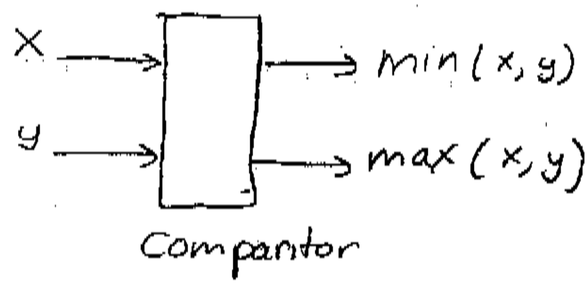
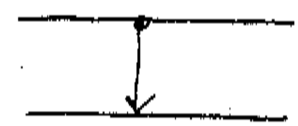
Comparison Networks

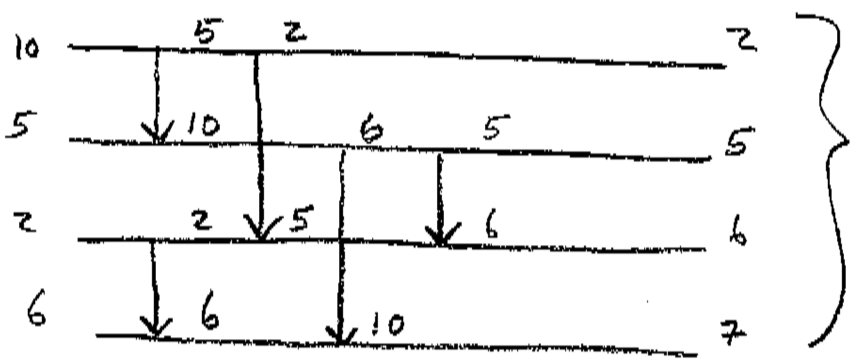
Wed 3/17
Michael Bender Lecturing



Notation:



Sorting Network [developed in 50s]

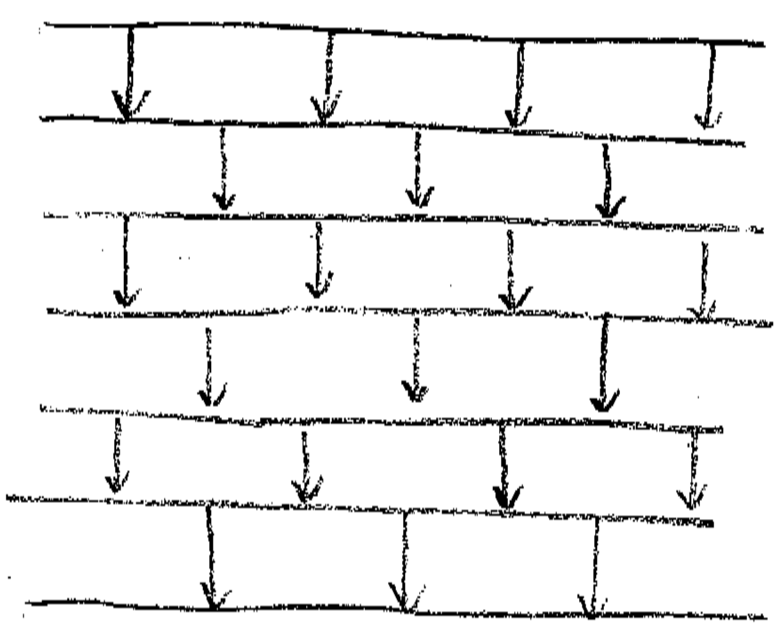


Sorted outputs

<<Why does it sort?>>

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

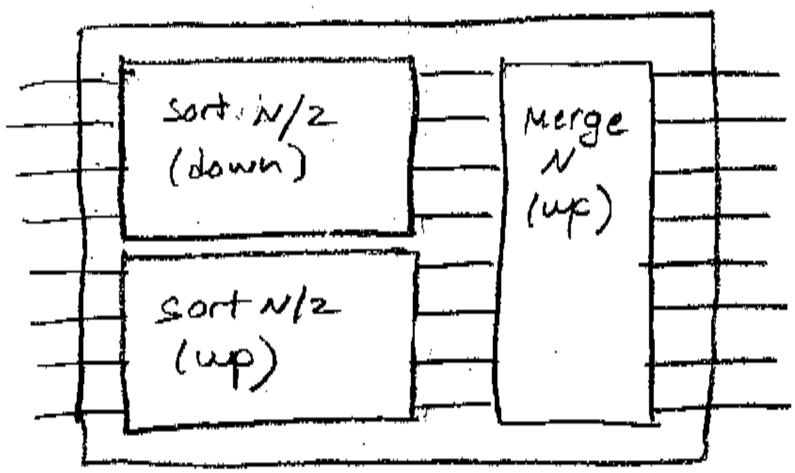
Odd-Even Transposition Sort



Depth = N

<<how low can you go?>>

step3 — Sorting network = mergesort [Batcher]



