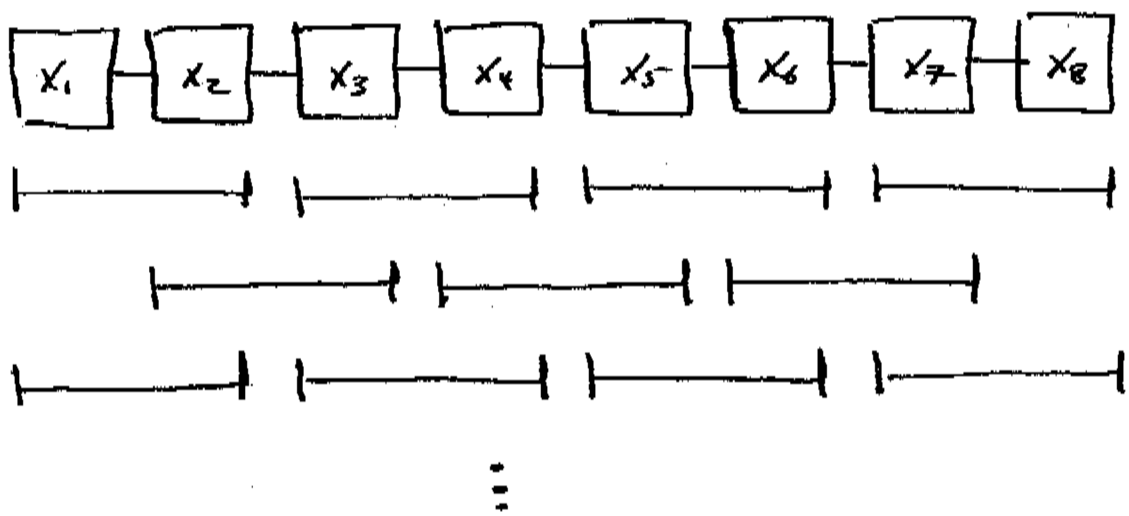
Mon 3/15/04
Michael Bender Lecturing

Sorting on 1 and 2D Arrays

Linear array: odd-even transposition sort.



~~Thm: odd-even trans sort runs in N steps (with 1 of OPT).~~

<u>Def:</u> Oblivious comparison-exchange alg. Comparisons prespecified. Independent of results of prev comparisons.

<<e.g. quicksort <u>Not</u> oblivious>>

<u>Thm:</u> If an obliv comparison-exchange alg sorts all 2^N sequences of 0's and 1's, it sorts all sequences of arbitrary #s

3/15/04

<u>Proof:</u> 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 timesteps, if alg transforms
$\langle a_1, x_2, \ldots, a_N \rangle \to \langle b_1, b_2, \ldots, b_N \rangle$,

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. Ie, network sorts all 0-1 seq, but $\exists \langle a_1, a_2, \ldots, a_N \rangle$ st $a_i < a_j$, but a_i comes after a_j in output.

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

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

Contradiction. ∎

<< threshold intuition >>

$\Rightarrow$ Need only construct 0-1 sorting algs!

3/15/04

<u>Thm:</u> Odd-even transposition sort runs in N steps (with 1 of OPT).

<< Result less interesting than proof method >>

<u>Pf:</u> Consider movement of rightmost 1.

1^{st} step: may not move $\underline{0\ 1}$

During subsequent steps, moves forward.

$\Rightarrow$ cannot block other 1s.

$\Rightarrow$ k^{th} leftmost 1 begins moving by step $k+1$. Must reach position $N-k+1$.

