CSE 6220/CX 4220
Introduction to HPC

Lecture 11: Bitonic Sorting

Helen Xu
hxu615@gatech.edu

Georgia Tech College of Computing
School of Computational
Science and Engineering

Parallel Distributed-Memory Sorting

Which existing sequential algorithm can serve as a good starting point?

- $O(n^2)$: Bubble Sort, Insertion Sort etc.
- $O(n \lg n)$: Heap Sort, Merge Sort, etc.

Divide and conquer fits well with parallel processing, so let's start with **Merge Sort**

- Divide the list into two equal size lists, sort them, and merge

$$T(n) = 2T(n/2) + O(n) = O(n \lg n)$$

Merge Sort

Distributed-Memory Parallel Merge Sort

- $T(n,p) = T\left(\frac{n}{2},\frac{p}{2}\right) + O(n)$
- $T(n,p) = T\left(\frac{n}{4},\frac{p}{4}\right) + O\left(\frac{n}{2}\right) + O(n)$

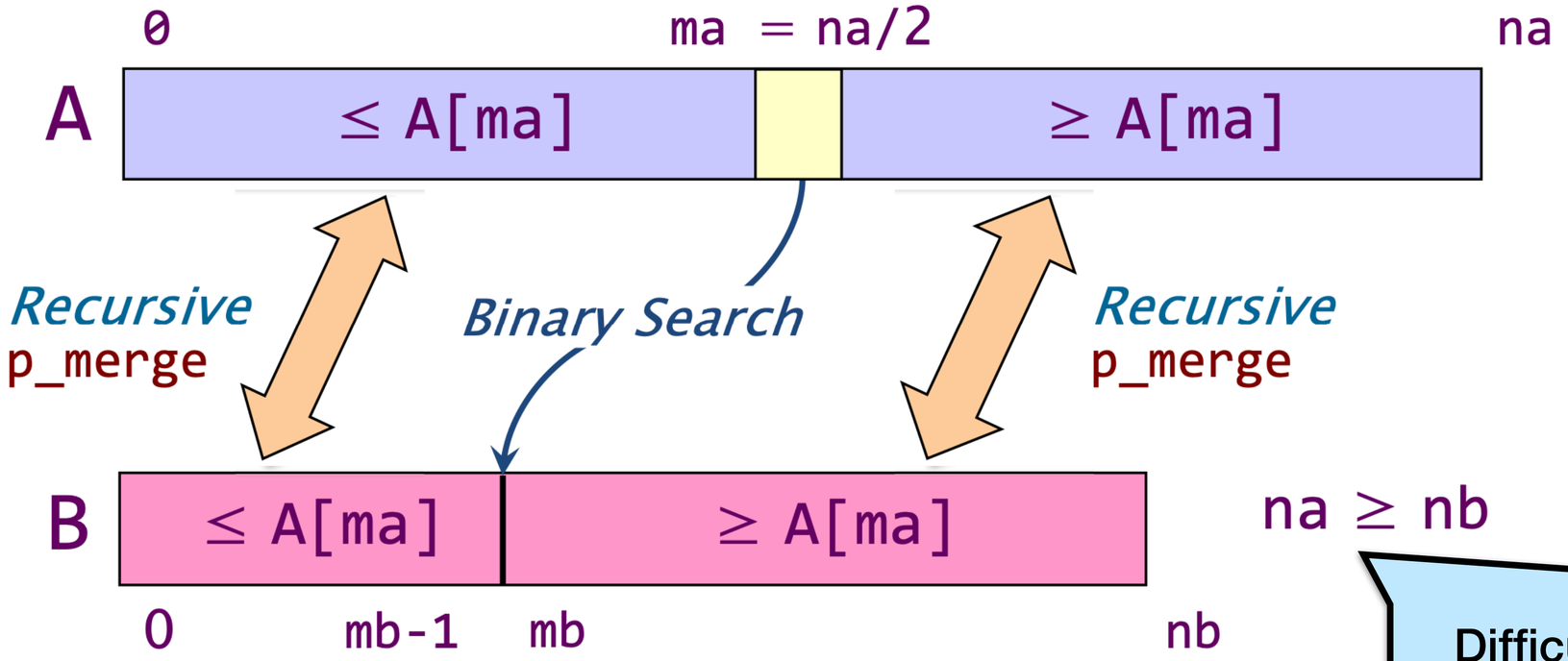
As we discussed before, the merge needs to be **parallelized** to make good use of the processors.

$$\Downarrow$$

- $T(n,p) = T\left(\frac{n}{p},1\right) + O\left(n + \frac{n}{2} + \frac{n}{4} + \cdots + \frac{n}{p/2}\right)$
- $T(n,p) = O\left(\frac{n}{p}\log\frac{n}{p} + n\right)$

Can't use more than $\log n$ processors!

Recall: Parallel Merge



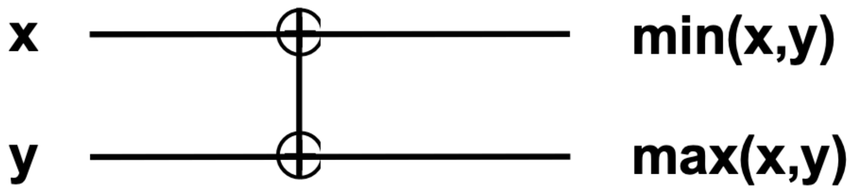
KEY IDEA: If the total number of elements to be merged in the two arrays is $n = na + nb$, the total number of elements in the larger of the two recursive merges is at most $(3/4)n$.

Difficult for distributed memory because the subproblems are uneven - **load balancing** is hard

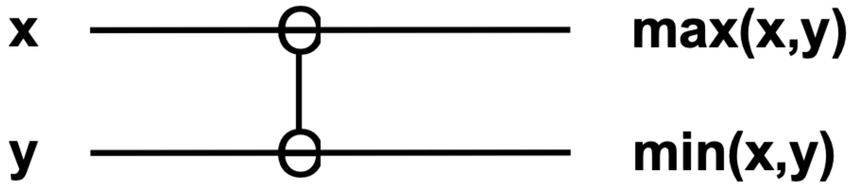
Bitonic Sort

Sorting Network

- **Network of comparators designed for sorting**
- **Comparator : two inputs x and y; two outputs x' and y'**
  - **—types**
    - **increasing (denoted ⊕): x' = min(x,y) and y' = max(x,y)**



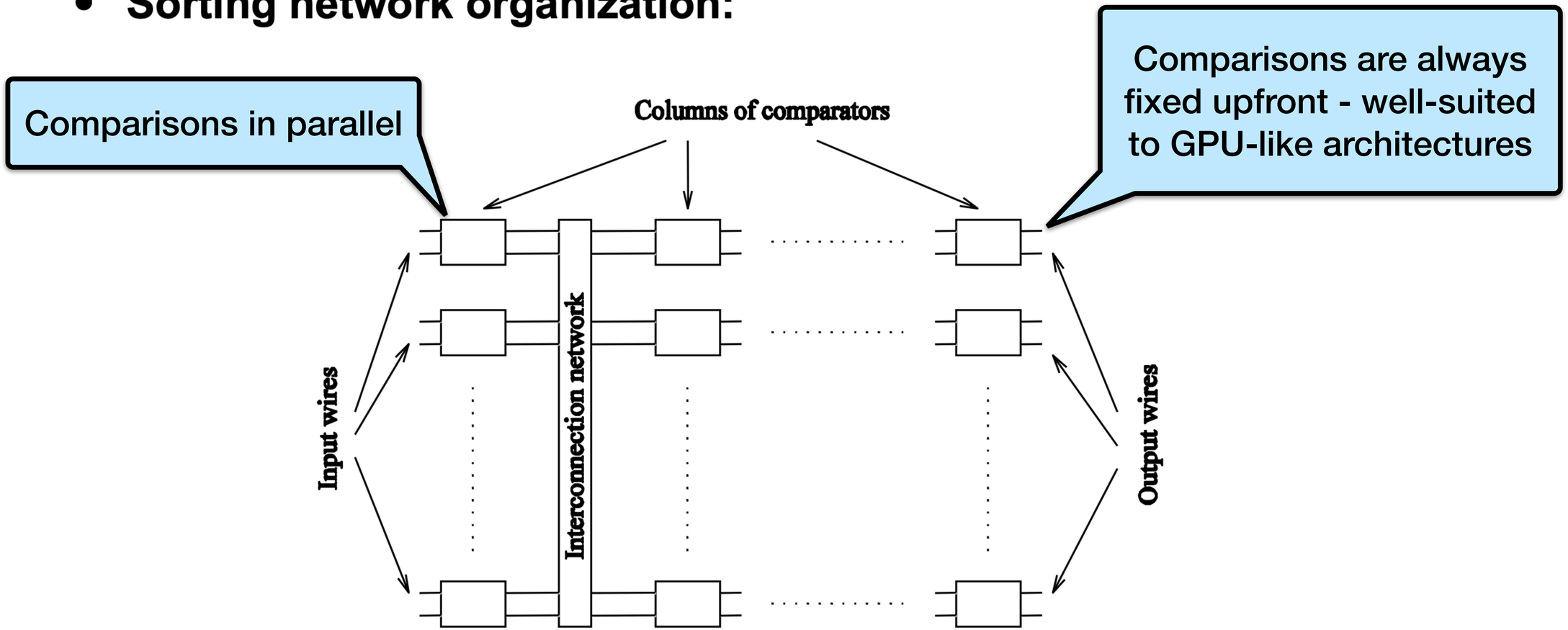
    - **decreasing (denoted ⊖) : x' = max(x,y) and y' = min(x,y)**



- **Sorting network speed is proportional to its depth**

Sorting Network

- **Network structure: a series of columns**
- **Each column consists of a vector of comparators (in parallel)**
- **Sorting network organization:**



From John Mellor-Crummey

Example: Bitonic Sorting Network

Bitonic sequence

- two parts: increasing and decreasing

  ⟨1,2,4,7,6,0⟩ : first increases and then decreases (or vice versa)

- cyclic rotation of a bitonic sequence is also considered bitonic

  ⟨8,9,2,1,0,4⟩ : cyclic rotation of ⟨0,4,8,9,2,1⟩

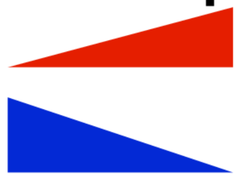
Bitonic sorting network

- sorts n elements in $\Theta(\lg^2 n)$ time
- network kernel: rearranges a bitonic sequence **into a sorted one**

From John Mellor-Crummey

Bitonic Split (Batcher, 1968)

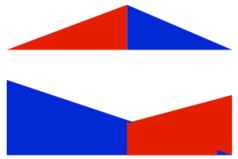
- Let $s = \langle a_0, a_1, \ldots, a_{n-1} \rangle$ be a bitonic sequence such that
  - $a_0 \le a_1 \le \cdots \le a_{n/2-1}$, and
  - $a_{n/2} \ge a_{n/2+1} \ge \cdots \ge a_{n-1}$



