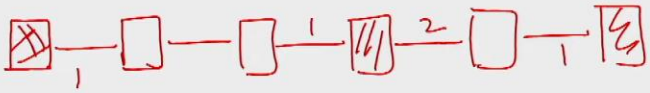
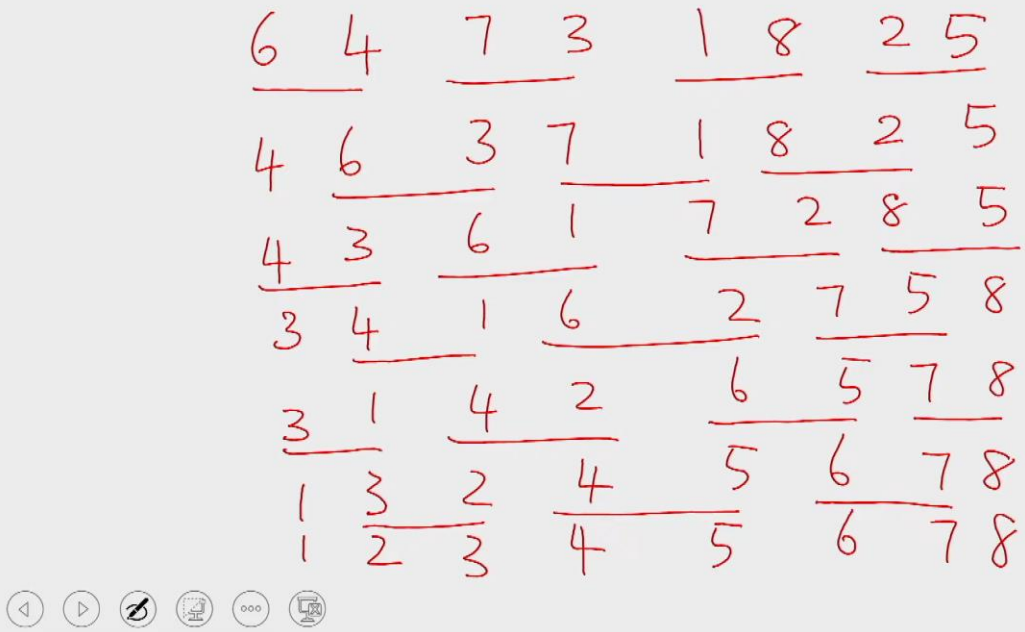
Interconnection Network Algorithms

Linear Array





How many steps are needed to sort N items on a LA of N processors?

0-1 Principle

If an algorithm A sorts correctly every 0-1 seq. then it sorts correctly any seq. drawn from a Linearly Ordered set

0-1 Sequences

$0^{\ell}\,1^{N-\ell}$

0 1 | 1 1 | 1 0 | 0 0

0 1 | 1 1 | 0 1 | 0 0

0 1 | 1 0 | 1 0 | 1 0

0 1 | 0 1 | 0 1 | 0 1

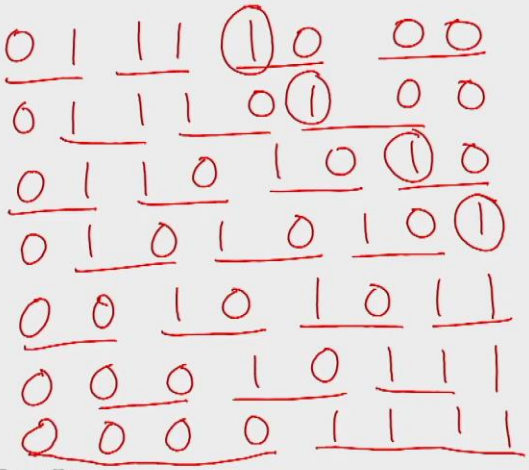
0 0 | 1 0 | 1 0 | 1 1

0 0 0 | 1 0 | 1 1 | 1

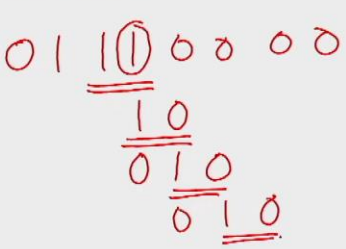
0 0 0 0 | 1 1 | 1 1

Rightmost 1 in the sequence once it starts moving, will keep moving.

