Sorting on 1 and 2D Arrays

Linear array: odd-even transposition sort

\[
\begin{align*}
X_1 & \rightarrow X_2 \\
& \downarrow \\
X_3 & \rightarrow X_4 \\
& \downarrow \\
& \vdots \\
X_6 & \rightarrow X_7 \\
& \downarrow \\
X_8 & \\
\end{align*}
\]

Def: Oblivious comparison-exchange alg. Comparisons prespecified, independent of results of prev comparisons. (eg, 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 \[
\min\{f(x), f(y)\} = f(\min\{x, y\}),
\]
\[
\max\{f(x), f(y)\} = f(\max\{x, y\}).
\]

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

<< See CLR >>

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

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 >>

\[ \Rightarrow \text{Need only construct 0-1 sorting algo!} \]
Thm: Odd-even transposition sort runs in $N$ steps (with 1 of OPT).
   << Result less interesting than proof method >>

Pf: Consider movement of rightmost 1.

1st step: may not move 2.

During subsequent steps, moves forward.
   => cannot block other 1's.

=> k'th leftmost 1 begins moving by step $k+1$.
   Must reach position $N-k+1$.

=> All elmts in final position by time $N$.
Organizing a 2D Grid

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

Natural Grid Orders:

![Diagram of natural grid orders]

"Broken" Alg:

Repeat:

1. \[ \text{ } \]
2. \[ \text{ } \]

Doesn't yield unique orders:

\[
0 \rightarrow \begin{cases} 
0 & \rightarrow \\
1 & \rightarrow \\
1 & \rightarrow 
\end{cases}
\]
**Shearsort**

**Repeat**

- 
- 
- 
- 
- 

**Thm:** Shearsort produces unique sorting order after time $O(INlgN)$. E.g., $O(lgN)$ phases sufficient to sort.

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

**Def:** 00 or 003 "clean" lines

41 or 11

00 01 1 3 "dirty" lines

10 or 0 0

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

```
  0
  dirty
  1
```

Divide dirty into pairs of rows:

Either:

```
  0  1
  1  0
```

Or:

```
  0  1
  1  0
```

After sorting columns:

Either:

```
  0  1
  1  1
```

Or:

```
  0  1
  1  0
```

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

⇒ after \( \lceil \log N \rceil \) phases \( \Theta (N \log N) \) time, all sorted.
Lemma: Shearsort runs in $\mathcal{O}(\log N)$ phases.

Proof:

Bad example: 0's in 1st column.

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

height of 1st column decreases by factor of 2 in each round.

Average Case:

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

- rowsort first: $E[\# 0's \ in \ 1st \ column] = \Theta(N)$.
- columnsort first: not true.
  best LB = $\mathcal{O}(N\log N)$. 

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

Proof of Correctness:

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

---

**Phase 2:**

---
**Phase 3**

Claim: 

| Dirty row in top half &
| " " " " " bottom half. |

<<gilding the lily, we already have a $\Theta(n)$ alg >>

**Proof:**

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

<table>
<thead>
<tr>
<th>Majority</th>
<th>So-So</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 0 0 0 0</td>
<td>1 1</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Majority</th>
<th>So-So</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 1 0 0</td>
<td>1 1</td>
</tr>
</tbody>
</table>

\[ \Rightarrow \text{dirty} \]

\[ \Rightarrow \text{2 dirty rows.} \]

\[ \Rightarrow \text{Sorted after phase 4} \]