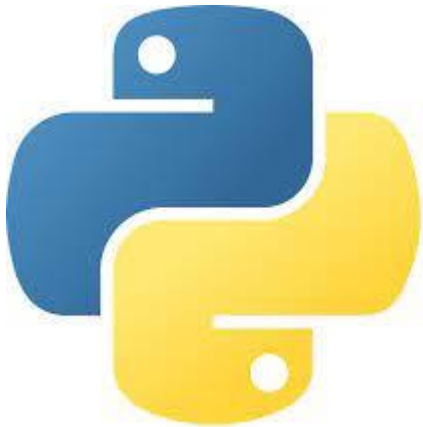
Trabajo independiente

- Indagación de los recursos bibliográficos físicos y electrónicos
- Talleres prácticos individuales o grupales.
- Prácticas de laboratorio para el desarrollo de aplicaciones
- Desarrollo de las actividades en las plataformas virtuales institucionales
- Desarrollo de actividades en las herramientas

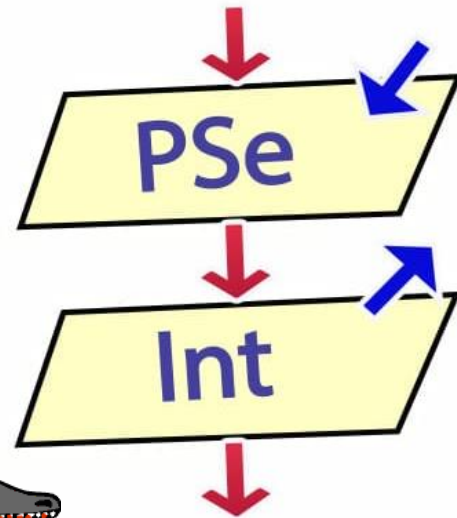
Cursos de programación

- <https://www.youtube.com/watch?v=G2FCfQj-9ig&list=PLU8oAlHdN5BlvPxziopYZRd55pdqFwkeS>
- <https://www.udemy.com/course/curso-python/?start=0>
- https://www.youtube.com/watch?v=4PZmLUh2Z-c&list=PLoGFizEtm_6jCjWqRU8A-dQYQuo5q5KNc
- https://www.youtube.com/watch?v=5m4WORAIfr4&list=PLgHCrivozIb0TRHpUfuAUZ-VKck8KYV_2
- <https://www.youtube.com/watch?v=JJ7BMoQotEY&list=PLgHCrivozIb0ULMKfJVV-rFdRG2OeEgfg>
- <https://www.youtube.com/watch?v=LplofeTqgpc&list=PLgHCrivozIb3GgqeEkDEA3j0dLGI6T6vm>

Herramientas del curso



Google Colaboratory

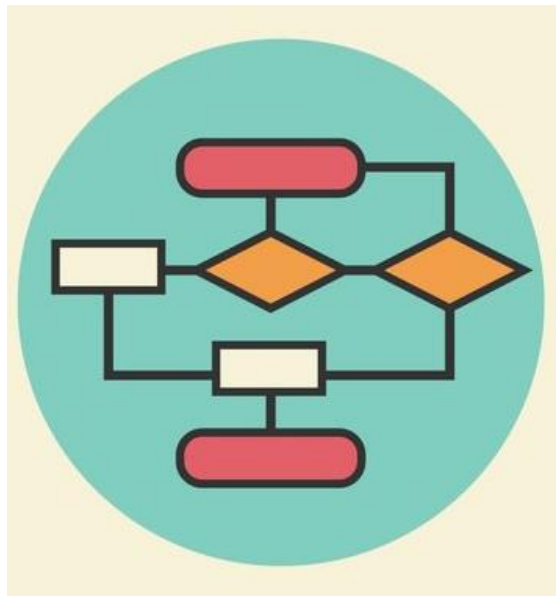


Unidad 1: Definiciones Básicas

- Definición de algoritmo
- Características de los algoritmos
- Pasos para la solución de problemas empleando algoritmos:
 - Análisis
 - Diseño
 - Codificación
 - Compilación
 - Ejecución
 - Pruebas (prueba de escritorio)
 - Documentación
 - Mantenimiento.
- Algoritmos con estructura secuencial.

Definición de algoritmo

Un algoritmo es un conjunto de instrucciones **definidas**, **ordenadas** y **acotadas** para resolver un problema, realizar un cálculo o desarrollar una tarea.

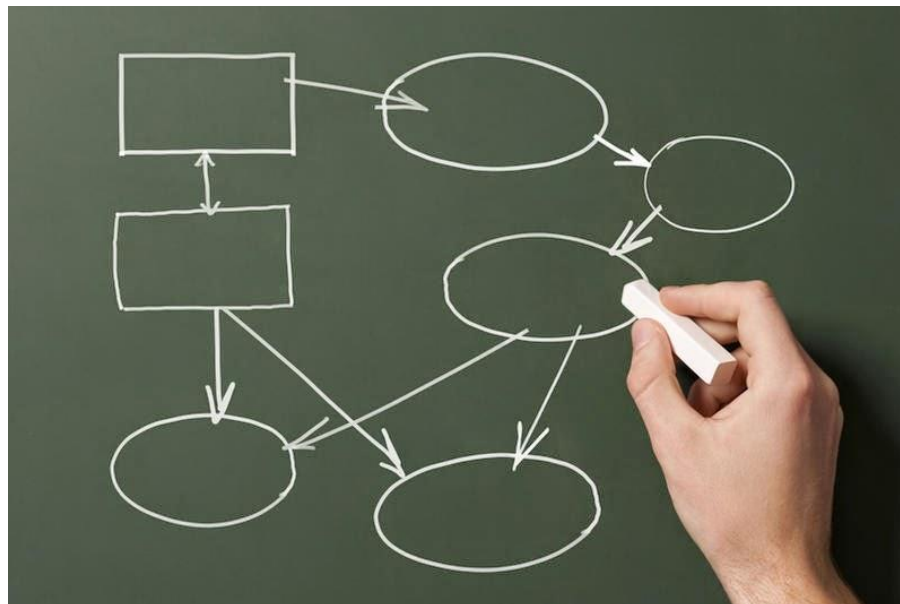


Definición de algoritmo



Definición de algoritmo

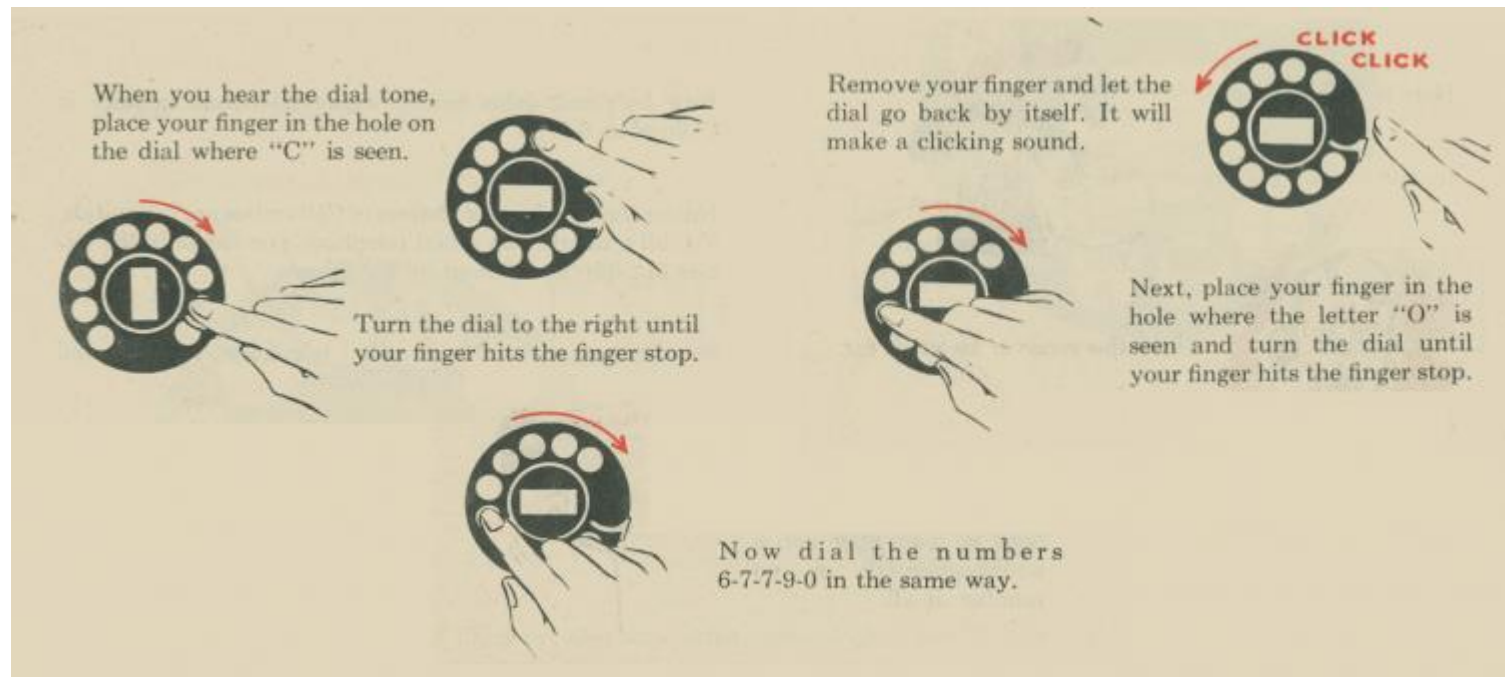
Un algoritmo es un procedimiento paso a paso para conseguir un fin. A partir de un estado e información iniciales, se siguen una serie de pasos ordenados para llegar a la solución de una situación.



