Comparison Networks

\[ x \rightarrow \min(x, y) \]
\[ y \rightarrow \max(x, y) \]

Comparator

Notation

Sorting Network [developed in 50s]

\[
\begin{array}{cccc}
10 & 5 & 2 & 2 \\
5 & 10 & 6 & 5 \\
2 & 5 & 6 & 5 \\
6 & 6 & 10 & 7 \\
\end{array}
\]

Sorted outputs:

\[ \{4, 5, 6\} \]

\[ \text{Why does it sort?} \]

Running time = depth = longest path of comparators (= 3)

Odd-Even Transposition Sort

Depth = \( N \)

(How low can you go?)
step 3 - Sorting network = mergesort [Batcher]

```
Depth \( D(N) = D(N/2) + \lg N \)
= \( \Theta(\lg^2 N) \)

Size \( S(N) = 2S(N/2) + \Theta(N\lg n) \)
= \( \Theta(N\lg^2 N) \)
```

Example:
Bitonic Sorting Network (Batcher)

**Step 1:** Sort "bitonic" sequence

**Def:** A bitonic sequence:

- or cyclic rotation

⇒ 0-1 bitonic sequence:

Key subnetwork: half cleaner

Claims: half output is clean (â€œbitonicâ€)

**Proof:**

<table>
<thead>
<tr>
<th>top</th>
<th>bottom</th>
<th>⇒</th>
<th>top</th>
<th>bottom</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<<other case>>
Sort bitonic sequence

Depth: \( D(N) = D(N/2) + 1 \)
\[ = \log N \]

Size: \( S(N) = 2S(N/2) + N/2 \)
\[ = \Theta(N \log N) \]

Step 2: Construct merging network, one sorted up, other down.
Frequently drawn where I means

Merging Circuit

effectively reversed

Batchers Circuit
Batcher's Odd-Even Mergesort

Step 4: binary merger of \( A = a_0 \ldots a_{n-1} \) and \( B = b_0 \ldots b_{n-1} \)

\[
\begin{array}{c}
a_0 \rightarrow 1 \\
a_1 \rightarrow 2 \\
a_2 \rightarrow 3 \\
a_3 \rightarrow 4 \\
b_0 \rightarrow 5 \\
b_1 \rightarrow 6 \\
b_2 \rightarrow 7 \\
b_3 \rightarrow 8 \\
\end{array}
\]

Interleave elements

Proof: 0-1 Lemma.

\[
\begin{array}{c}
T \quad 0 \quad T \quad 0 \\
1 \quad 1 \quad L \\
A \quad B
\end{array} \Rightarrow \left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor
\]

\[
\begin{array}{c}
0 \quad 0 \\
1 \quad 1 \\
A \quad B
\end{array}
\]

\Rightarrow \#0's in each list differs by 1.

\[
\begin{align*}
\text{Merge: } M(N) &= M(N/2) + 1 \\
&= \Theta(n \log n)
\end{align*}
\]

\[
\begin{align*}
\text{Sort: } S(N) &= S(N/2) + M(N) \\
&= \Theta(n \log^2 n).
\end{align*}
\]
Longstanding open question:
Does there exist sorting network with depth $O(\log n)$?
1983: yes! AKS sorting network (Ajtai, Komlós, Szemédi)

Depth: \( N \)

#Comparators: \( O(N\log N) \)

Unfortunately, very large constants: Many thousands!
Sorting on Mesh of Trees.

Def: 2-dimensional mesh of trees (MOT) $M_{2,N}$

- $N \times N$ grid → remove grid edges
- add tree above every row & column

$\# \text{Nodes}: N (2N-1) + N (N-1) = 3N^2 - 2N$

diameter: $4 \log N$

bisection width: $N$

recursive decomposition: Remove all roots

\[ \Rightarrow 4 \text{ separate } M_{2, N/2}. \]
Sort \( N^2 \) elements: \( \Omega(N) \) time (bitonic LB)
Sort \( N \) elements: \( \Theta(\lg N) \) time

## Algorithm:
1. Pass \( w_i \) along its row \& column.
2. In node \( pij \) (row \( i \), column \( j \))
   store \( \{ \), \( w_i \leq w_j \)
   \( \} \), \( w_i \geq w_j \)
3. Count \# 1's in its tree
   \( \Rightarrow \) rank of \( w_j \) in sorted order
4. If \( \text{rank}(w_j) = k \)
   \( \Rightarrow \) Send \( w_j \) to \( k^{th} \) router.

\( \Rightarrow \)

\( < \text{Waves Routing } > \)

\[ 2k + \log N \]

Note: \( \Theta(4k+\log N) \) bit steps for \( k \) bit \#s.
Send MSB first.
Sort $N^2$ elements: $\mathcal{O}(N)$ time (bisection LB)
Sort $N$ elements: $w_1, w_2, \ldots, w_N$ $\mathcal{O}(\log N)$ time

Idea: brute force. Do all comparisons.

Given $N$ $k$-bit #s, following bit-stamped sort in $2k + 5\log N$ steps.

1. For each row $i$, pass $w_i$ along $i$th column & row (MSB first) store at root of each column tree.

2. For each leaf, bitwise compare $w_i \neq w_j$.
   (Break ties with index $i,j$)
   Leaf $p_{ij}$ stores $\begin{cases} 1 & \text{if } w_i < w_j; \\ 0 & \text{if } w_i = w_j \end{cases}$

3. $H_j$, count #1's in leaves of $j$th column tree.
   $\Rightarrow$ rank of $w_j$ in sorted order.

4. If $\text{rank}(w_j) = r$, send $w_j$ to root of $r$th routine
Simulating Bipartite Graph / Ideal Computer on MOT

For large $N$, $K_{NN}$ not realistically implementable.

$\Rightarrow$ Simulate $K_{NN}$ by $M_{2N}$ with $2\log N$ delay

The catch: quadratic blowup in space/hardware