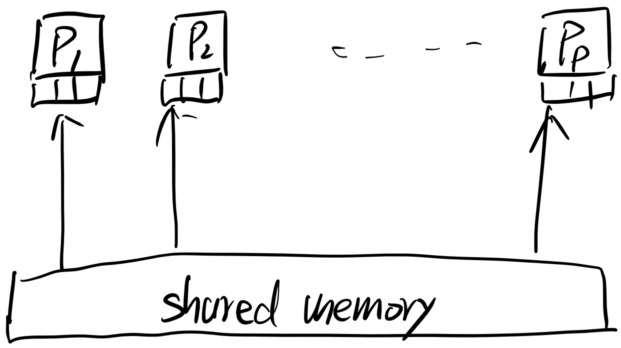
Parallel Random Access Memory (PRAM)



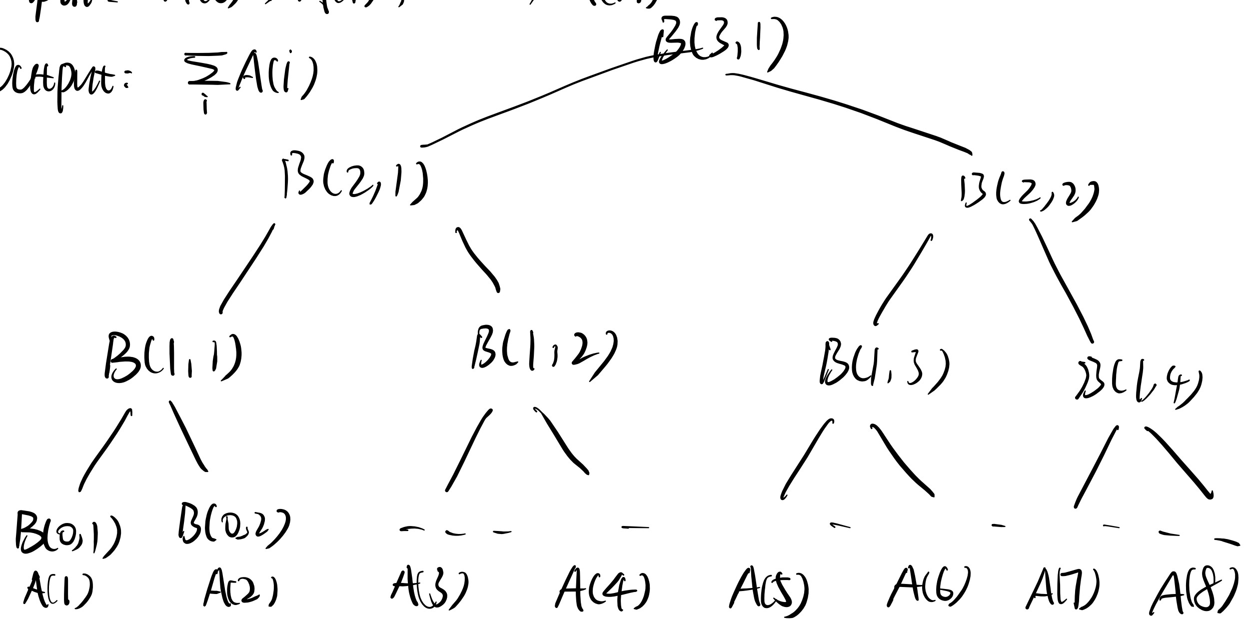
read, write, compute in $O(1)$ time

for P_i, $1 \le i \le p$ pardo
    P_i does something

Summation

Input: $A(0), A(1), \ldots, A(n)$

Output: $\sum_i A(i)$



for P_i, $1 \le i \le 4$ pardo
    $B(0, 2i-1) = A(2i-1)$
    $B(0, 2i) = A(2i)$
for $h = 1$ to 3
    for P_i, $1 \le i \le 4$ pardo

