**UNIVERSIDAD DE SAN BUENAVENTURA**

**ANALISIS Y DISEÑO DE ALGORITMOS**

**CLASIFICACIÓN Y TIPOS DE ALGORITMOS**

**LUIS FELIPE VELASCO TAO**

**26 DE FEBRERO**

**2020**

Contenido

[CLASIFICACIÓN Y TIPOS DE ALGORITMOS 4](#_Toc33771446)

[Definición de algoritmo 4](#_Toc33771447)

[1. Algoritmos según el sistema de signos 4](#_Toc33771448)

[Algoritmos Cualitativos 4](#_Toc33771449)

[Algoritmos Cuantitativos: 5](#_Toc33771450)

[Algoritmos Computacionales 6](#_Toc33771451)

[Algoritmos no computacionales: 6](#_Toc33771452)

[2. Algoritmos según su función 7](#_Toc33771453)

[Algoritmos de ordenamiento 7](#_Toc33771454)

[Algoritmos de búsqueda 7](#_Toc33771455)

[Algoritmos de encaminamiento 8](#_Toc33771456)

[Algoritmos de marcaje 8](#_Toc33771457)

[3. Algoritmos según la estrategia que usa para llegar a la solución 9](#_Toc33771458)

[Algoritmo de Montecarlo 9](#_Toc33771459)

[Algoritmo Cotidiano 9](#_Toc33771460)

[Algoritmo de escalada 10](#_Toc33771461)

[Algoritmo voraz 10](#_Toc33771462)

[Algoritmo determinista 10](#_Toc33771463)

[Algoritmo no determinista 11](#_Toc33771464)

[Algoritmo de vuelta atrás 11](#_Toc33771465)

[Programación dinámica 12](#_Toc33771466)

[LISTA DE ALGORITMOS 12](#_Toc33771467)

[Código para un algoritmo computacional 12](#_Toc33771468)

[Código del algoritmo de ordenamiento 13](#_Toc33771469)

[Código del algoritmo de búsqueda 14](#_Toc33771470)

[Código de algoritmo de Encaminamiento 17](#_Toc33771471)

[CLASE CALLE 17](#_Toc33771472)

[CLASE MAIN 17](#_Toc33771473)

[Código de algoritmo de Marcaje 19](#_Toc33771474)

[CLASE PRODUCTOS 19](#_Toc33771475)

[CLASE MAIN 20](#_Toc33771476)

[Código de algoritmo Determinístico 23](#_Toc33771477)

[Código de algoritmo Voraz 23](#_Toc33771478)

[Algoritmo no determinístico 25](#_Toc33771479)

[Programación dinámica 27](#_Toc33771480)

[Algoritmo de escalada 29](#_Toc33771481)

[Algoritmo de vuelta atrás 30](#_Toc33771482)

[Algoritmo de Montecarlo 30](#_Toc33771483)

[Algoritmo probabilistico 31](#_Toc33771484)

[Algoritmo de Las Vegas 32](#_Toc33771485)

[Algoritmo Numerico 32](#_Toc33771486)

# CLASIFICACIÓN Y TIPOS DE ALGORITMOS

Definición de algoritmo:

Un algoritmo se define como una serie de pasos finitos y consecutivos, todos con el fin de cumplir con una tarea o realizar un cálculo matemático. Ejemplos en la vida cotidiana de un algoritmo son las recetas para preparar alimentos, instrucciones para armar un mueble o los pasos que se necesitan para llegar de un lugar a otro.

Según la función que tenga el algoritmo se puede clasificar de la siguiente forma:

1. Algoritmos según el sistema de signos

En este conjunto de algoritmos, se clasifican por el tipo de elementos gráficos utilizados para expresar sus pasos, desde letras y palabras, hasta números y símbolos matemáticos.