merge

Depth $D(N) = D(N/2) + \lg N$

$= \Theta(\lg^2 N)$

Size $S(N) = 2S(N/2) + \Theta(N \lg N)$

$= \Theta(N \lg^2 N)$

Example:

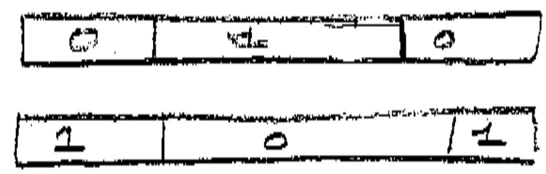
Bitonic Sorting Network (Batcher)

Step 1 Sort "bitonic" sequence

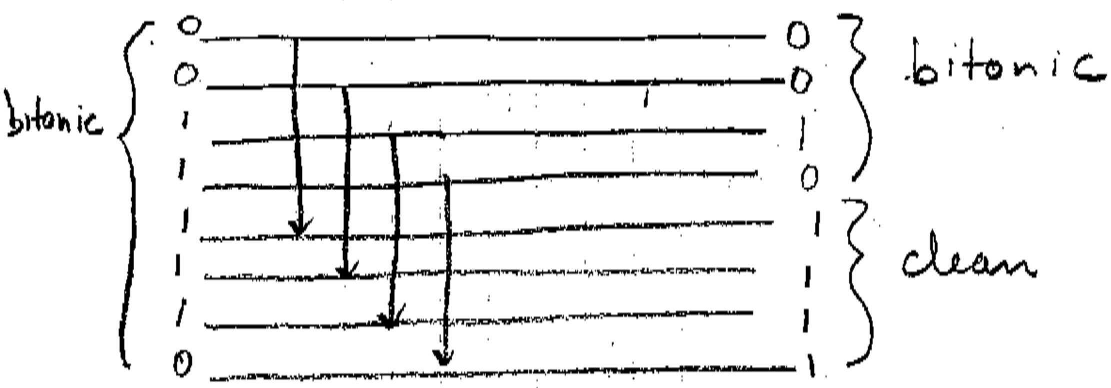
def: A bitonic sequence:

or cyclic rotation

⇒ 0-1 bitonic sequence:



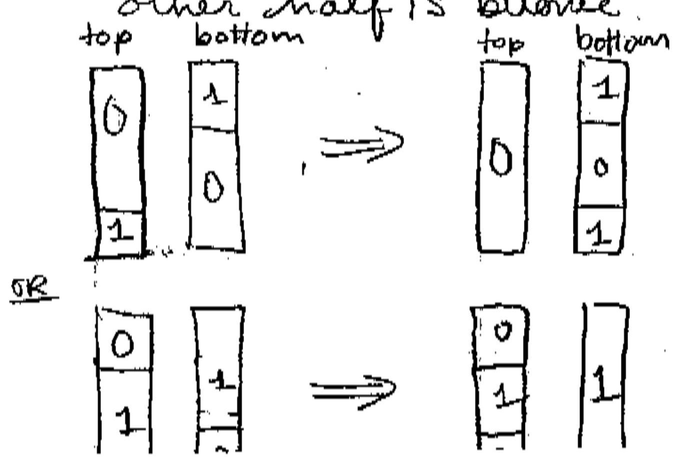
key subnetwork: half cleaner



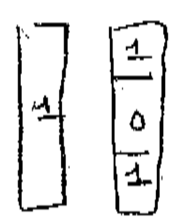
claim: half output is clean (& bitonic)

other half is bitonic

Pf:

