1. Supongamos que se dispone de un número m de funciones polinomiales de grado n, de tal forma que la i-ésima función polinomial se puede representar como:

$$\mathbf{f^i(x)= a_{i,0}+a_{i,1}+\dots a_{i,n-1}x^{n-1}+a_{i,n}x^n \;,\; i=0,1,\dots,m-1}$$

   Se desea evaluar todas las funciones sobre un valor real x=b y obtener el valor mínimo, es decir, obtener un valor v tal que:

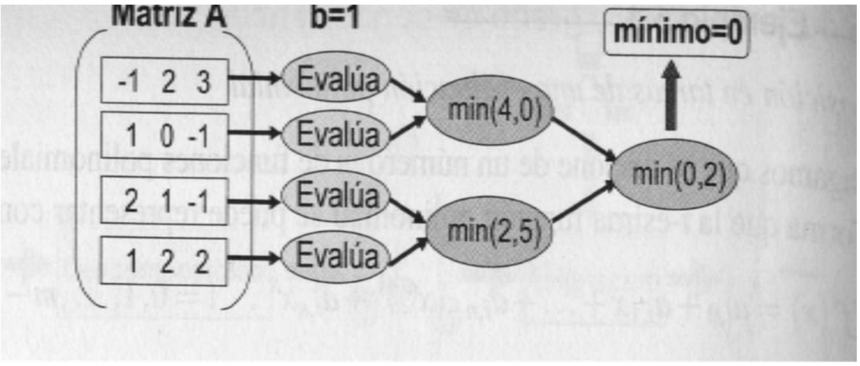
$$\mathbf{V= \min \{ f^i(x)\} \text{ para } i\{0,\dots,m-1\}}$$

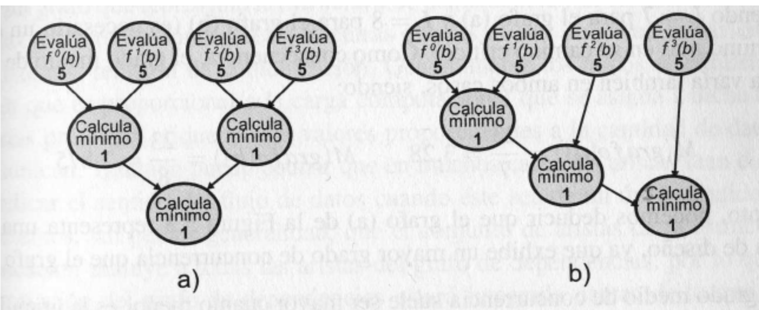
   Para ello, los coeficientes de cada una de las funciones polinomiales se guardan en una matriz A de tamaño m x (n+1) con la forma:

$$A=\begin{pmatrix} a_{0,0} & a_{0,1} & a_{0,n} \\ \dots & \dots & \dots \\ a_{m-1,0} & a_{m-1,2} & a_{m-1,n} \end{pmatrix}$$

   a. Plantear la descomposición de tareas para el problema de la evaluación de polinomios
   b. desarrollar grafo de dependencias para 4 polinomios de grado 2
      i. Máximo grado de concurrencia
      ii. Grado medio de concurrencia

Por ejemplo, podría ser





$$M(grafo(a)) = \frac{23}{7} = 3.28 \qquad M(grafo(b)) = \frac{23}{8} = 2.875$$

2. Realiza la descomposición centrada en los datos de entrada para la realización del producto escalar de vectores.

$$X = (x_0, x_1, \ldots, x_{n-1}),\ y\ (y_0, y_1, \ldots, y_{n-1})$$

$$x \cdot y = x_0 y_0 + x_1 y_1 + \ldots + x_{n-1} y_{n-1}$$

Hay que centrarse necesariamente en los datos de entrada, ya que los datos de salida están formados por un único escalar, sería razonable asignar el mismo número de elementos de x y de y a cada tarea, podríamos pensar en p tareas, cada una gestionando un bloque de los vectores y realizando la operación

$$d_i = \sum_{j=i\frac{n}{p}}^{(i+1)\frac{n}{p}-1} x_j y_j$$

(n/p) es el número de elementos que gestiona cada tarea, una vez obtenidas estas sumas parciales, las podemos consolidar con un enfoque recursivo igual que en el problema anterior.

3. Dado un arreglo A de longitud N, encontrar una descomposición recursiva de tareas que encuentre el mínimo elemento del arreglo. Comentar brevemente cómo funcionaría el algoritmo paralelo.

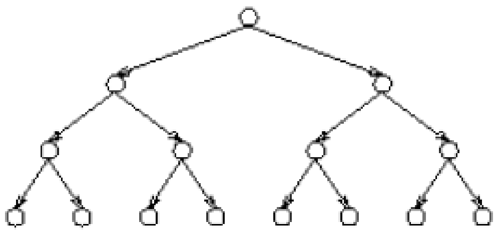
```
procedure SERIAL_MIN (A, n)
begin
     min = A[0];
for i := 1 to n - 1 do
     if (A[i] < min) min := A[i];
End for;
return min;
end SERIAL_MIN
```

Un programa en serie para encontrar el mínimo en una matriz de números A de longitud n.

