



Conceptos de Algoritmos Datos y Programas



CADP – Temas de la clase de hoy



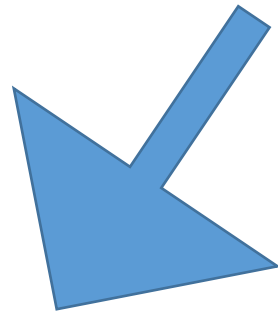
● Corrección

● Eficiencia

CADP – CORRECCION y EFICIENCIA



Cuando se desarrollan los algoritmos hay dos conceptos importantes que se deben tener en cuenta:



CORRECCION



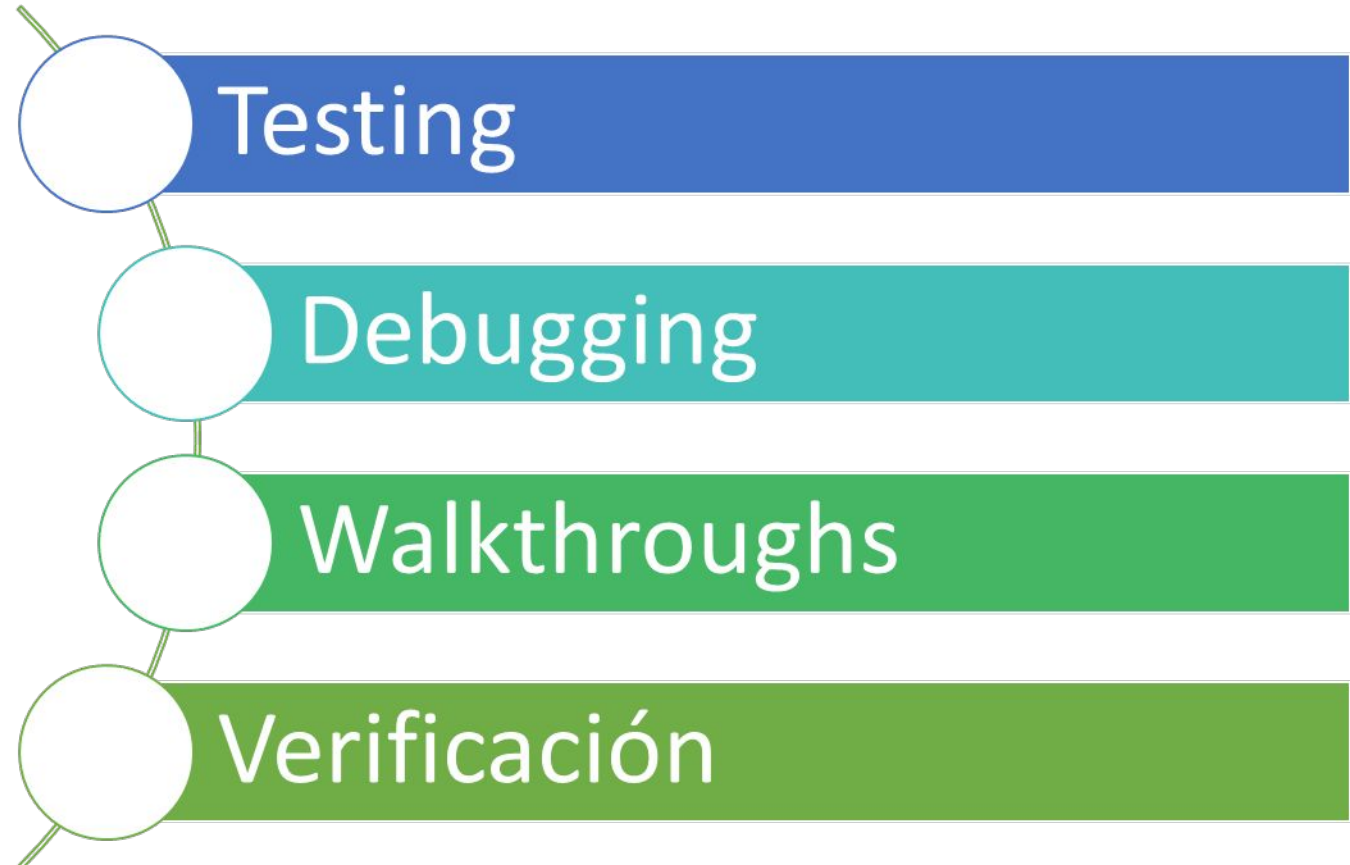
EFICIENCIA



CORRECCION

Un programa es correcto si se realiza de acuerdo a sus especificaciones.

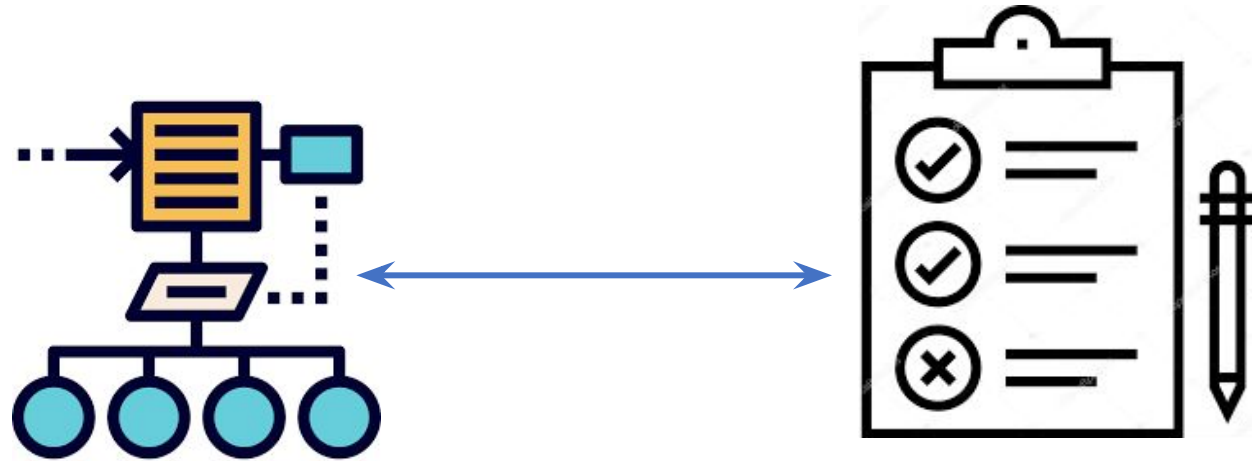
**Técnicas para
corrección de
programas**





TESTING

