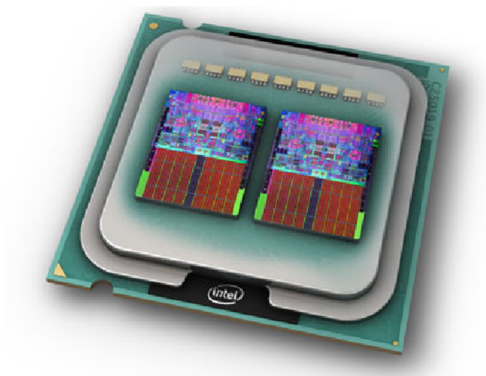
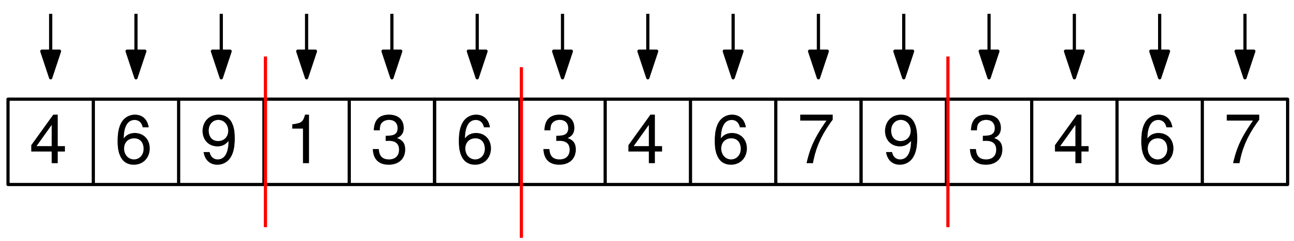
INFORMATION & COMPUTER SCIENCES
UNIVERSITY of HAWAIʻI at MĀNOA

AlgoPARC

ICS 443: Parallel Algorithms

Prof. Nodari Sitchinava

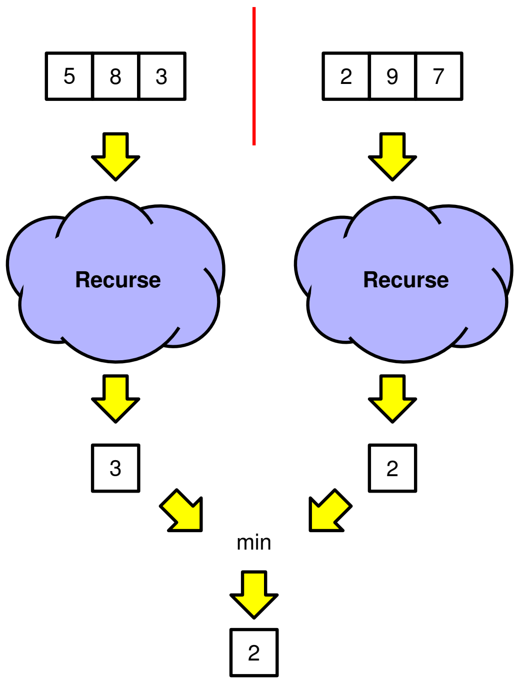




Lecture 8: Finding Minimum

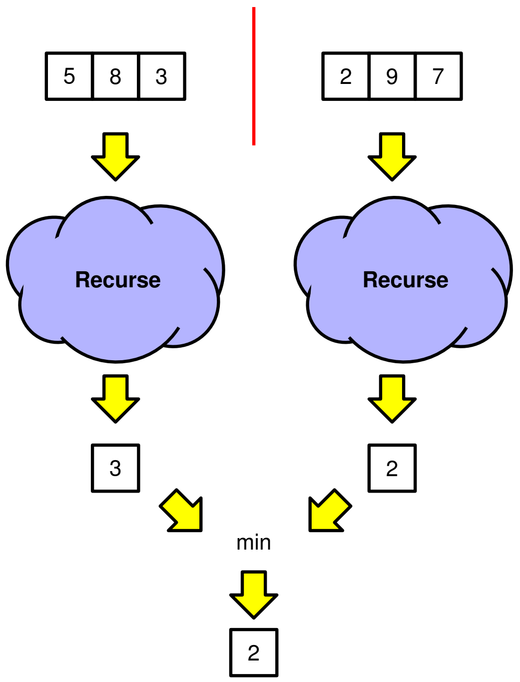
EREW Minimum

procedure EREW-MIN($A[\ell..r]$)
  if $\ell = r$ **then**
    return $A[\ell]$
  $mid = \lfloor \frac{\ell+r}{2} \rfloor$
  in parallel do
    $left$ = EREW-MIN($A[\ell..mid]$)
    $right$ = EREW-MIN($A[mid+1..r]$)
  return min($left, right$)



EREW Minimum

procedure EREW-MIN($A[\ell..r]$)
    if $\ell = r$ **then**
        return $A[\ell]$
    $mid = \lfloor \frac{\ell+r}{2} \rfloor$
    in parallel do
        $left$ = EREW-MIN($A[\ell..mid]$)
        $right$ = EREW-MIN($A[mid+1..r]$)
    return min($left, right$)



$$T(n) = \begin{cases} T(n/2) + O(1) & \text{if } n > 1 \\ O(1) & \text{if } n = 1 \end{cases} = O(\log n)$$

$$W(n) = \begin{cases} 2W(n/2) + O(1) & \text{if } n > 1 \\ O(1) & \text{if } n = 1 \end{cases} = O(n)$$

Common-CRCW Minimum

a	5	8	3	2	9	7

Common-CRCW Minimum

a	5	8	3	2	9	7
5						
8						
3						
2						
9						
7						

Common-CRCW Minimum

a	5	8	3	2	9	7
5						
8						
3						
2						
9						
7						

M

Common-CRCW Minimum

a	5	8	3	2	9	7
5						
8						
3						
2						
9						
7						

M

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

Common-CRCW Minimum

a	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

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

Common-CRCW Minimum

a	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

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

Common-CRCW Minimum

a	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

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

Common-CRCW Minimum

a	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
            $M[row, col] = 0$

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

Common-CRCW Minimum

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 $a[row] > a[col]$ **then**
      $M[row, col] = 1$
    else
      $M[row, col] = 0$

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

Common-CRCW Minimum

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[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
```

Common-CRCW Minimum

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**
  $x[i] = 0$

Common-CRCW Minimum

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 $M[row, col] == 1$ **then**
            $x[row] = 1$

Common-CRCW Minimum

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 $M[row, col] == 1$ **then**
            $x[row] = 1$

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

Common-CRCW Minimum

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 M[row, col] == 1 then
            x[row] = 1
```

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

Common-CRCW Minimum

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 M[row, col] == 1 then
            x[row] = 1
```

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

Common-CRCW Minimum

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

Analysis:

```
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 M[row, col] == 1 then
            x[row] = 1
```

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

Common-CRCW Minimum

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

Analysis:

$T(n) = O(1)$

$W(n) = O(n^2)$

```
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 M[row, col] == 1 then
            x[row] = 1
```

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

Common-CRCW Minimum

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

Analysis:

$T(n) = O(1)$

$W(n) = O(n^2)$

