

# Universidad Tecnológica de Santiago (UTESA)



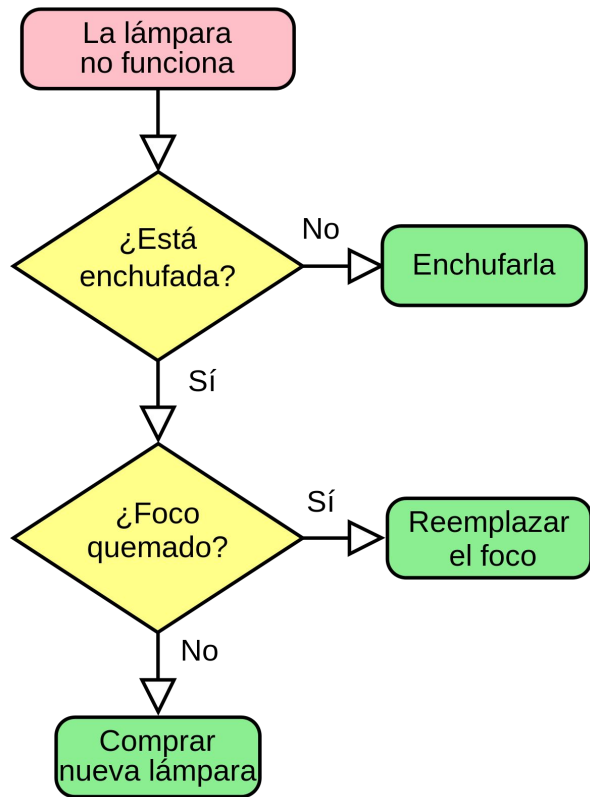
Tarea Semana 1:  
Algoritmos Paralelos Introducción.

PRESENTADO POR:  
José Rodolfo Morel. 1-16-0328.

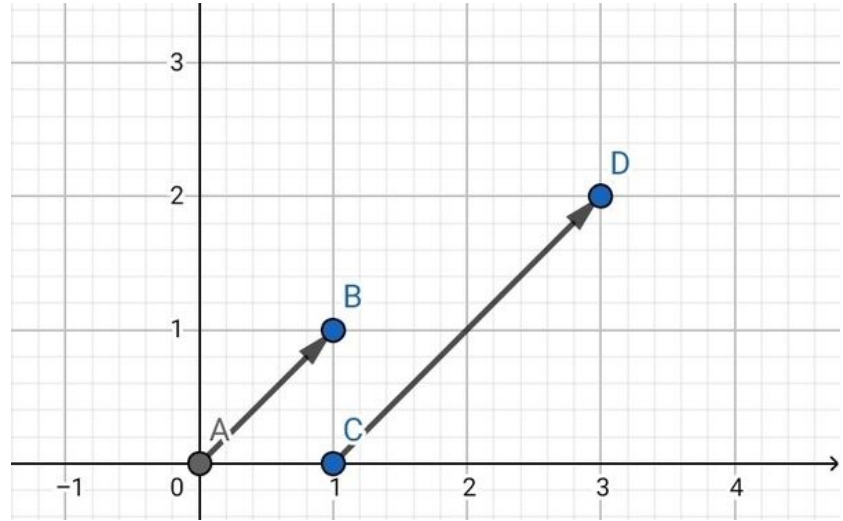
# Algoritmo

Es un conjunto bien definido de instrucciones que indican cómo realizar una operación específica o solucionar un problema determinado.

Un ejemplo de un algoritmo, sería una receta de cocina, o el manual de instalación de algún equipo o electrodoméstico, entre otras cosas.



# Paralelo



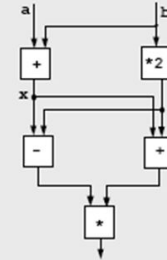
Se refiere a un objeto o entidad que existe próximo o al lado de otra, por ejemplo podemos tener procesos o actividades que pueden existir o realizarse al mismo tiempo.