El propósito del Testing es proveer evidencias convincentes que el programa hace el trabajo esperado.



**Diseñar un plan
de pruebas**

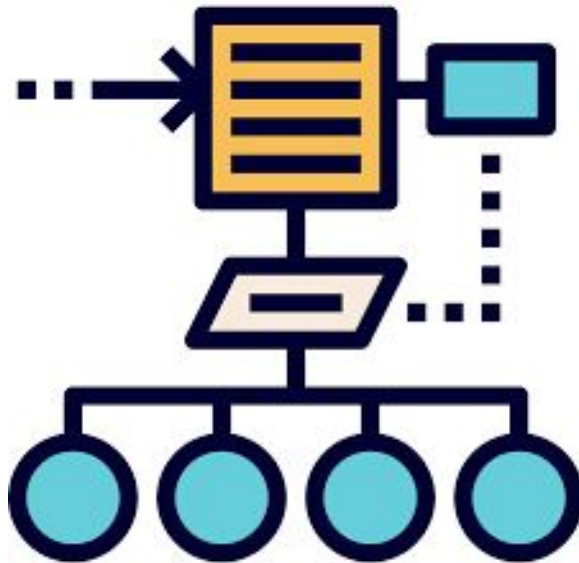


PLAN de PRUEBAS

- Decidir cuales aspectos del programa deben ser testeados y encontrar datos de prueba para cada uno de esos aspectos.
- Determinar el resultado que se espera que el programa produzca para cada caso de prueba.
- Poner atención en los casos límite.
- Diseñar casos de prueba sobre la base de lo que hace el programa y no de lo que se escribió del programa. Lo mejor es hacerlo antes de escribir el programa.



