PC3. Sanchez Sauñe.

Ejercicio 1: Suma de prefijos

Para $p < n$

Para $p = n$, tenemos:

→ for $j = 1$ to n do $\Big\}\ w_1 = O(n)$

→ For $h = 1$ to $\log n$ do $\Big\}\ w_2 = \sum_{j=1}^{\log n} \frac{n}{2^j}$
  for $j = 1$ to $\frac{n}{2^h}$ pardo

→ for $h = \log n$ to 0 do $\Big\}\ w_3 = \sum_{\log n}^{0} \frac{n}{2^h}$
  for $j = 1$ to $\frac{n}{2^h}$ pardo

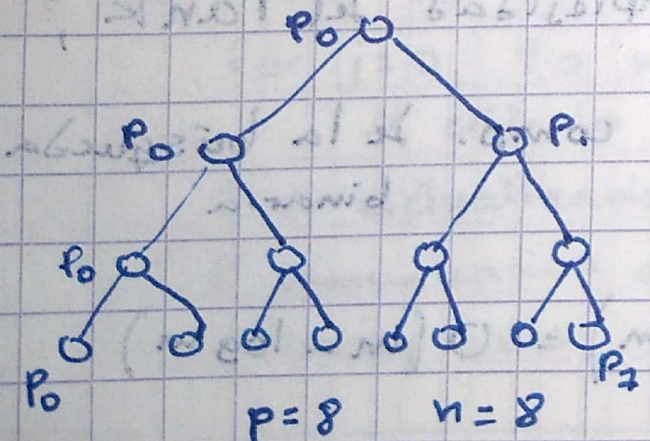
PRAM: $\frac{n}{p}$ $\quad n = 2^k$ $\quad p = 2^q$

$T(n, p)$ $\quad w(n, p)$ $\longleftrightarrow$ $T(n)$ $\quad w(n)$

$w(n) = w_1 + w_2 + w_3 = O(n)$

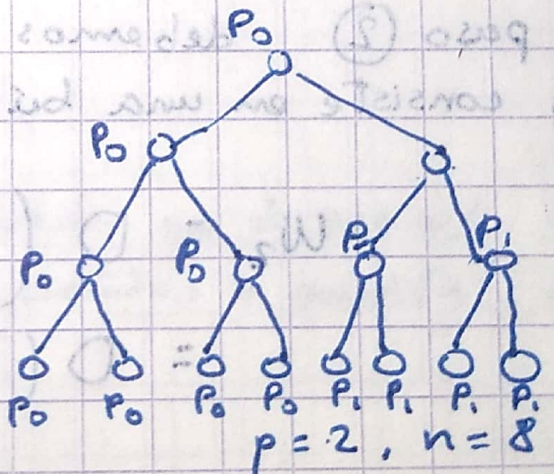
$T(n) = O(\log n)$

$p = n$

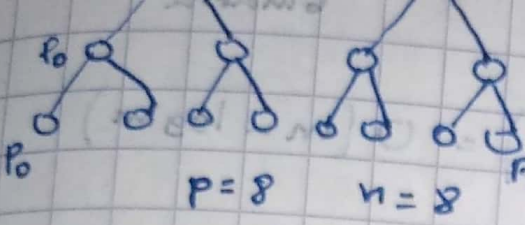


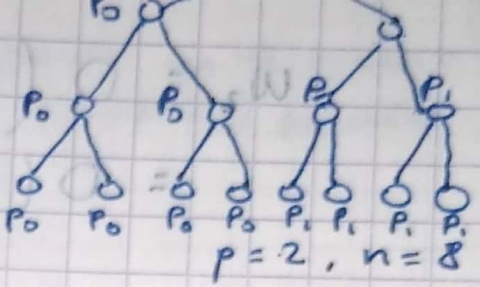
$p = 8$ $\quad n = 8$

$p < n$



$p = 2$, $n = 8$





Modificando para $p < n$:

→ for $j = 1$ to n

→ for $h = 1$ to $\log n$ do

for $j = 1$ to $\frac{n}{p \times 2^h}$ pardo

→ for $h = \log n$ to 0 do

for $j = 1$ to $\frac{n}{p \times 2^h}$ pardo

Usando el teorema de Brent.