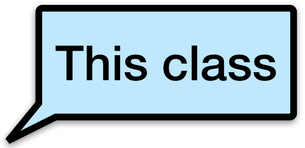
- Consider the following subsequences of s

$$s_1 = \langle \min(a_0, a_{n/2}), \min(a_1, a_{n/2+1}), \ldots, \min(a_{n/2-1}, a_{n-1}) \rangle$$

$$s_2 = \langle \max(a_0, a_{n/2}), \max(a_1, a_{n/2+1}), \ldots, \max(a_{n/2-1}, a_{n-1}) \rangle$$



- Sequence properties
  - s_1 and s_2 are both bitonic
  - $\forall_x \forall_y \; x \in s_1, y \in s_2 , x < y$

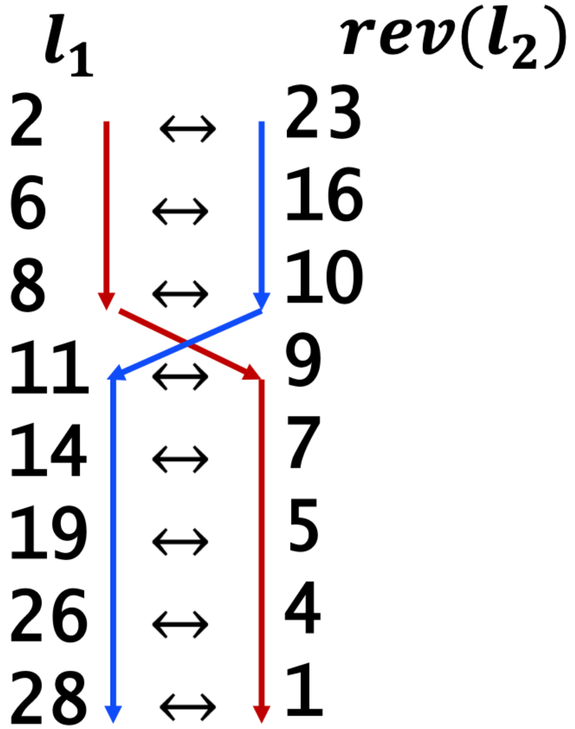


- Apply recursively on s_1 and s_2 to produce a sorted sequence
- Works for any bitonic sequence, even if $|s_1| \neq |s_2|$

Example: Bitonic Split

- l_1: 2, 6, 8, 11, 14, 19, 26, 28
- l_2: 1, 4, 5, 7, 9, 10, 16, 23

Non-decreasing sorted lists



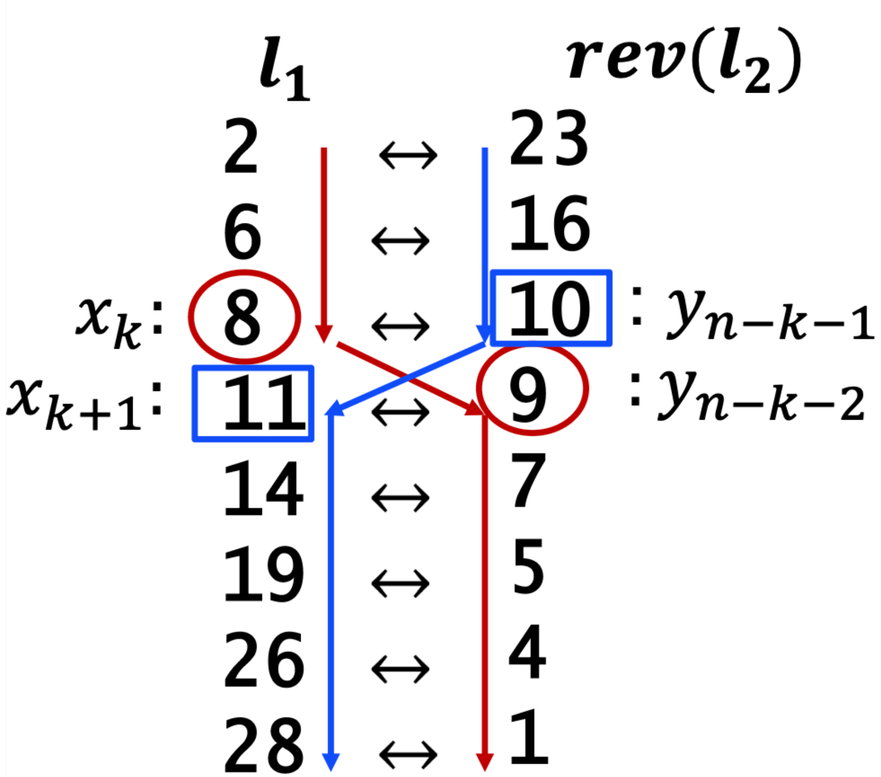
l_{min}	l_{max}
2	23
6	16
8	10
9	11
7	14
5	19
4	26
1	28

How to prove this?

All elements of l_{min} are smaller than l_{max}

Example: Bitonic Split

- $l_1: x_0,\ x_1, \cdots, x_{n-1}$
- $l_2: y_0,\ y_1, \cdots, y_{n-1}$

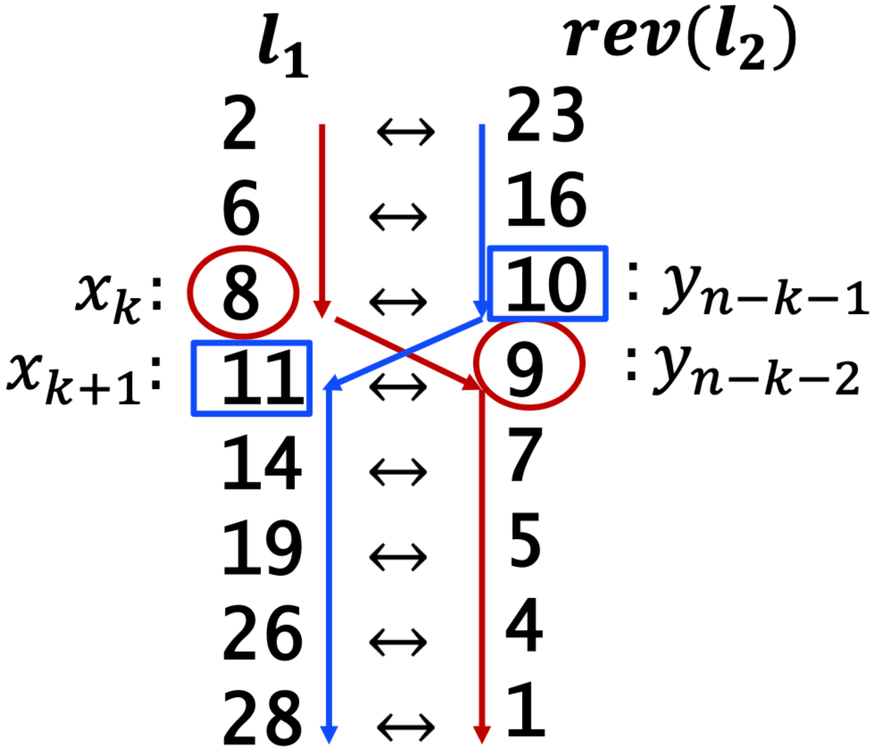


l_{min}
2
6
8
9
7
5
4
1

l_{max}
23
16
10
11
14
19
26
28

Proof: Bitonic Split

- Prove: $\max(l_{min}) \leq \min(l_{max})$
- $\max(l_{min}) = \max(x_k, y_{n-k-2})$
- $\min(l_{max}) = \min(x_{k+1}, y_{n-k-1})$
- Proof:
  1. $x_k \leq x_{k+1}$
     - l_1 is sorted
  2. $x_k \leq y_{n-k-1}$
     - Transition after k
  3. $y_{n-k-2} \leq x_{k+1}$
     - Transition after k
  4. $y_{n-k-2} \leq y_{n-k-1}$
     - l_2 is sorted



First Attempt: Bitonic Sort Analysis

Recursive sort

Pairwise comparison

Sort Imin and Imax

- $T(n,p) = T\left(\frac{n}{2},\frac{p}{2}\right) + O\left(\frac{n}{p}\right) + T\left(\frac{n}{2},\frac{p}{2}\right)$
- $T(n,p) = 2T\left(\frac{n}{2},\frac{p}{2}\right) + O\left(\frac{n}{p}\right)$
- $T(n,p) = 2\left(2T\left(\frac{n}{4},\frac{p}{4}\right) + \left(\frac{n/2}{p/2}\right)\right) + O\left(\frac{n}{p}\right)$
- $T(n,p) = 2^2T\left(\frac{n}{4},\frac{p}{4}\right) + O\left(2\frac{n}{p} + \frac{n}{p}\right)$
- $T(n,p) = 2^3T\left(\frac{n}{8},\frac{p}{8}\right) + O\left(2^2\frac{n}{p} + 2\frac{n}{p} + \frac{n}{p}\right)$
- $T(n,p) = pT\left(\frac{n}{p},1\right) + O\left(\frac{n}{p}\left(1 + 2 + 2^2 + \cdots \frac{p}{2}\right)\right)$ $(p-1)$
- Each processor sorts locally in $O\left(\frac{n}{p}\log\frac{n}{p}\right)$ time.
- $T(n,p) = O\left(p\frac{n}{p}\log\frac{n}{p} + n\right)$ $= O\left(n\log\frac{n}{p}\right)$

First Attempt: Bitonic Sort Analysis

Recursive sort

Pairwise comparison

- $T(n,p) = T\left(\frac{n}{2},\frac{p}{2}\right) + O\left(\frac{n}{p}\right) + T\left(\frac{n}{2},\frac{p}{2}\right)$

Sort lmin and lmax

- $T(n,p) = 2T\left(\frac{n}{2},\frac{p}{2}\right) + O\left(\frac{n}{p}\right)$
- $T(n,p) = 2\left(2T\left(\frac{n}{4},\frac{p}{4}\right) + \left(\frac{n/2}{p/2}\right)\right) + O\left(\frac{n}{p}\right)$
- $T(n,p) = 2^2T\left(\frac{n}{4},\frac{p}{4}\right) + O\left(2\frac{n}{p} + \frac{n}{p}\right)$
- $T(n,p) = 2^3T\left(\frac{n}{8},\frac{p}{8}\right) + O\left(2^2\frac{n}{p} + 2\frac{n}{p} + \frac{n}{p}\right)$
- $T(n,p) = pT\left(\frac{n}{p},1\right) + O\left(\frac{n}{p}\left(1 + 2 + 2^2 + \cdots \frac{p}{2}\right)\right)$

$(p-1)$

- Each processor sorts locally in $O\left(\frac{n}{p}\log\frac{n}{p}\right)$ time.
- $T(n,p) = O\left(\cancel{p}\frac{n}{\cancel{p}}\log\frac{n}{p} + n\right)$

$$= O\left(n\log\frac{n}{p}\right)$$

Even worse than merge sort with serial merge, which was $O(n\lg(n/p))$

First Attempt: Bitonic Sort Analysis

Recursive sort

Pairwise comparison

Sort lmin and lmax

Problem - exponentially increasing down the recursion - 4 total sorts and 2 can happen in parallel at once