# Algoritmos Paralelos

Normalmente en un algoritmo convencional un conjunto de instrucciones se ejecutan de manera secuencial en un único procesador una después de otra, pero en los algoritmos paralelos las instrucciones se separan en varias partes que son ejecutadas de forma simultánea en diferentes procesadores.

## Algoritmos paralelos

```
x = a + b;  
y = b * 2  
z = (x-y) * (x+y)
```



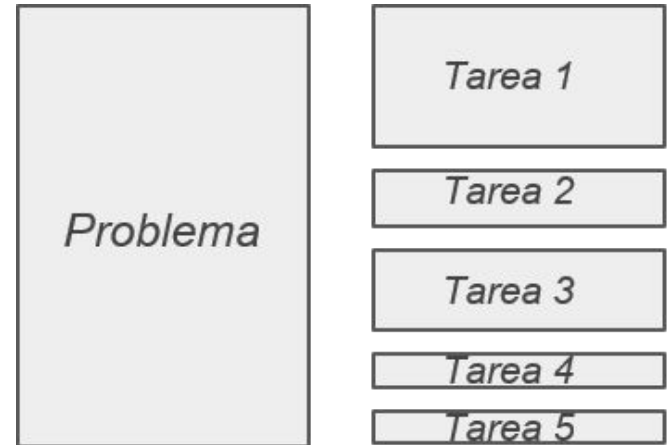
No conviene

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}$$

Conviene

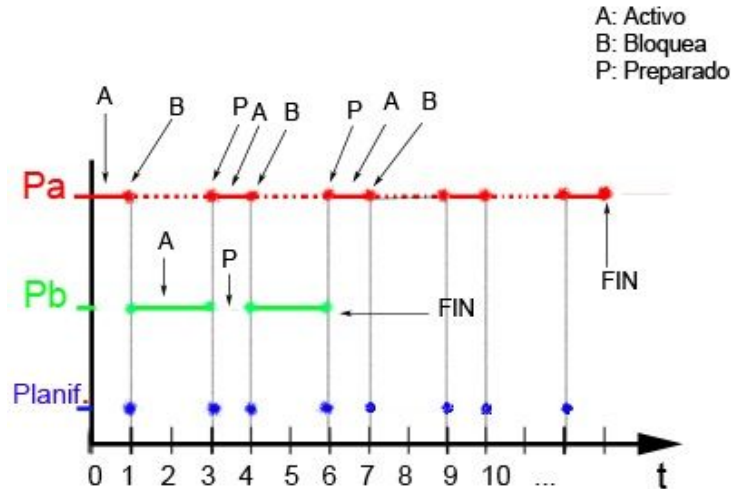
# Programación Paralela

La programación paralela se enfoca en dividir las instrucciones que se desean ejecutar en varias partes y ejecutarlas en diferentes hilos para acelerar la ejecución del programa o para ayudar a resolver problemas más complejos que no se podrían resolver de forma normal.

