CIS

OOP, Introducción a los Algoritmos

ESTRUCTURAS DE DATOS
Juan Gutiérrez-Cárdenas
2023-2

UNIVERSIDAD DE LIMA

Normativas

- Las clases serán de forma presencial, se tomará asistencia en una o dos oportunidades en cada sesión.
- En la semana 1 en BB se puede encontrar el sílabo, el plan de trabajo y la forma de evaluación; así como un ejemplo de asignación tomado anteriormente.
- En caso de dudas, académicas o administrativas, pueden consultar al profesor del curso; o en su defecto al delegado del aula. En caso de no encontrar respuesta puede dirigirse al coordinador del curso o al coordinador del área.
- Tratar que siempre exista un ambiente cordial y de respeto entre todos.

UNIVERSIDAD DE LIMA

Normativas

- Acerca del uso de herramientas de LLM, pueden ser utilizadas, pero no en las evaluaciones.
- Revisar las soluciones dadas por estos modelos, así como las explicaciones. Se recomienda revisar con otras fuentes y tener cuidado con los llamados “illusions”.
- El curso tiene un política de cero plagio o copia. En caso que suceda la nota será anulada y el caso elevado a Comité Académico y a Comité de Ética para las sanciones respectivas.
- Cada tema tiene diversos ejercicios, cada alumno lo puede avanzar a tu propio ritmo.

UNIVERSIDAD DE LIMA

Algoritmos

Definición:

Un algoritmo se puede definir como una secuencia de pasos computacionalmente correctos, estos toman una entrada (input) los cuales al ser procesados devuelven una salida (output) [1].

Ejemplo:

Problema: Ordenar una sequencia de números de forma creciente:

Input: Secuencia de n números $\langle a_1, a_2, \ldots, a_n \rangle$

Output: Una permutación $\langle \acute{a}_1, \acute{a}_2, \ldots, \acute{a}_n \rangle$ de tal manera que: $\acute{a}_1 \leq \acute{a}_2 \leq \cdots \leq \acute{a}_n$

UNIVERSIDAD DE LIMA

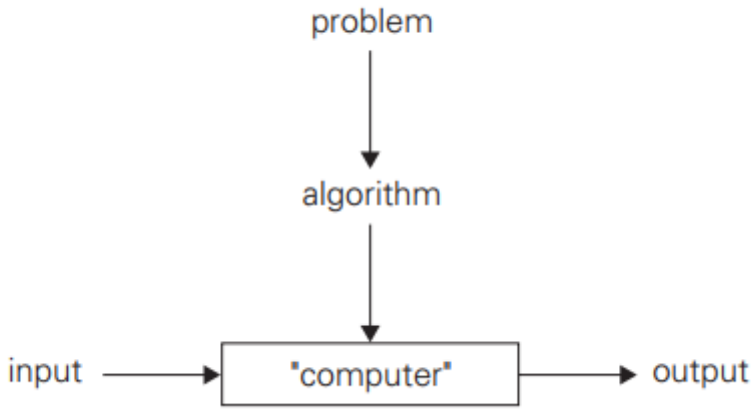
Algoritmos

```
Algoritmo Burbuja
{v es un vector de n elementos}
i←1
intercambia←V
mientras (intercambia)
        intercambia←F
        i←i+1
        repetir con pos desde 0 hasta n-i
                si v[pos]>v[pos+1] entonces
                        swap(v[pos],v[pos+1])
                        intercambia←V
                fin de si
        fin de repetir
fin de mientras
```

UNIVERSIDAD DE LIMA

Algoritmos

Un algoritmo también puede ser una secuencia de pasos no ambiguos, se puede dar el siguiente proceso [2]:

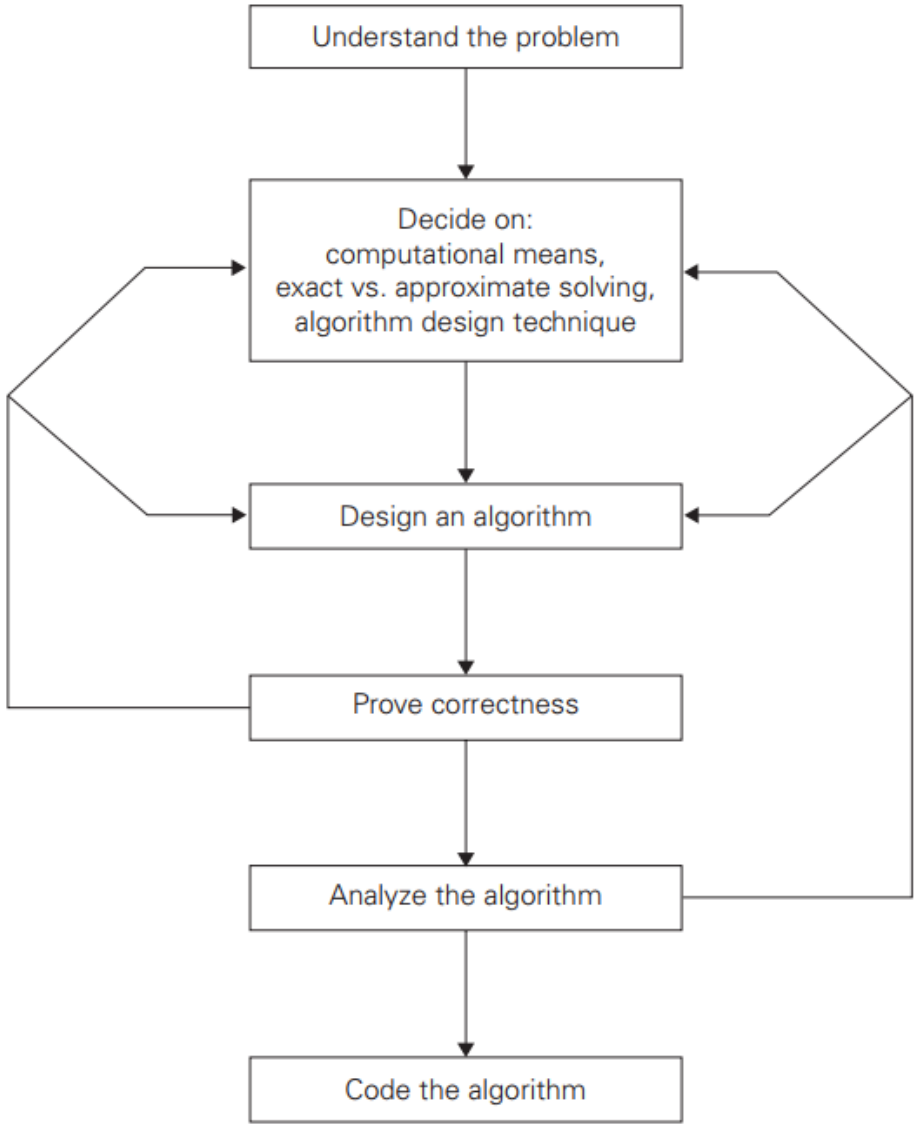


Ejemplo: Se desea hallar el MCD de un par de números.

UNIVERSIDAD DE LIMA

Algoritmos

Revisar la siguiente gráfica [2]:



Algoritmos

- Prove correctness

Consiste en probar que un algoritmo devuelme un resultado correcto en un tiempo dado, considerando todas las entradas posibles.

Se puede usar inducción o revisar si el error devuelto por un algoritmo no excede un límite.

- Analizar el algoritmo

Se da en términos de tiempo y espacio.

Formas de Representación

Ejemplo: Hallar el gcd de dos números.

Solución: Utilizaremos el algoritmo de Euclides:

Aplicar la siguiente formula hasta que el resultado de la operación de modulo sea 0:

$$\gcd(m,n) = \gcd(n, m \bmod n)$$

Algoritmo de Euclides para calcular gcd(m,n)

Paso 1: Si n=0 retornar m como el valor de salida y parar, sino ir al paso 2.

Paso 2: Dividir m entre n y asignar el valor del residuo a r.

Paso 3: Asignar el valor de n a m y el valor de r a n. Regresar al paso 1.