- $T(n,p) = T\left(\frac{n}{2},\frac{p}{2}\right) + O\left(\frac{n}{p}\right) + T\left(\frac{n}{2},\frac{p}{2}\right)$

  $2T\left(\frac{n}{2},\frac{p}{2}\right) + O\left(\frac{n}{p}\right)$

- $T(n,p) = 2\left(2T\left(\frac{n}{4},\frac{p}{4}\right) + \left(\frac{n/2}{p/2}\right)\right) + O\left(\frac{n}{p}\right)$
- $T(n,p) = 2^2 T\left(\frac{n}{4},\frac{p}{4}\right) + O\left(2\frac{n}{p} + \frac{n}{p}\right)$
- $T(n,p) = 2^3 T\left(\frac{n}{8},\frac{p}{8}\right) + O\left(2^2\frac{n}{p} + 2\frac{n}{p} + \frac{n}{p}\right)$
- $T(n,p) = pT\left(\frac{n}{p},1\right) + O\left(\frac{n}{p}\left(1 + 2 + 2^2 + \cdots \frac{p}{2}\right)\right)$

  $(p-1)$

- Each processor sorts locally in $O\left(\frac{n}{p}\log\frac{n}{p}\right)$ time.
- $T(n,p) = O\left(\cancel{p}\frac{n}{\cancel{p}}\log\frac{n}{p} + n\right)$

$$= O\left(n\log\frac{n}{p}\right)$$

Even worse than merge sort with serial merge, which was $O(n\lg(n/p))$

From Srinivas Aluru

First Attempt: Bitonic Sort Analysis

Recursive sort

Pairwise comparison

- $T(n,p) = T\left(\frac{n}{2},\frac{p}{2}\right) + O\left(\frac{n}{p}\right) + T\left(\frac{n}{2},\frac{p}{2}\right)$

Problem - exponentially increasing down the recursion - 4 total sorts and 2 can happen in parallel at once

Sort lmin and lmax

- $2T\left(\frac{n}{2},\frac{p}{2}\right) + O\left(\frac{n}{p}\right)$
- $T(n,p) = 2\left(2T\left(\frac{n}{4},\frac{p}{4}\right) + \left(\frac{n/2}{p/2}\right)\right) + O\left(\frac{n}{p}\right)$
- $T(n,p) = 2^2 T\left(\frac{n}{4},\frac{p}{4}\right) + O\left(2\frac{n}{p} + \frac{n}{p}\right)$
- $T(n,p) = 2^3 T\left(\frac{n}{8},\frac{p}{8}\right) + O\left(2^2\frac{n}{p} + 2\frac{n}{p} + \frac{n}{p}\right)$
- $T(n,p) = pT\left(\frac{n}{p},1\right) + O\left(\frac{n}{p}\left(1 + 2 + 2^2 + \cdots \frac{p}{2}\right)\right)$

Observation: lmin and lmax are not arbitrary sequences. They are bitonic - can we use this?

$(p-1)$

- Each processor sorts locally in $O\left(\frac{n}{p}\log\frac{n}{p}\right)$ time.
- $T(n,p) = O\left(p\frac{n}{p}\log\frac{n}{p} + n\right)$

$$= O\left(n\log\frac{n}{p}\right)$$

Even worse than merge sort with serial merge, which was $O(n\lg(n/p))$

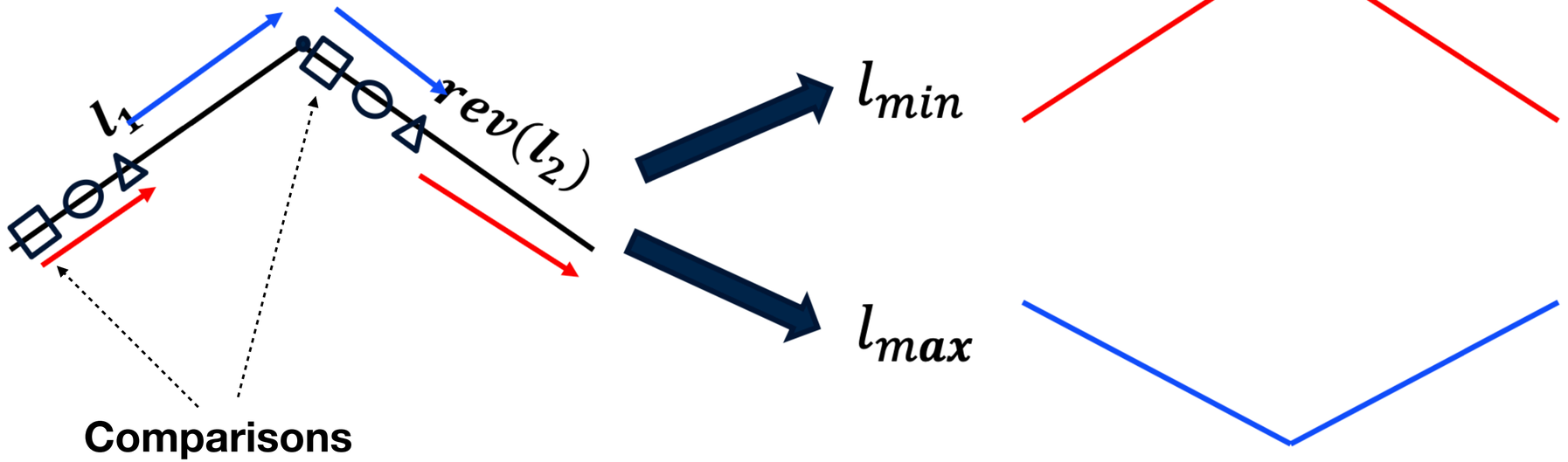
From Srinivas Aluru

Bitonic Splits

Two monotonic sequences

Inflection points may not be perfectly in half

- Bitonic sequence:

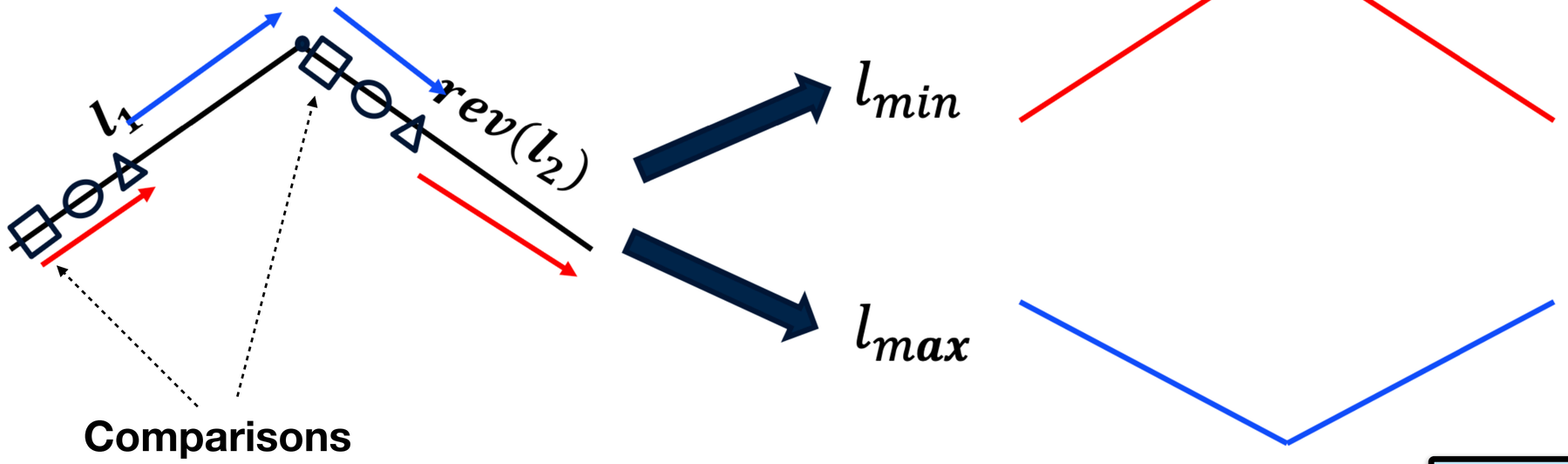


One bitonic sequence ⟶ Two bitonic sequences

Bitonic Splits

Two monotonic sequences

- Bitonic sequence:



Inflection points may not be perfectly in half

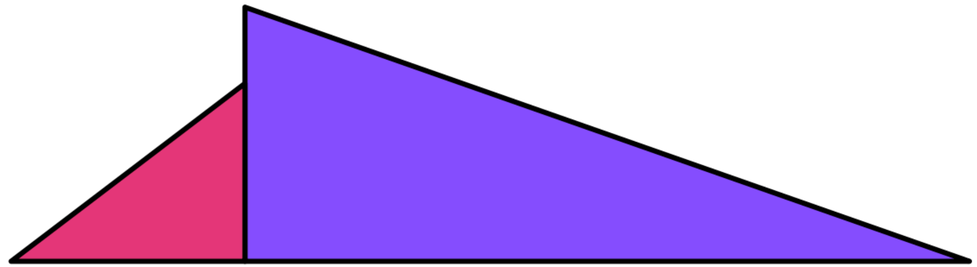
One bitonic sequence → **Two bitonic sequences**

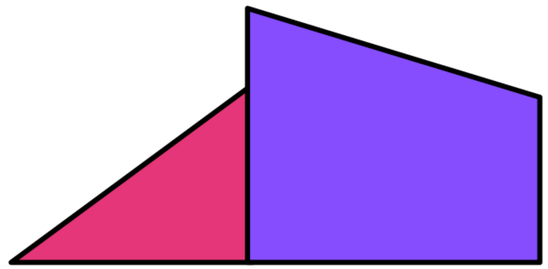
In this case - is it true in general?

