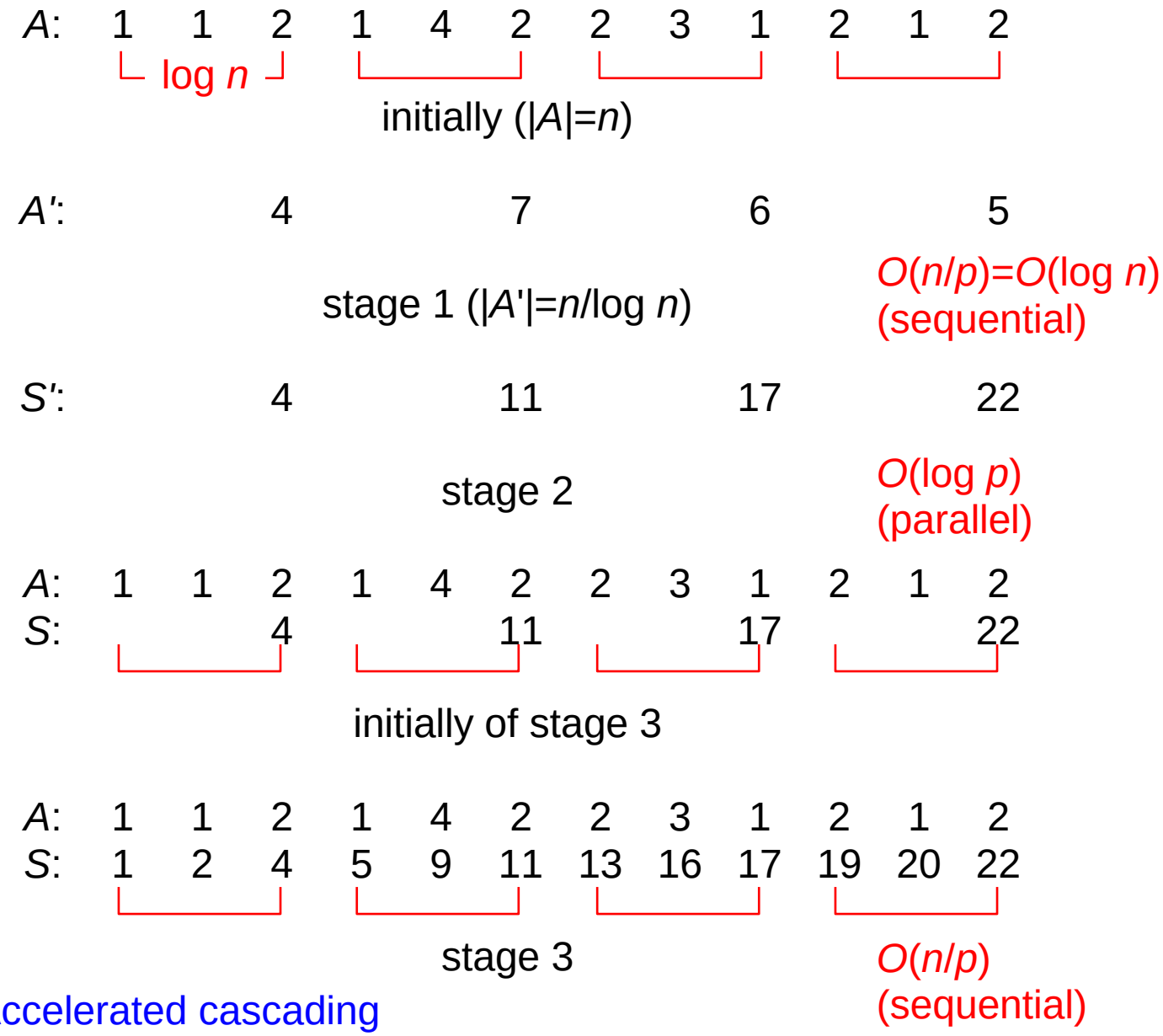
`A cost optimal prefix sums algorithm

SM-3: n PEs, $O(\log n)$ time => cost: $n \log n$, M: $n \log n$
sequential: 1 PEs, $O(n)$ time => cost: n

$n = 12$ $p = n/\log n = 4$



* Accelerated cascading

A_1 fast, non-optimal
\+
A_2 slow, optimal
= fast, optimal

■ Problems

Problem SM-11: Selection
Input: $S[1\ldots n]$, k
Output: the k-th smallest element in S
Model: EREW PRAM of $N = n^{1-\varepsilon}$, $0 < \varepsilon < 1$, processors

(1) **Sequential approach** (*prune-and-search*)

$r = 5$

1. Divide S into n/r subsequences of r integers (r>1, odd, constant).

2. Sort every group. Let m_i be the median of the i-th subsequence. $r^2 \times \frac{n}{r} = nr \Rightarrow O(n)$

3. Find the median m of m_i's. $T(n/r)$

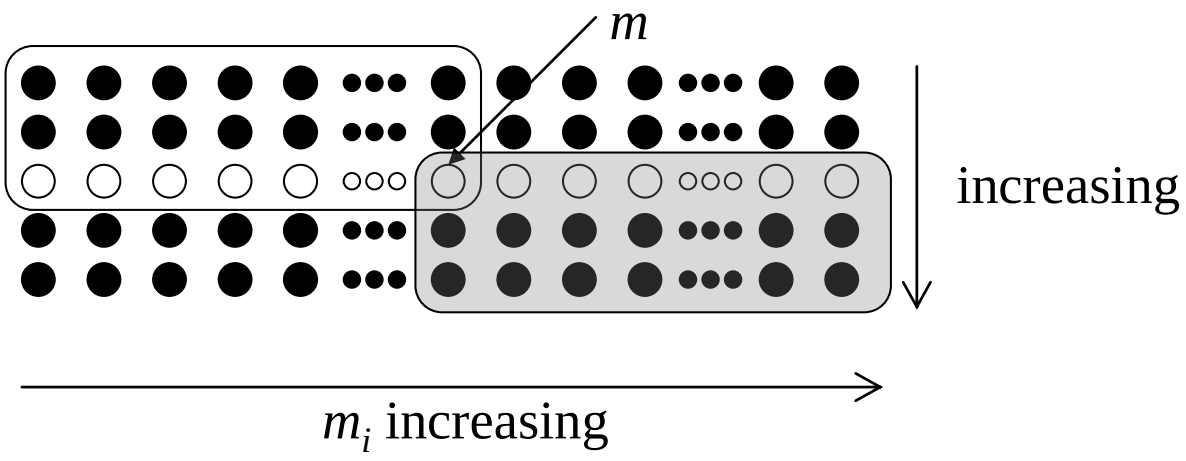
4. Partition S into three subsequences:

   $S_1=\{x|\ x < m\}$, $S_2=\{x|\ x = m\}$, and $S_3=\{x|\ x > m\}$.

   Then, the k-th smallest element of S is located in one of the three subsequences. $O(n)$

5. Repeat (1)~(4) for the chosen subsequence until the k-th smallest element is found. $O(?)$

***** Analysis (Assume $r = 5$.)



At least one fourth of S is discard. ($|S_1| \le 3|S|/4$ and $|S_3| \le 3|S|/4$)

step	1.	2.	3.	4.	5.
Time	O(1)	$O(n)$	$T(n/r)$	$O(n)$	$T(3n/4)$

$T(n) = T(n/r) + T(3n/4) + O(n)$

By induction, $T(n) = O(n)$ is derived.

(2) **Parallel approach** $n^{1-\varepsilon}$ PEs

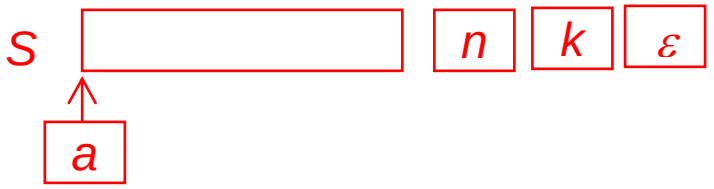
1. Divide S into $n^{1-\varepsilon}$ subsequences of length n^{ε}. Each subsequence is assigned to one processor.

2. Find the median m_i of the i-th subsequence by sequential approach.

3. Find the median m of m_i's by parallel approach.

4. Partition S into S_1, S_2, and S_3.

5. Repeat (1)~(4) for the chosen subsequence until the k-th smallest element is found.



*Analysis.

(Initially, S, the beginning address a of S, and the values n, k, ε are stored in the shared memory)

step 1. $O(\log n)$ time

$O(\log p)$ (i) The values of a, n, k, ε are broadcast to all processors. This can be done in $O(\log n^{1-\varepsilon}) = O(\log n)$ time.

(ii) Each processor P_i computes the addresses $a+(i-1)n^{\varepsilon}$ and $a+i*n^{\varepsilon}-1$ of the first and the last elements of its associated subsequence. This requires $O(1)$ time.

