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: \( n \)

\( n = 12 \quad p = n/\log n = 4 \)

\[
A: \begin{array}{ccccccccc}
1 & 1 & 2 & 1 & 4 & 2 & 2 & 3 & 1 & 2 & 1 & 2 \\
\end{array}
\]

\( \log n \) initially (\(|A| = n\))

\[
A': \begin{array}{cccccc}
4 & 7 & 6 & 5 \\
\end{array}
\]

stage 1 (\(|A'| = n/\log n\)) \( O(n/p) = O(\log n) \)

\( S': \begin{array}{cccccc}
4 & 11 & 17 & 22 \\
\end{array}
\)

stage 2 \( O(\log p) \)

\[
A: \begin{array}{cccccccccc}
1 & 1 & 2 & 1 & 4 & 2 & 2 & 3 & 1 & 2 & 1 & 2 \\
\end{array}
\]

\[
S: \begin{array}{cccccccccc}
4 & 11 & 17 & 22 \\
\end{array}
\]

initially of stage 3

\[
A: \begin{array}{cccccccccc}
1 & 1 & 2 & 1 & 4 & 2 & 2 & 3 & 1 & 2 & 1 & 2 \\
\end{array}
\]

\[
S: \begin{array}{cccccccc}
1 & 2 & 4 & 5 & 9 & 11 & 13 & 16 & 17 & 19 & 20 & 22 \\
\end{array}
\]

stage 3 \( O(n/p) \)

* Accelerated cascading

\[
A_1 \text{ fast, non-optimal} + A_2 \text{ slow, optimal} = \text{ fast, optimal}
\]
Problem SM-11: Selection

Input: $S[1…n], k$

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)*

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. (\(|S_1| \leq 3|S|/4 \) and \(|S_3| \leq 3|S|/4\))

<table>
<thead>
<tr>
<th>step</th>
<th>Time</th>
<th>1. O(1)</th>
<th>2. ( O(n) )</th>
<th>3. ( T(n/r) )</th>
<th>4. ( O(n) )</th>
<th>5. ( T(3n/4) )</th>
</tr>
</thead>
</table>

\[ T(n) = T(n/r) + T(3n/4) + O(n) \]

By induction, \( T(n) = O(n) \) is derived.
(2) **Parallel approach**

1. Divide $S$ into $n^{1-\varepsilon}$ subsequences of length $n^{\varepsilon}$. 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 $n$, $k$, $\varepsilon$ are stored in the shared memory)

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

$O(\log p)$ (i) The values of $a$, $n$, $k$, $\varepsilon$ 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.
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 $S_{i,3}$. ($O(n)$ time)

(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) $O(\log p)$

find $S_1$

(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!


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

\[ A = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & \end{bmatrix} \]

an undirected graph \( G \)  
two connected components of \( G \)
Initially

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
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 $L(x_i) = \min\{L(j) \mid A[i, j] = 1\}$. Then, determine the node $y_i$, where $L(y_i) = \min\{L(x_j) \mid L(j) = L(i)\}$. And then, set $\text{Next}(i) = i$ if $L(y_i) = L(i)$. Otherwise, set $\text{Next}(i) = y_i$.

step 2. (Combine sub-components together. $O(\log n)$ time.)

for $i, 1 \leq i \leq \log n$, do

begin

$L(i) = L(\text{Next}(i))$  

pointer jumping

$\text{Next}(i) = \text{Next}(\text{Next}(i))$

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.)

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

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:

\[ T \]

\[
\begin{array}{cccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \\
+1 & -1 & +1 & -1 & +1 & -1 & +1 & -1 \\
\end{array}
\]

\[ O(\log n) \text{ time} \]

\[ \Rightarrow \left\{ \begin{array}{c}
O(\log n) \text{ time} \\
O(n) \text{ cost}
\end{array} \right. \]
Problem SM-13: Tree Contraction

Input: $P[1...n]$ and $SIB[1...n]$: 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 \(A_{\text{odd}}\) that are left children. \((A_{\text{odd}} = (a_1, a_3, a_5, \ldots))\)

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

2.3. Set \(A = A_{\text{even}}\). \((A_{\text{even}} = (a_2, a_4, a_6, \ldots))\)

\[
\begin{array}{c}
\text{after step 1} \\
\end{array}
\]
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)

\[ O(n) \times \]
Applying tree contraction

idea: partial evaluation

- Phase 1. Tree contraction

Note: for each node $v$ in $T'$, the answer is the same
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**: $\text{val}(T)$.