PLAN de PRUEBAS: una vez que el programa ha sido implementado y se tiene el plan de pruebas:



Se analizan los casos de prueba.

Si hay errores se corrigen.

Estos dos pasos se repiten hasta que no haya errores.



DEBUGGING

Es el proceso de descubrir y reparar la causa del error.



El diseño y aplicación de pruebas adicionales para ubicar y conocer la naturaleza del error.

Es agregar sentencias adicionales en el programa para poder monitorear su comportamiento más cercanamente.



Los **errores** pueden provenir de dos fuentes:

El diseño del programa
no es el adecuado.



El programa no está
escrito correctamente.



Errores Sintácticos

- Se detectan en la compilación

Errores Lógicos

- Generalmente se detectan en la ejecución

Errores de Sistema

- Son muy raros los casos en los que ocurren



WALKTHROUGH

Es recorrer un programa frente a una audiencia.

La lectura de un programa a alguna otra persona provee un buen medio para detectar errores.



Esta persona no comparte preconceptos y está predispuesta a descubrir errores u omisiones.

A menudo, cuando no se puede detectar un error, el programador trata de probar que no existe, pero mientras lo hace, puede detectar el error, o bien puede que el otro lo encuentre.



VERIFICACION

Verificar un programa significa controlar que se cumplan las pre y post condiciones del mismo.



**PARA DETERMINAR LA
CORRECCION DE UN PROGRAMA
PUEDO UTILIZAR UNO, DOS, TRES O
LAS CUATRO TECNICAS DE
CORRECCION**





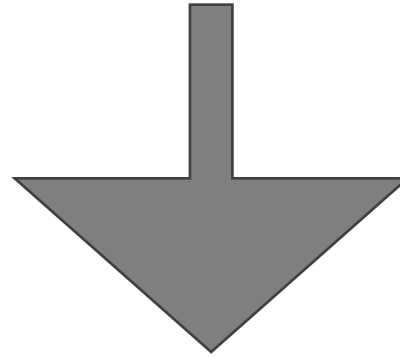
Cuál/es técnicas creen que se utiliza en la teoría de la materia?



Durante las consultas de práctica se utilizan las mismas?

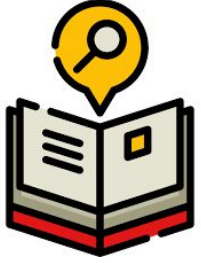


Una vez que se obtiene un algoritmo y se decide que es correcto, es importante determinar la **eficiencia** del mismo.



EFICIENCIA

El análisis de la eficiencia de un algoritmo estudia el **tiempo** que tarda un algoritmo en ejecutarse y la **memoria** que requiere.



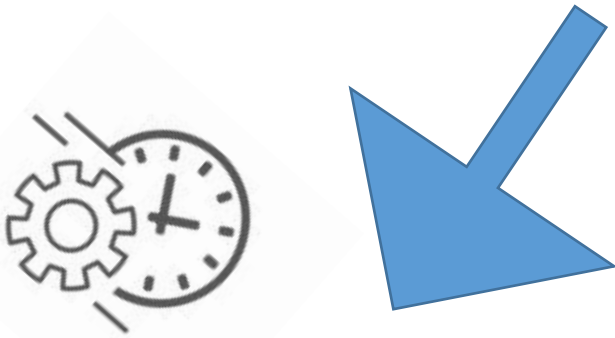
EFICIENCIA

Se relaciona con:





EFICIENCIA



TIEMPO



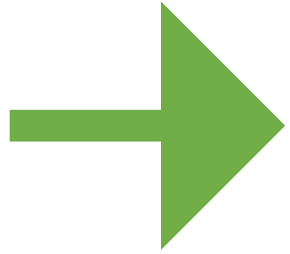
MEMORIA

Cómo se
miden?