|  |  |  |
| --- | --- | --- |
| **ALGORITMO** | **CASO** | **EJEMPLO** |
| Algoritmos Cualitativos: son todos aquellos algoritmos que se expresan en palabras, con elementos verbales, es decir que son transmitidos de forma oral. | Supongamos que necesitamos hacer una pasta para esta noche, y llamamos a una amiga para que nos aconseje como prepararla  Resultado de imagen para pasta carbonara  1 Receta para pasta a la carbonara  Ella nos da una lista de ingredientes:   1. 300 gramos de pasta 2. 200 gramos de tocineta 3. 150 gramos de queso parmesano rallado 4. 350 gramos de cebolla 5. 150 gramos de champiñones 6. Aceite de oliva virgen 7. Mantequilla (opcional) 8. Sal y pimienta | Receta para preparar pasta a la carbonara   1. Pon a calentar agua y esperar hasta que llegue a punto de ebullición 2. Picar los champiñones en laminas 3. Picar La cebolla en cubos 4. Cortar la tocineta en cuadritos 5. Al momento de que el agua este hirviendo, añadir la pasta, una pizca de sal y si desea, una cucharada de mantequilla 6. En una sartén caliente, añada aceite de oliva, la cebolla, los champiñones, la tocineta, una pizca de sal y pimienta al gusto 7. Dejar cocinar la mezcla anterior hasta que la tocineta tome color. 8. Al momento de que la pasta este al punto en que deseemos, emplatar y añadir la cocción realizada en la sartén encima. 9. Si desea, decorar con perejil. |
| Algoritmos Cuantitativos: en estos algoritmos utilizamos valores y símbolos matemáticos para representar los pasos en un cálculo matemático. | En matemáticas, existen los procedimientos o métodos indicados para realizar distintos tipos de integrales, cada uno de estos métodos especificados para distintos tipos de casos con los que podemos encontrarnos. Uno de estos métodos es la integración por partes, el cuales utilizado para integrar funciones que se encuentran de la siguiente forma:  Esta estructura de funciones en este método lo podemos representar de la siguiente forma  Resultado de imagen para metodo de integración por partes | Método de integración por partes  ejercicios resueltos integración por partes   1. Identificar u y dv 2. Derivar u 3. Integrar dv 4. Estructurar la ecuación de la siguiente forma 5. Integramos la ecuación obtenida en   Puede que se del caso en que encontremos una función de la forma  Por lo que se repetirán los pasos 2 – 5, hasta que encontremos una función   1. Al obtener la ecuación, representamos la ecuación en términos de x. |
| Algoritmos Computacionales:  Son aquellos algoritmos que son pueden ser realizados en un computador, por medio de un lenguaje de programación. Aquí podemos encontrar desde algoritmos realizados para procesos completos, o aquellos en los cuales podemos optimizar cálculos matemáticos | En matemáticas necesitamos encontrar la raíz de distintos números, algunos son raíces enteras, otros pueden ser valores que nos den valores flotantes. En este ultimo caso, podemos usar la factorización para representar el valor en términos de una multiplicación entre enteros y una raíz inexacta.  Resultado de imagen para raices cuadradas por factorizacion | Vamos a crear un algoritmo para hallar la raíz cuadrada de un número, y si es necesaria, representarla en términos de raíz inexacta y numero flotante.  [Click aquí para ir algoritmo](#_ALGORTIMO_PARA_UNA) |
| Algoritmos no computacionales: este tipo de algoritmos son todos aquellos en donde seguimos un conjunto de órdenes para cumplir una tarea. Comúnmente son los pasos o instrucciones escritas que seguimos para instalar algo. Estos conjuntos de instrucciones suelen estar acompañados de ayudas graficas (imágenes) para que quien las use, entienda más como | Supongamos que compramos un computador y requerimos instalarlo en un estudio. En la caja del computador viene un documento en donde se nos indica las instrucciones de instalación e inicio del equipo. | Instrucciones para la instalación e inicio de un computador   1. Desempaqueta todos los componentes que vengan en la caja de forma cuidadosa. 2. Instalar la torre de la CPU en el lugar donde vaya a instalar el computador 3. Conectar el monitor con la torre con el cable DVI 4. Conectar el monitor a la toma corriente 5. Conectar el mouse y el teclado a la torre 6. Conectar los parlantes a la torre 7. Enchufar la torre a una toma corriente (Se recomienda usar un protector contra sobretensiones o un sistema de alimentación ininterrumpida) 8. Encender el computador 9. Conectar a la red (Wifi o Modem) 10. Seguir las instrucciones de inicio del sistema |

## Algoritmos según su función

|  |  |  |
| --- | --- | --- |
| Algoritmo | Caso | Ejemplo |
| Algoritmos de ordenamiento:  Este tipo de algoritmos nos ayudan a ordenar o estructurar un conjunto de elementos o datos por medio de un parámetro dato. | Este será un caso simple:  Se le pedirá al usuario que ingrese un conjunto de valores numéricos enteros y nos dé el parámetro de ordenamiento de estos   1. Para ser ordenados de forma ascendente 2. Para ser ordenados de forma descendente | [Click aquí para ver el algoritmo de ordenamiento para el caso.](#_Código_del_algoritmo) |
| Algoritmos de búsqueda:  Este conjunto de algoritmos nos ayuda a hallar de forma óptima un elemento que se encuentra en un conjunto de estos. | En este caso, el programa tendrá un conjunto de datos almacenados que corresponderá a una lista de animales, y el usuario tendrá la posibilidad de hacer las siguientes búsquedas   1. Buscar un animal por su nombre 2. Buscar animales por su inicial | [Click aquí para ver el algoritmo de ordenamiento para el caso.](#_Código_del_algoritmo_1) |
| Algoritmos de encaminamiento:  Con este tipo de algoritmos podemos encontrar el camino más optimo en un conjunto de pasos | Un caso simple aplicando este tipo de algoritmos puede ser como llegar a una calle desde otra, donde se mostrará la distancia entre los puntos, solo sí se puede llegar de la calle origen al destino cruzando una sola calle entre ellas. | [Click aquí para ver el algoritmo de encaminamiento para el caso.](#_Código_de_algoritmo) |
| Algoritmos de marcaje: este tipo de algoritmos tienen como campo de aplicación el mercado, donde los precios deben estar continuamente adaptados para las ganancias del vendedor | A modo de prueba, se creará una lista de productos y se simulará el precio que tendrás estos en N días, todo esto con relación a variables que nos proporcione el usuario para cada tipo de producto. | [Click aquí para ver el algoritmo de marcaje para el caso.](#_Código_de_algoritmo_1) |