$$\frac{w}{p} \leq T(n,p) \leq \frac{w(n)}{p} + S(n)$$

$$T(n,p) = \frac{w}{p}(n) + S(n) = \frac{O(n)}{p} + O(\log n)$$

$$T(n,p) = O\left(\frac{n}{p}\right) + O(\log n) = O\left(\frac{n}{p} + \log n\right)$$

$$w(n,p) = O(n + p \log n)$$

Se observa que el algoritmo con $n = p$ es más eficiente en $T(n)$ y $w(n)$ comparado con $p \leq n$

Ejercicio 2:

$A = [a_1, \dots, a_n]$ $\quad \log(m)$

$B = [b_1, \dots, b_m)$ $\quad k_{(m)} = m / \log(m)$

① $j(0) = 0$ $\quad j(k) = n$ $\Big\}$ $w_1 = O(1)$

② for $i = 1$ to $k(m) - 1$ par do $\Big\}$ w_2

$j(i) = \text{rank}(b_{i \log m} : A)$

③ for $i = 0$ to $k(m) - 1$ par to $\Big\}$ $w_3 = O\left(\frac{m}{\log m}\right)$

$B_i = (b_{i \log m + 1}, \dots, b_{(i+1)\log m})$

$A_i = (a_{j(i)+1}, \dots, a_{j(i+1)})$

(a) En el paso ② debemos considerar la complejidad del rank, el cual consiste en una búsqueda binaria:

$w_2 = O(\log m \cdot (k-1))$ ← comp. de la búsqueda binaria

$= O\left(\cancel{\log m} \times \frac{m}{\cancel{\log m}} - \log m\right) = O(m - \log m)$

$\Rightarrow w = \cancel{w_1} + w_2 + w_3$

↳ no se suele considerar

$w = O\left(m - \log m + \frac{m}{\log m}\right) = O(m) \Rightarrow T(m) = \frac{O(m)}{=}$

(a) En el paso (2) debemos considerar la complejidad de rank, el cual consiste en una búsqueda binaria:

$$W_2 = O\left(\log m \cdot (k-1)\right) \rightarrow \text{comp. de la búsqueda binaria}$$

$$= O\left(\cancel{\log m} \times \frac{m}{\cancel{\log m}} - \log m\right) = O(m - \log m)$$

$$\Rightarrow \quad W = \cancel{W_1} + W_2 + W_3$$

↳ no se suele considerar

$$W = O\left(m - \log m + \frac{m}{\log m}\right) = O(m) \Rightarrow T(m) = \frac{O(m)}{P(m)} = O(1)$$

Sabemos que Merge sort tiene un tiempo total $T(m) = O(m \log m)$

⇒ Como el $T(m)$ encontrado es menor que $O(m \log m)$, es ÓPTIMO

(b) Analizando el algoritmo, el modelo PRAM que usa es el de lectura concurrente y escritura exclusiva

CREW

pues en el proceso (2) se accede a un mismo array A, que no es exclusivo para un proceso.
También en el proceso (3) y (2) se escribe exclusivamente B y A en cada proceso, mediante el PARDO y los índices de B, A y j respectivamente.

Ejercicio 3:

(a) Memoria compartida

→ for i=0 to n-1 pardo
    for j=0 to n-1 pardo
        // Cálculo aceleración usando memoria compartida
    // actualización velocidad y posiciones

(b) Memoria distribuida

→ for i=0 to n-1 pardo
    for j=0 to n-1 do
        // cálculo de aceleración
    // actualización velocidad y posición en el proceso actual
    // comunicación de los valores de velocidad y posición a los demás procesos.

(d) Por cada cuerpo tengo 7 floats. En total tenemos $7 \times N$ (con $N = 6 \times 10^6$) floats en memoria por iteración, también debemos considerar la memoria necesaria para las operaciones.

N si modifica el criterio (a) pues no sería muy útil usar una memoria compartida para una tamaño absurdo de datos float, por lo que es necesario usar memoria distribuida a pesar de sus contra respecto a la comunicación.