Search 1

SEARCH-1

Input: A sorted array A of size n and an item x.

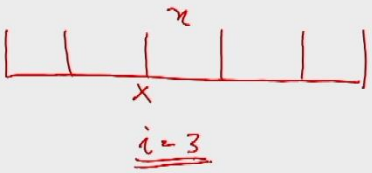
Output: The rank R of x in A. Model: CREW PRAM

```
{
    R = 0;
    pardo for 1 ≤ i ≤ n − 1
        if (A[i] ≤ x  &&  A[i + 1] > x)  R = i;
    return R;
}
```

Search

Sorted array of size n
element x
if x is present in the
array, what is its
rank?

Search 1



SEARCH-1

Input: A sorted array A of size n and an item x.

Output: The rank R of x in A. Model: CREW PRAM

```
{
    R = 0;
    pardo for 1 ≤ i ≤ n − 1
        if (A[i] ≤ x  &&  A[i + 1] > x)  R = i;
    return R;
}
```

n − 1 processors

O(1) time

CREW ?

x is known to all

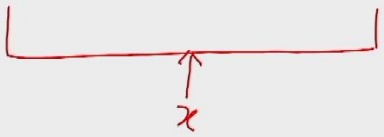
O(log n) time

EREW

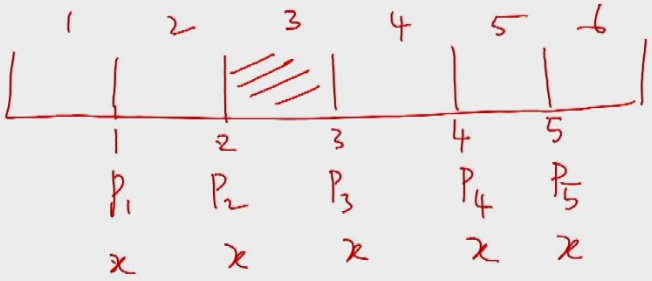
Search 2

$p \leq n-1$ processors

When we have one processor, binary search



$O(\log n)$
1 processors



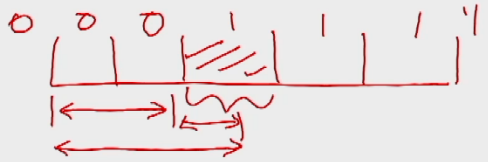
p procrs
$p = 5$

$n \longrightarrow \left\lceil \frac{n}{p+1} \right\rceil$

$O(\log_{p+1} n)$

$O\left(\frac{\log n}{\log (p+1)}\right)$

Search 2



SEARCH-2

Input: A sorted array A, an item x, and p the number of processors.

Output: The rank R of x in A. Model: CREW PRAM

```
{
    if (n ≤ p + 1) return SEARCH-1(A, x);
    pardo for 1 ≤ i ≤ p
        if (x ≥ A[i⌈n/(p+1)⌉])  B[i] = 0;
        else  B[i] = 1;
    B[0] = 0;  B[p + 1] = 1;
    pardo for 0 ≤ i ≤ p
        if (B[i] == 0  &&  B[i + 1] == 1)  R = i + 1;
    return (R − 1)⌈n/(p+1)⌉+ SEARCH-2(A[(R − 1)⌈n/(p+1)⌉ + 1 . . . R⌈n/(p+1)⌉], x);
}
```

Binary search

1 processor $O(\log_{1+1} n)$

$m-1$ processor $O(1)$

p processors $O(\log_{p+1} n)$ time

CREW PRAM

$O(\log p + \log_{p+1} n)$ t

Cross ranking

rank the rank of x in A is the no. of elements in A that are $\leq x$

cross rank A, B:
Find the rank every element of A in B & vice versa

0	0	1	5	5
A	D	G	X	Y

F	J	L	M	N
2	3	3	3	3

Cross Rank

CROSS-RANK-1

Input: Two sorted arrays $A[1 \ldots m]$ and $B[1 \ldots n]$; $n \geq m$.

Output: Arrays $\alpha[1 \ldots n]$ and $\beta[1 \ldots n]$ such that the rank of $A[i]$ in B is $\alpha[i]$ and the rank of $B[i]$ in A is $\beta[i]$. Model: CREW PRAM

{

pardo for $1 \leq i \leq n$

$\alpha[i] = \text{SEARCH-2}(B[1 \ldots m], A[i], 1)$;

pardo for $1 \leq i \leq m$

$\beta[i] = \text{SEARCH-2}(A[1 \ldots n], B[i], 1)$;

}

$O(\log m)$, $O(\log n)$ } $O(\log n)$

Merge of 2 sorted arrays

1	2	4	9	10	1	A
1	2	3	4	5	2	D
0	0	1	5	5	3	F
A	D	G	X	Y	4	G
					5	J
F	J	L	M	N	6	L
					7	M
2	3	3	3	3	8	N
1	2	3	4	5	9	X
3	5	6	7	8	10	Y

Merge

O(log n) time
n+m processors

MERGE-1

Input: Two sorted arrays $A[1 \ldots n]$ and $B[1 \ldots m]$; $n \geq m$.