## Algoritmos según la estrategia que usa para llegar a la solución

|  |  |  |
| --- | --- | --- |
| ALGORITMO | CASO | EJEMPLO |
| Algoritmos probabilísticos: son aquellos que nos da una respuesta relacionada con los valores contenidos, dándonos a conocer si se dio o no error en el proceso. | En este caso, vamos a pedirle al usuario que ingrese la edad de 10 personas y luego le pediremos que nos de una edad, para saber cual es la probabilidad de elegir a alguien que tenga esa edad. | [Click aquí para ver el algoritmo probabilístico para el caso.](#_Algoritmo_probabilistico) |
| Algoritmo numérico: este tipo de algoritmos presenta una solución aproximada lo más posible a la solución del problema. | Se le pide al usuario que nos de 4 valores de coma flotante para ubicarlos sobre el eje x, para darle el punto medio entre todos ellos. | [Click aquí para ver el algoritmo numérico para el caso.](#_Algoritmo_Numerico) |
| Algoritmo de las Vegas |  |  |
| Algoritmo de Montecarlo: en este tipo de algoritmos tenemos una alta probabilidad de que el valor evaluado nos dé una respuesta correcta, por lo tanto es mas fiable que el algoritmo de las vegas. | Como caso simple, se va a explicar el teorema de Fernat, el cual pretende identificar si un valor numérico entero es o no primo. | [Click aquí para ver el algoritmo de Montecarlo para el caso.](#_Algoritmo_de_Montecarlo) |
| Algoritmo Cotidiano: este conjunto de algoritmos son los que seguimos de forma natural en nuestro día a día, como yo en este momento que me encuentro escribiendo un en el teclado o usted el lector que está leyendo estas líneas. Se expresan como el conjunto natural de acciones que se hacen consciente o inconscientemente. | Como ejemplo podemos tener el conjunto de acciones que hacemos al despertarnos cada mañana. | 1. Abrir los ojos 2. Levantarnos de la cama 3. Ir a tomar un vaso con agua 4. Elegir la ropa que vamos a usar en el día 5. Ir al baño 6. Cepillarnos los dientes 7. Bañarnos 8. Secarnos 9. Vestirnos 10. …. |
| Algoritmo Heurístico: | Generar los n primeros enteros primos que el usuario pida | import java.util.Scanner;  public class Main {  static Scanner s = new Scanner(System.in);  public static void main(String[] args) {  System.out.println("INGRESE LA CANTIDAD DE n PRIMOS QUE DESEA GENERAR");  nPrimos(s.nextInt());  }  private static void nPrimos(int iter) {  int primo = 2;  while (iter > 0){  int cont = primo;  int i = 0;  while (cont > 0){  if (primo % cont == 0){  i++;  }  cont--;  }    if (i == 0){  System.out.println(" - "+primo);  }  primo++;  iter--;  }  }  } |
| Algoritmo de escalada: este tipo de algoritmos buscan encontrar valores o elementos partiendo de un valor, es decir, yendo de un elemento particular a n que hacen parte de la información necesitada | En métodos numéricos se explica el método de Newton Rapson, el cual, partiendo de un punto, encontrara la raíz o punto de corte con el eje x de una función.  Para este ejemplo usaremos la siguiente función | [Click aquí para ver el algoritmo de escalada para el caso.](#_Algoritmo_de_escalada) |
| Algoritmo voraz: este tipo de algoritmos son aplicados al momento de optimizar algoritmos que contienen n candidatos posibles a la solución del problema | Como ejemplo práctico, en una tienda se necesita saber cuál es la cantidad mínima de monedas que se le debe devolver a un cliente. Para esto se tienen monedas de $50, $100, $200, $500 y $1000    . | [Click aquí para ver el algoritmo voraz para el caso.](#_Código_de_algoritmo_2) |
| Algoritmo determinista: en estos algoritmos se ve una estructura lineal, donde los pasos determinan la acción del siguiente. Son algoritmos de forma lineal, en donde los n datos de salida son directamente relacionados con las n salidas | De forma simple podemos crear un programa para solucionar una operación como la siguiente:  Pidiéndole al usuario que ingrese los valores de x & y.  Otro ejemplo que podemos ver es como hallar el Maximo Común Divisor entre dos números por medio de | import java.util.Scanner;  public class Main {  static Scanner s = new Scanner(System.in);  public static void main(String[] args) {  System.out.println("INGRESE LOS VALORES DE x Y y para la siguiente operaciòn: \n"  + " (x+y)(x-y) ");  int x = s.nextInt(), y = s.nextInt(), z = (x + y)\*(x-y);  System.out.println("("+x+" + "+y+")("+x+" - "+y+") = "+z);  }  }  [Click aquí para ver el algoritmo determinista para el caso.](#_Código_de_algoritmo_3) |
| Algoritmo no determinista: este conjunto de algoritmos los vemos presente cuando trabajamos con estructuras de datos no lineales, o con sistemas donde el flujo del sistema no es tan preciso, es decir, en el camino, las n entradas, podrán entrar cada una a distintas líneas en el flujo y generar n salidas no tan directamente condicionadas con las variables de entrada. | Fácilmente podemos crear un algoritmo que nos dé el mayor valor de los 4 valores ingresados  W X Y Z  Y determinar el orden de mayor a menor de los valores ingresados. | [Click aquí para ver el algoritmo no determinista para el caso.](#_Algoritmo_no_determinístico) |
| Algoritmo de vuelta atrás: este tipo de algoritmos son usados en estudios donde se necesita analizar un elemento en base de otro. Es como realizar la deducción de un dato por medio de otro y como este último se ve afectado por distintas variables | En un programa pequeño, se pretende saber el precio unitario libre de impuestos, costos adicionales u otros pagos, es decir, el valor que gana empresa al vender un producto. | [Click aquí para ver el algoritmo de vuelta atras para el caso.](#_Algoritmo_de_vuelta) |
| Programación dinámica: se denomina programación a la utilización de técnicas que permitan ver el problema a programar como un conjunto de subprogramas que cooperen entre si para la optimización del programa | Un programa pequeño donde se muestra el uso de la programación dinámica es uno donde se le pedirá al usuario ingresar arreglo con valores enteros, no se dejará ingresar valores ya ingresados o que sean cero, y luego de esto, se imprimirá el arreglo ordenado de mayor a menor y se imprimirán los valores impares y pares ingresados; | [Click aquí para ver el programa de programación para el caso.](#_Programación_dinámica) |