$\Rightarrow$ All elmts in final position by time N

3/15/04

Sorting on 2D Grid

Lower bounds: $2\sqrt{N}-2$ (diameter)
$\Omega(\sqrt{N})$ (bisection)

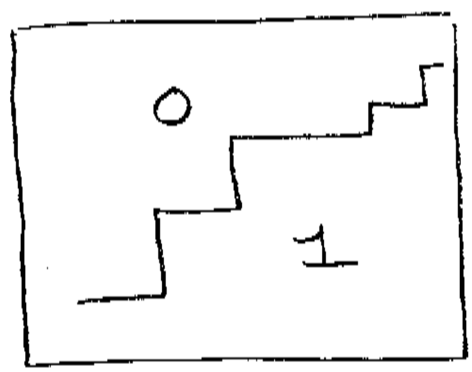
Natural Grid Orders:

"Broken" Alg:

Repeat:

(1) $\downarrow\downarrow\downarrow\downarrow$

(2) $\rightrightarrows$



Doesn't yield unique order:

$$\begin{matrix}0 & 1\\ 1 & 0\end{matrix} \Rightarrow \begin{cases} \begin{matrix}0 & 0\\ 1 & 1\end{matrix} & \downarrow\downarrow \ \rightrightarrows \\[1em] \begin{matrix}0 & 1\\ 0 & 1\end{matrix} & \rightrightarrows \ \downarrow\downarrow . \end{cases}$$

3/15/04

Shearsort

Repeat

- ↓↓↓↓
- →
←
→
←

Thm: Shearsort produces unique sorting order after time $O(\sqrt{N}\lg N)$. Eg, $O(\lg N)$ phases sufficient to sort.

Pf: Apply 0-1 lemma.

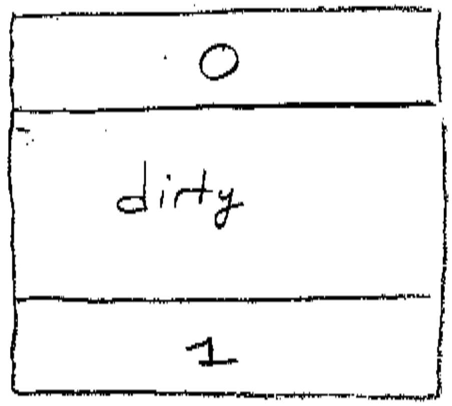
Def:

00 ———— 00
or
11 ———— 11 } "clean" lines

0 —— 01 —— 1
or
1 — 10 ——— 0 } "dirty" lines

Claim: After each phase, # dirty rows decreases by at least half.

Grid has 3 regions:



Divide dirty into pairs of rows:

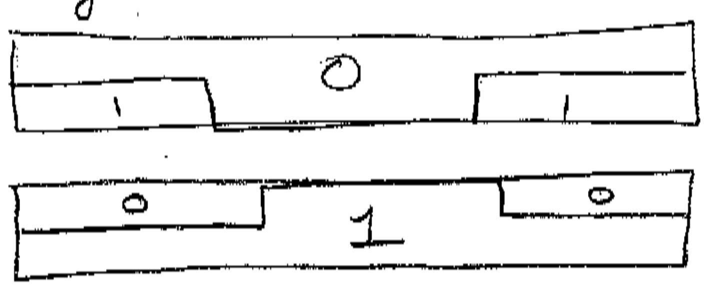
either:

or:

After sorting columns:

either

or

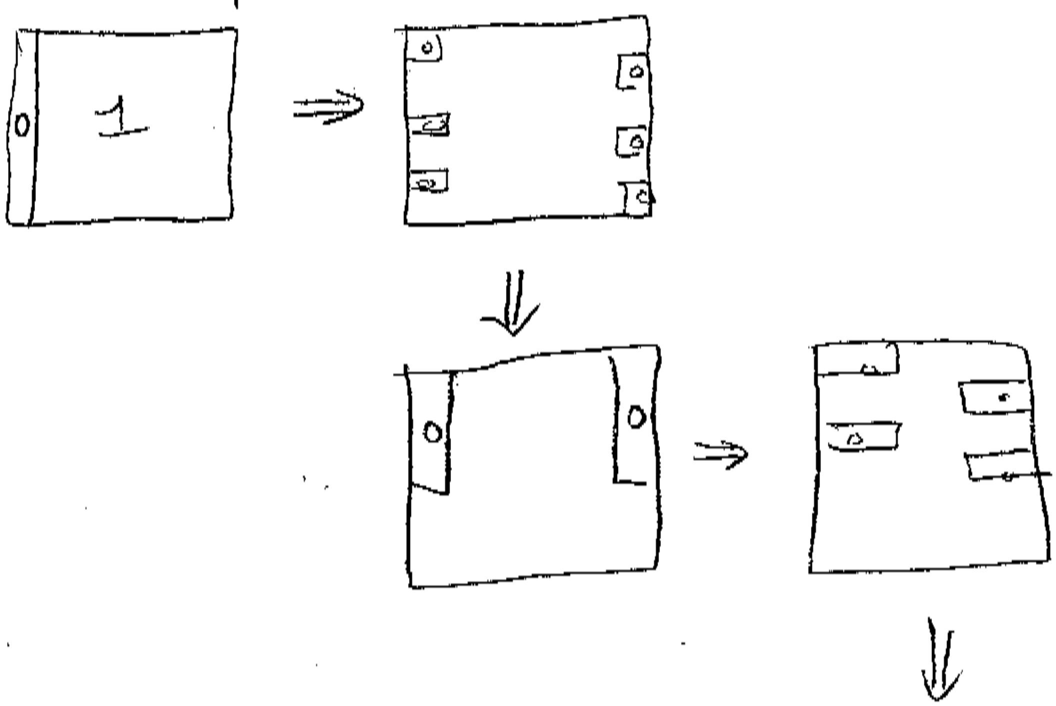


$\Rightarrow$ dirty region decreases by $\geq 1/2$.

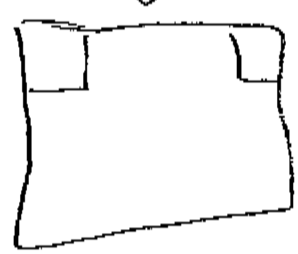
$\Rightarrow$ after $\lg N$ phases [$\Theta(\sqrt{N} \lg N)$ time]
all sorted.

<u>Lemma:</u> Shearsort runs in $\Omega(\lg N)$ phases.

<u>Pf:</u> Bad example 0's in 1st column.



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



<u>Average Case:</u>

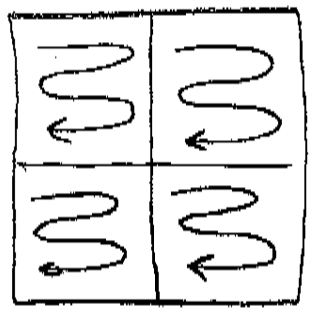
Substitute 0's for $\sqrt{N}$ smallest elmts

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

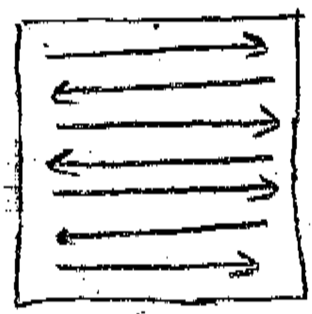
$O(\sqrt{N})$ Algorithm ($\le 8\sqrt{N}$)

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

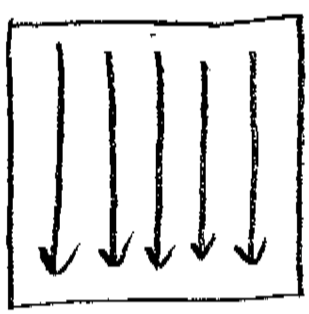
(1) Recursively sort each quadrant in snake order



