Adv Algo Assignment 4 ①
Roll: 2020701018
SRIKAR KALE

1) Consider the balanced binary tree approach for finding the prefix sum of an array of n elements. Does this run on an EREW model? Whats the asymptotic time & work complexity of the algo in the EREW model?

Ans) Yes, the balanced binary tree approach for finding the prefix sum of an array of n elements can be run in the EREW PRAM model. Let us consider the algorithm

ALGORITHM PREFIX Sums (Recursive approach):

```
I/p : Array of $n = 2^k$ elements $k \geq 0$, $(x_1, x_2, \dots x_n)$
O/p : The prefix sums $S_i$ for $1 \leq i \leq n$
begin
  1) If $n = 1$ then {set $S_1 = x_1$ ; exit}
  2) for $1 \leq i \leq n/2$ par do
        set $y_i = x_{2i-1} * x_{2i}$
  3) Recursively, compute the prefix sums of $\{y_1, y_2, \dots y_{n/2}\}$
and store them in $z_1, z_2, \dots z_{n/2}$
  4) for $1 \leq i \leq n$ par do
     { i even    : set $S_i = z_{i/2}$
       i = 1     : set $S_1 = x_1$
       i odd > 1 : set $S_i = z_{(i-1)/2} * x_i$
end.
```

→ Since steps 1, 2 & 4 of the above algo do not require concurrent read or write capability, this algo runs on the EREW model.

$T(n) = T(n/2) + a$ $\qquad$ $W(n) = W(n/2) + bn$

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

3) What would be the no of processors & work complexity of the parallel search algorithm when we require that the run time is in $O(\log\log n)$

Ans) $T(n) = O(\log_p n)$ ⎫ For parallel search, where
$W(n) = O(p \log_p n)$ ⎭ p = no of processors used.

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

$\Rightarrow c_1 \cdot \frac{\log n}{\log p} = c_2 \log\log n$ $\quad c_1, c_2 \to$ const

$\Rightarrow \log p = c_3 \frac{\log n}{\log \cdot \log n}$ $\quad c_3 \to$ const.

$\Rightarrow p = e^{c_3 \cdot \log n / \log\log n} = n^{c_3/\log\log n}$

taking natural log everywhere for easier calculation (without loss of generality)

$\therefore p = O(n^{1/\log\log n})$

$$W(n) = O\left(n^{c_3/\log\log n} \cdot \frac{\cancel{\log n}}{\frac{1}{\log \cdot \log n} \cdot c_3 \cancel{\log} n}\right)$$

$\Rightarrow W(n) = O\left(n^{1/\log\log n} \cdot \log \cdot \log n\right)$

2) Recall the merging problem discussed in class that has a time of $O(\log \log n)$ and a work of $O(n \log \log n)$. Complete the steps of arriving at an optimal $O(\log \log n)$ time merging algo in the CREW PRAM model

Ans) Old Merging problem:

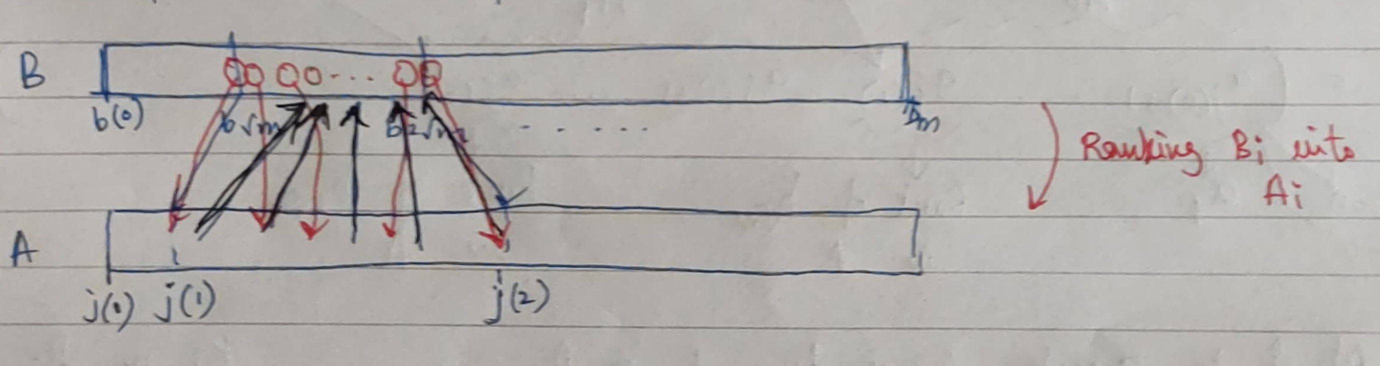
Size of a partition	: $\log n$
# partitions	: $n/\log n$
Bin Search	: $O(\log n)$ time
Seq Merge	: $O(\log n)$ time
$T(n)$	$= O(\log n)$
$W(n)$	$= O\left(\frac{n}{\log n} \cdot \log n\right) = O(n)$

} optimal

In the new partition strategy,

We want to merge 2 sorted arrays A & B where we rank $\sqrt{m}$ elem of B, that partition B into blocks of almost equal lengths, in the sorted seq A.