if $i \le \frac{8}{2^h}$

$B(h,i) = B(h-1,2i) + B(h-1,2i-1)$

else

stay idle

for P_i, $i=1$ pardo

Output $B(3,1)$

Work - Depth Presentation

for i, $1 \le i \le n$ pardo

$B(0,i) = A(i)$

for $h = 1$ to $\log n$

for i, $1 \le i \le \frac{n}{2^h}$ pardo

$B(h,i) = B(h-1,2i) + B(h-1,2i-1)$

for i, $i=1$ pardo

output $B(\log n, 1)$

Workload: total amount of work $W = \Theta(n)$

Depth: maximum amount work of any chain of dependencies $D = \Theta(\log n)$

p processors: running time $= \frac{W}{P} + D$

serial parallel

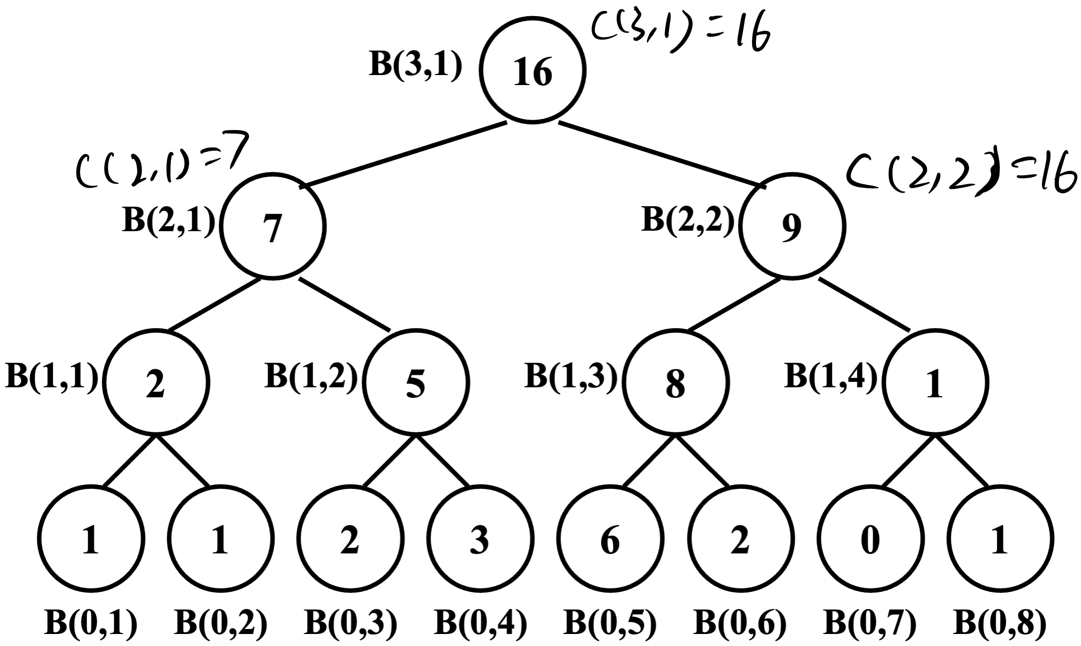
good

$O(n)$	$W=O(n)$	$W=O(n^2)$	$W=O(n)$
	$D=O(n)$	$D=O(\log n)$	$D=O(\log n)$

Prefix Sum

Input: $A(0), \ldots\ldots, A(n)$

Output: $\sum_{j=1}^{i} A(j)$ for any i.



$C(h,i)$ = the rightmost descendant leaf of node (h,i)

+

all the leaves left to it.

$C(h,1) = B(h,1)$

$C(h,i) = C(h+1, i/2)$ if i is even // right child

$C(h,i) = B(h,i) + C(h+1, (i-1)/2)$ if i is odd.