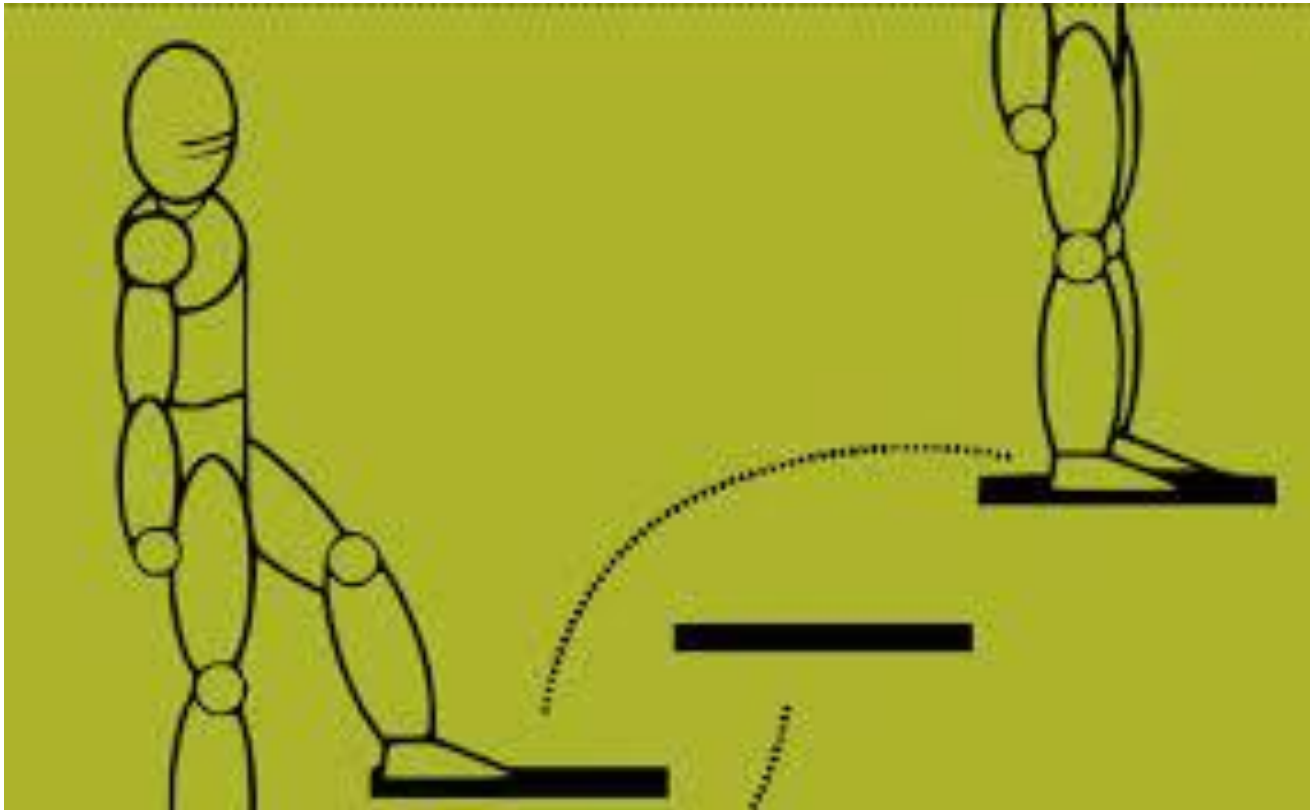
Ejemplos de algoritmos



Ejemplos de algoritmos



Ejemplos de algoritmos



Ejemplos de algoritmos

Nadie habrá dejado de observar que con frecuencia el suelo se pliega de manera tal que una parte sube en ángulo recto con el plano del suelo, y luego la parte siguiente se coloca paralela a este plano, para dar paso a una nueva perpendicular, conducta que se repite en espiral o en línea quebrada hasta alturas sumamente variables. Agachándose y poniendo la mano izquierda en una de las partes verticales, y la derecha en la horizontal correspondiente, se está en posesión momentánea de un peldaño o escalón. Cada uno de estos peldaños, formados como se ve por dos elementos, se sitúa un tanto más arriba y adelante que el anterior, principio que da sentido a la escalera, ya que cualquiera otra combinación producirá formas quizá más bellas o pintorescas, pero incapaces de trasladar de una planta baja a un primer piso...

Instrucciones para subir una escalera
Julio Cortázar

Algoritmos cotidianos

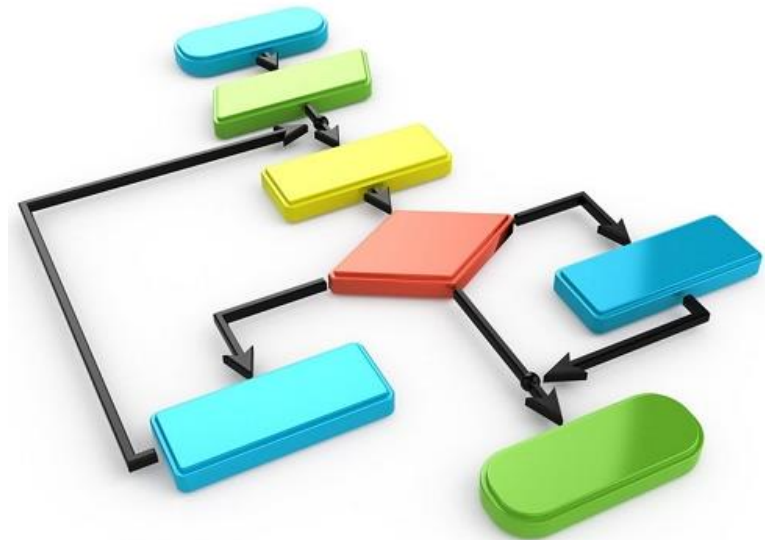
- Hacer el desayuno
- Venir a la universidad
- Cepillarse los dientes
- Cargar Un Celular
- Enviar un correo electrónico
- Lavarse las manos



Partes de un algoritmo

En programación, un algoritmo supone el paso previo a ponerse a escribir el código. Primero debemos encontrar la **forma de obtener la solución al problema**, para luego, a través del código, poder indicarle a la máquina qué acciones queremos que lleve a cabo.

Un programa informático no sería más que un **conjunto de algoritmos ordenados y codificados en un lenguaje de programación** para poder ser ejecutados en un ordenador.



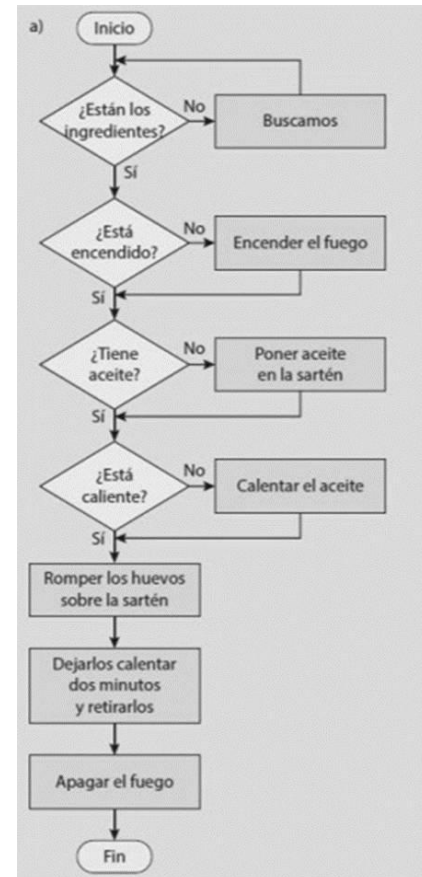
Partes de un algoritmo

- **Input:** Información que damos al algoritmo con la que va a trabajar para ofrecer la solución esperada.
- **Proceso:** Conjunto de pasos para que, a partir de los datos de entrada, llegue a la solución de la situación.
- **Output:** Resultados, a partir de la transformación de los valores de entrada durante el proceso.



Partes de un algoritmo

Un algoritmo informático parte de un estado inicial y de unos valores de entrada, sigue una serie de pasos sucesivos y llega a un estado final en el que ha obtenido una solución.