0-1 Sequences



$0^{\ell}\,1^{N-\ell}$

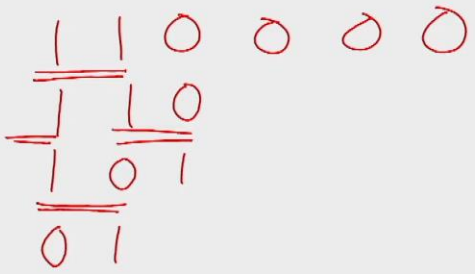


Right most 1 in the sequence
once it starts moving,
will keep moving.
Will start moving
either in the 1st step
or in the 2nd step

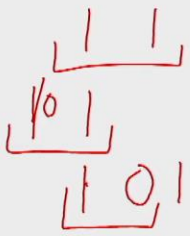
Destination : N
worst possible source : 1
RM 1 has to travel
a dist. of at most N-1

By, step N it reaches the destination

2nd RM 1



2nd RM 1 could blocked by RM 1 in step 1



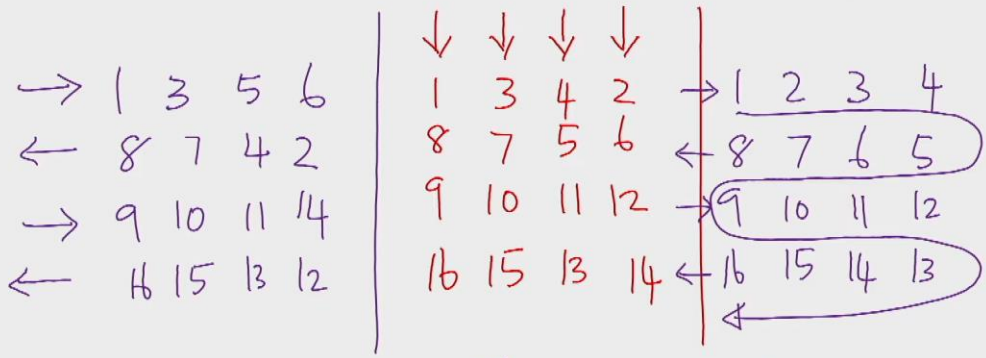
Dest. of 2nd RM 1
is $N-1$
Worst case source is 1
dist. $\leq N-2$
starts on or before step 3.
2nd RM 1 reaches in N
or earlier

i^{th} RM 1
start moving by the
$(i+1)$-st step.
dest: $N-i+1$
reaches by the N^{th} step.
every 1 reaches the dest.
by the N^{th} step

OET - sort runs in at most N steps

2D-mesh

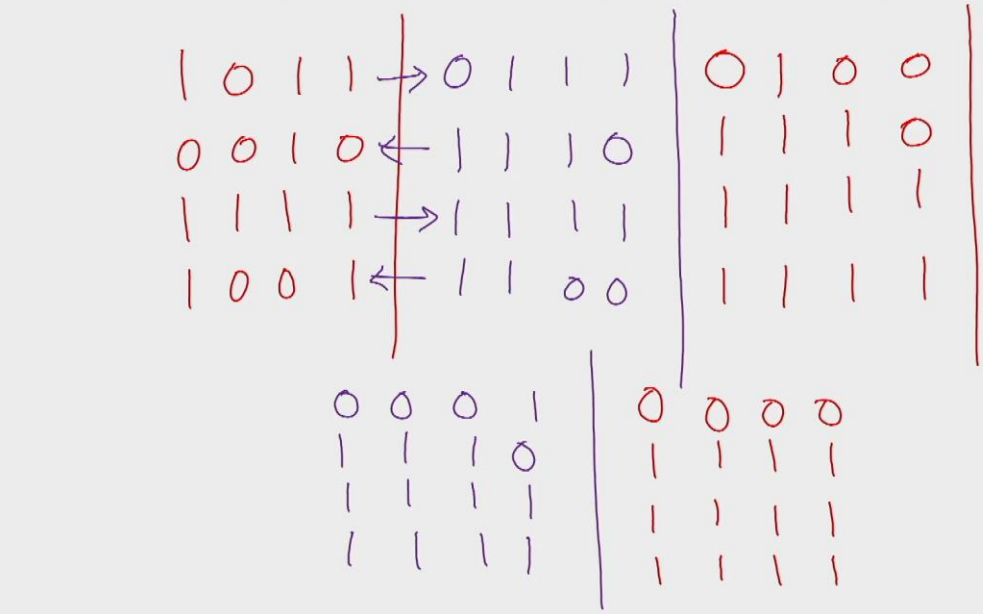
3	7	11	5	→	3	5	7	11	3	5	6	1
10	12	15	2	←	15	12	10	2	4	8	7	2
4	16	9	13	→	4	9	13	16	14	9	10	11
8	1	14	6	←	14	8	6	1	15	12	13	16



