**Taller N° 2: Análisis y diseño de algoritmos**

**Presentado a: Ing. Luis Barreto**

**Juan David Gonzalez Dimaté**

**Ing. Sistemas**

**Bogotá D.C**

**2020-1**

|  |  |  |
| --- | --- | --- |
| Algoritmo | Caso | Ejemplo |
| **Algoritmo Cualitativo** | Supongamos que queremos cocinar un huevo, el algoritmo que nos podría ayudar con esta tarea puede ser este. | Inicio CocinarHuevo  Encender Sarten  Colocar aceite en Sarten  Agarrar Huevo  Golpear Huevo con Mesón  Abrir Huevo sobre Sartén  Colocar Sal  Batir Huevo  Esperar a que Huevo = Esponjoso  Si Huevo == Esponjoso  Entonces Servir  Sino  Volver a Batir Huevo  Fin CocinarHuevo |
| **Algoritmo Cuantitativo** | Supongamos que hay una calculadora que halla el número faltante en una ecuación básica (tal como 2+? = 5). Entonces, dependiendo de la operación y el término que queramos despejar, realizaremos un procedimiento matemático.  En Java, el código puede ser de esta manera. | break;  case "\*":  if (prim.equals("x")) {  y0 = Double.parseDouble(seg);  z0 = Double.parseDouble(res);  res1 = z0/y0;  break;  } else if (seg.equals("x")) {  x0 = Double.parseDouble(prim);  z0 = Double.parseDouble(res);  res1 = z0/x0;  break;  } else if (res.equals("x")){  x0 = Double.parseDouble(prim);  y0 = Double.parseDouble(seg);  res1 = x0\*y0;  break;  }  break;  case "/":  if (prim.equals("x")) {  y0 = Double.parseDouble(seg);  z0 = Double.parseDouble(res);  res1 = z0\*y0;  break;  } else if (seg.equals("x")) {  x0 = Double.parseDouble(prim);  z0 = Double.parseDouble(res);  res1 = x0/z0;  break;  } else if (res.equals("x")){  x0 = Double.parseDouble(prim);  y0 = Double.parseDouble(seg);  res1 = x0/y0;  }  break;  default:  return 0;  }  if (ope.equals("/")){  return res1;  }else{  return res0;  }  }  public double hallarNumero(String prim, String seg, String res, String ope) {  int x, y, z, res0 = 0;  double x0,y0,z0,res1 = 0;  switch (ope) {  case "+":  if (prim.equals("x")) {  y = Integer.parseInt(seg);  z = Integer.parseInt(res);  res0 = z - y;  break;  } else if (seg.equals("x")) {  x = Integer.parseInt(prim);  z = Integer.parseInt(res);  res0 = z - x;  break;  } else if (res.equals("x")){  x = Integer.parseInt(prim);  y = Integer.parseInt(seg);  res0 = x + y;  break;  }  break;  case "-":  if (prim.equals("x")) {  y = Integer.parseInt(seg);  z = Integer.parseInt(res);  res0 = z + y;  } else if (seg.equals("x")) {  x = Integer.parseInt(prim);  z = Integer.parseInt(res);  res0 = x - z;  break;  } else if (res.equals("x")){  x = Integer.parseInt(prim);  y = Integer.parseInt(seg);  res0 = x - y;  break;  } |
| **Algoritmo Computacional** | Supongamos que hay un algoritmo que lee las notas de un alumno, saca el promedio de dichas notas y determina si el alumno aprueba, toma un examen o reprueba. En Java, el código puede ser de esta manera. | public static void main(String[] args) {  double not, prom = 0;  List<Double> notas = new ArrayList<>();  Scanner lc = new Scanner(System.in);  for (int i = 0; i < 4; i++) {  not = lc.nextDouble();  notas.add(not);  }  for (int i = 0; i < notas.size(); i++) {  prom = prom + notas.get(i);  }  prom = prom / notas.size();  if (prom > 7.0) {  System.out.println("Alumno aprobado");  } else if (prom < 5.0) {  System.out.println("Alumno reprobado\n");  } else {  System.out.printf("Media: %.1f ",prom);  System.out.println("\nAlumno en examen");  not = lc.nextDouble();  System.out.printf("Nota de examen: %.1f ",not);  prom = (prom + not)/2;  if(prom >= 5.0){  System.out.println("\nAlumno aprobado");  System.out.printf("Media final: %.1f ",prom);  }else{  System.out.println("\nAlumno reprobado");  System.out.printf("Media final: %.1f",prom);  }  }  } |
| **Algoritmo no Computacional** | Supongamos que queremos leer un libro en una habitación oscura. El algoritmo que nos ayudaría con esta tarea puede ser este. | Inicio LeerLibro  Si Libro está en mano  Sí luz está encendida  Leer Libro  Sino  Encender Luz  Volver a Inicio  Sino  Agarrar Libro  Volver a Inicio  Fin LeerLibro |