$\Downarrow$

$C(0,i) = \sum_{j=1}^{i} A(0)$

```
for i, 1 ≤ i ≤ n pardo
    B(0, i) = A(i)
for h = 1 to log n
    for i, 1 ≤ i ≤ n/2^h pardo
        B(h, i) = B(h-1, 2i-1) + B(h-1, 2i)
for h = log n to 0
    for i, 1 ≤ i ≤ n/2^h pardo
        if i == 1
            C(h, 1) = B(h, 1)
        if i is even
            C(h, i) = C(h+1, i/2)
        if i is odd
            C(h, i) = B(h, i) + C(h+1, (i-1)/2)
for i, 1 ≤ i ≤ n pardo
    output C(0, i)
```

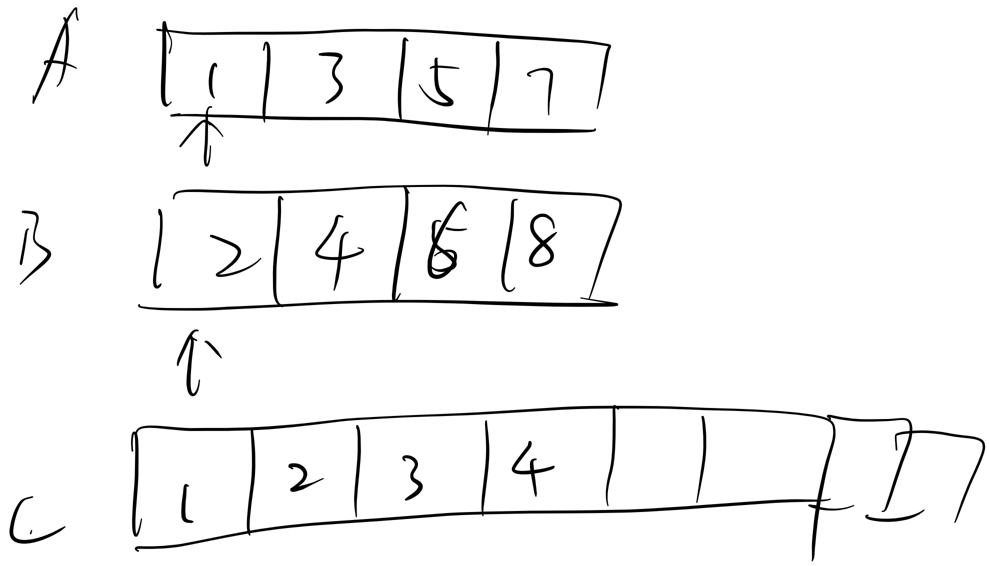
$W = \Theta(n)$

$D = \Theta(\log n)$

Merging Problem

Input: two increasing arrays $A[1, \ldots, n]$ and $B[1, \ldots, n]$
assume that all elements are distinct

Output: merge A and B into a sorted array C.



