ALGORITMOS DE RESOLUCIÓN DE PROBLEMAS

AUTOR: WALTER MADRIGAL CHAVES

NOVIEMBRE: 2020

San Marcos

Contenido

INTRODUCCIÓN 2
ALGORITMO 3
ESTRUCTURAS BÁSICAS DE CONTROL 4
Secuenciación 4
Bifurcación 5
Iteración 5
ALGORITMOS BÁSICOS DE BÚSQUEDA 6
Búsqueda secuencial 6
Búsqueda binaria 8
ALGORITMOS DE ORDENAMIENTO 10
Burbuja 10
Selección 11
Inserción 12
Quicksort 13
FUNCIONES RECURSIVAS 14
Factorial 14
Fibonacci 15
Multiplicación de enteros 15
Potencia de dos números 16
CONCLUSIONES Y RECOMENDACIONES 17
REFERENCIAS BIBLIOGRÁFICAS 18

INTRODUCCIÓN

Día a día se pone en práctica, a veces inconscientemente, el concepto de algoritmo. Realizar distintas tareas de nuestras vidas como visitar el supermercado, sacar dinero del cajero y preparar la cena son ejemplos en donde se planifican una serie de pasos lógicos y se sigue un determinado orden de ejecución.

En el mundo del desarrollo de software los algoritmos son la base de los proyectos, es gracias a ellos que se implementan grandes rutinas de código con una significativa disminución de errores, lo que con lleva a crear procesos más eficientes.

Existe una serie de algoritmos especializados en resolver problemas puntuales, como búsqueda de elementos, ordenamiento de datos, problemas matemáticos recursivos, entre otros. El propósito de esta lectura es detallar esas rutinas especializadas con el fin de aprovechar la virtud que este tipo de herramientas ofrecen a los desarrolladores.

ALGORITMO

Para Herrera (2017), Un algoritmo es un conjunto de acciones o pasos finitos, ordenados de forma lógica y que se utilizan para resolver un problema o para obtener un resultado.

El objetivo principal del algoritmo es dar un detalle de los pasos a seguir para resolver un problema planteado, algunas veces de forma general y otras tipificado a la dificultad que pretende resolver. Debe cumplir con una serie de requisitos para que sea tomado en cuenta como tal:

- **Preciso:** es importante que marque el orden preciso de ejecución de pasos.
- **Definido:** bajo las mismas condiciones siempre debe generar el mismo resultado.
- **Finito:** debe tener un final.
- **Legible:** los pasos deben ser claros y concisos, se debe evitar ambigüedades.
- **Marcado:** su estructura debe estar bien definida, en donde exista una entrada que alimenta al algoritmo, un proceso de información y una salida de resultados.

Los algoritmos suelen ser expresados de varias maneras, algunas de ellas son: mediante lenguaje natural, diagramas de flujo, pseudocódigo y lenguajes de programación. Cada forma de expresar tiene características diferentes haciéndolas poco o más atractivas para los programadores.

Por ejemplo, en el lenguaje natural la descripción de los pasos del algoritmo tiende a ser ambiguo y extenso. Los diagramas de flujo evitan muchas ambigüedades del lenguaje natural pero su estructura dista del código fuente

final. Al usar pseudocódigo las expresiones son formas más estructuradas para representar algoritmos; no obstante, se mantienen independientes de un lenguaje de programación específico.

La descripción de un algoritmo se establece en tres niveles:

- **Descripción de alto nivel:** Es un proceso general, en donde se establece el problema, si es necesario se selecciona un modelo matemático y por último se explica el algoritmo mediante ilustraciones y omitiendo detalles.
- **Descripción formal:** Se utiliza el pseudocódigo para narrar la sucesión de pasos que encuentran la solución al problema.
- **Implementación:** Es el algoritmo expresado en un lenguaje de programación específico, contemplando su sintaxis y reglas gramaticales.

El pseudocódigo es una forma de expresar los distintos pasos que va a realizar un programa, de la forma más parecida a un lenguaje de programación. Su principal función es la de representar por pasos la solución a un problema o algoritmo, de la forma más detallada posible, utilizando un lenguaje cercano al de programación. (Barnes & Kölling, 2017)