$O(n/p)$ step 2. $O(n^{\varepsilon})$ time find m_i

step 3. $T(n^{1-\varepsilon})$ time find m

step 4. $O(\log n)+O(n^{\varepsilon})=O(n^{\varepsilon})$

$O(\log p)$ (i) m is broadcast to all processors. ($O(\log n^{1-\varepsilon})$ time)

$O(n/p)$ (ii) processor P_i splits its associated subsequence into $S_{i,1}$, $S_{i,2}$, and $S_{i,3}$. ($O(n^{\varepsilon})$ time)

find S_1

(iii) Let $l_i = |S_{i,1}|$. Processor P_i computes $z_i = l_1 + l_2 + \ldots + l_i$. ($O(\log n^{1-\varepsilon})$ time) $O(\log p)$

(iv) The beginning address a_1 of S_1 is broadcast to all processors. ($O(\log n^{1-\varepsilon})$ time) $O(\log p)$

(v) Processors P_i write $S_{i,1}$ to S_1, starting at a_1+z_{i-1} ($z_0=0$). ($O(n^{\varepsilon})$ time) $O(n/p)$

(vi) S_2 and S_3 can be obtained similarly.

step 5. $T(3n/4)$

(Note that $|S_1|$ ($z_{n^{1-\varepsilon}}$), $|S_2|$, and $|S_3|$ have already been computed in step 4. Thus, what sequence contains the k-th smallest element can be determined in $O(1)$ time.)

(In fact, we only need to construct one of S_1, S_2, and S_3.)

$T(n) = O(\log n)+O(n^{\varepsilon})+T(n^{1-\varepsilon})+T(3n/4) = O(n^{\varepsilon})$

cost optimal!

Reference: R. Cole and C. K. Yap, "A parallel median algorithm," *Information Processing Letters*, vol. 20, pp. 137-139, 1985. ($O((\text{loglog } n)^2)$ time, CREW.)

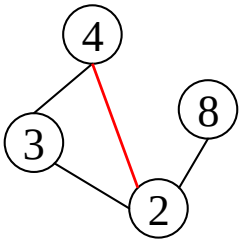
Reference: R. Cole, "An optimally efficient selection algorithm," *Information Processing Letters*, vol. 26, no. 6, pp. 295-299, 1988. ($O(\log n \log^* n)$-time, EREW, cost optimal.)

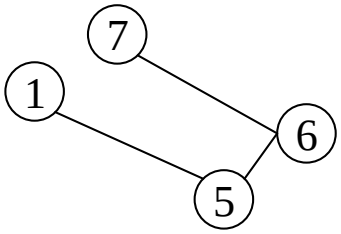
Problem SM-12: Connected component

Input: An adjacent matrix $A[1\ldots n, 1\ldots n]$ of an n-node graph, where $A[i, j]=1$ if and only if node i and node j are adjacent.

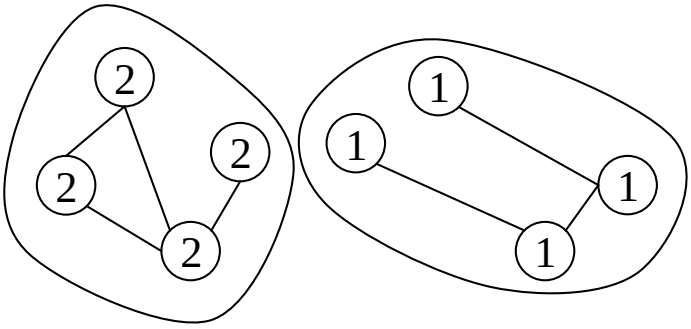
Output: For each connected component, label all nodes as the index of the smallest node in it.

Model: CREW PRAM of n^2 processors



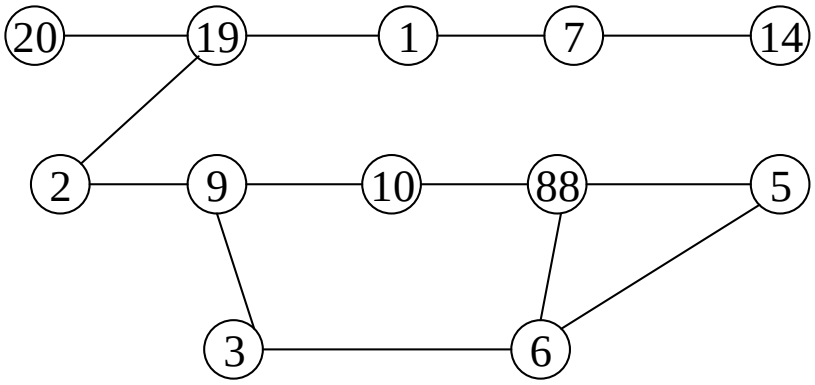


an undirected graph G

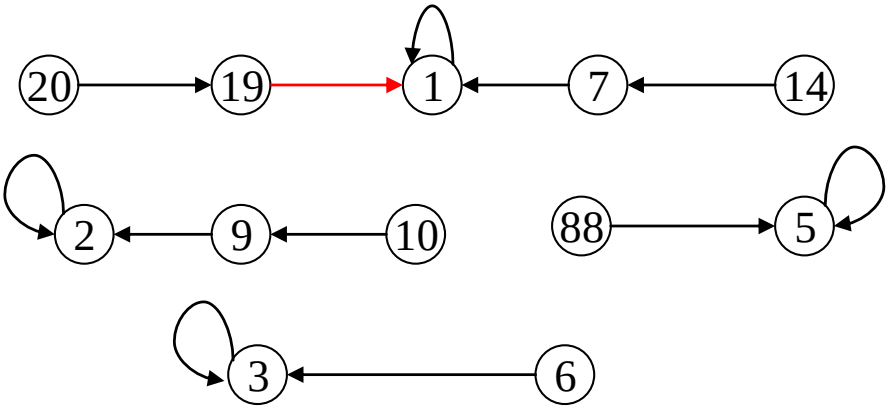


two connected components of G

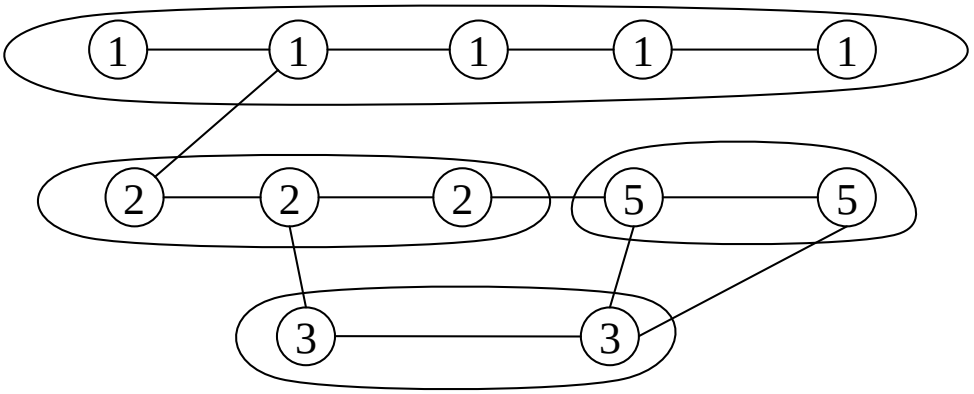
$$A = \begin{array}{c} \begin{matrix} 2 & 8 \end{matrix} \\ 4 \begin{bmatrix} 1 & 0 \end{bmatrix} \\ \\ 8\times 8 \end{array}$$



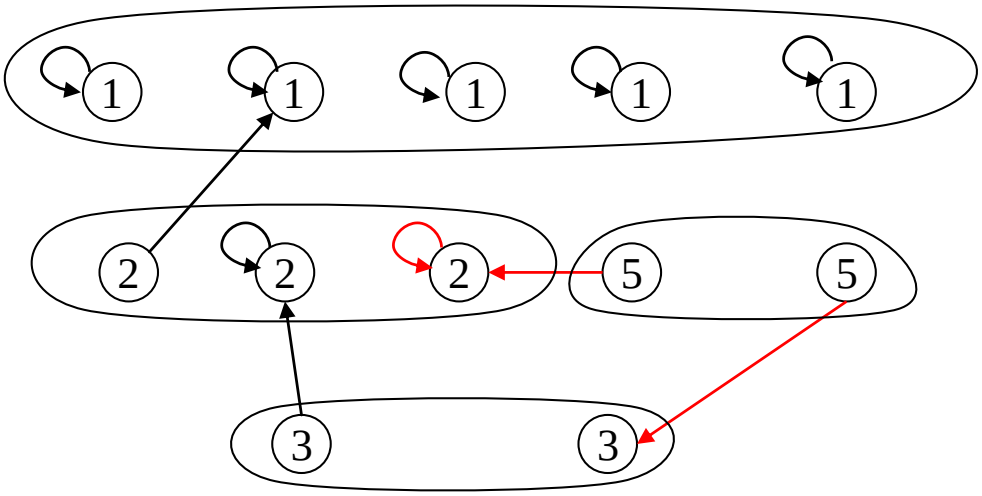
initially (only one component is shown.)



