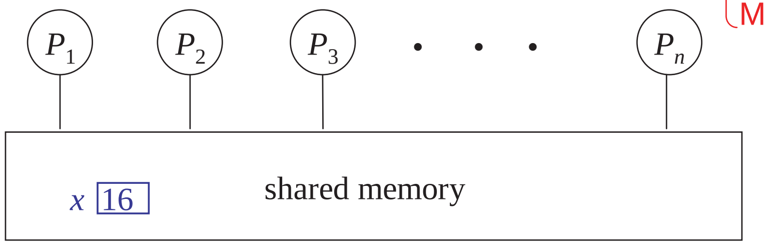
2

Shared-Memory Computers, Basic Techniques, and Brent's Theorem

■ Shared-Memory Computers

SISD
SIMD
MISD
MIMD



1. *EREW* (exclusive read, exclusive write): Simultaneous access to the same memory location is not allowed.

2. *CREW* (concurrent read, exclusive write): Simultaneous read to the same memory location is allowed, but simultaneous write is not allowed.

3. *ERCW* (exclusive read, concurrent write): Simultaneous read is not allowed, but simultaneous write is allowed.

P_1: $x = 2$
P_2: $x = 4$

4. *CRCW* (concurrent read, concurrent write): Both simultaneous read and simultaneous write are allowed.

* Shared-memory computer is of great theoretical interest, but current technology prohibits its realization.

* The shared-memory computer can be regarded as a completely connected interconnection network.

* Communication between any two processors takes $O(1)$ time through the shared memory.

* The SIMD shared-memory computer is also called *Parallel Random Access Machine* (*PRAM*)

not follow ⇒ unexpected

CW

* **Resolution rules for write conflicts** (L. Kucera, "Parallel computation and conflicts in memory access," *Information Processing Letters*, vol. 14, no. 2, pp. 93-96, 1982.)

  1. Weak conflict-resolution rule (also known as W-PRAM):
     (i) Simultaneous writing is allowed only to selected memory locations which can contain the numbers 0 or 1 only.
     (ii) Processors write simultaneously into the same location must write the value 1 and the final value remaining is 1.

  * O(1) for OR/AND of n bits SM-2a

  2. Equality conflict-resolution rule.

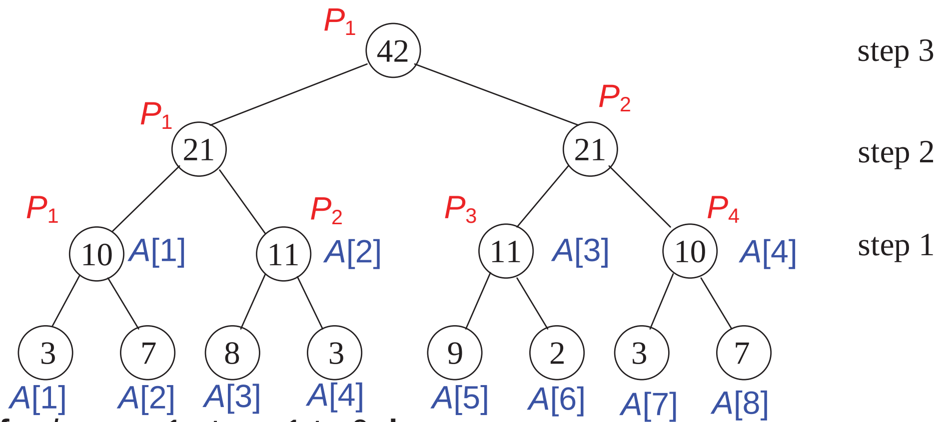
  3. Priority conflict-resolution rule (fixed or dynamically changeable).

SM-

* How to find the maximum in $O(1)$ time???

■ Basic Techniques

Problem SM-1: Summing
Input: $A[1 \ldots n] = \{3, 7, 8, 3, 9, 2, 3, 7\}$ ($n = 8$)
Output: $A[1] + A[2] + \ldots + A[n]$
Model: EREW PRAM of $n/2$ processors ($n = 2^m$)



for $k = m - 1$ **step** -1 **to** 0 **do**
    for i, $1 \le i \le 2^k$, **pardo** $\underbrace{A[i] = A[2i-1] + A[2i]}_{P_i \text{ do (ER or CR ???)}}$

* $T(n) = O(\log n)$
* *Balanced binary tree method*: Build a balanced binary tree on the input (or output) elements and traverse the tree forward and backward to and from the root.

