20 Sep 2021 Parallel Algorithms Intro

Models of parallel algorithms.

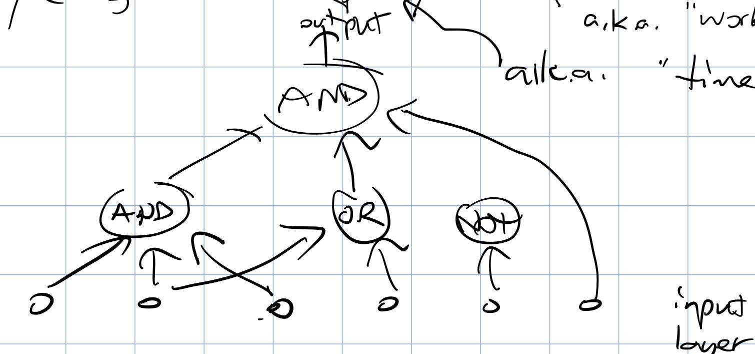
PRAM: CRCW, CREW, EREW

NC: ("Nick's class")

Boolean circuits (DAG's made up of AND, OR, NOT gates) with poly(n) size (# gates) and poly(log(n)) depth.

a.k.a. "work"

a.k.a. "time"



If $k \in \mathbb{N}$,

An algorithm is AC^k if it can be represented by a circuit as above with poly(n) nodes and $O(\log^k(n))$ depth.

It is NC^k if it's AC^k and in addition all gates have "fan-in" (# incoming edges) ≤ 2.

Clearly $NC^k \subseteq AC^k \subseteq NC^{k+1}$.

Adding binary numbers...

$$
\begin{array}{cccccccc}
 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\
 & 1 & 1 & 0 & 1 & 1 & 0 & 1 \\
 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\
\hline
1 & 1 & 0 & 0 & 1 & 0 & 1 & 1
\end{array}
\qquad \text{"carries"}
$$

To compute carry digit in column i, must compute

$$f_{i-1}(f_{i-2}(f_{i-3}(\cdots f_0(0))\cdots)$$

where

$$f_j = \begin{cases} \emptyset & \text{if column } j \text{ is } 0+0 \\ \mathrm{Id.} & \text{if } \text{"} \quad \text{"} \quad 0+1 \text{ or } 1+0 \\ 1 & \text{if } \text{"} \quad \text{"} \quad 1+1 \end{cases}$$

New problem: "parallel prefix"

Suppose $\mathcal{F}$ is a family of functions closed under composition.

(E.g. $\{\emptyset, 1, \mathrm{Id.}\}$)

Given $f_\emptyset, f_1, \ldots, f_{n-1} \in \mathcal{F}$

compute $g_i = f_i \circ f_{i-1} \circ f_{i-2} \circ \cdots \circ f_0$ for all i.

__Alg.__ Define $h_{i,k} = f_i \circ f_{i-1} \circ \cdots \circ f_{i-2^k+1}$.

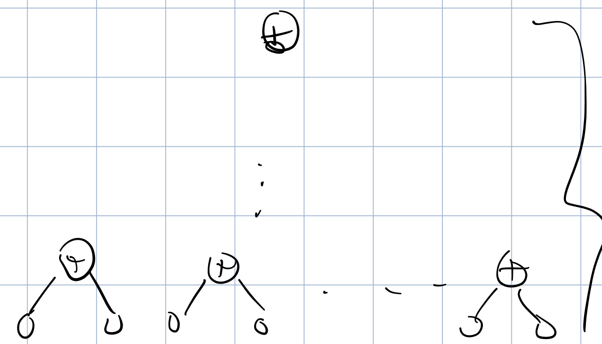
Observe $h_{i,0} = f_i$. (input layer)

$$h_{i,k} = h_{i,k-1} \circ h_{i-2^{k-1},k-1} \quad \text{(layer } k\text{)}$$

$$h_{i,\lceil \log_2(n) \rceil} = f_i \circ f_{i-1} \circ \dots \circ f_0.$$

Adding up n binary numbers, each b bits long.

Obvious alg:



depth $\log(n)$

This is NC^2.

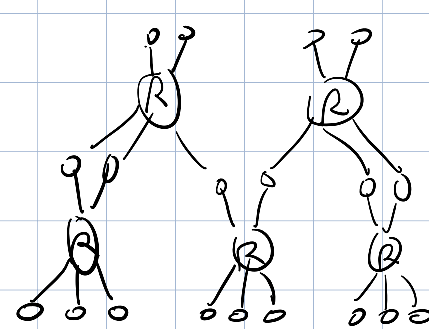
1 1 1 1 0 0 1

1 0 0 0 1 1 1

0 0 1 1 0 0 1

10 1 0 0 1 1 1

0 0 1 1 0 0 1 0



So adding up n numbers of b bits each ~~takes~~ $O(\log(bn))$ time.

Integer multiplication, $O(\log n)$ time using the above algorithm to sum up partial products.

Matrix multiplication. Just comes down to computing n^2 dot products in parallel.

Dot product is

- n mult's in parallel $O(\log b)$
- summing n numbers of size $\leq 2b$. $O(\log(bn))$.

Overall, mtrx mult is $O(\log(bn))$

Matrix determinant? Next lecture.