→ The computed ranks of the chosen elements will induce a partition on A into blocks (size unknown) such that each block of A has to fit b/w 2 B's chosen elements



Elements b/w $j(i)$ & $j(i+1)$ must be b/w $b_{i\sqrt{m}}, b_{(i+1)\sqrt{m}}$

→ Overall problem is reduced to ranking elements of each block of B into corresponding block of A (No more seq merge required).

→ Overall lets consider $|A| = n$, $|B| = m$

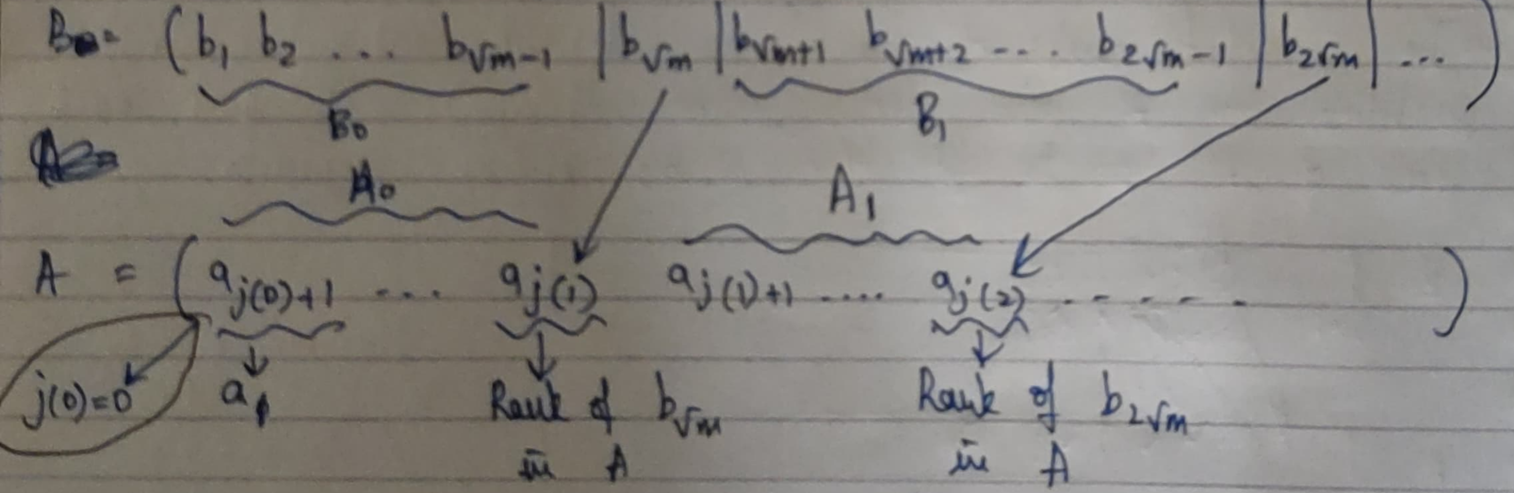
→ Concurrently rank $b_{\sqrt{m}}, b_{2\sqrt{m}}, b_{3\sqrt{m}}, \ldots b_{i\sqrt{m}}, \ldots b_m$ in A using parallel search algo. Where the no of processors are $\sqrt{n}$

$T(n)$, when $p = \sqrt{n}$ - $O(1)$ time

Let Rank $(b_{i\sqrt{m}}, A) = j(i)$ $\quad 1 \le i \le \sqrt{m}$

$j(0) = 0$ $\longrightarrow$ boundary condition.

For $0 \le i \le \sqrt{m} - 1$ Let $B_i = (b_{i\sqrt{m}+1} \ldots b_{(i+1)\sqrt{m}-1})$
and $A_i = (a_{j(i)+1} \ldots a_{j(i+1)})$



$B = (b_1\ b_2 \ldots b_{\sqrt{m}-1} \mid b_{\sqrt{m}} \mid b_{\sqrt{m}+1}\ b_{\sqrt{m}+2} \ldots b_{2\sqrt{m}-1} \mid b_{2\sqrt{m}} \mid \ldots)$
B_0
B_1
A_0
A_1
$A = (a_{j(0)+1} \ldots a_{j(1)}\ a_{j(1)+1} \ldots a_{j(2)} \ldots)$
$j(0)=0$
a_1
Rank of $b_{\sqrt{m}}$ in A
Rank of $b_{2\sqrt{m}}$ in A

Need to rank B_0 in A_0, B_1 in A_1 ... B_i in A_i

$$\rightarrow \text{If } \underbrace{\boxed{j(i) = j(i+1)}}_{\text{Exit condition of recurrence}}, \text{ then rank } (B_i : A_i) = (0, 0, \ldots 0)$$

else compute recursively rank $(B_i : A_i)$

→ Let $1 \le k \le m$ be any arbitrary index that is not a multiple of $\sqrt{m}$, Let $i = \left\lfloor \frac{k}{\sqrt{m}} \right\rfloor$ then

$$\text{rank}(b_k, A) = \underbrace{j(i)}_{\text{starting index of } b_{i\sqrt{m}}} + \underbrace{\text{rank}(b_k : A_i)}_{\text{Rank of } b_k \text{ in partition } A_i}$$

rank (b_k, B) = index k in B (straight forward)