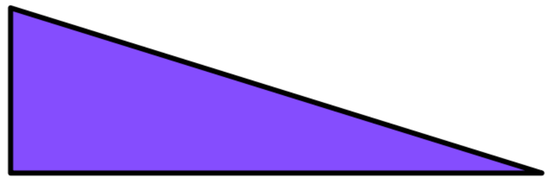
Goal: Apply repeated bitonic splits with no additional work (e.g., do not figure out where the inflection points are)

From Srinivas Aluru

Splitting Bitonic Sequences



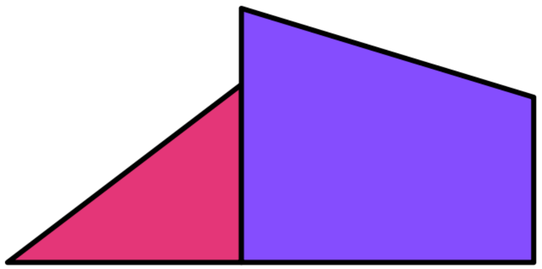


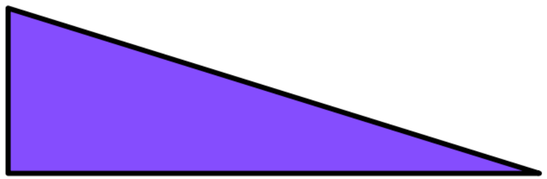


Sequence properties

s_1 and s_2 are both bitonic

$\forall_x \forall_y \; x \in s_1, y \in s_2 , x < y$



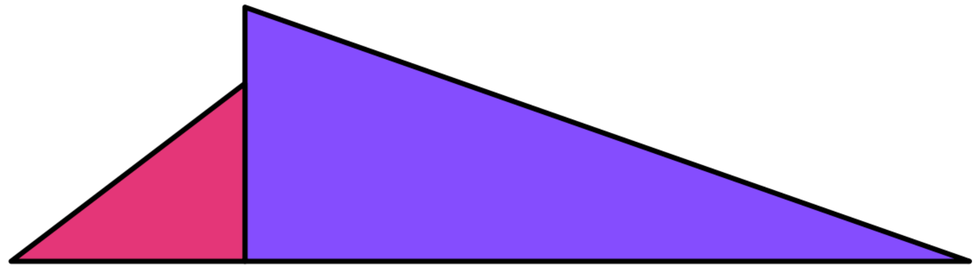


min

max

From John Mellor-Crummey

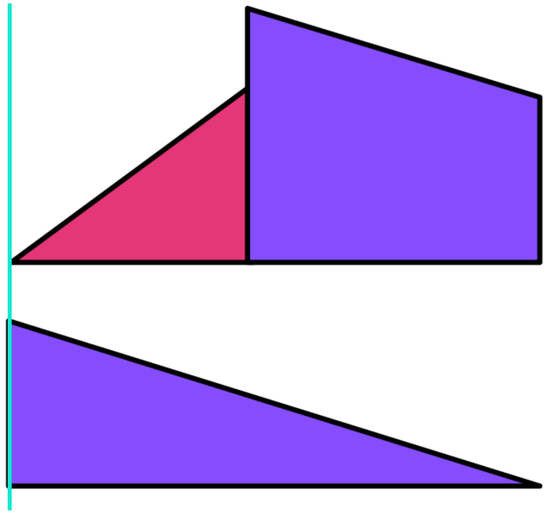
Splitting Bitonic Sequences

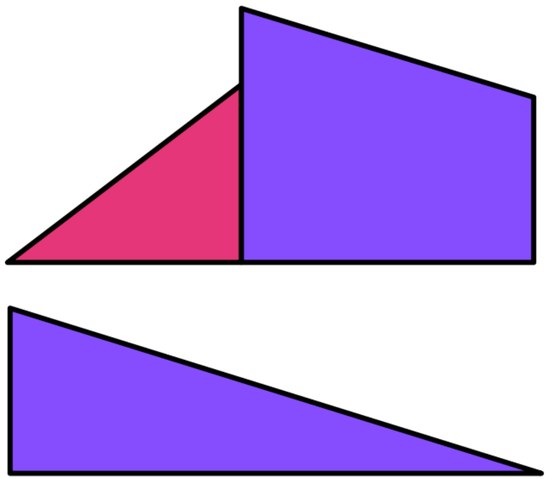


Sequence properties

s_1 and s_2 are both bitonic

$\forall_x \forall_y$ x $\in s_1$, y $\in s_2$, x < y



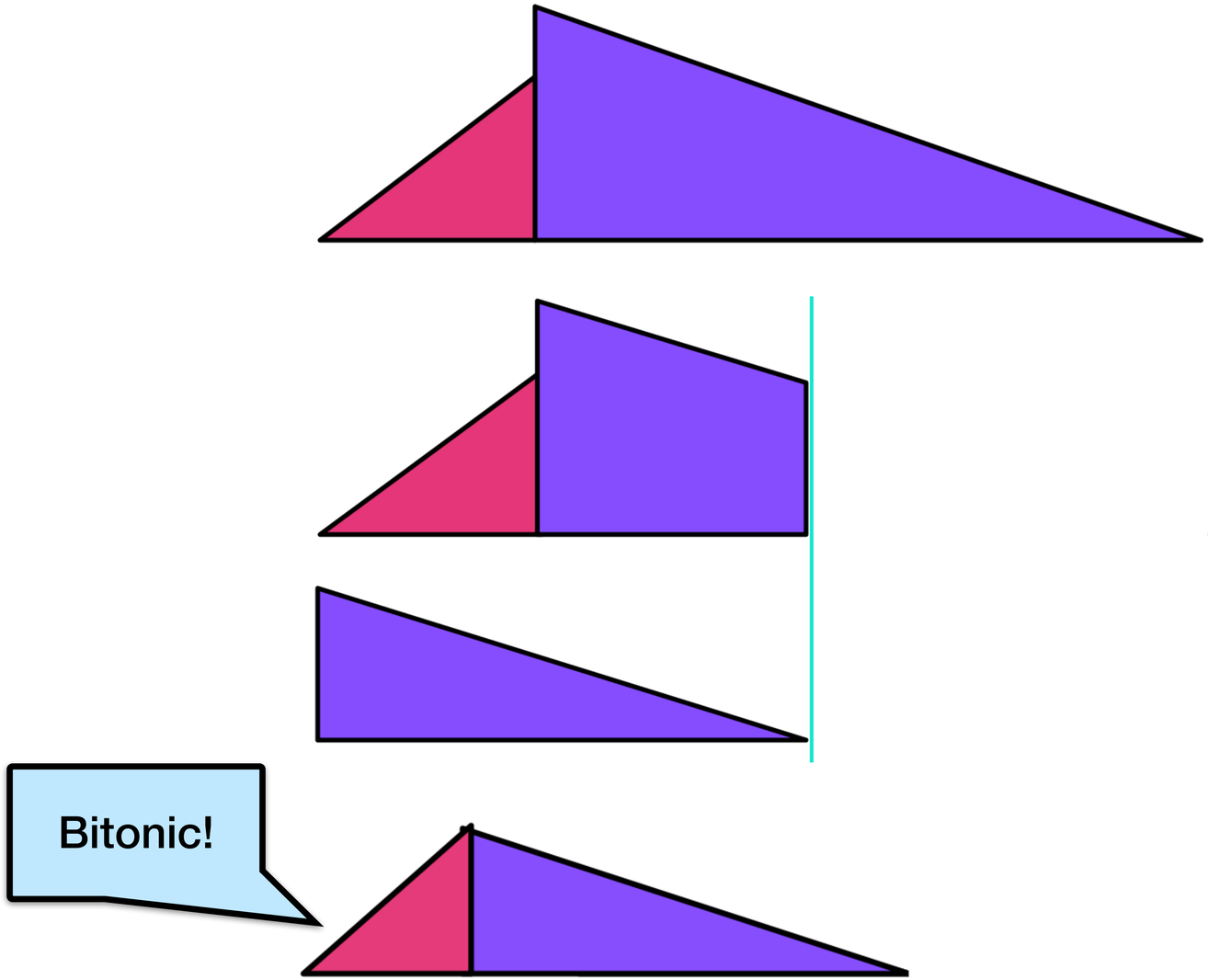


From John Mellor-Crummey

min

max

Splitting Bitonic Sequences



Sequence properties

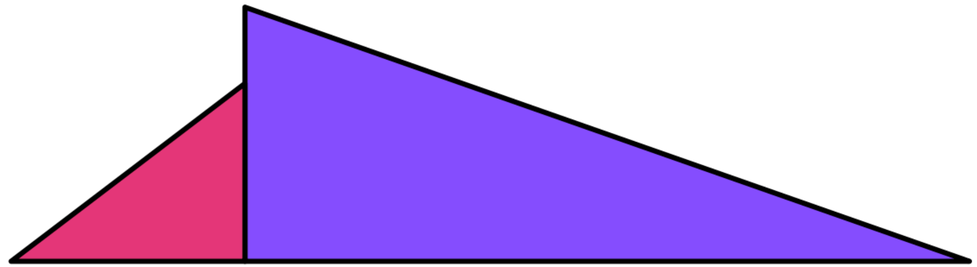
s_1 and s_2 are both bitonic

$\forall_x \forall_y \; x \in s_1, y \in s_2 , x < y$

min

max

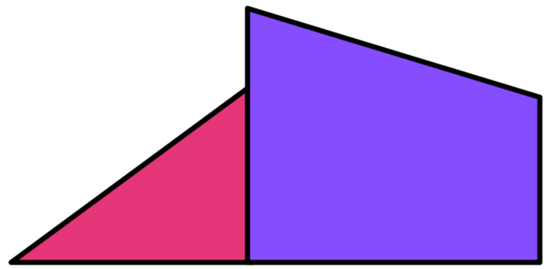
Splitting Bitonic Sequences

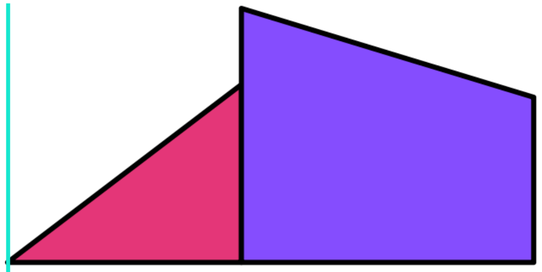


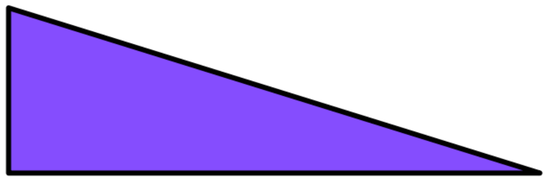
Sequence properties

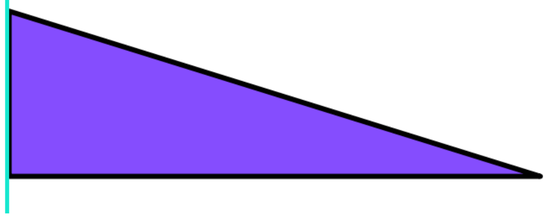
s_1 and s_2 are both bitonic

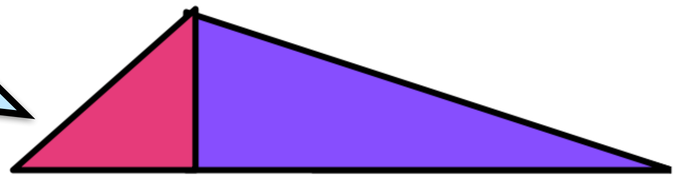
$\forall_x \forall_y \; x \in s_1, y \in s_2 , x < y$