ESTRUCTURAS BÁSICAS DE CONTROL

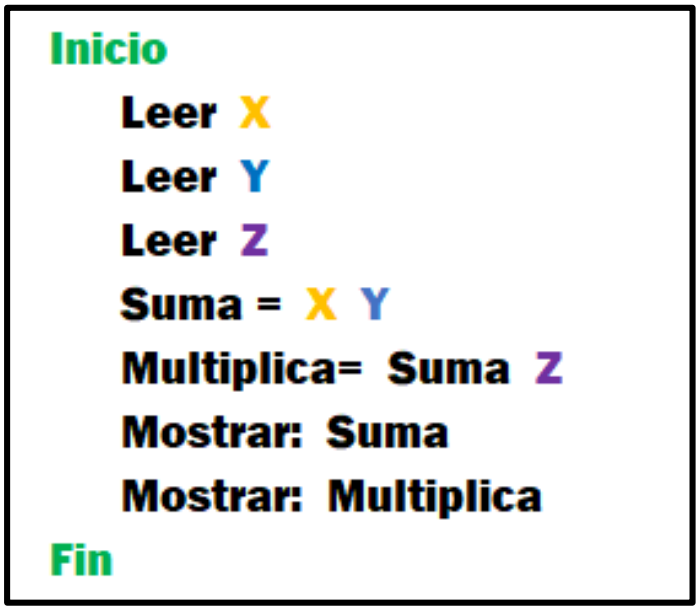
Cuando se emplea el término de estructuras básicas de control en algoritmos, se refiere a elementos que permiten modificar el flujo de ejecución de instrucciones o pasos de un programa. Se utilizan para resolver problemas más complejos.

Secuenciación

La secuenciación son las estructuras más simples de las estructuras básicas de control, consta de un conjunto de acciones que se ejecutan en forma secuencial, o sea una instrucción sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente

hasta el fin del proceso.

Imagen 1 Estructura secuencial



Fuente: Elaboración propia

En la imagen anterior se muestra un conjunto de instrucciones que deben ejecutarse secuencialmente para que funcione, por ejemplo, sino se realiza la carga de las variables iniciales la suma no se podría realizar y por consiguiente la multiplicación tampoco.

Bifurcación

Las bifurcaciones agregan un poco complejidad al algoritmo, permiten condicionar la ejecución de una o varias acciones. Su principal elemento es una acción llamada condicionante que es el intermediario que define el conjunto de código que se ejecutará. If y Switch son dos representantes de este grupo.

Iteración

Son estructuras de control repetitivas, utiliza el concepto de ciclos para ejecutar líneas de código las veces que sea necesario para cumplir con un criterio

planteado. Las sentencias While, do/While y for son representantes de este tipo estructura.

ALGORITMOS BÁSICOS DE BÚSQUEDA

Existen un conjunto de algoritmos con técnicas especializadas que ofrecen funciones específicas, como por ejemplo están los algoritmos de búsqueda y de ordenamiento. Un algoritmo de búsqueda toma un argumento y trata de encontrarlo en un conjunto de datos, tablas o registros. El algoritmo puede devolver el elemento completo o un apuntador a dicho registro.

Su producción es más funcional cuando se maneja grandes cantidades de información. Hay dos tipos los secuenciales y los binarios.

Búsqueda secuencial

Es conocida también como búsqueda lineal, su lógica consiste en buscar un elemento tomando como punto de partida la primera línea, si el elemento no se encuentra se sigue al siguiente ítem y así sucesivamente hasta llegar al final de la lista. Una característica importante para que pueda dar este tipo de búsqueda es que los datos deben estar ordenados secuencialmente.

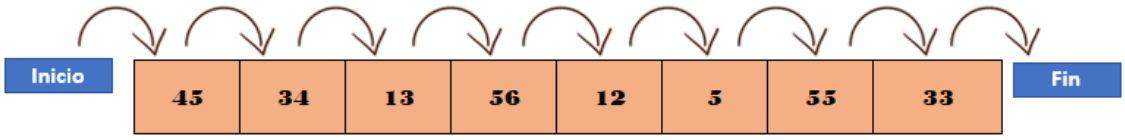
Ventajas

- Es un método simple, recomendado para conjunto de datos no muy grandes.
- Se puede utilizar aun cuando los datos no estén ordenados.

Desventajas

- Debe recorrer por completo todos los datos, por lo que lo hace poco eficiente.
- Es lento

Imagen 2 Representación gráfica de búsqueda secuencial



Fuente: Elaboración propia

En la imagen 2 se muestra la representación gráfica de la búsqueda secuencial en un vector, como se muestra el algoritmo verifica el primer elemento y seguidamente pasa al segundo en lista, sigue esta tendencia hasta llegar al último de los elementos. En la imagen 3 se detalla la implementación en Java.

El ciclo For es el encargado de visitar cada elemento del arreglo en busca del número solicitado.

