Deterministic list ranking

Basic idea: Colour list so that adjacent elements have different colour, eliminate elements of any single colour.

Algorithm for processor i:

(N processors total)

1. Legal colouring:

   colour[i] = i

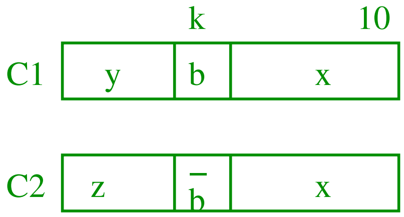
2. Colour reducing step:

   c_1 = colour[i], c_2 = colour[next[i]]

   k = LSB in which c_1 and c_2 differ.

   b = bit k of c_1.

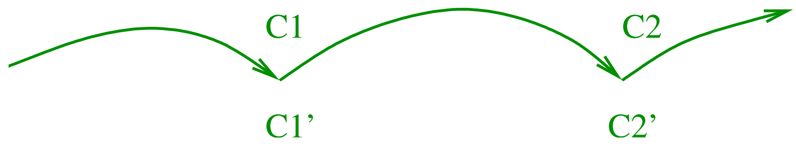
   colour[i] = $2k + b$



Example: $c_1 = 15 = 01111$, $c_2 = 19 = 10011$
$k = 2, b = 1 \Rightarrow c_1' = 5$

Correctness

Lemma: Valid colouring before $\Rightarrow$ Valid colouring afterward.



Proof:

Assume $c_1' = c_2'$

$c_1' = 2k_1 + b_1$, Bit k_1 of c_1 is b_1.

$c_2' = 2k_2 + b_2$, Bit k_2 of c_2 is b_2.

$c_1' = c_2' \Rightarrow b_1 = b_2, k_1 = k_2$

Bit k_1 of c_1 = Bit k_1 of c_2.

But c_1 and c_2 must differ in bit k_1.

Reduction in number of colours

Effect of one iteration:

Initial colour range: $[0, n-1]$

New colour $= 2k + b$, $\max(k) = \lfloor \log n - 1 \rfloor$

Max new colour $= 2 \lfloor \log n - 1 \rfloor + 1$

Number of new colours $= 2 \lfloor \log n - 1 \rfloor + 2$

Fixed point: Colours decrease until $n = 6$:

$$2 \lfloor \log(5) \rfloor + 2 = 6$$

Overall effect:

Colors initially: n

Colors after iteration 1: $2 \lfloor \log n - 1 \rfloor + 2$

Colors after iteration 2: $2 \lfloor \log 2 \lfloor \log n - 1 \rfloor + 2 \rfloor + 2$

Colors after iteration i: $O(\log^{(i)} n)$

Colours after iteration $O(\log^* n)$: 6

List Ranking Algorithm

N = list size, P = number of processors.

1. Six-colour list.
2. Determine most common colour.
3. Eliminate nodes of chosen colour.
4. Compact remaining nodes.
5. Recurse, if $> P$ nodes remain. Else use Wyllie's algorithm.
6. Fill in values for elements eliminated in step 3.

$$T(N) = O(\tfrac{N}{P}\log^* N + \tfrac{N}{P} + \log P) + T(\tfrac{5N}{6}) \quad \text{where } N > P$$

$$T(P) = O(\log P)$$

$$\Rightarrow T(N) = O(\tfrac{N}{P}\log^* N + \log P \log \tfrac{N}{P})$$

$$N = P \log P \log \tfrac{N}{P} \Rightarrow T = O(\tfrac{N}{P}\log^* N)$$

Determining Most Common Colour

N elements, P processors.