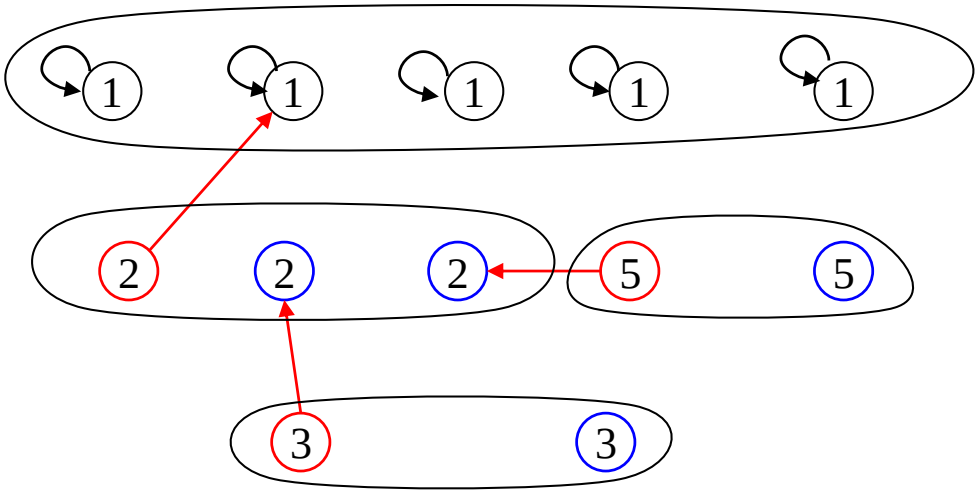
step 1 of stage 1



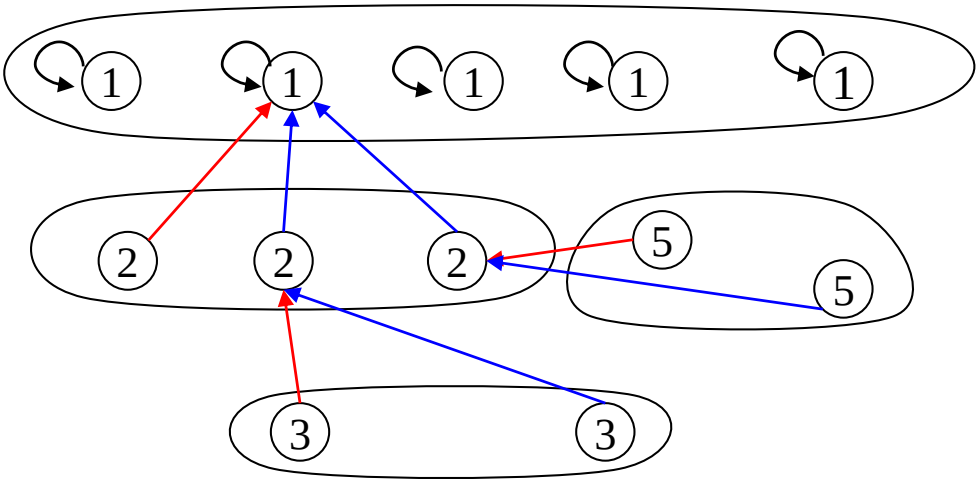
step 2 of stage 1



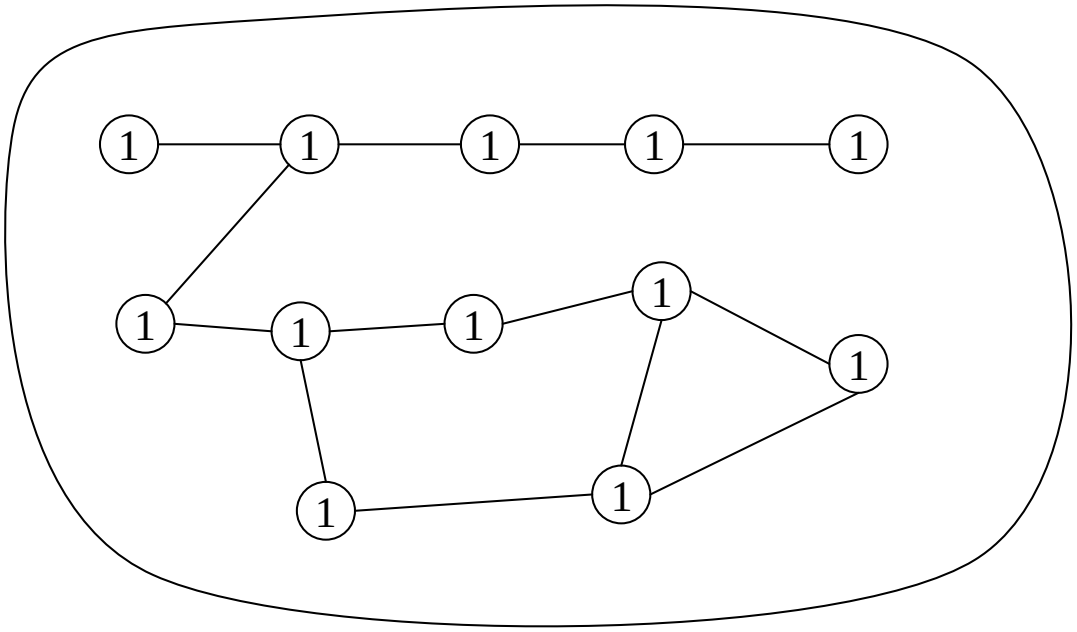
step 1-1 of stage 2



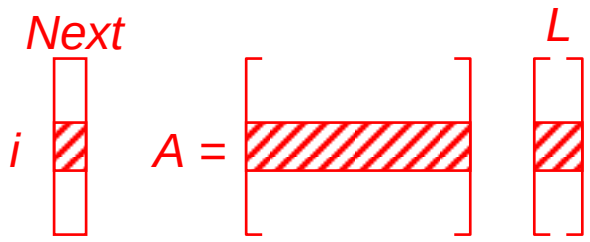
step 1-2 of stage 2



step 1-3 of stage 2



step 2 of stage 2



(Assign n PEs to each node. Initially, node i is labeled $L(i)=i$.)

step 1. (Each sub-component determines the smallest adjacent sub-component. It requires $O(\log n)$ time.)
For each node i, determine the node x_i, where $L(x_i)=\min\{L(j)|\ A[i, j]=1\}$. Then, determine the node y_i, where $L(y_i)=\min\{L(x_j)|\ L(j)=L(i)\}$. And then, set $Next(i)=i$ if $L(y_i)=L(i)$. Otherwise, set $Next(i)= y_i$.

step 2. (Combine sub-components together. $O(\log n)$ time.)
for i, $1 \le i \le \log n$, **do**
begin
$L(i)=L(Next(i))$ pointer jumping
$Next(i)=Next(Next(i))$
end
$O(\log n)$ iterations

step 3. Repeat (1) and (2) until no more combination of sub-components. (Since the number of components is reduced by a factor of at least 2 per iteration, at most log n times are required.)
why ???

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

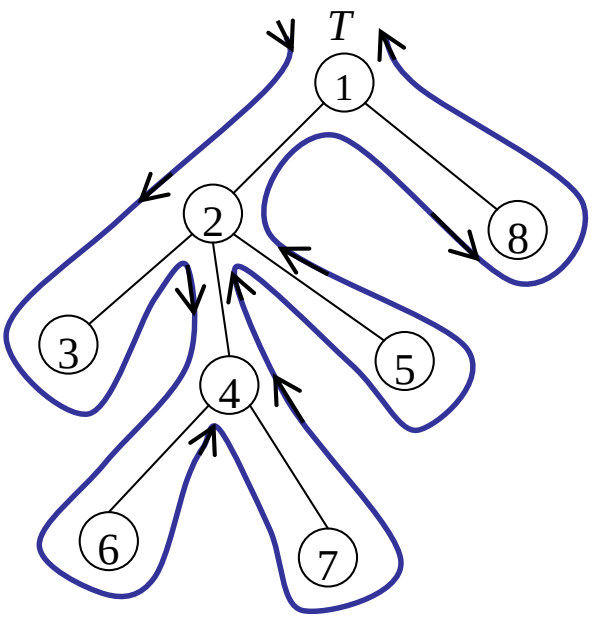
Reference: C. Savage and J. JáJá, "Fast, efficient parallel algorithms for some graph problems," *SIAM J. Comput.*, vol. 10, no. 4, 1981, pp. 681-691.

■ The Euler-tour technique

(J. JáJá, *An Introduction to Parallel Algorithms*, page 121)

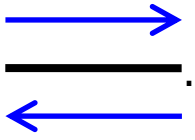
Euler-tour:

T: rooted tree



1 2 3 2 4 6 4 7 4 2 5 2 1 8 1

Theorem: Given a rooted tree *T* (adjacency list), an Euler-tour of it can be constructed in $O(1)$ time using n processors on the EREW PRAM.

(1. replace each edge by two:  .)

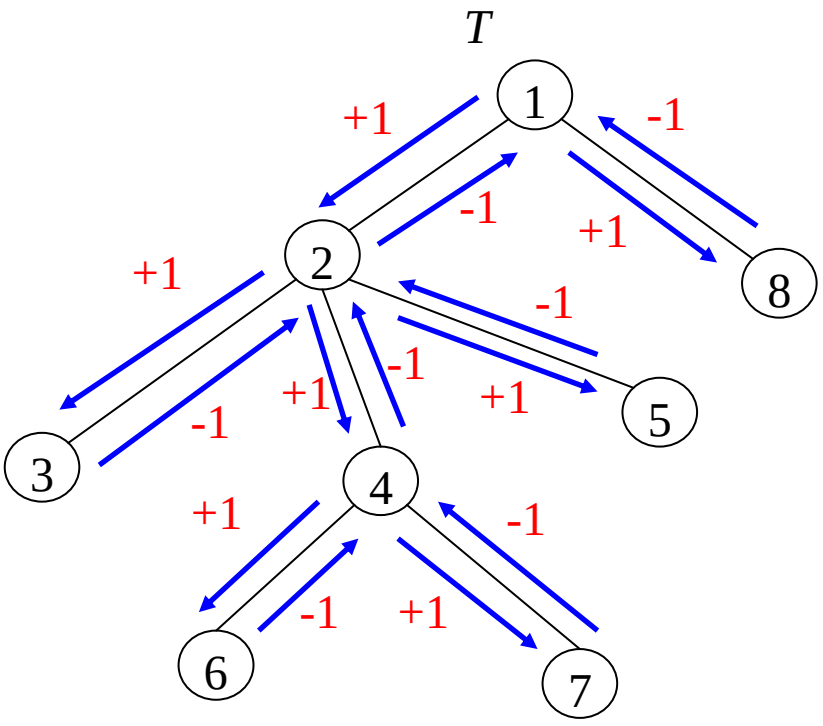
(2. determine the sequence (order).)

