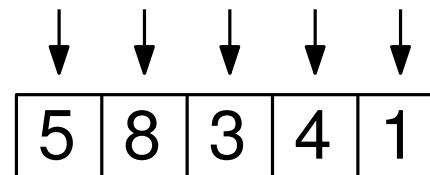
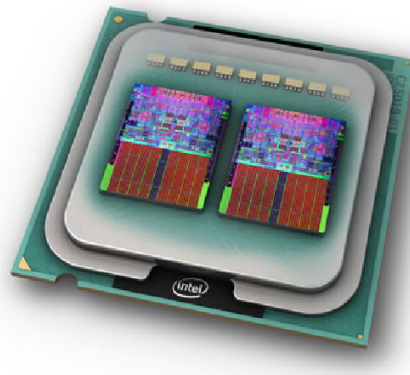




# ICS 443: Parallel Algorithms

Prof. Nodari Sitchinava

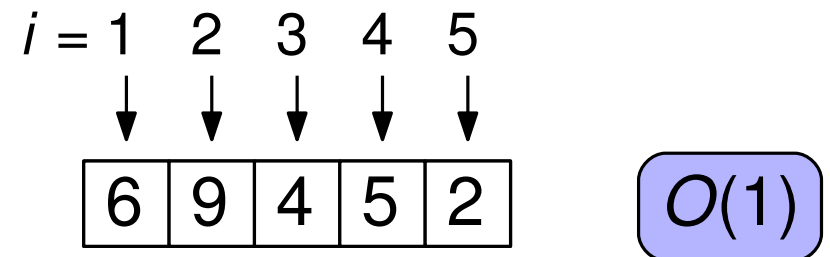
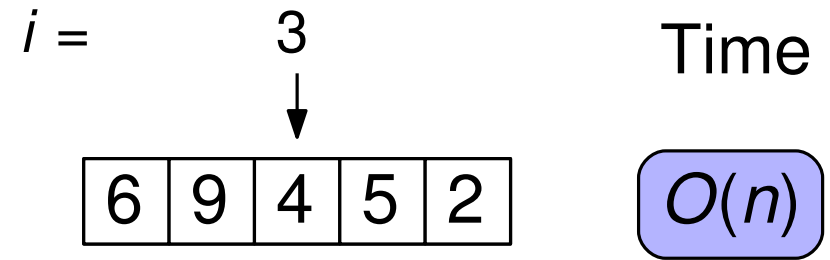


## Lecture 2: Models of Parallel Computation

# Example From Last Lecture

**for**  $i = 1$  to  $n$  **do**  
     $a[i] = a[i] + 1$

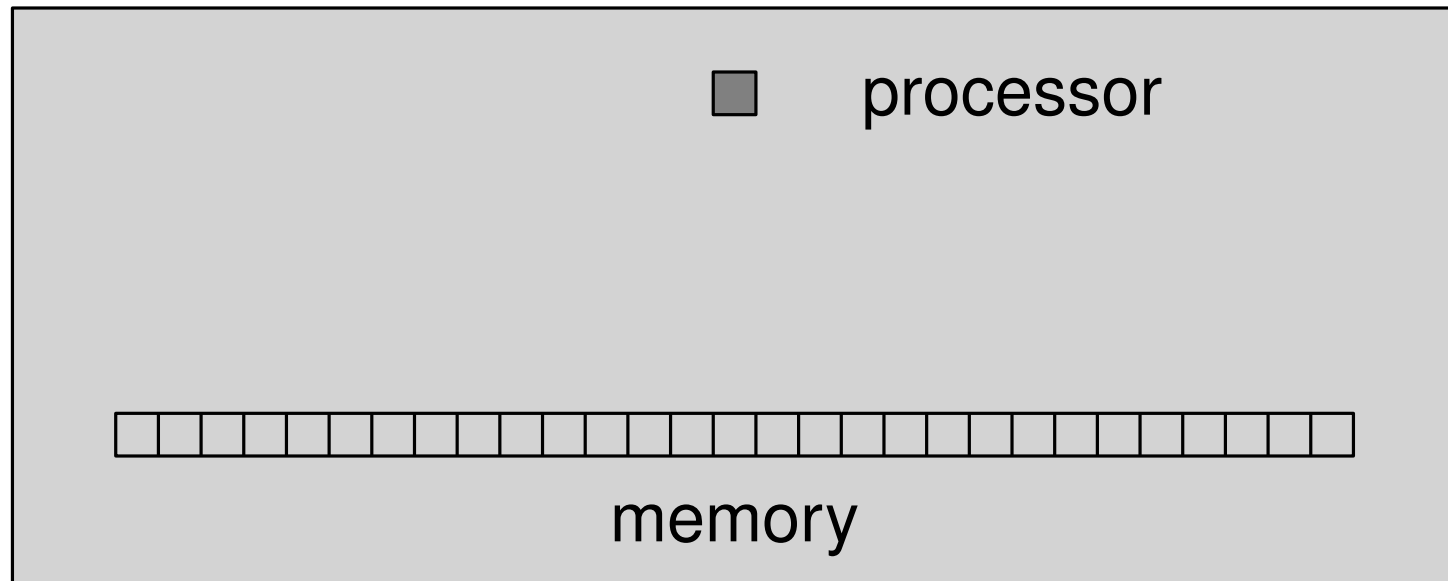
**for**  $i = 1$  to  $n$  **in parallel do**  
     $a[i] = a[i] + 1$



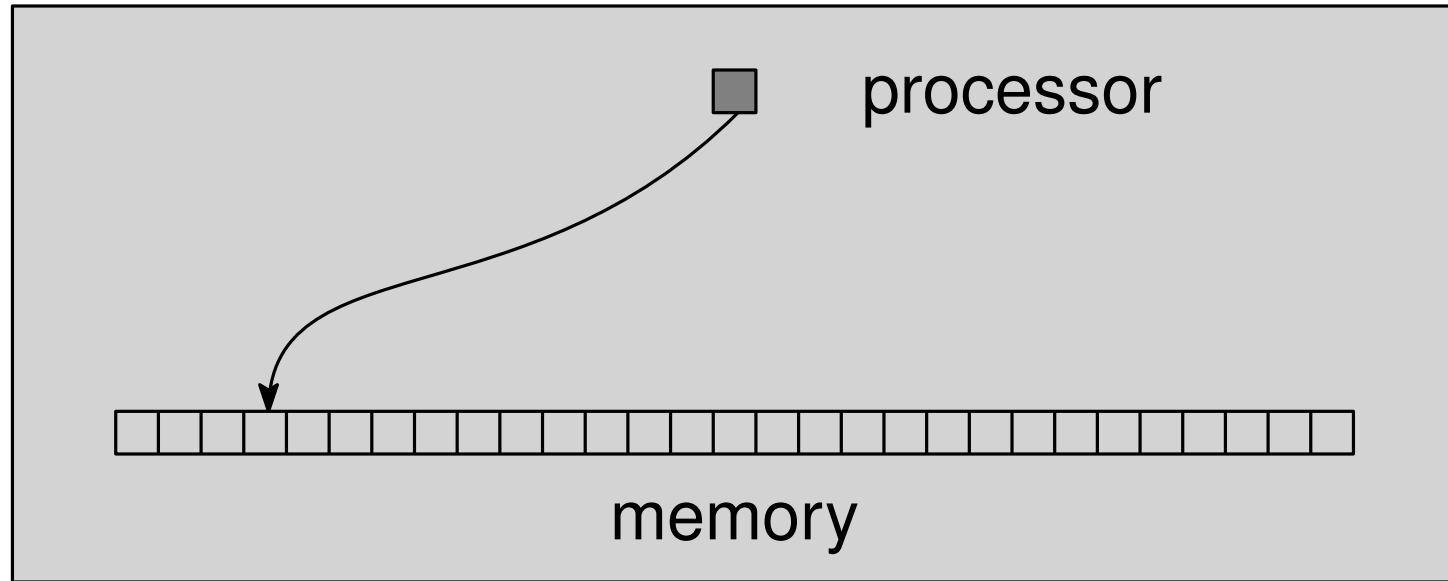
Start  $n$  threads  $t_1, t_2, \dots, t_n$   
Each thread  $t_i$  (where  $i = 1, 2, \dots, n$ ) **do**:  
     $a[i] = a[i] + 1$