Determining frequency of colour i: $i = 1 \ldots 6$

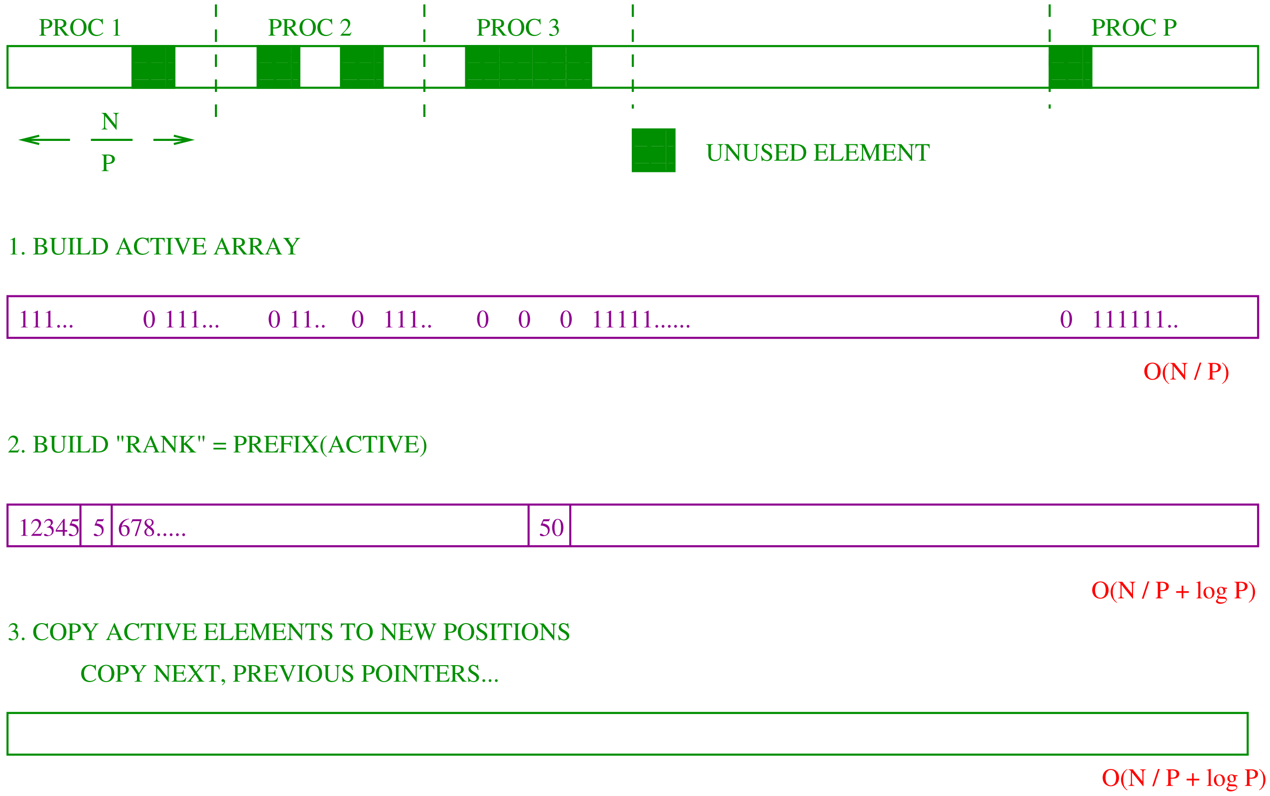
1. Each processor given $\frac{N}{P}$ elements. Computes number of elements of colour i among these.
   Time $= \frac{N}{P}$.
2. Global sum. Time $= \log P$.

Most common colour:

1. Find frequency of each of 6 colours.
   Time $= O(\frac{N}{P} + \log P)$.
2. Take max. Time $= O(1)$.

Total: $O(\frac{N}{P} + \log P)$

Compacting a List



Theorem

N element list can be 3-colored using P processors using in time $O\left(\frac{N}{P} + \log P\right)$

Proof: Homework
Sorting integers in $[0, \log P]$ can be done in time $O(\frac{N}{P} + \log P)$
1. Use 1 round of det. coin tosses
2. Rearrange list so that nodes are sorted by color
3. for $i = 3 \ldots \log(N-1)$,
   Recolor nodes with color i

$\Rightarrow$ N element list can be ranked in time $O(\frac{N}{P} + \log P \log\log P)$ using P processors

$\Rightarrow$ Linear speed up for $N = \Omega(P \log P \log\log P)$

Anderson-Miller, Cole-Vishkin : Time $= O(\frac{N}{P} + \log P)$
Linear speedup even for $N = \Omega(P \log P)$

Summary

Time to rank N element list using P processors (Simple randomized algorithm / complex deterministic algorithm of Cole Vishkin/ Anderson Miller)

