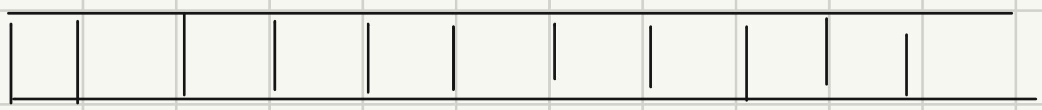
$a+b$ $\quad \underbrace{(\log a + \log b)}_{\downarrow}$ or $\underbrace{O(1)}_{\downarrow}$

Turing Machine $\qquad$ RAM (Random Access Machine)

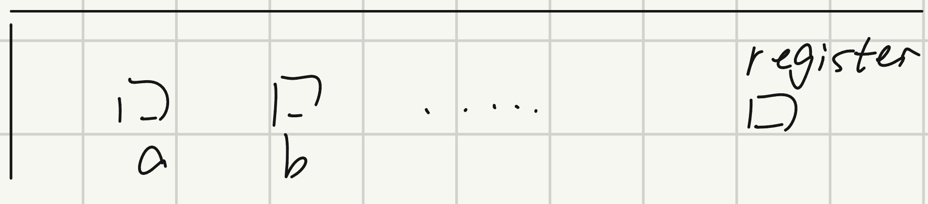
RAM

Memory : an infinite sequence of cells → store integer



address 1 2 3 4 5 . . .

CPU



CPU has 4 atomic operations

1. init register

   $a=1 \qquad a=b$

2. arithmetic

   $c = a+b \qquad - \qquad * \qquad /$

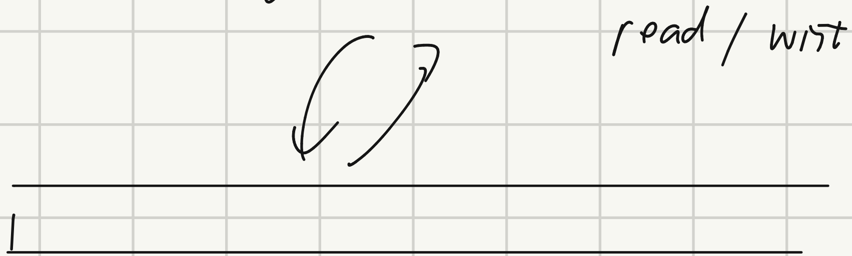
3. comparison

   $a<b$? $\qquad a>b$? $\qquad a=b$?