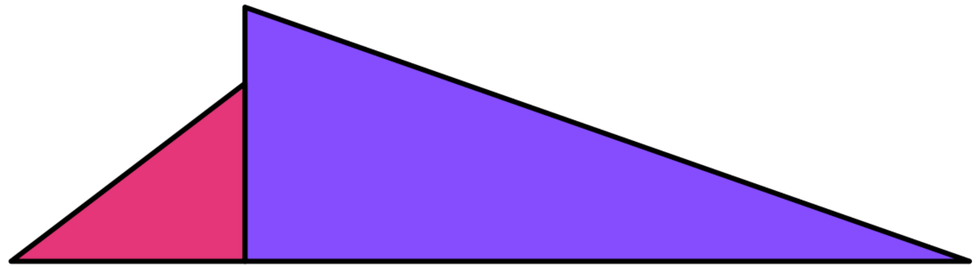




min

max

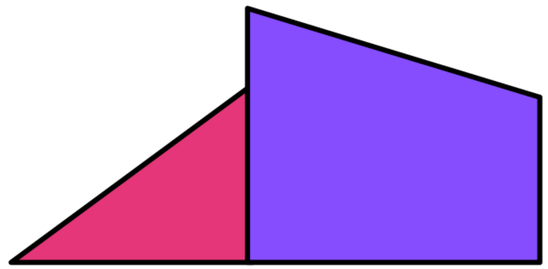
Splitting Bitonic Sequences

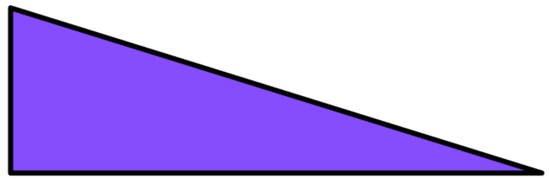


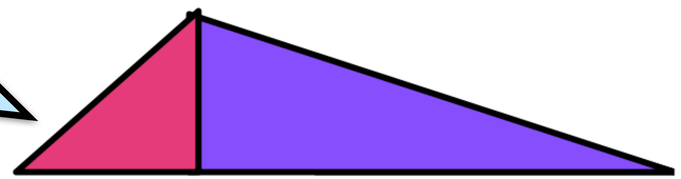
Sequence properties

s_1 and s_2 are both bitonic

$\forall_x \forall_y \; x \in s_1, y \in s_2 , x < y$

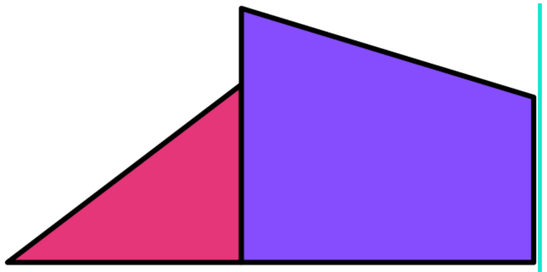


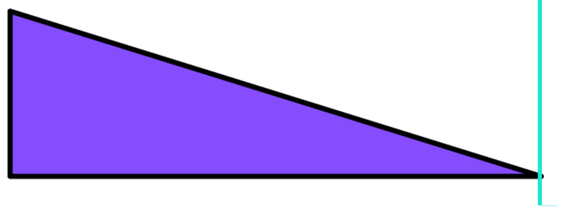


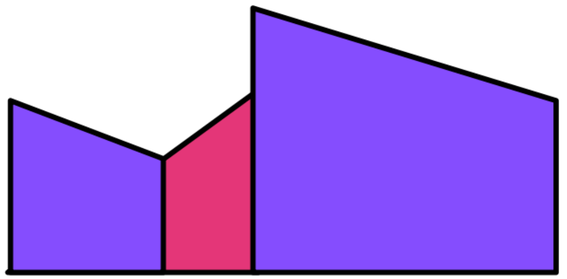


Bitonic!

min



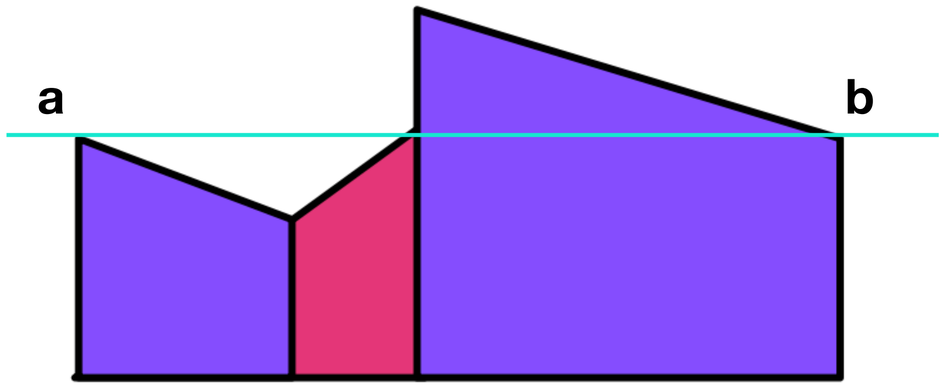




Not bitonic?

max

Observation: Properties of l_{max}



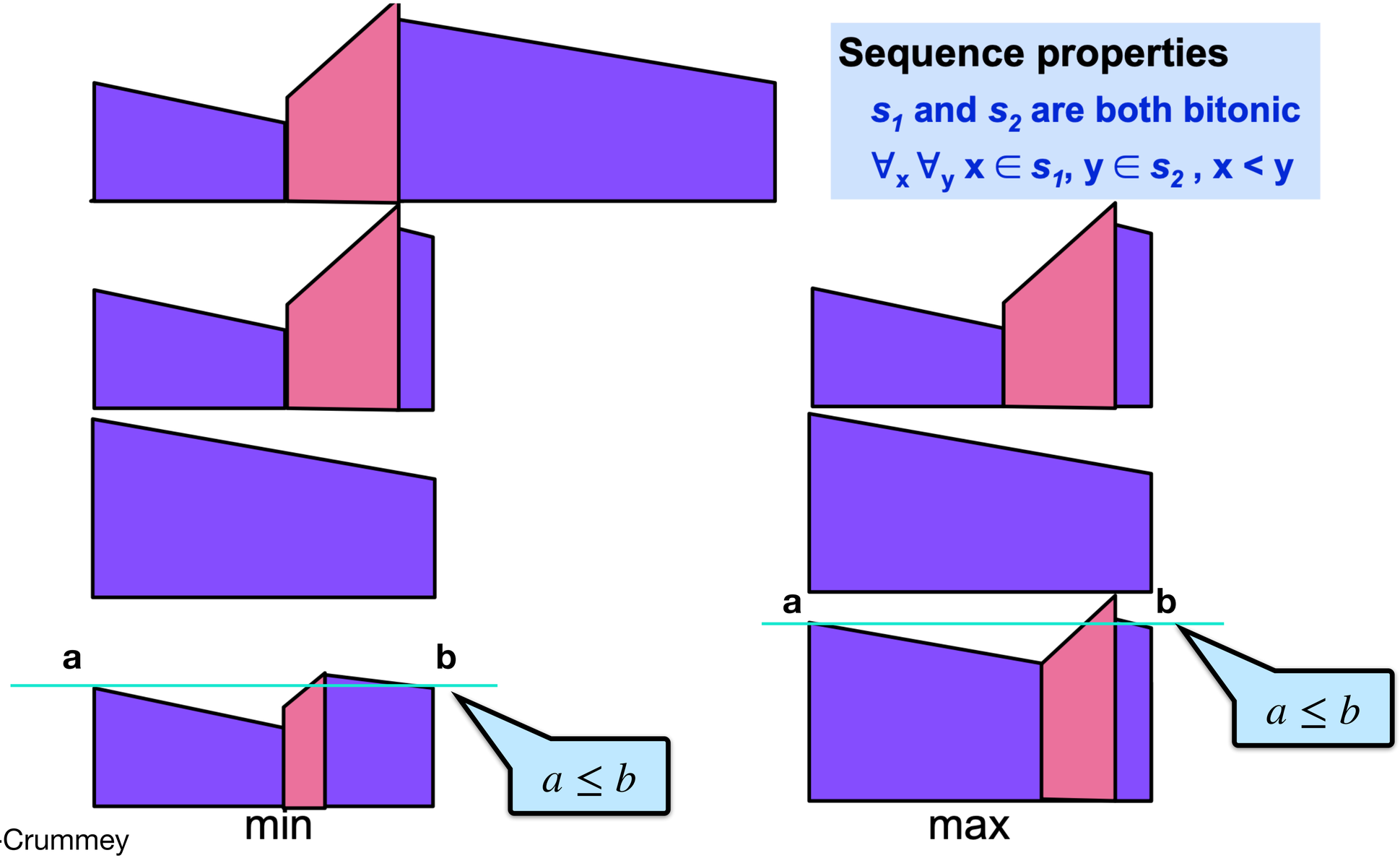
- By construction, $a \leq b$ because the part we got them from was monotonically decreasing.

- So l_{max} is a **cyclical shift** of a bitonic sequence.

To be clear, we do not want to actually find where the rotation point is

From John Mellor-Crummey

Splitting Bitonic Sequences

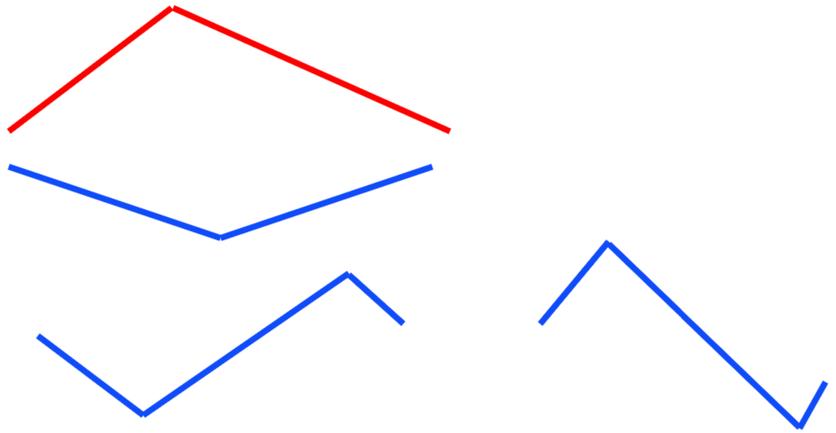


From John Mellor-Crummey

Bitonic Sequence - Formal Definition

- **Bitonic sequence**: $x_0,\ x_1, \cdots, x_{n-1}$ is bitonic if $\exists k$ such that
  - **(T1)** $x_0,\ x_1, \cdots, x_k$ is non-decreasing and $x_{k+1},\ x_{k+2}, \cdots, x_{n-1}$ is non-increasing OR
  - **(T2)** $x_0,\ x_1, \cdots, x_k$ is non-increasing and $x_{k+1},\ x_{k+2}, \cdots, x_{n-1}$ is non-decreasing OR
  - **(T3)** There is a cyclical shift of the sequence that makes (a) or (b) true.