Parallel Time = time of the slowest thread

# Random Access Memory (RAM) Model

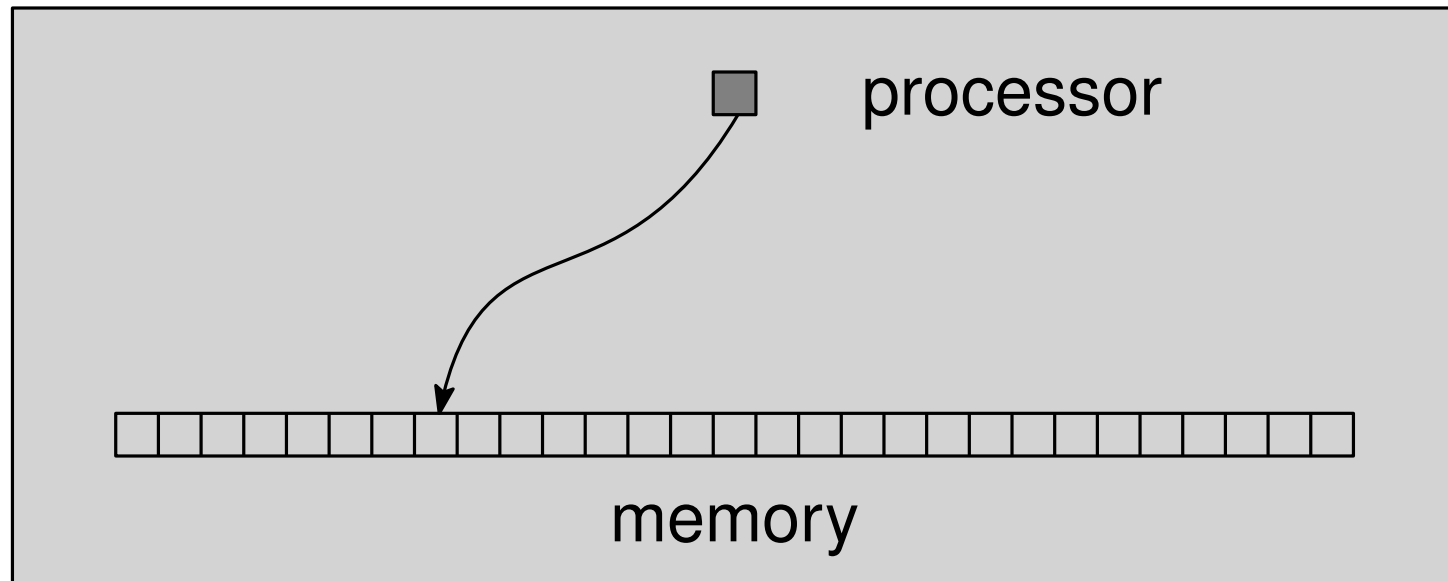


# Random Access Memory (RAM) Model



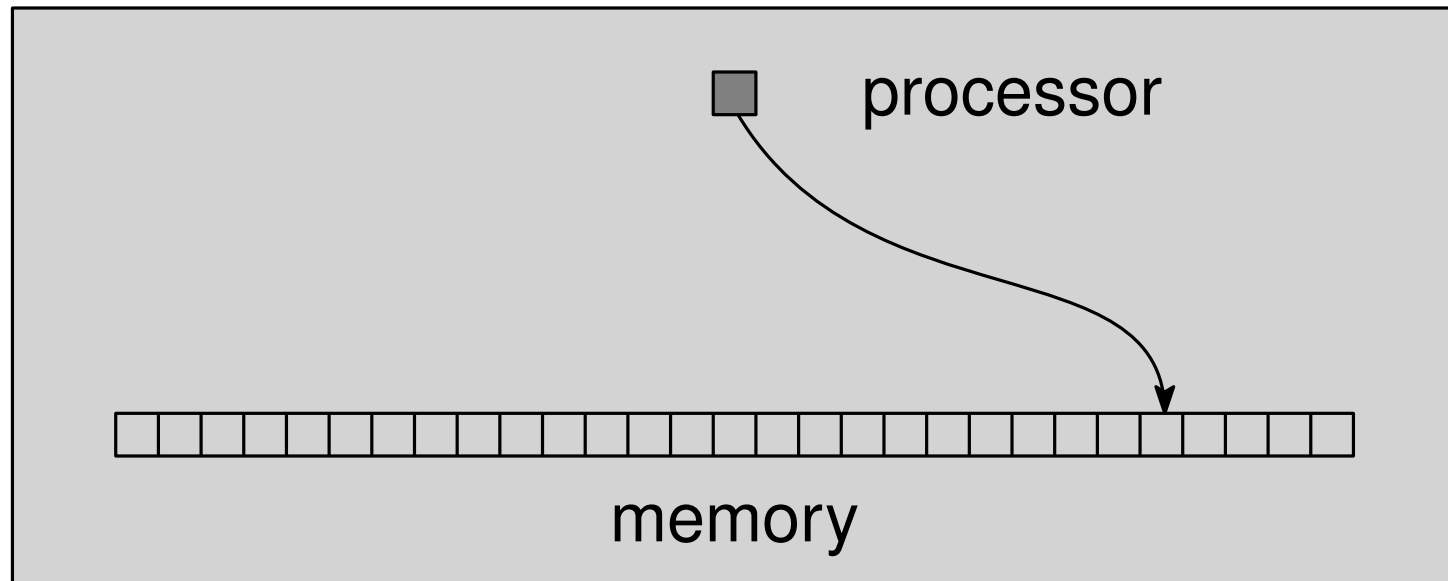
$\Theta(1)$  time per any access

# Random Access Memory (RAM) Model



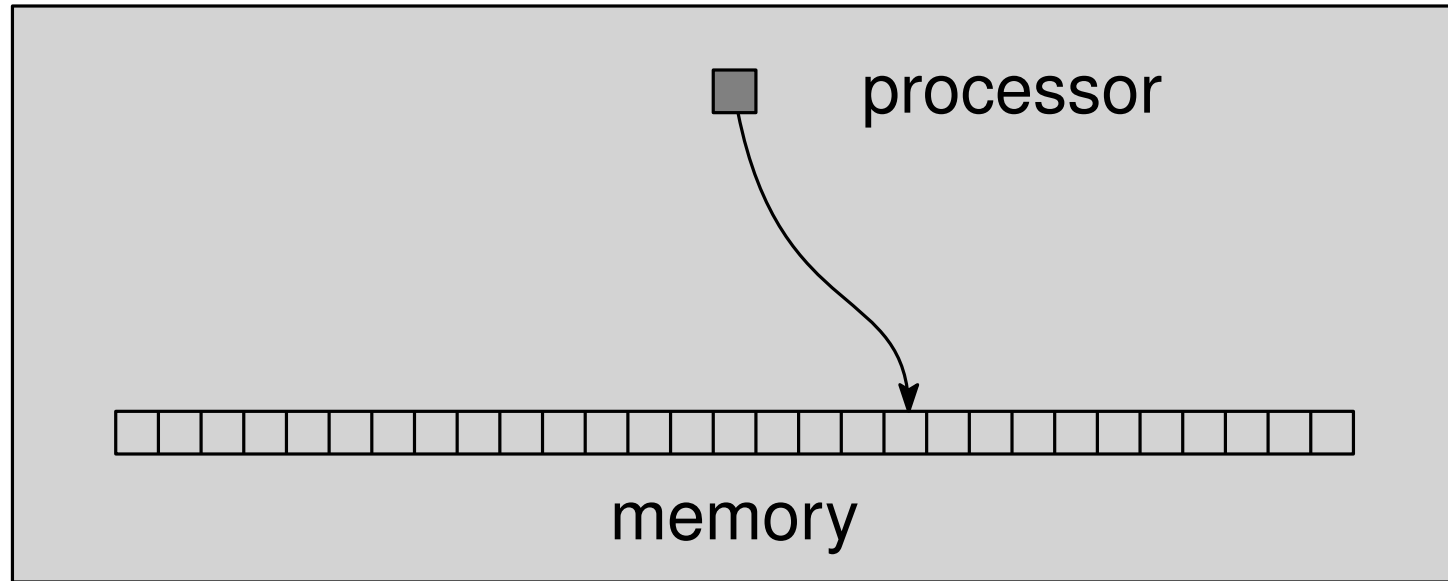
$\Theta(1)$  time per any access

# Random Access Memory (RAM) Model



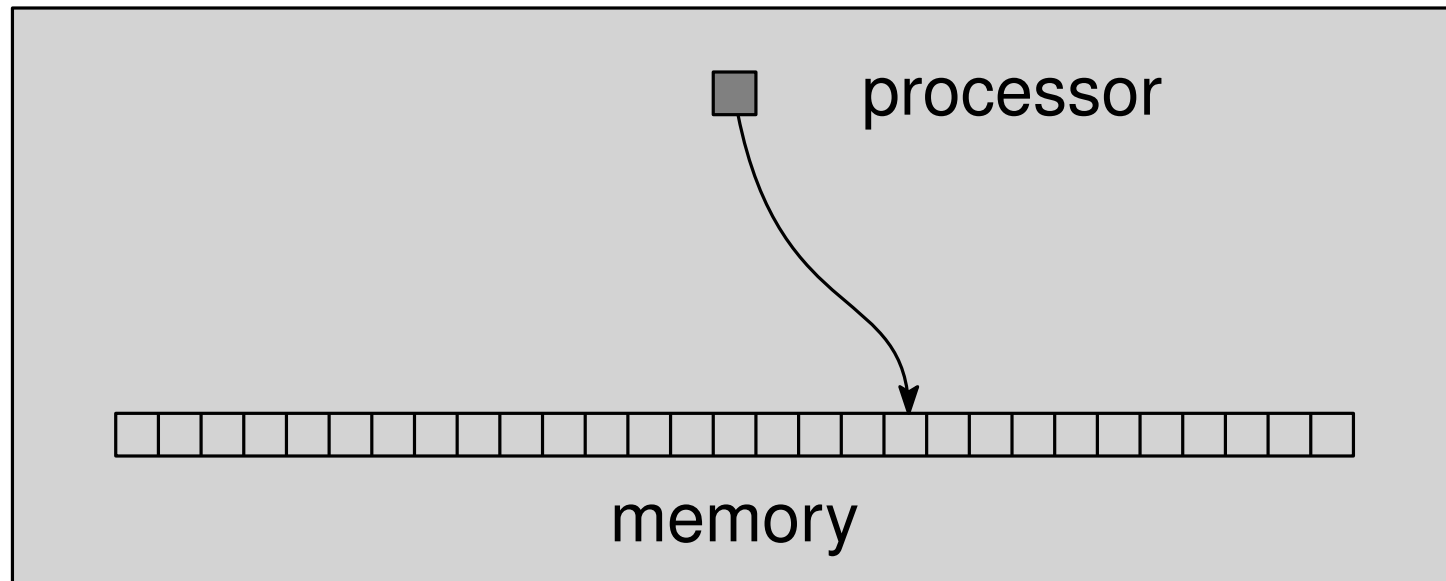
$\Theta(1)$  time per any access

# Random Access Memory (RAM) Model



$\Theta(1)$  time per any access

# Random Access Memory (RAM) Model



$\Theta(1)$  time per any access

Simple operations:  $O(1)$  accesses

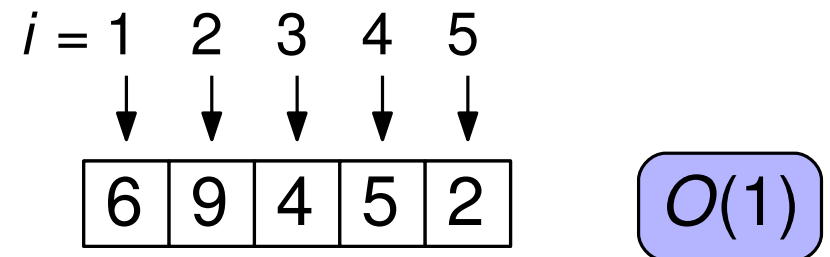
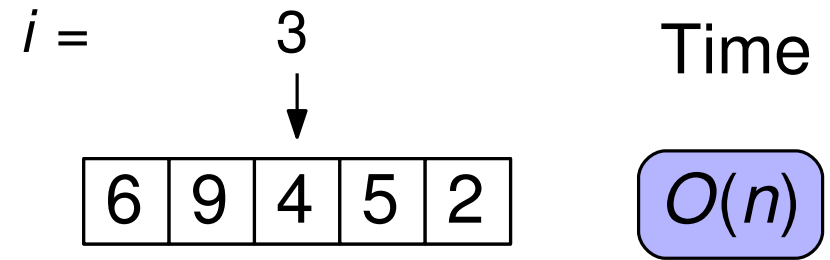
$$a = b + c$$



# Example from last lecture

**for**  $i = 1$  to  $n$  **do**  
     $a[i] = a[i] + 1$

**for**  $i = 1$  to  $n$  **in parallel do**  
     $a[i] = a[i] + 1$

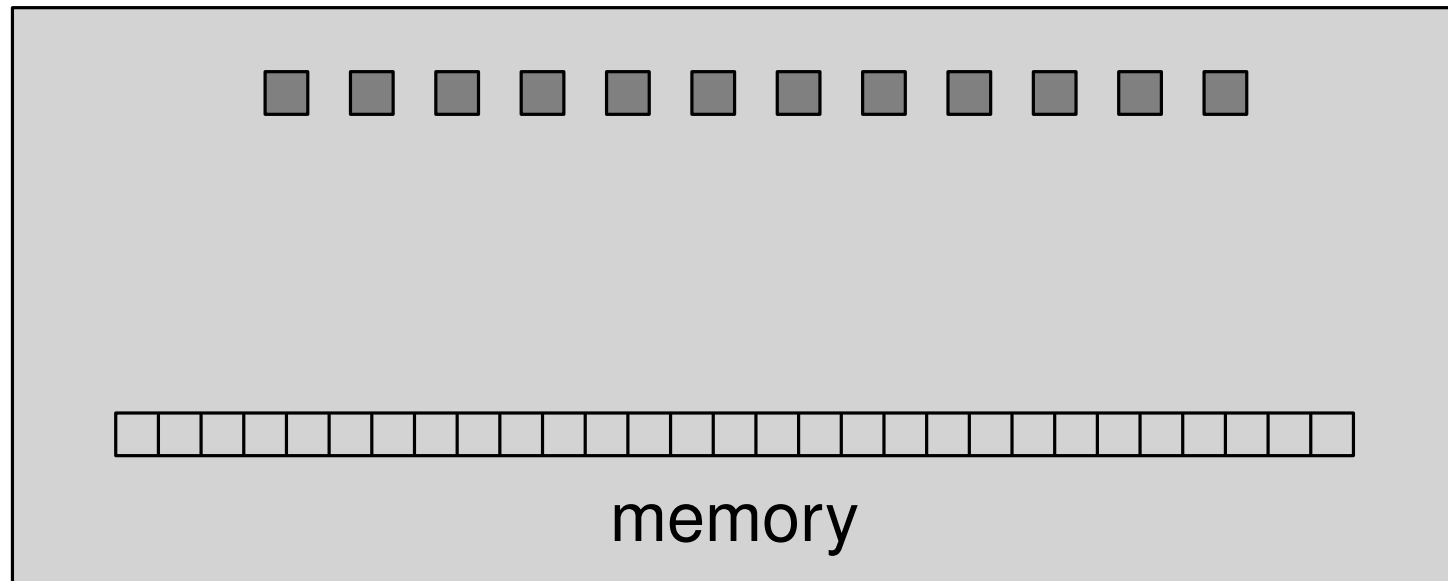


Start  $n$  threads  $t_1, t_2, \dots, t_n$   
Each thread  $t_i$  (where  $i = 1, 2, \dots, n$ ) **do**:  
     $a[i] = a[i] + 1$

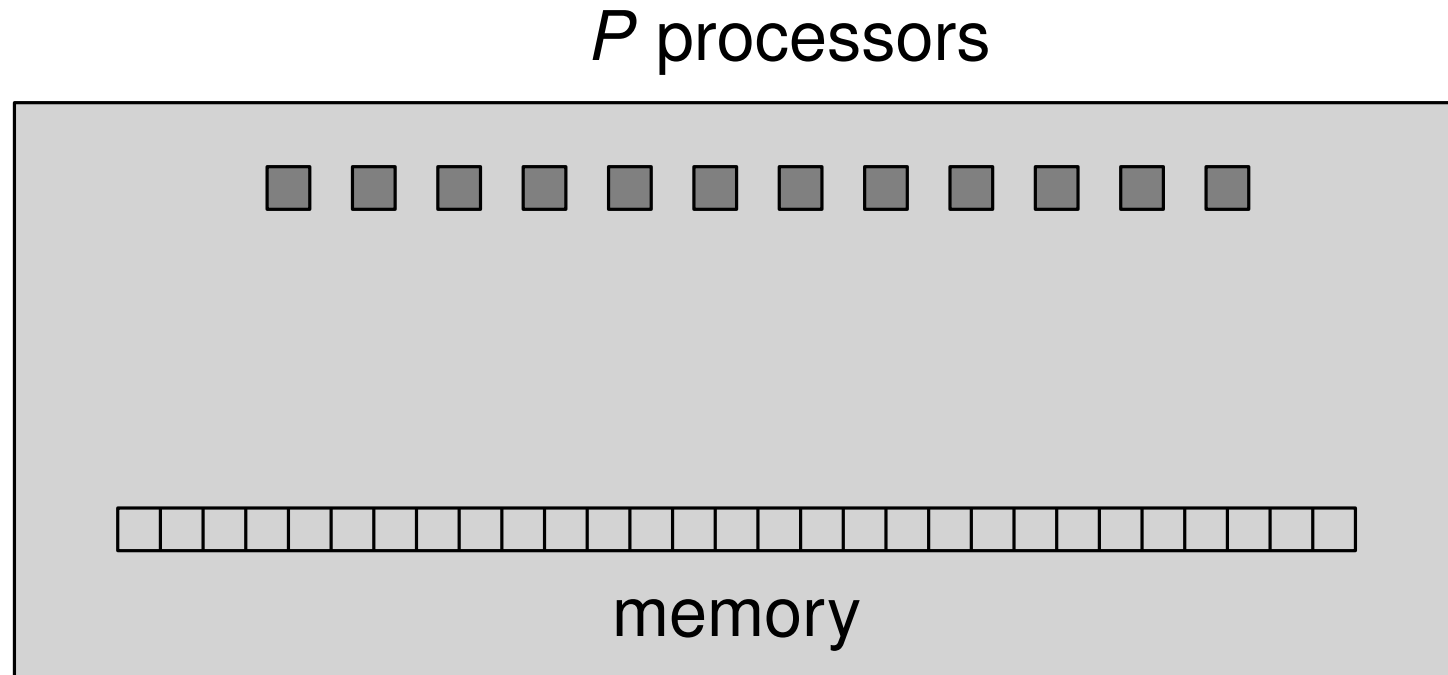
Parallel Time = time of the slowest thread

# Parallel RAM (PRAM) Model

$P$  processors



# Parallel RAM (PRAM) Model



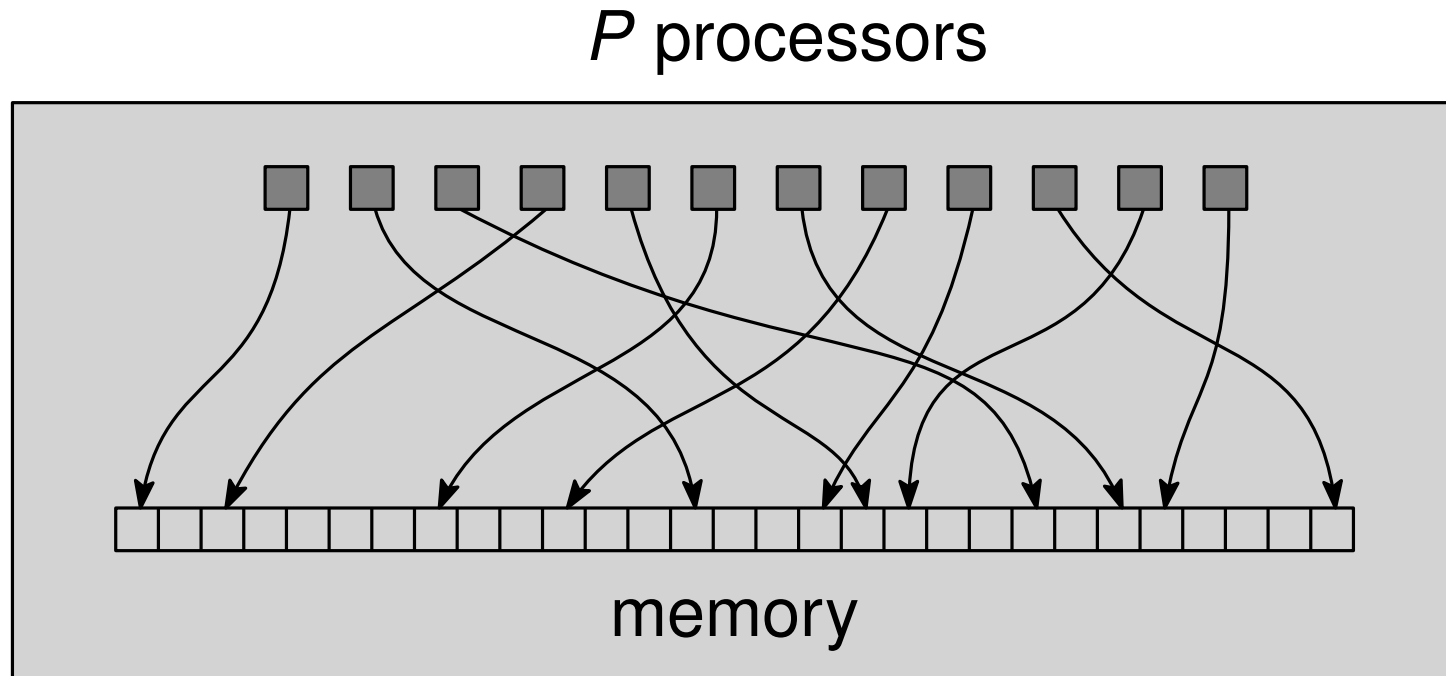
All process access memory concurrently

$O(1)$  time per access

Each operation:  $O(1)$  accesses

```
for  $i = 1$  to  $n$  in parallel do  
   $a[i] = a[i] + 1$ 
```