```
procedure FAST-MIN(A[1..n])
    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, co    0
    allocate new ar        ..n]
    for i = 1 t        arallel do
        x[
    for r    w = 1 to n in parallel do
        for col = 1 to n in parallel do
            if M[row, col] == 1 then
                x[row] = 1
    for row = 1 to n in parallel do
        if x[row] == 0 then
            min = a[row]
```

Not work-efficient

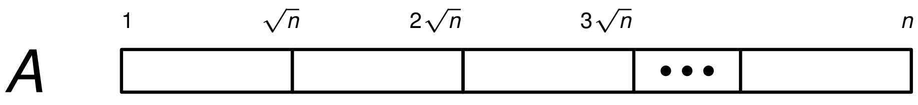
More Efficient Common-CRCW Minimum

More Efficient Common-CRCW Minimum

procedure LL-MIN($A[\ell..r]$)
    $n = r - \ell + 1$
    if $n = 1$ **then**
        return $A[\ell]$
    B = new array of size $k = \sqrt{n}$
    for $i = 1$ to k **in parallel do**
        $\ell' = \ell + k \cdot (i - 1)$
        $r' = \ell + k \cdot i - 1$
                        $\triangleright A_i = A[\ell'..r']$
        $B[i]$ = LL-MIN($A[\ell'..r']$)
    return FAST-MIN($B[1..k]$)

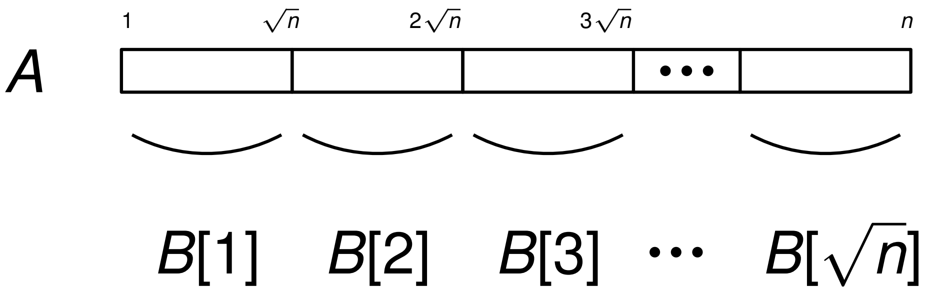
More Efficient Common-CRCW Minimum

procedure LL-MIN($A[\ell..r]$)
    $n = r - \ell + 1$
    if $n = 1$ **then**
        return $A[\ell]$
    B = new array of size $k = \sqrt{n}$
    for $i = 1$ to k **in parallel do**
        $\ell' = \ell + k \cdot (i - 1)$
        $r' = \ell + k \cdot i - 1$
                    $\triangleright A_i = A[\ell'..r']$
        $B[i]$ = LL-MIN($A[\ell'..r']$)
    return FAST-MIN($B[1..k]$)



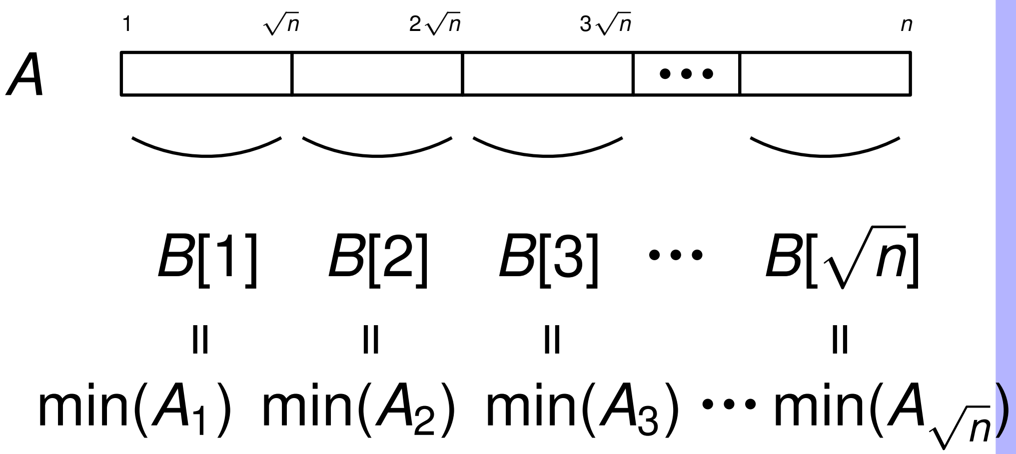
More Efficient Common-CRCW Minimum

procedure LL-MIN($A[\ell..r]$)
    $n = r - \ell + 1$
    if $n = 1$ **then**
        return $A[\ell]$
