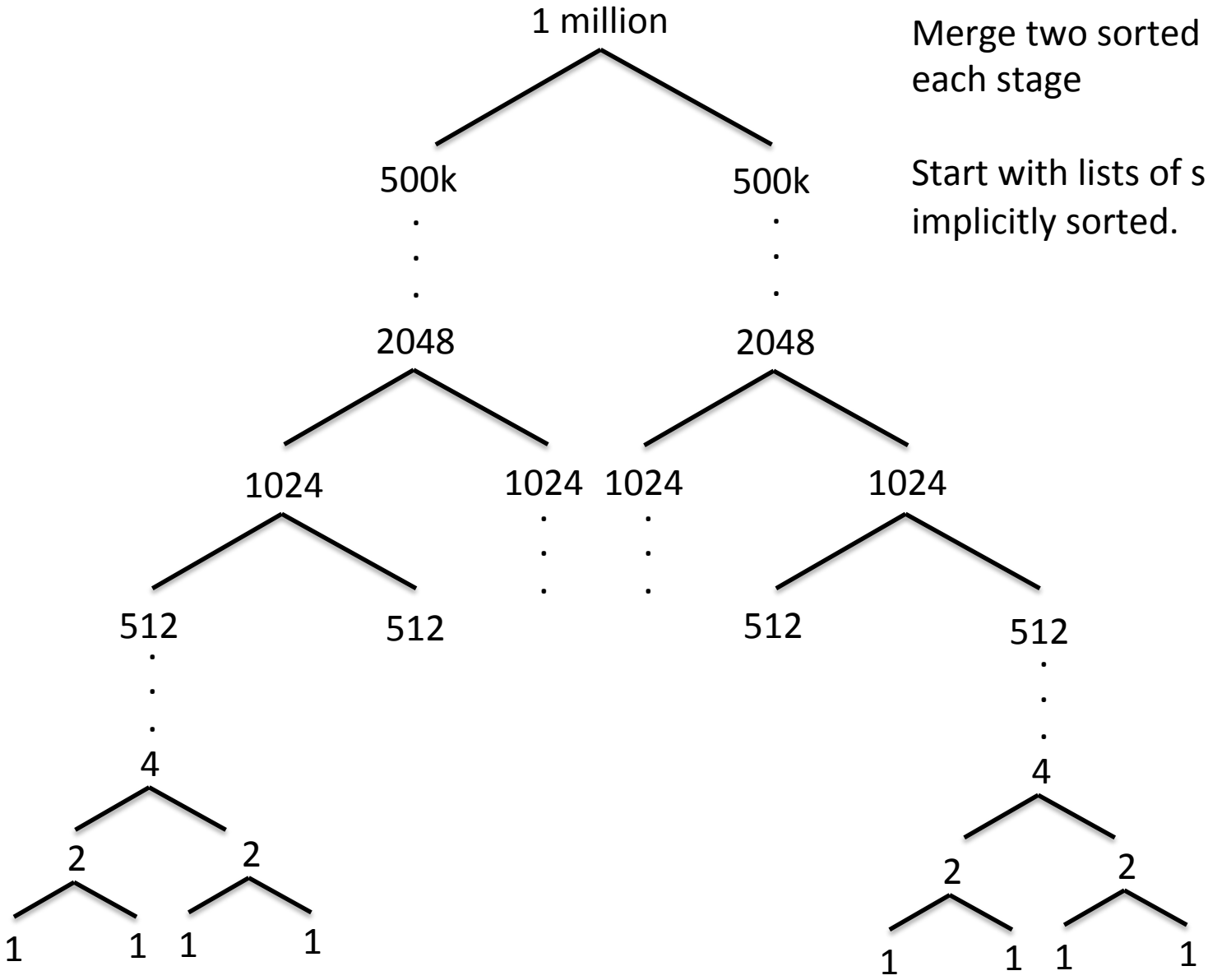
Sorting on GPU

Bryan Mills, PhD

Spring 2017

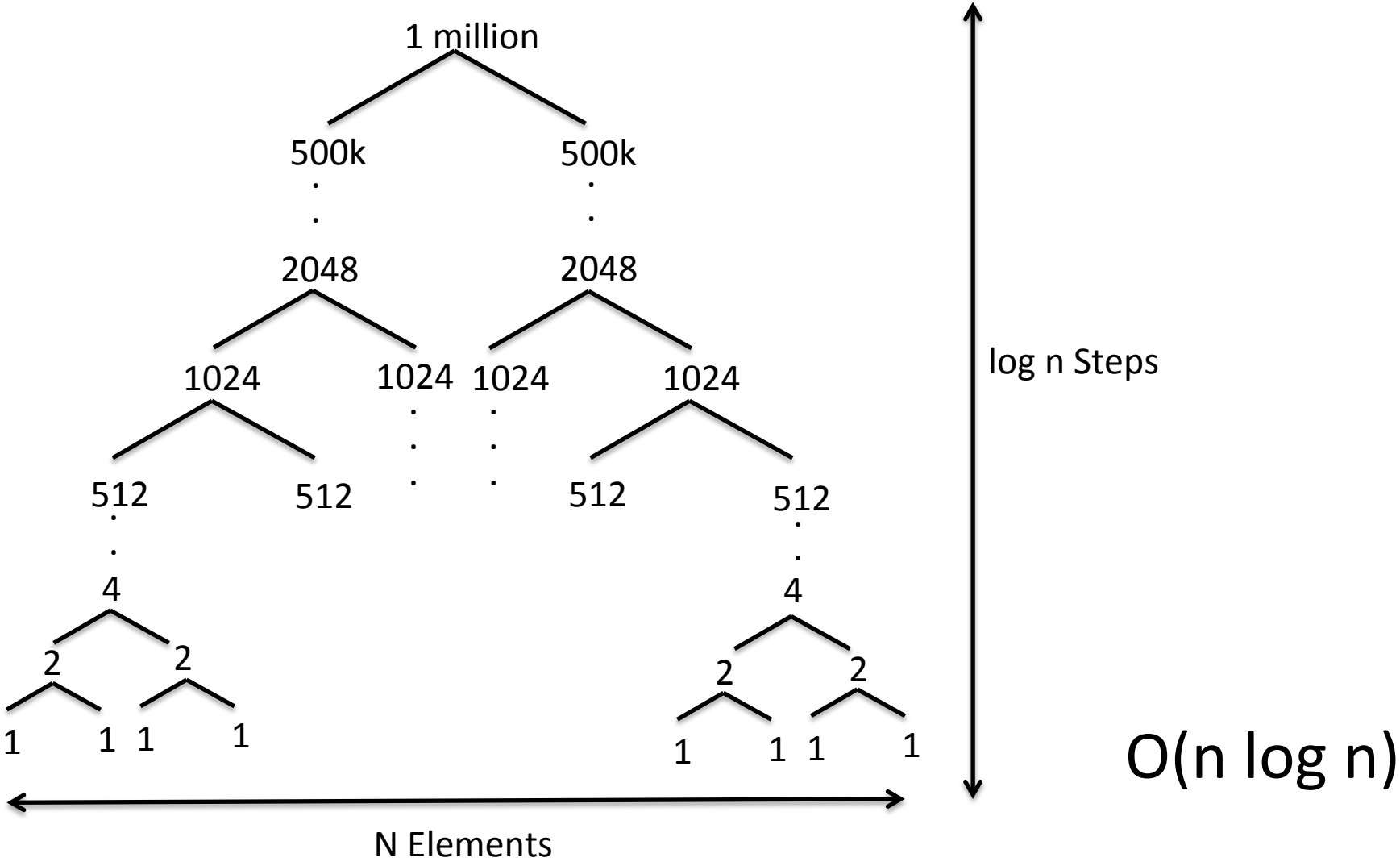
Merge Sort



Merge two sorted lists at each stage

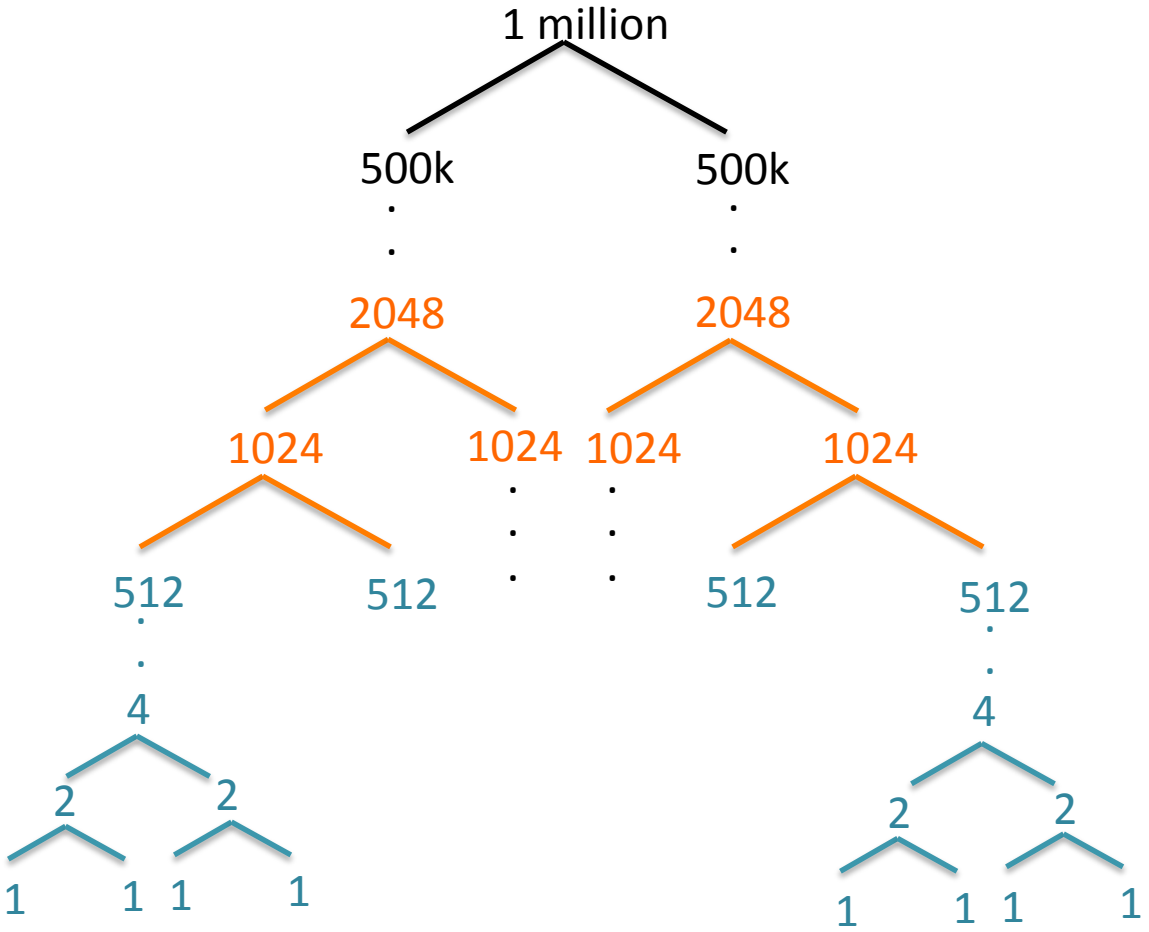
Start with lists of size 1, implicitly sorted.

Performance of Merge Sort



O(n log n)

Merge Sort using GPU