# Parallel RAM (PRAM) Model



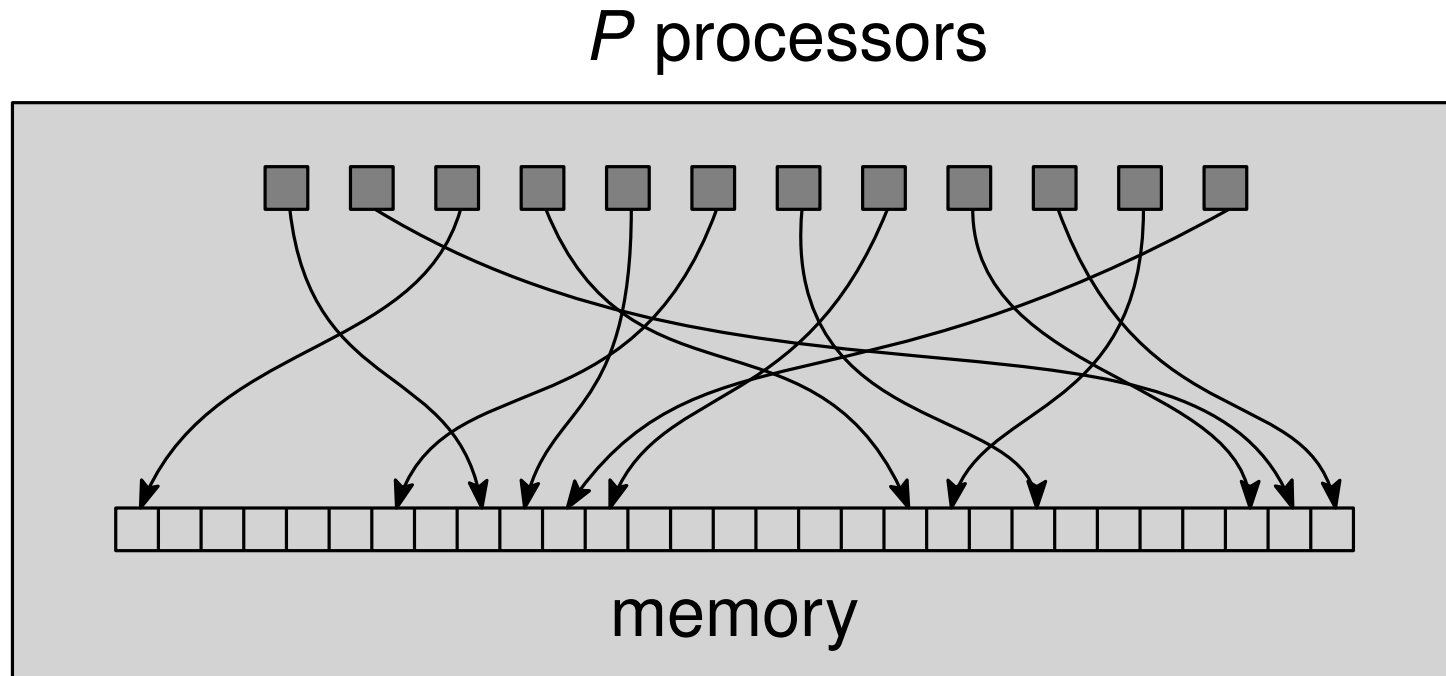
All process access memory concurrently

$O(1)$  time per access

Each operation:  $O(1)$  accesses

```
for  $i = 1$  to  $n$  in parallel do  
     $a[i] = a[i] + 1$ 
```

# Parallel RAM (PRAM) Model



All process access memory concurrently

$O(1)$  time per access

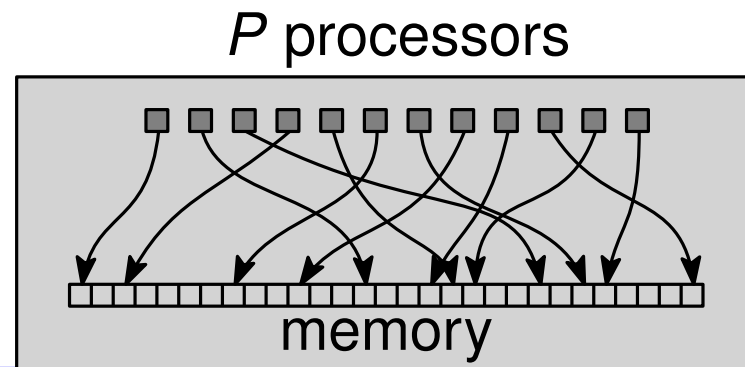
Each operation:  $O(1)$  accesses

```
for  $i = 1$  to  $n$  in parallel do  
     $a[i] = a[i] + 1$ 
```

# PRAM Variants

## Exclusive Read, Exclusive Write (EREW) PRAM

- No concurrent accesses by multiple processors at any time step



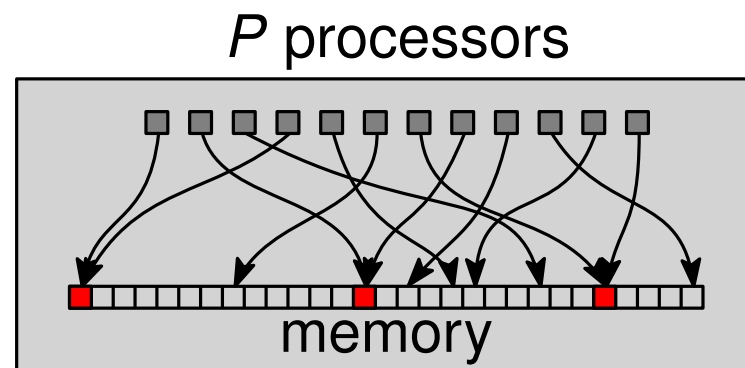
# PRAM Variants

## Exclusive Read, Exclusive Write (EREW) PRAM

- No concurrent accesses by multiple processors at any time step

## Concurrent Read, Exclusive Write (CREW) PRAM

- Only concurrent accesses for **reading** are allowed



# PRAM Variants

## Exclusive Read, Exclusive Write (EREW) PRAM

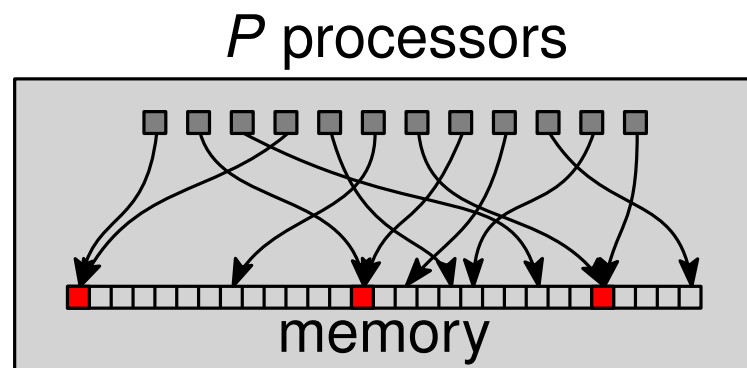
- No concurrent accesses by multiple processors at any time step

## Concurrent Read, Exclusive Write (CREW) PRAM

- Only concurrent accesses for **reading** are allowed

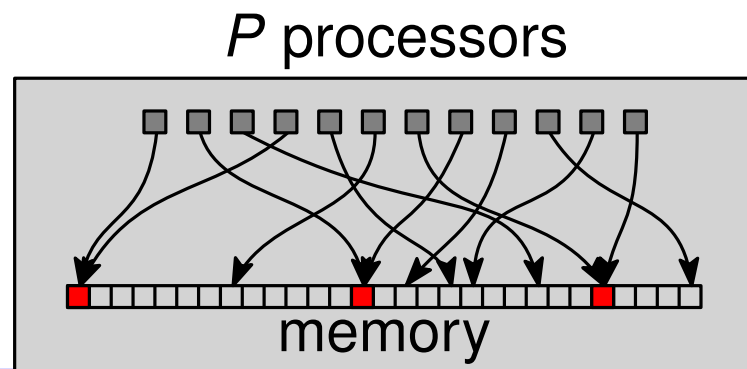
## Concurrent Read, Concurrent Write (CRCW) PRAM

- Concurrent accesses for both **reading** and **writing** are allowed





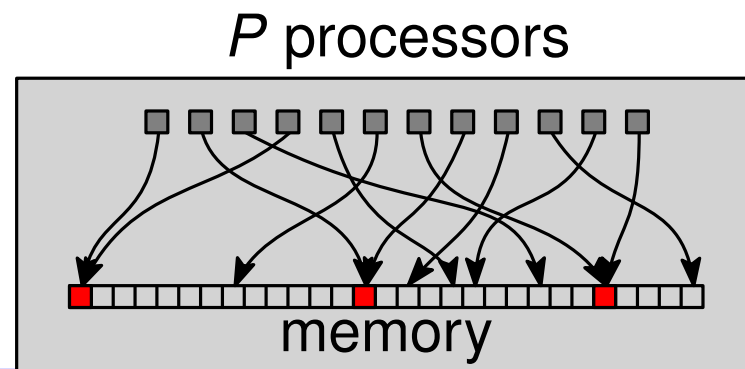
# CRCW PRAM Variants



# CRCW PRAM Variants

## Common-CRCW PRAM

- Concurrent accesses must write the same value



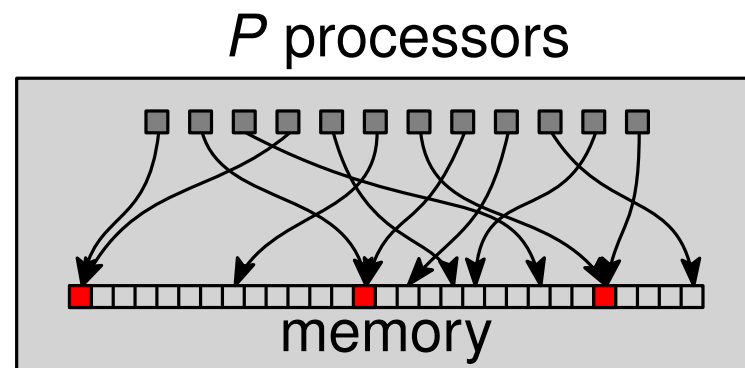
# CRCW PRAM Variants

## Common-CRCW PRAM

- Concurrent accesses must write the same value

## Arbitrary-CRCW PRAM

- One processor succeeds, don't know which one



# CRCW PRAM Variants

## Common-CRCW PRAM

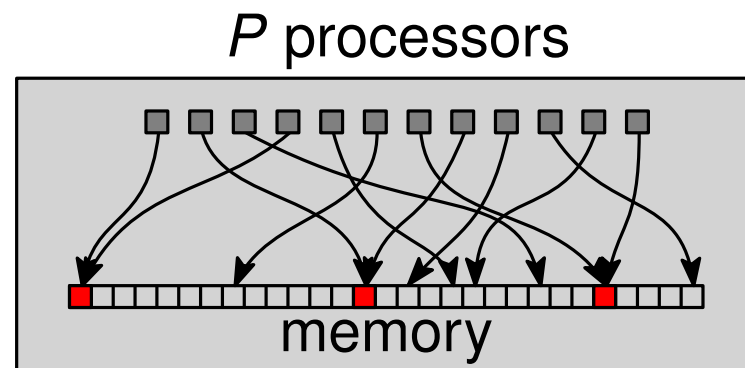
- Concurrent accesses must write the same value

## Arbitrary-CRCW PRAM

- One processor succeeds, don't know which one

## Priority-CRCW PRAM

- Processor with the smallest ID succeeds



# CRCW PRAM Variants

## Common-CRCW PRAM

- Concurrent accesses must write the same value

## Arbitrary-CRCW PRAM

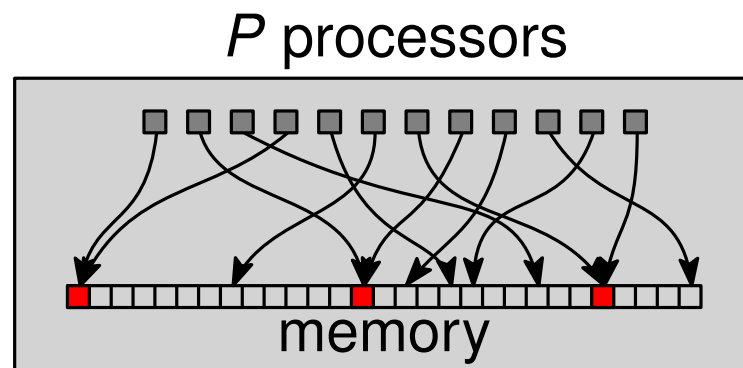
- One processor succeeds, don't know which one

## Priority-CRCW PRAM

- Processor with the smallest ID succeeds

## min-CRCW, sum-CRCW, OR-CRCW, XOR-CRCW PRAM

- The values of concurrent accesses are combined using a predetermined combining operation (e.g., min, sum, OR, XOR, etc) and the result is written