| **Algoritmo de Ordenamiento** | Supongamos que nos dan 3 números enteros y se requiere organizarlos ascendentemente. | public static void main(String[] args) {  Scanner lc = new Scanner(System.in);  int a;  List<Integer> lista = new ArrayList<>();  List<Integer> lisaux = new ArrayList<>();  for(int i = 0; i<3; i++){  a = lc.nextInt();  lista.add(a);  lisaux.add(a);  }  Collections.sort(lista);  for (Integer num : lista) System.out.println(num);  System.out.println(" ");  for (Integer num2 : lisaux) System.out.println(num2);  } |
| **Algoritmo de Búsqueda** | Supongamos que necesitamos encontrar la habitación de una paciente llamada “Ana” en una clínica que atiende a 9 pacientes más. | public static void main(String[] args) {  Scanner lc = new Scanner(System.in);  String nom = "";  HashMap<String,Integer> pacientes = new HashMap<>();  pacientes.put("Alberto", 5);  pacientes.put("Juliana", 1);  pacientes.put("Martín", 7);  pacientes.put("Camila", 9);  pacientes.put("Ana", 10);  pacientes.put("Daniella", 4);  pacientes.put("José", 2);  pacientes.put("Alvaro", 6);  pacientes.put("Camila", 3);  pacientes.put("Felipe", 8);  nom = lc.next();  System.out.println(pacientes.get(nom));  } |
| **Algoritmo de encaminamiento** | Supongamos que estamos en un país con 6 ciudades (A, B, C, D, E, F), y queremos hacer un viaje entre esas ciudades, fijando una ciudad origen y una ciudad destino. Existe una distancia en kilómetros entre cada ciudad, y se busca que al fijar un curso destino (por ejemplo de A a D) se haga un recorrido con el menor número posible de kilómetros. |  |
| **Algoritmo de Marcaje** | Supongamos que un estudiante de Sistemas quiere comprar un computador para programar. En la tienda de tecnología le ofrecen 3 opciones: Un PC con Procesador de alto rendimiento, un PC con un Disco duro de alta capacidad y un PC con Procesador de buen rendimiento y una memoria con suficiente capacidad. El vendedor le dice que pueden negociar el precio del equipo conforme a su presupuesto y a lo que más le convenga. El código en Java puede ser así. | public static void main(String[] args) {  Scanner lc = new Scanner(System.in);  String producto = "", price = "";  int prize = 0;  double precio = 0, presupuesto = 0;  do {  System.out.println("¿Qué producto le conviene más?\n"  + "Alto Rendimiento (AR)\n"  + "Alta Capacidad (AC)\n"  + "Equilibrio entre ambas (E)");  producto = lc.next();  producto = producto.toUpperCase();  System.out.println("¿Cuál es su presupuesto total?");  presupuesto = lc.nextInt();  switch (producto) {  case "AR":  precio = presupuesto \* 0.80;  price = String.valueOf(precio);  prize = Integer.parseInt(price);  System.out.println("Serían $" + prize);  break;  case "AC":  precio = presupuesto \* 0.65;  price = String.valueOf(precio);  prize = Integer.parseInt(price);  System.out.println("Serían $" +prize);  break;  case "E":  precio = presupuesto \* 0.85;  price = String.valueOf(precio);  prize = Integer.parseInt(price);  System.out.println("Serían $" + prize);  break;  default:  System.out.println("Lo siento, no puedo ayudarle");  break;  }  } while (prize == 0);  } |
| **Algoritmo numérico** | Supongamos que queremos hallar aproximadamente el valor de π, a través de las series de Gregory-Leibniz. El código en Java sería de la siguiente manera. | public static void main(String[] args) {  Scanner lc = new Scanner(System.in);  double pi = 4, den = 1;  int i = 0, cont = 0;  System.out.println("¿Número de repeticiones?");  i = lc.nextInt();  for (int j = 0; j < i; j++){  if(cont == 0){  den = den+2;  pi = pi - (pi/(den));  cont--;  }else{  den = den+2;  pi = pi + (pi/(den));  cont++;  }  }  System.out.println(pi);  } |