**Model**: EREW PRAM of $n$ processors
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.

*a computation tree form of $((((((4+5)\times2)+(2+(-5)))\times2)+20)$*
**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,

$$\text{val}(u) = (a_v \cdot \text{val}(v) + b_v) \cdot (a_w \cdot \text{val}(w) + b_w).$$

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

**How to maintain the Invariant?**

* if $o = \times$, 
  $$a'_w X + b'_w = a_u((a_vc_v + b_v) \cdot (a_w X + b_w)) + b_u$$
  $$= a_u((a_vc_v + b_v) \cdot a_w)X + a_u(a_vc_v + b_v) \cdot b_w + b_u$$

  $$(a'_w, b'_w) = (a_u((a_vc_v + b_v) \cdot a_w), a_u(a_vc_v + b_v) \cdot b_w + b_u).$$

* $o = \div$ 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 \(\text{val}(T) = \text{val}(T')\). (The label \((a_{\text{sib}(v)}, b_{\text{sib}(v)})\) is adjusted properly while a rake operation is performed at node \(v\).)

3. Determine \(\text{val}(T')\)
\[ T(n) = O(\log n), \text{ cost optimal by using Brent's Theorem.} \]

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

step 2: Each processor $P_i$, $1 \leq i \leq n$, sets $M[i]$ as 1 if $x \leq A[beginning + i \times \frac{length}{p}]$ and 0 otherwise.

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

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

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

\[
\begin{array}{cccccccccccccccc}
M: & 0 & & & & & & 1 & & & & & & & & & 1 & & \\
\end{array}
\]

→ stage 1 ← ←

$B \quad 7 \quad beginning = 6$

$E \quad 12 \quad length = 6$
Problem SM-17: Finding Maximum in a Restricted Domain
Input: $A[1…n] = \{1, 7, 2, 8, 7, 8, 3, 3, 2\}$ (assume $1 \leq a \leq n$)
Output: $\max\{ A[1], A[2], \ldots, A[n] \}$
Model: CRCW PRAM of $n$ processors $\mathcal{O}(1)$ time

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

\[ n = 9 \]

\[
\begin{array}{cccccccccc}
& P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 \\
A: & 1 & 7 & 2 & 8 & 7 & 8 & 3 & 3 & 2 \\
M: & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
\end{array}
\]

|\[
|M| = \sqrt{n}
\]

\[
\begin{array}{cccccccc}
1 \sim 3 & 1 & 4 \sim 6 & 0 & 7 \sim 9 & 1 \\
\end{array}
\]

\[ 1 \sim \sqrt{n} \quad \sqrt{n+1} \sim 2\sqrt{n} \quad 2\sqrt{n+1} \sim 3\sqrt{n} \]

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

\[ m = \text{MAX}\{1 \times 1, 0 \times 2, 1 \times 3\} = 3 \]

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

\[
\begin{array}{cccccccccc}
& P_1 & P_2 & P_3 & P_4 & P_5 & P_6 & P_7 & P_8 & P_9 \\
A: & 1 & 7 & 2 & 8 & 7 & 8 & 3 & 3 & 2 \\
M: & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{cccc}
7 & 8 & 9 \\
\end{array}
\]

step 4: Find \[ \text{max} = \text{MAX}(M[i] \times (i+(m-1)\times n^{1/2}) \mid 1 \leq i \leq n^{1/2}) \].

\[ \text{max} = \text{MAX}\{1 \times (1+6), 1 \times (2+6), 0 \times (3+6)\} = 8 \]

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