Lecture 28: Sorting

COSC 273: Parallel and Distributed Computing

Spring 2023

This Week

sub link soon

- Homework 03 Due Today
- Final Project group & topic selection, also due Today
  - Option 1: computing prime numbers
  - Option 2: sorting
  - Option 3: choose your own adventure

Sorting

The Task

Input. A (large) array `float[] values`

Task. Modify `values` so that the values are increasing.

Scale. `values.length = 1_048_576` ($= 2^{20}$)

Baseline. `Arrays.sort(values)` ← Java built-in

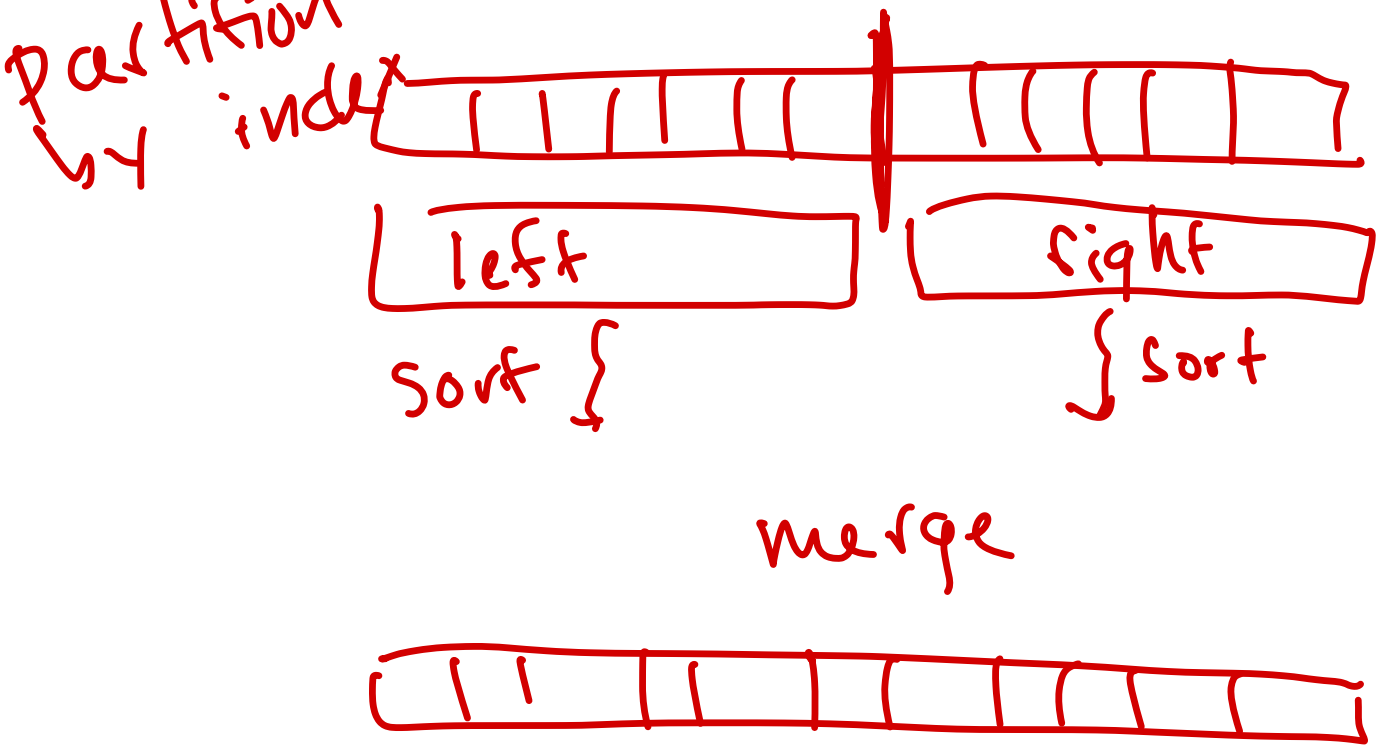
- `include java.util.Arrays`

Sorting by Divide-and-Conquer

General Strategy:

- break problem up into smaller sub-tasks
- solve sub-tasks
- combine sub-solutions to get total solution

Question 1. How to divide sorting problem?



Bubble Sort $\Theta(n^2)$

1,000

want to sort

Alt. Part. by value

Quick Sort

Sorting by Divide-and-Conquer

General Strategy:

- break problem up into smaller sub-tasks
- solve sub-tasks
- combine sub-solutions to get total solution

Question 1. How to divide sorting problem?

Question 2. How might parallelism help?