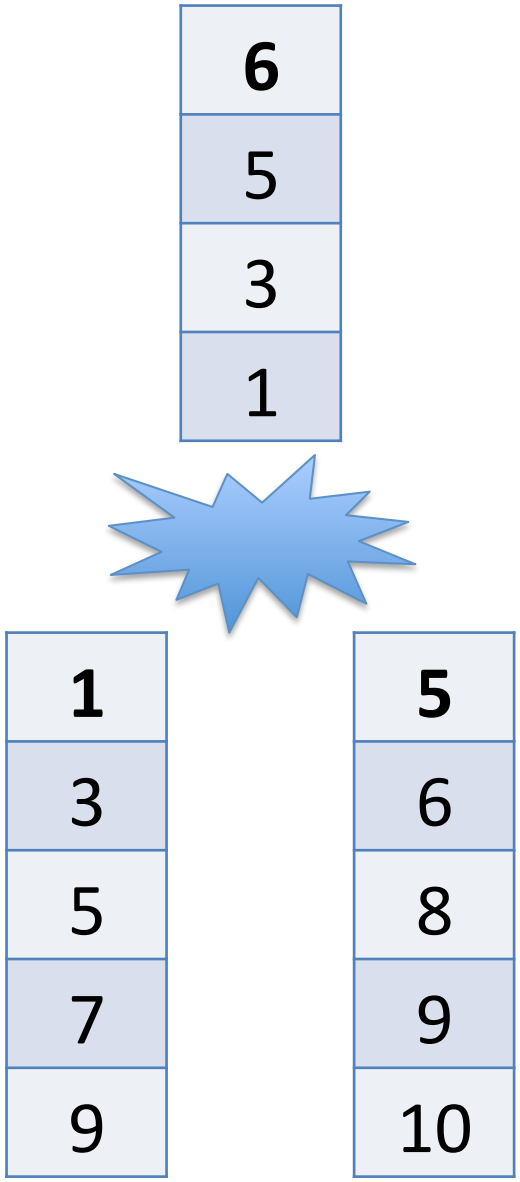


Stage 3
Single or small number of tasks
Very large task

Stage 2
Bunch of tasks
Each task is medium
Task per block

Stage 1
Lots of little merge tasks
Each task is small
Use one per thread

Serial Merge



Hard to break up task?
Need one person to look at both lists.

