# Computacíon Paralela y Distribuída

2022-II

José Fiestas

19/08/22

Universidad de Ingeniería y Tecnología
jfiestas@utec.edu.pe

# Unidad 1: Fundamentos de paralelismo y arquitecturas paralelas

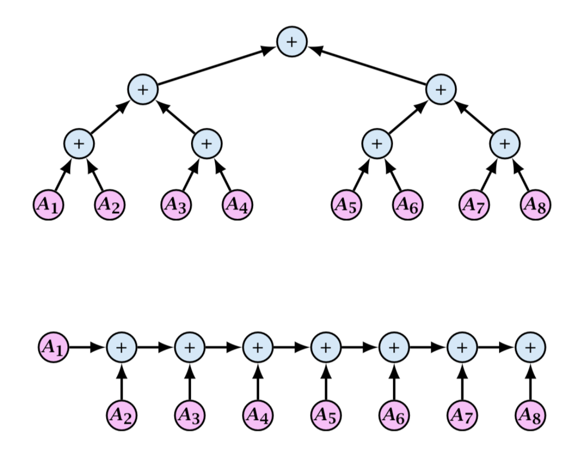
Al finalizar la unidad, los alumnos conocen:

1. Transición del procesamiento secuencial al paralelo
2. Taxonomia de Flynn
3. Arquitecturas paralelas: memoria compartida vs memoria distribuida.
4. Paradigmas del paralelismo
5. Metas del Paralelismo: velocidad y precisión
6. DAG (Directed Acyclic Graphs)

# 6. DAG (Directed Acyclic Graphs)

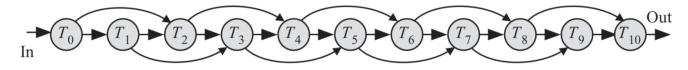
# Modelo DAG (Directed Acyclic Graph)

- vértices representan operaciones (instrucciones simples o en bloques)
- aristas representan dependencias (precedencia)

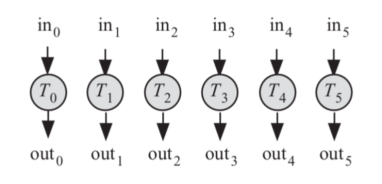


Podemos clasificar a los algoritmos de la siguiente forma:

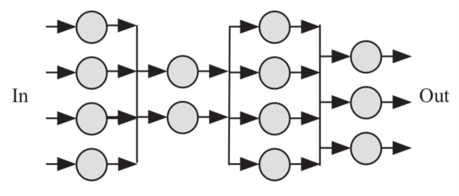
- **Secuenciales**, no pueden ser paralelizados porque todas las tareas tienen dependencias en tareas previas



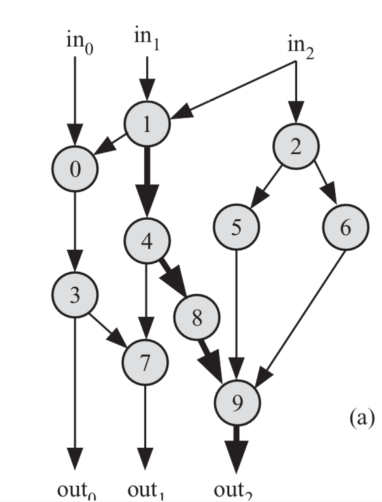

- **Paralelos**,donde todas las tareas pueden ser ejecutadas simultáneamente



- **SPA** (Secuencial-Paralelo) El algoritmo está separado en niveles, que se ejecutan en paralelo, pero los cuales tienen una forma secuencial de ejecución.



- **NSPA** (No-Secuencial-Paralelo) El algoritmo no contiene ninguno de los patrones anteriores



# Modelo de costo DAG

El **scheduling** o 'plan de ejecución' de un DAG (V,E), asigna a cada nodo v:
- un tiempo de ejecución $t_v$, $(u, v) \in E \rightarrow t_v = t_u + 1$
- un procesador $p_v$, $v \in \{0, ..., p-1\}$

donde, $t_x = 0$ para los nodos de input.
El algoritmo paralelo correspondiente se ejecuta en T $=$ MAX($\sum t_i$), que es la longitud del plan de ejecución, llamada **span**
El **trabajo** se define como la cantidad total de operaciones (instrucciones). Es decir, la complejidad secuencial.
Se busca que el algoritmo en paralelo tenga un **trabajo** eficiente (mínimo) y una profundidad (**span**) mínima

# Modelo de costo DAG

El **trabajo** equivale al tiempo secuencial (optimo): $W(n) = T_s^*(n)$
Ya que un proceso ejecuta una instruccion por unidad de tiempo, p procesos ejecutaran, **como máximo**, p instrucciones por unidad de tiempo. Por lo que p procesos tomaran por lo menos $T_s/p$ unidades de tiempo. Es decir

$$T_p \geq T_s/p$$

Esto nos permite definir la velocidad (speedup) de un proceso en paralelo como:

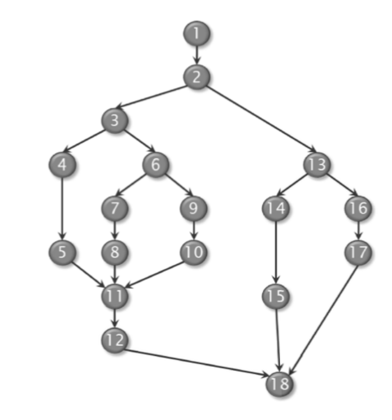
$$S = \frac{T_s}{T_p}$$

Un speedup igual (o proporcional) a p es un speedup lineal optimo, ya que $S = \frac{T_s}{T_p} \leq p$
Denotemos al camino critico del DAG como span ($T_\infty$). Tenemos que

$$T_p \geq T_\infty$$

# Ejemplo 1

Determine
span y trabajo en
el siguiente DAG si cada
tarea tiene complejidad
constante, O(1)