Imagen 3 Implementación en Java de búsqueda secuencial

```java
import java.util.Scanner;

public class main {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        int [] numeros = {45,34,13,55,12,5,55,33};
        Scanner leer = new Scanner(System.in);
        boolean existe= false;

        System.out.println("ingrese el número a buscar!");
        int numBuscado = leer.nextInt();

        for(int b = 0; b < numeros.length; b++){

            if(numeros[b]==numBuscado){

                System.out.println("el numero si existe, en la posicion "+(b+1));
                break;
            }
            if(b == numeros.length-1){
                existe = true;
            }
        }
        if(existe==true){

            System.out.println("el numero no existe");
        }
    }
}
```

Fuente: Elaboración propia

Búsqueda binaria

La búsqueda binaria es el método más complejo, pero más eficiente que la secuencial. Su lógica consiste en comparar el elemento central del arreglo con el valor buscado, si en esta primera interacción ambos coinciden finaliza la búsqueda, si no, se define si el valor buscado es mayor o menor al valor del centro del arreglo, posteriormente se realiza un sub-arreglo con la mitad que todavía está en "juego", con esto se disminuye el rango de búsqueda a la mitad.

Esta lógica se realiza hasta encontrar el elemento o que el arreglo sea tan pequeño que no se pueda partir en dos.

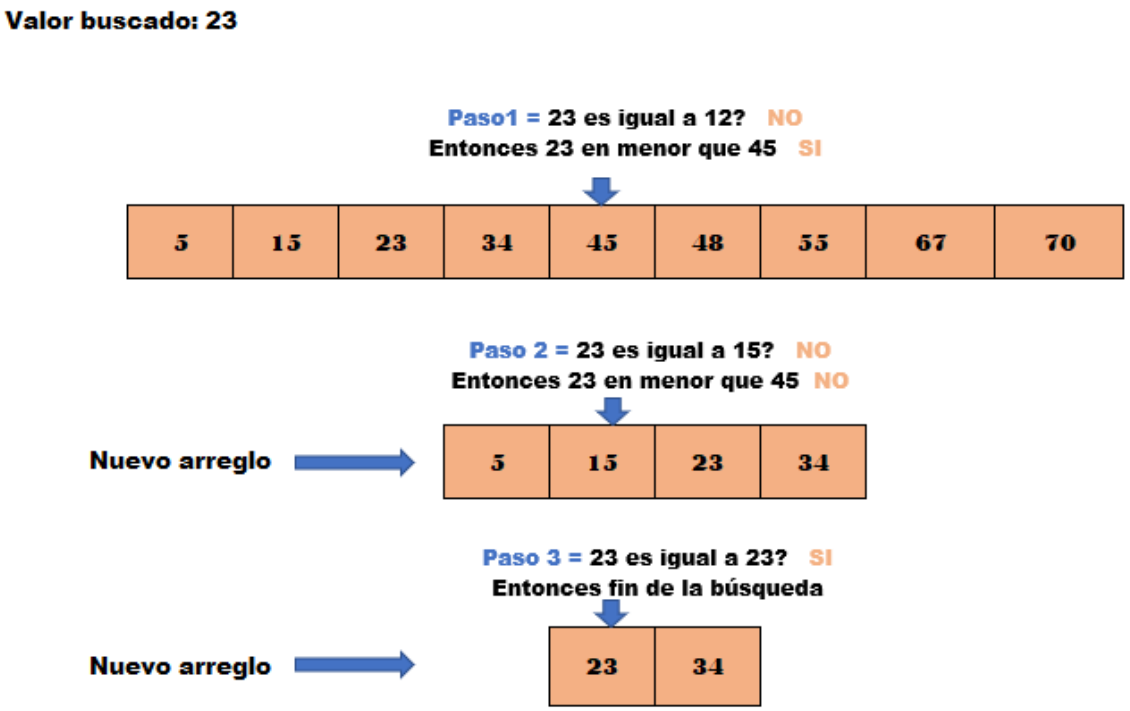
Ventajas

- Es muy eficiente para realizar búsquedas en arreglos ordenados.
- Se puede aplicar tanto a datos en listas lineales, como en árboles binarios.

Desventajas

- El arreglo debe estar ordenado.
- Debe conocerse el número de elementos.

Imagen 4 Representación gráfica de la búsqueda binaria



Fuente: Elaboración propia

En la imagen 4 se muestra la representación gráfica de la búsqueda binaria, la primera interacción comparara 23 con el dato de la mitad del arreglo, en este caso 45, al no coincidir, se realiza la pregunta si el dato es mayor o menor, se determina que es menor por consiguiente se descarta de la búsqueda todos los datos mayores a 45, posteriormente se crea un sub-arreglo con los datos no descartados.