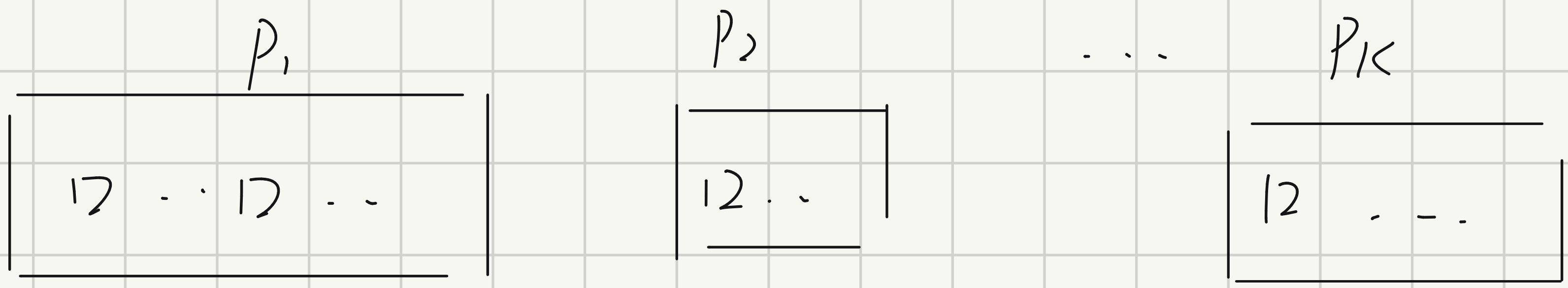
4. memory access

   a

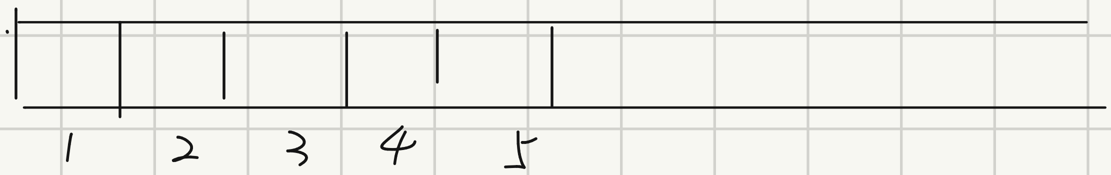
   read / write



PRAM

processors


↳ share the same memory, may cause conflict

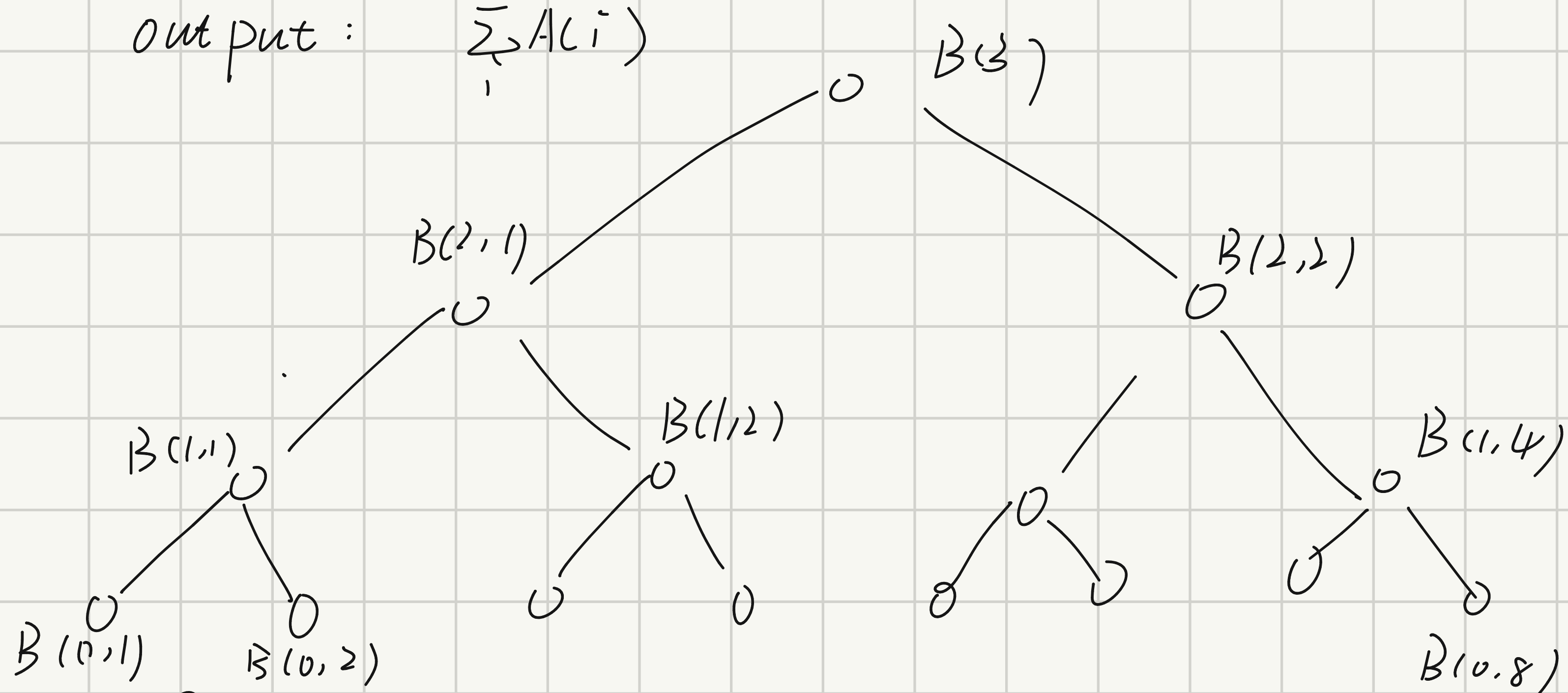


1. CREW (concurrent read and exclusive write)
2. EREW
3. CRCW

Summotion

Input : $A(1)$ . . . . $A(n)$

output : $\sum_i A(i)$



$n = 8$

for i, $1 \le i \le n$ pardo
    $B(0, i) = A(i)$
for $h = 1$ to $\log_2 n$
    for i, $1 \le i \le \frac{n}{2^i}$ pardo
        $B(h, i) = B(h-1, 2i-1) + B(h-1, 2i)$
return $B(\log_2 n, 1)$

$T_p(n)$: running time with p processors on input of size n,
$T_1(n) = O(n)$
→ called "work" W, : total amount of atomic operation.

$T_\infty(n) = O(\log n)$
→ called "Depth" D, length of the longest chain of sequential
dependences
(how parallel the alg is)

$T_p(n)$ for arbitrary p.

$\max\left(D, \frac{W}{p}\right) < T_p(n)$

Brent's theorm
$$T_p(n) \le \frac{W}{p} + D.$$

Proof : $\sum_i g_i = W$

g_1 . - - - —- g_D

no dependences inside unique g_i

So

$\lceil \frac{g_i}{p} \rceil$ $\quad\quad$ $\lceil \frac{g_i}{p} \rceil$

$$\therefore T_p(n) = \sum_{i=1}^{D} \lceil \frac{g_i}{p} \rceil \leq \sum_{i=1}^{D} \left(\frac{g_i}{p} + 1 \right)$$