- Examples:
  - 3, 5, 7, 10, 15, 14, 12, 9, 8, 4, 1
  - 14, 12, 9, 8, 4, 1, 3, 5, 7, 10, 15
  - 8, 4, 1, 3, 5, 7, 10, 15, 14, 12, 9



Bitonic Split

- **Bitonic split**: of a bitonic sequence $l = x_0,\ x_1, \cdots, x_{n-1}$ is defined as decomposition of l into

- $l_{min} = \min\left(x_0, x_{\frac{n}{2}}\right), \min\left(x_1, x_{\frac{n}{2}+1}\right), \cdots, \min\left(x_{\frac{n}{2}-1}, x_{n-1}\right)$

- $l_{max} = \max\left(x_0, x_{\frac{n}{2}}\right), \max\left(x_1, x_{\frac{n}{2}+1}\right), \cdots, \max\left(x_{\frac{n}{2}-1}, x_{n-1}\right)$

Lemma

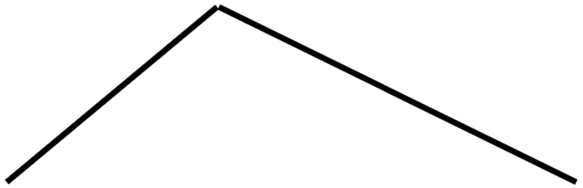
- Let l be a bitonic sequence, and l_{min} and l_{max} result from its bitonic split, then
  1. l_{min} and l_{max} are bitonic — for recursion
  2. $\max(l_{min}) \leq \min(l_{max})$ — avoid sorting

Algorithm - given the two sorted lists l_1, l_2, avoid sorting and just keep applying bitonic split

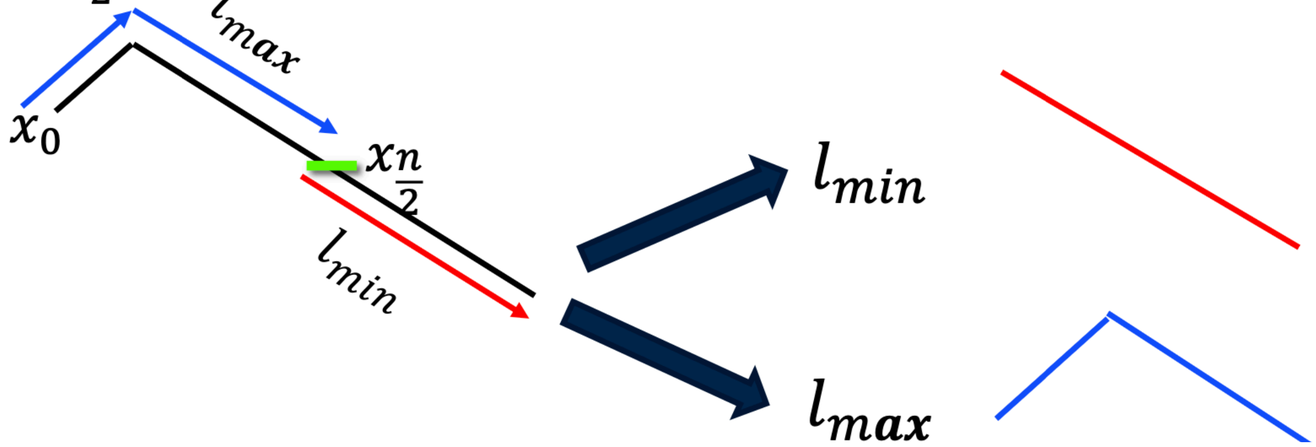
Lemma

- Let l be a bitonic sequence, and l_{min} and l_{max} result from its bitonic split, then
  1. l_{min} and l_{max} are bitonic — for recursion
  2. $\max(l_{min}) \leq \min(l_{max})$ — avoid sorting

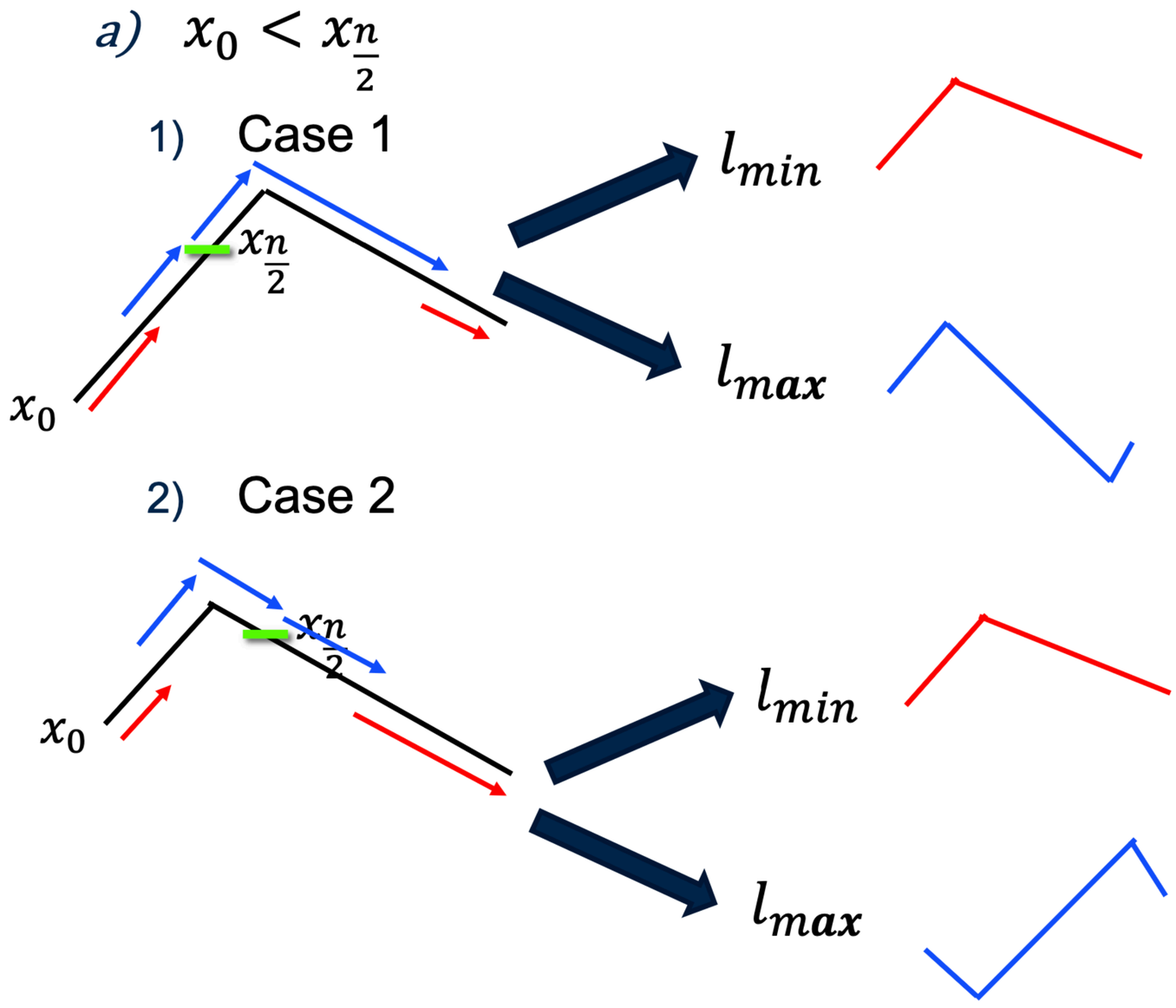
Algorithm - given the two sorted lists l_1, l_2, avoid sorting and just keep applying bitonic split



- **Partial Proof:**
- Let l be a bitonic sequence of type T1
  a) $x_0 > x_{\frac{n}{2}}$; Then, $k < \frac{n}{2}$



Partial proof of lemma



Sub-Lemma

- Let l be a bitonic sequence and l' be a cyclical shift
- Bitonic split l into l_{min} and l_{max}
- Bitonic split l' into l'_{min} and l'_{max}
- Then, l'_{min} is a cyclical shift of l_{min} and l'_{max} is a cyclical shift of l_{max}

not actually doing the rotation, just for analysis

Sub-Lemma

- Let l be a bitonic sequence and l' be a cyclical shift
- Bitonic split l into l_{min} and l_{max}
- Bitonic split l' into l'_{min} and l'_{max}
- Then, l'_{min} is a cyclical shift of l_{min} and l'_{max} is a cyclical shift of l_{max}

not actually doing the rotation, just for analysis

- Let
  - $l = x_0,\ x_1, x_2, \cdots, x_{n-1}$ and
  - $l' = x_k,\ x_{k+1}, \cdots, x_{n-1}, x_0, \cdots, x_{k-1}$
  - $l_{min} = \min\left(x_0, x_{\frac{n}{2}}\right), \min\left(x_1, x_{\frac{n}{2}+1}\right), \cdots, \min\left(x_{\frac{n}{2}-1}, x_{n-1}\right)$
  - $l'_{min} = \min\left(x_k, x_{\left(k+\frac{n}{2}\right) mod\ n}\right), \cdots$
- Cyclical shift does not change the bitonic nature of the sequence nor the min (or max) element in the sequence!