# LISTA DE ALGORITMOS

## Código para un algoritmo computacional

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

System.out.println("INGRESE EL VALOR PARA LA RAIZ CUADRADA:");

int n = s.nextInt();

System.out.println("La raíz representada por multiplos es:\n"+ metodo1(n)+" = "+Math.sqrt(n));

}

private static String metodo1(int n) {

String raiz = "";

int aux = n, x = 2, con = 0, i = 0;

int[] numeros = new int [10];

// Ciclo While para hallar los múltiplos del valor ingresado

while (aux != 1) {

if (aux%x == 0){

numeros[i] = x;

aux /= x;

i++;

} else {

x++;

}

}

i = 0;

// Ciclo While para hallar las constantes enteras y las raíces inexactas del valor n

while (numeros[i] != 0) {

if (i + 1 < 10) {

if (numeros[i] == numeros[i + 1] && numeros[i] \* numeros[i + 1] != 0) {

raiz += "("+numeros[i] + ")";

i = i + 2;

}

if (numeros[i] != 0 && numeros[i] \* numeros[i + 1] == 0 || numeros[i] != numeros[i + 1]) {

raiz += "(sqrt(" + numeros[i] + "))";

i++;

}

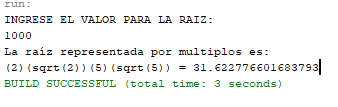
}

}

return raiz;

}

}



### Código del algoritmo de ordenamiento

import java.util.Scanner;

public class Main {

static Scanner s = new Scanner(System.in);

public static void main(String[] args) {

int[] cod = new int[10];

System.out.println("Ingrese el tipo de ordenamiento que desea realizar: \n 1.Ascendente\n 2.Descendente ");

switch (s.nextInt()) {

case 1:

cod = ordenamiento(llenarArreglo(), 1);

System.out.println("LISTA DE CODIGOS ORDENADOS DE FORMA ASCENDENTE");

for (int i = 0; i < cod.length; i++) {

System.out.println("CODIGO [" + i + "]" + cod[i]);

}

break;

case 2:

cod = ordenamiento(llenarArreglo(), 2);

System.out.println("LISTA DE CODIGOS ORDENADOS DE FORMA DESCENDENTE");

for (int i = 0; i < cod.length; i++) {

System.out.println("CODIGO [" + i + "]" + cod[i]);

}

break;

default:

System.out.println("Valor ingresado no valido");

}

}

private static int[] llenarArreglo() {

int[] codigos = new int[10];

for (int i = 0; i < codigos.length; i++) {

System.out.println("CODIGO [" + i + "]");

codigos[i] = s.nextInt();

}

return codigos;

}

private static int[] ordenamiento(int[] codigos, int orden) {

for (int i = 0; i < codigos.length; i++) {

for (int j = i + 1; j < codigos.length; j++) {

if (orden == 1) {

if (codigos[i] > codigos[j]) {

int aux = codigos[i];

codigos[i] = codigos[j];

codigos[j] = aux;

}

} else {

if (codigos[i] < codigos[j]) {

int aux = codigos[i];

codigos[i] = codigos[j];

codigos[j] = aux;

}

}

}

}

return codigos;

}

}

### Código del algoritmo de búsqueda

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

String[] animales = {"Perro", "Gato", "Elefante", "Jirafa", "Gusano", "Araña", "Pelicano", "Pato", "Mariposa", "Avestruz"};

int opc;

do {

System.out.println("----ANIMALES----\n"

+ "1. Mostrar todo\n"

+ "2. Buscar un animal por su nombre\n"

+ "3. Buscar animales por inicial\n"

+ "4. Salir\n");

opc = s.nextInt();

switch (opc) {

case 1:

for (int i = 0; i < animales.length; i++) {

System.out.println(i + ". " + animales[i]);

}

System.out.println("");

break;

case 2:

System.out.println("Ingrese el animal que quiere encontrar");

String busqueda = s.next();

boolean flag = false;

for (String animale : animales) {

if (animale.toUpperCase().equals(busqueda.toUpperCase())) {

flag = !flag;

break;