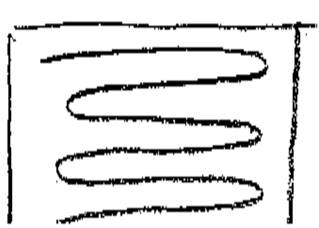
(2) Sort rows in alternate order



(3) Sort columns



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

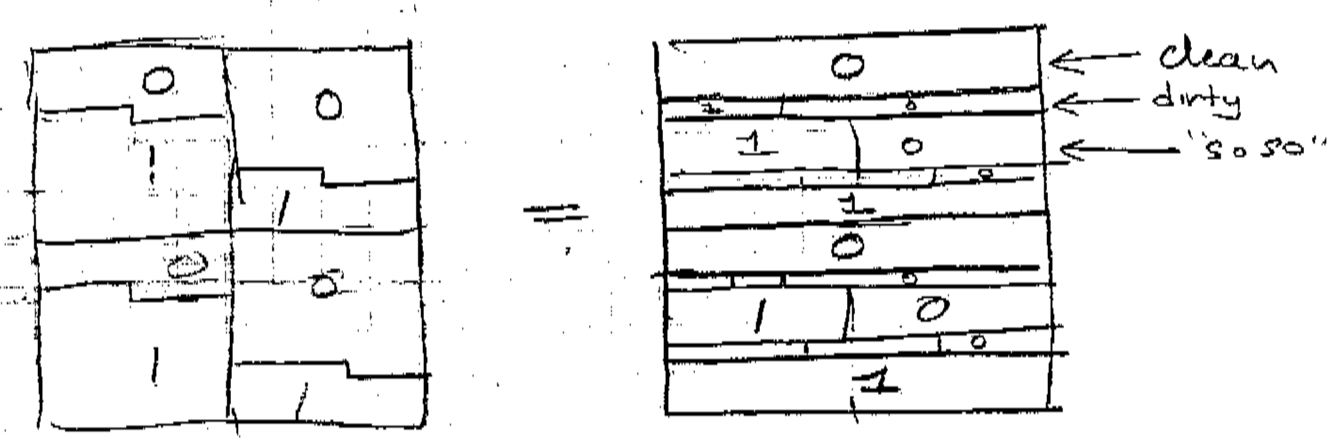


Running time:

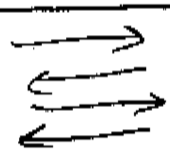
$$T(N) \leq T(N/4) + \sqrt{N} + \sqrt{N} + 2\sqrt{N}$$
$$\leq 8\sqrt{N}$$

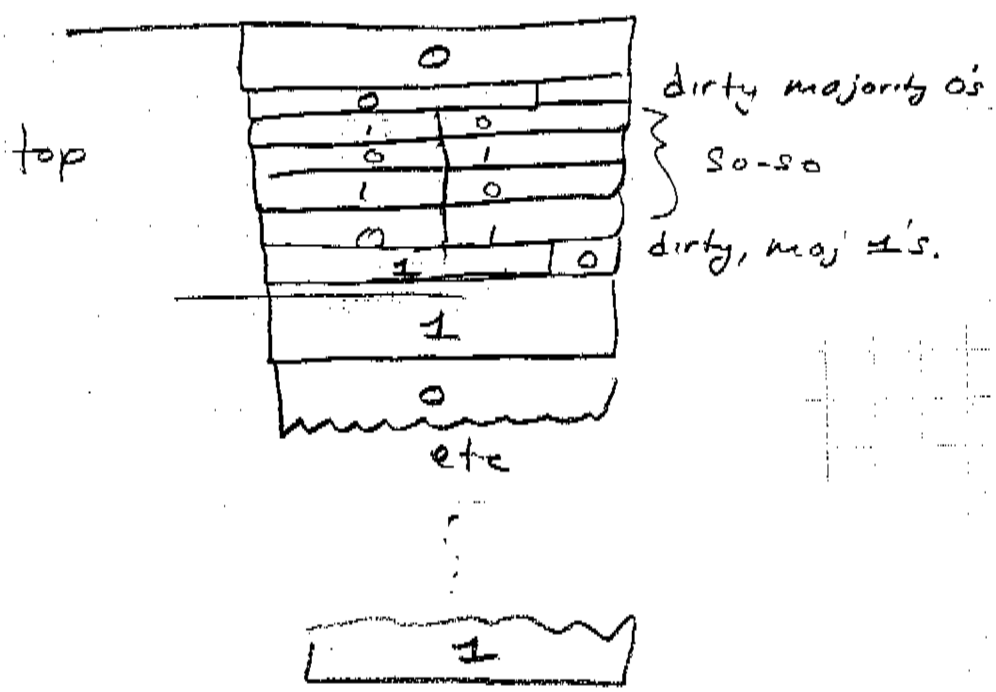
Proof of Correctness:

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



phase 2



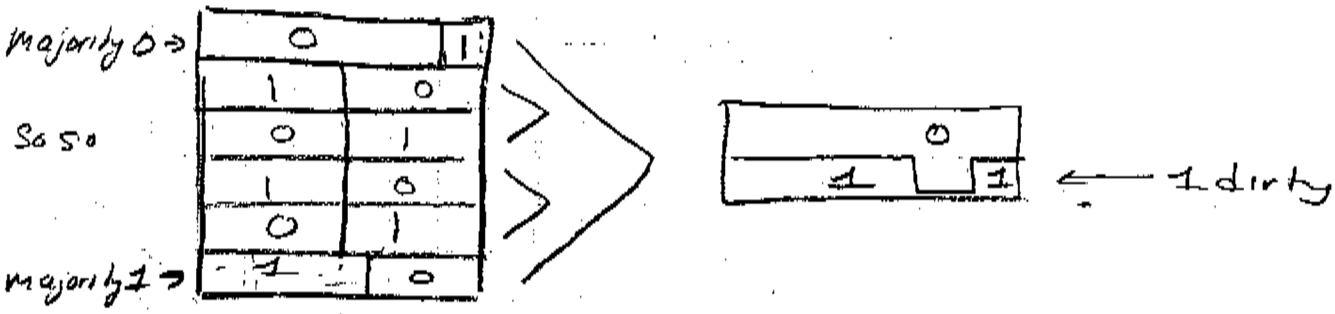


phase 3 ↓↓↓↓

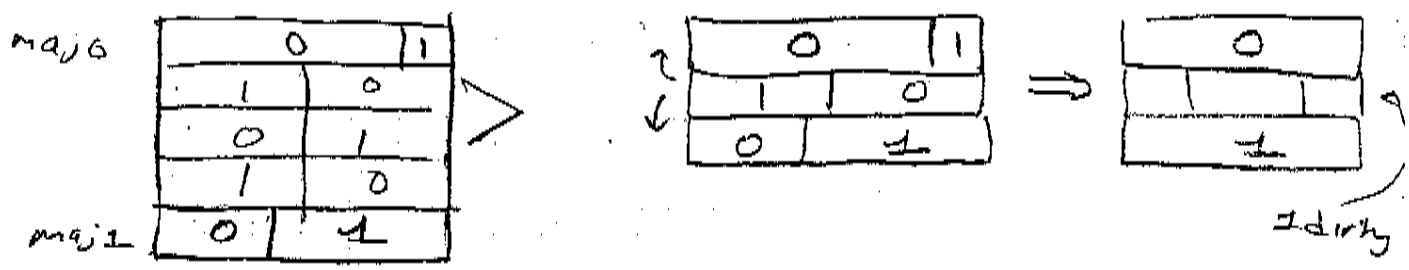
<u>clm:</u> ≤ 1 dirty row in top half &
" " " " bottom half.

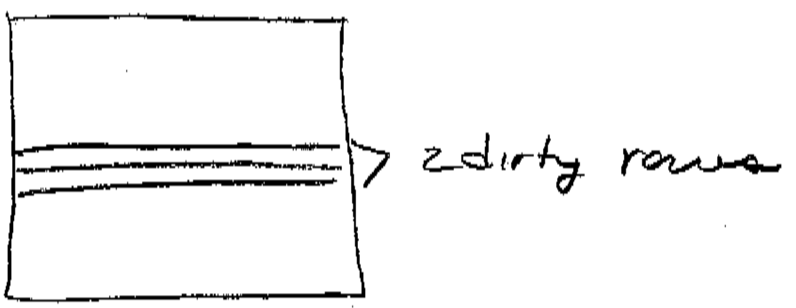
<< gilding the lily - we already have a $\Theta(\sqrt{n})$ alg >>

<u>Pf:</u> <u>Case 1:</u> # so-so rows even:



<u>Case 2:</u> # so-so rows odd:





⟹ Sorted after phase 4