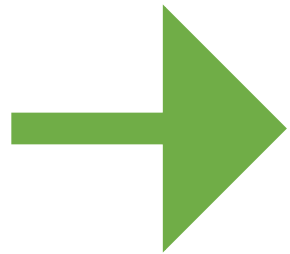


TIEMPO DE EJECUCION

El tiempo de un algoritmo puede definirse como una función de entrada :



Existen algoritmos que el tiempo de ejecución tiempo de ejecución no depende de las características de los datos de entrada sino de la cantidad de datos de entrada o su tamaño

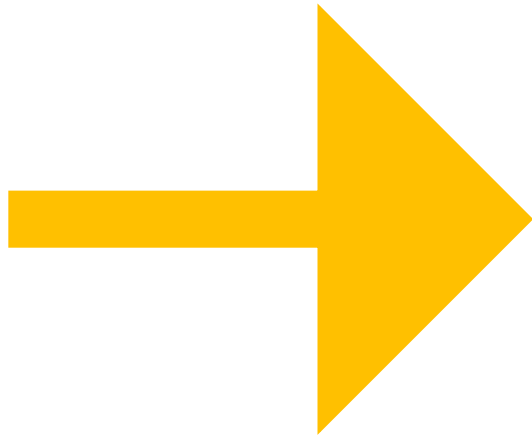


Existen otros algoritmos el tiempo de ejecución es una función de la entrada “específica”, en estos casos se habla del tiempo de ejecución del “peor” caso. En estos casos, se obtiene una cota superior del tiempo de ejecución para cualquier entrada

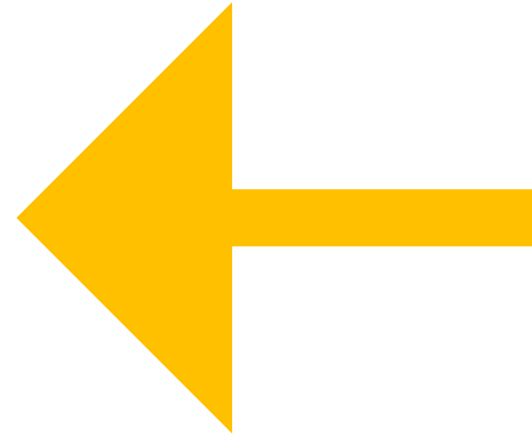


TIEMPO DE EJECUCION

Puede calcularse haciendo dos tipos de análisis:



ANALISIS
EMPIRICO



ANALISIS
TEORICO



ANALISIS EMPIRICO

Para realizar un análisis empírico, es necesario realizar el programa y medir el tiempo consumido.



Fácil de realizar.



Obtiene valores exactos para una máquina y unos datos determinados

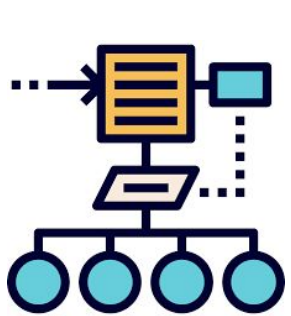
Completamente dependiente de la máquina donde se ejecuta

Requiere implementar el algoritmo y ejecutarlo repetidas veces.



ANALISIS TEORICO

Implica encontrar una cota máxima para expresar el tiempo de nuestro algoritmo, sin necesidad de ejecutarlo.



$O(\log)$

$O(n)$

$O(n^2)$

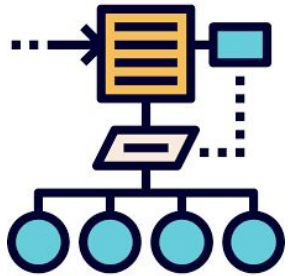
Se obtiene el tiempo teórico del algoritmo y a partir de allí obtiene el orden del mismo

Cómo obtengo el tiempo teórico?