Recall Parallel Compact

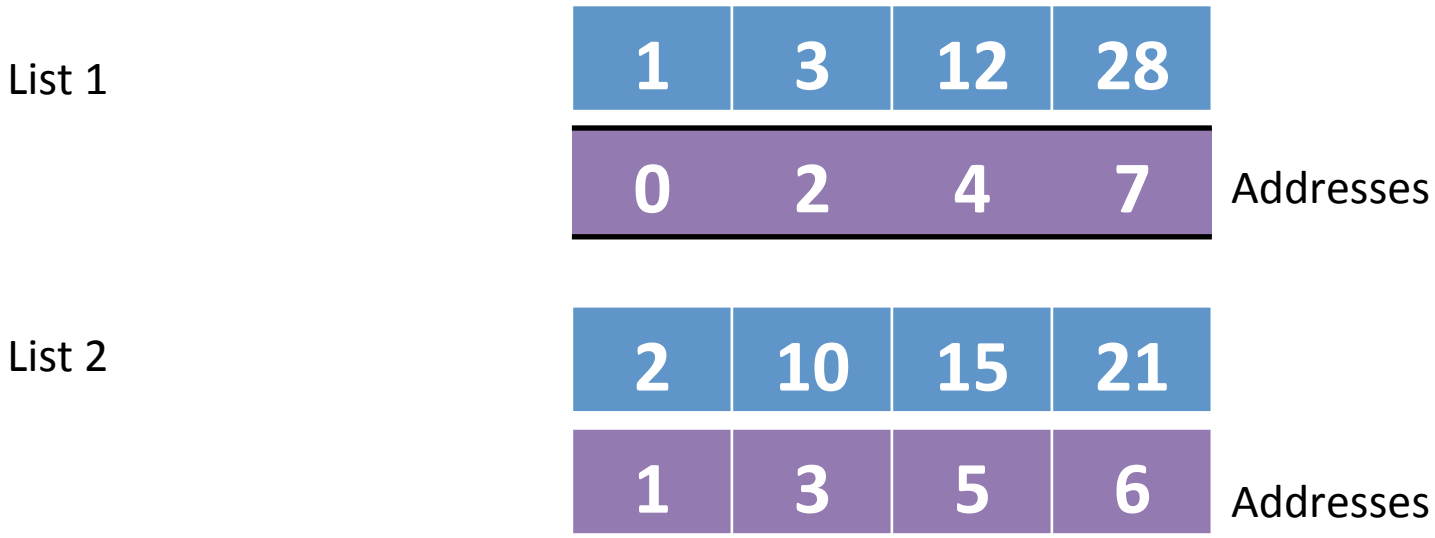
Parallel Merge

List 1

List 2

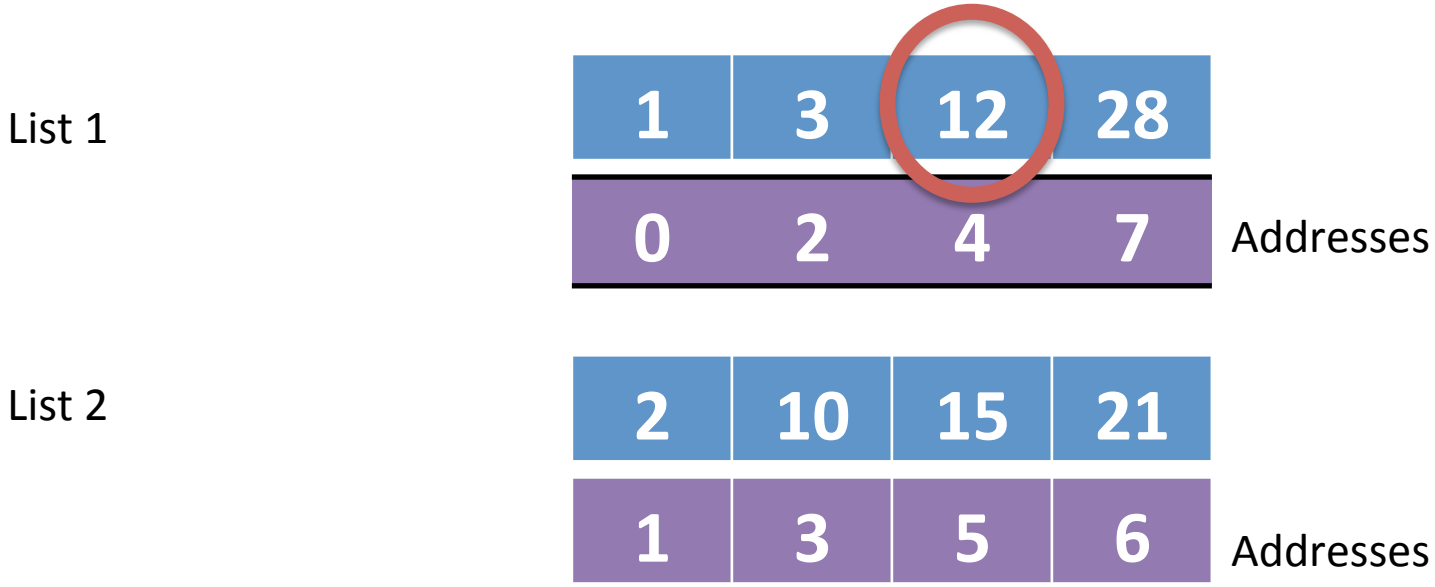
Where would each element be in the sorted list?

Parallel Merge



Assigning each thread an element how could they determine where to place the element in the final list?

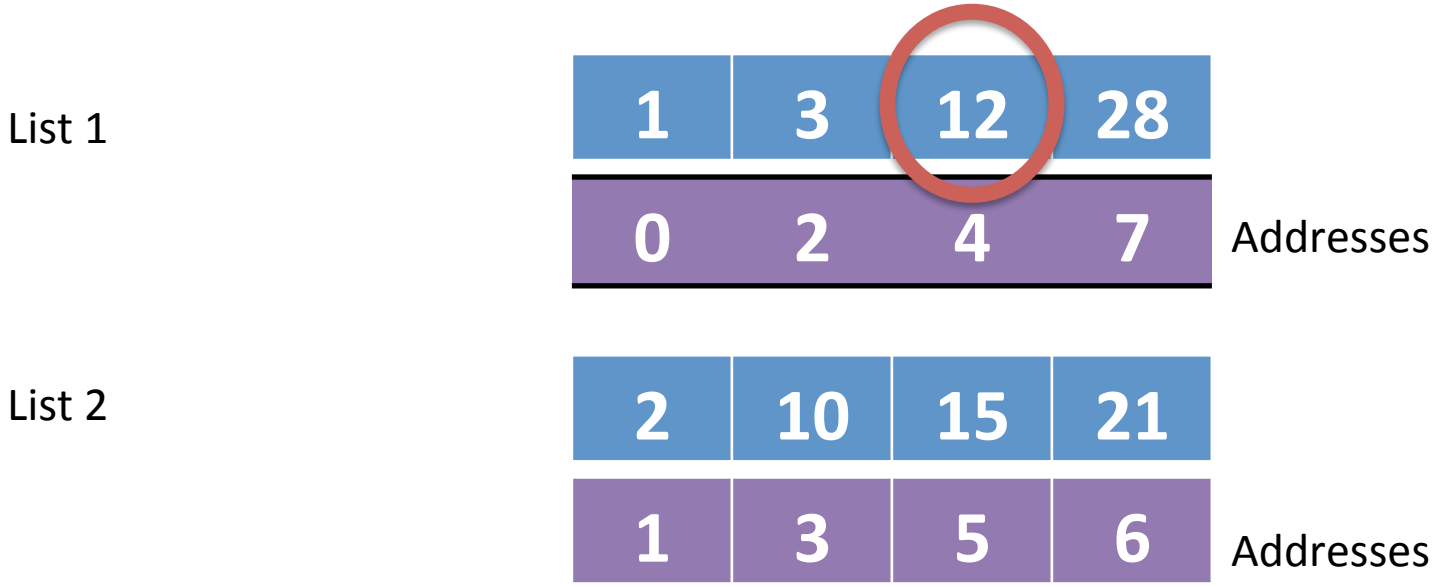
Parallel Merge



Lets look at one single element (12)

- Where is it in its own list?
- Where would it be in the other list?

Parallel Merge

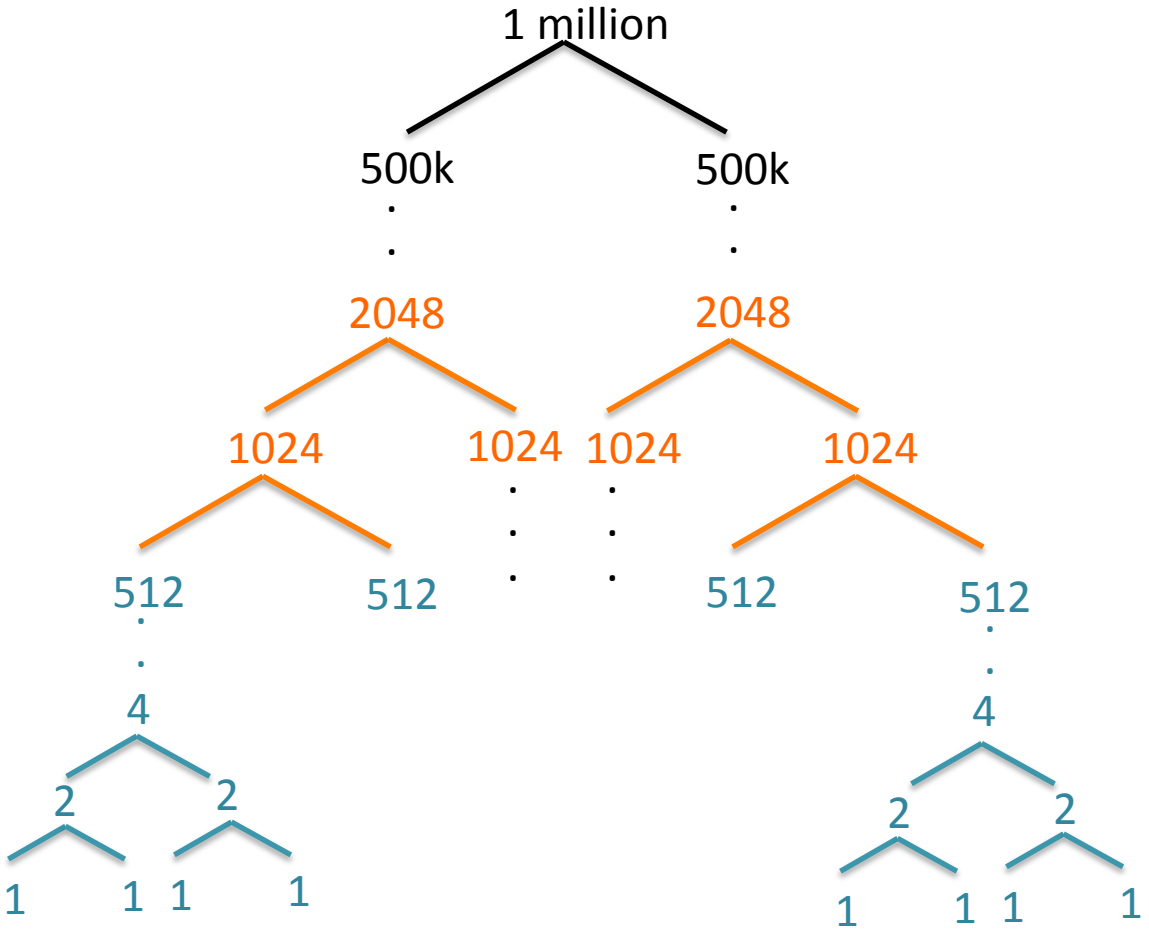


Lets look at one single element (12)

- Where is it in its own list? thread index
- Where would it be in the other list? binary search

Add those together (2 + 2) = 4 (my address!)