# CRCW PRAM Variants

## Common-CRCW PRAM

- Concurrent accesses must write the same value

## Arbitrary-CRCW PRAM

- One processor succeeds, don't know which one

## Priority-CRCW PRAM

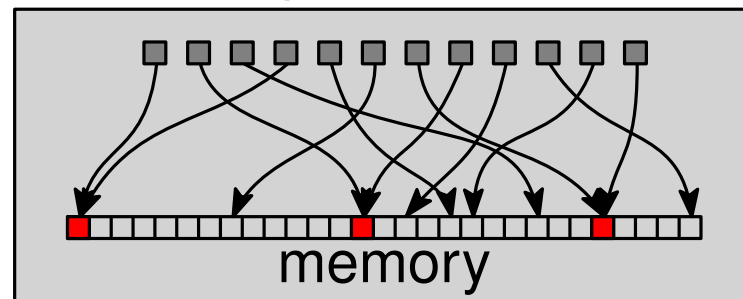
- Processor with the smallest ID succeeds

## min-CRCW, sum-CRCW, OR-CRCW, XOR-CRCW PRAM

- The values of concurrent accesses are combined using a predetermined combining operation (e.g., min, sum, OR, XOR, etc) and the result is written

More power

$P$  processors



# Power of Concurrent Writes: sum-CRCW

RAM

```
x = 0  
for i = 1 to n do  
    x = x + a[i]
```

Time:  $O(n)$

# Power of Concurrent Writes: sum-CRCW

RAM

```
x = 0  
for i = 1 to n do  
    x = x + a[i]
```

Time:  $O(n)$

Sum-CRCW PRAM:



# Power of Concurrent Writes: sum-CRCW

RAM

```
x = 0
for i = 1 to n do
  x = x + a[i]
```

Time:  $O(n)$

Sum-CRCW PRAM:

```
x = 0
for i = 1 to n in parallel do
  x = x + a[i]
```

# Power of Concurrent Writes: sum-CRCW

RAM

```
x = 0
for i = 1 to n do
  x = x + a[i]
```

Time:  $O(n)$

Sum-CRCW PRAM:

```
x = 0
for i = 1 to n in parallel do
  x = a[i]
```

# Power of Concurrent Writes: sum-CRCW

RAM

```
x = 0  
for i = 1 to n do  
    x = x + a[i]
```

Time:  $O(n)$

Sum-CRCW PRAM:

```
for i = 1 to n in parallel do  
    x = a[i]
```

# Power of Concurrent Writes: sum-CRCW

RAM

```
x = 0  
for i = 1 to n do  
  x = x + a[i]
```

Time:  $O(n)$

Sum-CRCW PRAM:

```
for i = 1 to n in parallel do  
  x = a[i]
```

Time:  $O(1)$

# E.g. Finding Minimum

**Problem 1.** *Given an array  $a$  of size  $n$ , find the smallest value*

# E.g. Finding Minimum

**Problem 1.** *Given an array  $a$  of size  $n$ , find the smallest value*

RAM

```
min =  $+\infty$   
for  $i = 1$  to  $n$  do  
  if  $a[i] < min$  then  
     $min = a[i]$ 
```

# E.g. Finding Minimum

**Problem 1.** *Given an array  $a$  of size  $n$ , find the smallest value*

RAM

```
min =  $+\infty$   
for  $i = 1$  to  $n$  do  
  if  $a[i] < min$  then  
     $min = a[i]$ 
```

Time:  $O(n)$

# E.g. Finding Minimum

**Problem 1.** *Given an array  $a$  of size  $n$ , find the smallest value*

RAM

```
min =  $+\infty$   
for  $i = 1$  to  $n$  do  
  if  $a[i] < min$  then  
     $min = a[i]$ 
```

Time:  $O(n)$

min-CRCW PRAM:



# E.g. Finding Minimum

**Problem 1.** *Given an array  $a$  of size  $n$ , find the smallest value*

RAM

```
min =  $+\infty$   
for  $i = 1$  to  $n$  do  
  if  $a[i] < min$  then  
     $min = a[i]$ 
```

Time:  $O(n)$

min-CRCW PRAM:

```
for  $i = 1$  to  $n$  in parallel do  
   $min = a[i]$ 
```

# E.g. Finding Minimum

**Problem 1.** *Given an array  $a$  of size  $n$ , find the smallest value*

RAM

```
min =  $+\infty$   
for  $i = 1$  to  $n$  do  
  if  $a[i] < \textit{min}$  then  
     $\textit{min} = a[i]$ 
```

Time:  $O(n)$

min-CRCW PRAM:

```
for  $i = 1$  to  $n$  in parallel do  
   $\textit{min} = a[i]$ 
```

Time:  $O(1)$

# Finding Minimum: common-CRCW

$a$ 

5	8	3	2	9	7
---	---	---	---	---	---

# Finding Minimum: common-CRCW

<i>a</i>	5	8	3	2	9	7
5						
8						
3						
2						
9						
7						

# Finding Minimum: common-CRCW

<i>a</i>	5	8	3	2	9	7
5						
8						
3						
2						
9						
7						

*M*

# Finding Minimum: common-CRCW

<i>a</i>	5	8	3	2	9	7
5						
8						
3						
2						
9						
7						

*M*

$$M[\textit{row}, \textit{column}] = (a[\textit{row}] > a[\textit{column}]) ? 1 : 0$$

# Finding Minimum: common-CRCW

<i>a</i>	5	8	3	2	9	7
5	0	0	1	1	0	0
8	1	0	1	1	0	1
3	0	0	0	1	0	0
2	0	0	0	0	0	0
9	1	1	1	1	0	1
7	1	0	1	1	0	0

*M*

$$M[\text{row}, \text{column}] = (a[\text{row}] > a[\text{column}]) ? 1 : 0$$

# Finding Minimum: common-CRCW

<i>a</i>	5	8	3	2	9	7
5	0	0	1	1	0	0
8	1	0	1	1	0	1
3	0	0	0	1	0	0
→ 2	0	0	0	0	0	0
9	1	1	1	1	0	1
7	1	0	1	1	0	0

*M*

$$M[\text{row}, \text{column}] = (a[\text{row}] > a[\text{column}]) ? 1 : 0$$



# Finding Minimum: common-CRCW

<i>a</i>	5	8	3	2	9	7
5	0	0	1	1	0	0
8	1	0	1	1	0	1
3	0	0	0	1	0	0
→ 2	0	0	0	0	0	0
9	1	1	1	1	0	1
7	1	0	1	1	0	0

*M*

$$M[\text{row}, \text{column}] = (a[\text{row}] > a[\text{column}]) ? 1 : 0$$

# Finding Minimum: common-CRCW

<i>a</i>	5	8	3	2	9	7
5	0	0	1	1	0	0
8	1	0	1	1	0	1
3	0	0	0	1	0	0
2	0	0	0	0	0	0
9	1	1	1	1	0	1
7	1	0	1	1	0	0

*M*

```
for row = 1 to n in parallel do
  for col = 1 to n in parallel do
    if a[row] > a[col] then
      M[row, col] = 1
    else
      M[row, col] = 0
```

$$M[\text{row}, \text{column}] = (a[\text{row}] > a[\text{column}]) ? 1 : 0$$

# Finding Minimum: common-CRCW

<i>x</i>	<i>a</i>	5	8	3	2	9	7
0	5	0	0	1	1	0	0
0	8	1	0	1	1	0	1
0	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
0	9	1	1	1	1	0	1
0	7	1	0	1	1	0	0

*M*

```
for row = 1 to n in parallel do
  for col = 1 to n in parallel do
    if a[row] > a[col] then
      M[row, col] = 1
    else
      M[row, col] = 0
```

$$M[\text{row}, \text{column}] = (a[\text{row}] > a[\text{column}]) ? 1 : 0$$

# Finding Minimum: common-CRCW

$x$	$a$	5	8	3	2	9	7
0	5	0	0	1	1	0	0
0	8	1	0	1	1	0	1
0	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
0	9	1	1	1	1	0	1
0	7	1	0	1	1	0	0

$M$

```
for row = 1 to n in parallel do
  for col = 1 to n in parallel do
    if  $a[\text{row}] > a[\text{col}]$  then
       $M[\text{row}, \text{col}] = 1$ 
    else
       $M[\text{row}, \text{col}] = 0$ 
```

```
allocate new array  $x[1..n]$ 
for  $i = 1$  to  $n$  in parallel do
   $x[i] = 0$ 
```

# Finding Minimum: common-CRCW

$x$	$a$	5	8	3	2	9	7
1	5	0	0	1	1	0	0
1	8	1	0	1	1	0	1
1	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
1	9	1	1	1	1	0	1
1	7	1	0	1	1	0	0

$M$

```
for row = 1 to n in parallel do
  for col = 1 to n in parallel do
    if  $a[\text{row}] > a[\text{col}]$  then
       $M[\text{row}, \text{col}] = 1$ 
    else
       $M[\text{row}, \text{col}] = 0$ 
```

```
allocate new array  $x[1..n]$ 
for  $i = 1$  to  $n$  in parallel do
   $x[i] = 0$ 
```

# Finding Minimum: common-CRCW

$x$	$a$	5	8	3	2	9	7
1	5	0	0	1	1	0	0
1	8	1	0	1	1	0	1
1	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
1	9	1	1	1	1	0	1
1	7	1	0	1	1	0	0

$M$

```
for row = 1 to n in parallel do
  for col = 1 to n in parallel do
    if  $a[\text{row}] > a[\text{col}]$  then
       $M[\text{row}, \text{col}] = 1$ 
    else
       $M[\text{row}, \text{col}] = 0$ 
```

```
allocate new array  $x[1..n]$ 
for  $i = 1$  to  $n$  in parallel do
   $x[i] = 0$ 
```

```
for row = 1 to n in parallel do
  for col = 1 to n in parallel do
    if  $a[\text{row}, \text{col}] == 1$  then
       $x[\text{row}] = 1$ 
```

# Finding Minimum: common-CRCW

$x$	$a$	5	8	3	2	9	7
1	5	0	0	1	1	0	0
1	8	1	0	1	1	0	1
1	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
1	9	1	1	1	1	0	1
1	7	1	0	1	1	0	0

$M$

```
for row = 1 to n in parallel do
  for col = 1 to n in parallel do
    if  $a[row] > a[col]$  then
       $M[row, col] = 1$ 
    else
       $M[row, col] = 0$ 
```

```
allocate new array  $x[1..n]$ 
for  $i = 1$  to  $n$  in parallel do
   $x[i] = 0$ 
```

```
for row = 1 to n in parallel do
  for col = 1 to n in parallel do
    if  $a[row, col] == 1$  then
       $x[row] = 1$ 
```

```
for row = 1 to n in parallel do
  if  $x[row] == 0$  then
     $min = a[row]$ 
```

# Finding Minimum: common-CRCW

$x$	$a$	5	8	3	2	9	7
1	5	0	0	1	1	0	0
1	8	1	0	1	1	0	1
1	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
1	9	1	1	1	1	0	1
1	7	1	0	1	1	0	0

$M$

Valid?

```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row] > a[col]$  then
       $M[row, col] = 1$ 
    else
       $M[row, col] = 0$ 
  
```

```

allocate new array  $x[1..n]$ 
for  $i = 1$  to  $n$  in parallel do
   $x[i] = 0$ 
  
```

```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row, col] == 1$  then
       $x[row] = 1$ 
  
```

```

for  $row = 1$  to  $n$  in parallel do
  if  $x[row] == 0$  then
     $min = a[row]$ 
  
```



# Finding Minimum: common-CRCW

$x$	$a$	5	8	3	2	9	7
1	5	0	0	1	1	0	0
1	8	1	0	1	1	0	1
1	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
1	9	1	1	1	1	0	1
1	7	1	0	1	1	0	0

Valid?

$M$

```
for row = 1 to n in parallel do
  for col = 1 to n in parallel do
    if a[row] > a[col] then
      M[row, col] = 1
    else
      M[row, col] = 0
```

```
allocate new array x[1..n]
for i = 1 to n in parallel do
  x[i] = 0
```

```
for row = 1 to n in parallel do
  for col = 1 to n in parallel do
    if a[row, col] == 1 then
      x[row] = 1
```

```
for row = 1 to n in parallel do
  if x[row] == 0 then
    min = a[row]
```

# Finding Minimum: common-CRCW

$x$	$a$	5	8	3	2	9	7
1	5	0	0	1	1	0	0
1	8	1	0	1	1	0	1
1	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
1	9	1	1	1	1	0	1
1	7	1	0	1	1	0	0

$M$

Runtime?

```
for row = 1 to n in parallel do
  for col = 1 to n in parallel do
    if a[row] > a[col] then
      M[row, col] = 1
    else
      M[row, col] = 0
```

```
allocate new array x[1..n]
for i = 1 to n in parallel do
  x[i] = 0
```

```
for row = 1 to n in parallel do
  for col = 1 to n in parallel do
    if a[row, col] == 1 then
      x[row] = 1
```

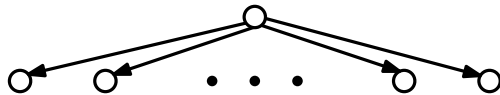
```
for row = 1 to n in parallel do
  if x[row] == 0 then
    min = a[row]
```

# Finding Minimum: common-CRCW

$x$	$a$	5	8	3	2	9	7
1	5	0	0	1	1	0	0
1	8	1	0	1	1	0	1
1	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
1	9	1	1	1	1	0	1
1	7	1	0	1	1	0	0

$M$

Runtime?



```
for row = 1 to n in parallel do
  for col = 1 to n in parallel do
    if  $a[\text{row}] > a[\text{col}]$  then
       $M[\text{row}, \text{col}] = 1$ 
    else
       $M[\text{row}, \text{col}] = 0$ 
```

```
allocate new array  $x[1..n]$ 
for  $i = 1$  to  $n$  in parallel do
   $x[i] = 0$ 
```

```
for row = 1 to n in parallel do
  for col = 1 to n in parallel do
    if  $a[\text{row}, \text{col}] == 1$  then
       $x[\text{row}] = 1$ 
```

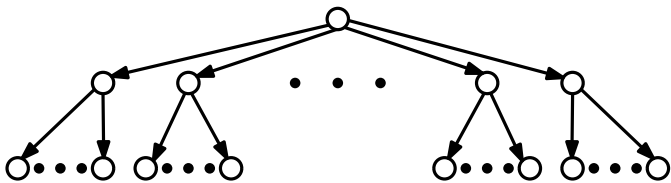
```
for row = 1 to n in parallel do
  if  $x[\text{row}] == 0$  then
     $\text{min} = a[\text{row}]$ 
```

# Finding Minimum: common-CRCW

$x$	$a$	5	8	3	2	9	7
1	5	0	0	1	1	0	0
1	8	1	0	1	1	0	1
1	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
1	9	1	1	1	1	0	1
1	7	1	0	1	1	0	0