ANALISIS TEORICO

Dado un algoritmo que es correcto se calcula el tiempo de ejecución de cada una de sus instrucciones. Para eso se va a considerar:



Sólo las instrucciones elementales del algoritmo: asignación, y operaciones aritmético/lógicas.

Una instrucción elemental utiliza un tiempo constante para su ejecución, independientemente del tipo de dato con el que trabaje.
1UT.

Se considera que cada operación elemental se ejecuta en una unidad de tiempo

Ejemplos



ANALISIS TEORICO

Program uno;

var
 aux,temp,x: integer;

Begin

 aux:= 58;

 aux:= aux * 5;

 temp:= aux;

 read (x);

End.

$$T(\text{alg}) = T(1) + T(2) + T(3) + T(4)$$

T(1)= asignación = 1UT

T(2)= multiplicación +
asignación =
2UT

T(3)= asignación = 1UT

T(4)= no se considera

Ejemplos

$$T(\text{alg}) = 4UT$$

CADP – EFICIENCIA

TIEMPO DE EJECUCION



Program uno;

var

aux,temp,x: integer;

Begin

aux:= 58;

aux:= aux * 5;

if (aux > 45) then

begin

temp:= aux - 5;

x:= temp + aux + 2;

end;

x:= x * 10;

end;

$$T(\text{alg}) = T(1) + T(2) + T(3) + T(4)$$

$$T(1) = \text{asignación} = 1\text{UT}$$

$$T(2) = \text{multiplicación} + \text{asignación} = 2\text{UT}$$

$$\begin{aligned} T(3) &= \text{evaluar la condición} + \text{resta} + \text{asignación} \\ &\quad \text{suma} + \text{suma} + \text{asignación} \\ &= 1\text{UT} + 2\text{UT} + 3\text{UT} = 6\text{UT} \end{aligned}$$

$$T(4) = \text{multiplicación} + \text{asignación} = 2\text{UT}$$

$$T(\text{alg}) = 11\text{ UT}$$

CADP – EFICIENCIA

TIEMPO DE EJECUCION



```
Program uno;  
var  
  aux,temp,x: integer;
```

```
Begin  
  aux:= 58;  
  aux:= aux * 5;  
  if (aux > 45) then  
    begin  
      temp:= aux - 5;  
      x:= temp + aux + 2;  
    end  
  else  
    x:= x * 10;  
  end;
```

$$T(\text{alg}) = T(1) + T(2) + T(3)$$

$$T(1) = \text{asignación} = 1\text{UT}$$

$$T(2) = \text{multiplicación} + \text{asignación} = 2\text{UT}$$

$$\begin{aligned} T(3) = & \text{evaluar la condición} + \\ & \text{MAX (if,else)} \\ & \text{if} = \text{resta} + \text{asignación} \\ & \quad \text{suma} + \text{suma} + \text{asignación} \\ & = 5\text{UT} \\ & \text{else} = \text{multiplicación} + \\ & \quad \text{asignación} = 2\text{ UT} \\ & 1 + \text{MAX} (5,2) = 6\text{ UT} \end{aligned}$$

$$T(\text{alg}) = 9\text{ UT}$$

CADP – EFICIENCIA

TIEMPO DE EJECUCION



```
Program uno;  
var  
  aux,temp,x,i: integer;
```

```
Begin  
  aux:= 58;  
  aux:= aux * 5;  
  for i:= 1 to 10 do  
    x:= x + 1;  
    temp:= aux * 2;  
  end;
```