$$= \frac{W}{p} + D.$$

$A_1 \quad W_1 \quad D_1$

$A_2 \quad W_2 \quad D_2$

1. run A_1
2. run A_2

$W = W_1 + W_2$

$D = D_1 + D_2$

1, 2 : for i $1 \leq i \leq 2$ pardo

A_i

$W = W_1 + W_2$

$D = \max\{D_1, D_2\}$

Prefix Sum

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

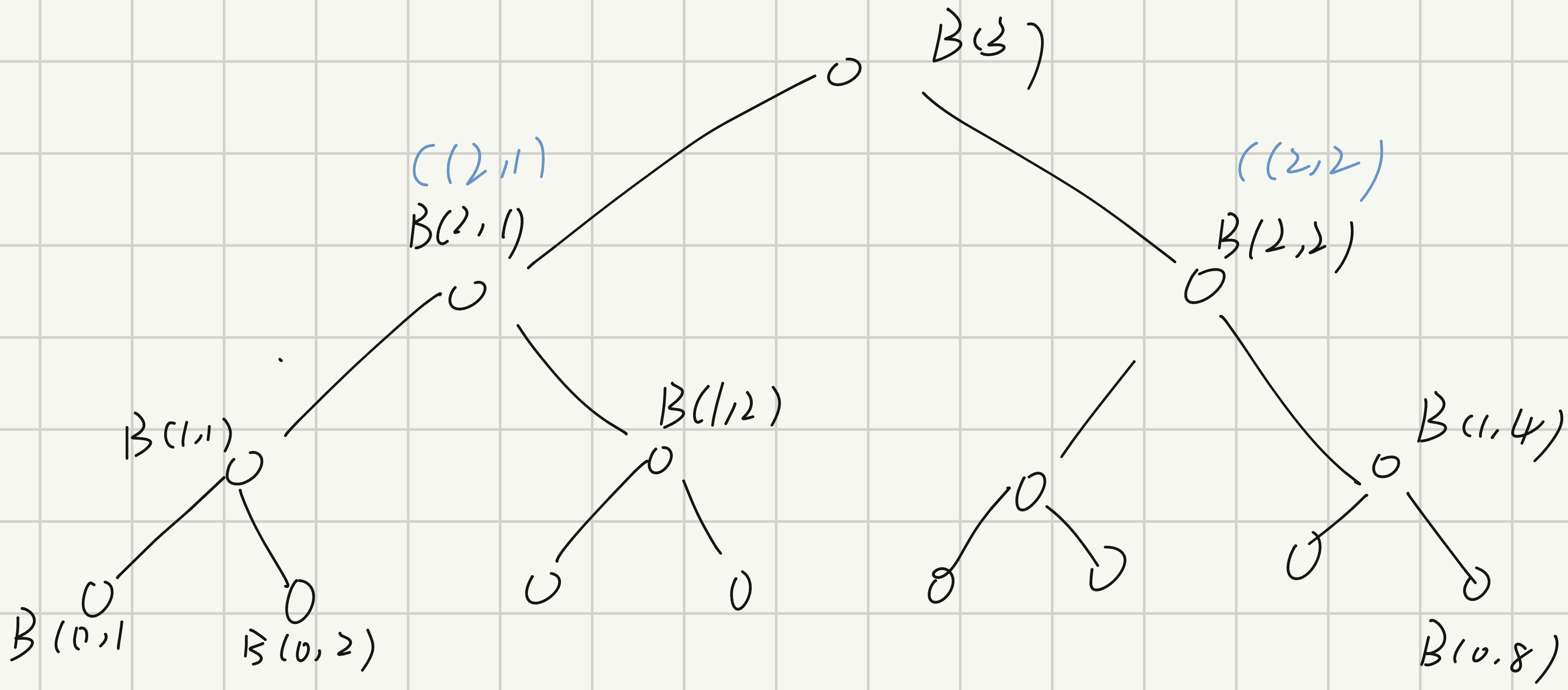
Output: $\sum_{i=1}^{1} A(i), \quad \sum_{i=1}^{2} A(i), \ldots, \sum_{i=1}^{n} A(i)$

Serial: $W = O(n)$

$D = O(n)$

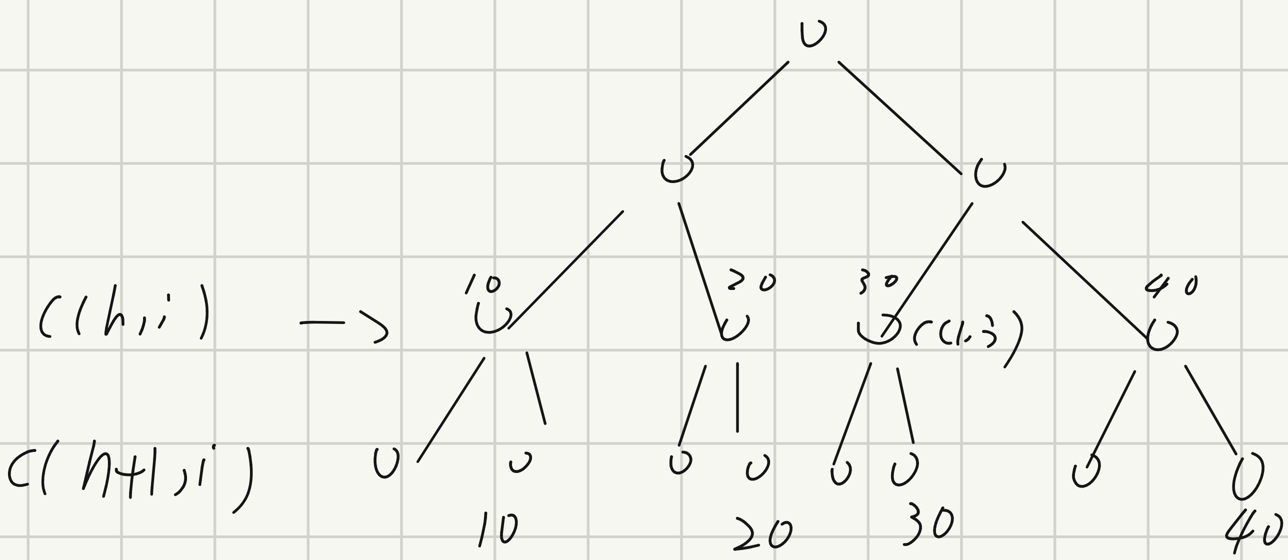
Naive: $W = \sum_{j=1}^{n} O(j) = O(n^2)$