    B = new array of size $k = \sqrt{n}$
    for $i = 1$ to k **in parallel do**
        $\ell' = \ell + k \cdot (i - 1)$
        $r' = \ell + k \cdot i - 1$
                        $\triangleright A_i = A[\ell'..r']$
        $B[i]$ = LL-MIN($A[\ell'..r']$)
    return FAST-MIN($B[1..k]$)



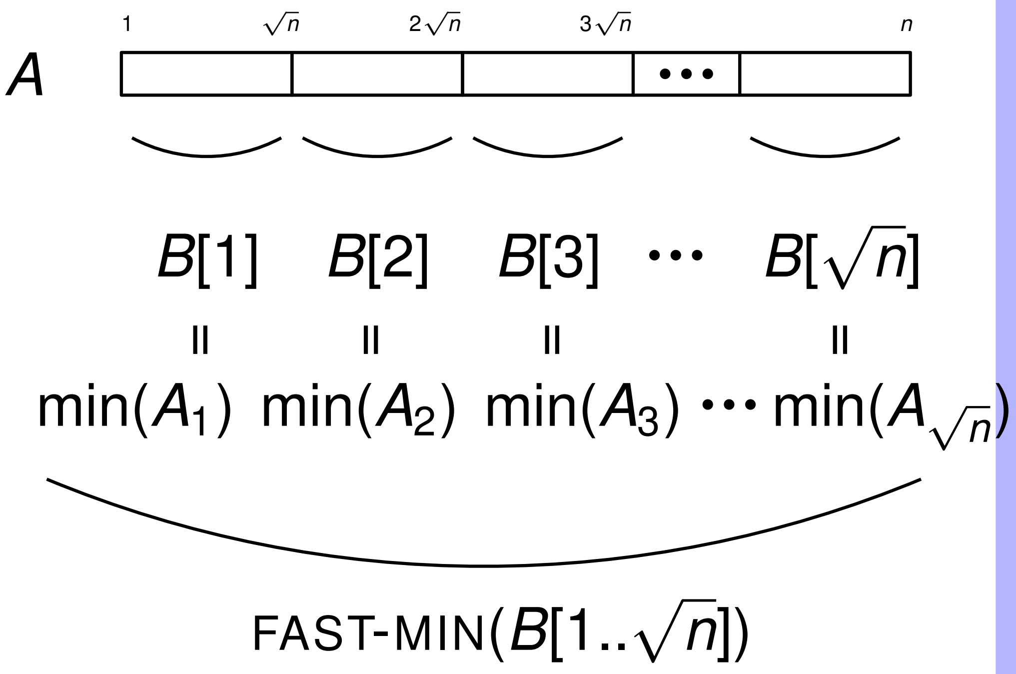
More Efficient Common-CRCW Minimum

procedure LL-MIN($A[\ell..r]$)
  $n = r - \ell + 1$
  if $n = 1$ **then**
    return $A[\ell]$
  B = new array of size $k = \sqrt{n}$
  for $i = 1$ to k **in parallel do**
    $\ell' = \ell + k \cdot (i - 1)$
    $r' = \ell + k \cdot i - 1$
      $\triangleright A_i = A[\ell'..r']$
    $B[i] = $ LL-MIN($A[\ell'..r']$)
  return FAST-MIN($B[1..k]$)



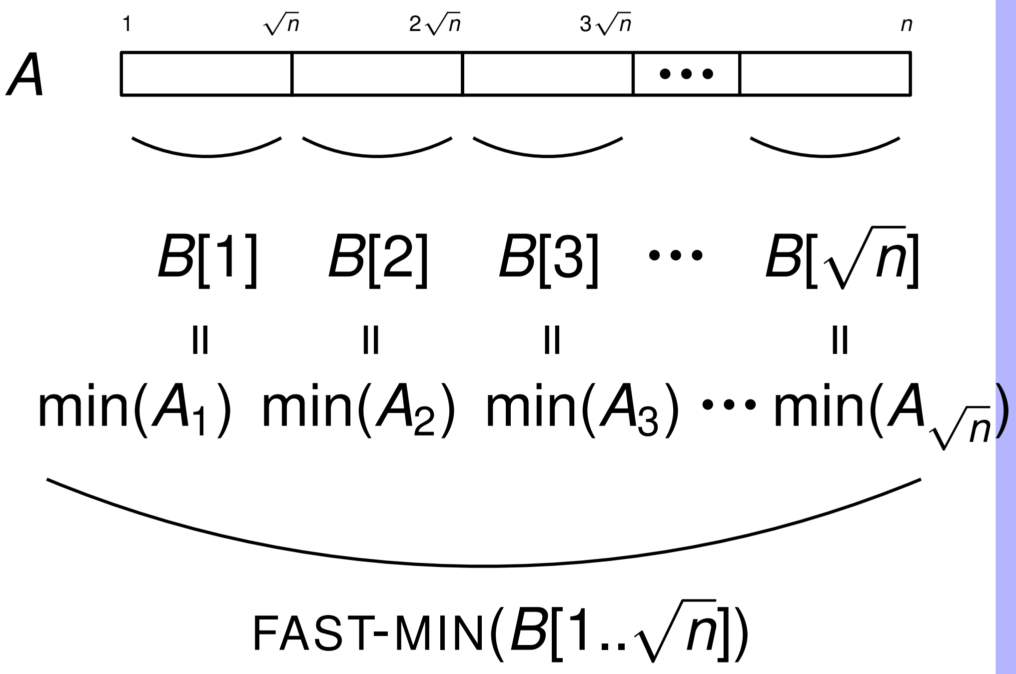
More Efficient Common-CRCW Minimum

procedure LL-MIN($A[\ell..r]$)
  $n = r - \ell + 1$
  if $n = 1$ **then**
    return $A[\ell]$
  B = new array of size $k = \sqrt{n}$
  for $i = 1$ to k **in parallel do**
    $\ell' = \ell + k \cdot (i - 1)$
    $r' = \ell + k \cdot i - 1$
        $\triangleright A_i = A[\ell'..r']$
    $B[i]$ = LL-MIN($A[\ell'..r']$)
  return FAST-MIN($B[1..k]$)



More Efficient Common-CRCW Minimum

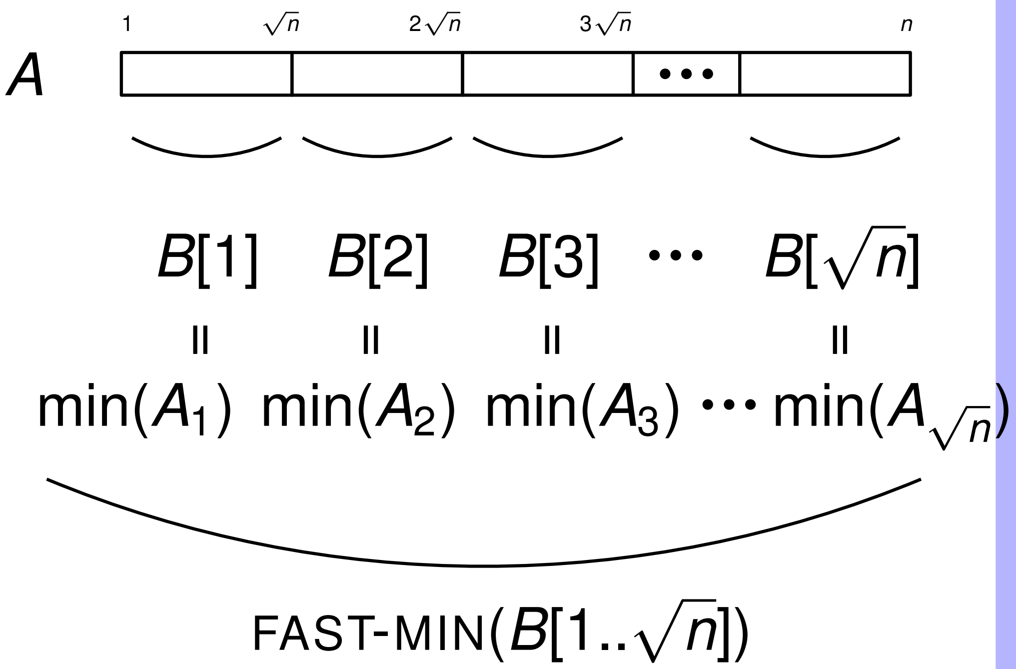
procedure LL-MIN($A[\ell..r]$)
  $n = r - \ell + 1$
  if $n = 1$ **then**
    return $A[\ell]$
  B = new array of size $k = \sqrt{n}$
  for $i = 1$ to k **in parallel do**
    $\ell' = \ell + k \cdot (i - 1)$
    $r' = \ell + k \cdot i - 1$
      $\triangleright A_i = A[\ell'..r']$
    $B[i]$ = LL-MIN($A[\ell'..r']$)
  return FAST-MIN($B[1..k]$)