The Euler-tour technique:

1. Construct an Euler-tour
2. Assign weights
3. Prefix computation
   (Pointer Jumping)
4. Find the answer

Example: Computing the level of each node.

Step 2:



$$\Rightarrow \begin{cases} O(\log n) \text{ time} \\ O(n) \text{ cost} \end{cases}$$

Problem SM-13: Tree Contraction

Input: $P[1\ldots n]$ and $SIB[1\ldots n]$: An n-node rooted binary tree T such that each internal node has exactly two children, $p(u)$ and $sib(u)$ are the parent and sibling of u.

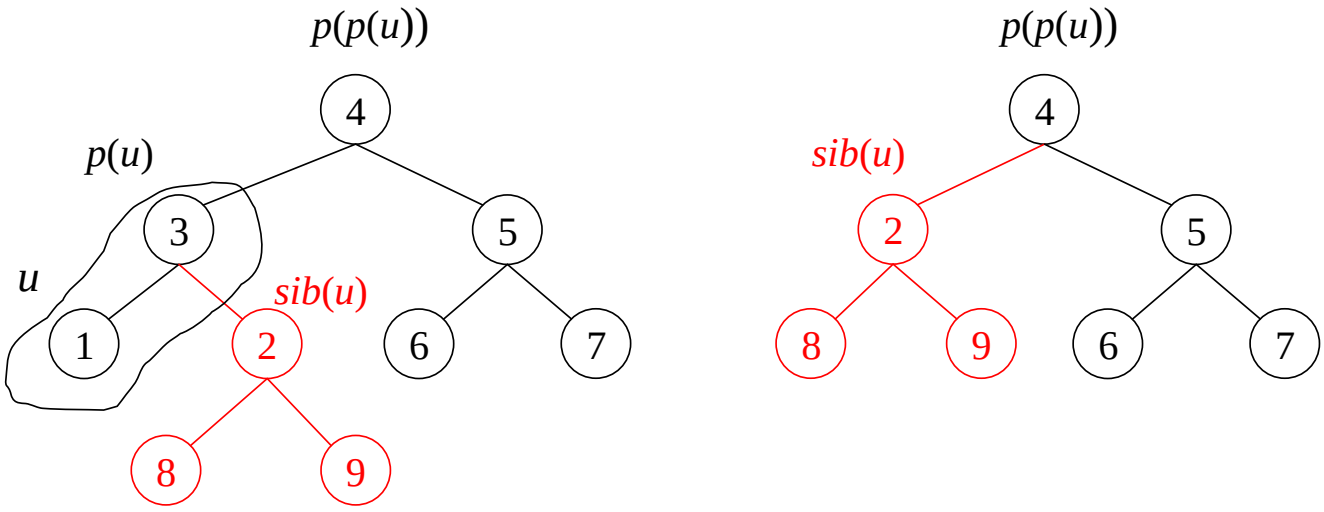
Output: T is contracted to a 3-node binary tree, by applying a sequence of ***rake*** operations.

Model: EREW PRAM of n processors

The rake operation (applied at a leaf u with $p(u) \neq r$):

(i) removing u and $p(u)$ from T, and

(ii) connecting $sib(u)$ to $p(p(u))$.



Apply the rake operation to node 1.

The algorithm

$O(\log n)$ time
The Euler-tour technique

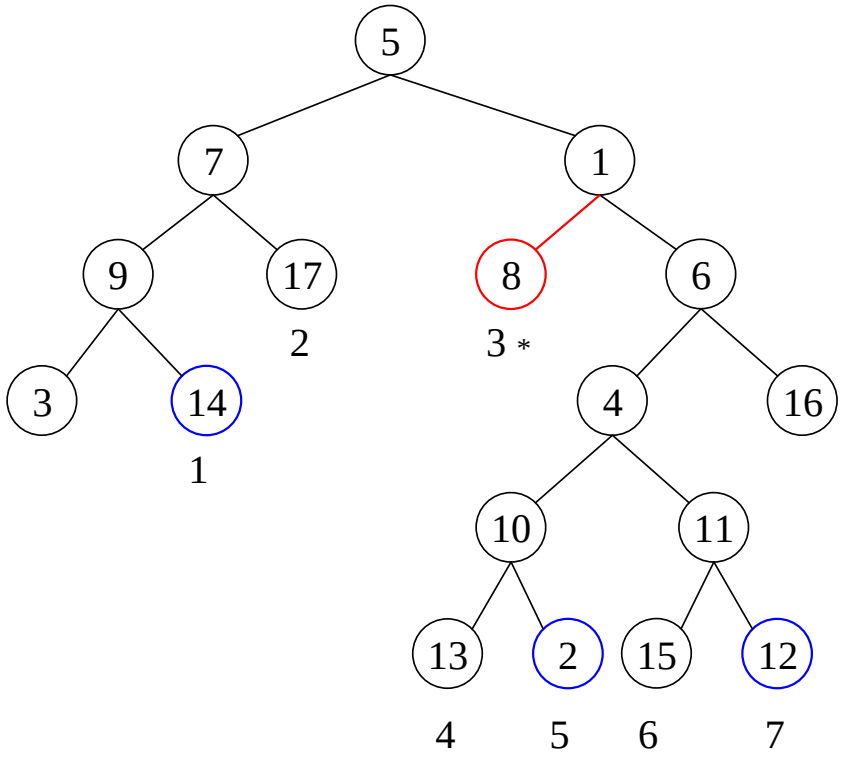
1. Label the leaves consecutively in order from left to right, excluding the leftmost and the rightmost leaves, and store the labeled leaves in an array A (of size at most n).

2. **for** log (n+1) iterations **do**

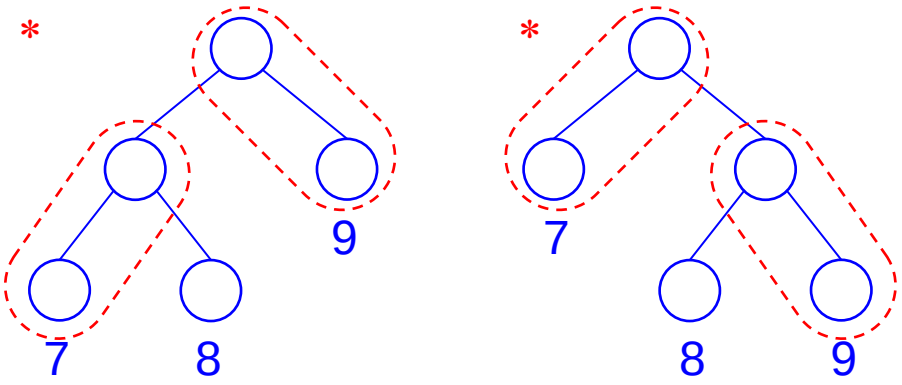
2.1. Apply the rake operation concurrently to all the elements of A_{odd} that are left children. (A_{odd} = (a_1, a_3, a_5, …))

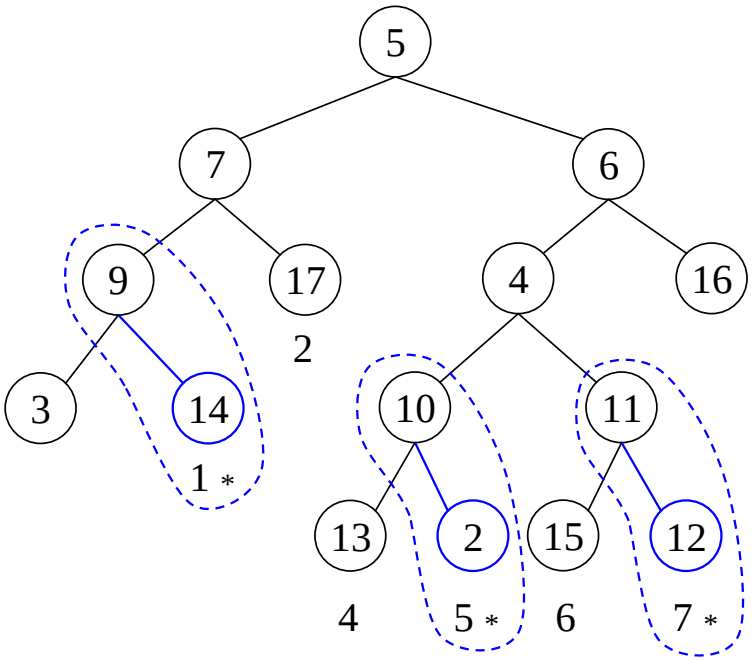
2.2. Apply the rake operation concurrently to the rest of the elements in A_{odd}.