$$D = \max \log_i = O(\log n)$$



$C(h,i) = \sum_{j=1}^{d} A(j)$ $A(d)$ is the rightest leaf of the subtree rooted $C(h,i)$

Goal : $C(0,1)$ $C(0,2)$ ...., $C(0,n)$



if $C(h+1, i)$ is a left child

$C(h+1, i) = C(h, \frac{i-1}{2}) + B(h+1, i)$

↓ the node left to its parent

remark : if $i == 1$

$$c(h+1, i) = B(h+1, i)$$

if $c(h+1, i)$ is a right child

$$c(h+1, i) = C(h, i/2)$$

$W_B = O(n)$

$D_B = O(\log n)$

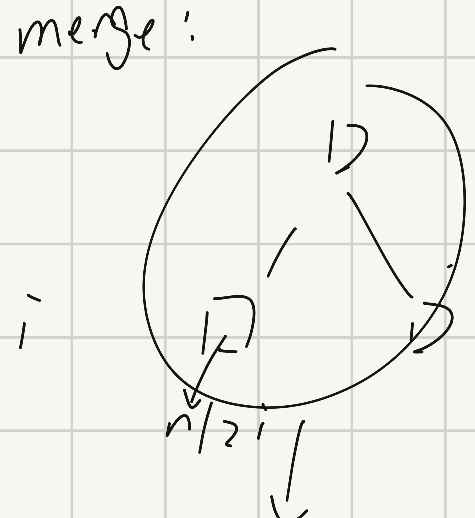
$W_C = O(n)$

$D(C) = O(\log n)$

$\{B, C\} \Rightarrow$ $W = O(n)$, $D = \log(n)$

Parallel merge sort

merge :



12 12

this operation in circle $\begin{cases} W = O(\frac{n}{2^i}) \\ D = O(\frac{n}{2^i}) \end{cases}$

$\Rightarrow D_i = \log \frac{n}{2^i} = \log n - i$

$\sum_i D_i = O(\log^2 n)$

level i $\begin{cases} W_{total} = O(n) \\ D_{total} = O(\frac{n}{2^i}) \end{cases}$

$W = \sum_i W_i = O(n \log n)$

$D = \sum_i W_i = O(n)$

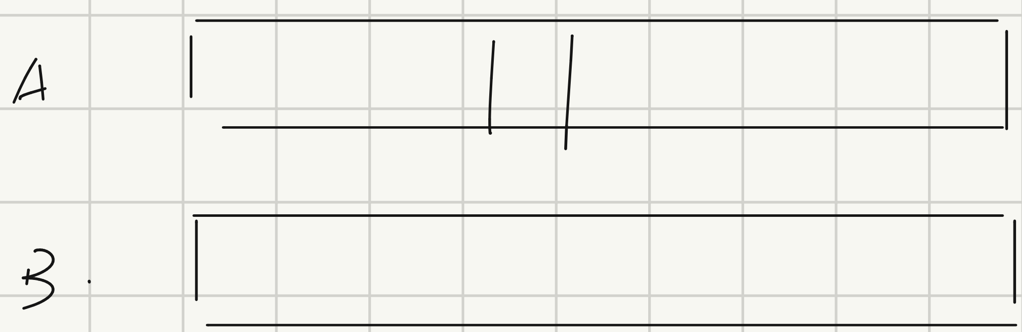
Merge:

Input: Sorted array A and B.