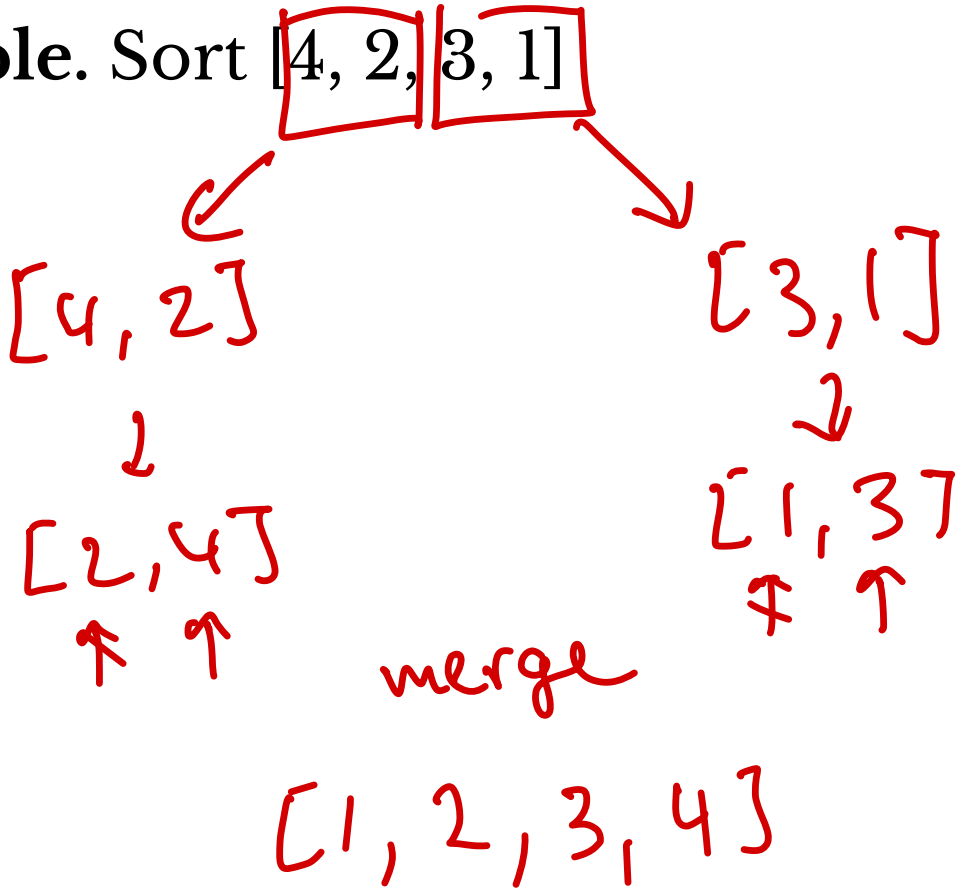
If have sub-tasks maybe can solve some subtasks in parallel.

Strategy 1: Divide by *Index*

Idea. (merge sort)

1. Split array into `left` and `right` sub-arrays
2. Sort `left` and `right` (recursively)
3. Merge sorted `left` and right

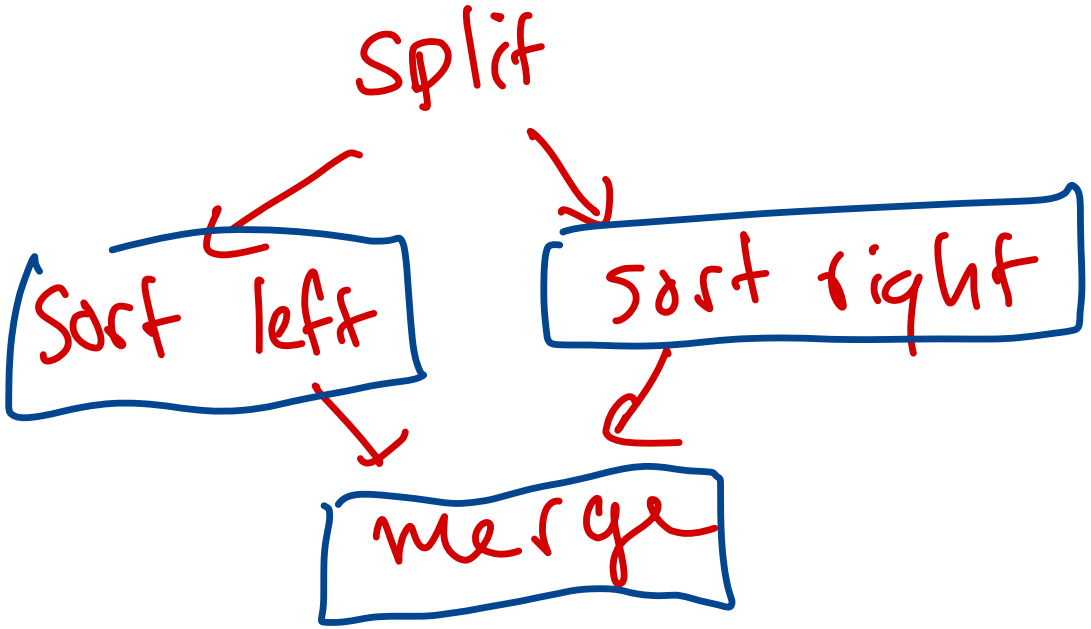
Example. Sort [4, 2, 3, 1]