Merge Sort using GPU



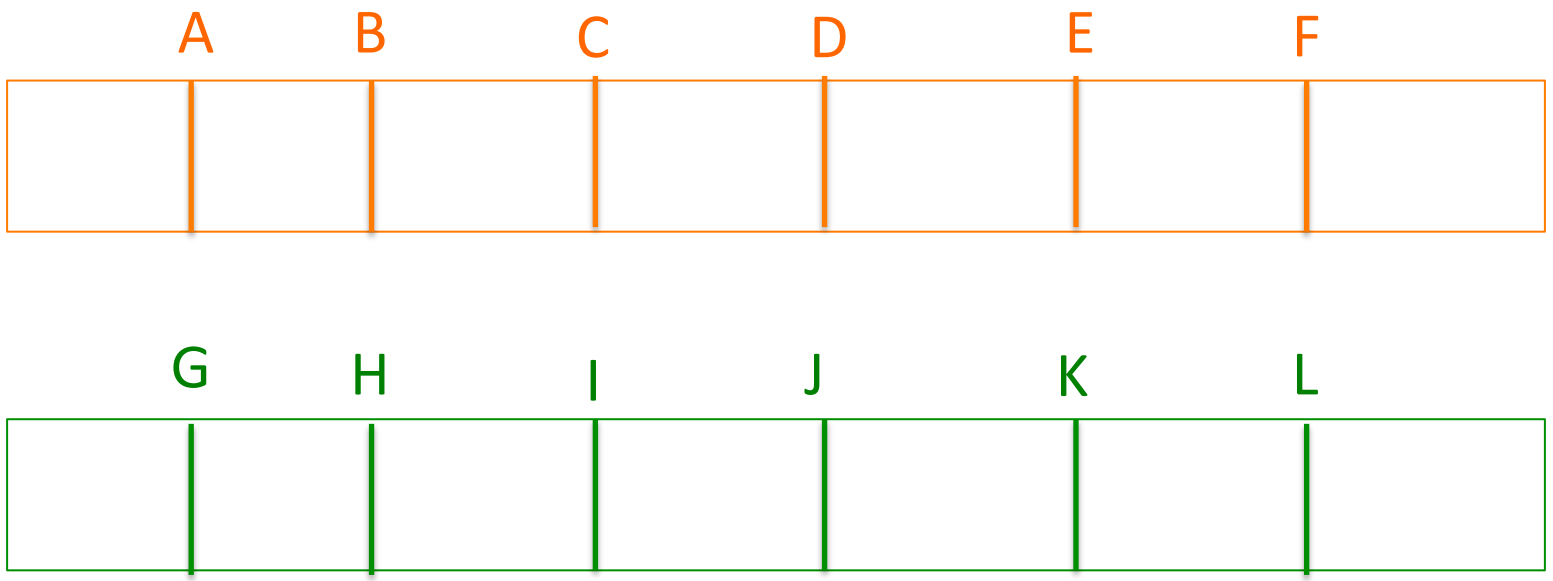
Stage 3
Single or small number of tasks
Very large task
HOW TO BREAK THIS UP?

Stage 2
Bunch of tasks
Each task is medium
Task per block

Stage 1
Lots of little merge tasks
Each task is small
Use one per thread

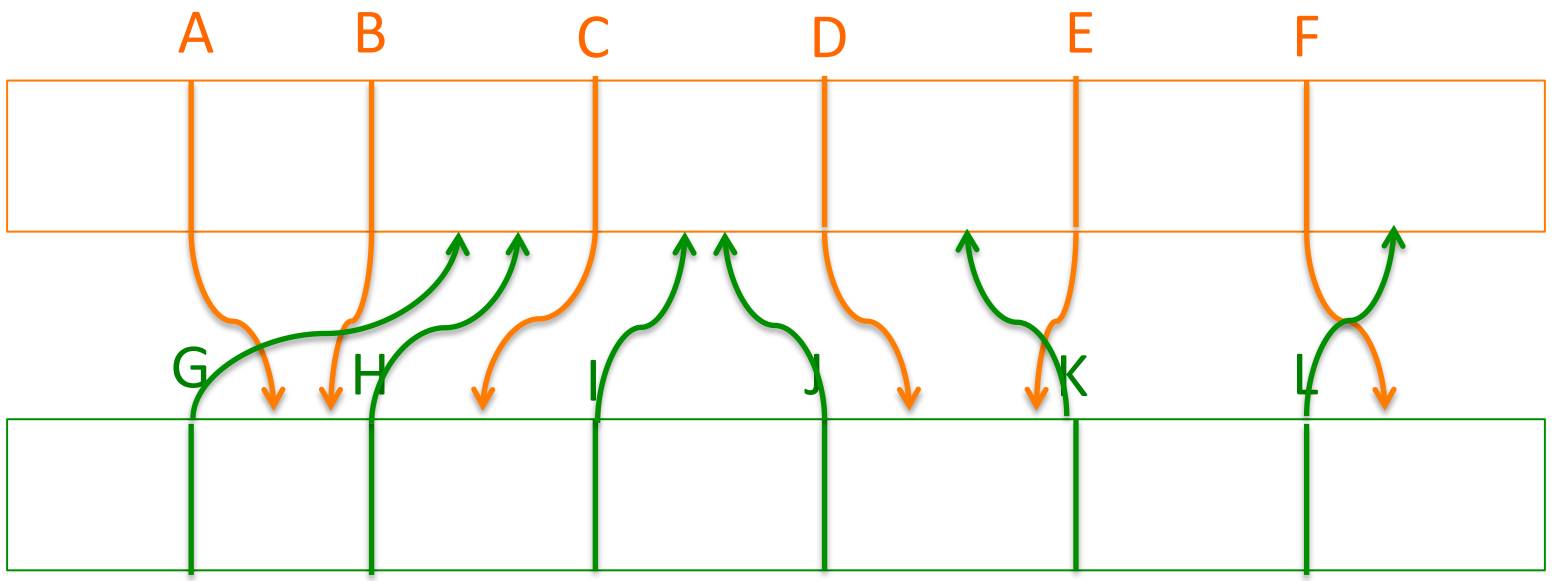
How to Merge BIG Lists

How to Merge BIG Lists



- Pick “splitter” elements from both large lists
  - For example every 256th element