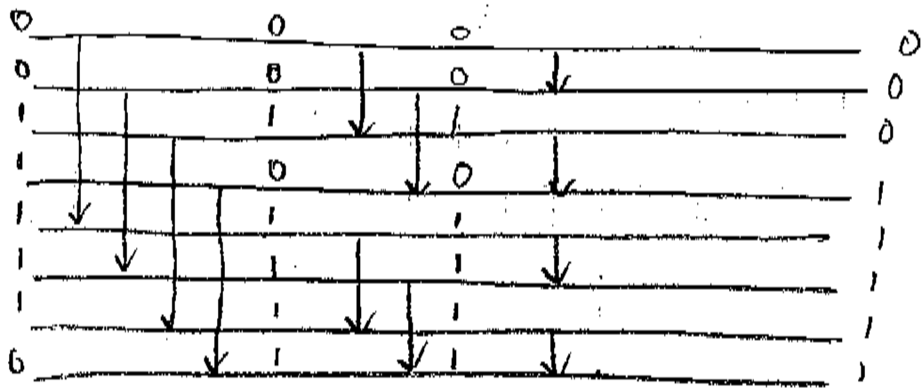


<<other case:



>>

Sort bitonic sequence



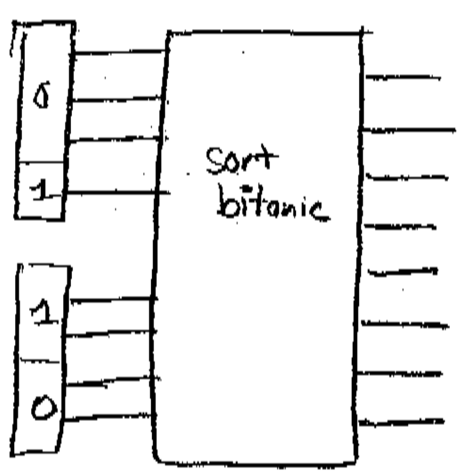
Depth: $D(N) = D(N/2) + 1$

$= \lg N$

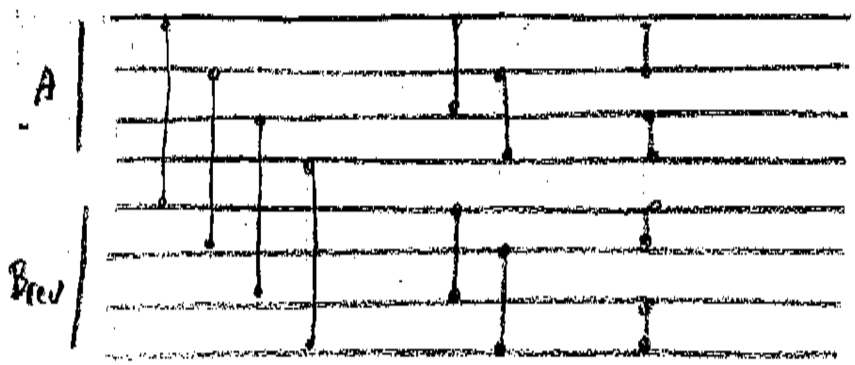
Size: $S(N) = 2S(N/2) + N/2$

$= \Theta(N \lg N)$.

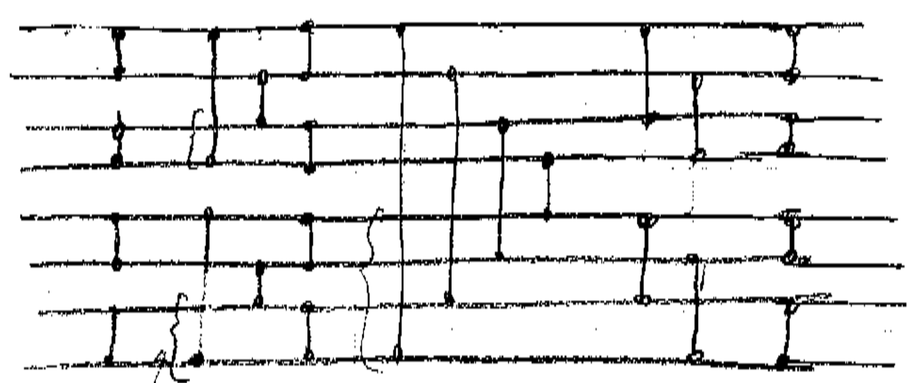
Step 2: Construct merging network, one sorted up, other down.