Output: a sorted array C

serial: $W = O(n)$

$D = O(n)$



$rank(i, B)$: rank of $A[i]$ in B.

$rank(i, A)$: rank of $B[i]$ in A.

(assume no duplicate numbers)

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

$C[i + rank(i, B)] = A[i]$ $\qquad W = O(n)$

$C[i + rank(i, A)] = B[i]$ $\qquad D = O(1)$

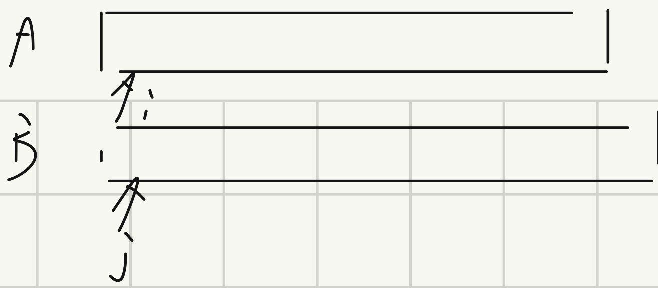
then how to get rank? $\downarrow$ using parallel: $W = O(n)$

$D = O(\log n)$

Ranking:

Output: $Rank(i, B)$ and $Rank(i, A)$ for all i

1. serial ranking



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

$rank(i, B) = j$

$i{+}{+}$

if $A[i] > B[j]$

$rank(j, A) = i$

$j{+}1$

$W = O(n)$

$D = O(n)$

2. binary Search

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

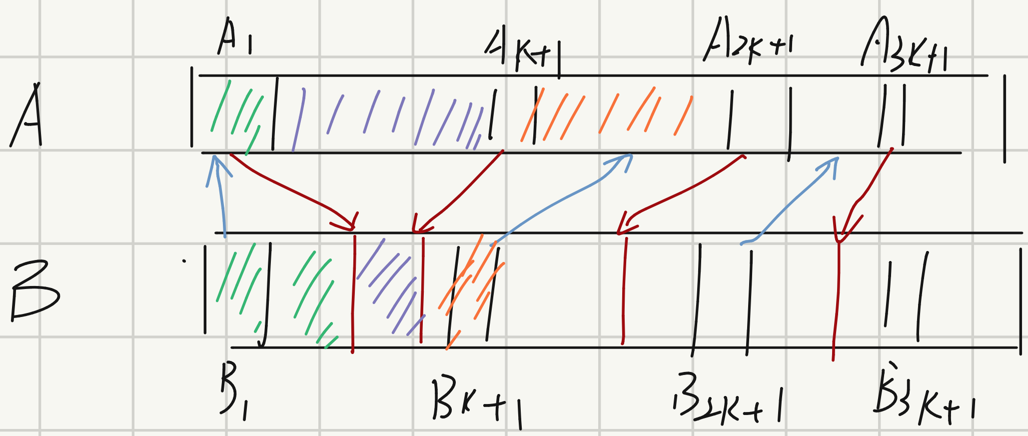
$rank(i, B) = BS(A[i], B)$

$rank(i, A) = BS(B[i], A)$

$W = O(n \log n)$

$D = O(\log n)$

3 parallel ranking



each color represents a group. order between groups is known

max length of a group is $2k$. otherwise has cross

1) using binary search ranking on selected entries

2) serial ranking in each group

$\Rightarrow$ 1): $W_1 = O\left(\frac{2n}{k} \cdot \log n\right)$ comparison from A_{k+1} to all in B.