# Ejemplo 2: función recursiva

Considere la función :
**f(x)=x**, si $x \leq 1$
**f(x)= f(x-1)+f(x-2)**, i.e. (a,b)=(f(x-1) || f(x-2))

Determine el span y trabajo

# Ejemplo 3: Suma de números

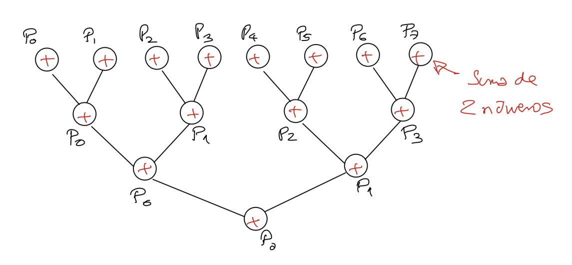
**a)** Construya un DAG eficiente para la suma de n enteros en paralelo, si se tiene n=16 y p=5 (procesadores disponibles)

**b)** Compare el DAG con el caso n=16, p=8

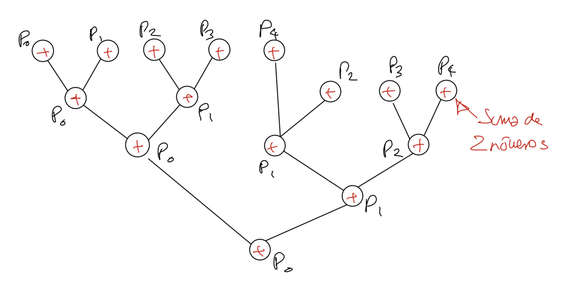
**c)** Determine $T_s$, $T_\infty$(n), W(n), S(n) en cada caso

# Ejemplo 3:

a)




b)



# Ejemplo 3

Determine $T_s$, $T_\infty$(n), W(n), S(n) en cada caso

Contabilizando aristas en cada DAG, se obtiene para el caso
**a)** $T_s(n = 16) = 15$, $T_\infty$(n=16)=4, $W(n = 16) = 15$, S(n)=15/4, E(n)=15/4/8=0.47
**b)** $T_s(n = 16) = 15$, $T_\infty$(n=16)=5, $W(n = 16) = 15$, S(n)=15/5, E(n)=15/5/5=0.60
Se utiliza la misma operación en cada nodo, lo que permite el cálculo por el conteo de aristas. Esto no es posible si las operaciones entre nodos difieren. No olvidar que son cantidades proporcionales al tiempo de ejecución.
Se observa un menor speedup y una mayor eficiencia en el segundo caso. Esto se explica por el mayor aprovechamiento de recursos (procesos).

# Ejemplo 4

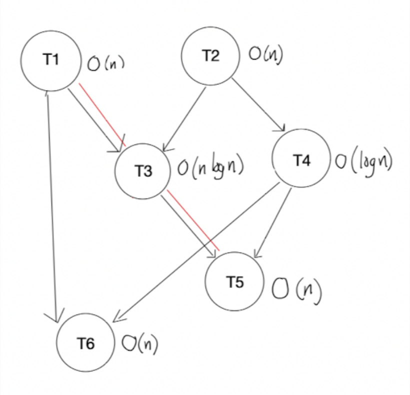
Diagrame el DAG correspondiente al siguiente código

```
double a[N],b[N],c[N],v=0.0,w=0.0;
T1(a,&v);
T2(b,&w);
T3(b,&v);
T4(c,&w);
T5(c,&v);
T6(a,&w);
```

Las funciones leen y modifican ambos argumentos.
Determine $T_s$, $T_\infty$(n), W(n), S(n) del algoritmo paralelo asumiendo una complejidad constante de las tareas dadas.
Ahora recalcule $T_s$, $T_\infty$(n), W(n), S(n) si la complejidad de T1, T2, T5 y T6 es O(n), mientras que T3(n)=O(n log n), y T4(n)=O(log n)

# Ejemplo 4

Complejidad O(1):
$T_s(n) = 6,\ W(n) = 6$
$T_p(n) = 3$
Complejidad real:
$T_s(n) = O(n + n + n + n + nlog(n) + log(n)) = O(nlog(n))$
$W(n) = O(nlog(n))$
$T_p(n) = O(n + nlog(n) + n) = O(nlog(n))$
$S(n) = O(\frac{nlog(n)}{nlog(n)}) = O(1)$
$E(n) = O(\frac{1}{p})$



# Bibliografía i


- David B. Kirk and Wen-mei W. Hwu *Programming Massively Parallel Processors: A Hands-on Approach*. 2nd. Morgan Kaufmann, 2013. isbn: 978-0-12-415992-1.
- Joseph Jájá *An introduction of parallel algorithms.* University of Maryland ***
- Norm Matloff. *Programming on Parallel Machines*. University of California, Davis, 2014.
- Peter S. Pacheco. *An Introduction to Parallel Programming.* 1st. Morgan Kaufmann, 2011. isbn: 978-0-12-374260- 5.
- Michael J. Quinn. *Parallel Programming in C with MPI and OpenMP.* 1st. McGraw-Hill Education Group, 2003. isbn: 0071232656.

# Bibliografía ii

Jason Sanders and Edward Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Program- ming*. 1st. Addison-Wesley Professional, 2010. isbn: 0131387685, 9780131387683.

*** en esta clase