| **Algoritmo de Montecarlo** | Supongamos que estamos jugando por el primer turno en Yu-Gi-Oh a través de un dado de seis caras. La única regla es que el ganador es aquel que obtiene un número mayor que el de su oponente. | public static void main(String[] args) {  System.out.println("Tirada del primer jugador: ");  int tiro1 = (int) Math.floor(Math.random() \* 6 + 1);  System.out.println("Tirada del segundo jugador: ");  int tiro2 = (int) Math.floor(Math.random() \* 6 + 1);  if (tiro1 > tiro2) {  System.out.println("El jugador uno empieza");  } else if (tiro2 > tiro1) {  System.out.println("El jugador dos empieza");  } else {  System.out.println("Vuelvan a tirar");  }  } |
| **Algoritmo de las Vegas** | Supongamos que estamos en una partida de Blackjack jugando contra la casa. Para ganar, necesitamos que el número total que obtenemos de los naipes sea 21 o menos; en caso de ser mayor, habremos perdido la partida. | public static void main(String[] args) {  Scanner lc = new Scanner(System.in);  int naipe = 0, res = 0, opc, res2 = 0, naipe2 = 0;  boolean flag = false;  System.out.println("Bienvenido a BJO (BlackJack Online)\n"  + "1. Jugar\n"  + "Presione otro numero para salir");  opc = lc.nextInt();  if (opc == 1) {  do {  if (res2 < 21) {  naipe2 = (int) Math.floor(Math.random() \* 11 + 1);  res2 = res2 + naipe2;  } else if (res2 == 21 || res2 > 21) {  }  naipe = (int) Math.floor(Math.random() \* 11 + 1);  res = res + naipe;  System.out.println("Naipe: " + naipe + " Total = " + res + "\n"  + "¿Seguir?\n"  + "1. Pedir\n"  + "2. Terminar");  opc = lc.nextInt();  if (opc == 1) {  } else {  flag = true;  }  } while (flag == false);  if (res <= 21 && res2 < res || res2 > 21) {  System.out.println("Resultado del jugador = " + res + "\n"  + "Resultado de la casa = " + res2 + "\n");  System.out.println("Has Ganado!");  } else if (res2 <= 21 && res < res2 || res > 21) {  System.out.println("Resultado del jugador = " + res + "\n"  + "Resultado de la casa = " + res2 + "\n");  System.out.println("Has Perdido");  } else if (res2 > 21 && res > 21 || res == res2) {  System.out.println("Resultado del jugador = " + res + "\n"  + "Resultado de la casa = " + res2 + "\n");  System.out.println("Empate!");  }  } else {  System.out.println("Hasta pronto");  }  } |
| **Algoritmo cotidiano** | Supongamos que deseamos ir a trabajar, estudiar o a una fiesta. La serie de pasos que podemos seguir para este propósito puede ser la siguiente. |  |
| **Algoritmo heurístico** | Supongamos que deseamos encontrar una “n” cantidad de números primos. El código en Java que podemos utilizar puede ser el siguiente. | public static void main(String[] args) {  Scanner lc = new Scanner(System.in);  List<Integer> primos = new ArrayList<>();  int cont = 2, sec, num = 2, counter = 2;  System.out.println("¿Cuál es el n número primo que desea encontrar?");  sec = lc.nextInt();  do{  if(num == 2){  primos.add(num);  num++;  cont++;  }else if(num%2 != 0 && num <= 7){  primos.add(num);  num++;  cont++;  }else if(num%2 != 0){  counter++;  if(num%counter !=0){  primos.add(num);  num++;  cont++;  counter = 2;  }else{  num++;  }  }  }while(cont < sec);  } |
| **Algoritmo de escalada.** | Supongamos que deseamos hallar π por medio de la serie de Gregory-Leibniz. El código en Java quedaría de la siguiente manera. | public static void main(String[] args) {  Scanner lc = new Scanner(System.in);  double pi = 4, den = 1;  int i = 0, cont = 0;  System.out.println("¿Número de repeticiones?");  i = lc.nextInt();  for (int j = 0; j < i; j++){  if(cont == 0){  den = den+2;  pi = pi - (pi/(den));  cont--;  }else{  den = den+2;  pi = pi + (pi/(den));  cont++;  }  }  System.out.println(pi);  } |
| **Algoritmo voraz.** | Supongamos que estamos en un país con 6 ciudades (A, B, C, D, E, F), y queremos hacer un viaje entre esas ciudades, fijando una ciudad origen y una ciudad destino. Existe una distancia en kilómetros entre cada ciudad, y se busca que al fijar un curso destino (por ejemplo de A a D) se haga un recorrido con el menor número posible de kilómetros. |  |