Analysis:

Searching {
$\sqrt{m}$ calls to parallel search, $\sqrt{n}$ processors
$\Rightarrow T(n) = O(1)$

$$\text{Total no of operations required} = O(\underbrace{\sqrt{m}}_{\text{No of elem in a block } B_i} \cdot \underbrace{\sqrt{n}}_{\text{no of processors}} \cdot \underbrace{1}_{\text{search time}})$$
}

All blocks of B run in parallel

A·M ≥ G·M

$$\Rightarrow \frac{m+n}{2} \ge \sqrt[2]{m \cdot n}$$

$$\Rightarrow O(m+n) \ge O((m \cdot n)^{1/2})$$

$$\rightarrow \; \le O(m+n)$$

Ranking : $T(n, m) = T(n, m^{1/2}) + O(1)$

$$\downarrow$$

$$T(m) = T(m^{1/2}) + O(1)$$

~~$T(\log m) = T(\frac{1}{2}\log m) + O(1)$~~

~~Take $k = \log m$, $T(k) = T(k/2) + O(1)$~~

~~$\Rightarrow T(k) = O(\log k)$~~

~~$\therefore T(m) = O(\log \cdot \log m)$~~

~~$T(2^x) = T(2^{x/2}) + O(1)$~~

~~$2^x = m \Rightarrow x = \log_2 m$~~

~~$T(x) = T(x/2) + O(1)$~~

~~$\Rightarrow T(x) = O(\log x)$~~

~~$= O(\log \cdot \log$~~

$$T(m) = T(m^{1/2}) + O(1)$$
$$= T(m^{1/4}) + O(1) + O(1)$$
$$\vdots$$
$$T(m^{1/2^i}) + i(O(1)$$

Assume, without loss of generality

~~T(m)~~ $m^{1/2^i} \geq 2$

Because $T(n) = 0 \quad \forall n \leq 2$

$$\therefore T(m) = T\left(m^{1/2^i}\right) + i(O(1))$$

$$m^{\frac{1}{2^i}} \geq 2$$

$$m^{\frac{2^i}{2^i}} \geq 2^{2^i}$$

$$\Rightarrow m \geq 2^{2^i} \quad \Rightarrow i \leq \log\log m$$

$$\therefore T(m) = T(2) + \log\log m \cdot O(1)$$

$$= O(\log\log m)$$

$$\therefore \text{Total work done} \leq O\Big(\underbrace{(m+n)}_{\text{work for each search}} \times \underbrace{(\log\log m)}_{\text{Ranking for each search}}\Big)$$

$$\text{Total time} = O(1 * \log\log m) = O(\log\log m)$$

4) Design a parallel algorithm to find the bitwise OR of n i/ps in the CRCW model. What is the runtime & the work complexity of your algorithm. Justify your answer.

Ans) bitwise OR of n i/ps in CRCW is simple.

```
for each processor i (1 ≤ i ≤ n) in parallel,
    if (A[i] = 1) then
                  o/p = A[i]
```

Initialize o/p to be 0.

→ $O(1)$ time on an n processor common CRCW PRAM.

Work done = $n \cdot T(n)$ ⇒ $W(n) = O(n \cdot c) = O(n)$

5) Suppose we are given p processors. Redo the analysis of the prefix sum algorithm to see how p processors can simulate the n processors used in that algorithm. Obtain asymptotic estimates on the time & work complexity as a fn of p & n.

→ Let the run time of a parallel algo using p processors be $T(n,p)$

Then the total work done of a parallel algo is:
$p \cdot T(n,p)$.

→ Divide array A into n/p sub arrays each of which are p sized.

→ perform the prefix sum upward traversal for each of the n/p sub arrays using p processors

→ For each sub array, the value computed during upward traversal is stored in another array C_i, $i = 1, 2, \ldots n/p$.

→ For each sub array, run time $= O(\log p)$ with work $\Theta(p)$.

→ For $i = 1 \ldots n/p$, do

```
    $Z[1] = 0$,
    $Z[i] = C_{i-1}[n/p] + Z[i-1]$
```

→ prefix sums calculated above are local to each subarray

→ This step is sequential, Time taken here $= O(n/p)$

→ This step combines the results of each sub array.

→ Again, for each sub array, perform downward traversal step and o/p the prefix sum using:

If $i == 1$ $\quad S_j[i] = Z[j] + C_j[i]$

If $i ==$ even $\quad S_j[i] = Z[j] + C_j[i/2]$

If $i ==$ odd $\quad S_j[i] = Z[j] + C_j[(i-1)/2]$

i → processor index

j → sub array index.

Run time $T(n, p) = O(n/p + \log p)$

If $n/p > \log p$, then $T(n/p) = O(n/p)$

Work complexity, $W(n) = p \cdot T(n, p)$

$$= O(p \cdot n/p + p \cdot \log p)$$

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

If $n/p > \log p$, then $W(n) \underline{\underline{= O(n)}}$