Tipos de algoritmos

Existen diversas clasificaciones de algoritmos, en función de diferentes criterios.

- Según su sistema de signos (cómo describen los pasos a seguir)
 - Algoritmos cuantitativos
 - Algoritmos cualitativos
- Si lo hacen a través de cálculos matemáticos o secuencias lógicas
- Si requieren o no el empleo de un ordenador para su resolución
 - Computacionales
 - No computacionales

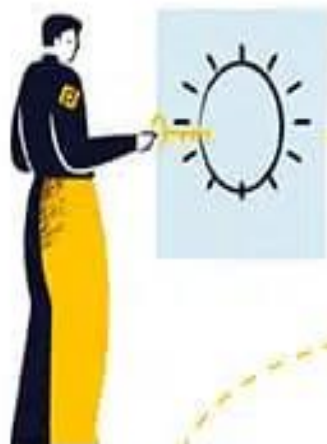
Tipos de algoritmos

Sin embargo, nos centraremos en:

- Su función (qué hace)
- Su estrategia para llegar a la solución (cómo lo hace)



5 tipos de algoritmos



1. Algoritmos de búsqueda

Los algoritmos de búsqueda localizan uno o varios elementos que presenten una serie de propiedades dentro de una estructura de datos.

2. Algoritmos de ordenamiento

Reorganizan los elementos de un listado según una relación de orden. Destacan el ordenamiento por inserción, por mezcla, por selección, de burbuja y el ordenamiento rápido.



3. Programación dinámica

Método que reduce el tiempo de ejecución de un algoritmo, al **dividir problemas en subproblemas** y almacenar su solución, para que no haya que volver a calcularlos.

4. Algoritmos voraces

Los algoritmos voraces adoptan la decisión más óptima en cada paso local con el objetivo de llegar a la mejor solución global.



5. Algoritmos probabilísticos

Utilizan un cierto grado de **azar** para proporcionar un resultado. De media proporcionan una buena solución al problema.

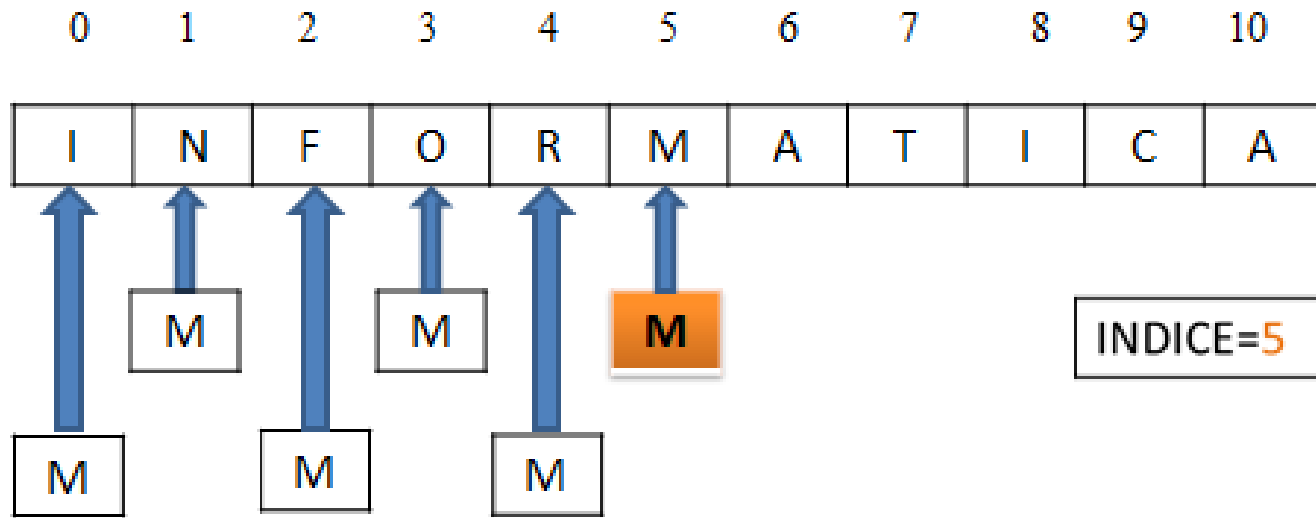


Sigue leyendo en profile.es/blog/



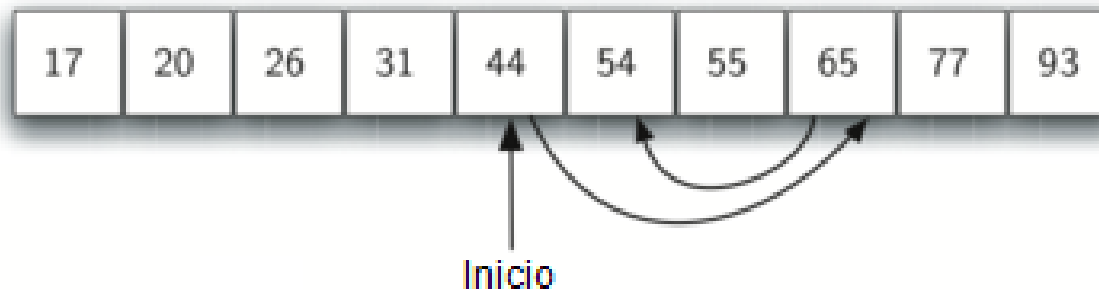
Algoritmos de búsqueda

Los algoritmos de búsqueda localizan uno o varios elementos que presenten una serie de propiedades dentro de una estructura de datos.

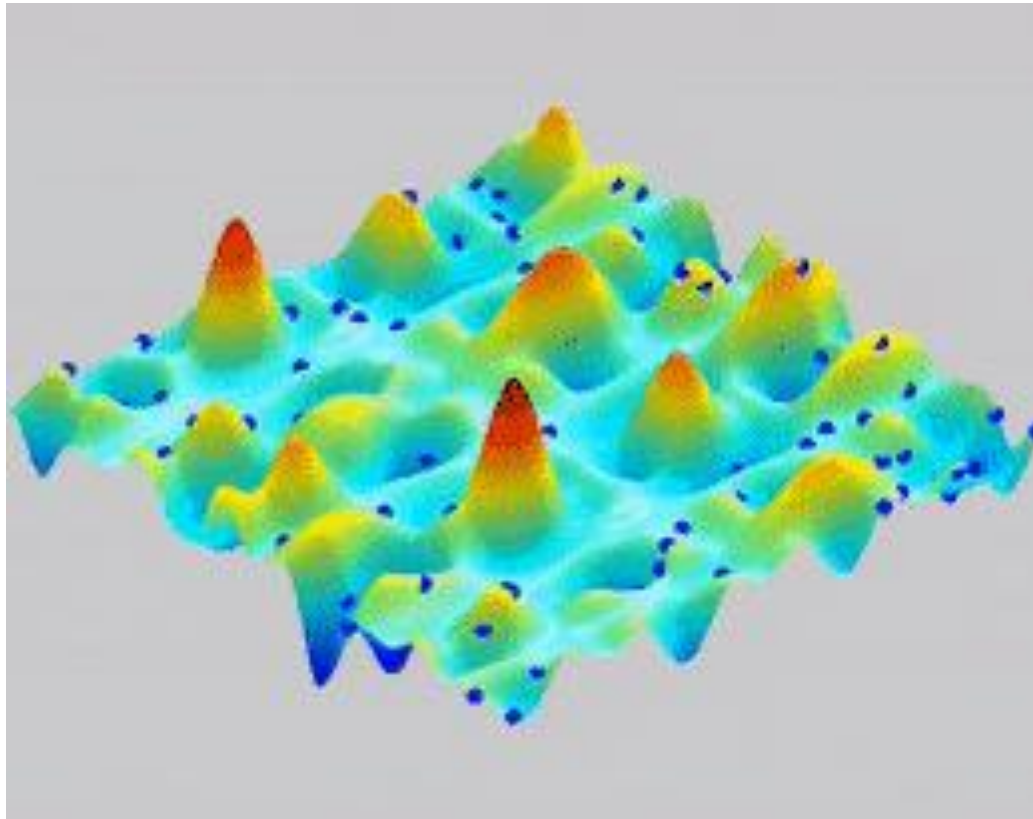


Algoritmos de búsqueda

- **Búsqueda secuencial:** En la que se compara el elemento a localizar con cada elemento del conjunto hasta encontrarlo o hasta que hayamos comparado todos.
- **Búsqueda binaria:** En un conjunto de elementos ordenados, hace una comparación con el elemento ubicado en el medio y, si no son iguales, continúa la búsqueda en la mitad donde puede estar. Y así sucesivamente en intervalos cada vez más pequeños de elementos.

