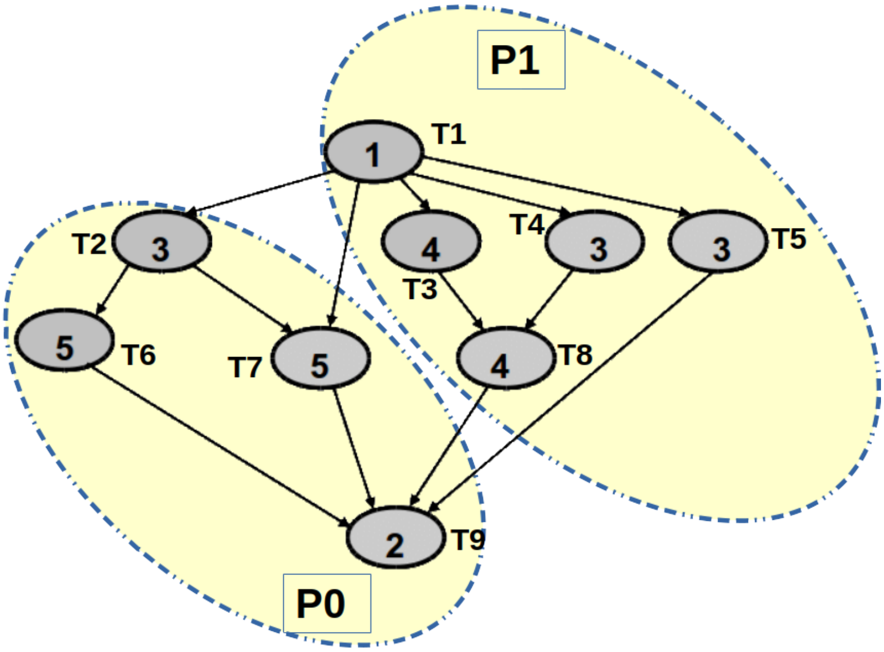
PROPUESTA DE SOLUCIÓN EJERCICIOS 1, 2,3. TEMA 3

1. ¿Cuál es el grado medio de concurrencia del siguiente grafo de dependencias? Asumiendo que el costo de comunicación fuese despreciable, cuál sería la asignación óptima de las tareas del grafo a dos procesos.



Longitud Camino crítico= L=1+4+4+2=11

Grado Concurrencia = (1+3+4+3+3+5+5+4+2)/11=30/11=2.72

Planificación de tareas óptima estática (no sería la única óptima):

<table>
<tr><td>P0</td><td></td><td colspan="2">T2</td><td colspan="3">T6</td><td colspan="2">T7</td><td></td><td>T9</td></tr>
<tr><td>P1</td><td>T1</td><td colspan="3">T3</td><td colspan="2">T4</td><td colspan="2">T5</td><td colspan="2">T8</td><td></td></tr>
</table>

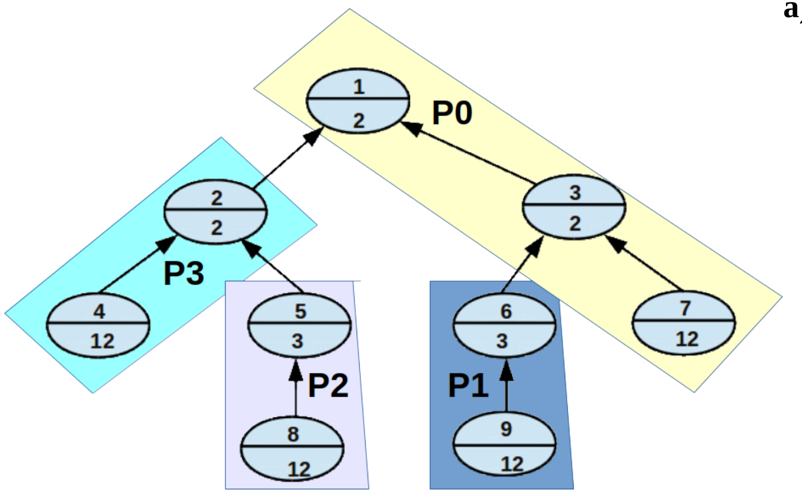
0 1 4 5 8 9 11 14 15 17

Tiempo de ejecución estimado =17 unidades

Todas las asignaciones a 2 procesos que permitan un tiempo final de 17 unidades son óptimas.

2. **Cuál es el grado medio de concurrencia de los siguientes grafos de dependencias entre tareas en el que cada nodo está etiquetado con su identificación en la parte superior y con su coste en la parte inferior. Asumiendo que el costo de comunicación entre tareas fuese despreciable, establecer cuál sería la asignación óptima de las tareas del grafo a cuatro procesos en el caso a) y a tres procesos en el caso b). Dibujar un diagrama de ejecución.**

a)



Longitud de camino crítico=L=19

Grado Concurrencia = 48/19=2.52

Planificación de tareas óptima estática (no sería la única óptima):

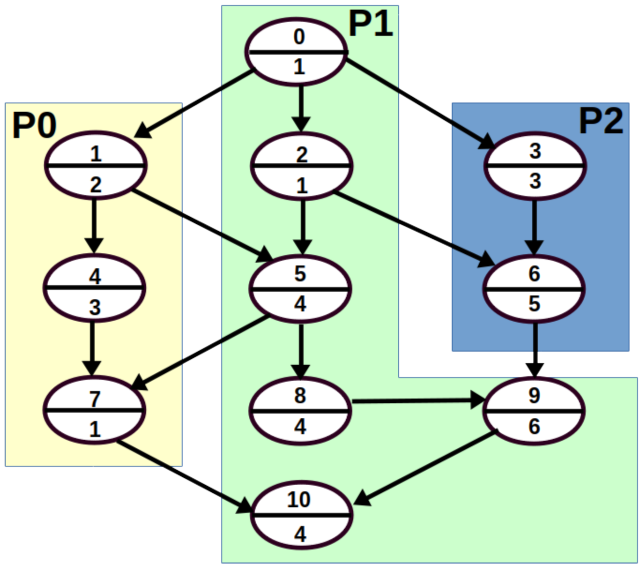
P0	**T7**		**T3**	**T1**
P1	**T9**	**T6**		
P2	**T8**	**T5**		
P3	**T4**		**T2**	