2.3. Set $A = A_{even}$. (A_{even} = (a_2, a_4, a_6, …))

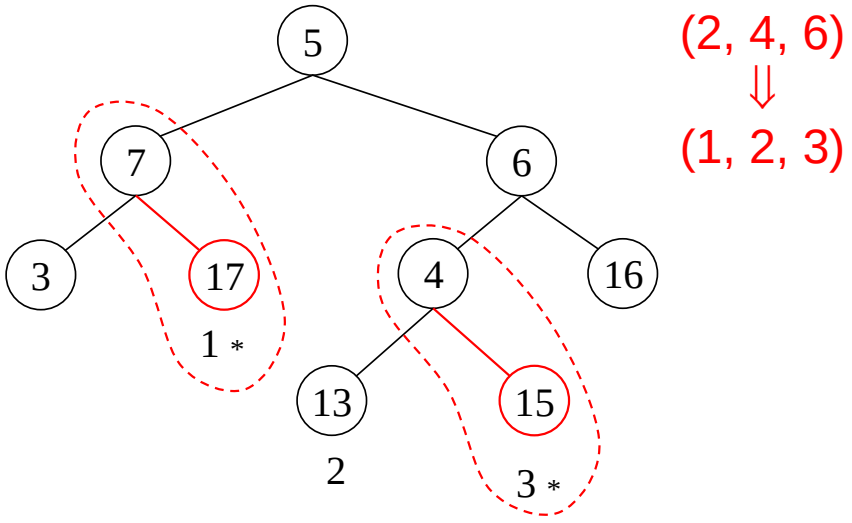


after step 1

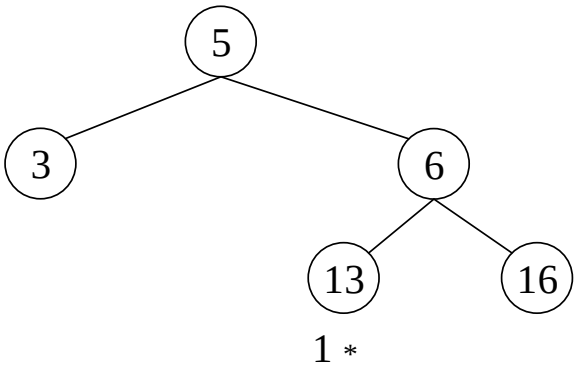




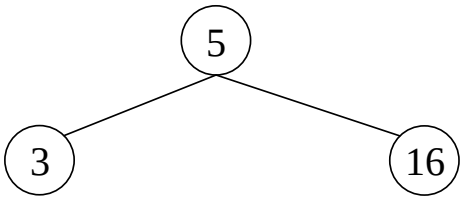
after step 2.1 of iteration 1



after step 2.2 and 2.3 of iteration 1



after step 2.1 (step 2.2, 2.3) of iteration 2



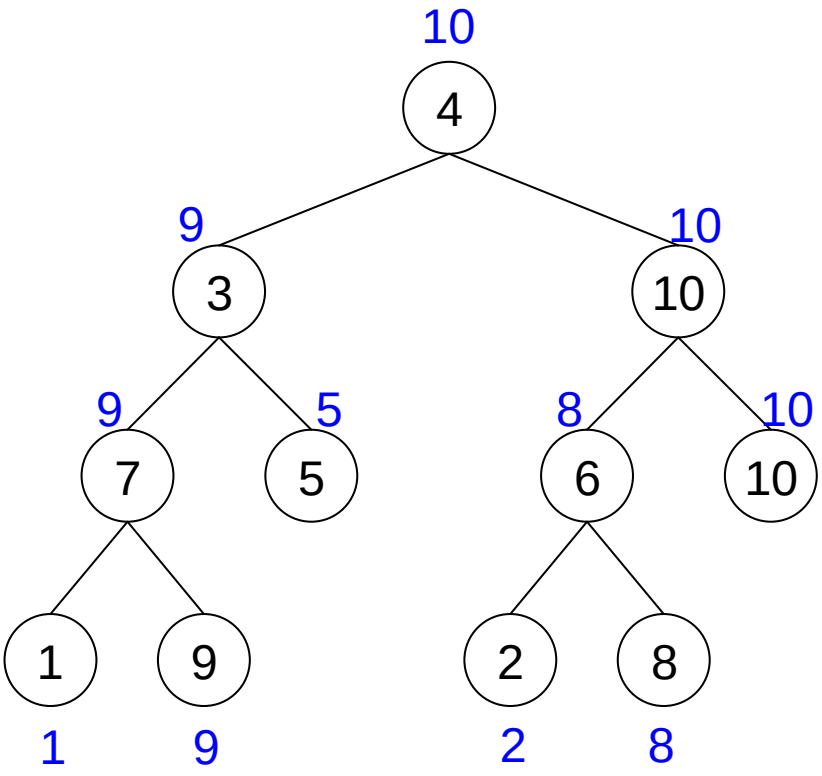
after step 2.1 (step 2.2, 2.3) of iteration 3

Remark: The step 1 of the above algorithm can be done in $O(\log n)$ time on an EREW PRAM of n processors by using the Euler-tour technique. ($O(n)$ total computation)

***$T(n)=O(\log n)$**, cost optimal by using Brent's Theorem.

$p=n/\log n$, $O(\log n)$ time.

Problem SM-14: Subtree maximum
Input: a node-weighted tree T
Output: find for each node v the maximum weight in T_v
Model: EREW PRAM of n processors



* Sequential: $O(n)$ (depth-first traversal).

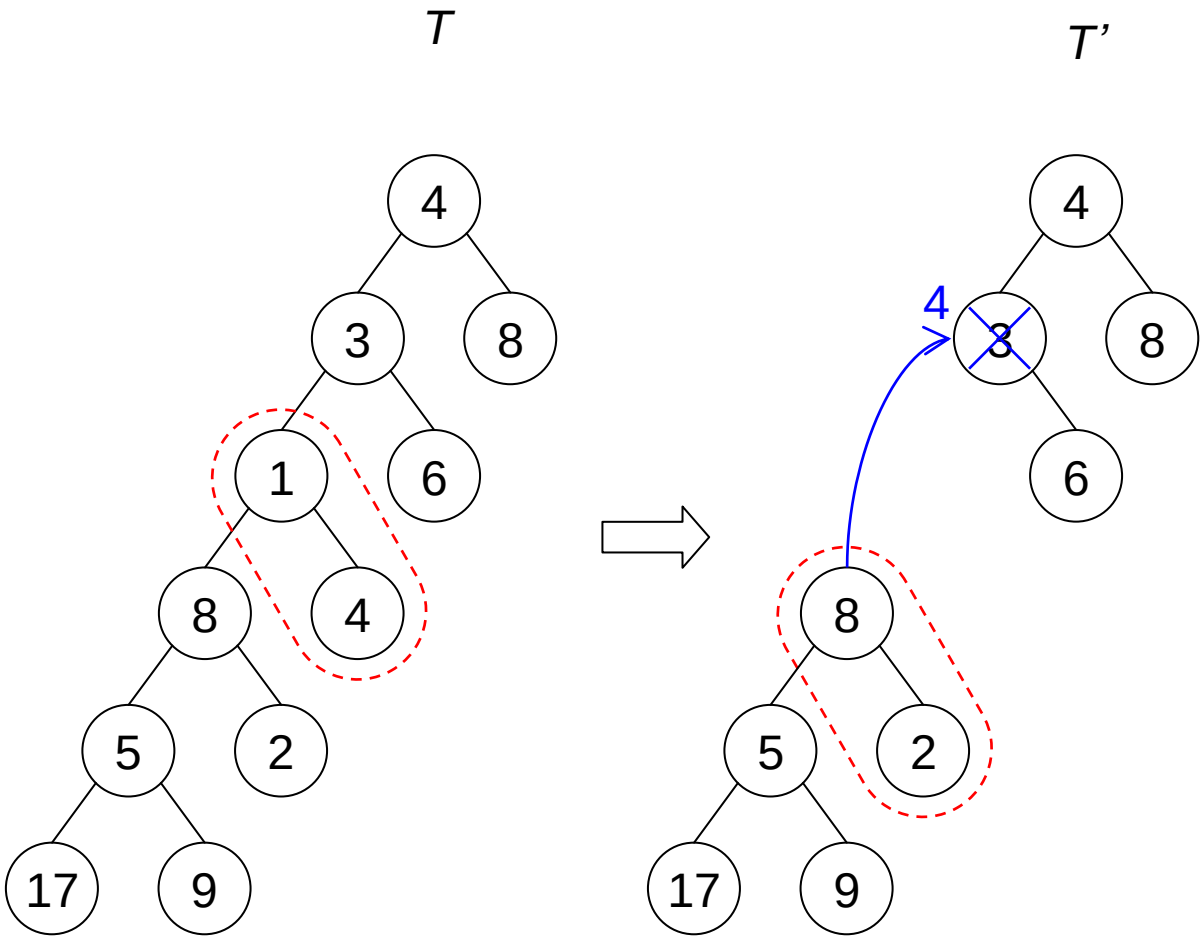
* Parallel: $O(h)$ (bottom-up)