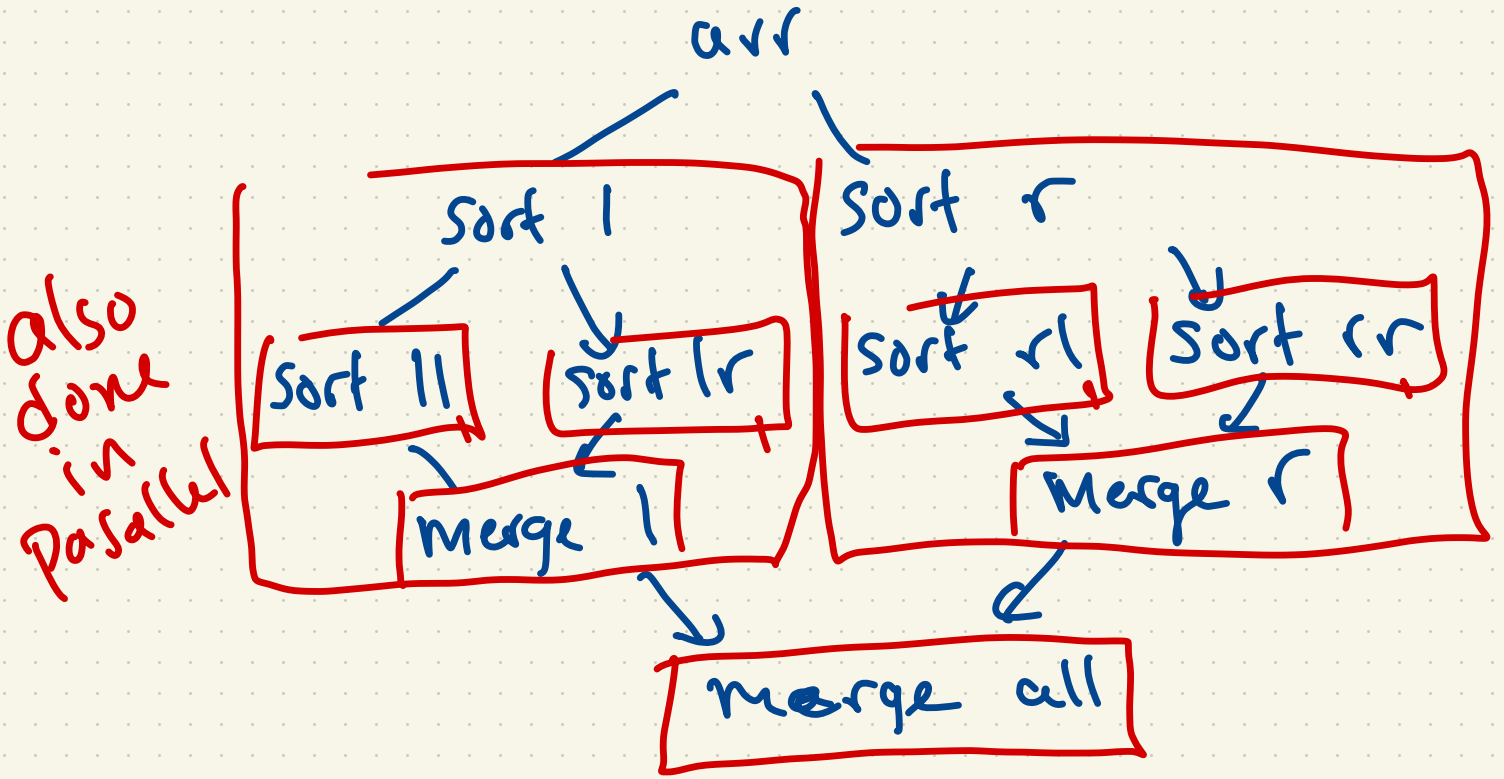
Parallelism?

Question. What can be performed in parallel?