0 12 14 15 17 19

Tiempo de ejecución estimado =19 unidades

Todas las asignaciones a 4 procesos que permitan un tiempo final de 19 unidades son óptimas.

b)



Longitud de camino crítico (T0-T1-T5-T8-T9-T10) =L=21

Grado Concurrencia = 34/21=1.62

Planificación de tareas óptima estática (no sería la única óptima):

P0		T1	T4		T7		
P1	T0	T2		T5	T8	T9	T10
P2		T3	T6				

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21

Tiempo de ejecución estimado = 21 unidades

Todas las asignaciones a 3 procesos que permitan un tiempo final de 21 unidades son óptimas.

3. ¿Se desea paralelizar un cálculo iterativo que tiene como entrada un vector de reales **y** de dimensión **N** (**y**=(**y1**,**y2**,…,**yN**)) y devuelve como salida otro vector real **dy** también de dimensión **N** que se calcula en como:

$$\boxed{dy_i^{(k+1)} = \frac{y_{i-1}^{(k)} + y_i^{(k)} * y_{i+1}^{(k)} - y_{i+2}^{(k)}}{8}, \qquad k=0,\ldots M.}$$

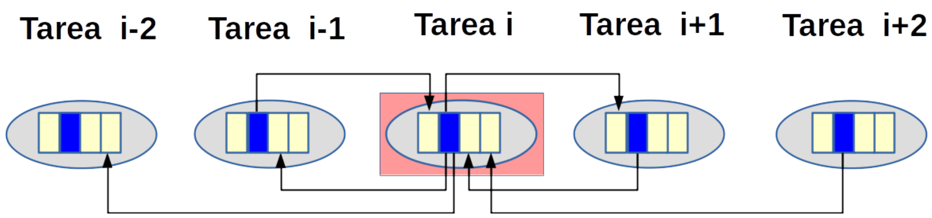
donde los valores del lado derecho que entran fuera del rango se determinan cíclicamente como: $\boxed{y_0^{(k)} = y_N^{(k)}, \quad y_{N+1}^{(k)} = y_1^{(k)}, \quad y_{N+2}^{(k)} = y_2^{(k)}}$

Se supone que la función se evaluará repetidamente durante un cierto número de iteraciones y en cada iteración el vector de salida de la iteración anterior será el vector de entrada en la iteración actual.

a) Establecer la descomposición de tareas así como la estructura de comunicación y las operaciones de comunicación necesarias para coordinar las tareas. Se puede asumir que or el reparto inicial del vector de entrada entre las tareas está ya hecho y que no es necesario recolectar el vector de salida en la última iteración.

Aplicando descomposición de dominio, podemos centrarnos en descomponer el vector de salida *dy*. La descomposición más agresiva consiste en generar tantas particiones como elementos hay en *dy*, es decir, asignar una tarea al cálculo de cada *dy*$_i$ en todas las iteraciones.

Un esbozo de la estructura de comunicación resultante, para una iteración y solo fijándonos en la Tarea i-ésima, sería esta:



donde cada tarea mantiene localmente un vector de 4 elementos donde la segunda celda del vector mantiene el valor *y*$_i$ en la iteración actual y se encarga de calcular el valor *dy*$_i$ en dicha iteración

La descripción del código de la Tarea i-ésima, asumiendo que se sigue un modelo de paso de mensajes y que cada tarea mantiene inicialmente el valor y_i en la iteración 0, podría ser la siguiente:

```
Tarea ( i, i=1,…,N )
double y[4];
y[1]=….;   // Dar valor inicial a yi
for  k=0 to M
   send(&y[1], (i+1) mod N);
   send(&y[1], (i-1+N) mod N);
   send(&y[1], (i-2+N) mod N);
   receive(&y[2], (i+1) mod N);
   receive(&y[3], (i+2) mod N);
   receive(&y[0], (i-1+N) mod N);
   // Por simplicidad el resultado se guarda en el vector de entrada para la siguiente iteración
   y[1]=(y[0]+y[1]+y[2]+y[3])/8;
```

También se podría optar por una descomposición flexible, donde cada tarea se encargara de calcular aproximadamente N/P elementos consecutivos del vector dy para un número P de tareas. Esto allanaría el trabajo en la etapa de asignación de tareas a procesos.

b) Definir una estrategia de asignación eficiente sobre 4 y 8 procesadores. Cómo se distribuyen el vector de entrada y el vector solución entre los procesadores con la solución que se propone.

Para P=4 o P=8, la asignación más eficiente consiste en asignar a cada proceso (cada proceso se ejecutaría sobre un procesador) un bloque de aproximadamente N/P elementos contiguos del vector *y*, de tal forma que cada proceso calcularía, en cada iteración, las mismas posiciones del vector resultado *dy* que tiene para el vector *y*. Para equilibrar la carga adecuadamente, se puede utilizar la distribución por bloques estándar para vectores, que asigna al proceso i (i=0,…, P-1) los siguiente índices de elementos del vector *y (se asume que N>>P)*:

Se calcula el tamaño de bloque: Bsize=ceil((float)N)/P).

El proceso i se encargaría de los elementos: i*Bsize, …., min(Índice inicial+Bsize, N).