| **Algoritmo determinista** | Supongamos que requerimos conocer el n-ésimo término de la serie de Fibonacci. | public static void main(String[] args) {  Scanner lc = new Scanner(System.in);  int j, n, sec;  System.out.println("¿Número n-ésimo de la serie?");  sec = lc.nextInt();  int Fibo[] = new int[sec];  Fibo[0] = 0;  Fibo[1] = 1;  for (int i = 2; i < Fibo.length; i++) {  Fibo[i] = Fibo[i - 1] + Fibo[i - 2];  }  System.out.println("La secuancia Fibonacci con "+sec+" digitos es: ");  for (int i = 0; i <= Fibo.length - 1; i++) {  System.out.print("[" + Fibo[i] + "]");  }  } |
| **Algoritmo no determinista** | Supongamos que tenemos un programa que permite a un usuario jugar Black Jack. En este caso, conforme el jugador robe un naipe, la casa también robará uno, y el juego terminará cuando el jugador decida no robar más. Los resultados (que serán la suma de los valores de los naipes robados) determinaran si el jugador o la casa ganaron, perdieron o empataron. | public static void main(String[] args) {  Scanner lc = new Scanner(System.in);  int naipe = 0, res = 0, opc, res2 = 0, naipe2 = 0;  boolean flag = false;  System.out.println("Bienvenido a BJO (BlackJack Online)\n"  + "1. Jugar\n"  + "Presione otro numero para salir");  opc = lc.nextInt();  if (opc == 1) {  do {  if (res2 < 21) {  naipe2 = (int) Math.floor(Math.random() \* 11 + 1);  res2 = res2 + naipe2;  } else if (res2 == 21 || res2 > 21) {  }  naipe = (int) Math.floor(Math.random() \* 11 + 1);  res = res + naipe;  System.out.println("Naipe: " + naipe + " Total = " + res + "\n"  + "¿Seguir?\n"  + "1. Pedir\n"  + "2. Terminar");  opc = lc.nextInt();  if (opc == 1) {  } else {  flag = true;  }  } while (flag == false);  if (res <= 21 && res2 < res || res2 > 21) {  System.out.println("Resultado del jugador = " + res + "\n"  + "Resultado de la casa = " + res2 + "\n");  System.out.println("Has Ganado!");  } else if (res2 <= 21 && res < res2 || res > 21) {  System.out.println("Resultado del jugador = " + res + "\n"  + "Resultado de la casa = " + res2 + "\n");  System.out.println("Has Perdido");  } else if (res2 > 21 && res > 21 || res == res2) {  System.out.println("Resultado del jugador = " + res + "\n"  + "Resultado de la casa = " + res2 + "\n");  System.out.println("Empate!");  }  } else {  System.out.println("Hasta pronto");  }  } |
| **Algoritmo de vuelta atrás** | Supongamos que tenemos una lista de 100 posibles caminos de un centro de conferencias a nuestra casa. Necesitamos una manera por la cual sepamos que, de esa lista, existe un camino tal que dicho camino sea el que posea el menor recorrido posible (en kilómetros). | public static void main(String[] args) {  Scanner lc = new Scanner(System.in);  int camino = 0;  List<Integer> caminos = new ArrayList<>();  for (int i = 0; i < 100; i++) {  camino = (int) Math.floor(Math.random() \* 100 + 1);  caminos.add(camino);  }  for (Integer camino1 : caminos) {  System.out.println(camino1);  }  camino = 100;  for (Integer camino1 : caminos) {  if(camino1 < camino){  camino = camino1;  }  }  System.out.println("El camino más rapido a tu casa es: "+camino);  } |
| **Programación Dinámica.** | Supongamos que necesitamos buscar un elemento dentro de un arreglo de números aleatorios de la manera mas óptima posible. | public static void main(String[] args) {  Scanner lc = new Scanner(System.in);  int[] ar = new int[30];  for(int j = 0; j< ar.length; j++){  ar[j] = (int) Math.floor(Math.random() \* 30 + 1);  }  System.out.println("¿Qué numero desea buscar? (Del 1 al 30)");  int n = lc.nextInt();  for(int j = 0; j < ar.length; j++){  if(n != ar[j] && j == ar.length){  System.out.println("No se encontró el número");  }  }  int posnE = encontrarN(ar, n, 0);  System.out.println("Se encuentra en la posicion "+posnE);  }    public static int encontrarN(int[] ar, int n, int i) {  if(i == ar.length || ar[i] == n){  if(i == ar.length){  return -1;  }else{  return i;  }  }else{  return encontrarN(ar, n, i+1);  }  } |