Frequently drawn where ↨ means ↧



Merging Circuit



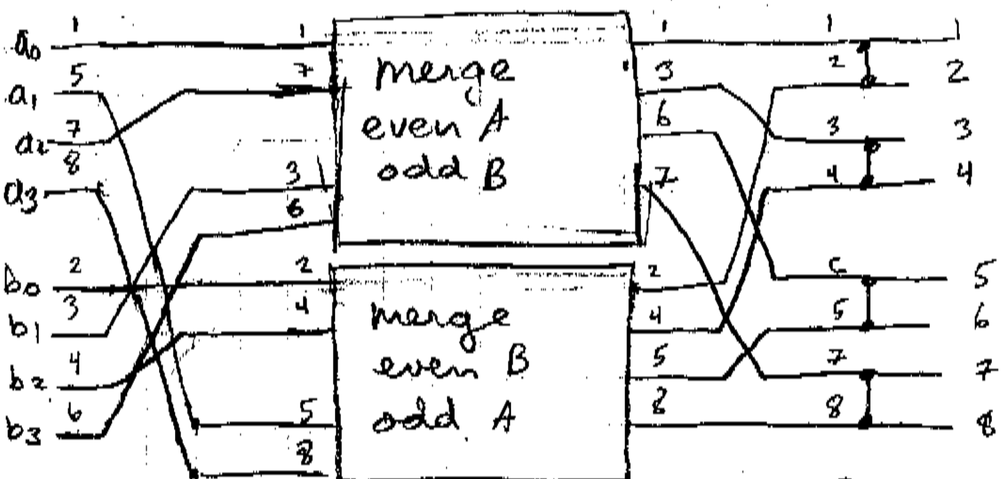
effectively reversed

effectively reversed

Batchers circuit

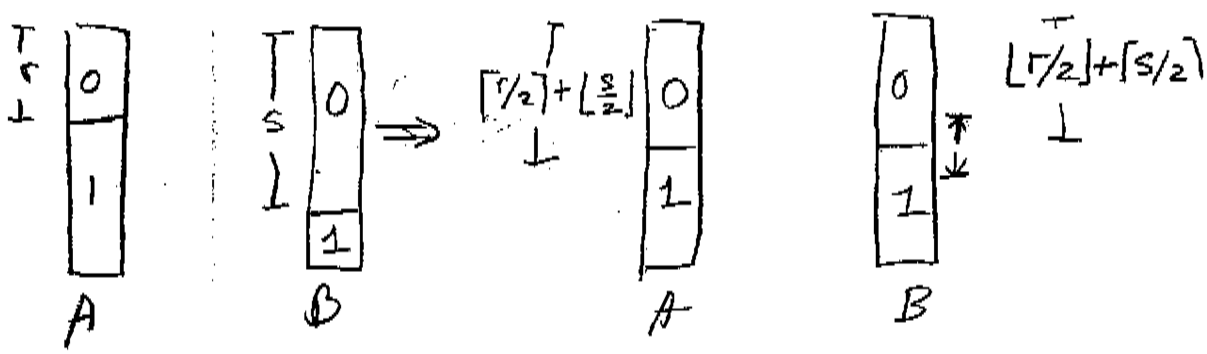
Batcher's Odd-Even Mergesort

step 1: build merger of $A = a_0 \dots a_{N/2}$ $B = b_0 \dots b_{N-1}$



interleave elements

Proof: 0-1 Lemma.



$\Rightarrow$ #0's in each list differs by 1.

Merge $M(N) = M(N/2) + 1$
$= \Theta(N \lg N)$

Sort: $S(N) = S(N/2) + M(N)$
$= \Theta(N \lg^2 N)$.

Longstanding open question:

Does there exist sorting network with depth $O(\lg N)$?

1983: yes! AKS sorting network (Ajtai, Komlós, Szemerédi

Depth: N

comparators: $O(N \lg N)$

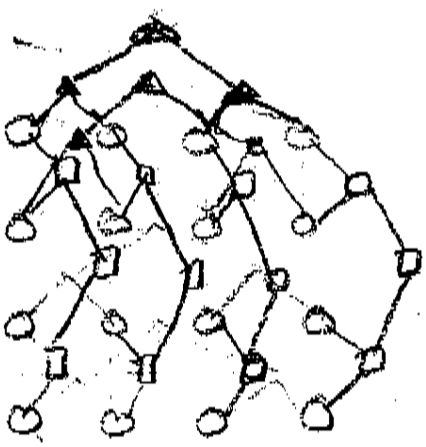
Unfortunately VERY large constants: many thousands!

Sorting on Mesh of Trees.