$O(n)$ ✗

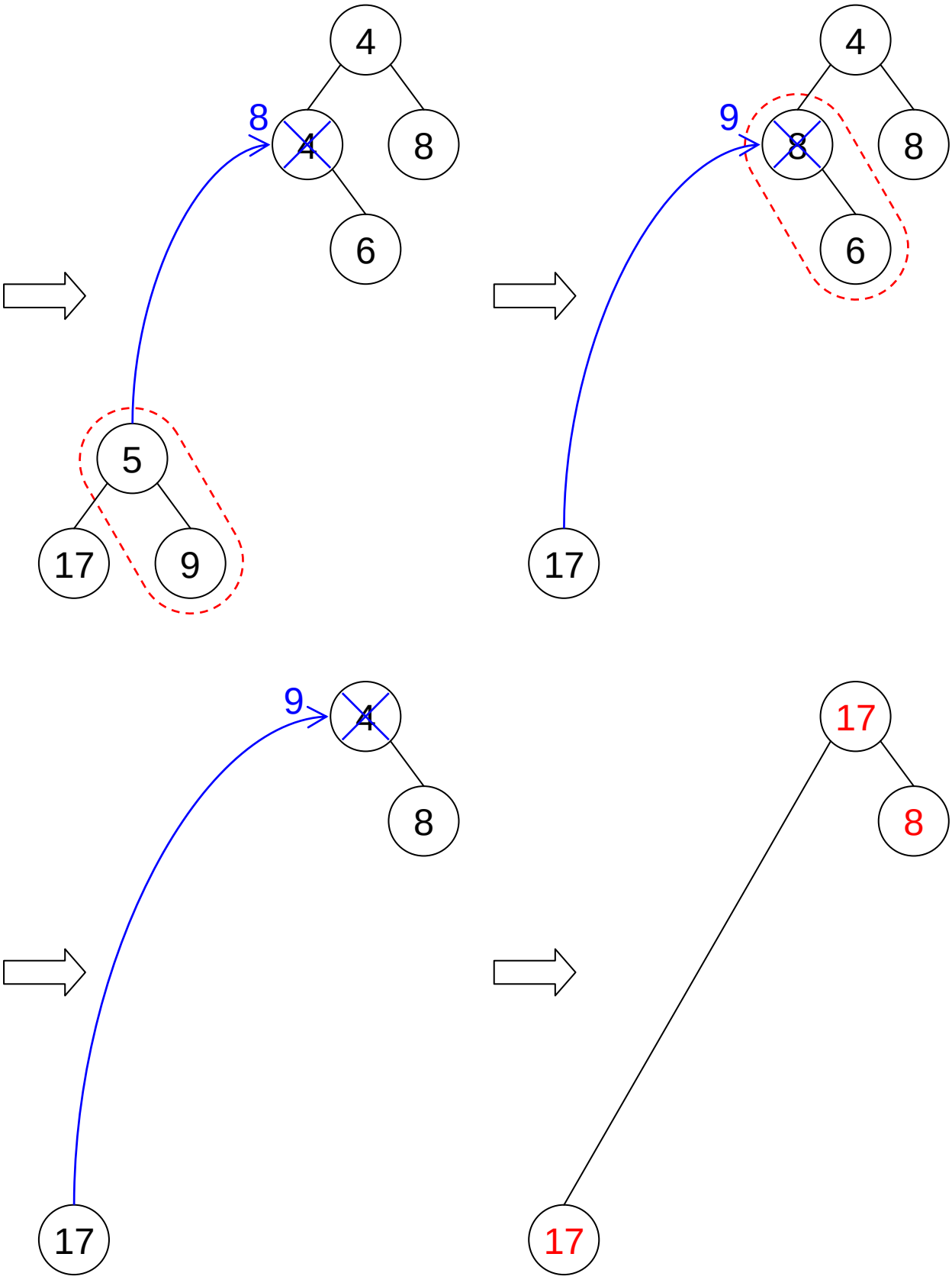
Applying tree contraction

idea: partial evaluation

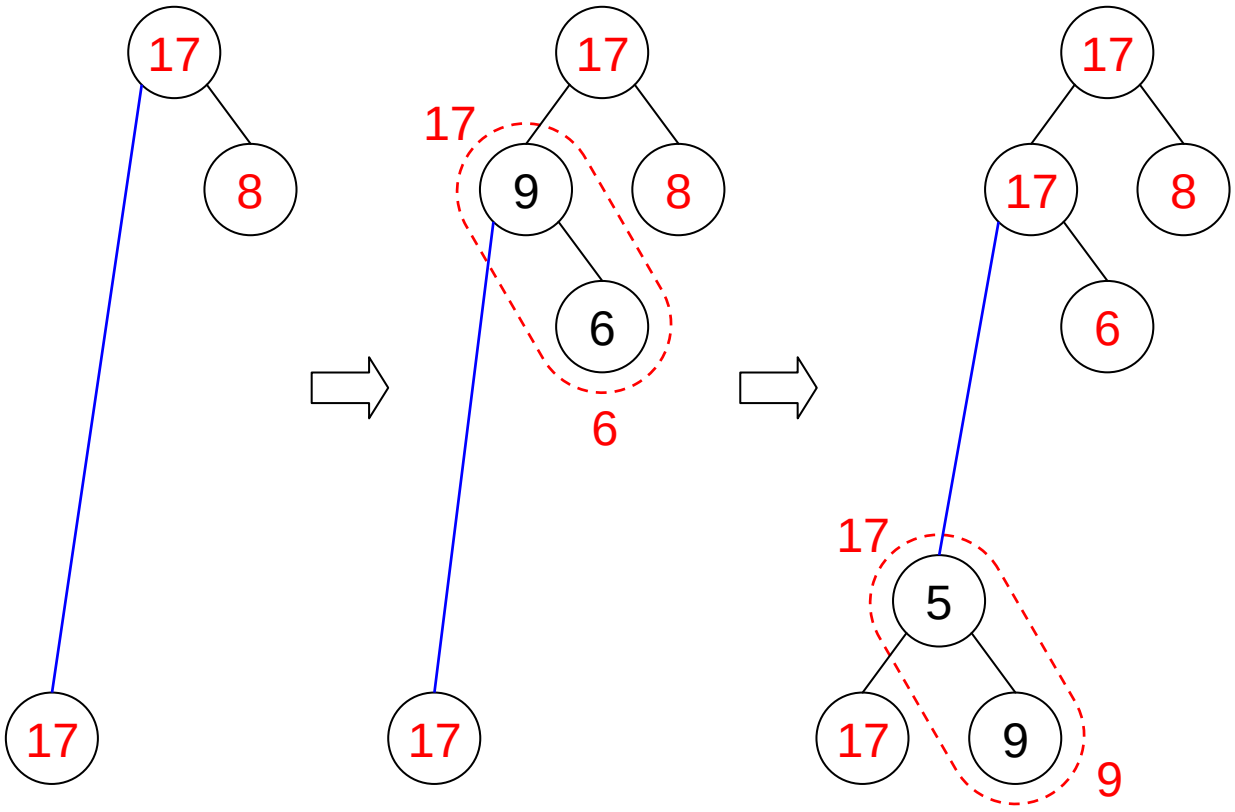
- Phase 1. Tree contraction

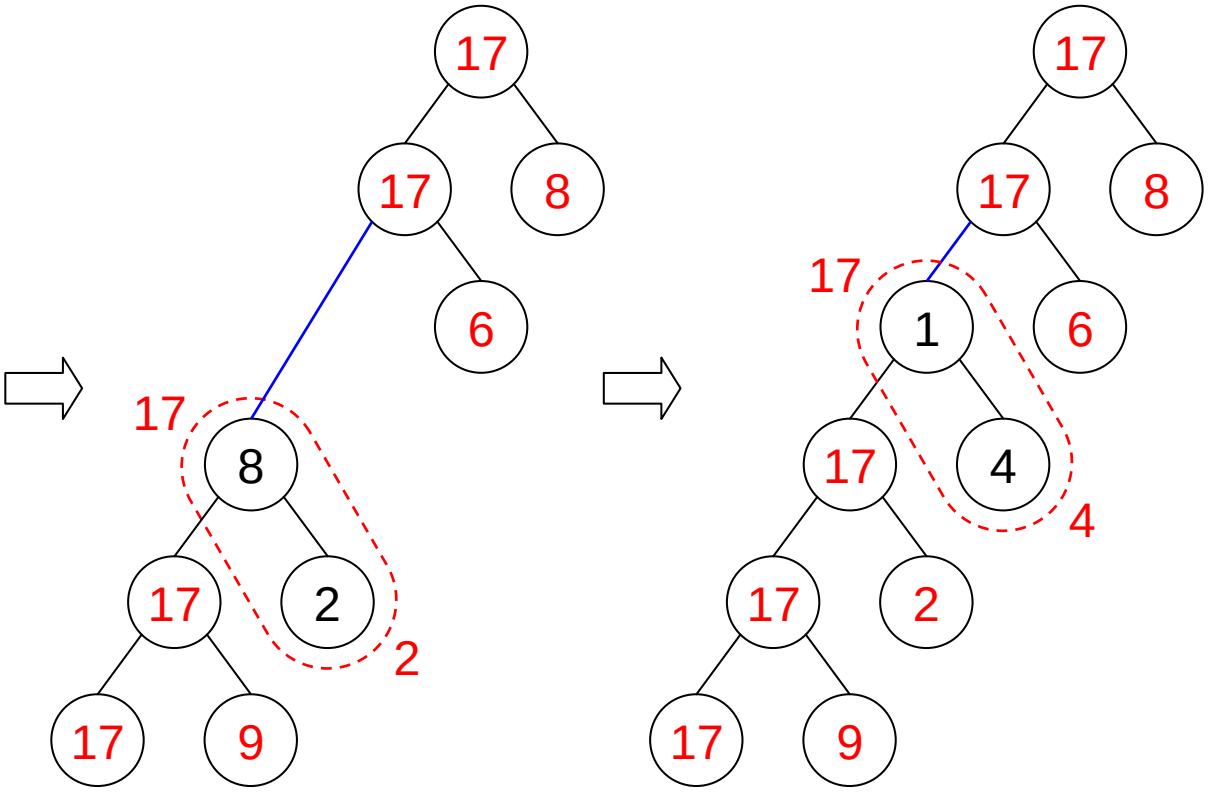


Note: for each node *v* in *T'*, the answer is the same


4
8
4
8
6
5
17
9
4
9
8
8
6
17
9
4
8
17
17
8
17

- **Phase 2:** Backtracking (all answers are computed)