Simple Applications: (1) Computing maximum
(2) Broadcasting
(3) OR, AND, XOR $X=(1,3,2,3)$

* Prefix sums (two phases) (tree machine) $S=(1,4,6,9)$

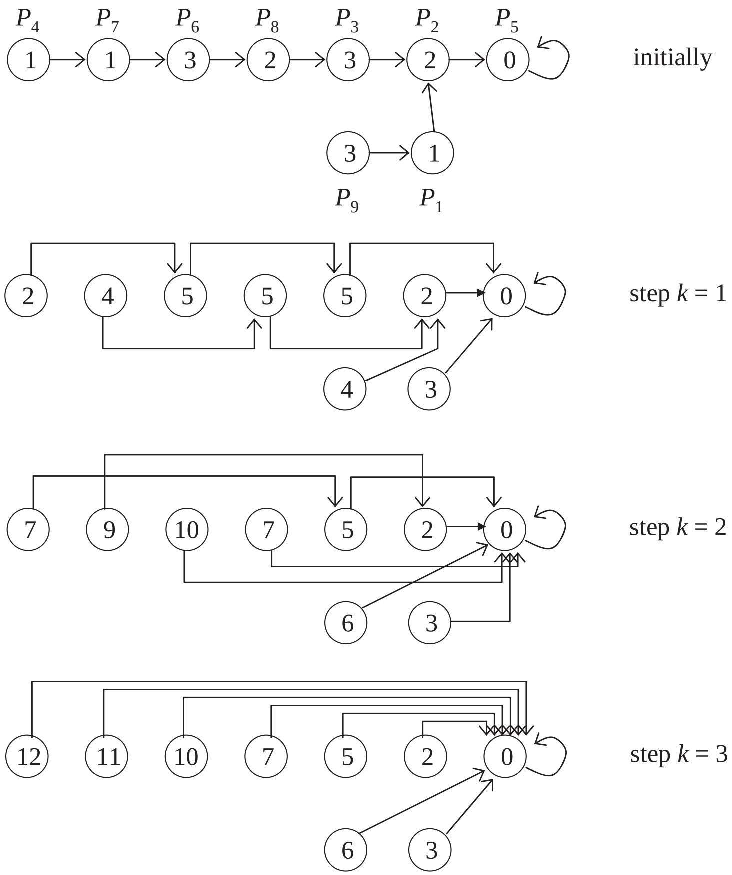
Problem SM-2: Parallel prefix on a rooted directed tree
Input: $P[1 \ldots n] = \{2, 5, 2, 7, 5, 8, 6, 3, 1\}$ and
$W[1 \ldots n] = \{1, 2, 3, 1, 0, 3, 1, 2, 3\}$.
(A rooted directed tree of n nodes; $P[i]$ and $W[i]$, $1 \le i \le n$, is the parent and weight of node i, respectively. The tree is self-loop at its root and the weight of the root is 0.)

Output: For each node i, $W[i]$ is set equal to the sum of the weights of nodes on the path from i to the root of its tree.
Model: CREW PRAM of n processors

for i, $1 \le i \le n$, **pardo** $S[i] = P[i]$
for $k = 1$ **to** $\log n$ **do**
    for i, $1 \le i \le n$, **pardo**
    begin
        $W[i] = W[i] + W[S[i]]$
        $S[i] = S[S[i]]$
    end

* Correctness??? Why CR??? ER for a list???