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

$N \times N$ Grid $\rightarrow$ · remove grid edges

· add tree above every row & column



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

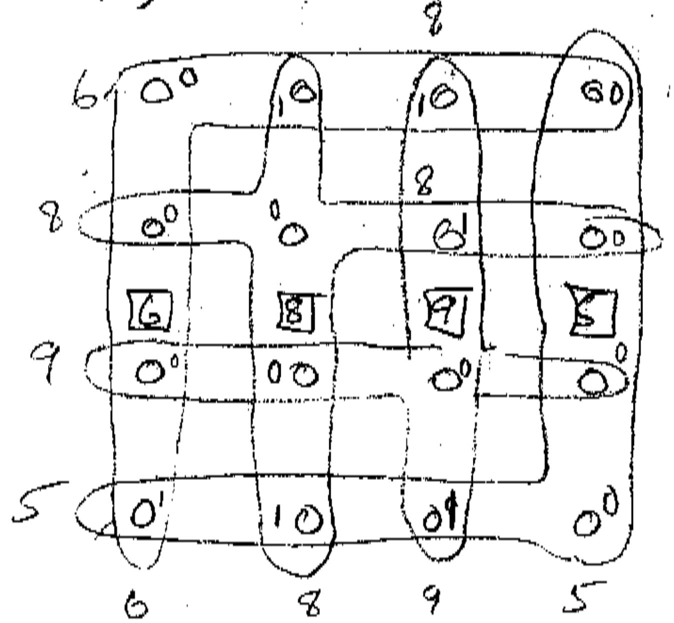
diameter: $4 \lg N$

bisection width: N

recursive decomposition: remove all roots
$\Rightarrow$ 4 separate $M_{2,N/2}$.

Sort: N^2 elmts: $\Omega(N)$ time (bisection LB)

Sort N elmts: $\Theta(\lg N)$ time
$(w_1 \ldots w_N)$



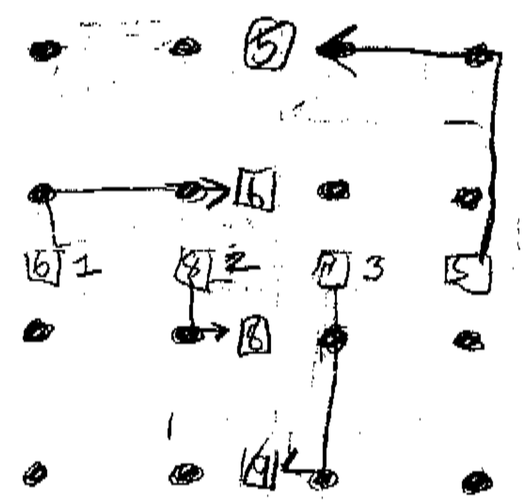
(1) pass w_i along ith row & column

(2) In node p_{ij} (row i, column j) store $\begin{cases} 1, & w_i \le w_j \\ 0, & w_j > w_i \end{cases}$

(3) count # 1's in jth tree
⇒ rank of w_j in sorted order

(4) if rank$(w_j) = r$
⇒ send w_j to rth row tree.

⇒



<< Wow - Routing ! >>

Note: $\Theta(k + \lg N)$ bitsteps for k bit #s. ($2k + 5 \lg N$)

Send MSB first.

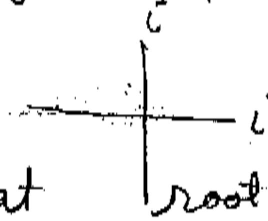
Sort N^2 elmts: $\Omega(N)$ time (bisection LB)

Sort N elmts: $w_1, w_2, \ldots, w_N$ $\Theta(\lg N)$ time

<u>Idea:</u> brute force. Do all comparisons.

Given N k-bit #s, following bit-step algy sorts in $2k + 5\lg N$ ste

(1) $\forall i$, pass w_i along



i^{th} column & row (MSB first) Store at root of each column tree.

(2) For each leaf, bitwise compare w_i & w_j.