Output: Array $C[1 \ldots m+n]$, the merge of A and B. Model: CREW PRAM

{

CROSS-RANK-1($A[1 \ldots n], B[1 \ldots m], \alpha[1 \ldots n], \beta[1 \ldots m]$); O(log n)

pardo for $1 \leq i \leq n$

$C[i + \alpha[i]] = A[i]$;

pardo for $1 \leq i \leq m$

$C[i + \beta[i]] = B[i]$;

} O(1) time

}

Merge Sort 1

CREW PRAM

Array A of size n ; n processors

Sort A

Divide A into 2 equal halves

Sort each half recursively

Merge the 2 halves

$$T(n) = T\left(\frac{n}{2}\right) + c \log n$$

$$= T\left(\frac{n}{4}\right) + c \cdot \log \frac{n}{2} + c \log n$$

Put $k = \log n - 1$, $T(2) = 1$

$$= T\left(\frac{n}{8}\right) + c\left[\log \frac{n}{4} + \log \frac{n}{2} + \log n\right]$$

$$\vdots$$

$$= T\left(\frac{n}{2^k}\right) + c\left[\log \frac{n}{2^{k-1}} + \ldots + \log n\right]$$

$\log n$, $\log n - 1$, $\log n - 2$, $\ldots$, 2

$$= T(2) + c\left[\log 4 + \log 8 + \ldots + \log n\right]$$

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

CREW PRAM

Sorting of n elements takes $O(\log^2 n)$ time using n processors

cost : $O(n \log^2 n)$

CREW PRAM

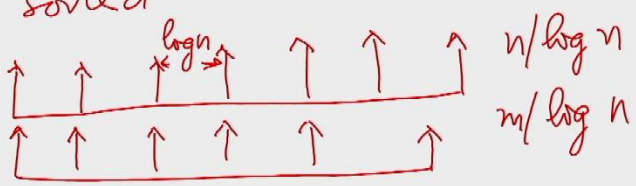
Sorting of n elements takes $O(\log^2 n)$ time using n processors

cost: $O(n \log^2 n)$

An optimal Merge Algorithm CREW PRAM

$\frac{n+m}{\log n}$ processors, $O(\log n)$ time where $n \geq m$.

A, B sorted



the leaders

A' : the array of leaders from A

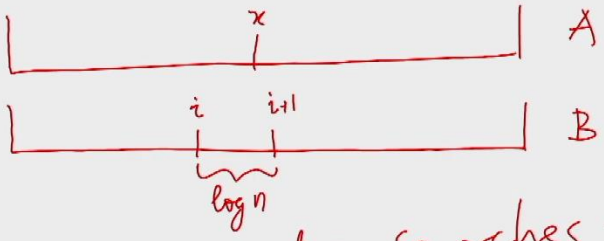
B' : ——— B

Cross rank A' & B'.

$r_1[i]$: rank of $A'[i]$ in B'

$r_2[i]$: ——— $B'[i]$ in A'