$M$

Runtime?



```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row] > a[col]$  then
       $M[row, col] = 1$ 
    else
       $M[row, col] = 0$ 
  
```

```

allocate new array  $x[1..n]$ 
for  $i = 1$  to  $n$  in parallel do
   $x[i] = 0$ 
  
```

```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row, col] == 1$  then
       $x[row] = 1$ 
  
```

```

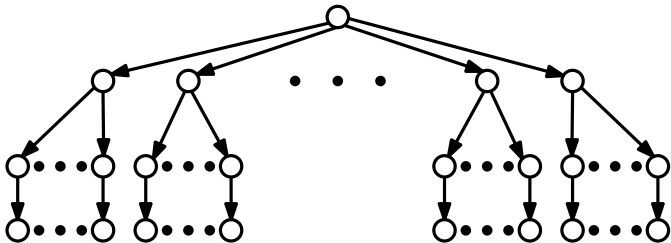
for  $row = 1$  to  $n$  in parallel do
  if  $x[row] == 0$  then
     $min = a[row]$ 
  
```

# Finding Minimum: common-CRCW

$x$	$a$	5	8	3	2	9	7
1	5	0	0	1	1	0	0
1	8	1	0	1	1	0	1
1	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
1	9	1	1	1	1	0	1
1	7	1	0	1	1	0	0

$M$

Runtime?



```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row] > a[col]$  then
       $M[row, col] = 1$ 
    else
       $M[row, col] = 0$ 
  
```

```

allocate new array  $x[1..n]$ 
for  $i = 1$  to  $n$  in parallel do
   $x[i] = 0$ 
  
```

```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row, col] == 1$  then
       $x[row] = 1$ 
  
```

```

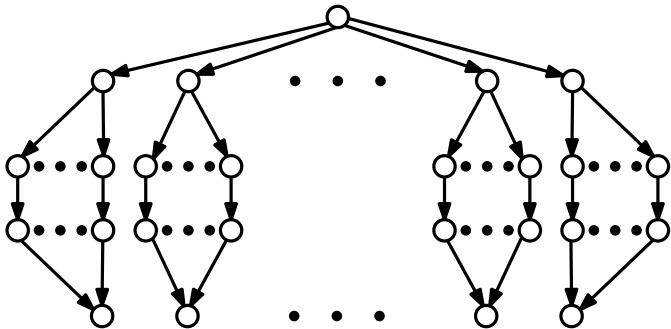
for  $row = 1$  to  $n$  in parallel do
  if  $x[row] == 0$  then
     $min = a[row]$ 
  
```

# Finding Minimum: common-CRCW

$x$	$a$	5	8	3	2	9	7
1	5	0	0	1	1	0	0
1	8	1	0	1	1	0	1
1	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
1	9	1	1	1	1	0	1
1	7	1	0	1	1	0	0

$M$

Runtime?



```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row] > a[col]$  then
       $M[row, col] = 1$ 
    else
       $M[row, col] = 0$ 
  
```

```

allocate new array  $x[1..n]$ 
for  $i = 1$  to  $n$  in parallel do
   $x[i] = 0$ 
  
```

```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row, col] == 1$  then
       $x[row] = 1$ 
  
```

```

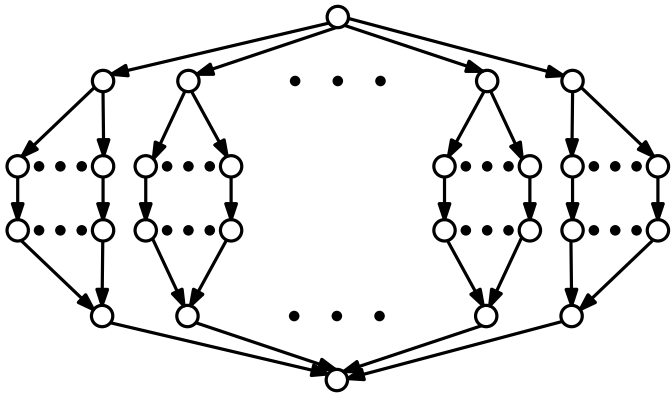
for  $row = 1$  to  $n$  in parallel do
  if  $x[row] == 0$  then
     $min = a[row]$ 
  
```

# Finding Minimum: common-CRCW

$x$	$a$	5	8	3	2	9	7
1	5	0	0	1	1	0	0
1	8	1	0	1	1	0	1
1	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
1	9	1	1	1	1	0	1
1	7	1	0	1	1	0	0

$M$

Runtime?



```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row] > a[col]$  then
       $M[row, col] = 1$ 
    else
       $M[row, col] = 0$ 
  
```

```

allocate new array  $x[1..n]$ 
for  $i = 1$  to  $n$  in parallel do
   $x[i] = 0$ 
  
```

```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row, col] == 1$  then
       $x[row] = 1$ 
  
```

```

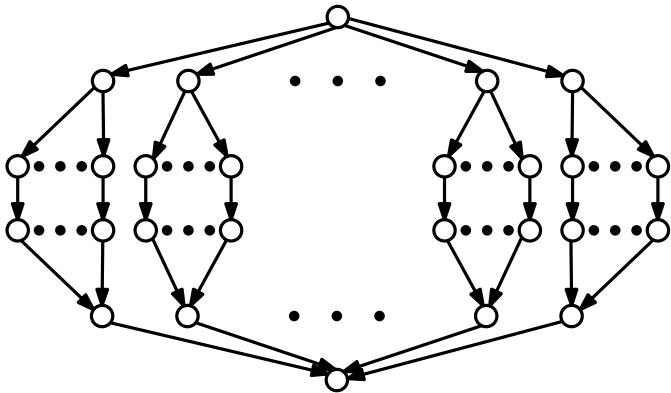
for  $row = 1$  to  $n$  in parallel do
  if  $x[row] == 0$  then
     $min = a[row]$ 
  
```

# Finding Minimum: common-CRCW

$x$	$a$	5	8	3	2	9	7
1	5	0	0	1	1	0	0
1	8	1	0	1	1	0	1
1	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
1	9	1	1	1	1	0	1
1	7	1	0	1	1	0	0

$M$

Runtime?



```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row] > a[col]$  then
       $M[row, col] = 1$ 
    else
       $M[row, col] = 0$ 
  
```

$O(1)$

```

allocate new array  $x[1..n]$ 
for  $i = 1$  to  $n$  in parallel do
   $x[i] = 0$ 
  
```

```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row, col] == 1$  then
       $x[row] = 1$ 
  
```

```

for  $row = 1$  to  $n$  in parallel do
  if  $x[row] == 0$  then
     $min = a[row]$ 
  
```

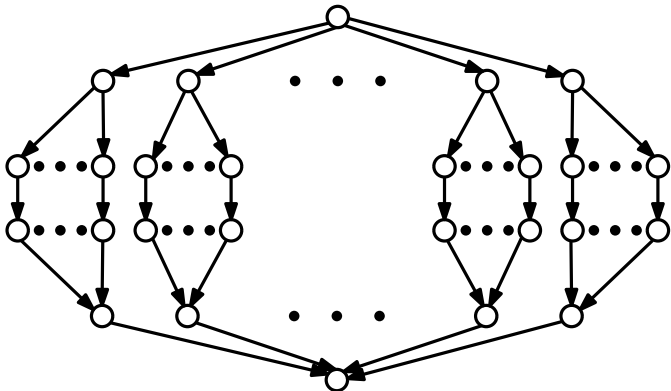


# Finding Minimum: common-CRCW

$x$	$a$	5	8	3	2	9	7
1	5	0	0	1	1	0	0
1	8	1	0	1	1	0	1
1	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
1	9	1	1	1	1	0	1
1	7	1	0	1	1	0	0

$M$

Runtime?



```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row] > a[col]$  then
       $M[row, col] = 1$ 
    else
       $M[row, col] = 0$ 
  
```

$O(1)$

```

allocate new array  $x[1..n]$ 
for  $i = 1$  to  $n$  in parallel do
   $x[i] = 0$ 
  
```

$O(1)$

```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row, col] == 1$  then
       $x[row] = 1$ 
  
```

```

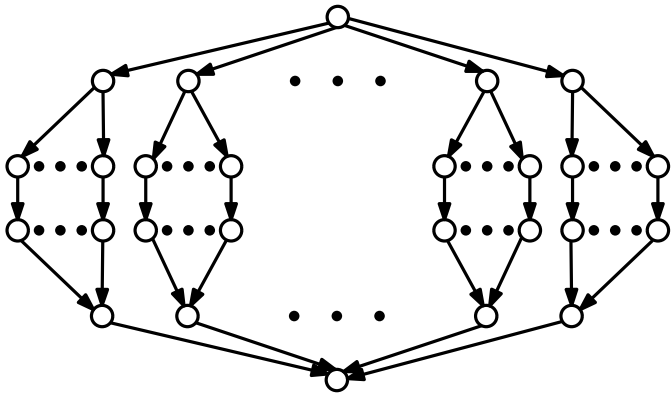
for  $row = 1$  to  $n$  in parallel do
  if  $x[row] == 0$  then
     $min = a[row]$ 
  
```

# Finding Minimum: common-CRCW

$x$	$a$	5	8	3	2	9	7
1	5	0	0	1	1	0	0
1	8	1	0	1	1	0	1
1	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
1	9	1	1	1	1	0	1
1	7	1	0	1	1	0	0

$M$

Runtime?



```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row] > a[col]$  then
       $M[row, col] = 1$ 
    else
       $M[row, col] = 0$ 
  
```

$O(1)$

```

allocate new array  $x[1..n]$ 
for  $i = 1$  to  $n$  in parallel do
   $x[i] = 0$ 
  
```

$O(1)$

```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row, col] == 1$  then
       $x[row] = 1$ 
  
```

$O(1)$

```

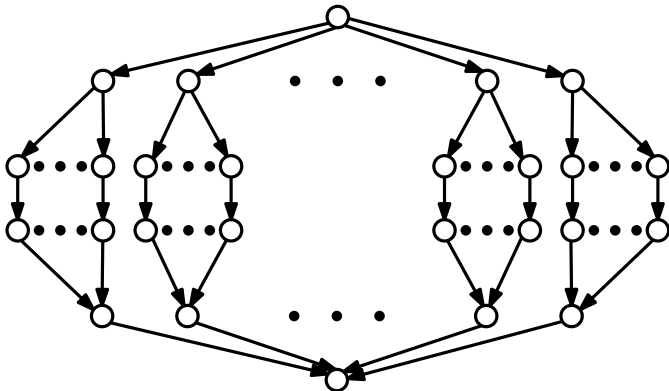
for  $row = 1$  to  $n$  in parallel do
  if  $x[row] == 0$  then
     $min = a[row]$ 
  
```

# Finding Minimum: common-CRCW

$x$	$a$	5	8	3	2	9	7
1	5	0	0	1	1	0	0
1	8	1	0	1	1	0	1
1	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
1	9	1	1	1	1	0	1
1	7	1	0	1	1	0	0

$M$

Runtime?



```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row] > a[col]$  then
       $M[row, col] = 1$ 
    else
       $M[row, col] = 0$ 
  
```

$O(1)$

```

allocate new array  $x[1..n]$ 
for  $i = 1$  to  $n$  in parallel do
   $x[i] = 0$ 
  
```

$O(1)$

```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row, col] == 1$  then
       $x[row] = 1$ 
  
```

$O(1)$

```

for  $row = 1$  to  $n$  in parallel do
  if  $x[row] == 0$  then
     $min = a[row]$ 
  
```

$O(1)$

# Finding Minimum: common-CRCW

$x$	$a$	5	8	3	2	9	7
1	5	0	0	1	1	0	0
1	8	1	0	1	1	0	1
1	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
1	9	1	1	1	1	0	1
1	7	1	0	1	1	0	0

$M$

Runtime?

Arbitrary-CRCW?

```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row] > a[col]$  then
       $M[row, col] = 1$ 
    else
       $M[row, col] = 0$ 
  
```

$O(1)$

```

allocate new array  $x[1..n]$ 
for  $i = 1$  to  $n$  in parallel do
   $x[i] = 0$ 
  
```

$O(1)$

```

for  $row = 1$  to  $n$  in parallel do
  for  $col = 1$  to  $n$  in parallel do
    if  $a[row, col] == 1$  then
       $x[row] = 1$ 
  
```

$O(1)$

```

for  $row = 1$  to  $n$  in parallel do
  if  $x[row] == 0$  then
     $min = a[row]$ 
  
```

$O(1)$

# Finding Minimum: common-CRCW

<i>x</i>	<i>a</i>	5	8	3	2	9	7
1	5	0	0	1	1	0	0
1	8	1	0	1	1	0	1
1	3	0	0	0	1	0	0
0	2	0	0	0	0	0	0
1	9	1	1	1	1	0	1
1	7	1	0	1	1	0	0

Runtime?

Arbitrary-CRCW?