$D_1 = O(\log n)$

$\Rightarrow$ 2) $W_2 = O(n)$

$D_2 = O(k)$

total $W = W_1 + W_2 = O(\frac{n}{k}\log n + n)$

$D = D_1 + D_2 = O(\log n + k)$

$k = \log n \Rightarrow W = O(n)$

$D = O(\log n)$.

Maximum finding

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

Output: $\max A(i)$

0. serial $W = O(n)$ $D = O(n)$

1. use the summation alg ($+ \to \max$)

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

2. compare all pairs

```
for i  1 ≤ i ≤ n  pardo.
    B(i) = 0
for every pair (i, j) with i<j  pardo
    if A[i] < A[j]
        B[i] = 1
    else
        B[j] = 1
for i,  1 ≤ i ≤ n  pardo
    if B[i] == 0
```

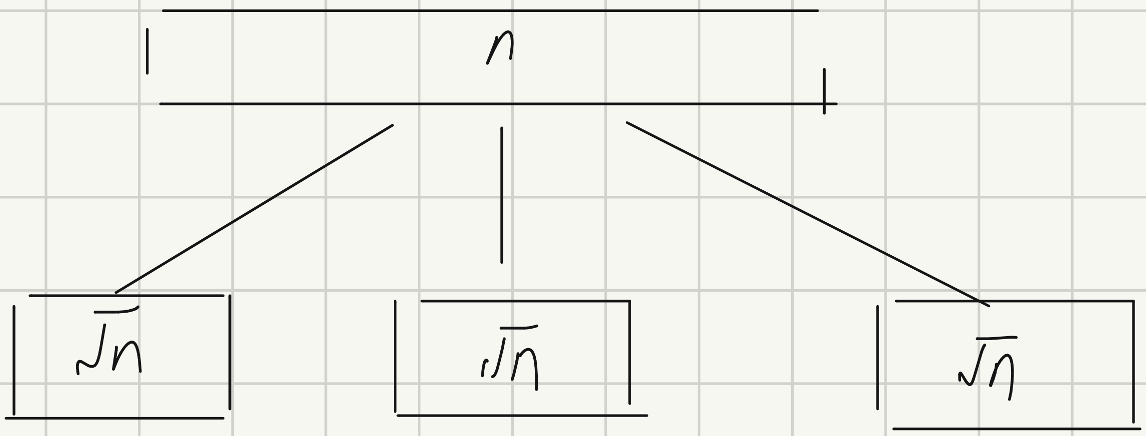
use CRCW.:
common CRCW,
if the value to be written the same, then allow writing.

A[i] is the maximum

$W = O(n^2)$

$D = O(1)$

3 Divide - and - conquer



1.) recursively solve $\sqrt{n}$ subproblems.

2.) find the maximum among the $\sqrt{n}$ numbers by comparing all pair.

