So far we have assumed the following model: RAM model

\[ (\text{unit time ops}) \]
\[ (+_x \div (\log n \text{ bits})) \]
\[ (\text{read/write into memory}) \]

A central model to describe Graph Algorithms.

Other models:
1. Ants
2. Pointer machines
RAM is unrealistic as $n$ goes to infinity.

1) Speed of light (large size machines)
2) Quantum effects (small size machines)

Bottom line: RAM

1) Many important algorithms were found using this model.
2) Most algorithms are coded in a RAM-like language.
   eg C
Parallel Models

Fixed connection machines
machines = infinite state machine
= RAM

A) Cellular Arrays 1D, 2D, 3D

1940's von Neumann
60's, 70's algorithms for CA,
Alvy Ray Smith 1974
80's Wolfram
Klaus Sutner
Conway Game of Life
60's Edgar Codd
80's HT Kung
Highly connected models

1) Hypercube \(= (V, E) \) 1980's

\[ V = \{ (a_1, \ldots, a_m) \mid a_i \in \{0,1\} \} \quad m = \log n \]

\[ (a_1, \ldots, a_m), (a_1, \ldots, \bar{a}_i, \ldots, a_m) \in E \]

2) Shuffle-exchange graph 1980's

\[ V = \{ (a_1, \ldots, a_m) \mid a_i \in \{0,1\} \} \]

\[ ((a_1, \ldots, a_m), (\bar{a}_1, a_2, \ldots, a_m)) \in E \]

\[ ((a_1, \ldots, a_m), (a_m a_1, a_2, \ldots, a_m)) \in E \]

3) Randomly connected graphs

possible models of the brain!

\( (\text{Valiant}) \)
Shared memory models

1) PRAM (Parallel Random Access Machine)

- Processors

Unit time ops read/write

<table>
<thead>
<tr>
<th>+, X, (:)</th>
<th>Read</th>
<th>Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>ER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EW</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CR</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CW</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PRAM issues:

Penalty for CR on an ER machine?
1) Machine crashes!
2) Garbage read!

How is synchronization handled?
1) After each unit of time!
2) Synchronous Parallel BSP Valiant
3) None!
Circuit Model

Inputs in either bits or words

node

nodes: 1) \& , V, \lor gates
2) Arithmetic ops

1) Constant fan in.
2) Arbitrary fan out.

Work = \# nodes

Time = longest path from input to output

Span (15-210)

Depth

Neural Nets
Naive Matrix Multiply

in the Circuit Model

\[ C_{ij} = \sum_{k=1}^{n} A_{ik} B_{kj} \quad C = A \cdot B \]

Input: \( A_{nn} \cdots A_{nn} \quad B_{nn} \cdots B_{nn} \)

\( A_{nn} \times B_{nn} \)

\( A_{ij} \times B_{kj} \)

\( A_{nn} \times B_{nn} \)

\( A_{in} \times B_{nj} \)

\( C_{i j} \)

\( C_{nn} \)

2n^2 input wires

n^3 nodes

Depth: \( \log n \), Size: \( n-1 \)

n^2 output wires
Circuit Total:

\[
\begin{align*}
\text{Work} & : O(n^3) \\
\text{Time} & : O(\log n)
\end{align*}
\]

Naive MM on PRAM

\[P = \# \text{processors} \quad T = \text{Parallel Time}\]

1) 1-processor/node CREW

\[
\begin{align*}
P & : O(n^3) \\
T & : O(\log n)
\end{align*}
\]

Note: Fan-out = reads

\[
\text{Fan-in} = \text{writes}
\]

2) Easy CREW (each processor reads its arguments)

3) EREW (Use binary tree to make copies)

\[
\begin{array}{c}
A_n \\
\downarrow \\
A_n \\
\downarrow \\
A_n \\
\downarrow \\
A_n \\
\downarrow \\
A_n
\end{array}
\]

This increases depth by additive \(\log n\)
PRAM Work = P \cdot T

Pay for each processor for life of run.

So far $O(n^3 \log n)$ work Alg.

Claim: $O(n^3/\log n)$ processor $O(\log n)$ time

for Naive MM Alg.

Start with $n^3$ mutts:

$A_{nn} \times B_{nn}$

$log$ size blocks

Each $P_i$ computes $\log$ mutt on $O(\log n)$ time.

$\text{mutt: } O(n^3/\log n) = P$

$O(\log n) = T$
Additions:

\[ a_1, a_2, \ldots, a_{n-1}, a_n \]

Replace with:

\[
(a_1 + \cdots + a_{2n}) \cdot (a_{4n} - a_{2n}) - (a_{6n} + \cdots + a_n)
\]

Add \( n \) numbers with \( O(n/\log n) \) processors in \( O(mn) \) time.

Finished Claim
The Slow Down Principle:

Given parallel alg Processor \( P \) & Time \( T \)

\( \forall P' \leq P \) run alg time \( (P'/P)T \) using \( P' \) processors.

Each processor simulates \( P/P' \) virtual processors

**Strassen's Alg**

Recall: Recurrence

\[
MM(n) = 7 \cdot MM(\sqrt{n}) + Cn^2
\]

\[\uparrow\]

7 recursive calls

\[\uparrow\]

matrix additions
Note: Matrix addition
$O(n^3)$ work
$O(1)$ time

Time! $T(n) = T(n^{1/2}) + O(1)$

parallel calls

$O(\log n)$

Processors $P(n) = 7P(n^{1/2}) + Cn^2$

we must pay for each call!

$O(n^{2.81})$

Work: $O(n^{\log_2 7} \log n)$