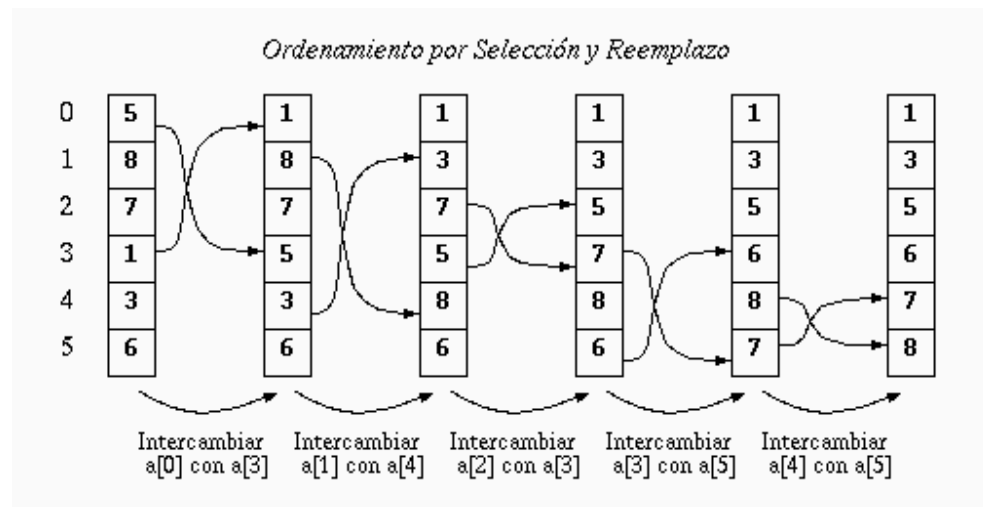


Algoritmos de búsqueda



Algoritmos de ordenamiento

Reorganizan los elementos de un listado según una relación de orden. Las más habituales son el orden numérico y el orden lexicográfico. Un orden eficiente optimiza el uso de algoritmos como los de búsqueda y facilitan la consecución de resultados legibles por personas y no solo máquinas.



Algoritmos de ordenamiento

- **Ordenamiento de burbuja:** Compara cada elemento de la lista a ordenar con el siguiente e intercambia su posición si no están en el orden adecuado. Se revisa varias veces toda la lista hasta que no se necesiten más intercambios.
- **Ordenamiento por selección:** Vamos colocando el elemento más pequeño disponible en cada una de las posiciones de la lista de forma consecutiva.
- **Ordenamiento rápido:** Elegimos un elemento del conjunto (pivote) y reubicamos el resto a cada uno de sus lados, en función de si son mayores o menores que el elemento que estamos tomando como referencia. Repetimos el procedimiento en cada subconjunto.

Algoritmos de ordenamiento

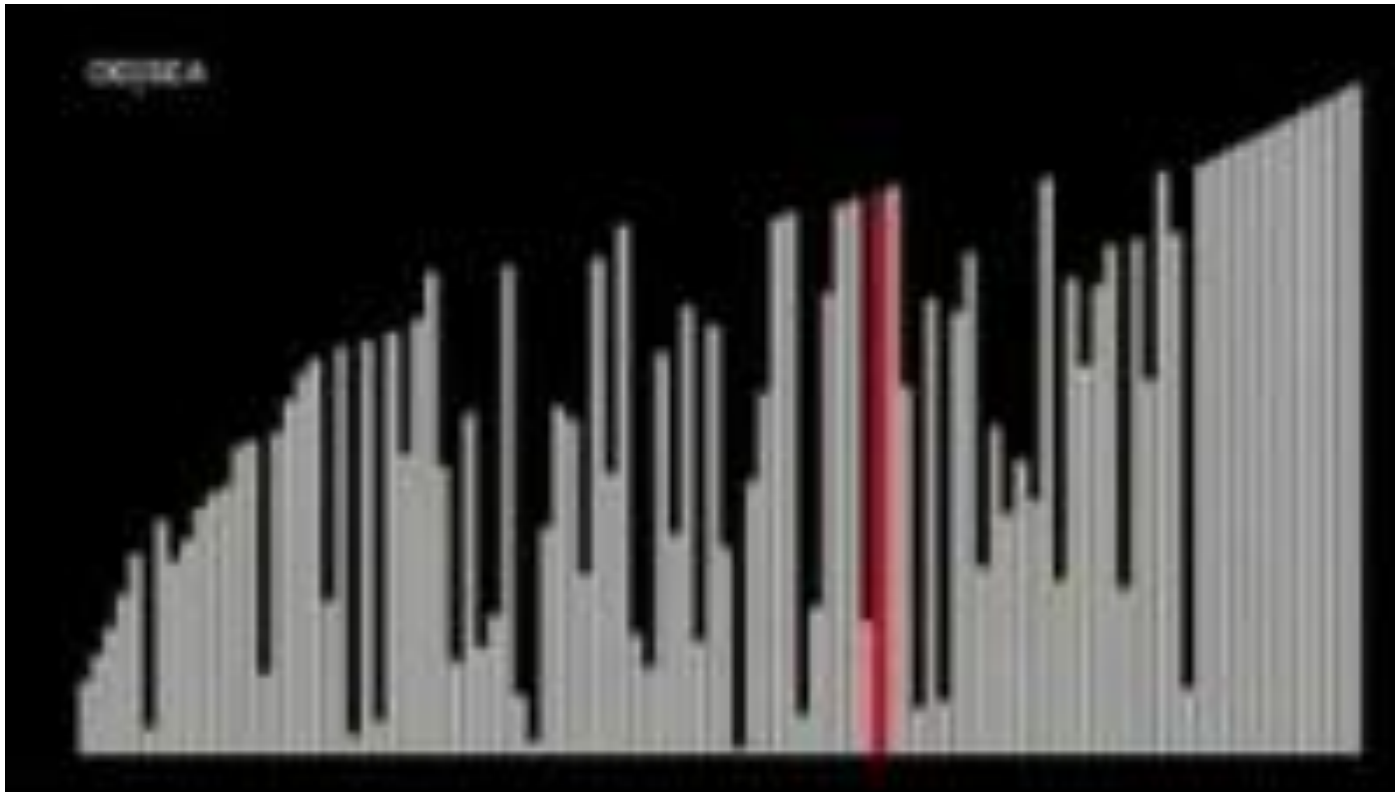
```
procedimiento DeLaBurbuja ( $a_0, a_1, a_2, \dots, a_{n-1}$ )  
  para  $i \leftarrow 0$  hasta  $n - 1$  hacer  
    para  $j \leftarrow 0$  hasta  $n - i - 1$  hacer  
      si  $a_{(j)} > a_{(j+1)}$  entonces  
         $aux \leftarrow a_{(j)}$   
         $a_{(j)} \leftarrow a_{(j+1)}$   
         $a_{(j+1)} \leftarrow aux$   
      fin si  
    fin para  
  fin para  
fin procedimiento
```

```
procedimiento DeLaBurbuja2 ( $a_{(0)}, a_{(1)}, a_{(2)}, \dots, a_{(n-1)}$ )  
   $i \leftarrow 1$   
  ordenado  $\leftarrow$  no  
  mientras  $(i < n) \wedge (\text{ordenado} = \text{no})$  hacer  
     $i \leftarrow i + 1$   
    ordenado  $\leftarrow$  si  
      para  $j \leftarrow 0$  hasta  $n - i$  hacer  
        si  $a_{(j)} > a_{(j+1)}$  entonces  
          ordenado  $\leftarrow$  no  
           $aux \leftarrow a_{(j)}$   
           $a_{(j)} \leftarrow a_{(j+1)}$   
           $a_{(j+1)} \leftarrow aux$   
        fin si  
      fin para  
    fin mientras  
  fin procedimiento
```