*M*

```

for row = 1 to n in parallel do
  for col = 1 to n in parallel do
    if a[row] > a[col] then
      M[row, col] = 1
    else
      M[row, col] = 0
  
```

$O(1)$

```

allocate new array x[1..n]
for i = 1 to n in parallel do
  x[i] = 0
  
```

$O(1)$

```

for row = 1 to n in parallel do
  for col = 1 to n in parallel do
    if a[row, col] == 1 then
      x[row] = 1
  
```

$O(1)$

```

for row = 1 to n in parallel do
  if x[row] == 0 then
    min = a[row]
  
```

$O(1)$

# CRCW PRAM Variants

## Common-CRCW PRAM

- Concurrent accesses must write the same value

## Arbitrary-CRCW PRAM

- One processor succeeds, don't know which one

## Priority-CRCW PRAM

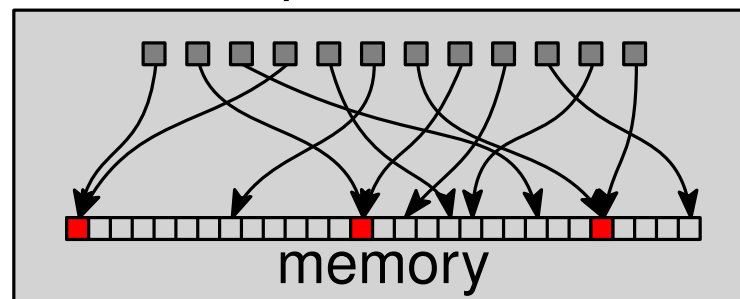
- Processor with the smallest ID succeeds

## min-CRCW, sum-CRCW, OR-CRCW, XOR-CRCW PRAM

- The values of concurrent accesses are combined using a predetermined combining operation (e.g., min, sum, OR, XOR, etc) and the result is written

More power

$P$  processors



# Finding Minimum: EREW PRAM

$a$ : 

5	8	3	2	9	7
---	---	---	---	---	---

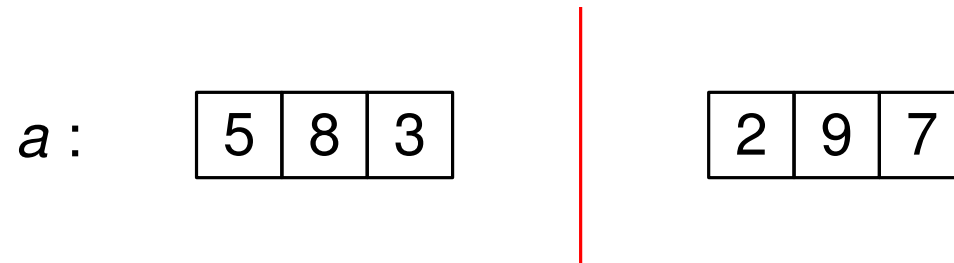
# Finding Minimum: EREW PRAM

*a* :

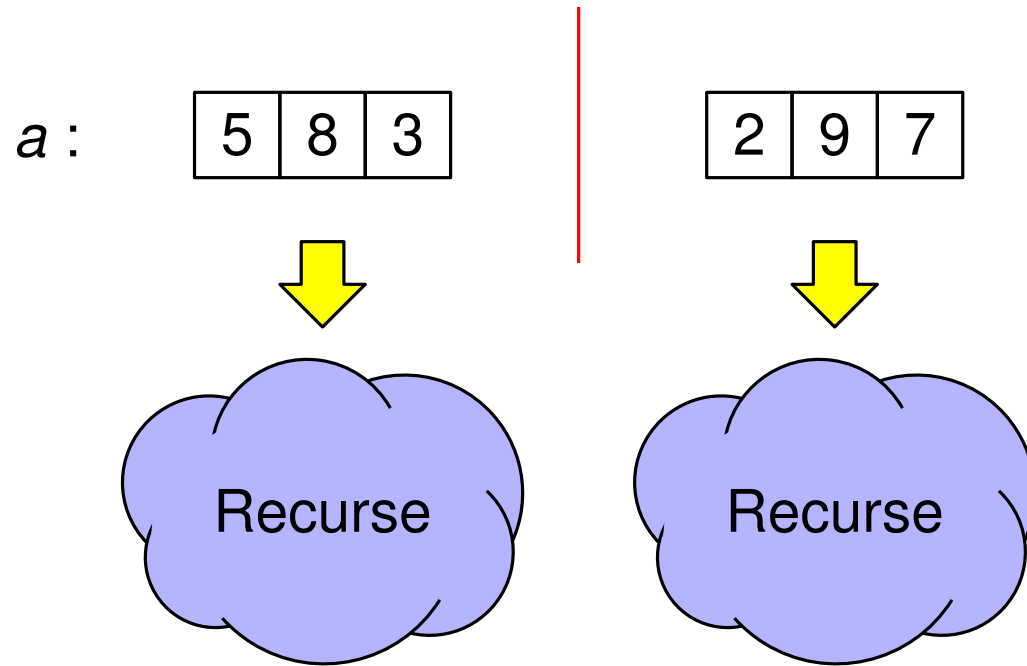
5	8	3	2	9	7
---	---	---	---	---	---



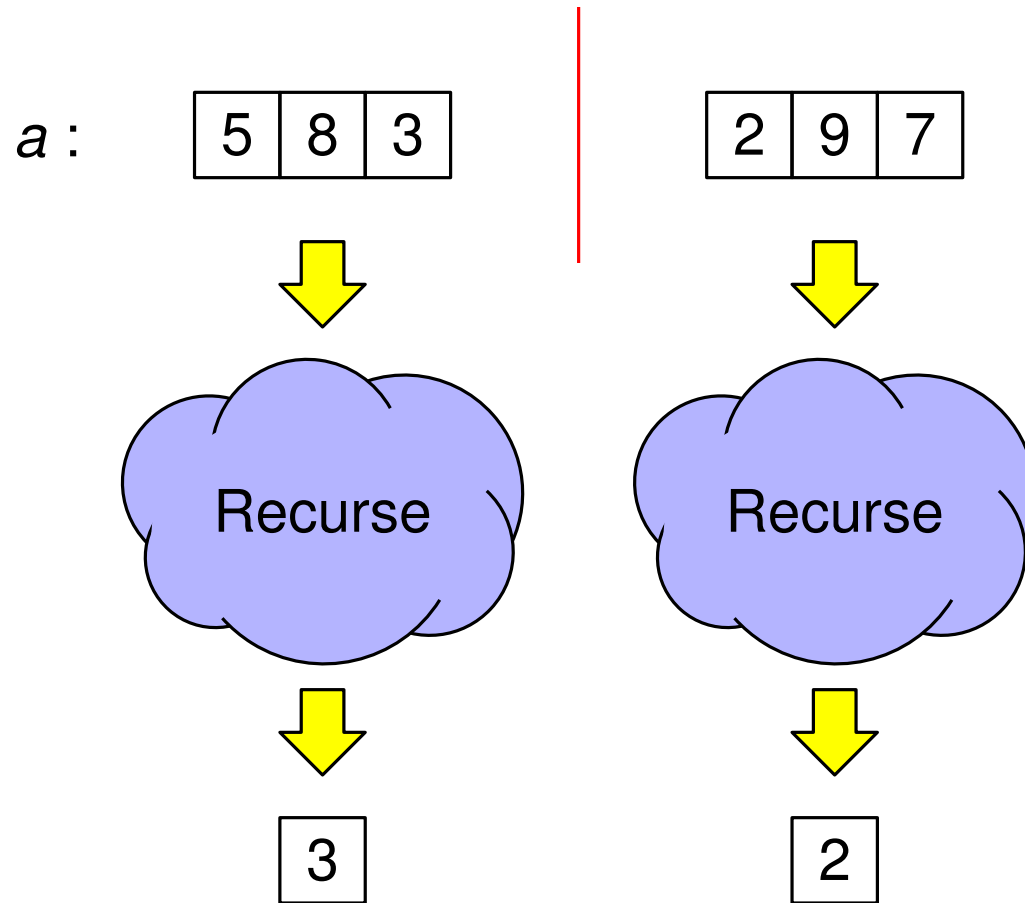
# Finding Minimum: EREW PRAM



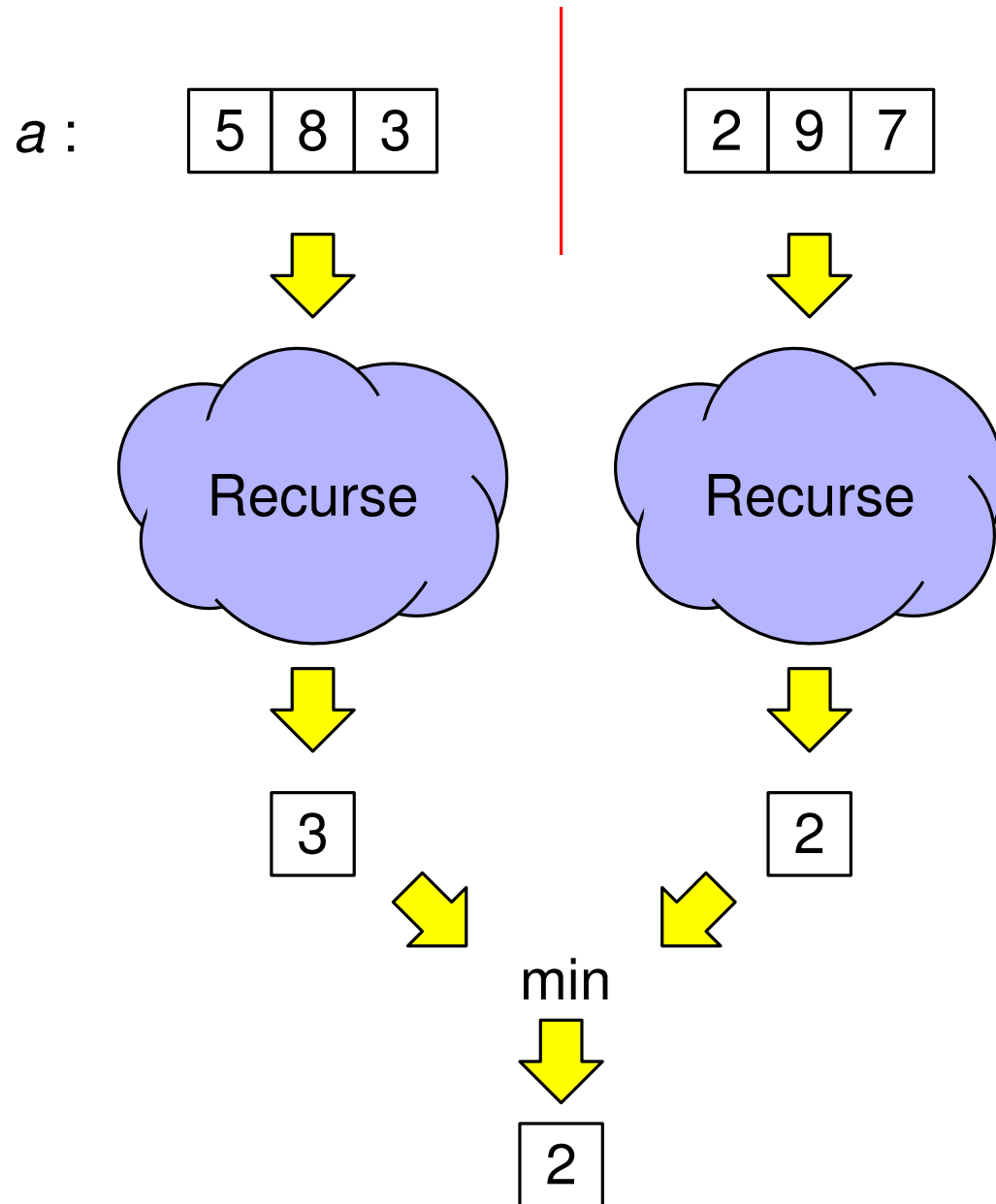
# Finding Minimum: EREW PRAM



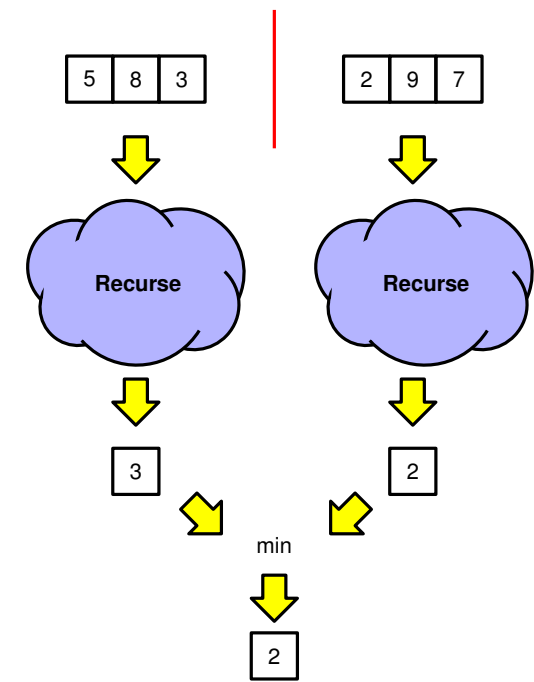
# Finding Minimum: EREW PRAM



# Finding Minimum: EREW PRAM

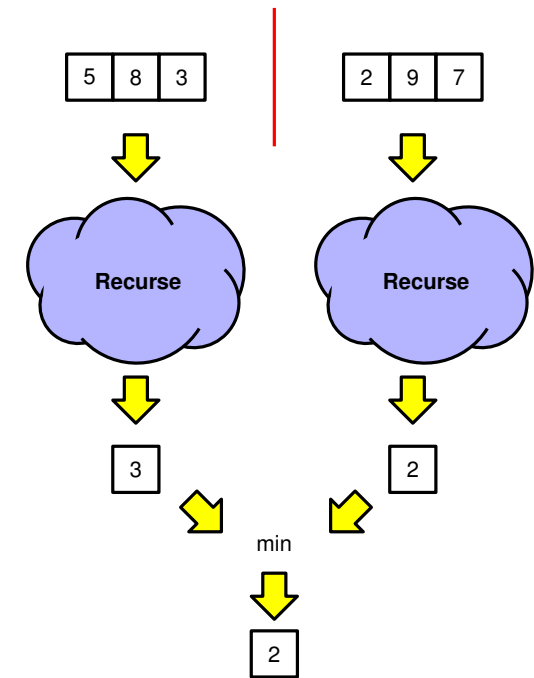


# Finding Minimum: EREW PRAM



# Finding Minimum: EREW PRAM

**procedure** MIN( $a[i..j]$ )



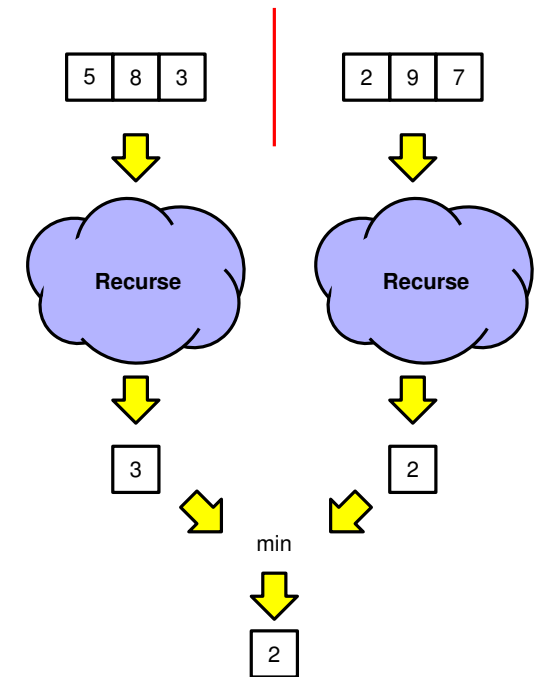
# Finding Minimum: EREW PRAM

**procedure** MIN( $a[i..j]$ )

$$mid = \lfloor \frac{i+j}{2} \rfloor$$

$left = \text{MIN}(a[i..mid])$

$right = \text{MIN}(a[mid + 1..j])$



# Finding Minimum: EREW PRAM

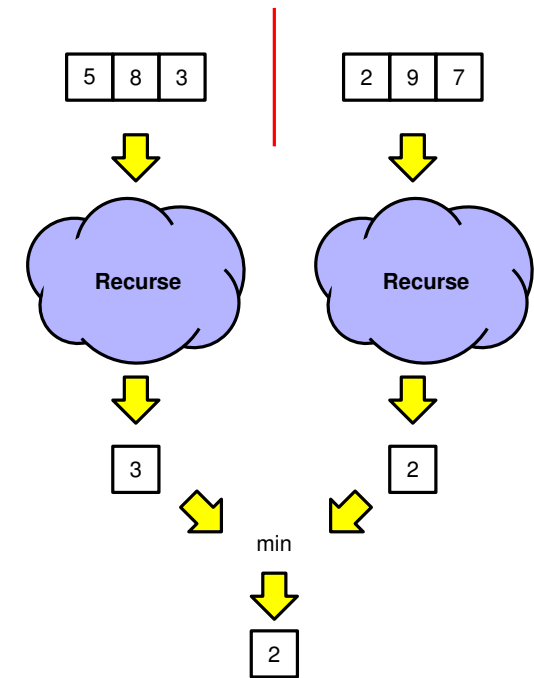
**procedure** MIN( $a[i..j]$ )