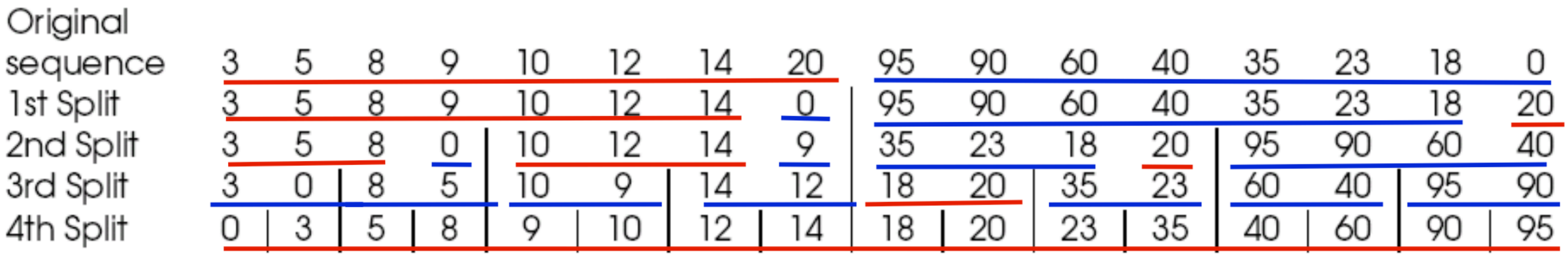
From Srinivas Aluru

Bitonic Merge Example

Sort a bitonic sequence through a sequence of bitonic splits

Example: use bitonic merge to sort 16-element bitonic sequence

How: perform a series of $\log_2 16 = 4$ bitonic splits



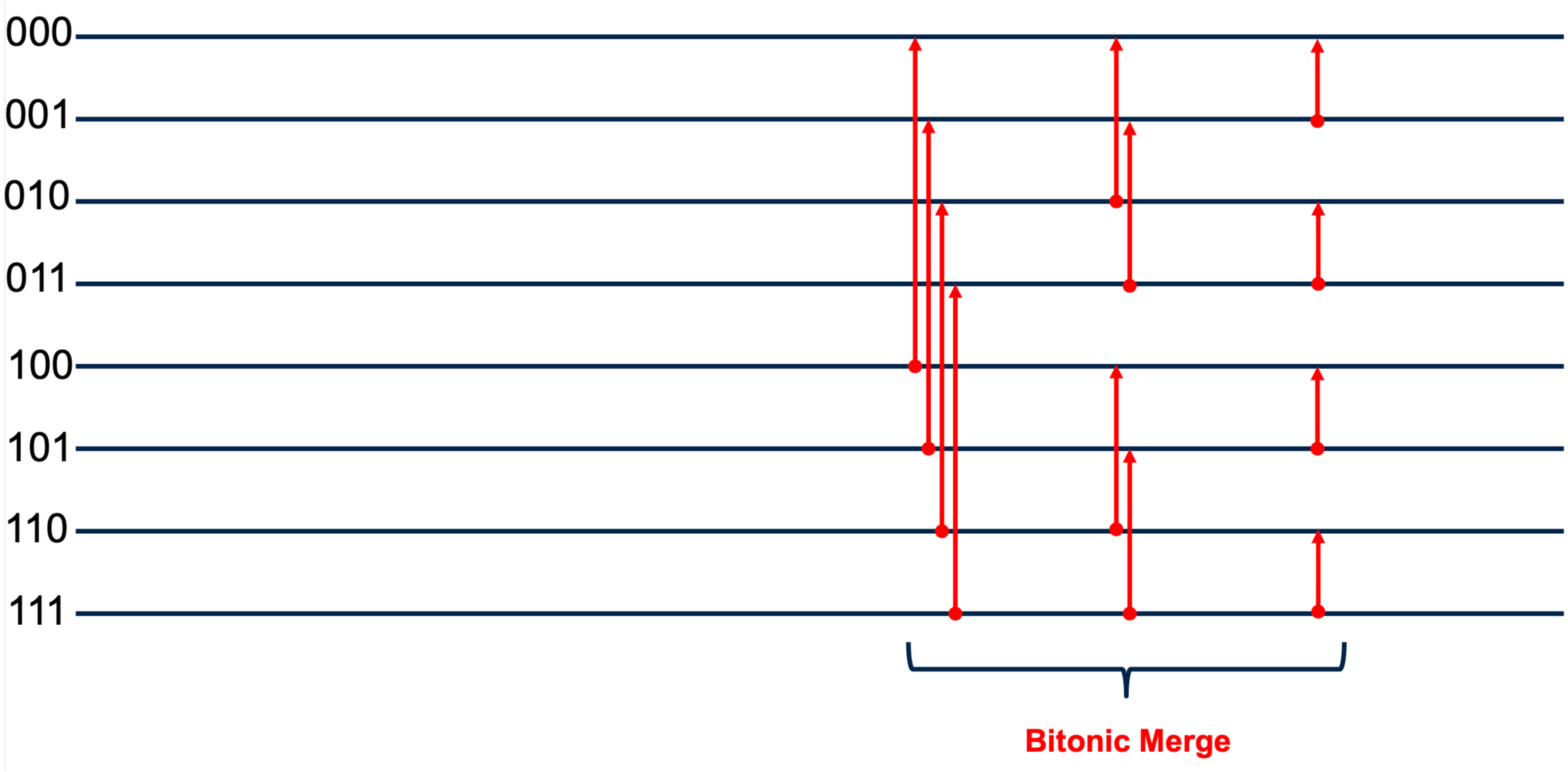
Bitonic_merge(p, p) = O(log p)

From John Mellor-Crummey

Sorting via Bitonic Merging Network

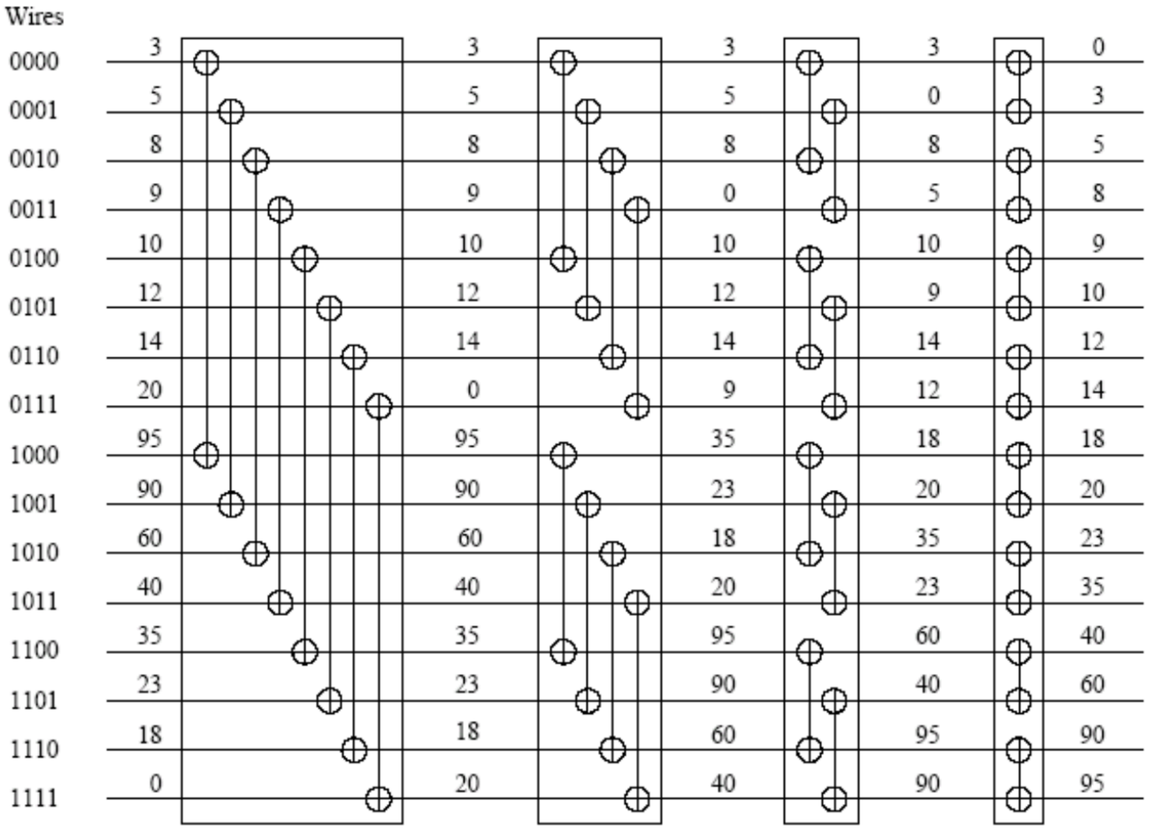
- **Sorting network can implement bitonic merge algorithm**
  - —***bitonic merging network***
- **Network structure**
  - —**$\log_2 n$ columns**
  - —**each column**
    - ***n/2* comparators**
    - **performs one step of the bitonic merge**
- **Bitonic merging network with *n* inputs: ⊕BM[n]**
  - —**produces an increasing sequence**
- **Replacing ⊕ comparators by ⊖ comparators: ⊖BM[n]**
  - —**produces a decreasing sequence**

Bitonic Merge Example



From Srinivas Aluru

Bitonic Merging Network



- **Input: bitonic sequence**
  - **— input wires are numbered 0,1,…, *n* - 1 (shown in binary)**
- **Output: sequence in sorted order**
- **Each column of comparators is drawn separately**

From John Mellor-Crummey

Bitonic Sort

How do we sort an unsorted sequence using a bitonic merge?

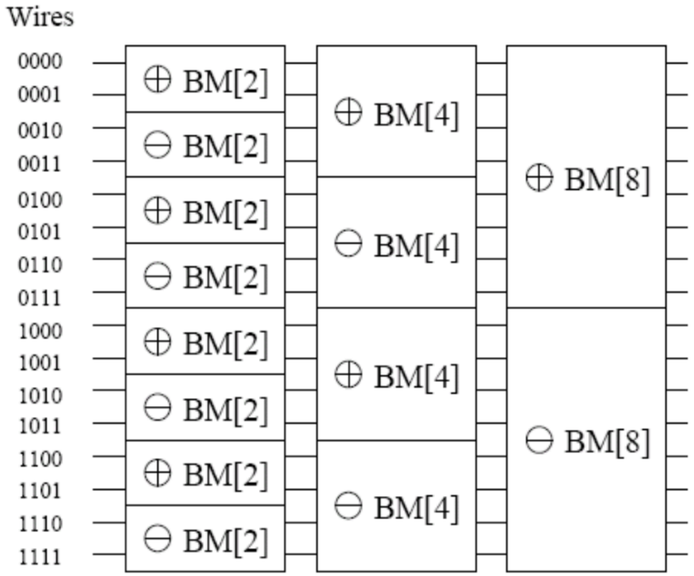
Two steps

1. Build a bitonic sequence

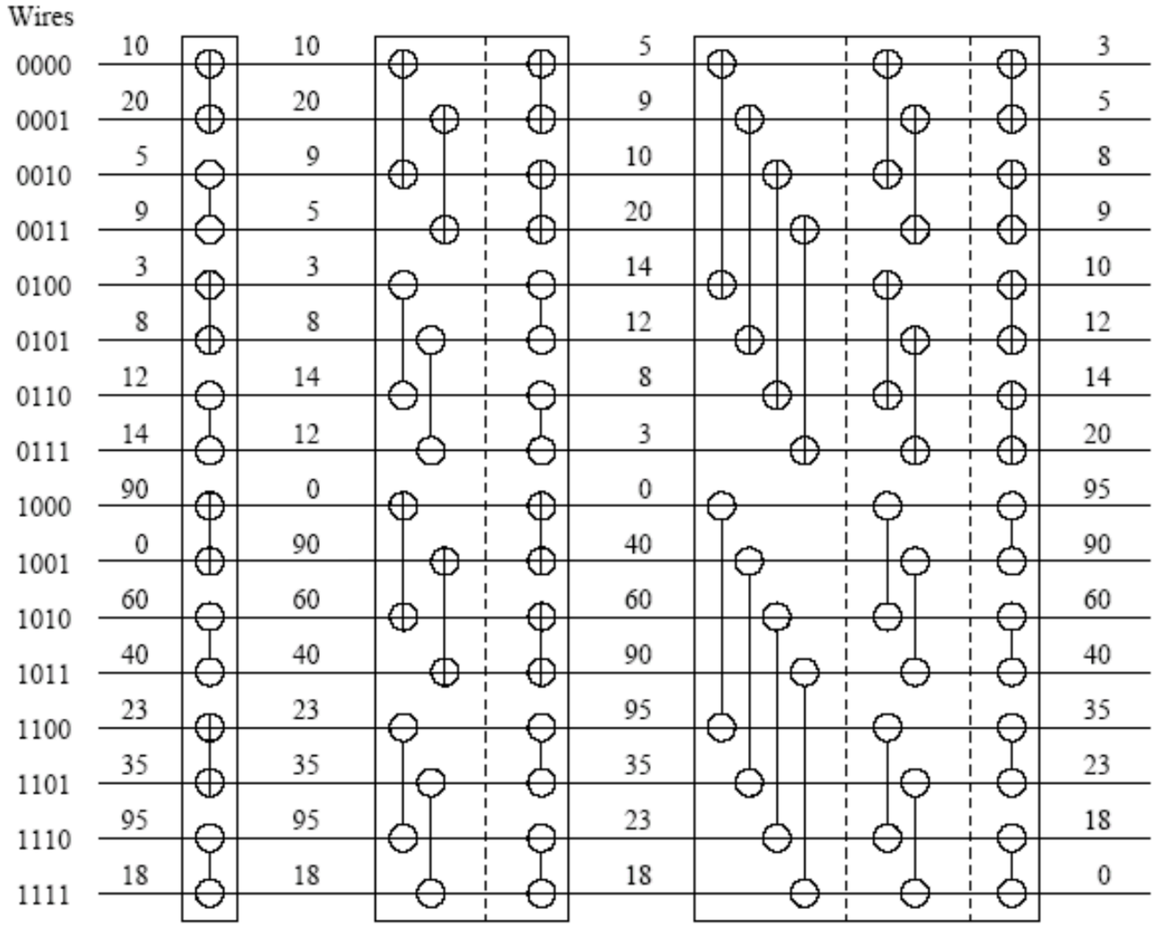
2. Sort it using a bitonic merging network