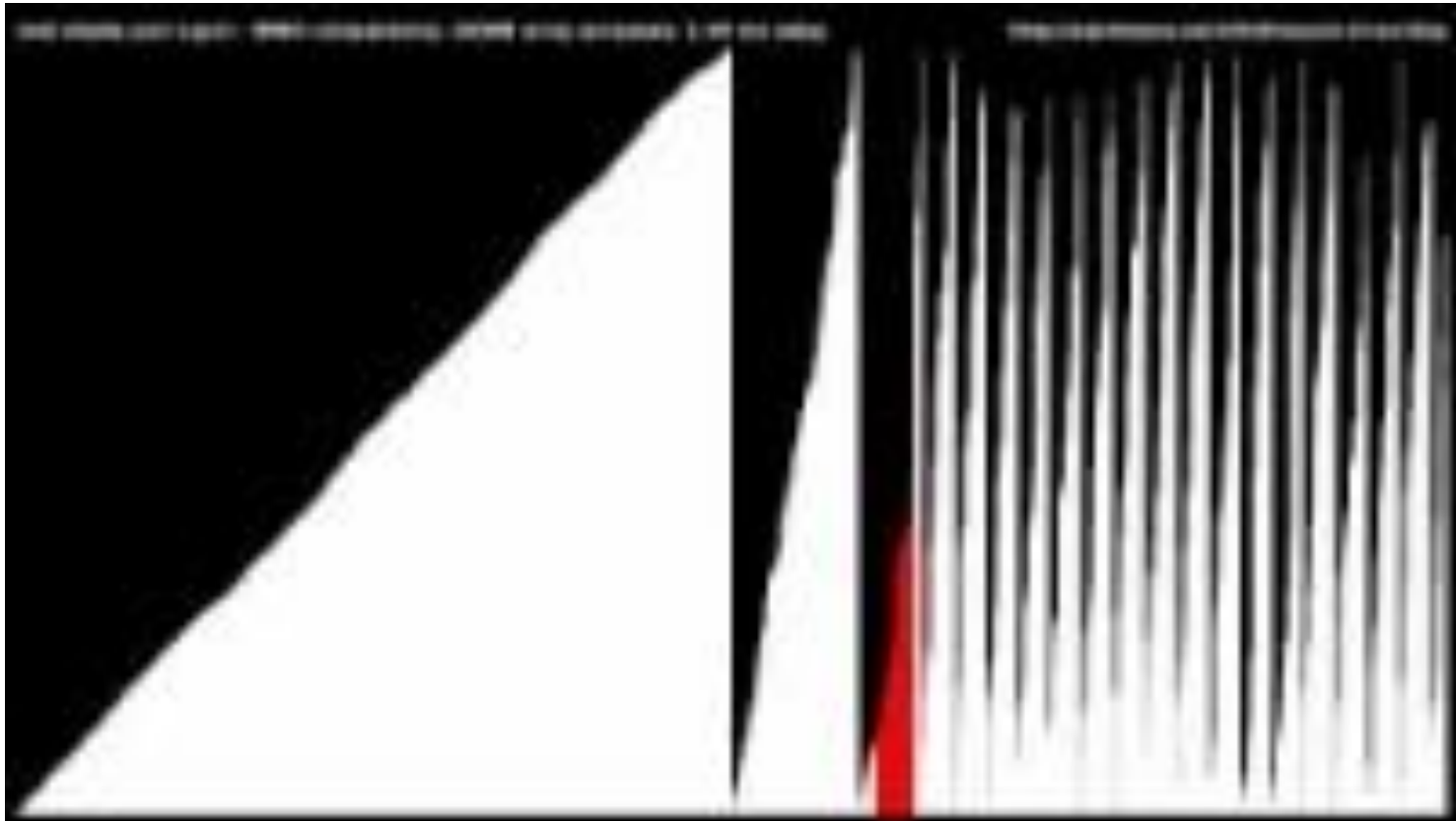
Algoritmos de ordenamiento

```
procedimiento DeLaBurbuja3 ( $a_{(0)}, a_{(1)}, a_{(2)}, \dots, a_{(n-1)}$ )  
   $i \leftarrow 1$   
  repetir  
     $i \leftarrow i + 1$   
    ordenado  $\leftarrow$  si  
    para  $j \leftarrow 0$  hasta  $n - i - 1$  hacer  
      si  $a_{(j)} > a_{(j+1)}$  entonces  
        ordenado  $\leftarrow$  no  
        aux  $\leftarrow a_{(j)}$   
         $a_{(j)} \leftarrow a_{(j+1)}$   
         $a_{(j+1)} \leftarrow aux$   
      fin si  
    fin para  
  hasta que  $\neg(i < n) \vee (\text{ordenado} = \text{si})$   
fin procedimiento
```

Algoritmos de ordenamiento

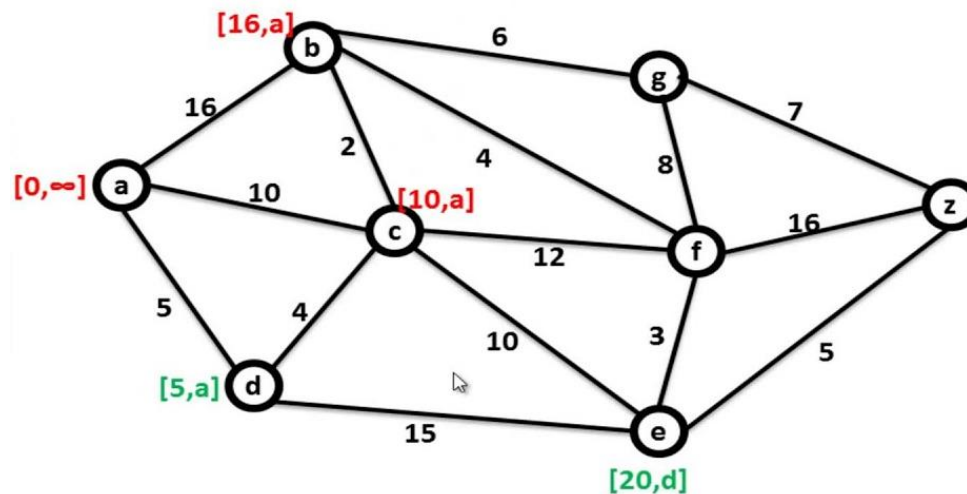


Algoritmos de ordenamiento



Algoritmos voraces

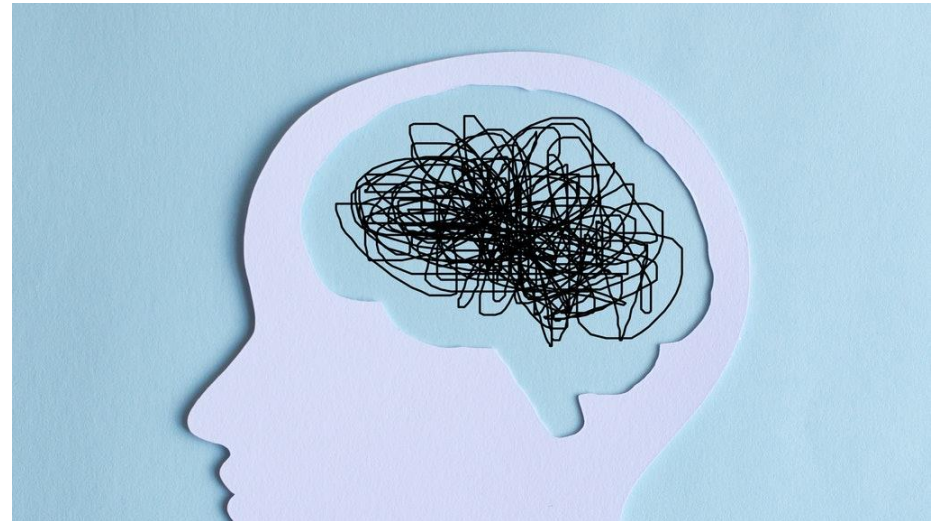
Consisten en una estrategia de búsqueda que sigue una **heurística** en la que se elige la mejor opción óptima en cada paso local con el objetivo de llegar a una solución general óptima. Es decir, en cada paso del proceso escogen el mejor elemento (elemento prometededor) y comprueban que pueda formar parte de una solución global factible. Normalmente se utilizan para resolver problemas de optimización.



Algoritmos voraces

Se conoce como **heurística** al conjunto de **técnicas o métodos para resolver un problema**. La palabra heurística es de origen griego εὐρίσκειν que significa “**hallar, inventar**”.

La heurística es vista como el arte de inventar por parte de los seres humanos, con la intención de procurar estrategias, métodos, criterios, que permitan resolver problemas a través de la creatividad, pensamiento divergente o lateral.



Algoritmos voraces

heurístico, ca 

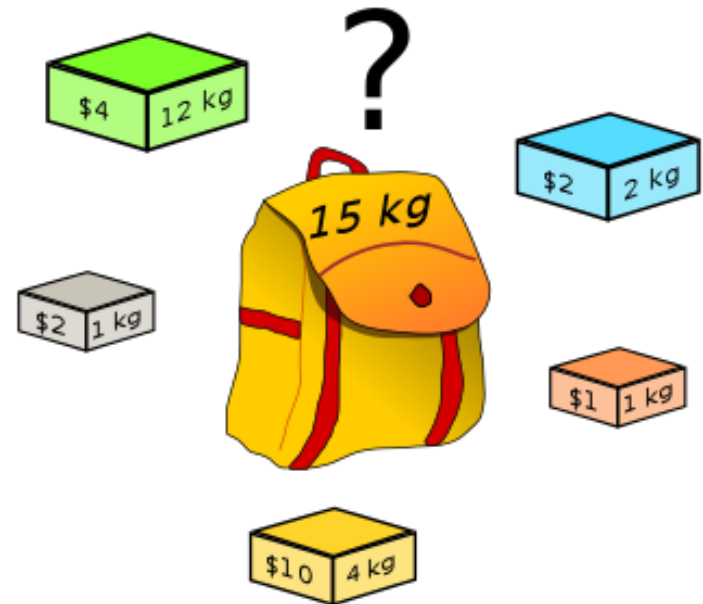
Del gr. εὕρισκειν *heurískein* 'hallar', 'inventar' y *-tico*.