$A[i]$,

$\quad Rank(i, B) = $ rank of $A[i]$ in $B - 1$

$B[i]$

$\quad Rank(i, A) = $ rank of $B[i]$ in $A - 1$

for i, $1 \le i \le n$ pardo

$\quad C[i + Rank(i,B)] = A[i]$

$\quad C[i + Rank(i,A)] = B[i]$

$W = \Theta(n)$

$D = O(1)$

Serial Ranking

$\quad i = j = 1$

$\quad$ while $i \le n$ and $j \le n$

$\qquad$ if $A[i] < B[j]$

$\qquad\quad Rank(i, B) = j - 1$

$\qquad\quad i = i + 1$

$W = \Theta(n)$

$D = W = \Theta(n)$

else

$\quad Rank(j, A) = i-1$

$\quad j = j+1$

Binary Search

for i, $1 \le i \le n$ pardo

$\quad Rank(i, B) = BS(A[i], B)$

$\quad Rank(i, A) = BS(B[i], A)$

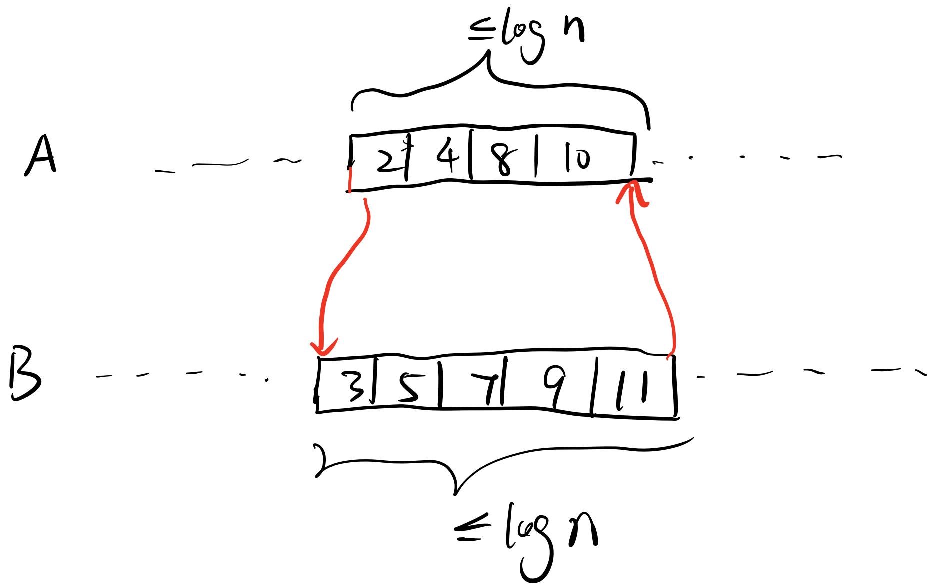
$W = O(n \log n)$

$D = O(\log n)$

Parallel Ranking

$W = O(n)$

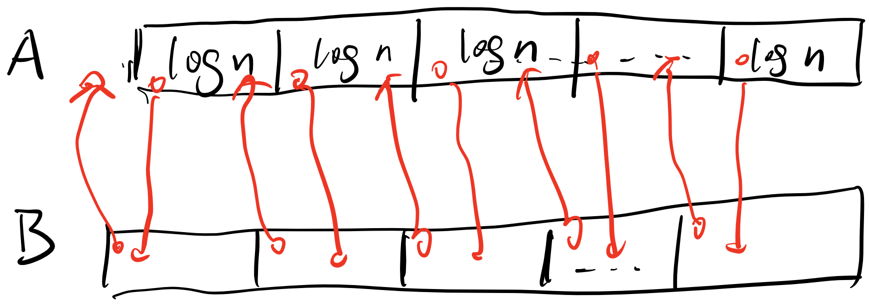
$D = O(\log n)$



$W = O(\# \text{ elements})$

$\Rightarrow O(\# \text{ elements}) = O(\log n)$

$D = O(\#\text{elements}) = O(\log n)$



$k = \frac{n}{\log n}$

Step 1: $W = \frac{n}{\log n} \log n = O(n)$

$D = O(\log n)$

Step 2: serial rank for each group.

$W = O(n) = O(n)$

$D = O(\log n)$

maximum finding

Input: $A(0), \cdots, A(n)$

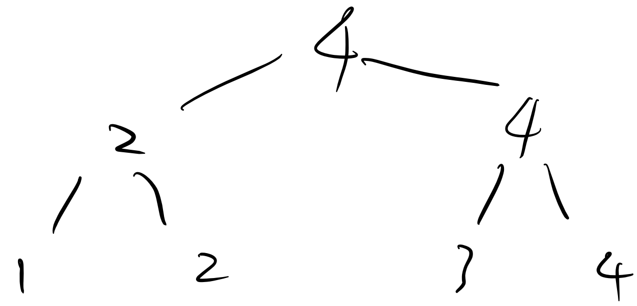
Output: maximum $A(i)$

A0 serial

$W = O(n)$

$D = O(n)$

A1 as summation '+' → 'max'



$W = \Theta(n)$
$D = O(\log n)$

A2

for i, $1 \le i \le n$ pardo
  $B(i) = 0$
for (i, j), $1 \le i < j \le n$ pardo
  if $A(i) < A(j)$:
    $B(i) = 1$
  else
    $B(j) = 1$
for i, $1 \le i \le n$ pardo
  if $B(i) == 0$
    $A(i)$ is the maximum