}

}

if (flag) {

System.out.println(busqueda + " si se encuentra en la lista");

} else {

System.out.println(busqueda + " no se encuentra en la lista");

}

break;

case 3:

System.out.println("Ingrese la inicial (en MAY) de los animales que quiere buscar");

char b = s.next().charAt(0);

String lista = "";

for (String animale : animales) {

if (animale.charAt(0) == b) {

lista += " - " + animale + "\n";

}

}

if (!"".equals(lista)) {

System.out.println("LISTA DE ANIMALES CON LA LETRA " + b + "\n" + lista);

} else {

System.out.println("No hay animales que inicien por la letra " + b);

}

break;

case 4:

System.out.println("FIN PROGRAMA");

break;

default:

System.out.println("Valor invalido");

break;

}

} while (opc != 4);

}

### Código de algoritmo de Encaminamiento

#### CLASE CALLE

public class Calle {

String nombre;

double distancia;

String siguiente;

public Calle(String nombre, double distancia, String siguiente) {

this.nombre = nombre;

this.distancia = distancia;

this.siguiente = siguiente;

}

public Calle(){

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public double getDistancia() {

return distancia;

}

public void setDistancia(double distancia) {

this.distancia = distancia;

}

public String getSiguiente() {

return siguiente;

}

public void setSiguiente(String siguiente) {

this.siguiente = siguiente;

}

@Override

public String toString() {

return nombre+" - Extension : "+distancia+ " km. - Calle Siguiente : "+siguiente+"\n";

}

}

#### CLASE MAIN

import java.util.ArrayList;

import java.util.Scanner;

import usb.Calle;

public class Main {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

Calle calle1 = new Calle("Calle 38", 8.4, "Calle 100");

Calle calle2 = new Calle("Calle 100", 10.5, "Calle 50");

Calle calle3 = new Calle("Calle 78", 8.4, "Calle 3");

Calle calle4 = new Calle("Calle 50", 8.4, "Calle 12");

Calle calle5 = new Calle("Calle 101", 8.4, "Calle 100");

Calle calle6 = new Calle("Calle 12", 8.4, "Calle 6");

Calle calle7 = new Calle("Calle 3", 8.4, "Calle 100");

Calle calle8 = new Calle("Calle 4", 8.4, "Calle 6");

Calle calle9 = new Calle("Calle 5", 8.4, "Calle 12");

Calle calle10 = new Calle("Calle 6",8.4,"Ciega");

Calle[] calles = new Calle[11];

calles[1] = calle1;calles[2] = calle2;calles[3] = calle3;calles[4] = calle4;calles[5] = calle5;

calles[6] = calle6;calles[7] = calle7;calles[8] = calle8;calles[9] = calle9;calles[10] = calle5;

System.out.println("----Lista de cale----");

imprimirLista(calles);

System.out.println("Ingrese la calle origen y la calle destino");

String calleOrigen = s.nextLine(),calleDestino = s.nextLine();

encaminamiento(calleOrigen,calleDestino,calles);

}

private static void imprimirLista(Calle[] calles) {

for (int i = 1; i < calles.length; i++) {

System.out.println(calles[i].toString());

}

}

private static void encaminamiento(String calle1, String calle2, Calle[] calles) {

Calle calleOrigen = existenciaCalle(calle1, calles);

Calle calleDestino = existenciaCalle(calle2, calles);

if (calleOrigen != null && calleDestino != null) {

distanciarecorrida(calleOrigen, calleDestino, calles);

} else {

System.out.println(" Por favor revice los parametros ingresados");

}

}

private static Calle existenciaCalle(String calleOrigen, Calle[] calles) {

for (int i = 1; i < calles.length; i++) {

if (calles[i].getNombre().equals(calleOrigen)) {

return calles[i];

}

}

return null;

}

private static void distanciarecorrida(Calle calleOrigen, Calle calleDestino, Calle[] calles) {

Calle interm = existenciaCalle(calleOrigen.getSiguiente(), calles);

double dist1 = 0.0, dist2 = 0.0;

if (calleOrigen.getSiguiente().equals(calleDestino.getNombre())) {

dist1 = calleOrigen.getDistancia();

} else if (interm.getSiguiente().equals(calleDestino.getNombre())) {

dist2 = calleOrigen.getDistancia() + interm.getDistancia();

}

if (dist1 != 0.0 && dist2 == 0.0) {

System.out.println("Desde la calle: " + calleOrigen.getNombre() + " a la calle: " + calleDestino.getNombre() + " solo son " + calleOrigen.getDistancia());

} else if (dist2 != 0.0 && dist1 == 0.0) {

System.out.println("Desde la calle origen hasta la calle destino\nse hace el siguiente recorrido con una distancia de " + dist2 + " km. : \n"

+ "1. " + calleOrigen.getNombre() + "\n"

+ "2. " + interm.getNombre() + "\n"

+ "2. " + calleDestino.getNombre() + "\n");

} else {

System.out.println("No hay forma de llegar desde la calle : " + calleOrigen.getNombre() + " hasta la calle :" + calleDestino.getNombre() + " cruzando solo dos calles");

}

}

### Código de algoritmo de Marcaje

#### CLASE PRODUCTOS

public class Producto {

int cod;