Shear Sort

0-1 Principle

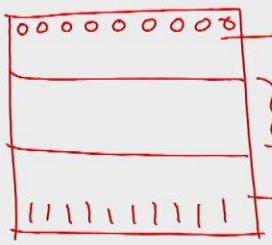
Shear sort works correctly on any binary sequence.



$\sqrt{N} \times \sqrt{N}$ mesh on a 0-1 sequence

initially

- 0 { clean — All 0's rows
- dirty — } rows containing 0's & 1's
- 0 { clean — All 1 rows

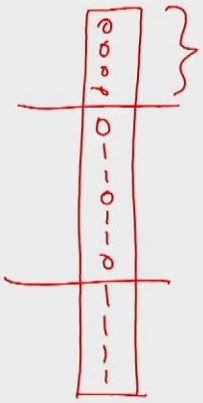


The initial size of the dirty band is $\leq \sqrt{N}$

1

The i^{th} iteration

1. Sort the rows
   odd : L to R
   even : R to L.
2. Sort the columns



Thought exercise

The cols are not sorted using OETS but "a different algo"

The "diff. algo."

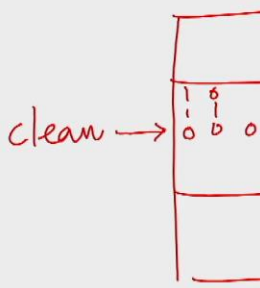
pair off the dirty rows

odd
even

```
0 0 0 1      0 1 1 1      0 0 1 1
1 1 0 0      1 1 0 0      1 1 0 0
-------      -------      -------
0 0 0 0 ✓    0 1 0 0      0 0 0 0 ✓
1 1 0 1      1 1 1 1 ✓    1 1 1 1 ✓
```

every pair gives us at least one clean row.

move a clean row of 0's all the way to the top clean band



. the columns continue to be sorted

dirty rows reduces by a factor of 2

if we had x dirty rows before, we have $\lceil x/2 \rceil$ dirty rows now

we start with $\sqrt{N}$ dirty rows.

$\lceil \log \sqrt{N} \rceil$ iterations to reduce the dirty rows to 1

each iteration takes $2N$ steps

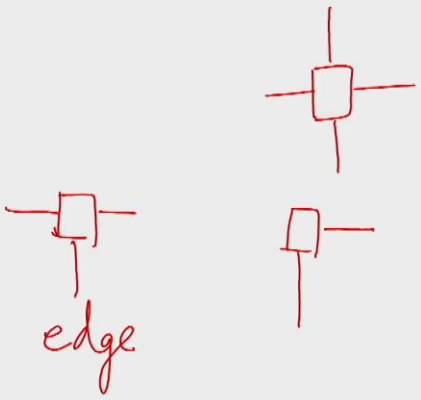
$$2\sqrt{N}\left(\lceil \log \sqrt{N} \rceil\right) + \sqrt{N}$$