$\to O(\sqrt{n}^2)$

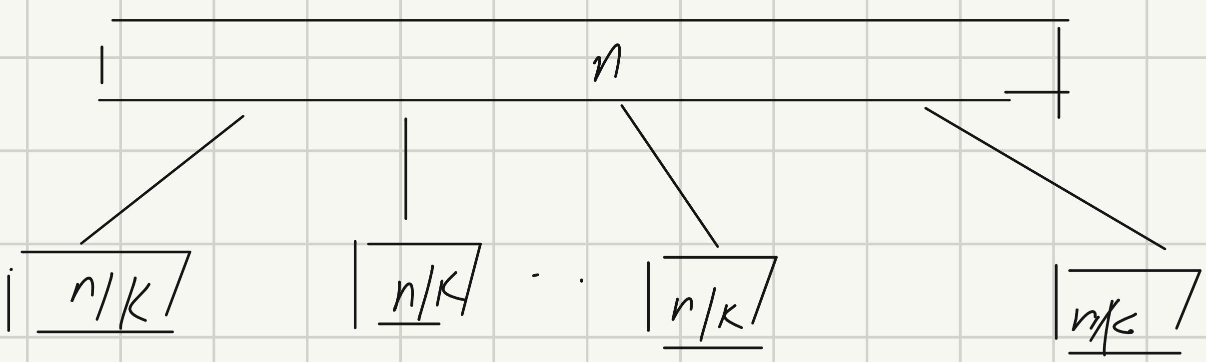
$W(n) = \sqrt{n}\, W(\sqrt{n}) + O(n)$

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

$\Rightarrow W(n) = O(n \log\log n)$

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

4



between groups: parallel

in groups : sequencial

1.) solve subproblems using serial ranking

$W_1 = O(n)$

$D_1 = O(n/K)$

2) find the maximum among the K numbers using D & C

$W_2 = O(K \log\log K)$

$D_2 = O(\log\log K)$

total = $W = O(n + K\log\log K)$

$D = O(n/K + \log\log K)$

let $K = \frac{n}{\log\log n}$

$\Rightarrow W = O(n)$

$D = (\log\log n)$

5 Random sampling

$W = O(n)$

$D = O(1)$

with high probability $1 - \frac{1}{n^c}$

return maximum

A $\qquad |A| = n$

↓ random sample

B $\qquad |B| = n^{\frac{7}{8}}$

↓ partition into $n^{\frac{3}{4}}$ group. $\qquad W = O(n^{\frac{7}{8}})$

$D = O(1)$

. . . . .

$n^{\frac{3}{4}}$ groups. each of size: $n^{\frac{1}{8}}$.

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

$D = O(1)$

$\downarrow$ find the maximum of each group by comparing all pairs

C

$|C| = n^{\frac{3}{4}}$

$\downarrow$ partition $n^{\frac{1}{2}}$ groups each of size $n^{\frac{1}{4}}$

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

$D = O(1)$

$\downarrow$

D

$|D| = n^{\frac{1}{2}}$

$\downarrow$ finding the max

$W = O(n)$

$D = O(1)$

rank: sample

for i $1 \le i \le n^{\frac{7}{8}}$ pardo

$B[i]$ = random select from A

---

round 2.

for i, $1 \le i \le n^{\frac{7}{8}}$ pardo

$B[i]$

for i $1 \le i \le$ pardo

if $A[i] > m$,

throw $A[i]$ into a random place of B

find a maximum of B

$W = O(n)$

$D = O(1)$

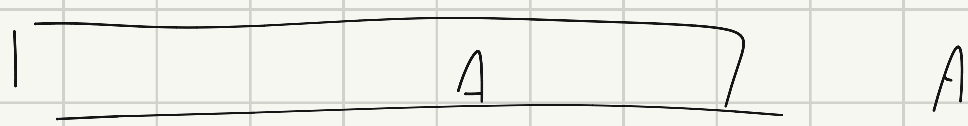
$\text{rank}(m) \le n^{\frac{1}{4}}$ and all $A[i] > m$ was throw in different places

$\Downarrow$

success

$$\Pr(\text{success}) \geq \Pr(E_1 \cap E_2)$$

$$\geq \Pr(E_1) \cdot \Pr(E_2 \mid E_1)$$



$$\frac{n^{\frac{1}{4}}}{n} = \frac{1}{n^{\frac{3}{4}}}$$

$$\Pr(E_1) \geq 1 - \left(1 - \frac{1}{n^{\frac{3}{4}}}\right)^{n^{\frac{7}{8}}}$$

$$\geq 1 - \left(1 - \frac{1}{n^{\frac{3}{4}}}\right)^{n^{\frac{1}{4}} \cdot n^{\frac{1}{8}}}$$

$$\geq 1 - \left(e^{-n^{\frac{3}{4}}}\right)^{n^{\frac{1}{8}}}$$

$$\geq 1 - e^{-n^{\frac{5}{8}}}$$

not request

$$\Pr(E_2 \mid E_1)$$