String nombre;

double precio;

public Producto(int cod, String nombre, double precio) {

this.cod = cod;

this.nombre = nombre;

this.precio = precio;

}

public int getCod() {

return cod;

}

public void setCod(int cod) {

this.cod = cod;

}

public String getNombre() {

return nombre;

}

public void setNombre(String nombre) {

this.nombre = nombre;

}

public double getPrecio() {

return precio;

}

public void setPrecio(double precio) {

this.precio = precio;

}

@Override

public String toString() {

return " - "+nombre+" - Codigo:"+cod+" Precio : $"+precio;

}

}

#### CLASE MAIN

import java.util.Scanner;

public class Main {

static Scanner s = new Scanner(System.in);

public static void main(String[] args) {

Producto producto1 = new Producto(1010465, "Leche Alpina", 1600.15);

Producto producto2 = new Producto(1010465, "Arroz Diana", 1550.20);

Producto producto3 = new Producto(1010465, "Arroz FlorHuila", 1650.00);

Producto producto4 = new Producto(1010465, "Arroz Supremo", 1700.15);

Producto producto5 = new Producto(1010465, "Leche Algarra", 1500.15);

Producto producto6 = new Producto(1010465, "Aceite Gourmet", 10400.15);

Producto producto7 = new Producto(1010465, "Aceite Premier", 15471.15);

Producto producto8 = new Producto(1010465, "Torta Ramo", 15471.15);

Producto[] listaProductos = new Producto[8];

listaProductos[0] = producto1;

listaProductos[1] = producto2;

listaProductos[2] = producto3;

listaProductos[3] = producto4;

listaProductos[4] = producto5;

listaProductos[5] = producto6;

listaProductos[6] = producto7;

listaProductos[7] = producto8;

System.out.println("INGRESE LOS DIAS QUE QUIERA SIMULAR:");

int dias = s.nextInt();

System.out.println("-----LISTA INICIAL-----");

imprimirLis(listaProductos);

variacion(listaProductos, dias);

}

public static void imprimirLis(Producto[] listaProductos) {

for (Producto aux : listaProductos) {

System.out.println(aux.toString());

}

}

public static void variacion(Producto[] listaProductos, int dias) {

int x = 0;

while (dias != x) {

System.out.println("-----DIA " + (x + 1) + "-----\n Ingrese la variación de los precios al publico para los productos:\n"

+ "1.Ingrese la variacion de la leche\n"

+ "2. Ingrese la variacion de los arroces\n"

+ "3. Ingrese la variacion de los aceites");

double leche = s.nextDouble(), arroz = s.nextDouble(), aceite = s.nextDouble();

for (Producto aux : listaProductos) {

asignación(leche, arroz, aceite, aux);

}

System.out.println("------LISTA DE PRODUCTOS DIA " + (x + 1) + "-----");

imprimirLis(listaProductos);

x++;

}

}

private static void asignación(double leche, double arroz, double aceite, Producto aux) {

double variacion = 0.0;

if (aux.getNombre().contains("Arroz")) {

variacion = Math.round(aux.getPrecio() \* (arroz + 0.01));

aux.setPrecio(variacion);

} else if (aux.getNombre().contains("Leche")) {

variacion = Math.round(aux.getPrecio() \* (leche) + 3.4);

aux.setPrecio(variacion);

} else if (aux.getNombre().contains("Aceite")) {

variacion = Math.round(aux.getPrecio() \* (aceite + 0.14));

aux.setPrecio(variacion);

} else {

System.out.println("------------------- PRODUCTO SIN CLASIFICACIÓN-------------------\n"

+ aux.toString());

}

}

}

### Código de algoritmo Determinístico

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

System.out.println("Ingrese los dos valores para hallar su M.C.D ");

int num1 = s.nextInt(), num2 = s.nextInt(),mcd = minimoComunMultiplo(num1, num2);

if (mcd != 1){

System.out.println("El MCD entre "+num1+" y "+num2+" es "+mcd);

} else {

System.out.println("Entre los numeros "+num1+" y "+num2+" no existe un MCD ---->"+mcd);

}

}

public static int minimoComunMultiplo(int num1, int num2) {

int aux = 0;

while (num2 != 0) {

aux = num2;

num2 = num1 % num2;

num1 = aux;

}

return num1;

}

}

### Código de algoritmo Voraz

import java.util.Scanner;

public class Main {

static int[] monedas = {1000, 500, 200, 100, 50};

static int[] vueltas = {0, 0, 0, 0, 0};

static Scanner s = new Scanner(System.in);

public static void main(String[] args) {

System.out.println("INGRESE EL VALOR DE VUELTAS ");

vueltas(s.nextInt());

imprimir();

}

private static void vueltas(int v) {

for (int i = 0; v != 0; i++) {

if (v % monedas[i] == 0) {

vueltas[i] = v / monedas[i];

break;

} else {

vueltas[i] = v / monedas[i];

v = v % monedas[i];

}

}

}

private static void imprimir (){

for (int i = 0; i < vueltas.length; i++){

System.out.println("Monedas de "+monedas[i]+" son de "+vueltas[i]);

}

}

}

### Algoritmo no determinístico

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

Scanner s = new Scanner(System.in);