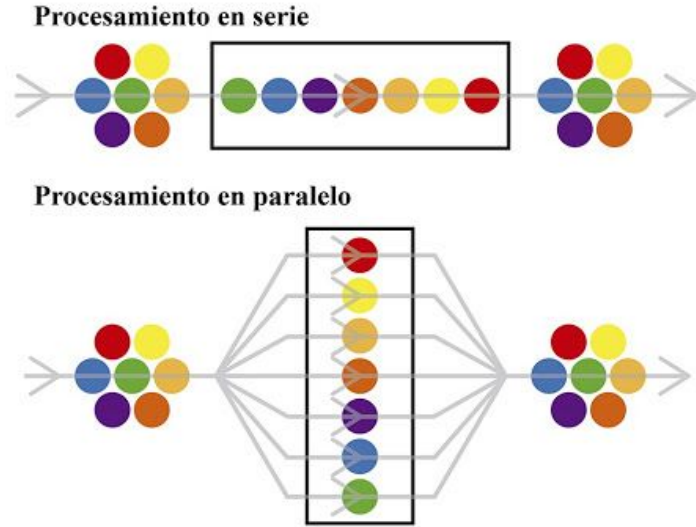


# Programación concurrente

Muchas veces la concurrencia se confunde con el paralelismo, mientras que el paralelismo varias tareas se ejecutan al mismo tiempo en procesadores separados, en la concurrencia se ejecutan varias tareas a la vez pero no al mismo tiempo, en un mismo procesador, en diferentes hilos de tiempo, un ejemplo es la imagen de abajo.



# Paralelismo

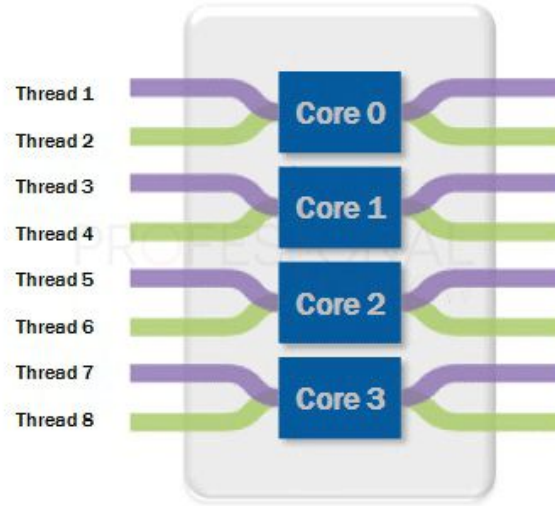


La concurrencia es similar a una persona realizando 3 tareas al mismo tiempo, alternando entre cada una, el paralelismo sería equivalente a tener 3 personas cada una realizando cada una de estas tareas.

# Hilos

A nivel de procesador los hilos no existen físicamente ya que se trata de una pila lógica de instrucciones a las cuales el procesador les asigna una prioridad de ejecución en un tiempo dado.

Los hilos son la forma como los procesadores agrupan diferentes instrucciones para después ejecutarlas posteriormente.





# Lenguajes o Frameworks de programación que usan paralelismo o programación concurrente.

- **CUDA** (usado por nvidia para acelerar el procesamiento de gráficos)
- **APACHE HADOOP** (Framework usado en servidores)
- **OPENMP** (Lenguaje de creación de API y bibliotecas de recursos compartidos en paralelo importables en FORTRAN y C/C++)
- **MPI** (Message Passing Interface) (Es un lenguaje paralelo que permite la creación de un interface de comunicación entre las diferentes rutinas en ejecución)
- **TITANIUM** (Lenguaje paralelo con un dialecto basado en JAVA, enfocada a la programación en múltiples procesadores)
- **COLECCIÓN CONCURRENTES (CNC)** (Lenguaje diseñado para el procesamiento de grandes cantidades de datos en forma de tuplas o registros de forma concurrente)

---

---

# EJERCICIOS

Por: José Rodolfo Morel

---

---

---

# EJERCICIO 1

Por: José Rodolfo Morel

---

ejercicio\_1.go > ...

```
1 package main
2
3 import ("fmt")
4
5 //Funcion que imprime un texto pasado por parametro
6 func print(str string) {fmt.Println(str)}
7
8 //Metodo princiapal
9 func main() {
10     //Imprime un grupo de mensajes de forma asincrona
11     go print("Hola 1")
12     go print("Hola 2")
13     go print("Hola 3")
14     go print("Hola 4")
15     go print("Hola 5")
16     go print("Hola 6")
17
18     //Hago un scan para bloaquear el programa y este no finalize hasta que
19     //el usuario inserte algun caracter
20     var wait string
21     fmt.Scanln(&wait)
22 }
23
```