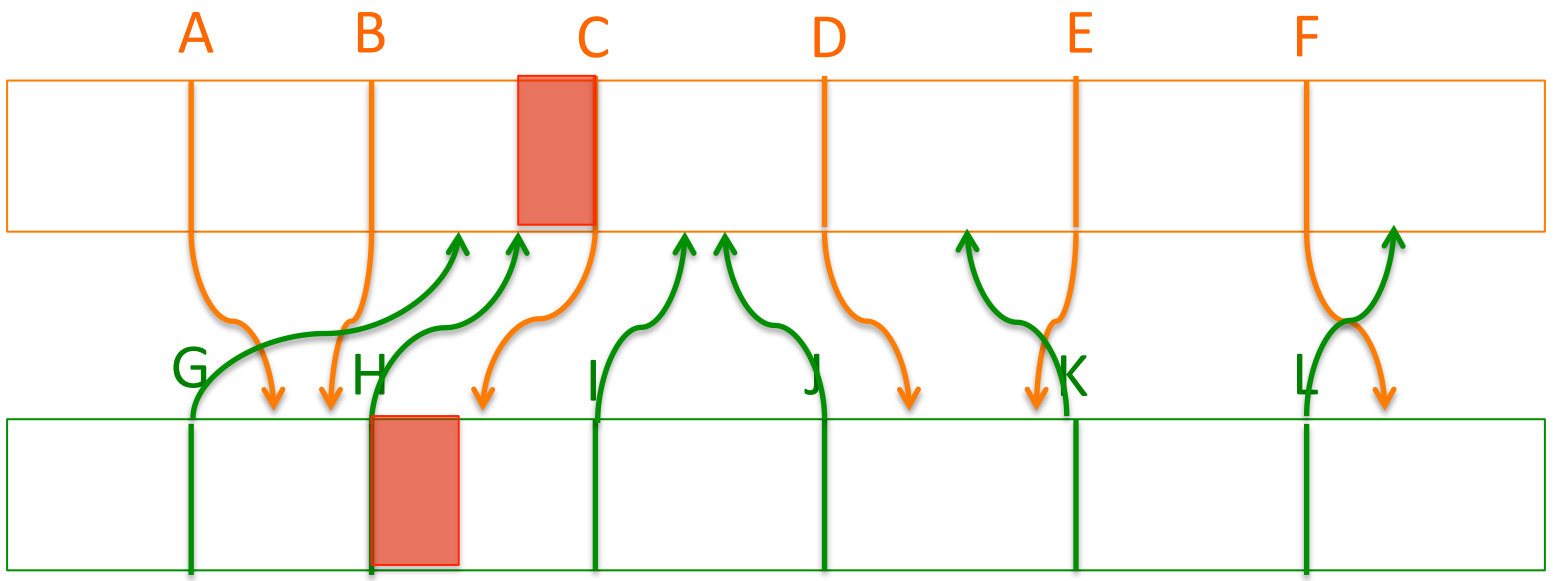
How to Merge BIG Lists



- Merge these two subsets of elements

ABGHCIJDKELF

How to Merge BIG Lists

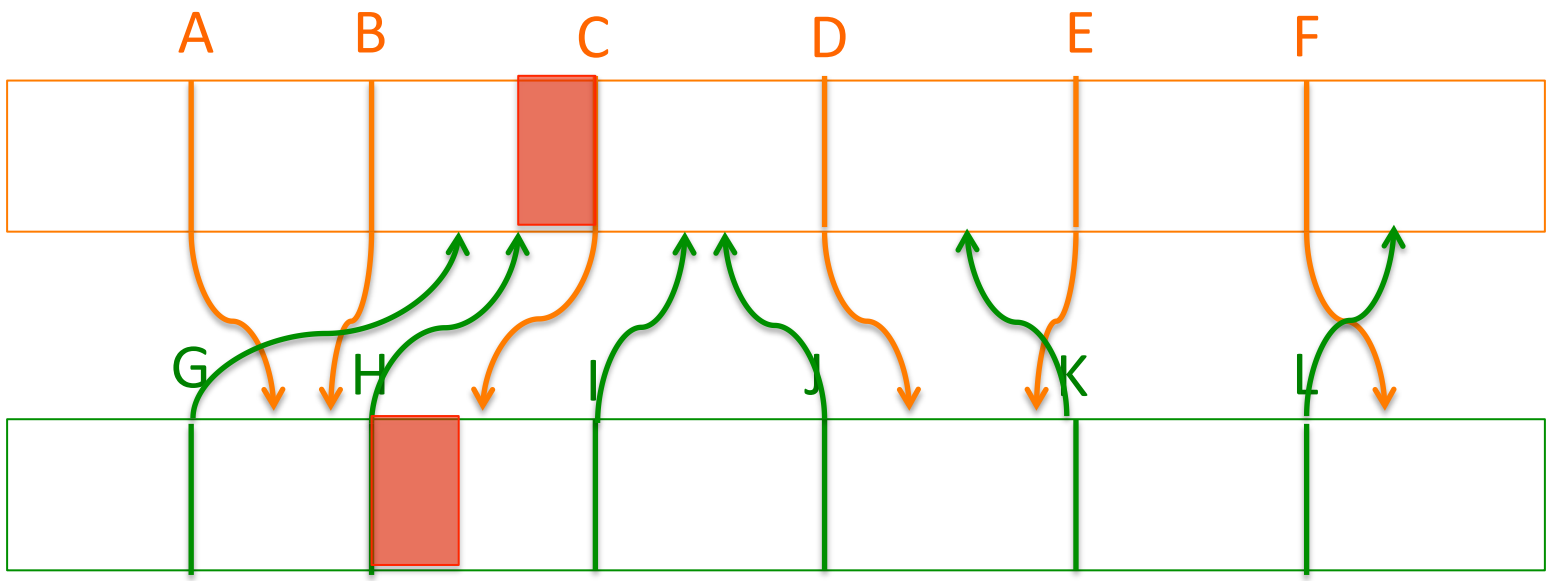


- Pick the subsets between any two splitters and divide that work up and send to multiple processors

ABGHCIJDKELF

Note the number of elements between any two of these can be no greater than 2x the splitter size. In our case 2*256 = 512

How to Merge BIG Lists



- Pick the subsets between any two splitters and divide that work up and send to multiple processors

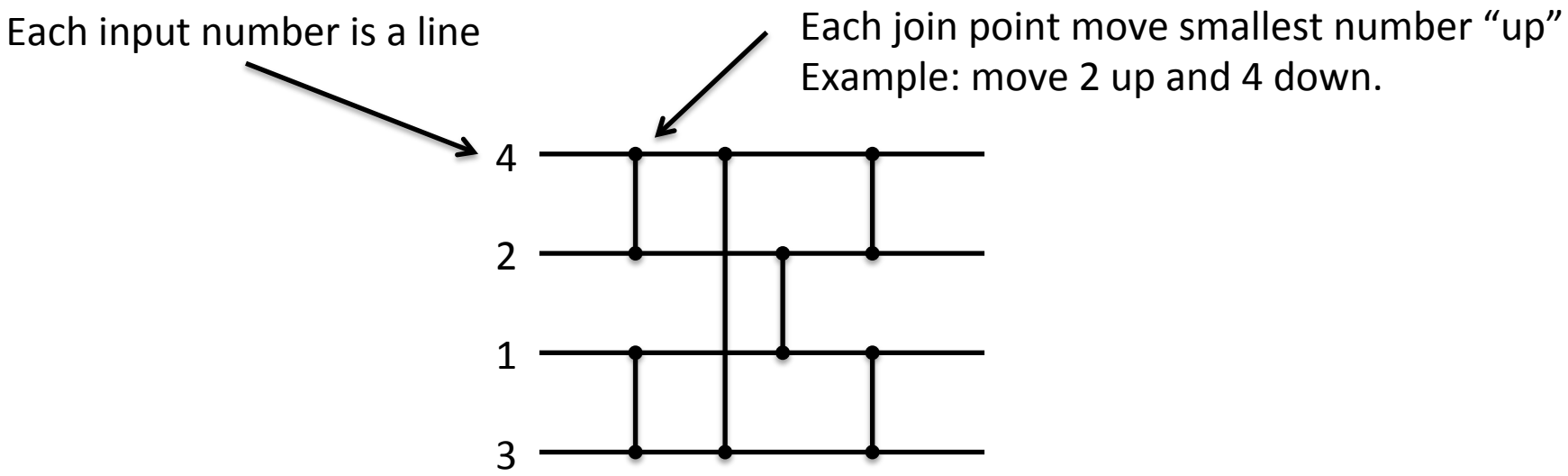
ABGHCIJDKELF

Bitonic Sort

- Sorting Networks
- Example in Class
- Even-Odd Sorting

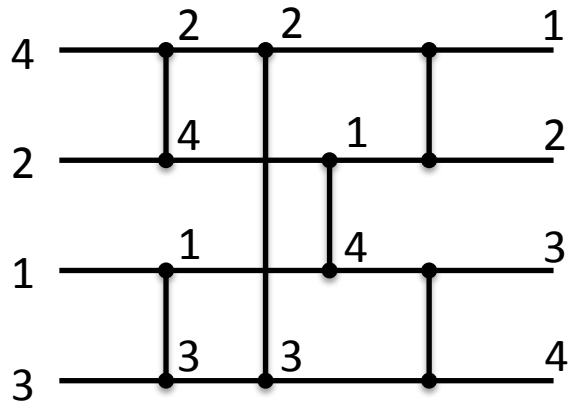
Sorting Networks

- Oblivious sorting algorithm
  - Regardless of input the steps to produce sorted output is always the same



Sorting Networks

- Walk thru this example.