1. **adj.** Perteneciente o relativo a la **heurística**.
2. **f.** Técnica de la indagación y del descubrimiento.
3. **f.** Búsqueda o investigación de documentos o fuentes históricas.
4. **f.** En algunas ciencias, manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas, **etc.**

Algoritmos voraces

- **Problema de la mochila fraccional (KP):**

Disponemos de una colección de objetos (cada uno de ellos con un valor y un peso asociados) y debemos determinar cuáles colocar en la mochila para lograr transportar el valor máximo sin superar el peso que puede soportar.



Ejemplo del problema de la mochila: dada una mochila con una capacidad de 15 kg que puedo llenar con cajas de distinto peso y valor, ¿qué cajas elijo de modo de maximizar mis ganancias y no exceder los 15 kg de peso permitidos?

Algoritmos voraces

Maximizar $\sum_{i=1}^n b_i x_i$

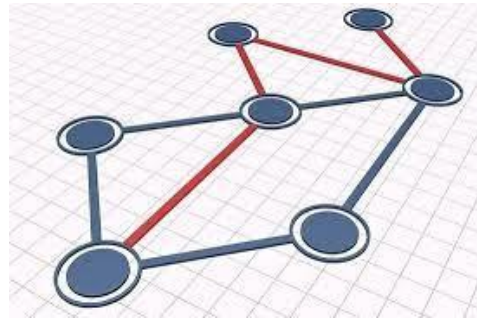
sujeto a

$$\sum_{i=1}^n c_i x_i \leq P$$

$$x_i \in \{0, 1\} \text{ con } i = 1, \dots, n$$

Algoritmos voraces

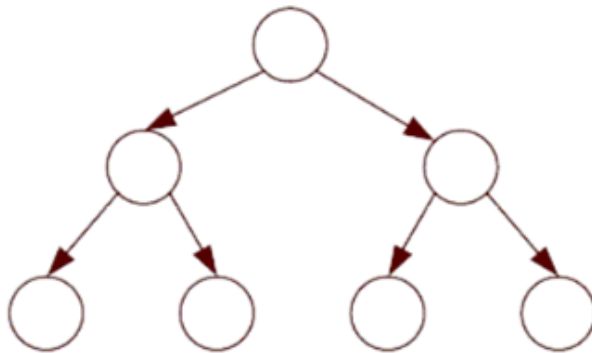
- **Algoritmo de Dijkstra:** Utilizado para determinar el camino más corto desde un vértice origen hasta los demás vértices de un grafo, que tiene pesos en cada arista.



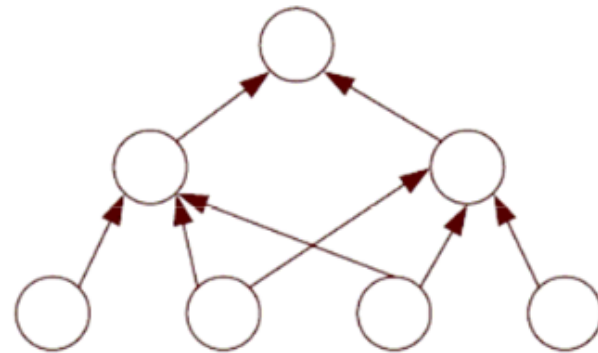
- **Codificación Huffman:** Método de compresión de datos sin perder información, que analiza la frecuencia de aparición de caracteres de un mensaje y les asigna un código de longitud variable. Cuanto mayor sea la frecuencia le corresponderá un código más corto.

Programación dinámica

La programación dinámica es un método de resolución de problemas en el que dividimos un problema complejo en subproblemas y calculamos y almacenamos sus soluciones, para que no haga falta volver a calcularlas más adelante para llegar a la solución del problema. La programación dinámica reduce el tiempo de ejecución de un algoritmo al optimizar la recursión.



diviser et régner



programmation dynamique

Programación dinámica

1. Dividir el problema en subproblemas más pequeños.
2. Resolver estos problemas de manera óptima usando este proceso de tres pasos recursivamente.
3. Usar estas soluciones óptimas para construir una solución óptima al problema original.

```
FUNC fib( $\downarrow$ n: NATURAL): NATURAL
INICIO
    SI n = 0 ENTONCES
        DEVOLVER 0
    SI NO, SI n = 1 ENTONCES
        DEVOLVER 1
    SI NO
        devolver fib(n-1) + fib(n-2)
    FIN SI
FIN
```

Programación dinámica

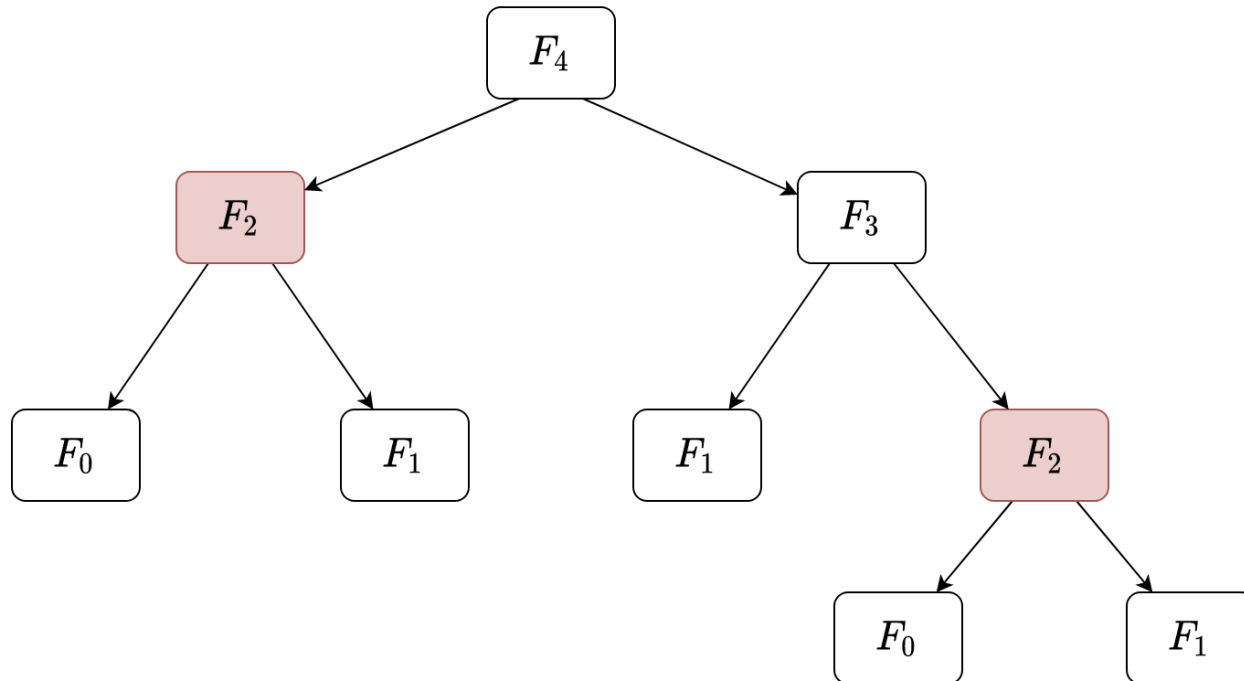
La serie de Fibonacci: Sucesión de números que comienza con “0” y “1” y, a partir de ellos, cada número es resultado de la suma de los dos que le preceden. La relación de recurrencia la define.

$$F_0 = 1$$

$$F_1 = 1$$

$$F_n = F_{n-1} + F_{n-2} \quad \forall n \geq 2$$

Programación dinámica



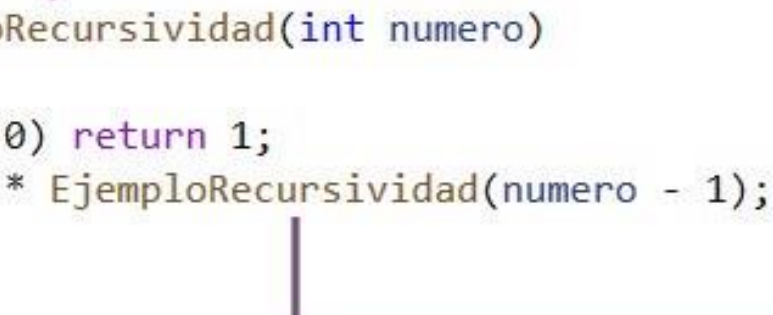
Enfoque recursivo: Mayor computo

Programación dinámica

La **recursividad** consiste en funciones que se llaman a sí mismas, evitando el uso de bucles y otros iteradores.

