3.1 PRAM: Parallel Random Access Machine

Each processor has its own set of registers and program counter.
Assume that reading and writing takes unit time.
Assume that there is a global synchronized clock so all processors work in lockstep (this is the least realistic of the assumptions).

3.1.1 Read Write Conflicts

- Exclusive Read = one read from a memory location at a time
- Concurrent Read = multiple reads from a memory location at a time
- Exclusive Write = one write to a memory location at a time
- Concurrent Write = multiple writes to a memory location at a time
  - Arbitrary: an arbitrary one of the write attempts wins
  - Priority: Assume there is a priority on the processors and whichever processor has highest priority wins
  - Combining: somehow combine values (sum, max, etc)
  - Detection: some special value gets written when there is a concurrent write

You can combine either of the read strategies with either of the write strategies, but the exclusive read - concurrent write combination is usually not considered

- The only reason to do an exclusive read is if a concurrent read is too expensive
- If a concurrent read is too expensive, then a concurrent write will probably also be too expensive
3.2 Networked Models

Some examples of networks are:

- Crossbar
- Butterfly
- Grid
  - Bus - only one request can go through at once

Latency = time message takes to get through
Throughput = how often a message can be injected into the network

For the purpose of simulating PRAM assume

- Latency = log n
- Throughput = one message per cycle
- memory split up sequentially across memory banks

If multiple processors request data from the same memory bank, since the throughput is 1 per cycle, the requests queue up to the memory banks and it has to send data back one at a time so reading and writing is not unit time in those situations.

One proposed solution is to randomly distribute memory to memory banks

- This doesn’t guarantee there will be no problems but it reduces the chance that it will happen
- This would still have problems if multiple processors try to read the exact same memory location
  - A fix to this would be to combine identical requests into a single request and then break up the response on the way back so it gets to everyone who requested it
3.3 DAG Models (Work/Depth)

DAG = Directed Acyclic Graph
nodes = instructions
edges = dependencies

Start

Work = number of nodes = 8
Depth = length of longest path = 3

3.3.1 Nested Parallelism

series composition \hspace{1cm} parallel composition \hspace{1cm} node

\[
G = \begin{array}{c}
Ga \\
Gb
\end{array} \quad \text{or} \quad \begin{array}{c}
Ga \\
Gb
\end{array}
\]

3.3.2 Mergesort

Assume we have a parallel merge function that has \( W = O(n) \) and \( D = O(\log^2 n) \)
mergesort:

\[ W(n) = 2 \times W \left( \frac{n}{2} \right) + O(n) = O(n \log n) \]
\[ D(n) = D \left( \frac{n}{2} \right) + O(\log^2 n) = O(\log^3 n) \]

### 3.3.3 Mapping from DAGs to PRAM

**Definitions:**

- A node is ready when all its parents have completed. All ready jobs can be run in parallel.
- In a greedy schedule, if there are \( p \) or less nodes ready where \( p \) is the number of available processors, then assign all ready nodes to the processors, otherwise assign any \( p \) ready nodes to processors.