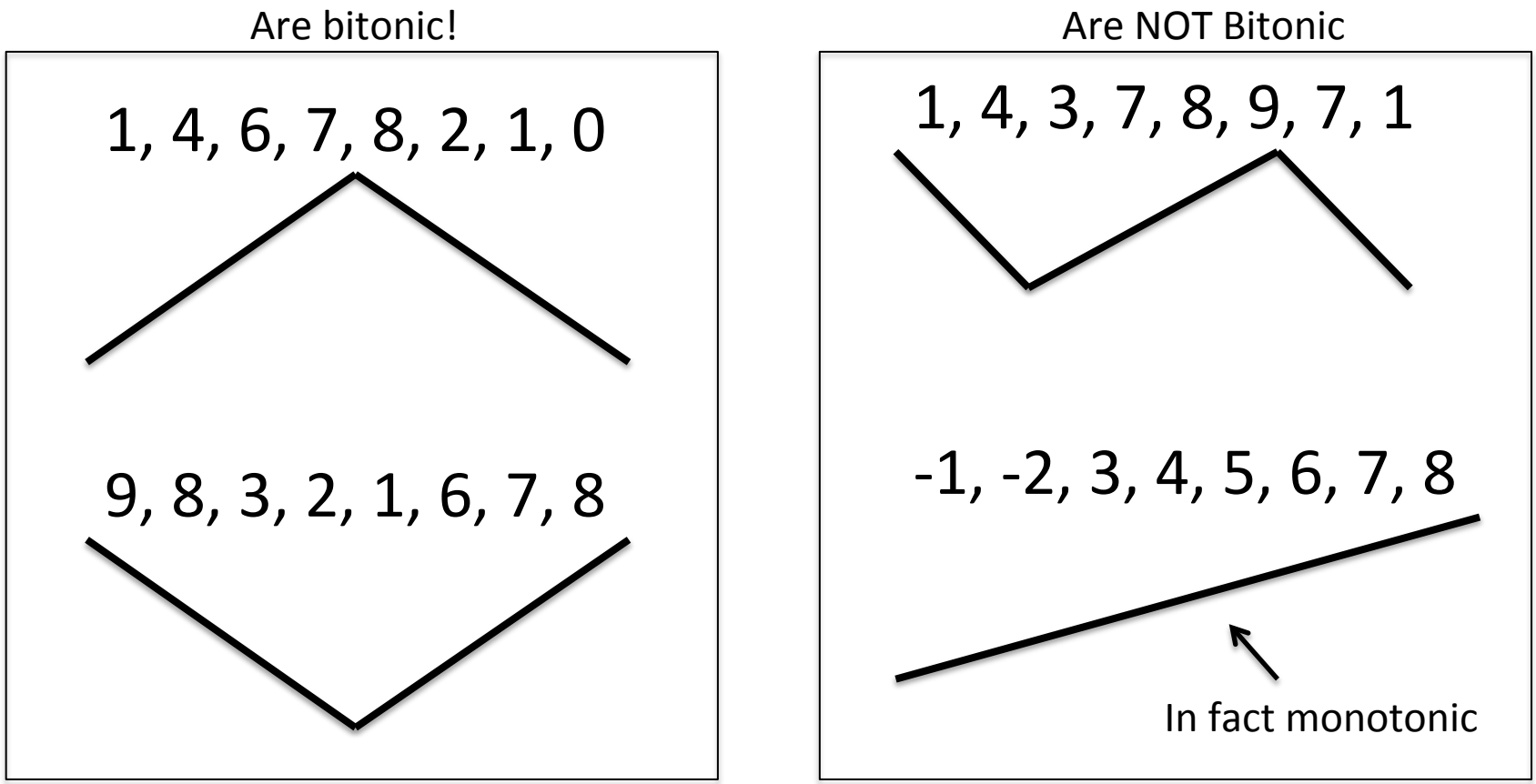


Bitonic Sorting

- How to build this network?
  - Bitonic sorting is one such method
  - Previously in class we saw even-odd sorting, which could also be adapted to sorting network

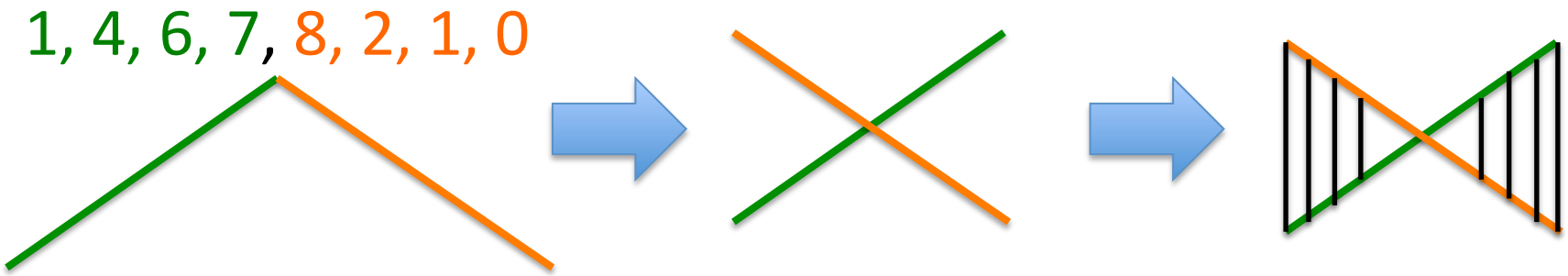
What is a bitonic sequence?

- Series which has one inflection point



Sorting Bitonic Sequence

- Overlay and compare pair-wise

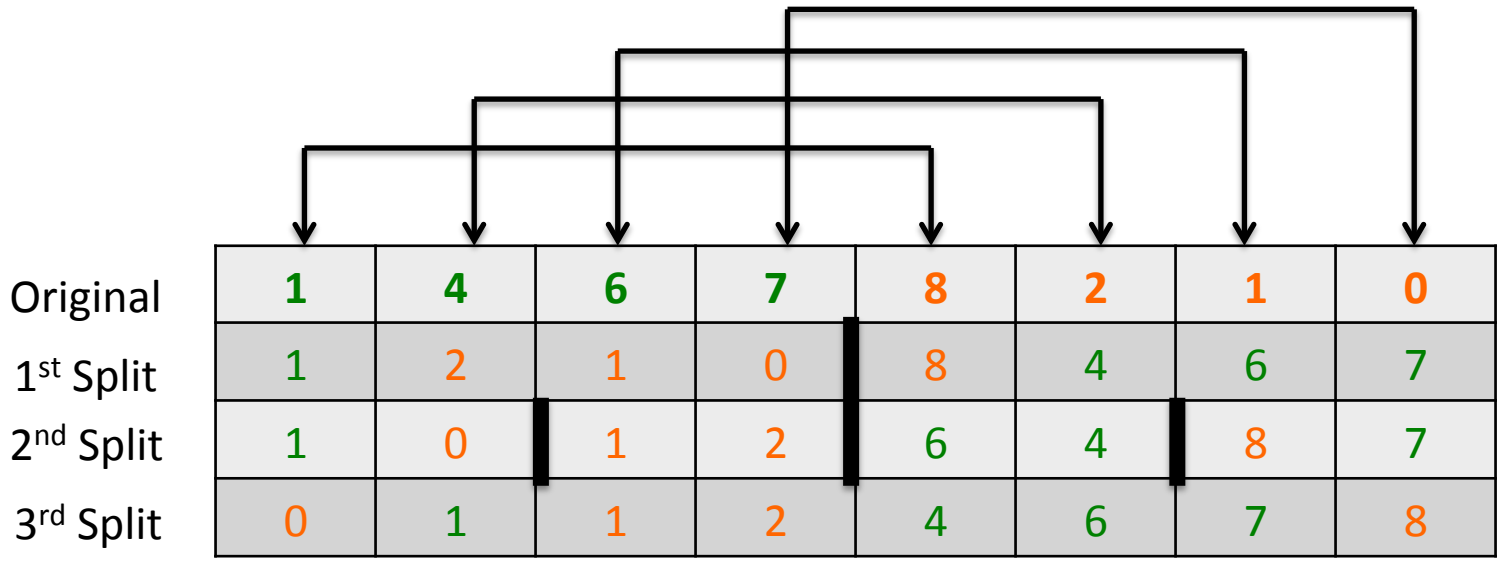


- Splitting list into high and low

- Recurse on both halves to produce sorted list

Sorting Bitonic Sequence Example

- Overlay and compare pair-wise, swapping lower to the right



Bitonic Sort

- Once we have to bitonic sequences we can then easily sort them using a network
- Note that any pair of numbers is bitonic
  - <2,3> <5,6> <9,2> even <1,1>
- Two bitonic sequences placed in the “opposite” order will create another bitonic sequence
  - <2,3,5,6> <5,6,9,2> <2,3,9,2> <9,2,2,3>