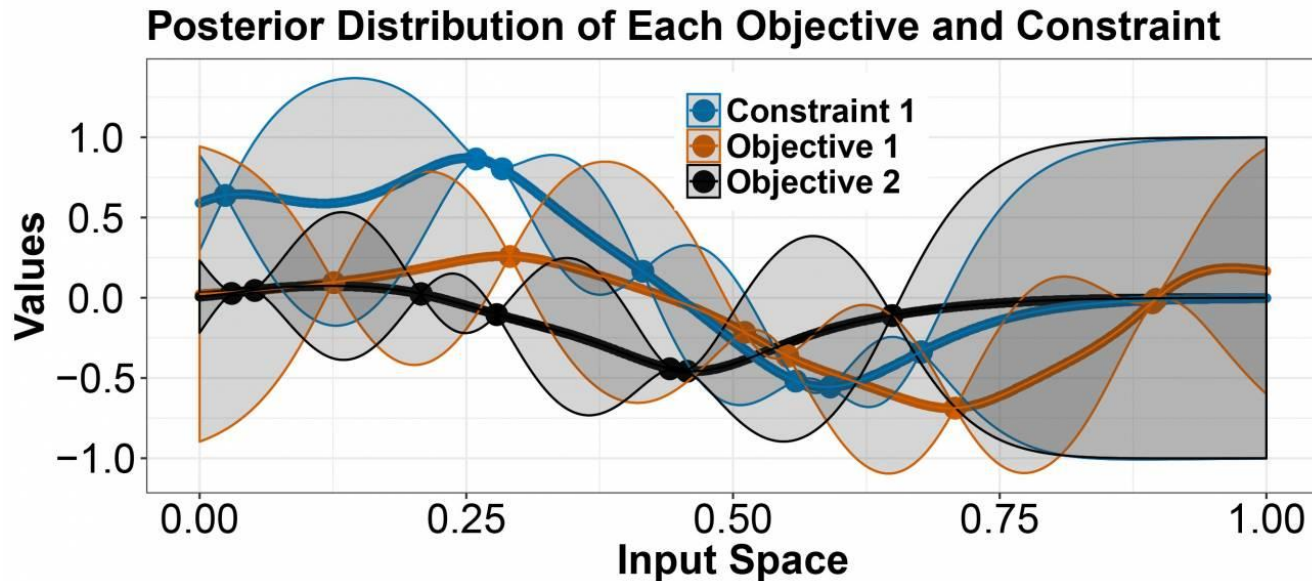
1 reference

```
public int EjemploRecursividad(int numero)
{
    if (numero == 0) return 1;
    return numero * EjemploRecursividad(numero - 1);
}
```



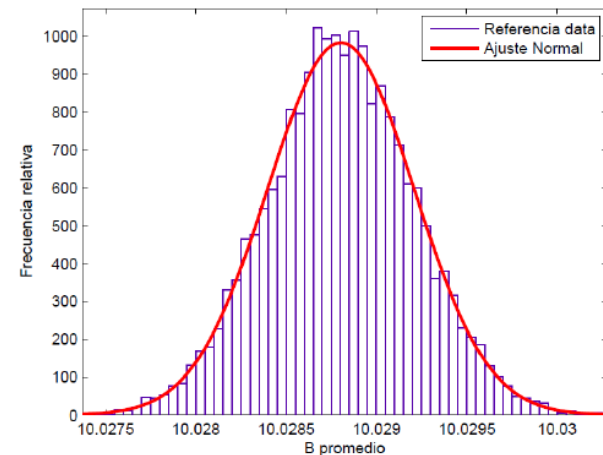
Algoritmos probabilísticos

Es una técnica que usa una fuente de aleatoriedad como parte de su lógica. Mediante un muestreo aleatorio de la entrada llega a una solución que puede no ser totalmente óptima, pero que es adecuada para el problema planteado.



Algoritmos probabilísticos

Se utiliza en situaciones con limitaciones de tiempo o memoria y cuando se puede aceptar una buena solución de media, ya que a partir de los mismos datos se pueden obtener soluciones diferentes y algunas erróneas. Para que sea más probable ofrecer una solución correcta, se repite el algoritmo varias veces con diferentes submuestras aleatorias y se comparan los resultados.



Algoritmos probabilísticos



- **Algoritmo de Montecarlo:** Dependiendo de la entrada, hay una pequeña probabilidad de que no acierte o no llegue a una solución. Se puede reducir la probabilidad de error aumentando el tiempo de cálculo.
- **Algoritmo de Las Vegas:** Se ejecuta en un periodo de tiempo concreto. Si encuentra una solución en ese tiempo ésta será correcta, pero es posible que el tiempo se agote y no encuentre ninguna solución.

Características de los algoritmos

- Un algoritmo debe ser **preciso** e indicar el orden de realización de cada paso.
- Un algoritmo debe estar **definido**. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.
- Un algoritmo debe ser **finito**. Si se sigue un algoritmo, se debe terminar en algún momento; o sea, debe tener un número finito de pasos.

Características de los algoritmos

- **Precisos:** Una de las principales características de un algoritmo es la precisión. Es decir, debe brindar el orden de los pasos que se deben realizar para poder llegar al resultado final. No puede haber ambigüedades.
- **Definidos:** Otra de las características de un algoritmo que debes tener en cuenta si deseas construir el tuyo, es que deben estar bien definidos tanto la entrada, como el proceso y la salida. ¿Qué queremos decir con esto? Que se debe lograr obtener siempre el mismo resultado al utilizar diversas variables con la misma fórmula de entrada.
- **Finitos:** Como su mismo nombre lo dice, el algoritmo debe ser finito. ¿A qué se refiere? Básicamente, que en algún momento se debe terminar el proceso; es decir, hay un número determinado de pasos que se deben seguir.

Características de los algoritmos

- **Legibles:** El texto que describe el algoritmo debe ser claro y ordenado para que pueda ser leído y comprendido de manera correcta.
- **Concretos:** Finalmente, otra de las características de un algoritmo tiene que ver con contar con una solución específica para los problemas que se desean resolver. Definitivamente, esta cualidad resulta clave, ya que los pasos diseñados para un algoritmo responden a problemas específicos.

Pasos para la solución de problemas empleando algoritmos

Los pasos para la construcción de un programa para resolver un problema mediante la computadora son:

- **Análisis** del problema
- **Diseño** del algoritmo
- **Programación**
- **Ejecución y pruebas**

El paso cero sería **Entender el problema**, parece banal, pero no lo es cuando se piensa en la gran cantidad de proyectos de computación que se desarrollaron sin haber comprendido bien para que se hacían, o cual era el problema que supuestamente iban a resolver.

Pasos para la solución de problemas empleando algoritmos

Diseño de programas

- Análisis del problema
- Diseño del algoritmo
- Verificación manual del algoritmo

En la computadora

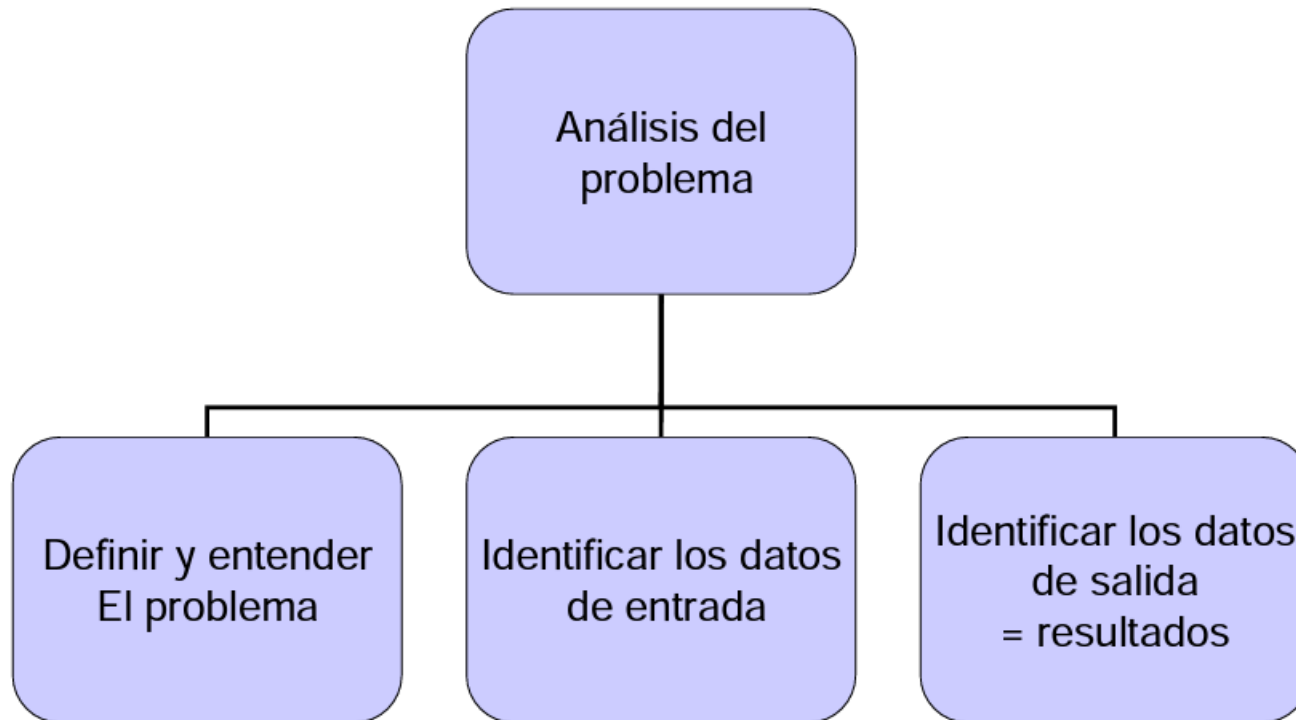
- Codificación del algoritmo
- Ejecución del programa
- Verificación del programa
- Mantenimiento (documentación)

Pasos para la solución de problemas empleando algoritmos: Análisis del problema

El análisis consiste en una clara definición del problema, donde se contemple exactamente lo que debe hacer el programa y el resultado o solución deseada. Dado que se busca una solución por computadora, se precisan especificaciones detalladas de entradas y salidas.