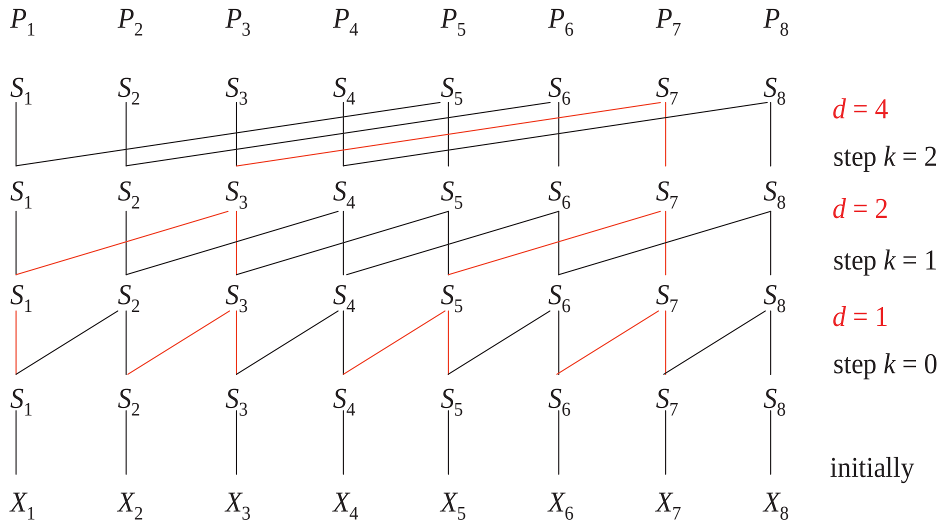
* $T(n) = O(\log n)$ Why not O(log l) ???
* *Pointer Jumping (Doubling)*: The computation proceeds by a recursive application of the calculation in hand to all elements over a certain distance (in the data structure) from each individual element. This distance doubles in successive steps.



Simple Applications:
(1) Parallel prefix (computing prefix sums on linked list)
(2) List ranking (EREW, $n/\log n$ processors, $O(\log n)$ time)
(3) Parallel prefix on a rooted directed forest
(4) Finding the root for each node of a given forest
(5) Determining the length of the path from each node (of a given forest) to its root.

$X=(1,3,2,3)$
$S=(1,4,6,9)$

Problem SM-3: Prefix sums (prefix computation)
Input: $X[1\ldots n]$
Output: $S[1\ldots n]$, where $S[i] = X[1] + X[2] + \ldots + X[i]$
Model: EREW PRAM of n processors



for i, $1 \le i \le n$, **pardo** $S[i] = X[i]$
for $k = 0$ **to** $\log n - 1$ **do**
  for i, $2^k+1 \le i \le n$, **pardo** $S[i] = S[i] + S[i-2^k]$

SM-6a

* $T(n) = O(\log n)$
* *Doubling*

Problem SM-4: Parallel *m*-way search
Input: a sorted sequence $A[1, n] = \{1, 2, \ldots, 18\}$, and a value $x = 11$
Output: k such that $A[k] = x$ (Assume that k uniquely exists.)
Model: CREW PRAM of p processors

step 0: *beginning* = 0; *length* = n;

step 1: If (*length* $\le p$)
each processor P_i, $1 \le i \le$ *length*, sets k = *beginning* + i if A[*beginning* + i] = x; and then, terminates.

step 2: Each processor P_i, $1 \le i \le n$, sets $M[i]$ as 1 if $x \le A$[*beginning* + $i \times$(*length*/p)] and 0 otherwise.

step 3: Each processor P_i, $1 \le i \le p$, sets $M[i] = M[i] - M[i-1]$. (Assume $M[0]=0$.) And then, if $M[i] = 1$, sets *beginning* = *beginning* + $(i-1)\times$(*length*/p) and *length* = *length*/p.

step 4: Repeat 1~3 until k is found.

Example: $x = 11$, $n = 18$, $p = 3$

B $\boxed{1}$ *beginning* = 0
E $\boxed{18}$ *length* = 18

						P_1						P_2						P_3
A:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
M:						0						1						1

stage 1

B $\boxed{7}$ *beginning* = 6
E $\boxed{12}$ *length* = 6

								P_1		P_2		P_3						
A:	*	*	*	*	*	*	7	8	9	10	11	12	*	*	*	*	*	*
M:								0		0		1						

stage 2

B $\boxed{11}$ *beginning* = 10
E $\boxed{12}$ *length* = 2

											P_1	P_2						
A:	*	*	*	*	*	*	*	*	*	*	<u>11</u>	12	*	*	*	*	*	*

$k = 11$

stage 3

* A simple extension of binary search

* $T(n) = T(n/p) + O(1) = O(\log_p n)$

* While $p = n^{1/k}$ for some constant k, the proposed algorithm requires $O(1)$ time. We will have an example of $k = 2$ in **Problem** SM-5 (convex hull).