Building a Bitonic Sequence

- **Build a single bitonic sequence from the given sequence**
  - **—any sequence of length 2 is a bitonic sequence.**
  - **—build bitonic sequence of length 4**
    - **sort first two elements using ⊕BM[2]**
    - **sort next two using ⊖BM[2]**
- **Repeatedly merge to generate larger bitonic sequences**
  - **—⊕BM[*k*] & ⊖BM[*k*]: bitonic merging networks of size *k***



Building a Bitonic Sequence

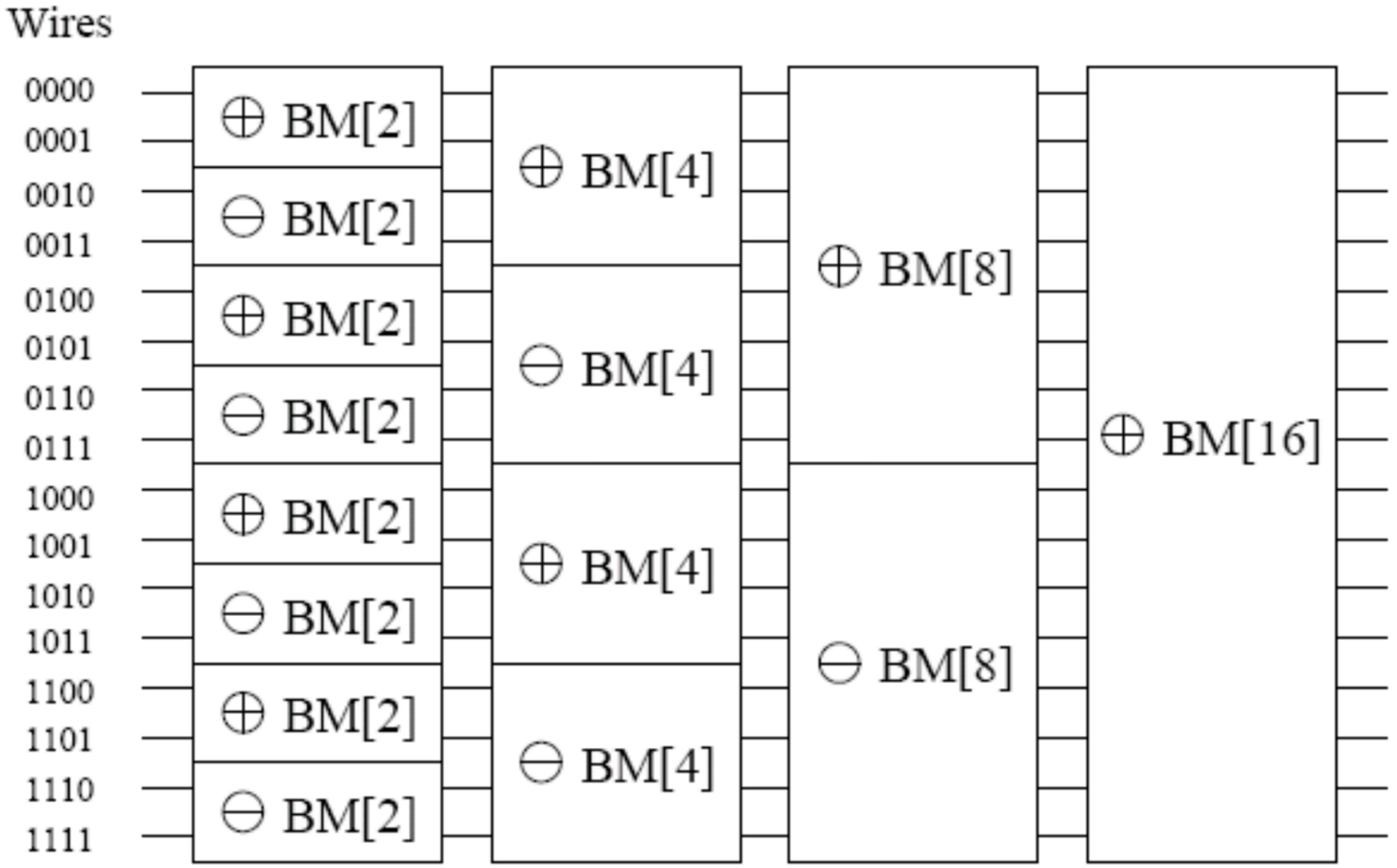


Input: sequence of 16 unordered numbers

Output: a bitonic sequence of 16 numbers

From John Mellor-Crummey

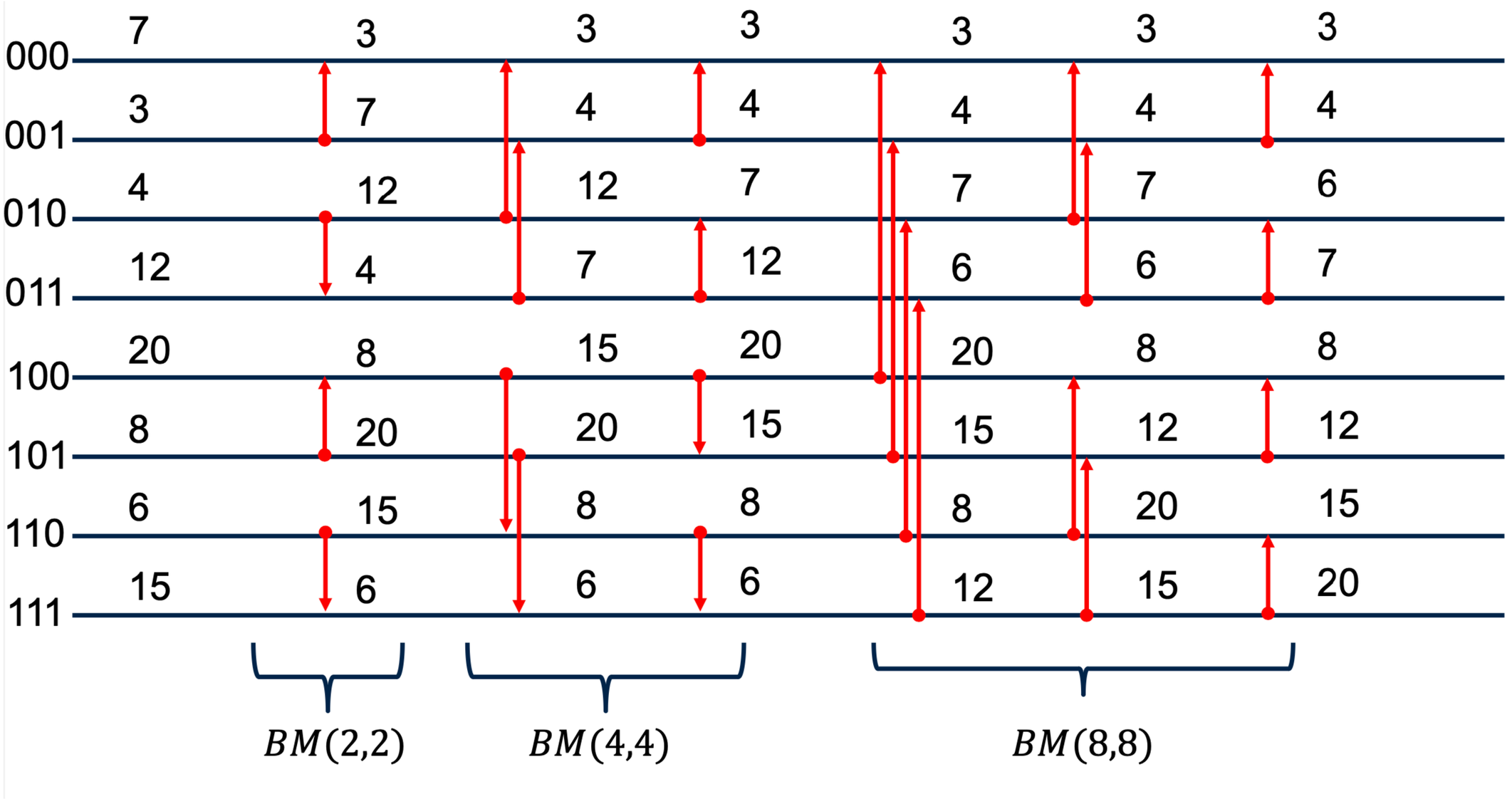
Bitonic Sort. n = 16



- **First 3 stages create bitonic sequence input to stage 4**
- **Last stage (⊕BM[16]) yields sorted sequence**

From John Mellor-Crummey

Bitonic Sort Example



From Srinivas Aluru

Complexity of Bitonic Sorting Networks

- **Depth of the network is** $\Theta(\log^2 n)$

  BM(p, p) = $O(\lg^2 p)$

  - **$\log_2 n$ merge stages**
  - **j^{th} merge stage depth is $\log_2 2^j = j$**
  - **depth =** $\sum_{j=1}^{\log_2 n} \log_2 2^j = \sum_{i=1}^{\log_2 n} j = (\log_2 n + 1)(\log_2 n)/2 = \theta(\log^2 n)$
- **Each stage of the network contains *n*/2 comparators**
- **Complexity of serial implementation =** $\Theta(n \log^2 n)$

Bitonic Sort Pseudocode

```
for i=0 to (log p)-1 do
  for j=i downto 0 do
     if (i+1)st bit is 0 then
        Compare_Exchange↑ with (Rank XOR 2^j)
     else
        CompareExchange↓ with (Rank XOR 2^j)
  endfor
endfor
```

Generalizing Bitonic Sort

- $T(p,p) = O(\log^2 p)$
- Computation time = $O(\log^2 p)$
- Comm. Time = $O(\tau \log^2 p + \mu \log^2 p)$

- $\boldsymbol{n > p?}$
- Step 1: Sort n/p local elements.
- Step 2: Run bitonic sort with `CompareExchange` replaced by `MergeSplit`
- Computation time = $O\left(\frac{n}{p}\log\frac{n}{p} + \frac{n}{p}\log^2 p\right)$
- Comm. Time = $O\left(\tau \log^2 p + \mu \frac{n}{p}\log^2 p\right)$

From Srinivas Aluru