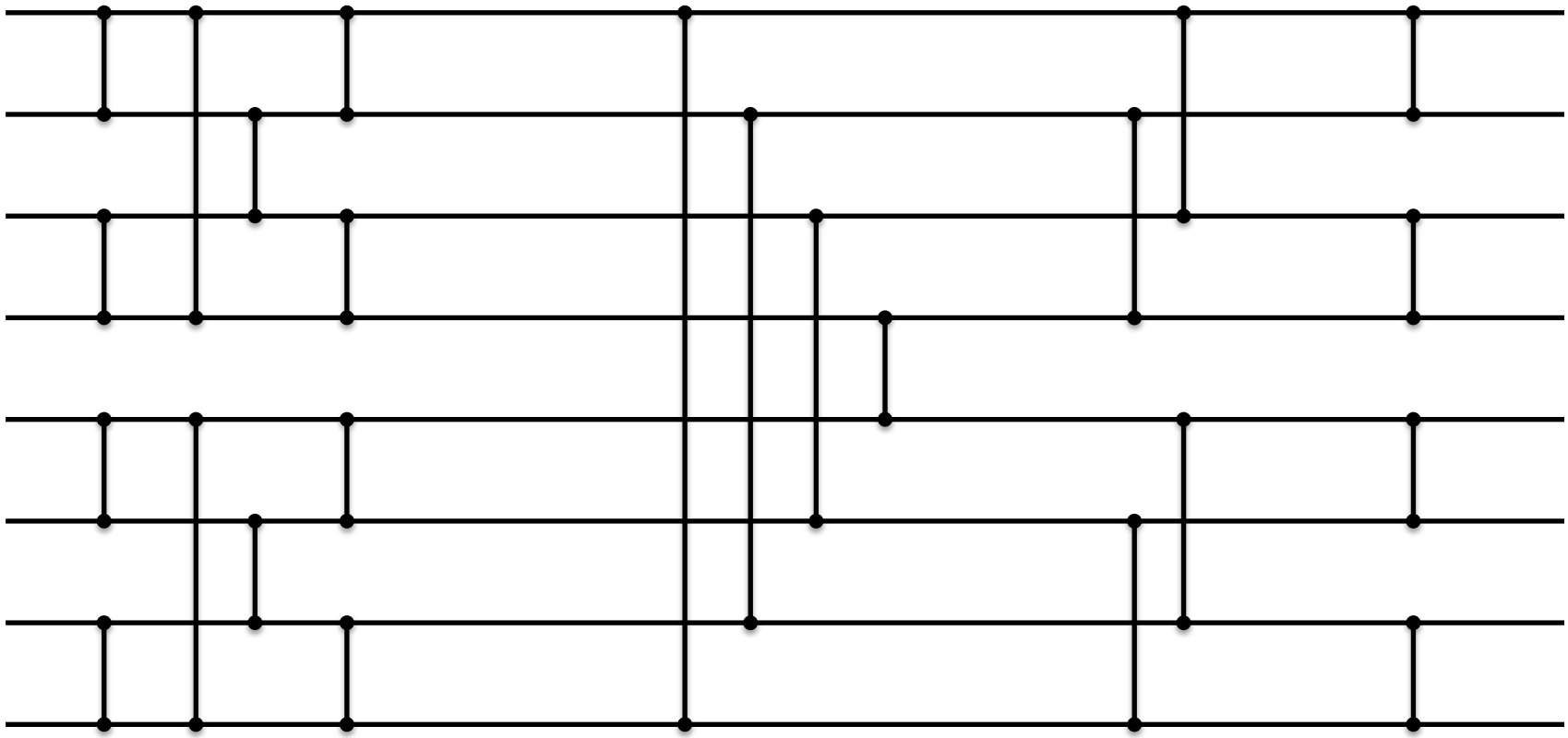
Bitonic Sort

- Two steps:
  - Split input into two bitonic sequences
  - Sort bitonic sequences

8 input Bitonic Sort

Create 2 Bitonic Sequences

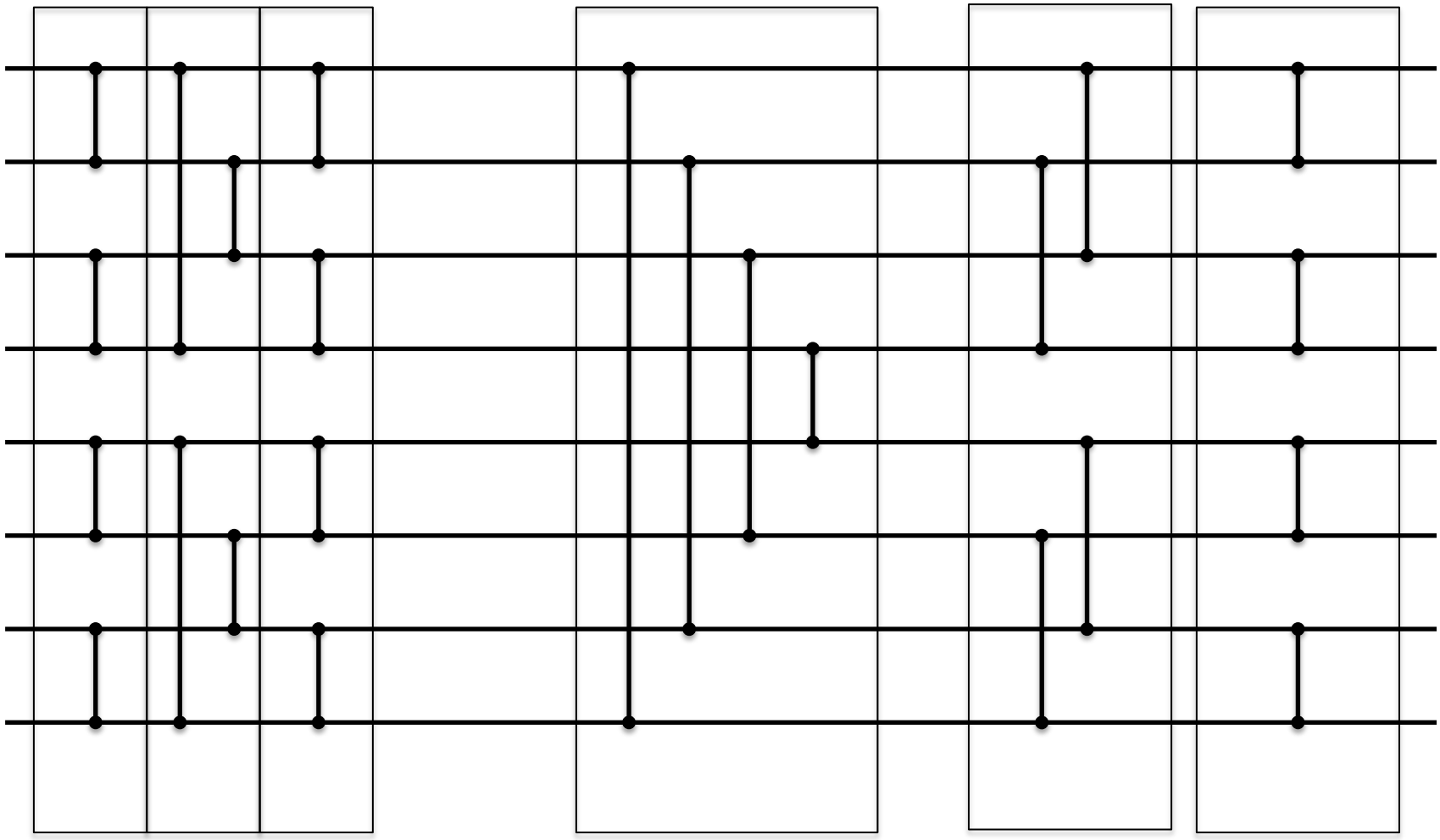
Sort Bitonic Sequence



Bitonic Sort

- Note that the lines can be done in parallel
- Implemented in GPU as a series of steps at each step one thread keeps either the smallest or largest element, depending on step.