each leader knows the segment on the other side, in which it belongs

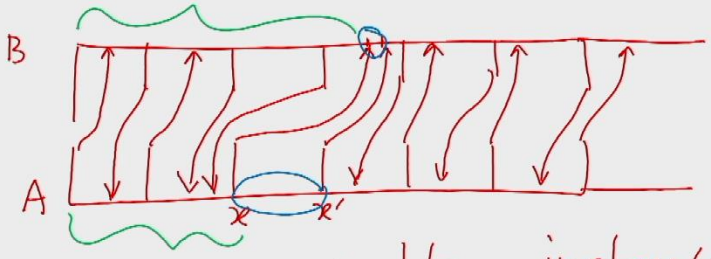


The processor at x searches the segment sequentially

→ rank of x in B

$R_1[i]$: rank of $A'[i]$ in B

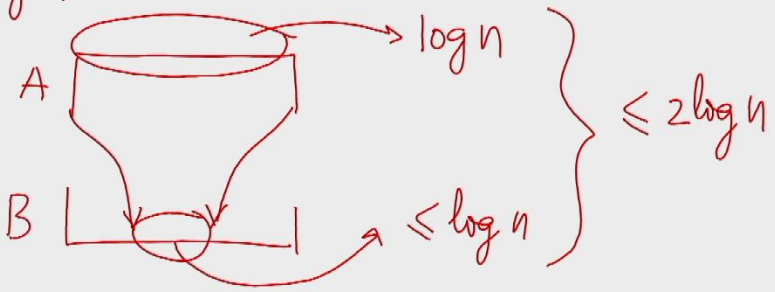
$R_2[i]$: ——— $B'[i]$ in A

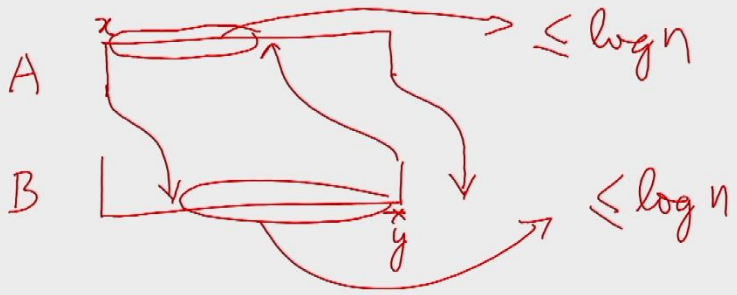


Recursive problem instances

Maximum problem size of the sub instances

$O(\log n)$ at the most

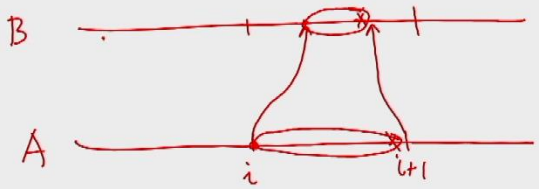




$O(\log n)$

Case 1

$L = \log n$



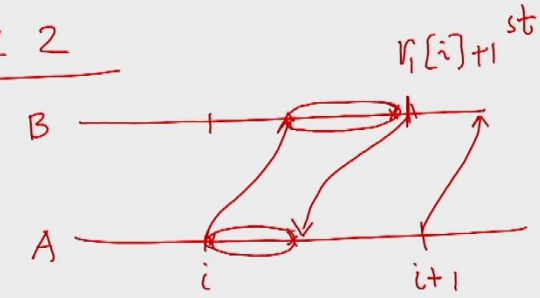
AAleft : $(i-1)L+1$

ABleft : $R_1[i]+1$

AAright : $i * L$

ABright : $R_1[i+1]$

Case 2



$AAleft = (i-1)L + 1$

$ABleft = R_1[i] + 1$

$ABright = r_1[i] * L$

$AAright = R_2[r_1[i]+1]$

Merge (Cont'd)

pardo for $1 \leq i \leq n$

MERGE-SEQ($A[AAleft[i] \ldots AAright[i]], B[ABleft[i] \ldots ABright[i]],$
$C[AAleft[i] + ABleft[i] - 1, \ldots AAright[i] + ABright[i]]$);

pardo for $1 \leq i \leq nBL$

MERGE-SEQ($A[BAleft[i] \ldots BAright[i]], B[BBleft[i] \ldots BBright[i]],$
$C[BAleft[i] + BBleft[i] - 1, \ldots BAright[i] + BBright[i]]$);

}

Merge

MERGE-2

Input: Two sorted arrays $A[1 \ldots N]$ and $B[1 \ldots M]$; $N \geq M$.

Output: Array $C[1 \ldots M+N]$, the merge of A and B. Model: CREW PRAM

{

$L = \lceil \log MN \rceil; \quad n = \lceil N/L \rceil; \quad m = \lceil M/L \rceil;$

pardo for $1 \leq i \leq n \quad A'[i] = A[(i-1) * L + 1]$;

pardo for $1 \leq i \leq m \quad B'[i] = B[(i-1) * L + 1]$;

} O(1) time

CROSS-RANK-1$(A'[1 \ldots n], B'[1 \ldots m], r_1[1 \ldots n], r_2[1 \ldots m])$; O(log n)

pardo for $1 \leq i \leq n$

$R_1[i] = (r_1[i]-1) * L +$ SEARCH-2$(B[(r_1[i]-1) * L + 1 \ldots r_1[i] * L], A'[i], 1)$;

pardo for $1 \leq i \leq m$

$R_2[i] = (r_2[i]-1) * L +$ SEARCH-2$(A[(r_2[i]-1) * L + 1 \ldots r_2[i] * L], B'[i], 1)$;

} O(log n)

pardo for $1 \leq i \leq n$

{

$AAleft[i] = (i-1) * L + 1; \quad ABleft[i] = R_1[i] + 1$;

if $(r_1[i] == r_1[i+1])$ { $AAright[i] = i * L; \quad ABright[i] = R_1[i+1]$; }

else { $ABright[i] = r_1[i] * L; \quad AAright[i] = R_2[r_1[i]+1]$; }

}

} O(1)

pardo for $1 \leq i \leq m$

{

$BBleft[i] = (i-1) * L + 1; \quad BAleft[i] = R_2[i] + 1$;

if $(r_2[i] == r_2[i+1])$ { $BBright[i] = i * L; \quad BAright[i] = R_2[i+1]$; }

else { $BAright[i] = r_2[i] * L; \quad BBright[i] = R_1[r_2[i]+1]$; }

}

} O(1)

Merge (Cont'd)

$O(\log n)$

pardo for $1 \leq i \leq n$

MERGE-SEQ($A[AAleft[i] \ldots AAright[i]], B[ABleft[i] \ldots ABright[i]],$
$C[AAleft[i] + ABleft[i] - 1, \ldots AAright[i] + ABright[i]]$);

pardo for $1 \leq i \leq$ ~~nBL~~ m

MERGE-SEQ($A[BAleft[i] \ldots BAright[i]], B[BBleft[i] \ldots BBright[i]],$ $O(\log n)$
$C[BAleft[i] + BBleft[i] - 1, \ldots BAright[i] + BBright[i]]$);

}

Algorithm runs in $O(\log n)$ time

$\frac{n+m}{\log n}$ processors

CREW PRAM