$T(\text{alg}) = 75 \text{ UT}$

$$T(\text{alg}) = T(1) + T(2) + T(3)$$

$T(1) =$ asignación = 1UT

$T(2) =$ multiplicación +
asignación = 2UT

$T(3) =$ tiempo del for = $3(N)+2 + N(\text{cuerpo})$
 $N = 10$

$$3(N) + 2 = 32 \text{ UT}$$

$N(\text{cuerpo}) =$

10 (suma + asignación = 2UT) +
(multip + asignación = 2UT)

$$10 (4) 40 \text{ UT}$$

$$32 + 40 = 72 \text{ UT}$$

**A veces no se conoce N y
quedará expresado en
función de N**

CADP – EFICIENCIA

TIEMPO DE EJECUCION



```
Program uno;  
var  
  aux,temp,x,i: integer;
```

```
Begin  
  read (aux);  
  for i:= 1 to aux do  
    x:= x + 1;  
    temp:= aux * 2;  
  end;
```

$$T(\text{alg}) = T(1) + T(2)$$

$T(1) =$ no se tiene en cuenta

$T(2) =$ tiempo del for = $3(N) + 2 + N(\text{cuerpo})$

$N =$ no sabemos el valor, ya que depende de aux, entonces se escribe de manera genérica

$$3(N) + 2 = \text{se deja así}$$

$$N(\text{cuerpo}) =$$

$$N(\text{suma} + \text{asignación} = 2UT) +$$
$$(\text{multip} + \text{asignación} = 2UT)$$

$$N * (4) UT$$

$$T(\text{alg}) = 3N + 2 + 4N = 7N + 2$$

CADP – EFICIENCIA

TIEMPO DE EJECUCION



```
Program uno;  
var  
  aux,temp,x,i: integer;
```

```
Begin  
  aux:= 5;  
  while (aux > 0) do  
    x:= x + 1;  
    aux:= aux - 1;  
  end;
```

$$T(\text{alg}) = T(1) + T(2)$$

$$T(1) = \text{asignación} = 1\text{UT}$$

$$T(2) = \text{tiempo del while} = N + 1 (\text{evaluar cond}) + N (\text{cuerpo})$$

$$N = 5$$

$$6 (\text{cond } 1 \text{ UT}) = 6 \text{ UT}$$

$$5 ((\text{suma} + \text{asignación} = 2\text{UT}) + (\text{suma} + \text{asignación} = 2\text{UT}))$$

$$5 * (4) = 20\text{UT}$$

$$T(\text{alg}) = 6\text{UT} + 20\text{UT} = 26 \text{ UT}$$



```
Program uno;  
var
```

```
Begin  
....  
....  
end;
```

$$T(\text{alg}) = 36 \text{ UT}$$

$$O(\text{alg}) = \text{constante}$$

```
Program dos;  
var
```

```
Begin  
....  
....  
end;
```

$$T(\text{alg}) = 2N + 4 \text{ UT}$$

$$O(\text{alg}) = N$$

```
Program tres;  
var
```

```
Begin  
....  
....  
end;
```

$$T(\text{alg}) = 2N^2 + 4N \text{ UT}$$

$$O(\text{alg}) = N^2$$

Mirando el orden puedo comparar algoritmos y elegir el más eficiente en cuanto a tiempo de ejecución

CADP – EFICIENCIA



```
Program uno;  
var  
    aux,temp,i,x: ineteger;
```

```
Begin  
    aux:= 58;  
    for i:= 1 to 10 do  
        x:= x + 1;  
        temp:= temp * 2;  
  
    for i:= 2 to 4 do  
        temp:= temp + 1;  
end;
```

TIEMPO DE EJECUCION



```
Program dos;  
var  
    aux,temp,i,x: ineteger;
```

```
Begin  
    read (aux);  
    if (aux >5) and (aux >10) then  
        for i:= 1 to 6 do  
            x:= x + 1;  
        else  
            for i:= 2 to 4 do  
                temp:= temp + 1;  
end;
```



Dados dos programas que resuelven el mismo problema. Si me piden que elija el mas eficiente, con cuál me quedo?



Dados dos programas que resuelven el mismo problema. La solución con menos instrucciones de código es la más eficiente en cuanto a tiempo de ejecución?