Comparison Networks

Notations

Sorting Network [developed in 50s]

10  5  2
5 10  6  5
2  5  6
2  5 10

Sorted outputs

"Why does it sort?"

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

Odd-Even Transposition Sort

Depth = N

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

\[
\text{Depth } D(n) = D(n/2) + \lg n
\]

\[
= \Theta(\lg^2 n)
\]

\[
\text{Size } S(n) = 2S(n/2) + \Theta(n\lg n)
\]

\[
= \Theta(n\lg^2 n)
\]

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

Claim: half output is clean (\& bitonic)

**Pf:**

\[
\begin{array}{c|c}
0 & 1 \\
1 & 0 \\
\end{array} \quad \Rightarrow \quad \begin{array}{c|c}
0 & 1 \\
1 & 0 \\
\end{array}
\]

\[\text{or}
\begin{array}{c|c}
0 & 1 \\
1 & 1 \\
\end{array} \quad \Rightarrow \quad \begin{array}{c|c}
0 & 1 \\
1 & 1 \\
\end{array}
\]
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.
Batcher's Odd-Even Merge Sort

Step 1: Build merger of $A = a_0 \ldots a_m$ and $B = b_0 \ldots b_{n-1}$

Interleave elements.

Proof: 0-1 Lemma.

\[
\begin{array}{c|c|c|c|c}
T & 0 & 1 & 0 & 1 \\
\hline
A & 0 & 1 & 0 & 1 \\
B & 1 & 0 & 1 & 0 \\
\end{array}
\Rightarrow \left\lfloor \frac{r}{2} \right\rfloor + \left\lceil \frac{r}{2} \right\rceil
\begin{array}{c|c|c|c|c}
T & 0 & 1 & 0 & 1 \\
\hline
A & 0 & 1 & 0 & 1 \\
B & 1 & 0 & 1 & 0 \\
\end{array}
\Rightarrow \left\lceil \frac{r}{2} \right\rceil + \left\lfloor \frac{r}{2} \right\rfloor
\]

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

Merge \( M(N) = M(N/2) + 1 \)

= \( \Theta(N \log N) \)

Sort: \( S(N/2) = S(N/2) + M(N) \)

= \( \Theta(N \log^2 N) \).
Longstanding open question:
Does there exist sorting network with depth $O(\log n)$?
1983: yes! AKS sorting network (Kjell, Komlos, Szemerédi)
Depth: $N$

#Comparisons: $O(N\log N)$

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

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

N×N grid → remove grid edges
- add tree above every row & column

# Nodes: \( N (2N-1) + N (N-1) = 3N^2 - 2N \)

diameter: \( 4 \lg N \)

bisection width: \( N \)

recursive decomposition: remove all roots
\( \Rightarrow 4 \) separate \( M_{2,\frac{N}{2}} \).
Sort \( N^2 \) elmts: \( \Theta(N^2) \) time (bisection \( CB \))

Sort \( N \) elmts: \( \Theta(\log N) \) time

\( \text{W} \ldots \text{W}N \)

(1) pass \( W \) along 1st row & 1st column

(2) In node \( p_{ij} \)
    (row \( i \), column \( j \))
    store \( \{ \), \( W_i \leq W_j \)
    \( \{ \), \( W_i > W_j \)

(3) count #1's in jth tree
    \( \Rightarrow \) rank of \( W_j \) in sorted order

(4) if \( \text{rank}(W_j) = k \)
    \( \Rightarrow \) send \( W_j \) to \( j \)'s routine.

\( \Rightarrow \)

2k + 5 \log N

Note: \( \Theta(k + \log N) \) bit steps for \( k \) bit #s.

Send MSB first.
Sort $N^2$ elements: $\Theta(N)$ time (binary search tree)
Sort $N$ elements: $w_1, w_2, \ldots, w_N$ $\Theta(\log N)$ time

Idea: brute force. Do all comparisons.

Given $N$ $k$-bit #s, following bit-stably sorts in $2k + 5 \log N$ steps:

1. Pass $w_i$ along $i$: its column & row
   (MSB first) Store at root of each column tree.

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

3. $w_j$ count #1's in leaves of $j$'s column tree
   \[ \Rightarrow \text{rank of } w_j \text{'s sorted order.} \]

4. If rank($w_j$) = $r$, send $w_j$ to root of $r$'s row tree.
Simulating Bipartite Graph/Ideal Computer on MOT

For large $N$, Knn not realistically implementable.

\[ \rightarrow \text{Simulate Knn by M2N with } 2\log N \text{ delay} \]

The catch: quadratic blowup in space/hardware.