Sorting on 1 and 2D Arrays

Linear array: odd-even transposition sort

\[ X_1 \rightarrow X_2 \rightarrow X_3 \rightarrow X_4 \rightarrow X_5 \rightarrow X_6 \rightarrow X_7 \rightarrow X_8 \]

Odd-even transposition sorts in \( 2N \) steps (with \( N \) odd).

**Def:** Oblivious comparison-exchange alg. Comparisons prespecified. Independent of results of prev comparisons. *(e.g., quicksort *not* oblivious)*

**Thm:** If an oblivious comparison-exchange alg sorts all \( 2^N \) sequences of 0's and 1's, it sorts all sequences of arbitrary #s.
Proof: In 2 parts:

(1) Let $f$ be monotonically increasing function. Then
\[
\begin{align*}
\min \{ f(x), f(y) \} & = f(\min \{ x, y \}) \\
\max \{ f(x), f(y) \} & = f(\max \{ x, y \}).
\end{align*}
\]

By induction on timesteps, if alg transforms
\[
\langle a_1, x_2, \ldots, a_n \rangle \to \langle b_1, x_2, \ldots, b_n \rangle,
\]
then it transforms
\[
\langle f(a_1), f(a_2), \ldots, f(a_n) \rangle \to \langle f(b_1), f(b_2), \ldots, f(b_n) \rangle.
\]

<< See CLR >>

(2) Suppose false, i.e., network sorts all 0-1 seq., but \( \exists \langle a_1, a_2, \ldots, a_n \rangle \) s.t. \( a_i < a_j \), but \( a_i \) comes after \( a_j \) in output.

Define \( f(x) = \begin{cases} 0, & \text{if } x \leq a_i \\ 1, & \text{if } x > a_i \end{cases} \).

But network fails to sort \( \langle f(a_1), f(a_2), \ldots, f(a_n) \rangle \)

Contradiction.

<< Threshold induction >>

\( \Rightarrow \) Need only construct 0-1 sorting algo!
There: Odd-even transposition sort runs in \( N \) steps (with \( \frac{1}{2} \) of \( \text{OPT} \)).

Result less interesting than proof method.

Proof: Consider movement of rightmost 1.

1st step: may not move 1.

During subsequent steps, moves forward.

\( \Rightarrow \) cannot block other 1's.

\( \Rightarrow \) kth leftmost 1 begins moving 1 step: \( k+1 \).

Must reach position \( N-k+1 \).

\( \Rightarrow \) All elements in final position by time \( N \).
Sorting an 2D Grid

Lower bounds: \[ 2 \sqrt{N} - 2 \quad \text{(diameter)} \]
\[ \frac{1}{2} \sqrt{N} \quad \text{(bisection)} \]

Natural Grid Orders:

```
\[ \Longrightarrow \quad \downarrow \downarrow \downarrow \quad \leftarrow \leftarrow \]
```

"Broken" Alg:

Repeat:
1. \[ \downarrow \downarrow \downarrow \]
2. \[ \leftarrow \leftarrow \]

Doesn't yield unique order:

\[ \begin{align*}
0 \uparrow 10 & \Rightarrow \begin{cases}
01 \quad \downarrow 1 \Rightarrow \\
01 \Rightarrow \downarrow \downarrow
\end{cases}
\end{align*} \]
**Shearsort**

Repeat

\[
\begin{align*}
\text{Thm: Shearsort produces unique sorting order after time } O(JN \log N). \text{ Eg., } O(\log N) \text{ phases sufficient to sort.}
\end{align*}
\]

**Pf:** Apply 0-1 lemma.

Def: 00 - 00 \(\Rightarrow\) "clean" lines

\[
\text{or}
\]

0 - 0 \(\text{or} - 1 \Rightarrow\) "dirty" lines

Claim: After each phase, # dirty rows decreases by at least half.
Grid has 3 regions:

divide dirty into pairs of rows:

either:

\[
\begin{array}{c}
0 \\
1 \\
0 \\
1 \\
0 \\
1 \\
0
\end{array}
\]

or:

After sorting columns:

either:

\[
\begin{array}{c}
1 \\
0 \\
1 \\
0
\end{array}
\]

or:

\[
\begin{array}{c}
0 \\
1 \\
0 \\
1 \\
0
\end{array}
\]

⇒ dirty region decreases by \( \geq \frac{1}{2} \).

⇒ after \( \log N \) phases [\( \Theta(\sqrt{N} \log N) \) time] all sorted.
Lemma: Shearsort runs in $\mathcal{O}(\lg N)$ phases.

Proof: Bad example: 0's in 1st column.

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

\[
\downarrow
\]

\[
\begin{array}{c}
\begin{array}{c}
0
\end{array}
\end{array}
\Rightarrow
\begin{array}{c}
\begin{array}{c}
0
\end{array}
\end{array}
\]

height of 1$^\text{st}$ column decreases by factor of 2 in each round.

Average Case:

Substitute 0's for $\sqrt{N}$ smallest elms.

- Row sort first: $E[\# 0's \text{ in } 1\text{st column}] = \Theta(N)$.
- Column sort first: not true.
  best LB $= \mathcal{O}(N\sqrt{\lg N})$. 

$O(\sqrt{N})$ Algorithm \ $(\leq 8\sqrt{N})$

Assume $\sqrt{N}$ is power of 2

1. Recursively sort each quadrant in snake order.

2. Sort rows in alternate order.

3. Sort columns.

4. Do $2\sqrt{N}$ steps of 1D odd-even transposition on overall snake order.
Running time:

\[ T(N) = T(N/4) + \sqrt{N} + \sqrt{N} + 2\sqrt{N} \]
\[ \leq 8\sqrt{N} \]

Proof of Correctness:

**Phase 1**: In each quadrant at most one of rows is dirty and rest are clean.

**Phase 2**
**Phase 3**

- **Claim:** ≤ 1 dirty row in top half & bottom half.
  - (**gilding the lily, we already have a O(Tn) alg**)

**Proof:**

**Case 1:** # so-so rows even.

**Case 2:** # so-so rows odd.

=> Sorted after phase 4