$W = \Theta(n^2)$
$D = O(1)$

A3. partition in $\sqrt{n}$ group

$G_1 = \{A(1), \ldots, A(\sqrt{n})\} \rightarrow M_1$
$G_2 = \{A(\sqrt{n}+1), \ldots, A(2\sqrt{n})\} \rightarrow M_2$

$\vdots$

$G_{\sqrt{n}} = \{A(n-\sqrt{n}+1), \ldots\ldots, A(n)\} \rightarrow M_{\sqrt{n}}$

$M_1, M_2, \ldots\ldots M_{\sqrt{n}} \rightarrow \max$

$W = O(n)$
$D = O(1)$

$W(n) = O(n) + \sqrt{n} \cdot W(\sqrt{n})$
$D(n) = O(1) + D(\sqrt{n})$

$\Downarrow$

$\begin{cases} W(n) = O(n \log\log n) \\ D(n) = O(\log\log n) \end{cases}$

A_4 partition in $\frac{n}{h}$ groups where $h = \log\log n$

$G_1 = \{A_1, \ldots\ldots A_h\} \rightarrow M_1$
$G_2 = \qquad \rightarrow M_2$

$\vdots$

$G_{\frac{n}{h}} = \{A_{(\frac{n}{h}-1)h+1}, \ldots A_n\} \rightarrow M_{\frac{n}{h}}$

$W = h \cdot \frac{n}{h} = O(n)$

$D = O(h)$

$$M_1, M_2, \ldots, M_{\frac{n}{h}}$$

$\downarrow$ run A3

max

$h = \log\log n$

$$W = O\left(\frac{n}{\cancel{h}} \cdot \cancel{\log\log\frac{n}{h}}\right) = O(n)$$

$$D = O\left(\log\log\frac{n}{h}\right) = O(\log\log n)$$

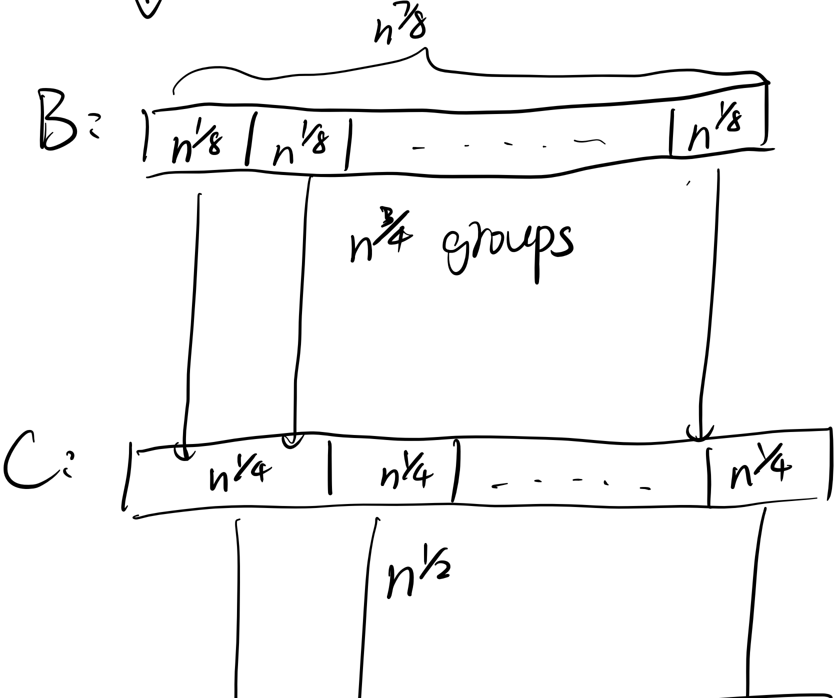
A5 random sampling

In $W = O(n)$, $D = O(1)$, return the maximum with high probility $1 - \frac{1}{n^c}$

A: n elements

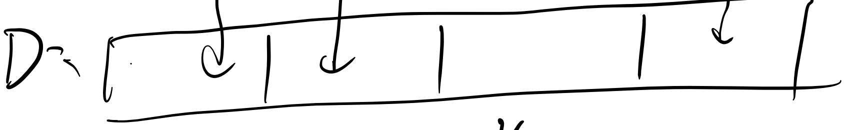
$\downarrow$ random semple $n^{\frac{7}{8}}$ elements from A.