$$O\left(\frac{N}{P} + \log P\right)$$

Time on P processor butterfly using PRAM Butterfly

$$O\left(\log P\left(\frac{N}{P} + \log P\right)\right)$$

Butterfly lowerbound

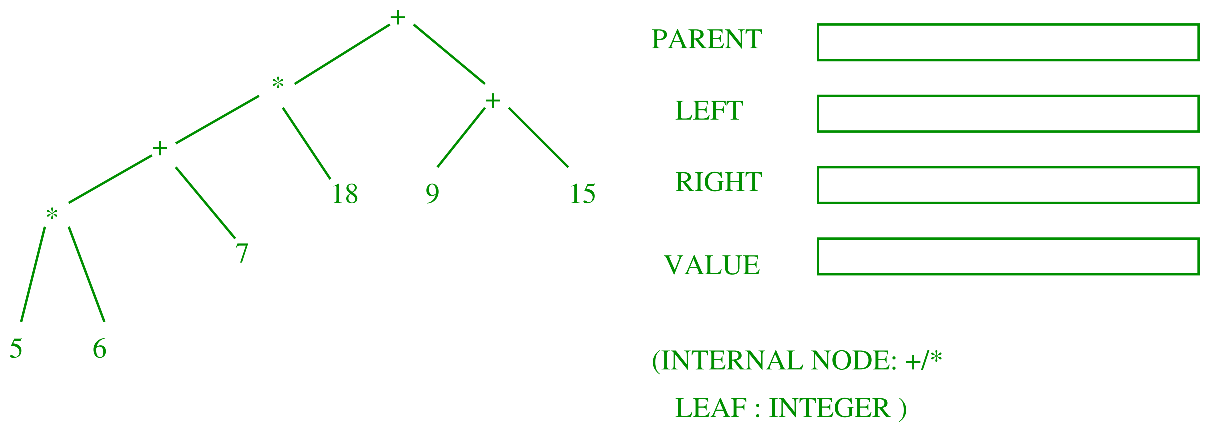
$$\Omega\left(\frac{N}{P}\log P\right) \quad \text{... later}$$

Time for $N = P\log P$

$$\theta\left(\log^2 P\right)$$

Expression Tree evaluation

INPUT:



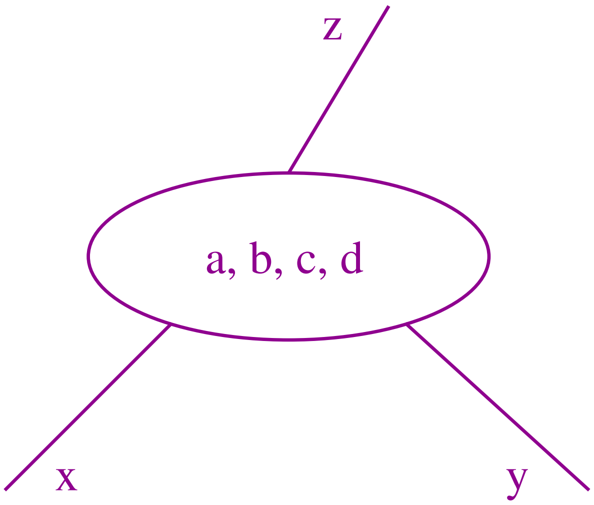
OUTPUT : VALUE OF EXPRESSION

Theorem : Any N node expression tree can be evaluated on a P processor EREW PRAM in time $O\left(\frac{N}{P} + \log P\right)$

$\Rightarrow$ Linear speedup for $N = \Omega(P \log P)$

Main Ideas : Tree can be "constructed" in parallel without altering value–Use list ranking to coordinate