$$\leq \sqrt{N}(\log N + 1) \text{ steps}$$

to sort N elements

$N^{3/2}(\log N + 1)$ cost

2D-mesh



interior processor

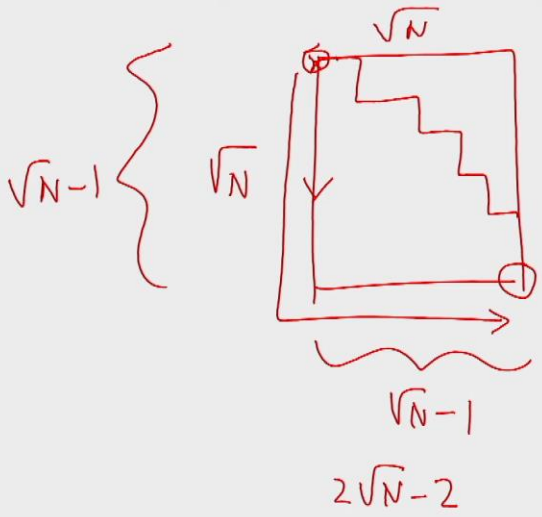
degree is

2, 3, 4

2D-mesh algorithms are far slower than PRAM algorithms

Lower Bound for sorting on 2D-meshes

$2\sqrt{N} - 2$ is a trivial lower bound.



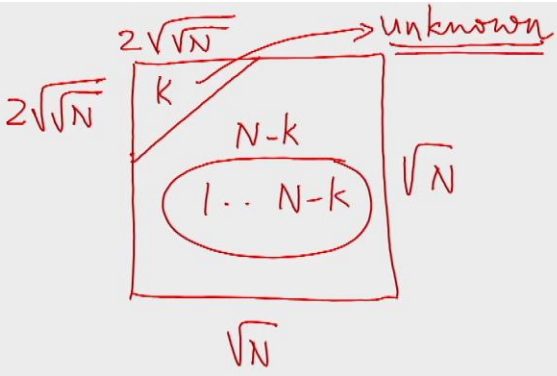
mesh

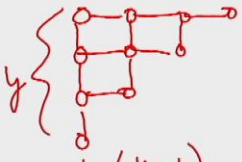
$\Omega(\sqrt{N})$

7

Tighter Lower Bound

$3\sqrt{N} + o(\sqrt{N})$ steps at least





$\frac{y(y+1)}{2}$

$y = 2\sqrt{\sqrt{N}} = 2N^{1/4}$

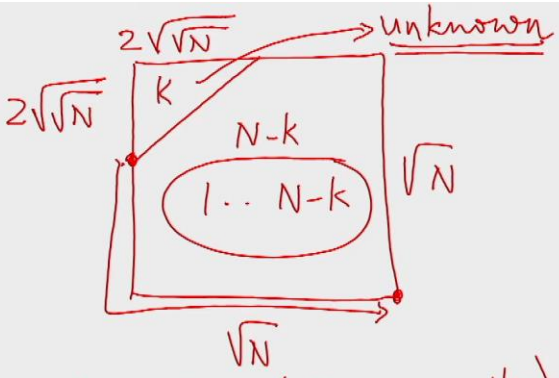
$k = \frac{2 \cdot N^{1/4}\left(2\sqrt{\sqrt{N}}+1\right)}{2}$

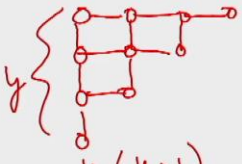
$\approx 2\sqrt{N}$

A: your algorithm

run A on this input

Stop after $2\sqrt{N} - 2N^{1/4} - 3$ steps





$\frac{y(y+1)}{2}$

$y = 2\sqrt{\sqrt{N}} = 2N^{1/4}$

$(\sqrt{N}-1) + (\sqrt{N}-1-2N^{1/4})$

$= 2\sqrt{N} - 2N^{1/4} - 2$

$k = \frac{2 \cdot N^{1/4}(2\sqrt{\sqrt{N}}+1)}{2}$

$\approx 2\sqrt{N}$

A: your algorithm

run A on this input

Stop after $2\sqrt{N} - 2N^{1/4} - 3$ steps

x is independent of the unknown elements

Replace the unknown elements with m 0's & $k-m$ N's

$c(m, x)$ as the column no. where x ends up when the shaded portion has exactly m 0's.

vary m, $\frac{2\sqrt{N} - 2N^{1/4} - 3}{\sqrt{N} - 1}$ +

$c(m, x)$ will vary

$c(m, x)$ will attain every value in $1..\sqrt{N}$

m' is the value twice that causes $c(m', x)$ to 1 $3\sqrt{N} - O(\sqrt{N})$