En la imagen 5 se aplica un ejemplo en java, en este caso el ciclo do/while se encarga de comparar el dato central con el elemento en búsqueda.

Imagen 5 Implementación en Java de búsqueda binaria

```java
public class BusquedaBinaria {

    public static int busquedaBin(double[] matriz, double valorBuscado) {
        if (matriz.length == 0) {
            return -1;
        }
        int mitad, inferior = 0;
        int superior = matriz.length - 1;
        do {
            mitad = (inferior + superior) / 2;
            if (valorBuscado > matriz[mitad]) {
                inferior = mitad + 1;
            } else {
                superior = mitad
          - 1;
        }
        } while (matriz[mitad] != valorBuscado && inferior <= superior);
        if (matriz[mitad] == valorBuscado) {
            return mitad;
        } else {
            return -1;
        }

    }

}

```

Fuente: Elaboración propia

ALGORITMOS DE ORDENAMIENTO

Como su nombre lo indica son algoritmos que utilizan técnicas para ordenar la información dentro de listas, existen varios tipos con diferentes características rendimiento y complejidad, algunos de ellos son: burbuja, selección, inserción, mezclas, conteo, Quicksort, entre otros.

Burbuja

Este es el más sencillo, popular y fácil de implementar, aunque su eficiencia es pobre, actúa comparando e intercambiando los elementos adyacentes que no están en orden, hasta que toda la lista de elementos esté en secuencia. Si un elemento es mayor que el que está en la siguiente posición se intercambian.

Imagen 6 Algoritmo de ordenamiento burbuja

```
static void burbuja_lims(int T[], int inicial, int final)
{
  int i, j;
  int aux;
  for (i = inicial; i  i; j--)
      if (T[j] < T[j-1])
  {
     aux = T[j];
     T[j] = T[j-1];
      T[j-1] = aux;
 }
}
```

Fuente: https://elbauldelprogramador.com/algoritmos-de-ordenacion/

Selección

Este algoritmo es fácil de codificar, pero no es muy eficiente porque tiene altos tiempos de respuesta, se puede utilizar en arreglos pequeños. Consiste en hallar el menor de todos los elementos del arreglo e intercambiarlo con el que está en la primera posición, posteriormente el segundo más pequeño, y así sucesivamente hasta ordenarlo todo.

Imagen 7 Algoritmo de ordenamiento selección

```
static void seleccion_lims(int T[], int inicial, int final)
{
  int i, j, indice_menor;
  int menor, aux;
  for (i = inicial; i < final - 1; i++) {
    indice_menor = i;
    menor = T[i];
    for (j = i; j < final; j++)
      if (T[j] < menor) {
 indice_menor = j;
   menor = T[j];
      }
    aux = T[i];
    T[i] = T[indice_menor];
    T[indice_menor] = aux;
  };
}
```

Fuente: https://elbauldelprogramador.com/algoritmos-de-ordenacion/

Inserción

Se considera la forma más lógica para ordenar datos, su lógica consiste en ir insertando un elemento de la lista en la parte ordenada de la misma, asumiendo que el primer elemento es la parte ordenada, el algoritmo ira comparando un elemento de la parte desordenada de la lista con los elementos de la parte ordenada, insertando el elemento en la posición correcta dentro de la parte ordenada, y así sucesivamente hasta obtener la lista ordenada.

Imagen 8 Algoritmo de ordenamiento inserción

```
static void insercion_lims(int T[], int inicial, int final)
{
  int i, j;
  int aux;
  for (i = inicial + 1; i < final; i++) {
    j = i;
    while ((T[j]  0)) {
      aux = T[j];
      T[j] = T[j-1];
      T[j-1] = aux;
      j--;
    };
  };
}
```

Fuente: https://elbauldelprogramador.com/algoritmos-de-ordenacion/

Quicksort

Considerado por muchos el mejor, por su eficiencia y rapidez, consiste en crear la figura del pivote para dividir el arreglo en dos, de un lado los valores menores al pivote y del otro lado los valores mayores al pivote, con esto se obtiene dos sub-arreglos más sencillos de ordenar.