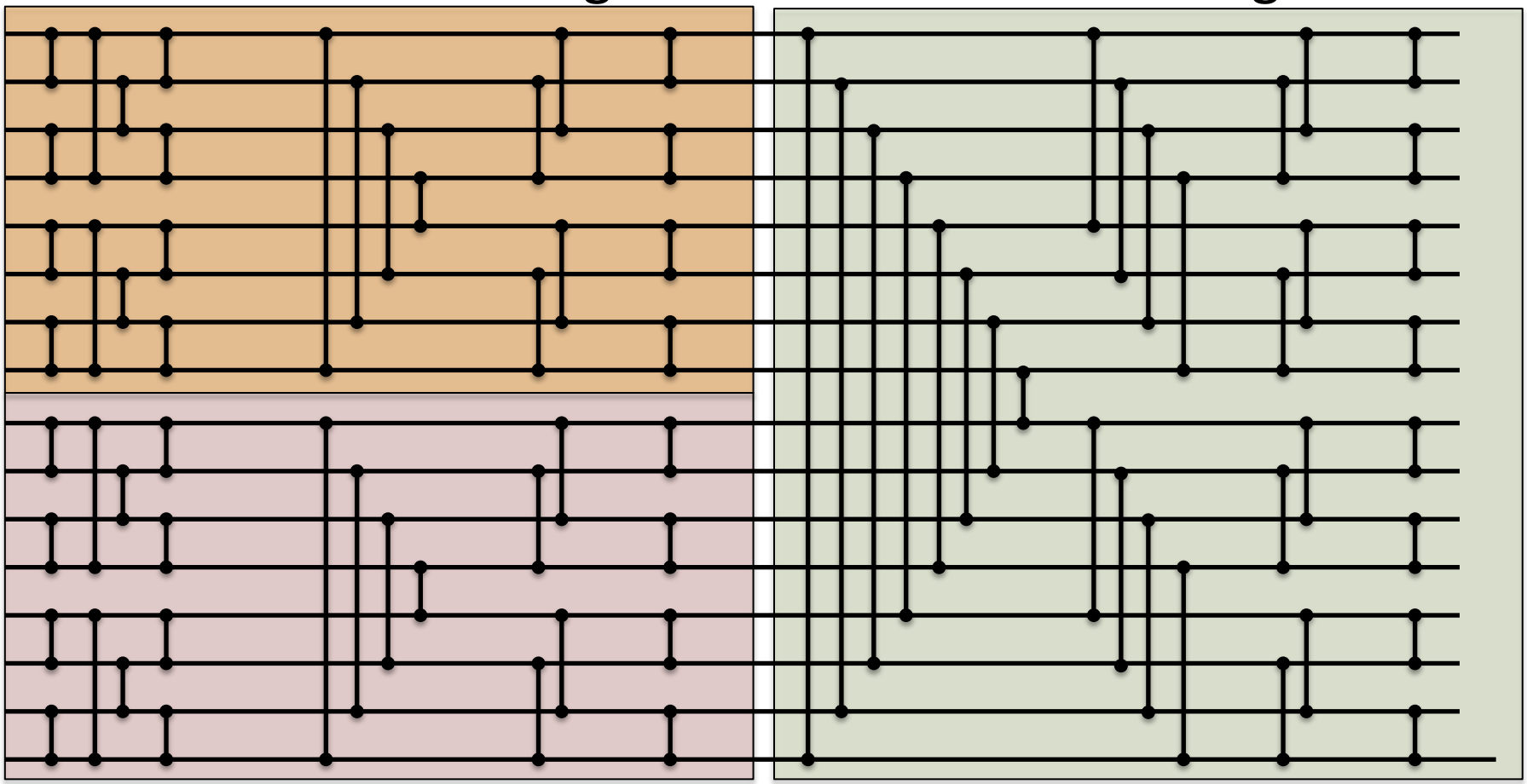
8 input Bitonic Sort



Each block can be done in parallel

Bitonic Sort

- Increase the size by connecting smaller sorting networks and running the recursive bitonic merge.



Radix Sort

- Sort by looking at individual bits in the numbers
- Start at least significant bit, group maintaining order, move to second, etc…

Decimal	Binary
0	0 0 0
5	1 0 1
2	0 1 0
7	1 1 1
1	0 0 1
3	0 1 1
6	1 1 0
4	1 0 0

Radix Sort

- Sort by looking at individual bits in the numbers
- Start at least significant bit, group maintaining order, move to second, etc…

Decimal	Binary	1st Pass
0	0 0 0	0 0 0
5	1 0 1	0 1 0
2	0 1 0	1 1 0
7	1 1 1	1 0 0
1	0 0 1	1 0 1
3	0 1 1	1 1 1
6	1 1 0	0 0 1
4	1 0 0	0 1 1

Sorting by least significant bit
Maintaining original order

Radix Sort

- Sort by looking at individual bits in the numbers
- Start at least significant bit, group maintaining order, move to second, etc…

Decimal	Binary	1st Pass	2nd Pass
0	0 0 0	0 0 0	0 0 0
5	1 0 1	0 1 0	1 0 0
2	0 1 0	1 1 0	1 0 1
7	1 1 1	1 0 0	0 0 1
1	0 0 1	1 0 1	0 1 0
3	0 1 1	1 1 1	1 1 0
6	1 1 0	0 0 1	1 1 1
4	1 0 0	0 1 1	0 1 1

Sorting by 2nd least significant bit
Maintaining order from previous step

Radix Sort

- Sort by looking at individual bits in the numbers
- Start at least significant bit, group maintaining order, move to second, etc…

Decimal	Binary	1st Pass	2nd Pass	3rd Pass
0	0 0 0	0 0 0	0 0 0	0 0 0
5	1 0 1	0 1 0	1 0 0	0 0 1
2	0 1 0	1 1 0	1 0 1	0 1 0
7	1 1 1	1 0 0	0 0 1	0 1 1
1	0 0 1	1 0 1	0 1 0	1 0 0
3	0 1 1	1 1 1	1 1 0	1 0 1
6	1 1 0	0 0 1	1 1 1	1 1 0
4	1 0 0	0 1 1	0 1 1	1 1 1

Sorting by 3nd least significant bit

Maintaining order from previous step

Radix Sort

- Sort by looking at individual bits in the numbers
- Start at least significant bit, group maintaining order, move to second, etc…

Decimal	Binary	1st Pass	2nd Pass	3rd Pass	SORTED!
0	0 0 0	0 0 0	0 0 0	0 0 0	0
5	1 0 1	0 1 0	1 0 0	0 0 1	1
2	0 1 0	1 1 0	1 0 1	0 1 0	2
7	1 1 1	1 0 0	0 0 1	0 1 1	3
1	0 0 1	1 0 1	0 1 0	1 0 0	4
3	0 1 1	1 1 1	1 1 0	1 0 1	5
6	1 1 0	0 0 1	1 1 1	1 1 0	6
4	1 0 0	0 1 1	0 1 1	1 1 1	7

Radix Sort

- Sort by looking at individual bits in the numbers
- Start at least significant bit, group maintaining order, move to second, etc…

Decimal	Binary	1st Pass
0	0 0 0	0 0 0
5	1 0 1	0 1 0
2	0 1 0	1 1 0
7	1 1 1	1 0 0
1	0 0 1	1 0 1
3	0 1 1	1 1 1
6	1 1 0	0 0 1
4	1 0 0	0 1 1

What parallel operation have we seen that can take a group of numbers and product a subset?

Compact!

Radix Sort

- O(kn)
  - Where k is the number of digits
  - Recall that compact is O(n log n)
- Currently one of the fastest sorting algorithms on GPUS