$$mid = \lfloor \frac{i+j}{2} \rfloor$$

$left = \text{MIN}(a[i..mid])$

$right = \text{MIN}(a[mid + 1..j])$

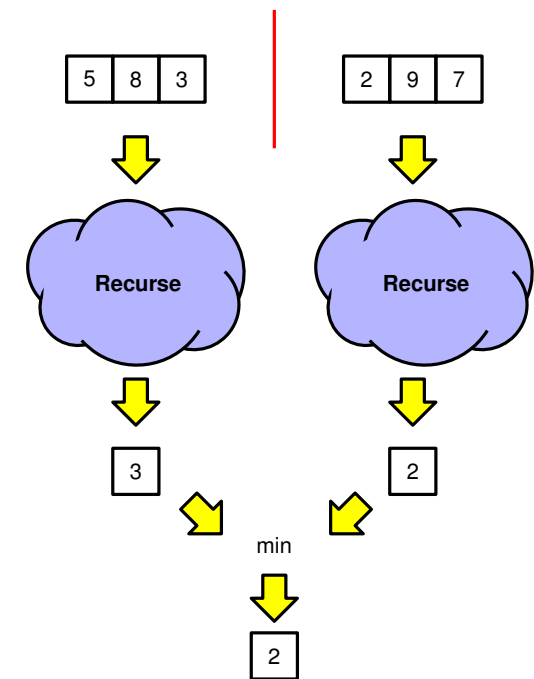
**return** min( $left, right$ )





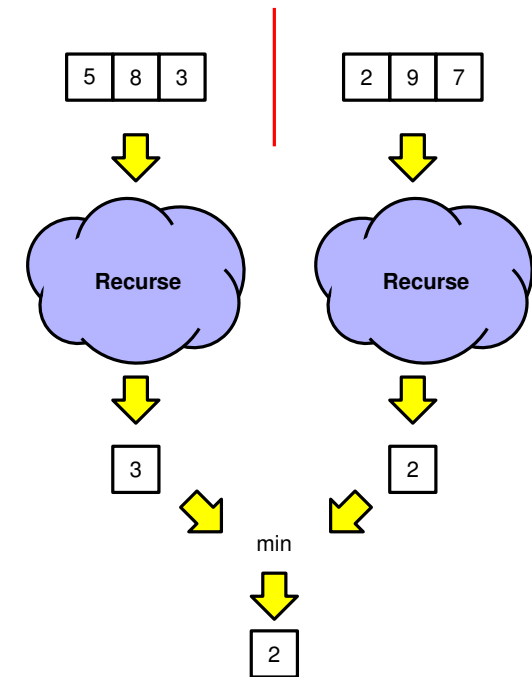
# Finding Minimum: EREW PRAM

```
procedure MIN( $a[i..j]$ )  
  if  $i == j$  then  
    return  $a[i]$   
  else  
     $mid = \lfloor \frac{i+j}{2} \rfloor$   
  
     $left = \text{MIN}(a[i..mid])$   
     $right = \text{MIN}(a[mid + 1..j])$   
    return  $\min(left, right)$ 
```



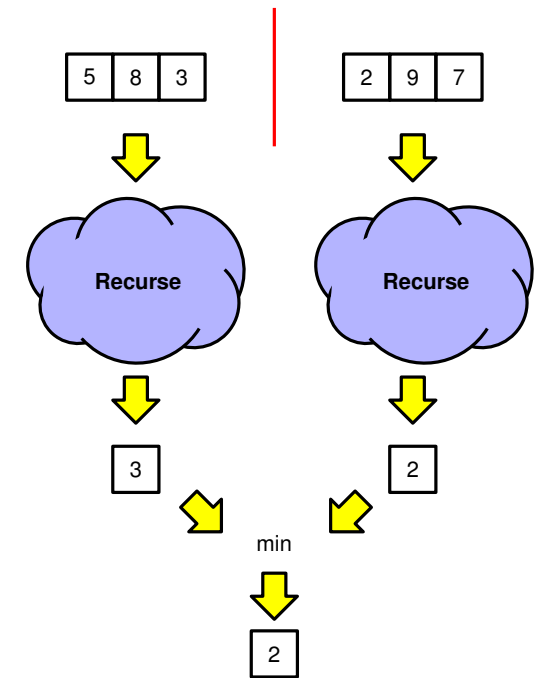
# Finding Minimum: EREW PRAM

```
procedure MIN( $a[i..j]$ )  
  if  $i == j$  then  
    return  $a[i]$   
  else  
     $mid = \lfloor \frac{i+j}{2} \rfloor$   
    in parallel do  
       $left = \text{MIN}(a[i..mid])$   
       $right = \text{MIN}(a[mid + 1..j])$   
    return  $\min(left, right)$ 
```



# Finding Minimum: EREW PRAM

```
procedure MIN( $a[i..j]$ )  
  if  $i == j$  then  
    return  $a[i]$   
  else  
     $mid = \lfloor \frac{i+j}{2} \rfloor$   
    in parallel do  
       $left = \text{MIN}(a[i..mid])$   
       $right = \text{MIN}(a[mid + 1..j])$   
    return  $\min(left, right)$ 
```



Valid EREW?

# Finding Minimum: EREW PRAM

```
procedure MIN( $a[i..j]$ )
```

```
  if  $i == j$  then
```

```
    return  $a[i]$ 
```

```
  else
```

```
     $mid = \lfloor \frac{i+j}{2} \rfloor$ 
```

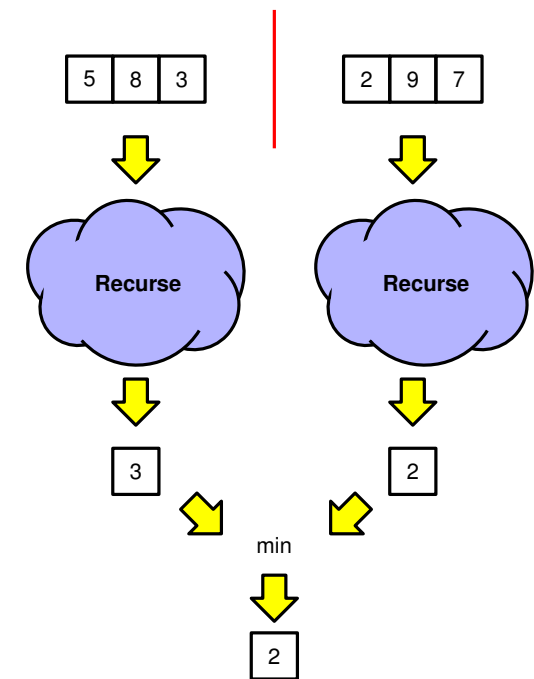
```
    in parallel do
```

```
       $left = MIN(a[i..mid])$ 
```

```
       $right = MIN(a[mid + 1..j])$ 
```

```
    return  $\min(left, right)$ 
```

parallelism



Valid EREW?

# Finding Minimum: EREW PRAM

```
procedure MIN( $a[i..j]$ )
```

```
  if  $i == j$  then
```

```
    return  $a[i]$ 
```

```
  else
```

```
     $mid = \lfloor \frac{i+j}{2} \rfloor$    $rmid = mid + 1$ 
```

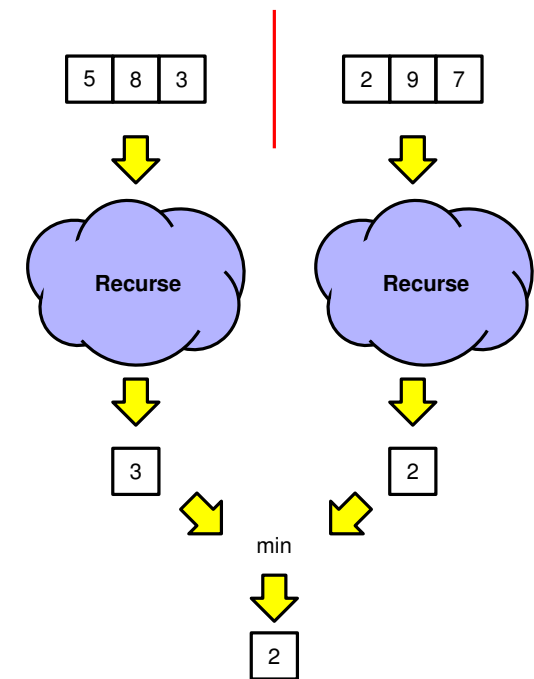
```
    in parallel do
```

```
       $left = MIN(a[i..mid])$ 
```

```
       $right = MIN(a[rmid..j])$ 
```

```
    return  $\min(left, right)$ 
```

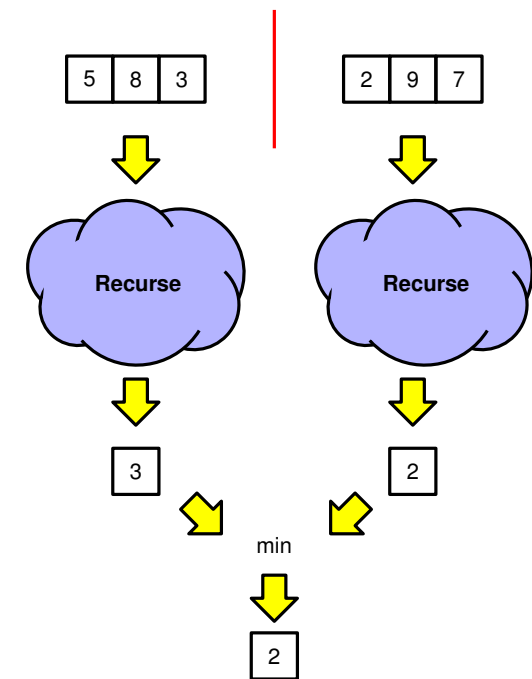
parallelism



Valid EREW?

# Finding Minimum: EREW PRAM

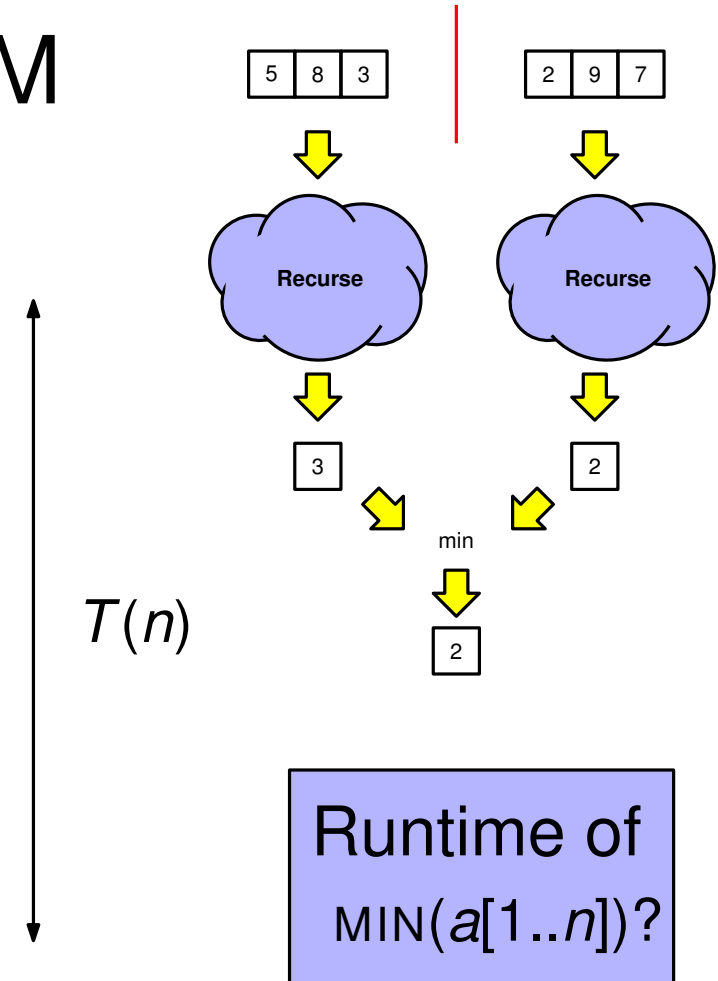
```
procedure MIN( $a[i..j]$ )  
  if  $i == j$  then  
    return  $a[i]$   
  else  
     $mid = \lfloor \frac{i+j}{2} \rfloor$     $rmid = mid + 1$   
    in parallel do  
       $left = MIN(a[i..mid])$   
       $right = MIN(a[rmid..j])$   
    return  $\min(left, right)$ 
```



Runtime of  
 $MIN(a[1..n])$ ?