Generalized Node Representation



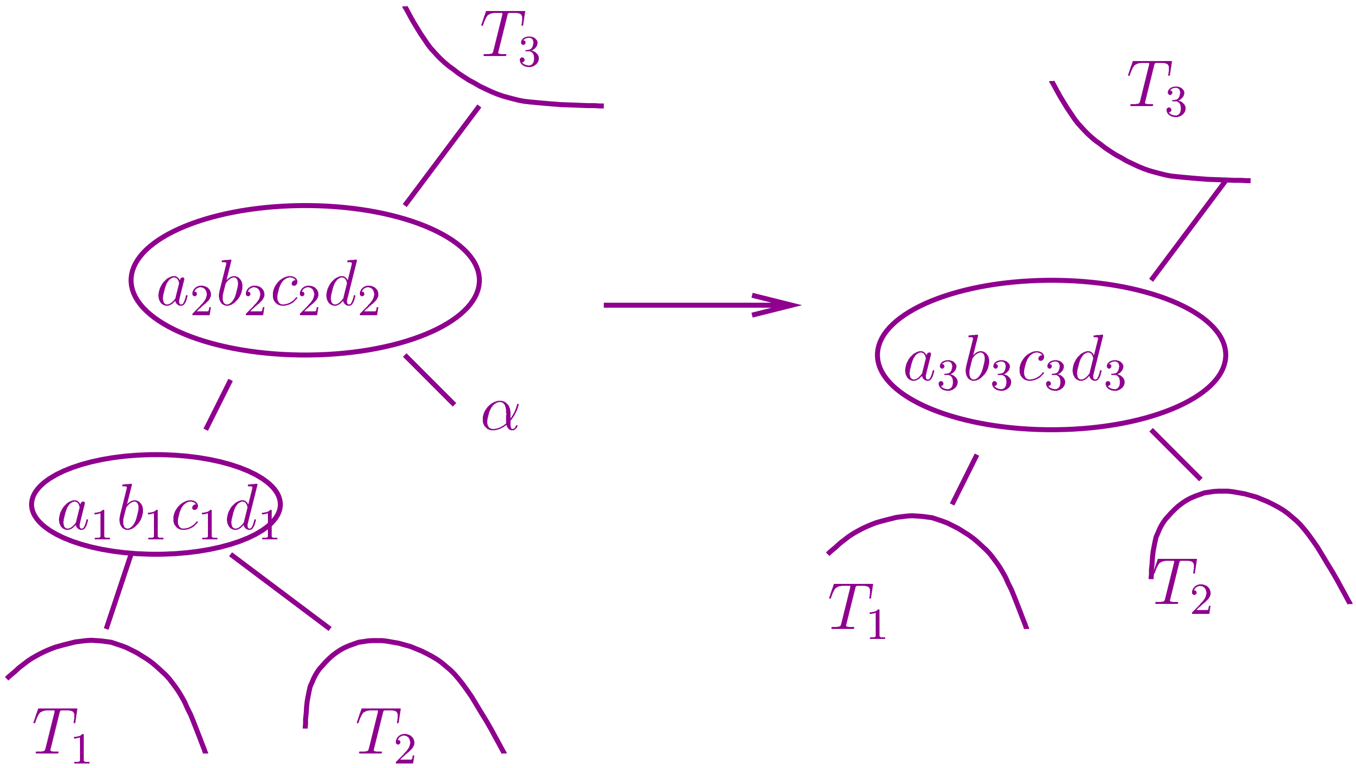
Action of an operator (a, b, c, d)