$W = O(n^{\frac{7}{8}})$
$D = O(1)$



$W = n^{\frac{1}{4}} \cdot n^{\frac{3}{4}} = O(n)$
$D = O(1)$

$W = n^{1/2} \cdot n^{1/2} = O(n)$
$D = O(1)$



$n^{1/2}$ element

$\downarrow$

Maximum M

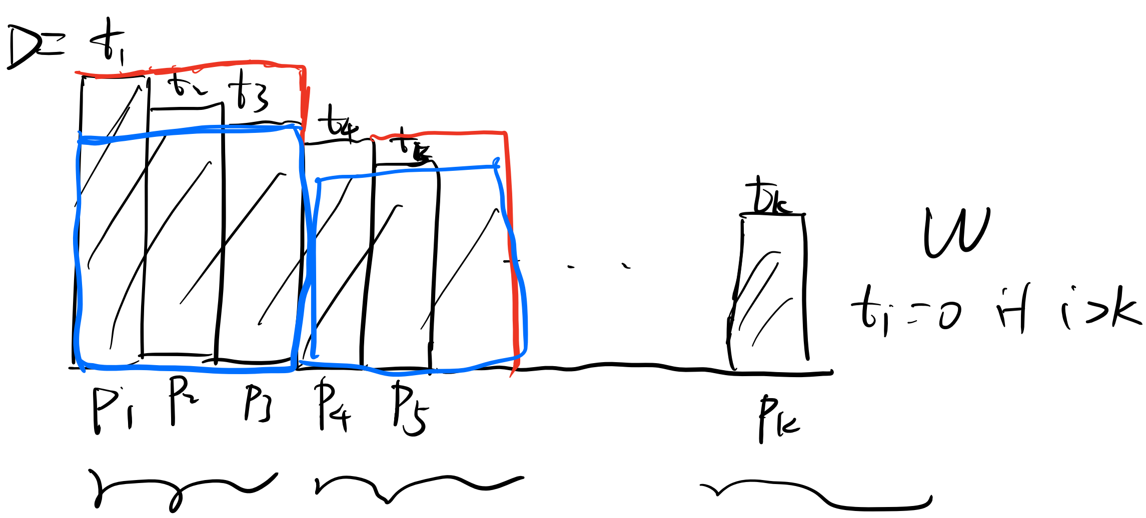
$W = O(n)$
$D = O(1)$

for i, $1 \le i \le n^{7/8}$ pardo — $W = O(n^{\frac{7}{8}})$
    create $B[i]$ — $D = O(1)$

for i, $1 \le i \le n$. pardo — $W = O(n)$
    if $A[i] > M$, — $D = O(1)$
        put $A[i]$ in a random place in $B[j]$

find the maximum of B — $W = O(n)$
$D = O(1)$

$W = O(n)$
$D = O(1)$



p processors

$$T = t_1 + t_{p+1} + t_{2p+1} + \cdots + t_{(g-1)p+1}$$

$$pT = t_1 p + t_{p+1} p + t_{2p+1} p + \cdots + t_{(g-1)p+1}$$

$$\leq \sum_{i=1}^{g} \left(t_{ip} + t_{(i-1)p+1} - t_{ip+1} \right) p$$

$$\leq \sum_{i=1}^{g} t_{ip} \cdot p + p \sum_{i=1}^{g} \left(t_{(i-1)p+1} - t_{ip+1} \right)$$

$$\leq W + p\left(t_1 - t_{gp+1}\right)$$

$$t_1 \to D \qquad t_{gp+1} \to 0$$

$$\leq W + pD$$