Analysis

More Efficient Common-CRCW Minimum

procedure LL-MIN($A[\ell..r]$)
  $n = r - \ell + 1$
  if $n = 1$ **then**
    return $A[\ell]$
  B = new array of size $k = \sqrt{n}$
  for $i = 1$ to k **in parallel do**
    $\ell' = \ell + k \cdot (i - 1)$
    $r' = \ell + k \cdot i - 1$
      $\triangleright A_i = A[\ell'..r']$
    $B[i]$ = LL-MIN($A[\ell'..r']$)
  return FAST-MIN($B[1..k]$)

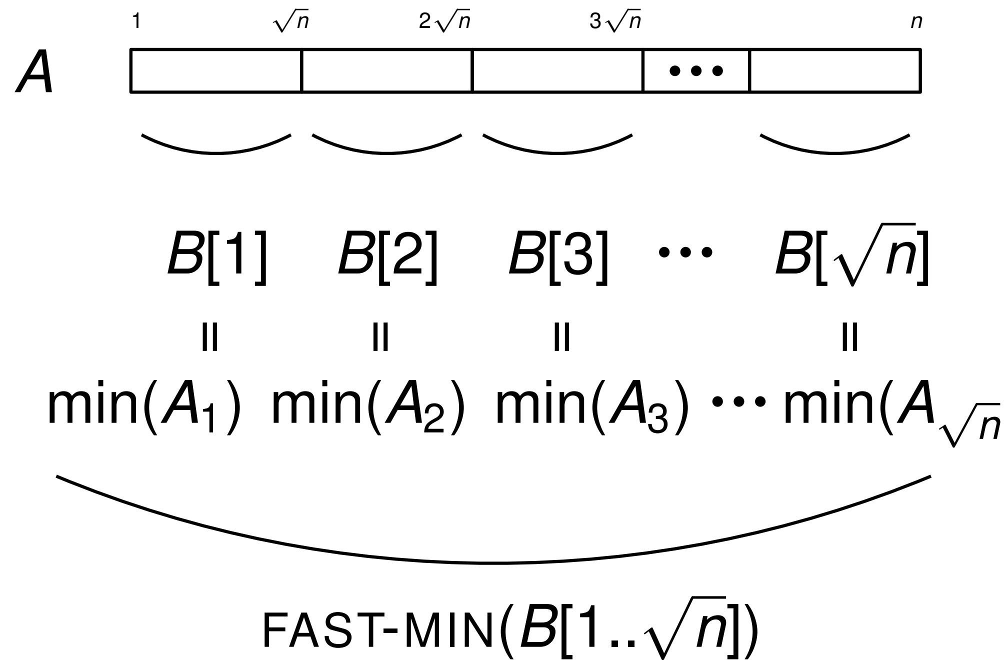


Analysis

$T(n) =$

More Efficient Common-CRCW Minimum

procedure LL-MIN($A[\ell..r]$)
  $n = r - \ell + 1$
  if $n = 1$ **then**
    return $A[\ell]$
  B = new array of size $k = \sqrt{n}$
  for $i = 1$ to k **in parallel do**
    $\ell' = \ell + k \cdot (i - 1)$
    $r' = \ell + k \cdot i - 1$
      $\triangleright A_i = A[\ell'..r']$
    $B[i]$ = LL-MIN($A[\ell'..r']$) ← recursion
  return FAST-MIN($B[1..k]$)



Analysis

$T(n) =$

More Efficient Common-CRCW Minimum

procedure LL-MIN($A[\ell..r]$)

  $n = r - \ell + 1$

  if $n = 1$ **then**

    return $A[\ell]$

  B = new array of size $k = \sqrt{n}$

  for $i = 1$ to k **in parallel do**

    $\ell' = \ell + k \cdot (i - 1)$

    $r' = \ell + k \cdot i - 1$

      $\triangleright A_i = A[\ell'..r']$

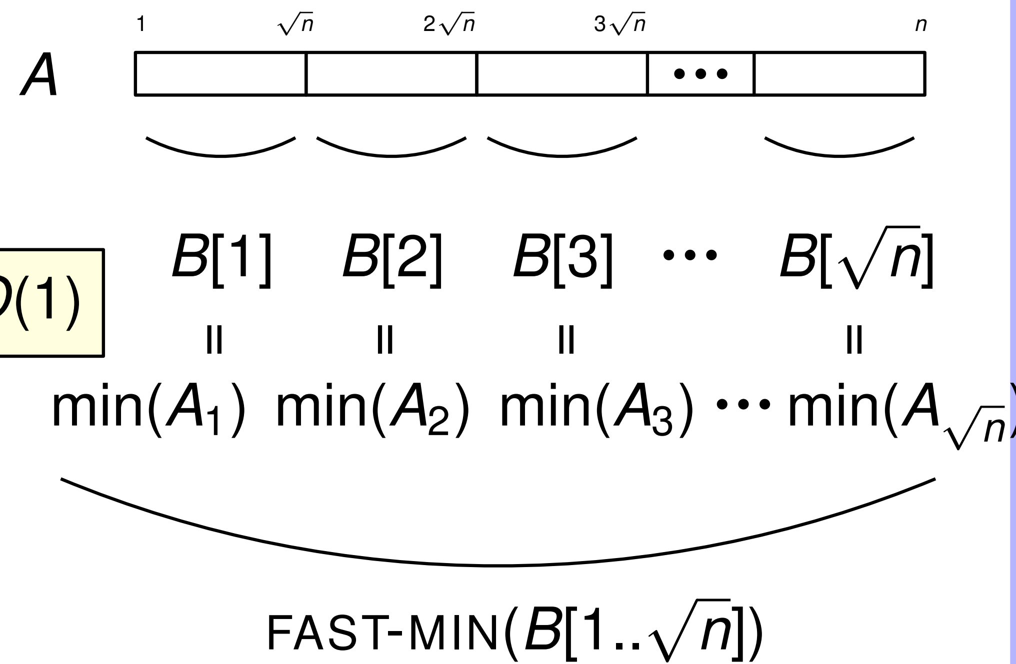
    $B[i]$ = LL-MIN($A[\ell'..r']$)

  return FAST-MIN($B[1..k]$)

$O(1)$

recursion

$O(1)$



Analysis

$T(n) =$

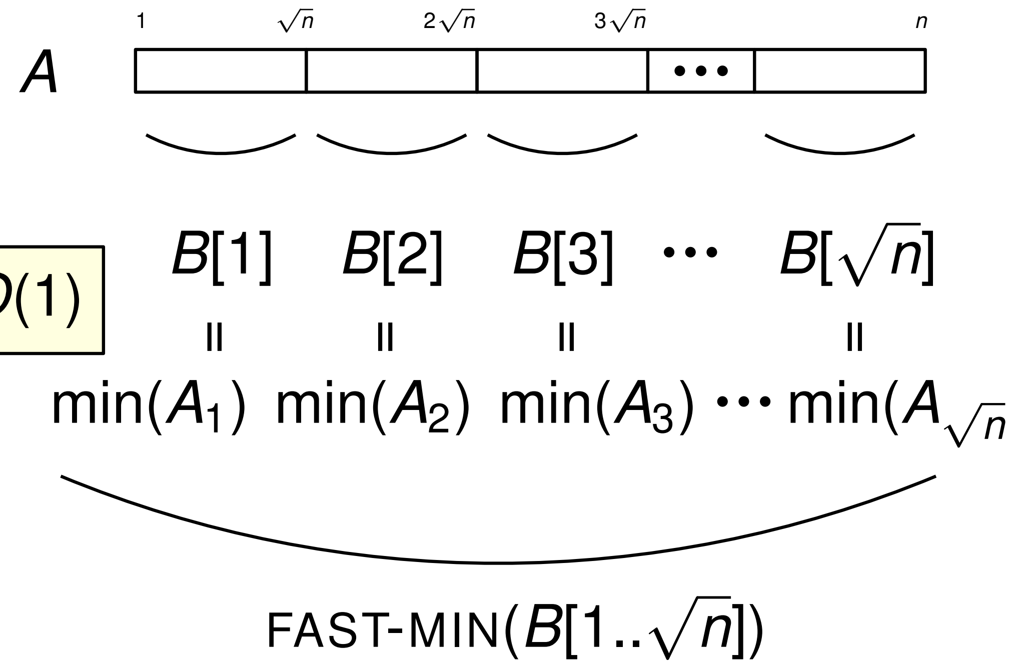
More Efficient Common-CRCW Minimum

procedure LL-MIN($A[\ell..r]$)
    $n = r - \ell + 1$
    if $n = 1$ **then**
        return $A[\ell]$
    B = new array of size $k = \sqrt{n}$