Pasos para la solución de problemas empleando algoritmos: Análisis del problema



Pasos para la solución de problemas empleando algoritmos: Análisis del problema

Es necesario conocer cuáles son los 5 mejores estudiantes de Lógica de Programación para darles un premio por su buen rendimiento.

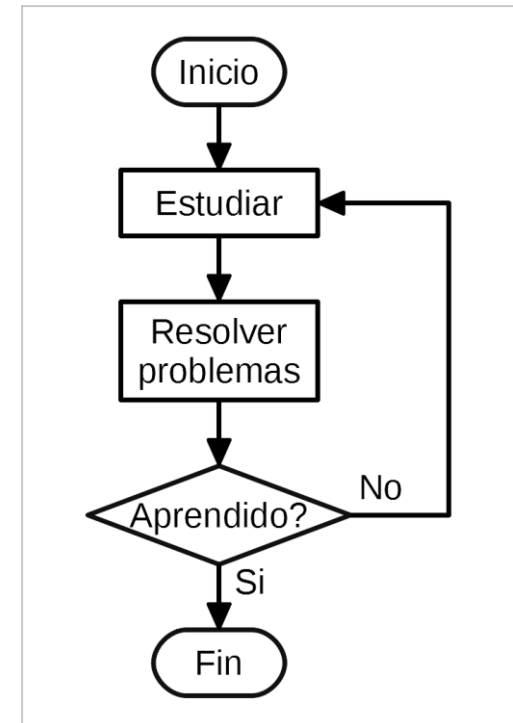
Realice el análisis del problema



Pasos para la solución de problemas empleando algoritmos: Diseño del algoritmo

Un algoritmo es un método para resolver problemas, una vez analizado el mismo se precisa diseñar un algoritmo que indique claramente los pasos a seguir para resolverlo.

Para realizar un determinado proceso, se le debe suministrar al ordenador una fórmula para la resolución de un problema (algoritmo), cuyo diseño debe ser independiente de la computadora que resuelve el problema.



Pasos para la solución de problemas empleando algoritmos: Diseño del algoritmo

- **Pseudocódigo:** es un lenguaje algorítmico, muy parecido al español, pero más conciso que permite la redacción rápida del algoritmo.
- **Diagramas de flujo:** ha sido la herramienta de programación por excelencia, y aún hoy sigue siendo muy utilizada. Es fácil de diseñar pues el flujo lógico del algoritmo se muestra en un diagrama en lugar de palabras.

Pasos para la solución de problemas empleando algoritmos: Programación



Una vez que el diagrama de flujo o el algoritmo de resolución del problema está definido se pasa a la fase de codificación del programa en cualquier lenguaje (**C, basic, cobol, pascal, Python, etc.**) cuyo resultado será el programa fuente, el cual sigue las **reglas de sintaxis** que el lenguaje escogido exija.

Pasos para la solución de problemas empleando algoritmos: Programación

Después de codificado el programa, se introduce en el ordenador mediante unos programas especiales llamados editores.

Una vez dentro del ordenador, el programa deber ser **traducido al único lenguaje** que éste entiende: **Lenguaje de máquina**. Dicha operación se realiza mediante el correspondiente programa traductor o compilador del lenguaje en el que está escrito el programa.

Pasos para la solución de problemas empleando algoritmos: Programación

Código fuente

```
{ ...  
begin  
  writeln('** Calcular la raíz cuadrada de 12  
  **');  
  writeln('Entrar x (> 0): ');  
  readln(x);  
  y := sqrt(abs(x)); (* Raíz cuadrada del  
  valor absoluto de x para evitar raíces  
  imaginarias *)  
  writeln;  
  if (x<0) then (* Si x es negativo, el  
  resultado se notifica como imaginario *)  
    writeln('La raíz cuadrada de ', x, ' es el  
  número imaginario ', y, 'i')  
  else  
    writeln('La raíz cuadrada de ', x, ' es ',  
  y);  
  writeln;  
  writeln('** Fin **');  
end.  
...}
```

Archivo: miPrimerPrograma.pas

COMPILADOR

Código máquina

```
00000010 10000000 00000000  
00000110 2060 10001000 10000000  
01000000 00000010 11001010  
00000001 00000000 00000000 2060  
00010000 10111111 11111111  
11111011 10000110 10000000  
11000000 00001011 10000001  
11000011 11100000 00000100  
00000000 00000000 00000000  
00010100 00000000 00000000  
00010111 10111000 00000010  
10000000 00000000 00000110 2060  
10001000 10000000 01000000  
00000010 11001010 00000001  
00000000 00000000 2060 00010000  
10111111 11111111 11111011  
10000110 10000000 11000000  
00001011 10000001 11000011  
11100000 00001010 00000000  
00000000 00000000 00010100  
00000000 00000000 00010111  
10111000
```

Archivo: miPrimerPrograma.exe

Pasos para la solución de problemas empleando algoritmos: Ejecución y pruebas

El hecho de haber diseñado un buen algoritmo y luego haberlo codificado en algún lenguaje de programación no significa que el programa resuelva correctamente el problema en cuestión.

Por eso, antes de dar por finalizada cualquier labor de programación, es fundamental preparar un conjunto de datos lo más representativo posible del problema, que permitan probar el programa cuando se ejecute y así verificar los resultados.

Pasos para la solución de problemas empleando algoritmos: Ejecución y pruebas

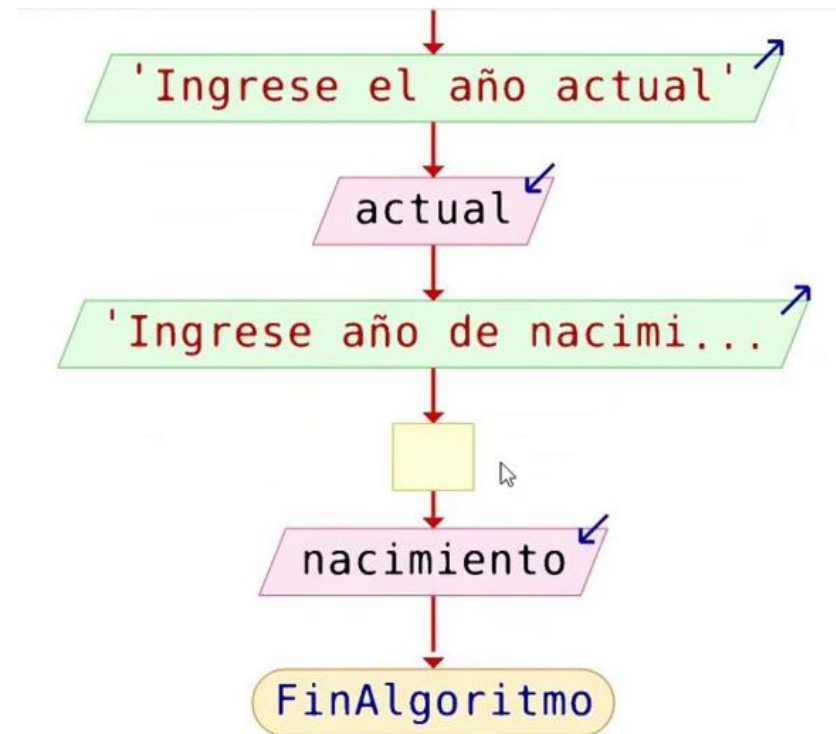


Pasos para la solución de problemas empleando algoritmos: Ejecución y pruebas



Algoritmos con estructura secuencial

Es el conjunto de movimientos en la que una acción sigue a otra, mediante una secuencia que sigue a una operación programada. Al igual que otras estructuras realizadas en un sistema, tienen una entrada y una salida.



Algoritmos con estructura secuencial

- **Asignación:** Consiste en los pasos para generar resultados en la memoria, donde son reconocidos como variables.
- **Símbolos:** Son los mandos enviados a través de un terminal de salida
- **Entrada de datos:** Es la lectura que se capta por medio de un dispositivo de entrada.
- **Cálculos:** Es el resultado generado de los datos de entrada y salida
- **Salida de datos:** Se refiere a al resultado final de toda la estructura secuencial