System.out.println("----------MAYOR O MENOR----------------"

+ "\nINGRESE CUATRO VALORES NUMERICOS:");

mayorMenor(s.nextInt(), s.nextInt(), s.nextInt(), s.nextInt());

}

private static void mayorMenor(int w, int x, int y, int z) {

String resul = "";

if (w == x && w == y && w == z && x == y && x == z && y == z) {

System.out.println(w + " = " + x + " = " + y + " = " + z);

} else if (w > x && w > y && w > x) {

resul += w + " > ";

if (x > y && x > z) {

resul += x + " > ";

if (y > z) {

resul += y + " > " + z;

} else {

resul += z + " > " + y;

}

} else if (y > x && y > z) {

resul += y + " > ";

if (x > z) {

resul += x + " > " + z;

} else {

resul += z + " > " + x;

}

} else if (z > x && z > y) {

resul += z + " > ";

if (x > y) {

resul += x + " > " + y;

} else {

resul += y + " > " + x;

}

}

} else if (x > w && x > y && x > z) {

resul += x + " > ";

if (w > y && w > z) {

resul += w + " > ";

if (y > z) {

resul += y + " > " + z;

} else {

resul += z + " > " + y;

}

} else if (y > w && y > z) {

resul += y + " > ";

if (w > z) {

resul += w + " > " + z;

} else {

resul += z + " > " + w;

}

} else if (z > w && z > y) {

resul += z + " > ";

if (w > y) {

resul += w + " > " + y;

} else {

resul += y + " > " + w;

}

}

} else if (y > w && y > x && y > z) {

resul += y + " > ";

if (w > x && w > z) {

resul += w + " > ";

if (x > z) {

resul += x + " > " + z;

} else {

resul += z + " > " + x;

}

} else if (x > w && x > z) {

resul += x + " > ";

if (w > z) {

resul += w + " > " + z;

} else {

resul += z + " > " + w;

}

} else if (z > w && z > x) {

resul += z + " > ";

if (w > x) {

resul += w + " > " + x;

} else {

resul += x + " > " + w;

}

}

} else {

resul += z + " > ";

if (w > x && w > y) {

resul += w + " > ";

if (x > y) {

resul += x + " > " + y;

} else {

resul += y + " > " + x;

}

} else if (x > w && x > y) {

resul += x + " > ";

if (w > y) {

resul += w + " > " + y;

} else {

resul += y + " > " + w;

}

} else if (y > w && y > x) {

resul += y + " > ";

if (w > x) {

resul += w + " > " + x;

} else {

resul += x + " > " + w;

}

}

}

System.out.println(resul);

}

}

### Programación dinámica

import java.util.Scanner;

public class Main {

static Scanner s = new Scanner(System.in);

static int [] arreglo = new int [10];

public static void main(String[] args) {

System.out.println("------PROGRAMA DE ARREGLOS NUMERICOS------");

imprimirArreglo(arregloOrdenado(llenarArray()));

System.out.println("------PARES E IMPARES------");

paresInpares();

}

private static int[] llenarArray() {

for (int i = 0; i < arreglo.length; i++) {

System.out.println("-Elemento de la posicion : " + (i + 1));

boolean flag;

do {

int v = s.nextInt();

flag = existenciaNumero(v);

if (v == 0) {

System.out.println("Ingrese un valor diferente de cero");

} else {

if (flag) {

System.out.println(v + " Ya existe en la lista, ingrese otro valor");

} else {

arreglo[i] = v;

}

}

} while (flag);

}

return arreglo;

}

public static int [] arregloOrdenado(int [] arreglo){

int aux = 0;

System.out.println("------ARREGLO NUMERICO ORDENADO------");

for (int i = 0; i < arreglo.length;i++){

for(int j = i+1; j < arreglo.length;j++){

if (arreglo[i] < arreglo[j]){

aux = arreglo[i];

arreglo[i] = arreglo[j];

arreglo[j] = aux;

}

}

}

return arreglo;

}

public static void paresInpares (){

String pares = "Valores Pares = [",inpares = "Valores Inpares = [";

for (int i = 0; i < arreglo.length;i++){

if (arreglo[i] % 2 == 0){

pares += arreglo[i]+" ";

} else {

inpares +=" "+ arreglo[i]+" ";

}

}

if ("Valores Pares = [".equals(pares)){

System.out.println("No hay elementos pares en la lista");

} else{

System.out.println(pares+"]");

}

if ("Valores Inpares = [".equals(inpares)){

System.out.println("No hay elementos pares en la lista");

} else{

System.out.println(inpares+"]");

}

}

public static void imprimirArreglo(int [] arreglo){

for (int i = 0; i < arreglo.length;i++){

System.out.println(" - "+arreglo[i]);

}

}

private static boolean existenciaNumero(int v) {

for (int i = 0; i < arreglo.length;i++){

if (v == arreglo[i]){

return true;

}

}

return false;

}

}

### Algoritmo de escalada

import java.util.Scanner;

public class Main {

static Scanner s = new Scanner(System.in);

static double tol = 0.0;

static double xSiguiente = 0.0;

static double x0 = 0.0;

static double error = 0.0;

public static void main(String[] args) {

System.out.println("----------------------METODO DE NEWTON-RHAPSON----------------------\n"