    for $i = 1$ to k **in parallel do**
        $\ell' = \ell + k \cdot (i - 1)$
        $r' = \ell + k \cdot i - 1$
            $\triangleright A_i = A[\ell'..r']$
        $B[i]$ = LL-MIN($A[\ell'..r']$)
    return FAST-MIN($B[1..k]$)

$O(1)$

recursion

$O(1)$

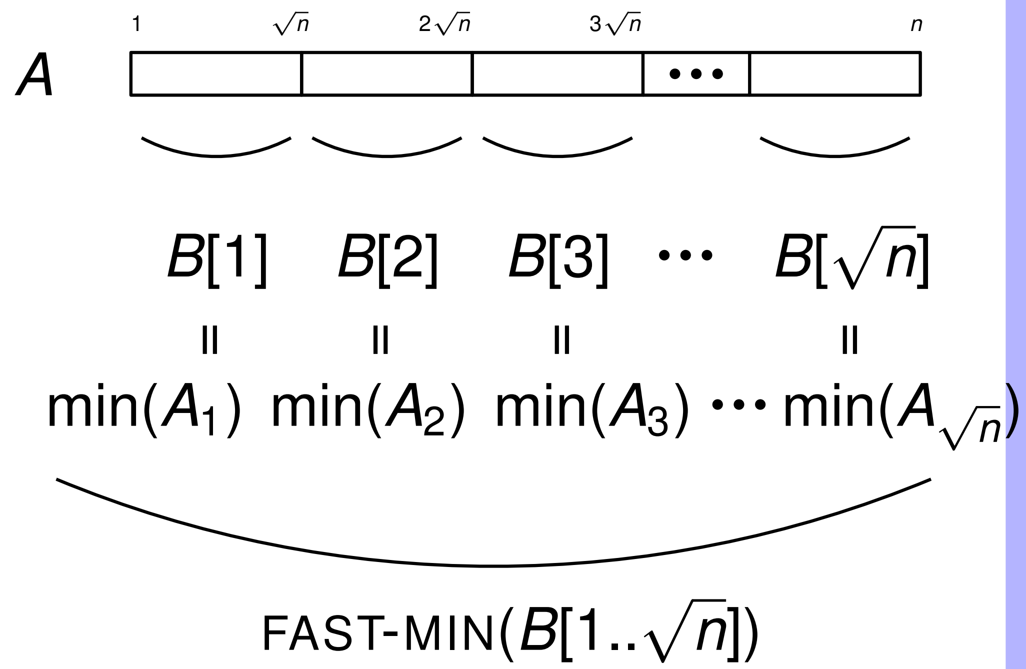


Analysis

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

More Efficient Common-CRCW Minimum

procedure LL-MIN($A[\ell..r]$)
    $n = r - \ell + 1$
    if $n = 1$ **then**
        return $A[\ell]$
    B = new array of size $k = \sqrt{n}$
    for $i = 1$ to k **in parallel do**
        $\ell' = \ell + k \cdot (i - 1)$
        $r' = \ell + k \cdot i - 1$
            $\triangleright A_i = A[\ell'..r']$
        $B[i]$ = LL-MIN($A[\ell'..r']$)
    return FAST-MIN($B[1..k]$)



Analysis

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

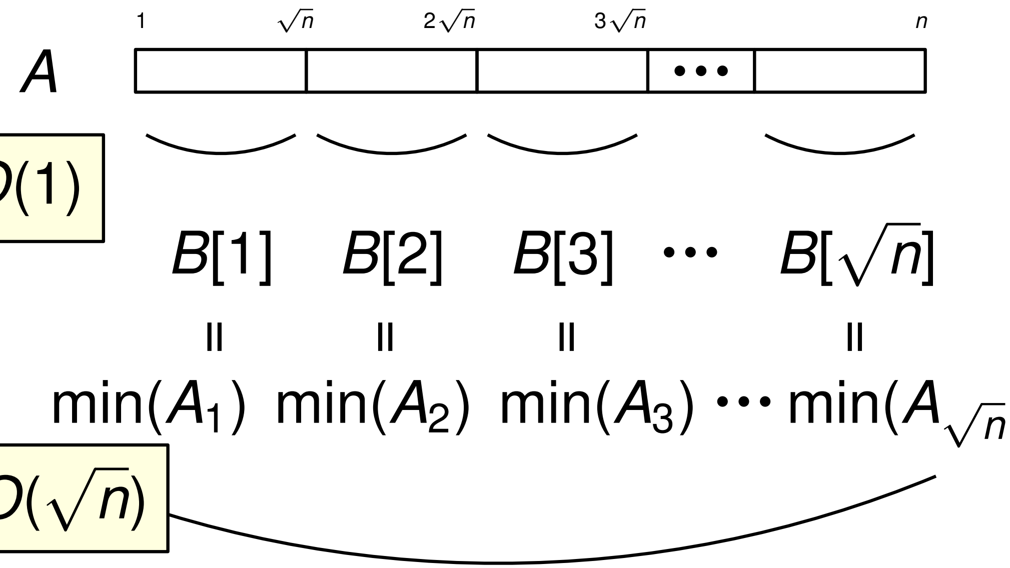
$W(n) =$

More Efficient Common-CRCW Minimum

procedure LL-MIN($A[\ell..r]$)
  $n = r - \ell + 1$
  if $n = 1$ **then**
    return $A[\ell]$
  B = new array of size $k = \sqrt{n}$
  for $i = 1$ to k **in parallel do**
    $\ell' = \ell + k \cdot (i - 1)$
    $r' = \ell + k \cdot i - 1$
      $\triangleright A_i = A[\ell'..r']$
    $B[i]$ = LL-MIN($A[\ell'..r']$)
  return FAST-MIN($B[1..k]$)

$O(1)$

$O(\sqrt{n})$



FAST-MIN($B[1..\sqrt{n}]$)

recursion $\times \sqrt{n}$

$O\left(\left(\sqrt{n}\right)^2\right) = O(n)$

Analysis

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

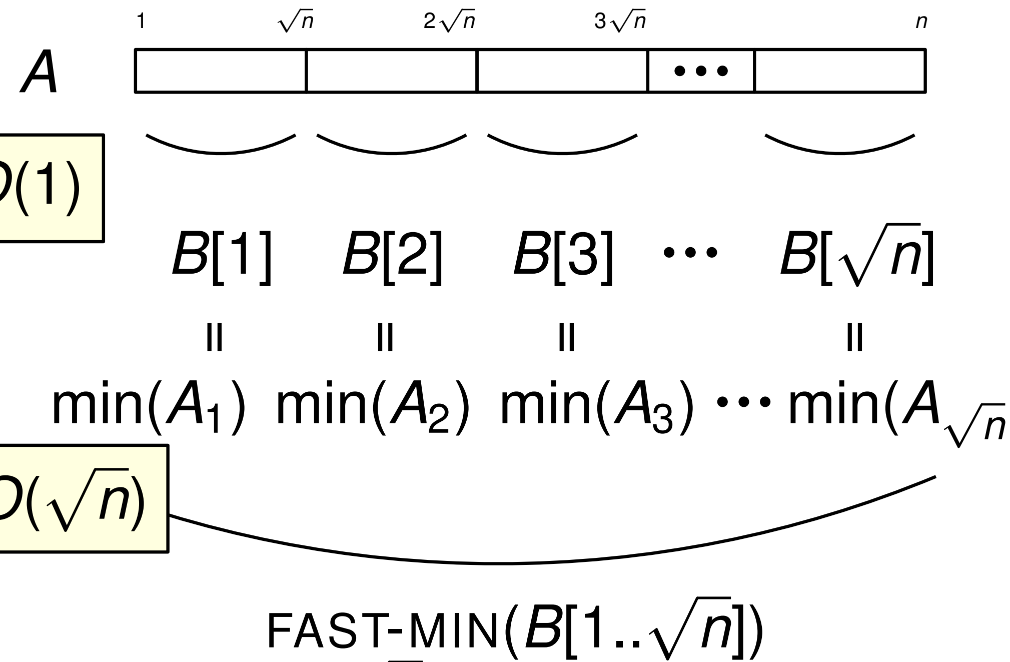
$W(n) =$

More Efficient Common-CRCW Minimum