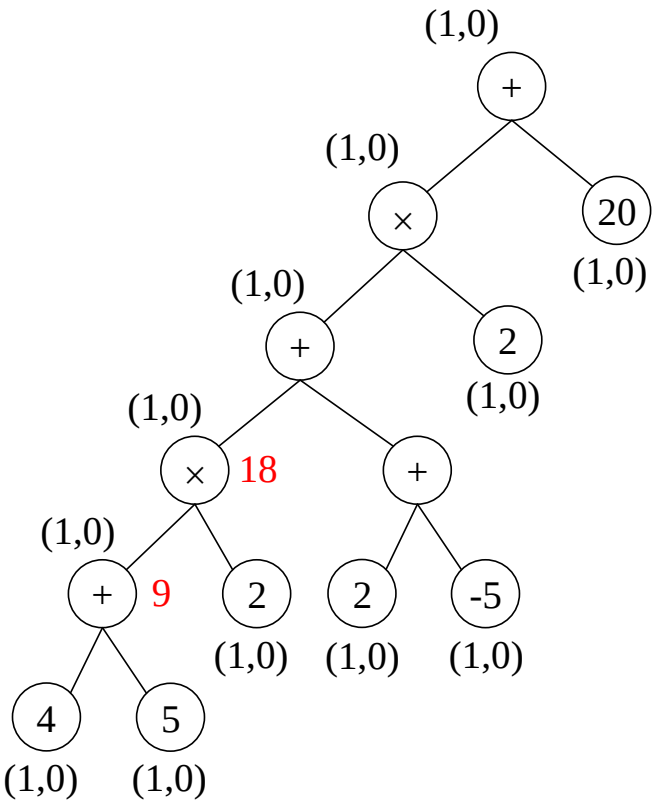


Problem SM-15: Evaluation of Computation Tree Forms

Input: A computation tree form *T* of an arithmetic expression, in which the operators are either + or *.

Output: *val*(*T*).

Model: EREW PRAM of *n* processors



$O(n)$

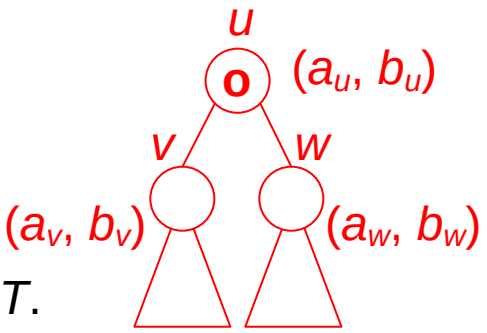
a computation tree form of (((((4+5)×2)+(2+(–5)))×2)+20)

Main Idea: We attempt to "partially evaluate" an internal node that has only a leaf node and then remove it.

To accomplish this partially evaluation, we associate with each node v a label (a_v, b_v) such that during the evaluating process, the following **invariant** is maintained.

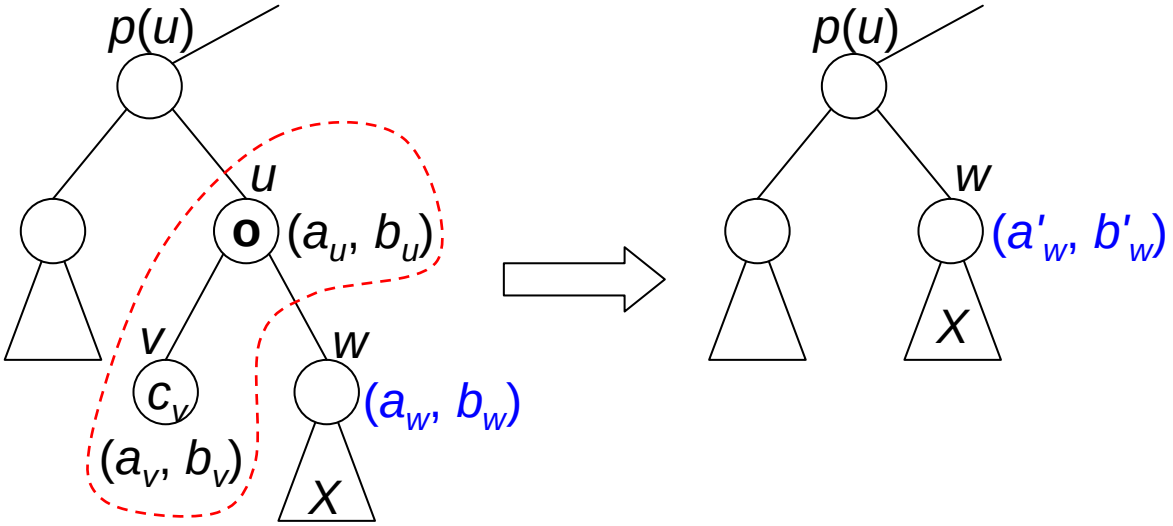
Invariant: Let u be an internal node of the current tree such that u holds the operator **o** ('+' or '×') and has the children v and w. Then,

$$val(u) = (a_v * val(v) + b_v) \mathbf{o} (a_w * val(w) + b_w).$$



Clearly, initially $(a_v, b_v) = (1, 0)$ for each v in T.

How to maintain the Invariant?



***** if **o** = '×',

$$\overbrace{a'_w X + b'_w}^{val(u)} = a_u((a_v c_v + b_v)\mathbf{o}(a_w X + b_w)) + b_u$$
$$= a_u((a_v c_v + b_v)\mathbf{o}a_w)X + a_u(a_v c_v + b_v)\mathbf{o}b_w + b_u$$

$$(a'_w, b'_w) = (a_u((a_v c_v + b_v)\mathbf{o}a_w),\ a_u(a_v c_v + b_v)\mathbf{o}b_w + b_u).$$

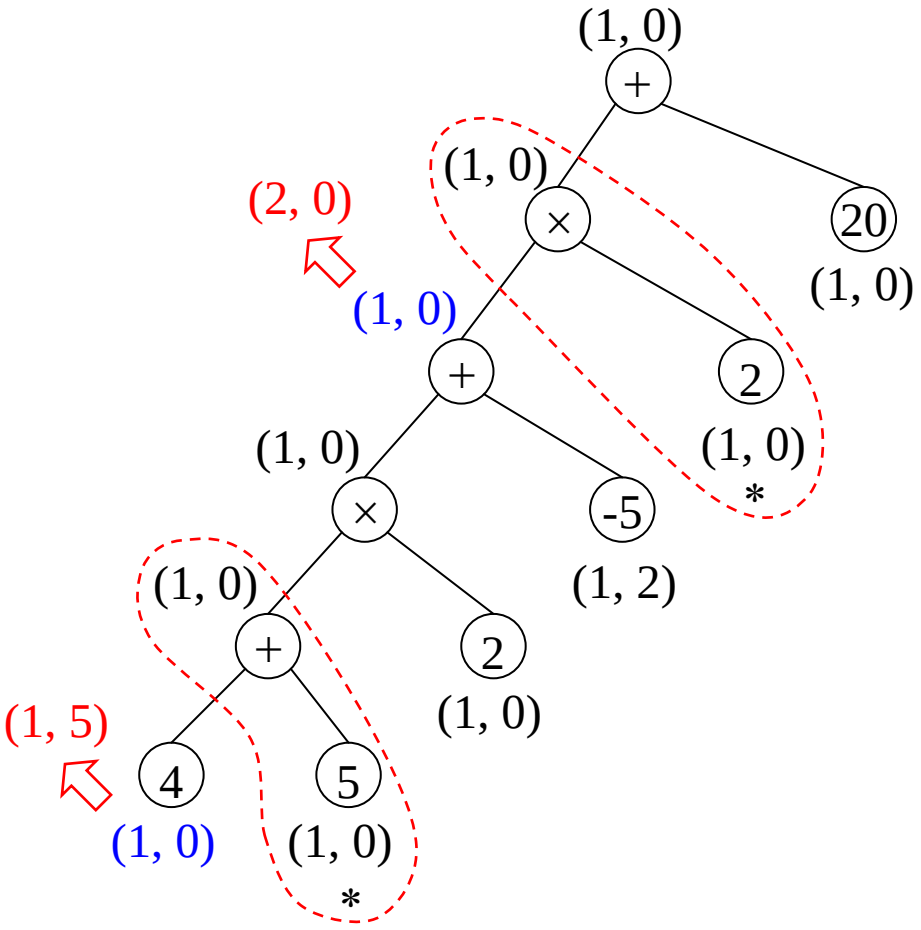
***** **o** = '+' is similar.