# Finding Minimum: EREW PRAM

```
procedure MIN( $a[i..j]$ )  
  if  $i == j$  then  
    return  $a[i]$   
  else  
     $mid = \lfloor \frac{i+j}{2} \rfloor$     $rmid = mid + 1$   
    in parallel do  
       $left = MIN(a[i..mid])$   
       $right = MIN(a[rmid..j])$   
    return  $\min(left, right)$ 
```

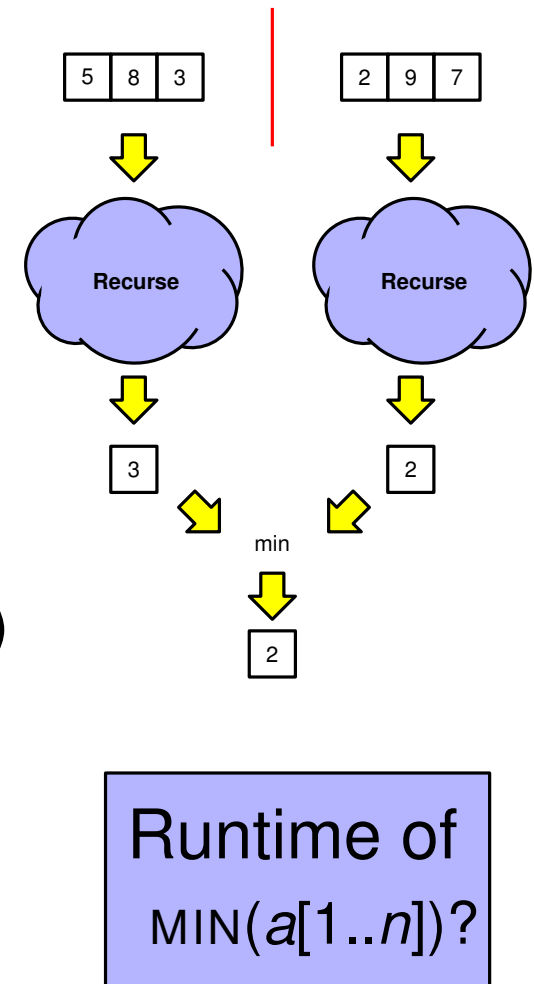


# Finding Minimum: EREW PRAM

```
procedure MIN( $a[i..j]$ )  
  if  $i == j$  then  
    return  $a[i]$   
  else  
     $mid = \lfloor \frac{i+j}{2} \rfloor$   $rmid = mid + 1$   
    in parallel do  
       $left = MIN(a[i..mid])$   
       $right = MIN(a[rmid..j])$   
    return  $\min(left, right)$ 
```

$O(1)$

$T(n)$



Runtime of  
 $MIN(a[1..n])?$



# Finding Minimum: EREW PRAM

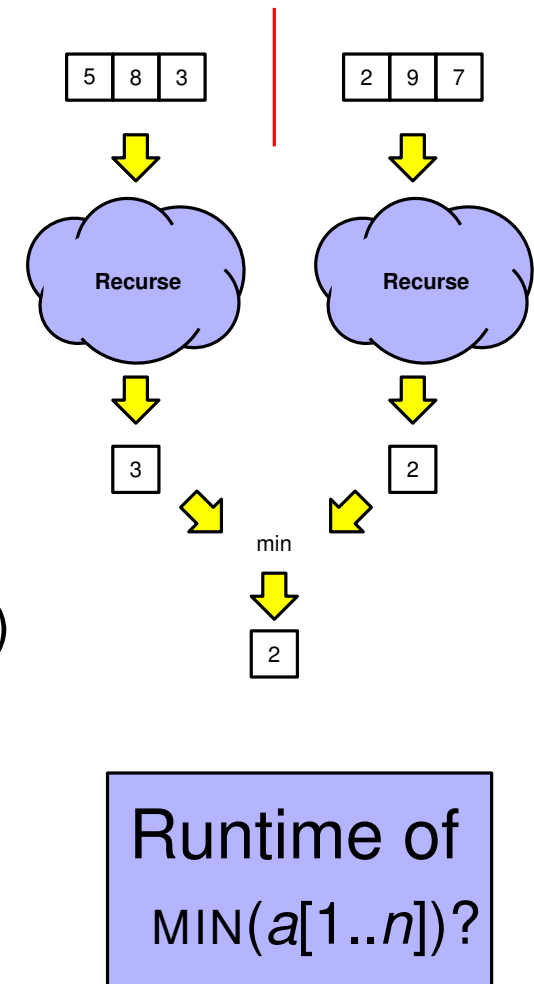
```

procedure MIN( $a[i..j]$ )
  if  $i == j$  then
    return  $a[i]$ 
  else
     $mid = \lfloor \frac{i+j}{2} \rfloor$     $rmid = mid + 1$ 
    in parallel do
       $left = \text{MIN}(a[i..mid])$ 
       $right = \text{MIN}(a[rmid..j])$ 
    return  $\min(left, right)$ 
  
```

$O(1)$

$T(n/2)$

$T(n)$



# Finding Minimum: EREW PRAM

```
procedure MIN( $a[i..j]$ )
```

```
  if  $i == j$  then
```

```
    return  $a[i]$ 
```

```
  else
```

```
     $mid = \lfloor \frac{i+j}{2} \rfloor$    $rmid = mid + 1$ 
```

```
    in parallel do
```

```
       $left = MIN(a[i..mid])$ 
```

```
       $right = MIN(a[rmid..j])$ 
```

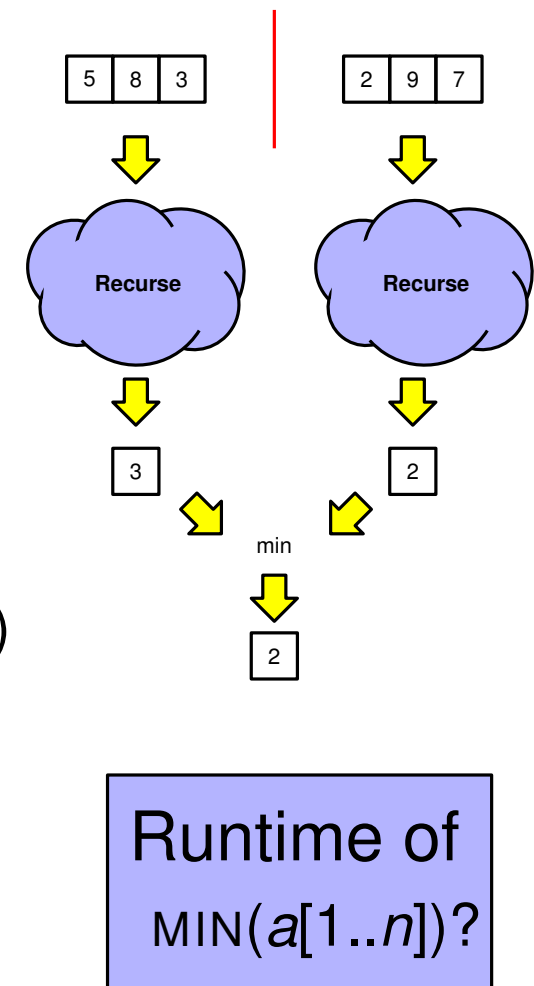
```
    return  $\min(left, right)$ 
```

$O(1)$

$T(n/2)$

$T(n/2)$

$T(n)$



# Finding Minimum: EREW PRAM

```

procedure MIN( $a[i..j]$ )
  if  $i == j$  then
    return  $a[i]$ 
  else
     $mid = \lfloor \frac{i+j}{2} \rfloor$     $rmid = mid + 1$ 
    in parallel do
       $left = \text{MIN}(a[i..mid])$ 
       $right = \text{MIN}(a[rmid..j])$ 
    return  $\min(left, right)$ 
  
```

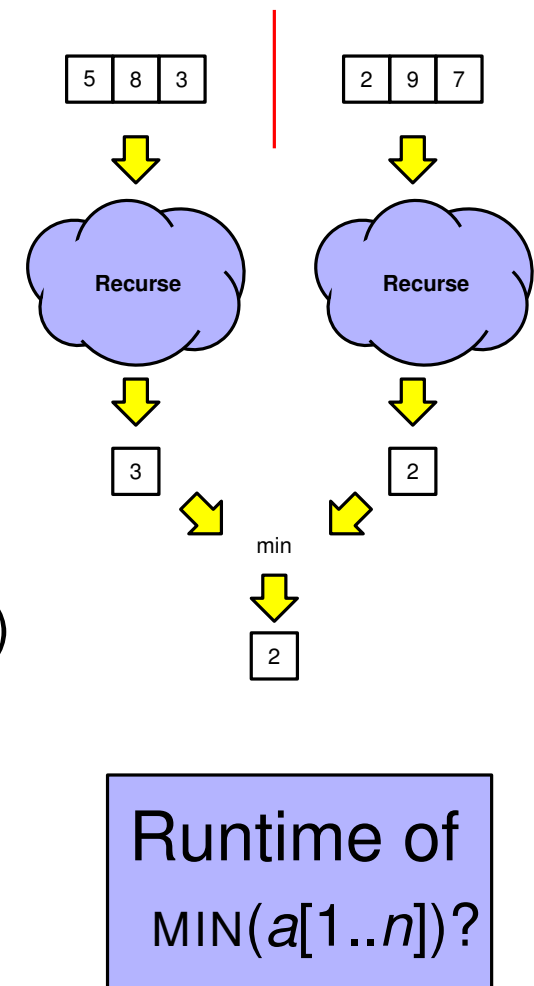
$O(1)$

$T(n/2)$

$T(n/2)$

$O(1)$

$T(n)$



# Finding Minimum: EREW PRAM

```
procedure MIN( $a[i..j]$ )
```

```
  if  $i == j$  then
```

```
    return  $a[i]$ 
```

```
  else
```

```
     $mid = \lfloor \frac{i+j}{2} \rfloor$    $rmid = mid + 1$ 
```

```
    in parallel do
```

```
       $left = MIN(a[i..mid])$ 
```

```
       $right = MIN(a[rmid..j])$ 
```

```
    return  $\min(left, right)$ 
```

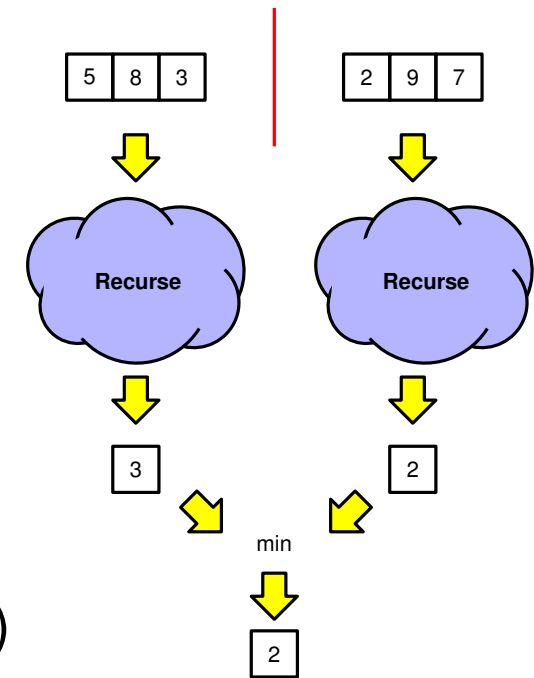
$O(1)$

$T(n/2)$

$T(n/2)$

$O(1)$

$T(n)$



Runtime of  
 $MIN(a[1..n])$ ?

$$T(n) = O(1) + \max \left\{ \begin{array}{l} T(n/2) \\ T(n/2) \end{array} \right\} + O(1)$$

# Finding Minimum: EREW PRAM

```
procedure MIN( $a[i..j]$ )
```

```
  if  $i == j$  then
```

```
    return  $a[i]$ 
```

```
  else
```

```
     $mid = \lfloor \frac{i+j}{2} \rfloor$    $rmid = mid + 1$ 
```

```
    in parallel do
```

```
       $left = MIN(a[i..mid])$ 
```

```
       $right = MIN(a[rmid..j])$ 
```

```
    return  $\min(left, right)$ 
```

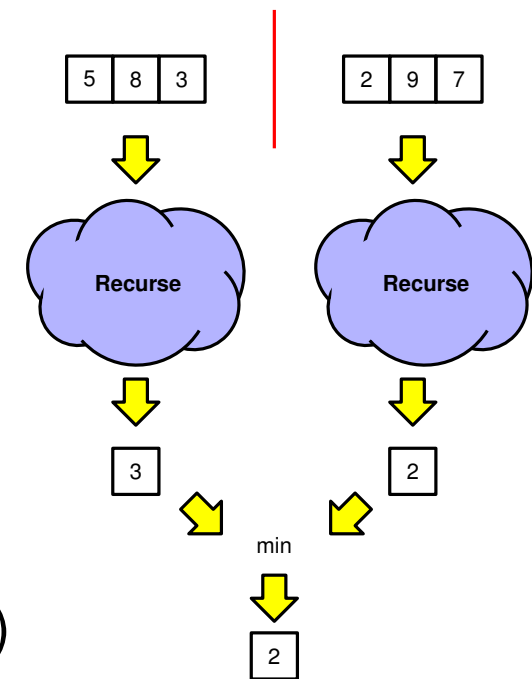
$O(1)$

$T(n/2)$

$T(n/2)$

$O(1)$

$T(n)$



Runtime of  
 $MIN(a[1..n])?$

$$T(n) = O(1) + \max \left\{ \begin{array}{l} T(n/2) \\ T(n/2) \end{array} \right\} + O(1)$$

$$= T(n/2) + O(1)$$

# Finding Minimum: EREW PRAM

```
procedure MIN( $a[i..j]$ )
```

```
  if  $i == j$  then
```

```
    return  $a[i]$ 
```

```
  else
```

```
     $mid = \lfloor \frac{i+j}{2} \rfloor$    $rmid = mid + 1$ 
```

```
    in parallel do
```

```
       $left = MIN(a[i..mid])$ 
```

```
       $right = MIN(a[rmid..j])$ 
```

```
    return  $\min(left, right)$ 
```

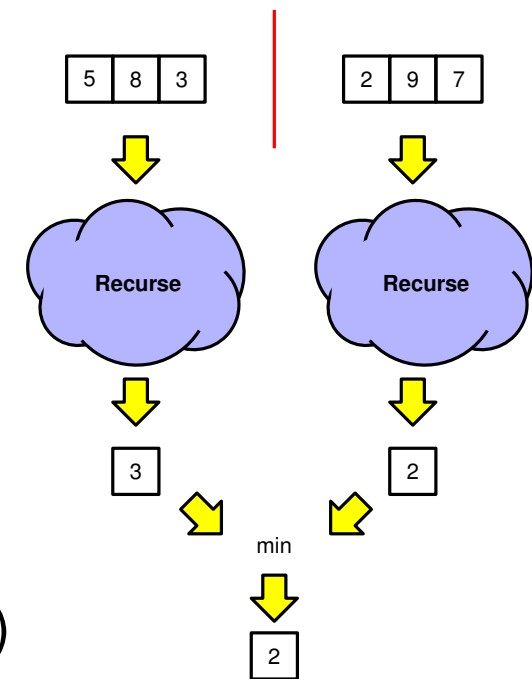
$O(1)$

$T(n/2)$

$T(n/2)$

$O(1)$

$T(n)$



Runtime of  
 $MIN(a[1..n])?$

$$\begin{aligned}
 T(n) &= O(1) + \max \left\{ \begin{array}{l} T(n/2) \\ T(n/2) \end{array} \right\} + O(1) \\
 &= T(n/2) + O(1) \\
 &= \Theta(\log n)
 \end{aligned}$$