Formas de representación

(no relevante si es en español o en inglés)

Algoritmo Euclides (m,n)

//Calcula el gcd(m,n)

//Entrada: Dos números enteros no negativos m, n

//Salida: El máximo común divisor de m, n

while n≠0

    r← $m \; mod \; n$

    m ←n

    n ←r

end while

return m

UNIVERSIDAD DE LIMA

OOP: Recuerdo

Primero conviene definir que es un ADT (Abstract Data Type). Ejemplo: Operaciones aritméticas con números, se tiene algo como:

int a=0;

a++;

En este caso el usuario no tiene que preocuparse de cómo se almacena el dato en el computador (bytes en el rango de $-2^{-31}\ a\ 2^{32}-1$)

Se le llama built-in data type.

UNIVERSIDAD DE LIMA

OOP: Recuerdo

ADT: Tiene las siguientes características [5]:

a) Es un tipo de dato externo que almacena algún dato (puede ser una clase)
b) Tiene operaciones que se pueden .
c) No es necesario tener detalles sobre su implementación interna.

Existen desde la programación imperativa (anterior a la OOP), pero en la OOP surgió el concepto de herencia el cual amplió este concepto de ADT.

OOP: Recuerdo

Clase: Es un generador de objetos, tiene un nombre, atributos y operaciones.

Objeto: Son las entidades generadas por las clases.

Las clases pueden relacionarse entre ellas de diversas formas (asociación o herencia).

Forma de manipulación: (supongamos que tenemos una clase cuadrado).

```
objCuad1=new Cuadrado(4)       //cualquier sintaxis
print(objCuad1.calcularArea())
```

Siempre es de la forma objeto.operación

Ejercicio: Implementarlo en Java, puede usar NetBeans o también un compilador en línea:

https://www.programiz.com/java-programming/online-compiler/

UNIVERSIDAD DE LIMA

OOP: Recuerdo

```
import java.lang.Math;


class Cuadrado{
    private double a;
    public Cuadrado(double a){
        this.a=a;
    }
    public double area(){
        return Math.pow(a,2);
    }
}


class HelloWorld {
    public static void main(String[] args) {
        Cuadrado objC=new Cuadrado(4);
        System.out.println("area es "+objC.area());
    }
}
```

UNIVERSIDAD DE LIMA

Estructuras de Datos

Inducción: [3]

- Los computadores trabajan con bits, pero el usuario prefiere su manipulación con otros tipos de datos, por ejemplo, enteros.
- Algunos tipos de datos vienen con los lenguajes de programación, pero un usuario puede definir nuevos tipos de datos.
- Estos nuevos tipos de datos requieren una estructura. Ejemplo: matrices o grafos.
- El campo de las estructuras de datos estudia este tipo de nuevas estructuras analizando sus requerimientos en término de espacio y tiempo.

Estructuras de Datos

- Se revisarán en el curso algunas estructuras de datos clásicas así como sus operaciones con éstas.
- Mayormente nos centraermos en el tipo de estructuras de datos no primitivas, dentro de estructuras primitivas tenemos, por ejemplo los enteros.
- Las estructuras de datos no primitivas se dividen en lineales: arreglos, listas, pilas, colas y no lineales como grafos y árboles.
- Se utilizará un lenguaje de programación para su implementación.
- Se prescindirá del uso de librerías, se programará desde las bases.

Otros temas a revisar de programación

- Revisión de principios fundamentales de flujos de datos: secuencia, selección, repetición.
- Revisión de entrada y salida de datos.
- Revisión de funciones.
- Revisión rápida del paradigma orientado a objetos (no es imprescindible, pero ayudará en la implementación de diversas soluciones).
- Se deberá tener aptitud para revisar y modificar fragmentos de código ya realizados. Tratar de documentar los suyos propios.
- Libro de referencia [4] (justificación del uso de Java).

Referencias

(sin un formato en particular)

[1] Cormen et al., Introduction to Algorithms, 3rd. Edition.

[2] Levitin, Introduction to the Design and Analysis of Algorithms, 3rd. Edition.

[3] Drozdek, A. (2010). Data Structures and Algorithms in Java

[4] Liang, Y. (2017). Introduction to Java Programming and Data Structures (11th ed.), New York, Pearson.

[5] Boone, Barry et al. (1998), Abstract Data Types in Java, McGraw-Hill.

UNIVERSIDAD DE LIMA