```
procedure LL-MIN(A[ℓ..r])
    n = r − ℓ + 1
    if n = 1 then
        return A[ℓ]
    B = new array of size k = √n
    for i = 1 to k in parallel do
        ℓ′ = ℓ + k · (i − 1)
        r′ = ℓ + k · i − 1
                        ▷ A_i = A[ℓ′..r′]
        B[i] = LL-MIN(A[ℓ′..r′])
    return FAST-MIN(B[1..k])
```

$O(1)$

$O(\sqrt{n})$



recursion $\times \sqrt{n}$

$O\left(\left(\sqrt{n}\right)^2\right) = O(n)$

Analysis

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

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

Solving Recurrences

```
procedure LL-MIN(A[ℓ..r])
    n = r − ℓ + 1
    if n = 1 then
        return A[ℓ]
    B = new array of size k = √n
    for i = 1 to k in parallel do
        ℓ′ = ℓ + k · (i − 1)
        r′ = ℓ + k · i − 1
                            ▷ A_i = A[ℓ′..r′]
        B[i] = LL-MIN(A[ℓ′..r′])
    return FAST-MIN(B[1..k])
```

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

Solving Recurrences

procedure LL-MIN($A[\ell..r]$)
  $n = r - \ell + 1$
  if $n = 1$ **then**
    return $A[\ell]$
  B = new array of size $k = \sqrt{n}$
  for $i = 1$ to k **in parallel do**
    $\ell' = \ell + k \cdot (i - 1)$
    $r' = \ell + k \cdot i - 1$
    $\triangleright A_i = A[\ell'..r']$
    $B[i] =$ LL-MIN($A[\ell'..r']$)
  return FAST-MIN($B[1..k]$)

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

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

Solving Recurrences

procedure LL-MIN($A[\ell..r]$)
  $n = r - \ell + 1$
  if $n = 1$ **then**
    return $A[\ell]$
  B = new array of size $k = \sqrt{n}$
  for $i = 1$ to k **in parallel do**
    $\ell' = \ell + k \cdot (i - 1)$
    $r' = \ell + k \cdot i - 1$
        $\triangleright A_i = A[\ell'..r']$
    $B[i]$ = LL-MIN($A[\ell'..r']$)
  return FAST-MIN($B[1..k]$)

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

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

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

Solving Recurrences

procedure LL-MIN($A[\ell..r]$)
    $n = r - \ell + 1$
    if $n = 1$ **then**
        return $A[\ell]$
    B = new array of size $k = \sqrt{n}$
    for $i = 1$ to k **in parallel do**
        $\ell' = \ell + k \cdot (i - 1)$
        $r' = \ell + k \cdot i - 1$
                $\triangleright A_i = A[\ell'..r']$
        $B[i]$ = LL-MIN($A[\ell'..r']$)
    return FAST-MIN($B[1..k]$)

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

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

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

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

Solving Recurrences

procedure LL-MIN($A[\ell..r]$)
    $n = r - \ell + 1$
    if $n = 1$ **then**
        return $A[\ell]$
    B = new array of size $k = \sqrt{n}$
    for $i = 1$ to k **in pa**[illegible] **do**
        $\ell' = \ell + k$ [illegible]
        $r' = \ell$ [illegible] $- 1$