(Break ties with index i, j)

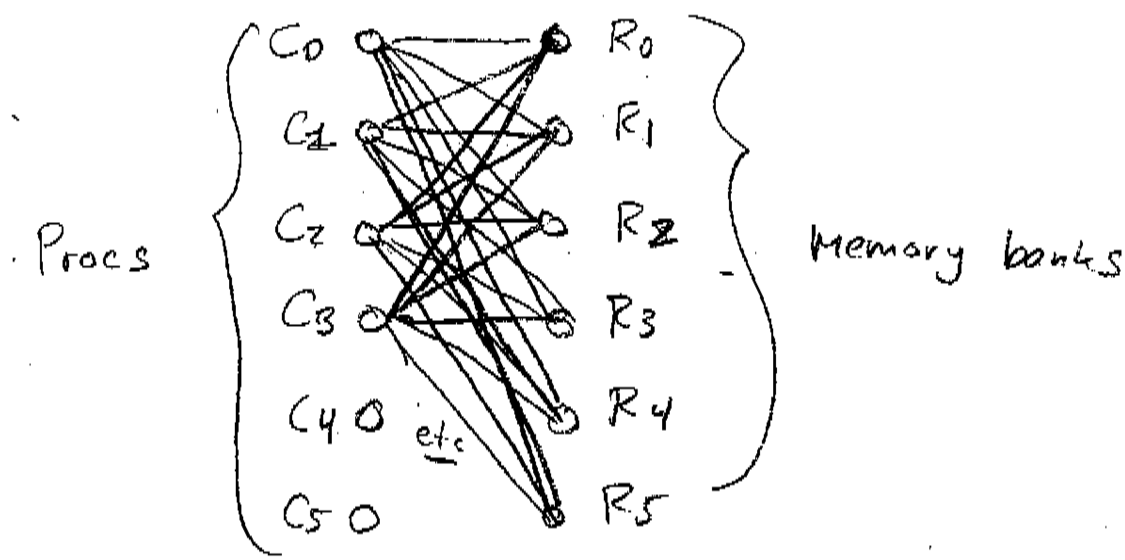
Leaf p_{ij} stores $\begin{cases} 1 & \text{if } w_i < w_j \\ 0 & w_i \geq w_j \end{cases}$

(3) $\forall j$, count # 1's in leaves of j^{th} column tree

$\Rightarrow$ rank of w_j in sorted order.

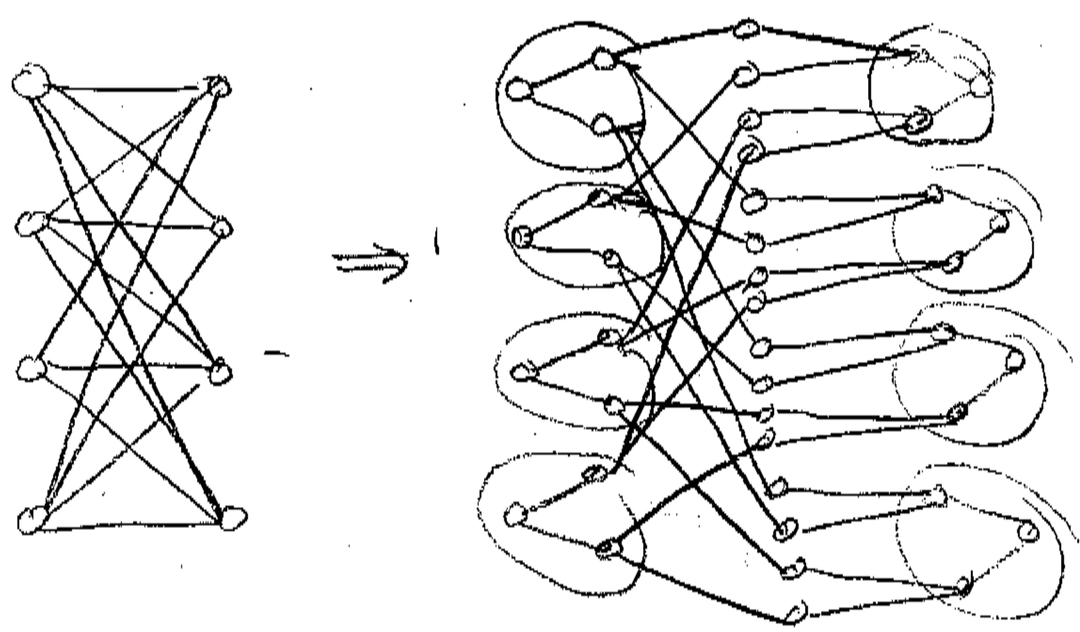
(4) If $\text{rank}(w_j) = r$, send w_j to root of r^{th} row tree

Simulating Bipartite Graph / Ideal Computer on MOT



For large N, K_{NN} not realistically Implementable.

$\Rightarrow$ ~~Simulate K_{NN}~~ by M_{2N} with $2 \lg N$ delay



The catch: quadratic blowup in space/hardware