+ "Ingrese el punto de partida(flotante) para evaluar la siguiente función\n"

+ " 18x^3-8x^2+7x-4");

x0 = s.nextDouble();

System.out.println("Ingrese la tolerancia a error de la función");

tol = s.nextDouble();

metodoNewton();

System.out.println("----------------------SOLUCÓN----------------------\n"

+ "RAÍZ MAS O MENOS EN EL PUNTO X = "+x0);

}

private static void metodoNewton() {

System.out.println("VALOR INICIAL = " + x0);

do {

double f1 = f(x0), f2 = fp(x0);

xSiguiente = x0 - ( f1/ f2);

if (xSiguiente > x0){

error = xSiguiente-x0;

} else {

error = x0-xSiguiente;

}

System.out.println("x = " + x0 + ""

+ " f(x) = " + f1 + ""

+ " x(i+1) = " + xSiguiente + ""

+ " f(x(i+1)) = " + f(xSiguiente) + ""

+ " error = " + error);

x0 = xSiguiente;

} while (tol < error);

}

public static double f (double x){

return (18\*x\*x\*x)-(8\*x\*x)+(7\*x)-4;

}

public static double fp (double x){

return (54\*x\*x)-(16\*x)+7;

}

### Algoritmo de vuelta atrás

import java.util.Scanner;

public class Main {

static Scanner s = new Scanner(System.in);

public static void main(String[] args) {

System.out.println("INGRESE EL PRECIO DEL PRODUCTO:");

vueltaAtras(s.nextDouble());

}

private static void vueltaAtras(double precio) {

// Se halla el valor del IVA

double iva = (precio\*0.19);

precio = precio -iva;

System.out.printf("IVA 19%% = %.2f \n",iva);

// Se halla el porcentaje para los trabajadores

double trabajador = (precio\*0.005);

precio = precio -trabajador;

System.out.printf("PAGO A TRABAJADORES 0.5%% = %.2f \n",trabajador);

// Se halla el valor de produccion

double producion = (precio\*0.079);

precio = precio -producion;

System.out.printf("COSTO DE PRODUCCION 0.5%% = %.2f \n",producion);

System.out.printf("PRECIO LIBRE DE PAGOS, IMPUESTOS Y DEMAS COSTOS DEL PRODUCTO = %.2f \n",precio);

}

}

### Algoritmo de Montecarlo

import java.util.Scanner;

public class Main {

static Scanner s = new Scanner(System.in);

public static void main(String[] args) {

int [] arreglo = new int [5];

arreglo = llenararreglo(arreglo);

validarnum(arreglo);

}

private static int[] llenararreglo(int[] arreglo) {

for (int i = 0; i < arreglo.length;i++){

arreglo[i] = s.nextInt();

}

return arreglo;

}

private static boolean comprobarPrimo(int x){

int a = x-2;

if (Math.pow(a, x-1)%x == 1){

return true;

} else{

return false;

}

}

private static void validarnum(int [] arreglo){

for(int i = 0; i < arreglo.length;i++){

if (comprobarPrimo(arreglo[i])){

System.out.println("EL NUMERO ES PRIMO");

} else{

System.out.println("EL NUMERO NO ES PRIMO");

}

}

}

}

### Algoritmo probabilistico

import java.util.Scanner;

public class Main {

static Scanner s = new Scanner(System.in);

public static void main(String[] args) {

int [] arreglo = new int [10];

System.out.println("INGRESE LA EDAD DE LAS PERSONAS :");

arreglo = llenararreglo(arreglo);

System.out.println("INGRESE LA EDAD LA CUAL QUIERE SABER LA PROBABILIDAD DE ENCONTRAR EL VALOR");

validarEdad(arreglo,s.nextInt());

}

private static int[] llenararreglo(int[] arreglo) {

for (int i = 0; i < arreglo.length;i++){

arreglo[i] = s.nextInt();

}

return arreglo;

}

private static int validarDato(int x, int y){

if (x == y){

return 1;

} else {

return 0;

}

}

private static void validarEdad(int [] arreglo, int num){

int aux = 0;

double res = 0.0;

for(int i = 0; i < arreglo.length;i++){

aux = aux + validarDato(num, arreglo[i]);

}

if (aux == 0) {

System.out.println("No hay ninguna probabilidad de que esta edad este en la lista");

} else {

res = (double) (aux/10.0)\*100;

System.out.println("LA PROBABILIDAD DE QUE HAYA ALGUIEN QUE TENGA "+num+" es igual a "+aux+" / "+arreglo.length+" = "+res+" %");

}

}

### Algoritmo de Las Vegas

### Algoritmo Numerico

import java.util.Scanner;

public class Main {

static Scanner s = new Scanner(System.in);

public static void main(String[] args) {

System.out.println("Ingrese 4 numeros de coma flotante para ubicar sobre el eje X y hallar su punto medio");

puntomedio (s.nextDouble(),s.nextDouble(),s.nextDouble(),s.nextDouble());

}

private static void puntomedio(double a, double b, double c, double d) {

double puntoMedio = (a + b + c + d)/4;

System.out.printf("PUNTO MEDIO APROX. A DOS CIFRAS DECIMALES SIGNIFICATIVAS= %.2f \n",puntoMedio);

}

}