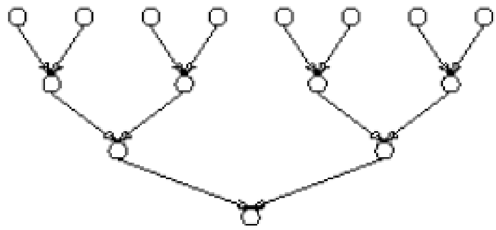
```
procedure RECURSIVE_MIN (A, n)
begin
if (n = 1) then
    min := A[0];
else
    lmin := RECURSIVE_MIN (A, n/2);
    rmin := RECURSIVE_MIN (&(A[n/2]), n - n/2);
    if (lmin < rmin) then
        min := lmin;
    else
        min := rmin;
    endelse;
endelse;
return min;
end RECURSIVE_MIN
```

5. Para los grafos de dependencia de tareas de la Figura siguiente, determinar:

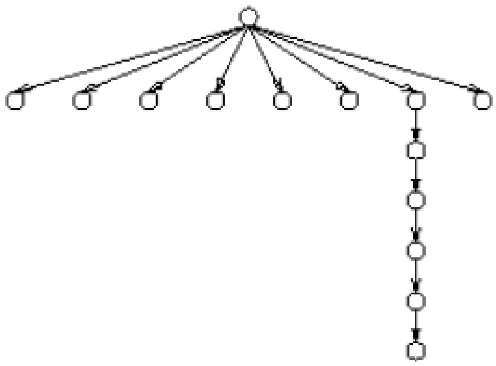
   a. Máximo grado de concurrencia.
   b. Longitud del camino crítico.
   c. Máximo speed-up alcanzable sobre un sistema monoprocesador si un número arbitrariamente grande de procesos están disponibles.
   d. Mínimo número de procesos para obtener el máximo speed-up.

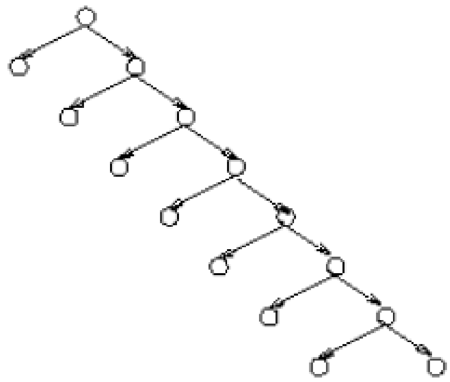


(a)



(b)





Asumimos un peso de 1 en cada nodo

1. (a) 8, (b) 8, (c) 8, (d) 8.
2. (a) 4, (b) 4, (c) 7, (d) 8.
3. (a) 15/4, (b) 15/4, (c) 2, (d) 15/8.
4.- (a) 15/4, (b)15/4, (c) 8, (d) 15/8

6. Supongamos que debemos evaluar la expresión:

$$x = \frac{a * b + c}{a * b - d}$$

Genera un programa capaz de ejecutarse en un computador paralelo mostrando el grafo de dependencia de datos para la expresión anterior.

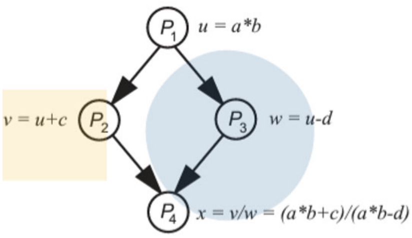
P1 = u = a*b

P2 = v= u+c

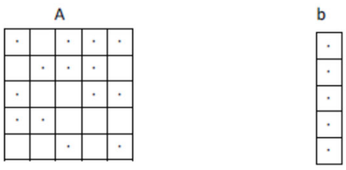
P3 = w = u-d

P4 = v/w

COnsiderando la dependencia de tareas tendríamos



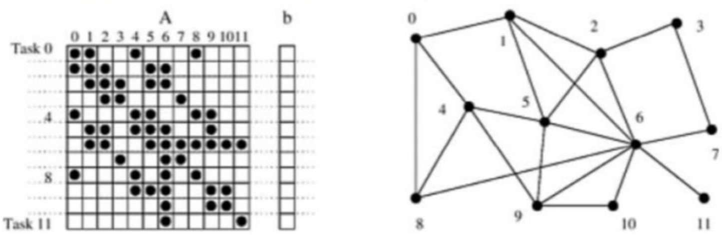
7. Diseñar el grafo de dependencia e interacción para una matriz dispersa:



$$\sum_{0 \leq j \leq 11, A[i,j] \neq 0} A[i, j].b[j]$$

MIRAR el ejemplo de la Transparencia, es exactamente igual sólo cambian los datos que comparten las tareas, cada tarea procesa una fila y el correspondiente valor del vector b.

Multiplicación de matriz dispersa: $\sum_{0 \leq j \leq 11, A[i,j] \neq 0} A[i, j].b[j]$



Descomposición: 11 tareas → A[i,*] b[i]

y[0] = A[0, 0].b[0] + A[0, 1].b[1] + A[0, 4].b[4] + A[0, 8].b[8]

Task 4 envía *b[4]* a *Tasks 0, 5, 8, 9* y coge *b[0], b[5], b[8], b[9]*