The algorithm

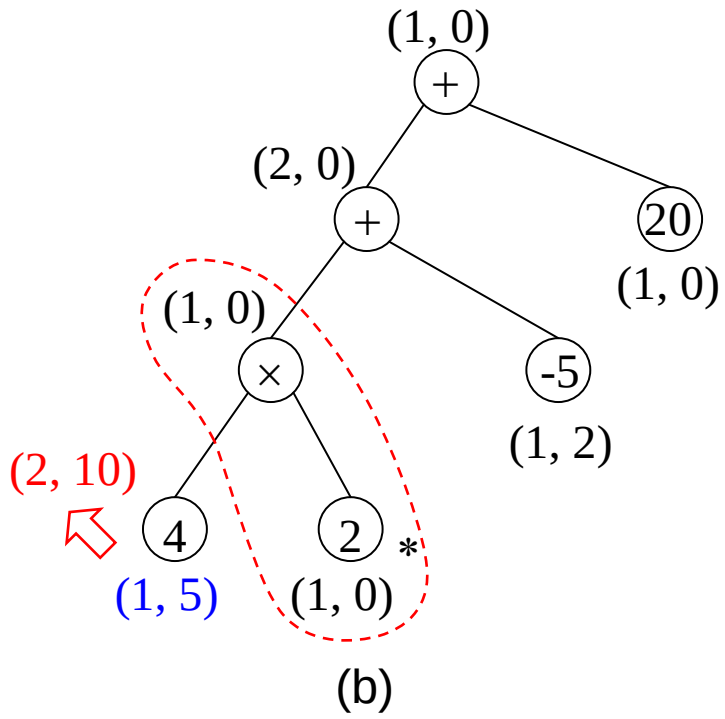
1. Initially, assign (1, 0) to each node.

2. Apply the tree-contraction algorithm to reduce the tree T into a 3-node T', such that $val(T)=val(T')$. (The label $(a_{sib(v)}, b_{sib(v)})$ is adjusted properly while a rake operation is performed at node v.)

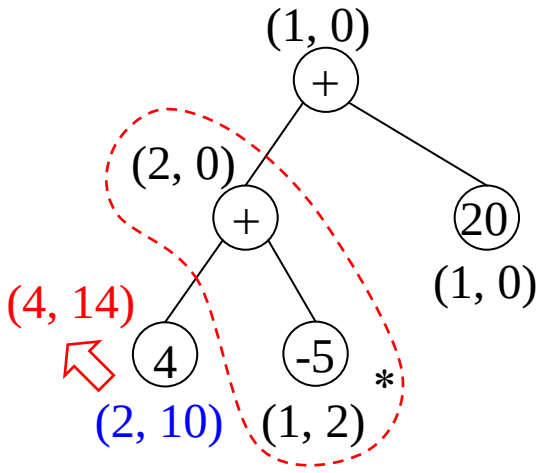
3. Determine $val(T')$



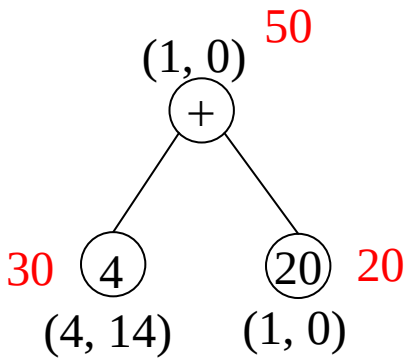
(a)



(b)



(c)



(d)

* $T(n) = O(\log n)$, cost optimal by using Brent's Theorem.

* We can compute the value of each node by processing the tree in reverse order. (backtracking)

Problem SM-16: Searching
Input: a sorted sequence $A[1, n] = \{1, 2, \ldots, 18\}$, and a value $x = 11$
Output: k such that $A[k] = x$
Model: CREW PRAM of p processors
(Assume that k uniquely exists.)
step 0: *beginning* = 0; *length* = n;

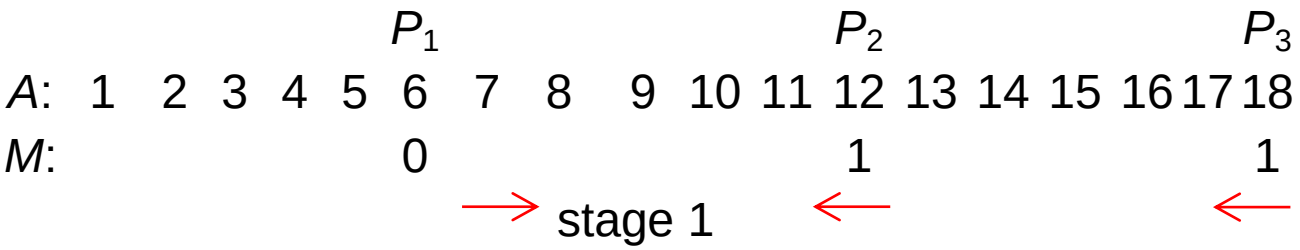
step 1: If (*length* $\le p$)
each processor P_i, $1 \le i \le$ *length*, sets k = *beginning* + i if A[*beginning* + i] = x; and then, terminates.

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.

(x = 11, n=18, p=3)



B 7 *beginning* = 6
E 12 *length* = 6

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

stage 2

											P_1	P_2						
A:	*	*	*	*	*	*	*	*	*	*	11	12	*	*	*	*	*	*

$k = 11$

stage 3

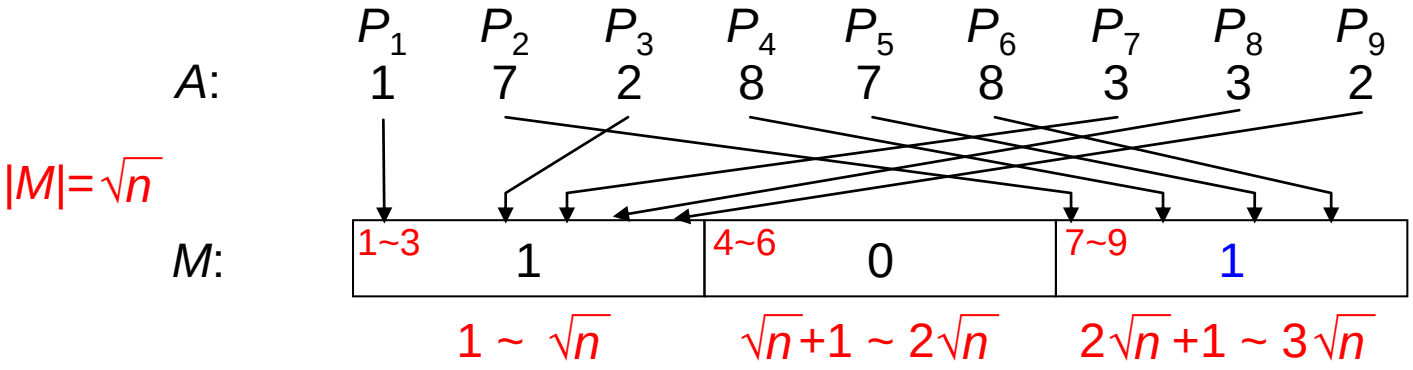
* $T(n) = T(n/p) + O(1) = O(\log_p n)$

* While $p = n^{1/k}$ for some constant k, the proposed algorithm requires $O(1)$ time. We have an example of $k = 2$ in **Problem** SM-3 (convex hull).

Problem SM-17: Finding Maximum in a Restricted Domain
Input: $A[1…n] = \{1, 7, 2, 8, 7, 8, 3, 3, 2\}$ (assume $1 \le a_i \le n$)
Output: **max**{ $A[1], A[2], …, A[n]$ }
Model: CRCW PRAM of n processors $O(1)$ time

(F. E. Fich, P. L. Ragde, and A. Wigderson, "Relations between current-write models of parallel computation," *SIAM J. Comput.* 17 (1988), pp. 606-627.)

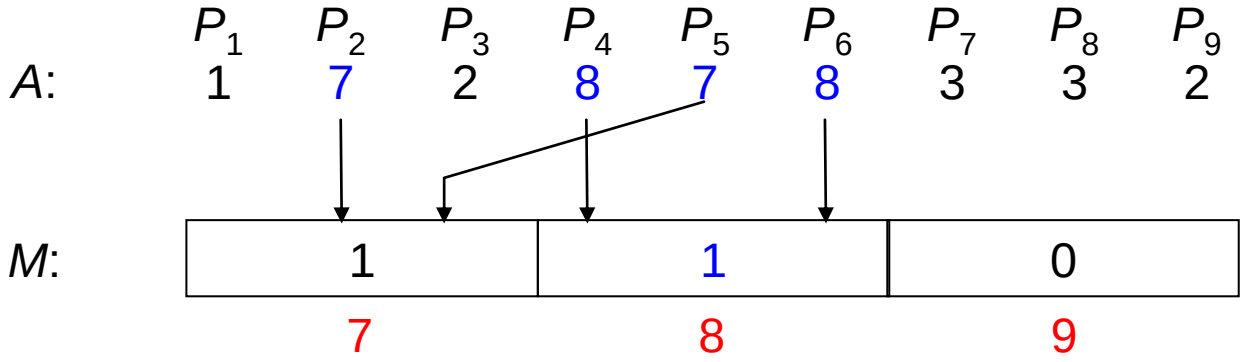
step 1: Each processor P_i sets $M[i] = 0$ if $(1 \le i \le n^{1/2})$, and then sets $M[(A[i] + (n^{1/2} - 1))/n^{1/2}] = 1$. $n = 9$



step 2: Find m = **MAX**($M[1]\times1$, $M[2]\times2$, …, $M[n^{1/2}]\times n^{1/2}$}.
Using n PEs, this takes $O(1)$ time. (Pro. SM-7)

m = **MAX**{1×1, 0×2, 1×3} = 3

step 3: Each processor P_i sets $M[i] = 0$ if $(1 \le i \le n^{1/2})$, and then sets $M[A[i]-(m-1)\times n^{1/2}] = 1$ if $(A[i] > (m-1)\times n^{1/2})$.



step 4: Find max = **MAX**($M[i]\times(i+(m-1)*n^{1/2})$ | $1 \le i \le n^{1/2}$}.

max = **MAX**{1×(1+6), 1×(2+6), 0×(3+6)} = 8
7 8 9

* $T(n) = O(1)$