        $\triangleright A_i = A[\ell'..r']$
        $B[i]$ = LL-MIN($A[\ell'..r']$)
    return FAST-MIN($B[1..k]$)

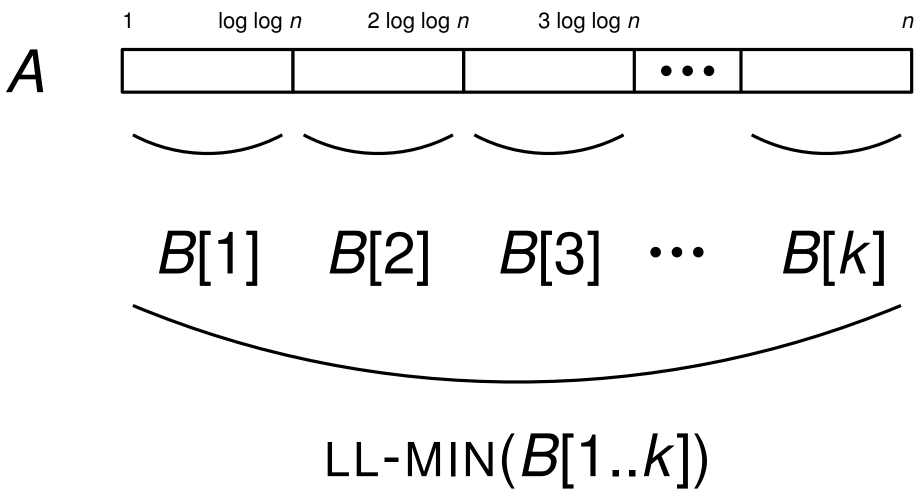
Not work-efficient

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

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

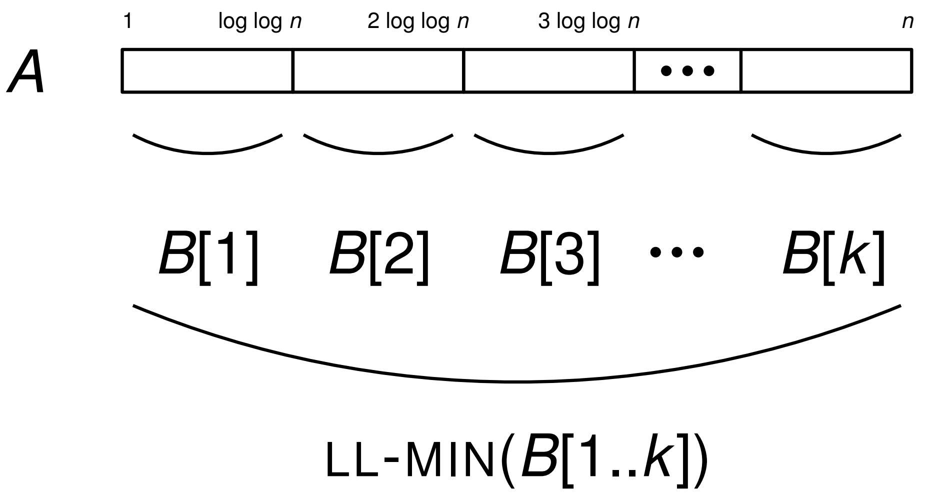
Work-Efficient Common-CRCW Minimum

procedure CRCW-MIN($A[1..n]$)
  B = new array of size $k = \frac{n}{\log\log n}$
  for $i = 1$ to k **in parallel do**
    $\ell = 1 + k \cdot (i - 1)$
    $r = k \cdot i$
    $B[i]$ = SEQ-MIN($A[\ell..r]$)
  return LL-MIN($B[1..k]$)



Work-Efficient Common-CRCW Minimum

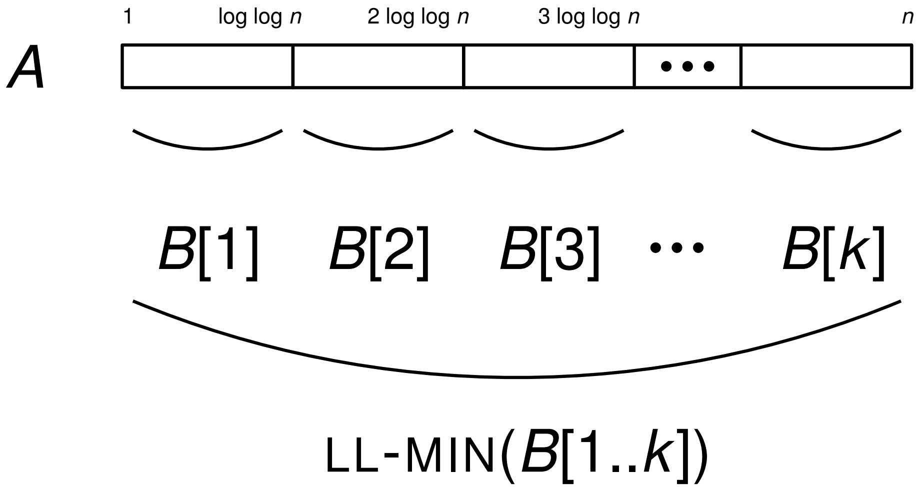
procedure CRCW-MIN($A[1..n]$)
    B = new array of size $k = \frac{n}{\log\log n}$
    for $i = 1$ to k **in parallel do**
        $\ell = 1 + k \cdot (i - 1)$
        $r = k \cdot i$
        $B[i]$ = SEQ-MIN($A[\ell..r]$)
    return LL-MIN($B[1..k]$)



Analysis

Work-Efficient Common-CRCW Minimum

procedure CRCW-MIN($A[1..n]$)
  B = new array of size $k = \frac{n}{\log\log n}$
  for $i = 1$ to k **in parallel do**
    $\ell = 1 + k \cdot (i - 1)$
    $r = k \cdot i$
    $B[i]$ = SEQ-MIN($A[\ell..r]$)
  return LL-MIN($B[1..k]$)

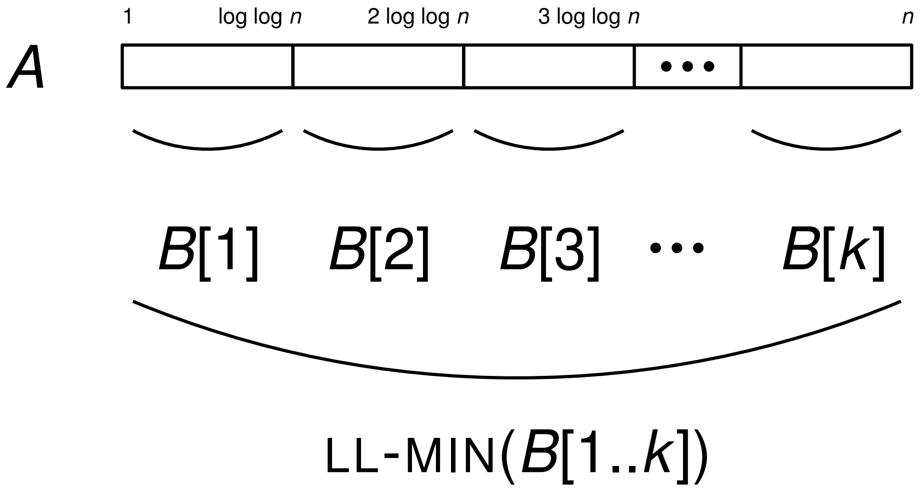


Analysis

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

Work-Efficient Common-CRCW Minimum

procedure CRCW-MIN($A[1..n]$)
  B = new array of size $k = \frac{n}{\log\log n}$
  for $i = 1$ to k **in parallel do**
    $\ell = 1 + k \cdot (i - 1)$
    $r = k \cdot i$
    $B[i]$ = SEQ-MIN($A[\ell..r]$)
  return LL-MIN($B[1..k]$)

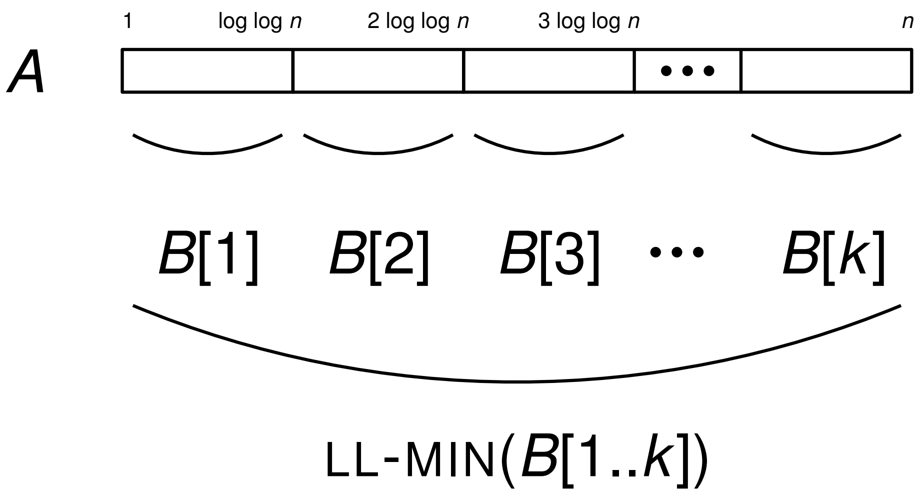


Analysis

$$T(n) = O(\log\log n) + O\left(\log\log \frac{n}{\log\log n}\right)$$
$$= O(\log\log n) + O\left(\log\log n - \log^{(4)} n\right)$$

Work-Efficient Common-CRCW Minimum

procedure CRCW-MIN($A[1..n]$)
  B = new array of size $k = \frac{n}{\log\log n}$
  for $i = 1$ to k **in parallel do**
    $\ell = 1 + k \cdot (i - 1)$
    $r = k \cdot i$
    $B[i]$ = SEQ-MIN($A[\ell..r]$)
  return LL-MIN($B[1..k]$)

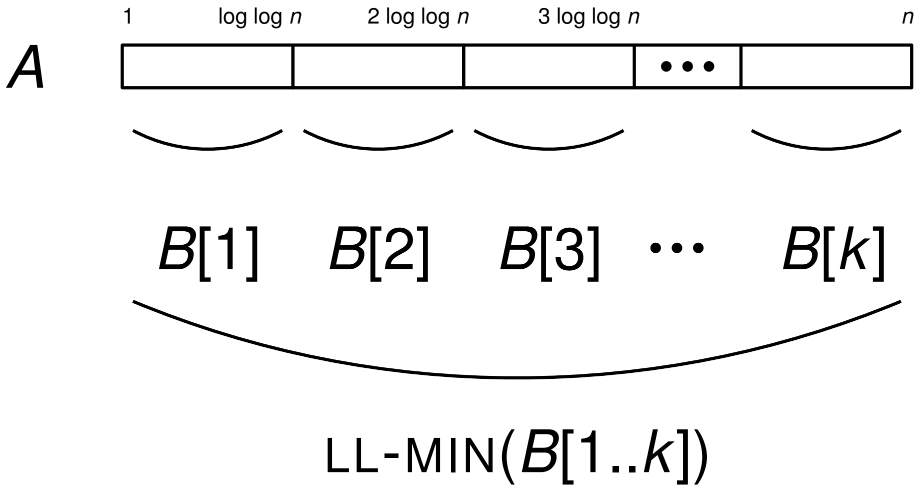


Analysis

$$T(n) = O(\log\log n) + O\left(\log\log \frac{n}{\log\log n}\right)$$
$$= O(\log\log n) + O\left(\log\log n - \log^{(4)} n\right) = O(\log\log n)$$

Work-Efficient Common-CRCW Minimum

procedure CRCW-MIN($A[1..n]$)
  B = new array of size $k = \frac{n}{\log\log n}$
  for $i = 1$ to k **in parallel do**
    $\ell = 1 + k \cdot (i - 1)$
    $r = k \cdot i$
    $B[i]$ = SEQ-MIN($A[\ell..r]$)
  return LL-MIN($B[1..k]$)

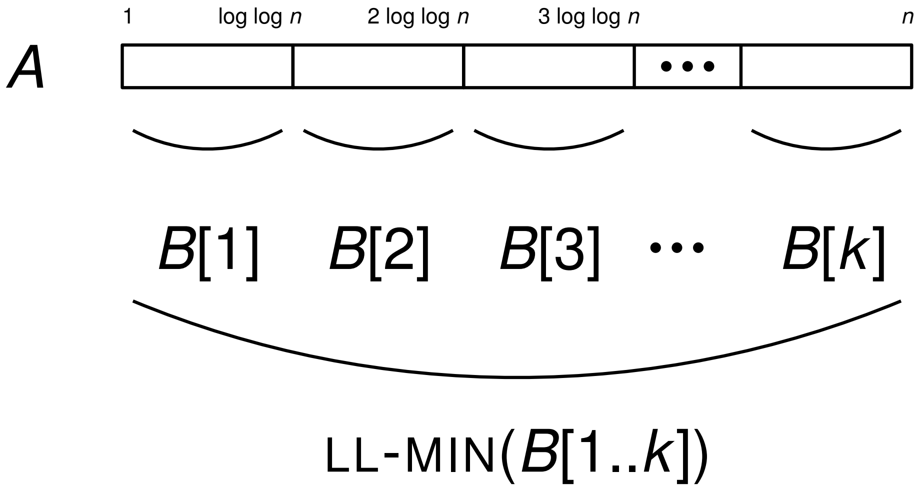


Analysis

$$T(n) = O(\log\log n) + O\left(\log\log \frac{n}{\log\log n}\right)$$
$$= O(\log\log n) + O\left(\log\log n - \log^{(4)} n\right) = O(\log\log n)$$
$$W(n) = \frac{n}{\log\log n} \cdot O(\log\log n) + O\left(k \log\log k\right)$$

Work-Efficient Common-CRCW Minimum

procedure CRCW-MIN($A[1..n]$)