PROBLEMS 34

OUTPUT

DEBUG CONSOLE

TERMINAL

1: go



PS C:\Users\Rodolfo\Documents\Documentos\ALGORITMOS PARALELOS\TAREA SEMANA 1\EJERCICIOS> go run .\ejercicio\_1.go

Hola 1

Hola 4

Hola 3

Hola 6

Hola 5

Hola 2

---

---

# EJERCICIO 2

Por: José Rodolfo Morel

---

ejercicio\_2.go

```
1 package main
2
3 import ("fmt" "sync")
4
5 //Declaramos la cola global para poder acceder a ella desde diferentes metodos
6 var cola sync.WaitGroup
7
8 //Metodo que imprime una secuencia de mensajes n veces
9 func rutina(str string) {
10     for i := 0; i < 3; i++ {
11         fmt.Println(str, i)
12     }
13     //Notifico a la cola que esta rutina termino
14     cola.Done()
15 }
16
17 //Metodo principal
18 func main() {
19     //Le indicamos a la cola que debe esperar 3 rutinas
20     cola.Add(3)
21     //Ejecuta 3 tareas de forma concurrente pero una vez que una de ellas
22     //inicia las otras se bloan hasta que esta termine imprmiendo un mensaje
23     //desde el punto de contro de cada rutina
24
25     go rutina("rutina 1: ")
26     go rutina("rutina 2: ")
27     go rutina("rutina 3: ")
28
29     //Bloquea el programa para que este no finalize hasta que el metodo Done()
30     //sea llamado 3 veces
31     cola.Wait()
32 }
33
```

```
PS C:\Users\Rodolfo\Documents\Documentos\ALGORITMOS>
rutina 2: 0
rutina 2: 1
rutina 2: 2
rutina 1: 0
rutina 3: 0
rutina 3: 1
rutina 3: 2
rutina 1: 1
rutina 1: 2
```

---

---

# EJERCICIO 3

Por: José Rodolfo Morel

---

go ejercicio\_4.go > ...

```
1 package main
2
3 import ("fmt")
4 //Funcion que imprime un texto pasado por parametro
5 func print(str string) {fmt.Println(str)}
6 //Metodo que realiza la sumattoria de todos los numeros dentro de un rango y retorna el total
7 func sumarRango(desde int, hasta int) int {
8
9     var suma int
10    for i := desde; i < hasta; i++ {
11        fmt.Println("de ", desde, "a", hasta, " => ", suma, " + ", i, " = ", (suma + i))
12        suma = suma + i
13    }
14    return suma
15 }
16
17 //Metodo principal
18 func main() {
19
20    //Realiza la suma de un rango de numeros desde x hasta y, para esto divide el rango
21    //en 4 partes las cuales se suman de forma separadas en diferentes rutinas para luego
22    //unir los resultados agilizando asi por 4 la velocidad en la realizacion de dicha suma.
23
24    var desde int
25    var hasta int
26    var total int
```



```

27 //Le pido al usuario los rangos desde y hasta
28 print("Ingrese el rango de la suma concurrente. ")
29 print(" desde => ")
30 fmt.Scanln(&desde)
31 print(" hasta => ")
32 fmt.Scanln(&hasta)
33
34
35 //Divido el total de numeros dentro del rango en 4 partes para sumarlas por separado
36 offset := (hasta - desde) / 4
37
38 //Realizo la sumatoria en diferentes rangos, por ejemplo si el rango es de 1 a 100,
39 //realizo la suma en 4 rangos de a 25 numeros cada uno en diferentes rutinas para
40 //acelerar el tiempo x 4.
41 go func() { total = total + sumarRango(desde, desde+offset)}()
42 go func() { total = total + sumarRango(desde+offset, desde+(offset*2))}()
43 go func() { total = total + sumarRango(desde+(offset*2), desde+(offset*3))}()
44 go func() { total = total + sumarRango(desde+(offset*3), hasta) }()
45
46 //Hago un scan para bloquear el programa y este no finalice hasta que el usuario
47 //inserte algun caracter
48 var wait string
49 fmt.Scanln(&wait)
50
51 //Imprimo el total de la sumatoria
52 fmt.Println("total = ", total)
53
54