Imagen 9 Algoritmo de ordenamiento Quicksort

```
static void quicksort_lims(int T[], int inicial, int final)
{
  int k;
  if (final - inicial < UMBRAL_QS) {
    insercion_lims(T, inicial, final);
  } else {
    dividir_qs(T, inicial, final, k);
    quicksort_lims(T, inicial, k);
    quicksort_lims(T, k + 1, final);
  };
}
```

Fuente: https://elbauldelprogramador.com/algoritmos-de-ordenacion/

FUNCIONES RECURSIVAS

Las funciones recursivas son aquellas que se invocan a sí mismas en algún momento en su ejecución. Utiliza una variable para acumular los productos y obtener la solución. En la solución recursiva se realizan llamadas al propio método con valores de n cada vez más pequeños para resolver el problema.

Factorial

La factorial de un número es producto de todos los números enteros y positivos, comenzando en 1 y terminando en el elemento evaluado. Por ejemplo, la factorial de 4 es 24, se obtiene de la siguiente manera: 1 x 2 x 3 x 4= 24

Imagen 10 Función recursiva Factorial

```
// Método Java recursivo para calcular el factorial de un número
public double factorial(int n){
    if (n==0)
        return 1;
    else
        return n*(factorial(n-1));
}
```

Fuente: http://puntocomnoesunlenguaje.blogspot.com/2012/04/recursividad-en-java.html

Fibonacci

Fibonacci es un problema matemático que se puede solucionar fácilmente utilizando funciones recursivas, consiste en una sucesión de números que comienzan con 0 y 1, posteriormente se suman estos números para calcular el tercero en lista y así sucesivamente. Cada término siguiente es la suma de los dos anteriores.

Ejemplo:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597 \ldots$$

Imagen 11 Función recursiva Fibonacci

```
1  static int fibonacci (int n)
2  {
3  if ((n == 0) || (n == 1))
4          return 1;
5          else
6          return fibonacci(n-1) + fibonacci(n-2);
7  }
8
```

Fuente: http://puntocomnoesunlenguaje.blogspot.com/2012/04/recursividad-en-java.html

Multiplicación de enteros

Se aplica recursividad para multiplicar dos números enteros sin usar el operador *.

Imagen 12 Función recursiva multiplicar dos números enteros

```
1   public int multiplica (int m, int n) {
2           if (n > 0) { return 0;
3           } else  { return m * multiplica (m, n-1); }
4       } //Cierre del método
```

Fuente: http://puntocomnoesunlenguaje.blogspot.com/2012/04/recursividad-en-java.html

Potencia de dos números

Es la función recursiva para calcular la potencia de números

Imagen 13 Función recursiva potencias a números enteros

```
1   public int potenciaConRecursion (int m, int n) {
2           if (n==0) { return 1;
3           } else  { return m * potenciaConRecursion (m, n-1); }
4       } //Cierre del método
```

Fuente: http://puntocomnoesunlenguaje.blogspot.com/2012/04/recursividad-en-java.html

CONCLUSIONES Y RECOMENDACIONES

- Los algoritmos son un elemento fundamental dentro del desarrollo del software, deben de cumplir con una serie de requisitos como: ser precisos, tener un inicio y un final, es importante que sea legible, todos sus pasos deben ser claros y por último debe tener bien definida su estructura.

- Las estructuras básicas de control son formas de interactuar con el código, su objetivo es romper el flujo normal de su recorrido, existen los de tipo secuenciación, bifurcación e iteración.

- Es muy recomendado aprovechar los algoritmos especializados en alguna rutina como los de ordenamiento y búsqueda. Por lo general estos códigos son más eficientes, rápidos y robustos que si se implantaran “a pie”.

- El ordenamiento de datos se puede realizar con varios algoritmos disponibles en el mercado, siempre es importante antes de aplicar alguno, ver las ventajas que ofrece, sus desventajas, nivel de complejidad y consumo de recursos. Es muy recomendable el llamado Quicksort, pero no quiere decir que es el mejor para su proyecto específico.

REFERENCIAS BIBLIOGRÁFICAS

Alcalde, A. (24 de 11 de 2020). *El baúl del programador*. Obtenido de https://elbauldelprogramador.com/algoritmos-de-ordenacion/

Barnes, D., & Kölling, M. (2017). *Programación oerientada a objetos con Java usando Bluej 6a. Ed.* Madrid: Pearson.

Blasco, F. (2020). *Programación orientada a objetos en Java.* Bogotá: Ediciones la U.

García Hernández, E. (11 de 30 de 2020). *Programación Java*. Obtenido de http://puntocomnoesunlenguaje.blogspot.com/2012/04/recursividad-en-java.html

Herrera Morales, J., Gutiérrez Posada, J., & Pulgarín Giraldo, R. (2017). *Introducción a la lógica de programación.* Armenia: ElIZCOM S.A.A.

San Marcos

www.usanmarcos.ac.cr

San José, Costa Rica