  B = new array of size $k = \frac{n}{\log\log n}$
  for $i = 1$ to k **in parallel do**
    $\ell = 1 + k \cdot (i - 1)$
    $r = k \cdot i$
    $B[i]$ = SEQ-MIN($A[\ell..r]$)
  return LL-MIN($B[1..k]$)

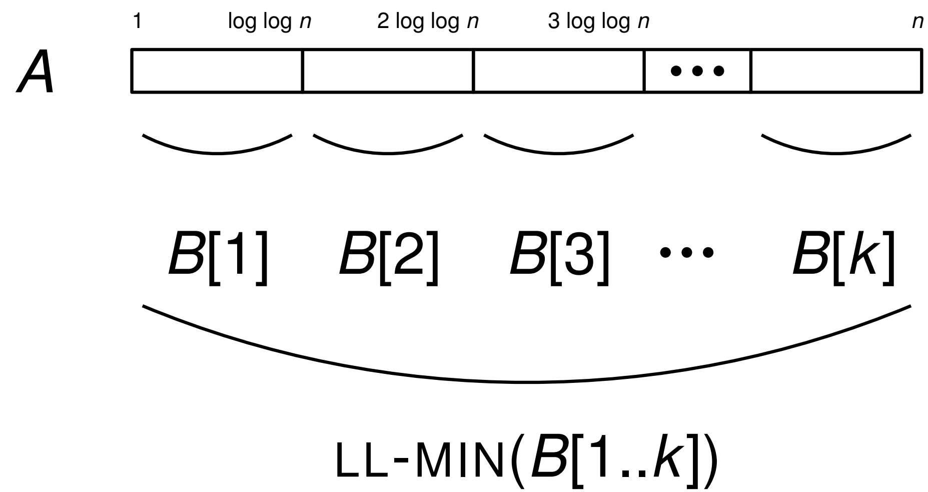


Analysis

$$T(n) = O(\log\log n) + O\left(\log\log \frac{n}{\log\log n}\right)$$
$$= O(\log\log n) + O\left(\log\log n - \log^{(4)} n\right) = O(\log\log n)$$
$$W(n) = \frac{n}{\log\log n} \cdot O(\log\log n) + O\left(k \log\log k\right)$$
$$= O\left(n + \frac{n}{\log\log n} \cdot \log\log \frac{n}{\log\log n}\right)$$

Work-Efficient Common-CRCW Minimum

procedure CRCW-MIN($A[1..n]$)
  B = new array of size $k = \frac{n}{\log\log n}$
  for $i = 1$ to k **in parallel do**
    $\ell = 1 + k \cdot (i - 1)$
    $r = k \cdot i$
    $B[i]$ = SEQ-MIN($A[\ell..r]$)
  return LL-MIN($B[1..k]$)

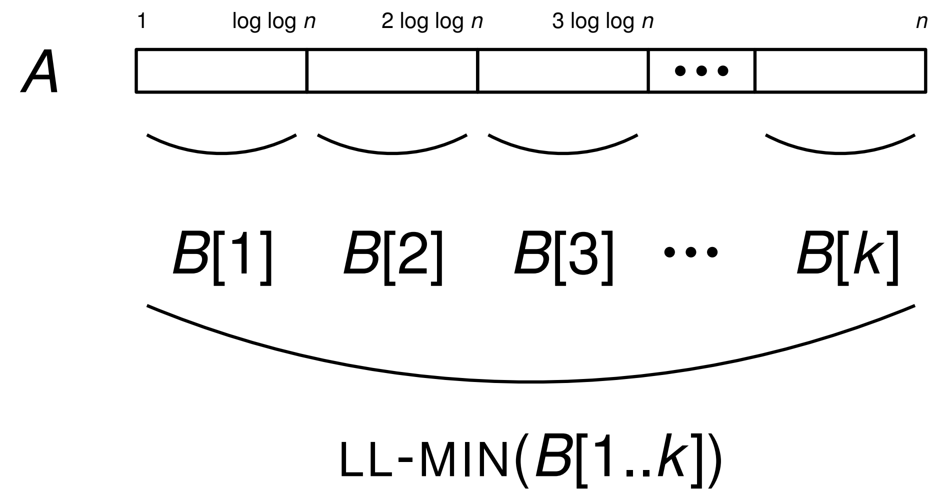


Analysis

$$T(n) = O(\log\log n) + O\left(\log\log \frac{n}{\log\log n}\right)$$
$$= O(\log\log n) + O\left(\log\log n - \log^{(4)} n\right) = O(\log\log n)$$
$$W(n) = \frac{n}{\log\log n} \cdot O(\log\log n) + O(k \log\log k)$$
$$= O\left(n + \frac{n}{\log\log n} \cdot \log\log \frac{n}{\log\log n}\right) = O\left(n + n - \frac{n}{\log\log n} \cdot \log^{(4)} n\right)$$

Work-Efficient Common-CRCW Minimum

procedure CRCW-MIN($A[1..n]$)
  B = new array of size $k = \frac{n}{\log\log n}$
  for $i = 1$ to k **in parallel do**
    $\ell = 1 + k \cdot (i - 1)$
    $r = k \cdot i$
    $B[i]$ = SEQ-MIN($A[\ell..r]$)
  return LL-MIN($B[1..k]$)

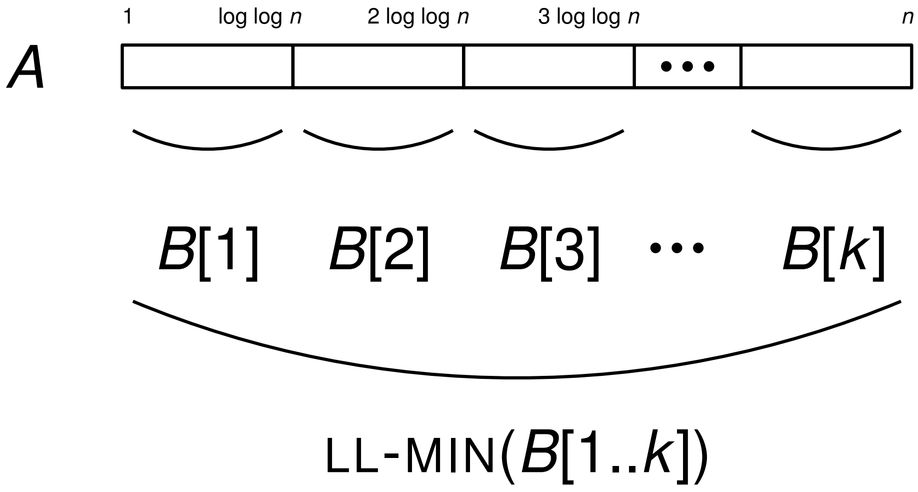


Analysis

$$T(n) = O(\log\log n) + O\left(\log\log \frac{n}{\log\log n}\right)$$
$$= O(\log\log n) + O\left(\log\log n - \log^{(4)} n\right) = O(\log\log n)$$
$$W(n) = \frac{n}{\log\log n} \cdot O(\log\log n) + O(k \log\log k)$$
$$= O\left(n + \frac{n}{\log\log n} \cdot \log\log \frac{n}{\log\log n}\right) = O\left(n + n - \frac{n}{\log\log n} \cdot \log^{(4)} n\right) = O(n)$$

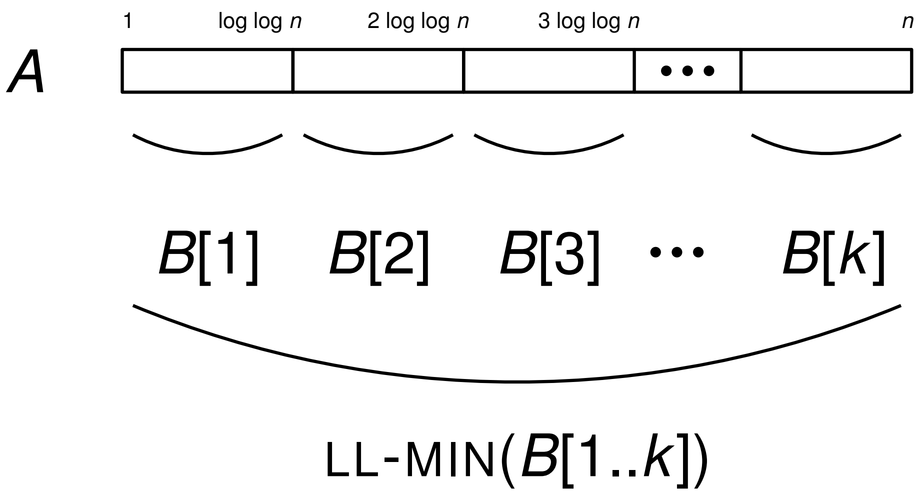
Attaining Work-Efficiency

procedure CRCW-MIN($A[1..n]$)
    B = new array of size $k = \frac{n}{\log\log n}$
    for $i = 1$ to k **in parallel do**
        $\ell = 1 + k \cdot (i - 1)$
        $r = k \cdot i$
        $B[i]$ = SEQ-MIN($A[\ell..r]$)
    return LL-MIN($B[1..k]$)



Attaining Work-Efficiency

procedure CRCW-MIN($A[1..n]$)
  B = new array of size $k = \frac{n}{\log\log n}$
  for $i = 1$ to k **in parallel do**
    $\ell = 1 + k \cdot (i - 1)$
    $r = k \cdot i$
    $B[i]$ = SEQ-MIN($A[\ell..r]$)
  return LL-MIN($B[1..k]$)



- Reduce the size of the original problem
  - Solve many small problems using ***slow*** but ***work-efficient*** algorithm
- Solve the reduced problem using ***fast*** but ***work-inefficient*** algorithm

Summary

	Time	Work
EREW PRAM:	$\Theta(\log n)$	$\Theta(n)$
Common-CRCW PRAM:	$\Theta(1)$	$\Theta(n^2)$
	$\Theta(\log \log n)$	$\Theta(n)$