$$z = axy + bx + cy + d$$

"+" : $a = d = 0,\ b = c = 1$
"*" : $a = 1\ ,\ b = c = d = 0$

Leaf with value v : $a = b = c = 0,\ \ d = v$

Raking a Leaf

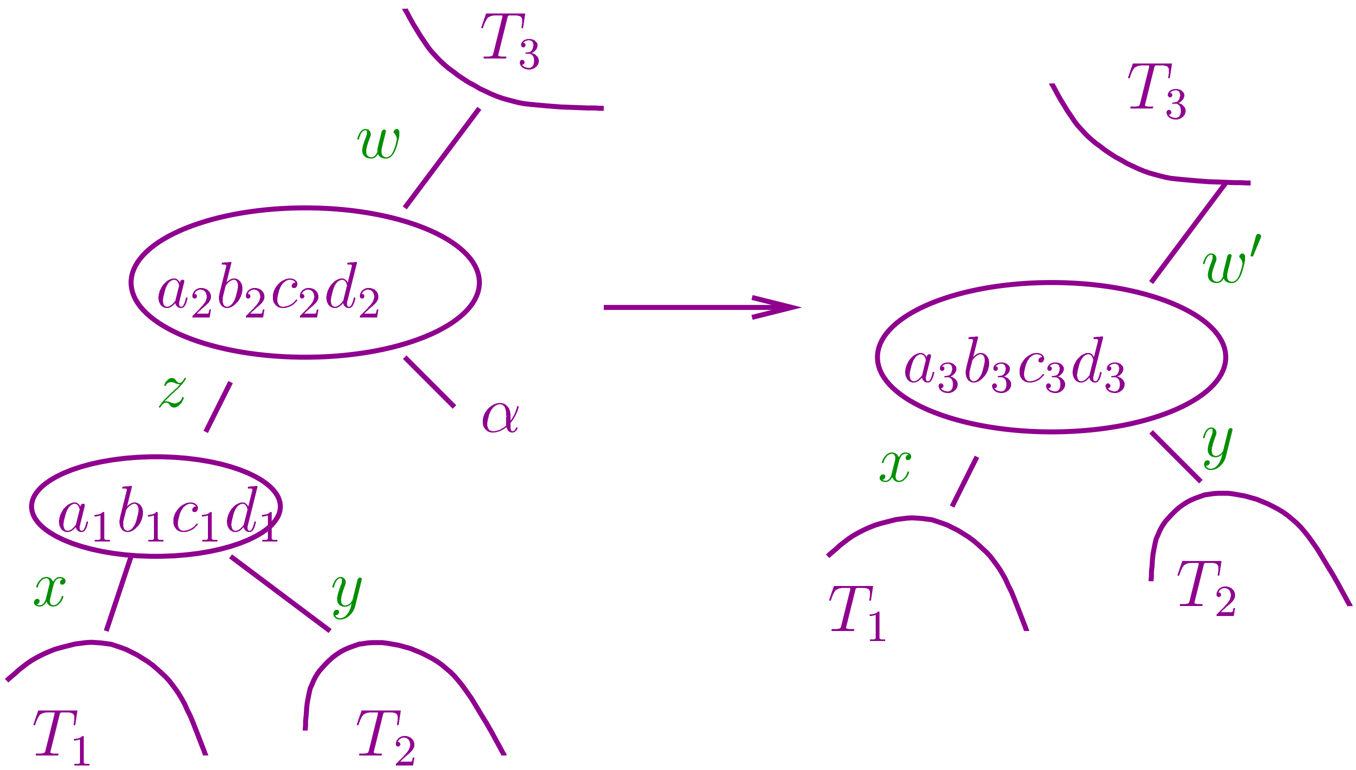


$$
\begin{aligned}
a_3 &= a_1a_2\alpha + a_1b_2 \\
b_3 &= b_1a_2\alpha + b_1b_2 \\
c_3 &= c1a_2\alpha + c_1b_2 \\
d_3 &= d_1a_2\alpha + d_1b_2 + c_2\alpha + d_2
\end{aligned}
$$

Claim 1 *Raking does not change the value of tree*

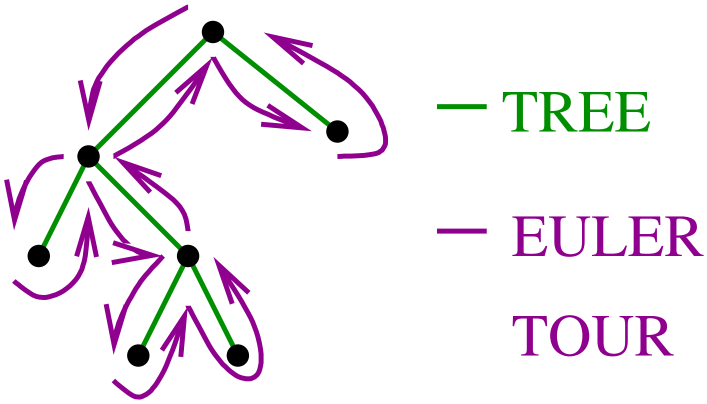
Observation : Raking is a local operation. Several leaves can be raked in parallel

Proof of Claim



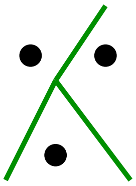
$$
\begin{aligned}
z &= a_1xy + b_1x + c_1y + d_1 \\
w &= a_2(a_1xy + b_1x + c_1y + d_1)\alpha \\
&\quad +b_2(a_1xy + b_1x + c_1y + d_1) \\
&\quad +c_2\alpha \\
&\quad +d_2 \\
&= Axy + Bx + Cy + D \\
w' &= a_3xy + b_3x + c_3y + d_3 \text{ where } a_3 = a_1a_2x + a_1b_2
\end{aligned}
$$

Euler Tour of a Tree



TREE NODE

$\equiv$ 3 tour element



Claim 2 *Given a tree, represented using arrays* Left, Right *we can construct Euler tour in* $O\left(\frac{N}{P}\right)$ *time*

$$\begin{aligned}
\text{Tree node } i &\equiv \text{Tour}[3i] \\
&\quad\ \text{Tour}[3i+1] \\
&\quad\ \text{Tour}[3i+2] \\
\text{Tour}[3i] &= 3\ \text{Left}[i] \text{ if Left}[i] \neq \text{Nil} \\
&\quad\ 3i+1 \text{ or else}
\end{aligned}$$

Leaf Numbering

Define Value$[0 \ldots, (3N-1)]$

Value$[3i+2] = 1$ if i is leaf

=0 or else

Value$[3i] = 0$

Value$[3i+1] = 0$

Compute list suffix using

- Tour
- Value

Algorithm

N: # Leaves in tree

1. Number of leaves left to right $0 \ldots N-1$
2. (a) Rake odd numbered left leaves
   (b) Rake odd numbered right leaves
3. $\left.\begin{array}{l}\text{(a)}\\ \text{(b)}\end{array}\right\}$ Leaves numbered $4i+2$
4. $\left.\begin{array}{l}\text{(a)}\\ \text{(b)}\end{array}\right\}$ Leaves numbered $8i+4$
5. $\vdots$

Time: Step 1: $O\left(\frac{N}{P} + \log P\right) \ldots$ Next
Step 2...: $O\left(\frac{N}{P} + \log P\right) \ldots$ Homework