```

```

PS C:\Users\Rodolfo\Documents\Documentos\
Ingrese el rango de la suma concurrente.
desde =>
1
hasta =>
10
de 3 a 5 => 0 + 3 = 3
de 3 a 5 => 3 + 4 = 7
de 7 a 10 => 0 + 7 = 7
de 5 a 7 => 0 + 5 = 5
de 5 a 7 => 5 + 6 = 11
de 7 a 10 => 7 + 8 = 15
de 7 a 10 => 15 + 9 = 24
de 1 a 3 => 0 + 1 = 1
de 1 a 3 => 1 + 2 = 3

total = 45

```

---

---

# EJERCICIO 4

Por: José Rodolfo Morel

---

ejercicio\_6.go

```
1 package main
2
3 import ("fmt" "math")
4
5 //Funcion que imprime un texto pasado por parametro
6 func print(str string) {fmt.Println(str)}
7
8 //Variable global compartida entre rutinas que almacenara el mayor numero encontrado
9 var mayor int
10 //Esta funcion recibe un array y busca desde la izquierda hacia la derecha el mayor numero
11 //y se detiene en el centro.
12 func buscarIzq(array [10]int) {
13
14     print("Buscando de izquierda a derecha")
15     var hasta int
16     hasta = int(math.Round(float64(len(array) / 2)))
17
18     for i := 0; i < hasta; i++ {
19         if array[i] > mayor {
20             fmt.Println("Desde la Izquierda => ", array[i], "mayor que ", mayor)
21             mayor = array[i]
22         }
23     }
24 }
```

```

26 //Esta funcion recibe un array y busca desde la derecha hacia la izquierda el mayor numero
27 //y se detiene en el centro.
28 func buscarDer(array [10]int) {
29
30     print("Buscando de derecha a izquierda")
31     var hasta int
32     hasta = int(math.Round(float64(len(array) / 2)))
33
34     for i := (len(array) - 1); i > hasta; i-- {
35         if array[i] > mayor {
36             fmt.Println("Desde la Derecha => ", array[i], "mayor que ", mayor)
37             mayor = array[i]
38         }
39     }
40 }
41
42 //Metodo principal
43 func main() {
44
45     //buscar el mayor numero en el arreglo para esto lo divide en 2 partes y comienza
46     //a recorrerlo por izquierda y derecha hasta terminar en la mitad
47     print("Buscando el mayor en: ")
48     array := [10]int{2, 3, 6, 7, 8, 1, 23, 34, 4, 5}
49     fmt.Println(array)
50
51     //busco el mayor recorriendo la parte derecha y izquierda del array al mismo tiempo
52     go buscarIzq(array)
53     go buscarDer(array)
54
55     //Hago un scan para bloquear el programa y este no finalice hasta que el usuario
56     //inserte algun caracter
57     var wait string
58     fmt.Scanln(&wait)
59
60     //Imprimo el mayor
61     fmt.Println("El mayor es = ", mayor)
62 }
63


```

```

PS C:\Users\Rodolfo\Documents\Documentos'
Buscando el mayor en:
[2 3 6 7 8 1 23 34 4 5]
Buscando de derecha a izquierda
Desde la Derecha => 5 mayor que 0
Desde la Derecha => 34 mayor que 5
Buscando de izquierda a derecha

```

El mayor es = 34

A close-up photograph of a hand holding a black smartphone. The hand is positioned in the lower right quadrant of the frame. The background is a blurred red surface with white vertical stripes. Overlaid on the image is the text 'Gracias por su atencion!' in a large, white, sans-serif font. The text is centered horizontally and vertically, with 'Gracias por su' on the top line and 'atencion!' on the bottom line.

**Gracias por su  
atencion!**