1. Split array into `left` and `right` sub-arrays
2. Sort `left` and `right`
3. Merge sorted `left` and right



done in parallel



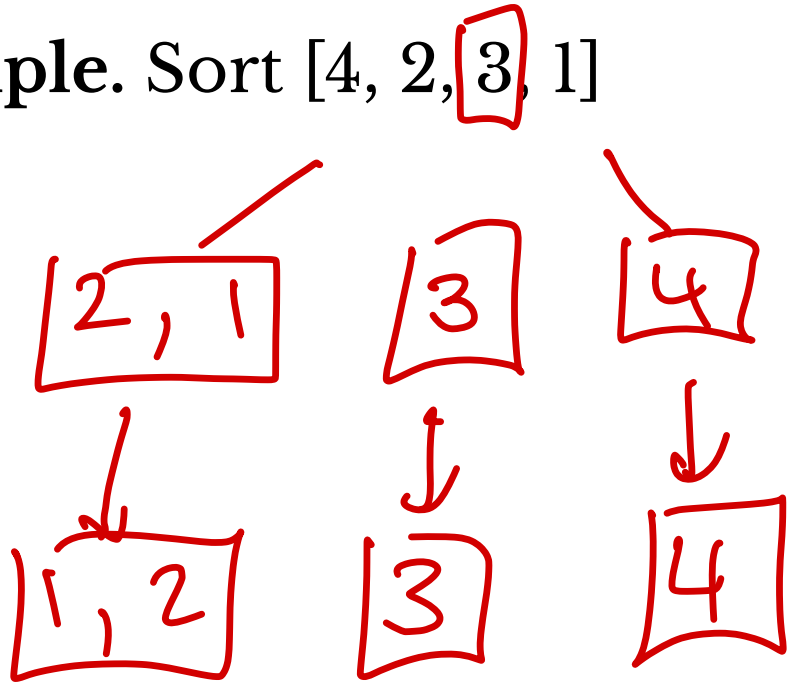
Fork-join Pool

Strategy 2: Divide by *Value*

Idea. (quick sort)

1. Select a value `pivot` from array
2. Partition array into `left` and `right` sub-arrays
   - values in `left` are `<= pivot`, values in `right` are `>`
3. Sort `left` and `right`

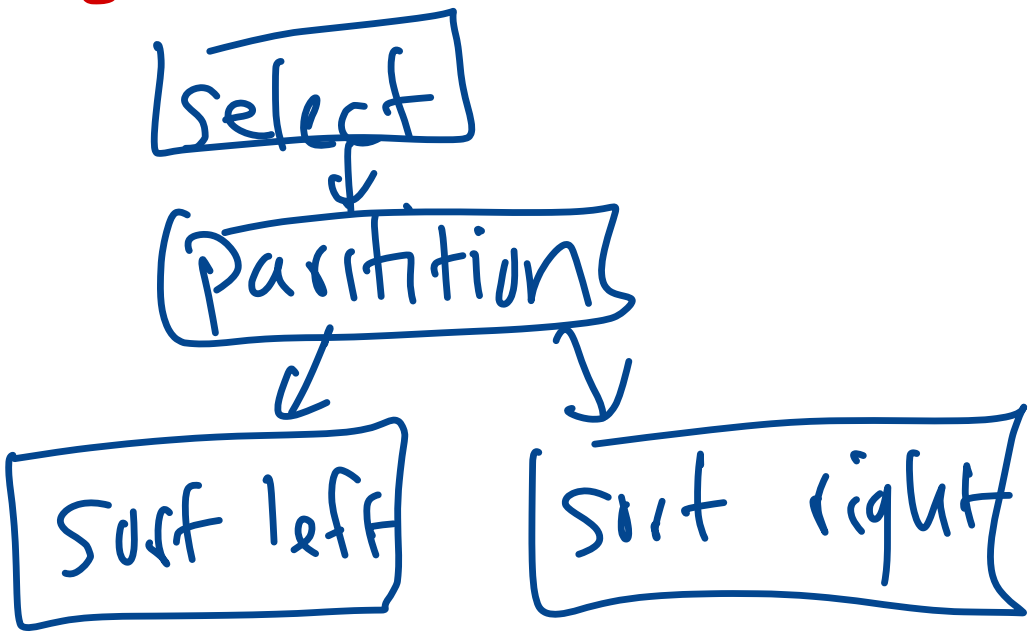
Example. Sort [4, 2, 3, 1]



Parallelism?

Question. What can be performed in parallel?

1. Select a value `pivot` from array
2. Partition array into `left` and `right` sub-arrays
   - values in `left` are `<= pivot`, values in `right` are `>`
3. Sort `left` and `right`



(Dis)advantages?

choose mid index

Merge sort: split, sub-sort, merge hard to parallelize

+ Lots of parallelizable ops
  → complicated to assign tasks to threads?

+ Even splitting

− merge hard to parallelize

− must wait for other threads to finish

(Dis)advantages?

Merge sort: split, sub-sort, merge

Quick sort: partition, sort

left
right

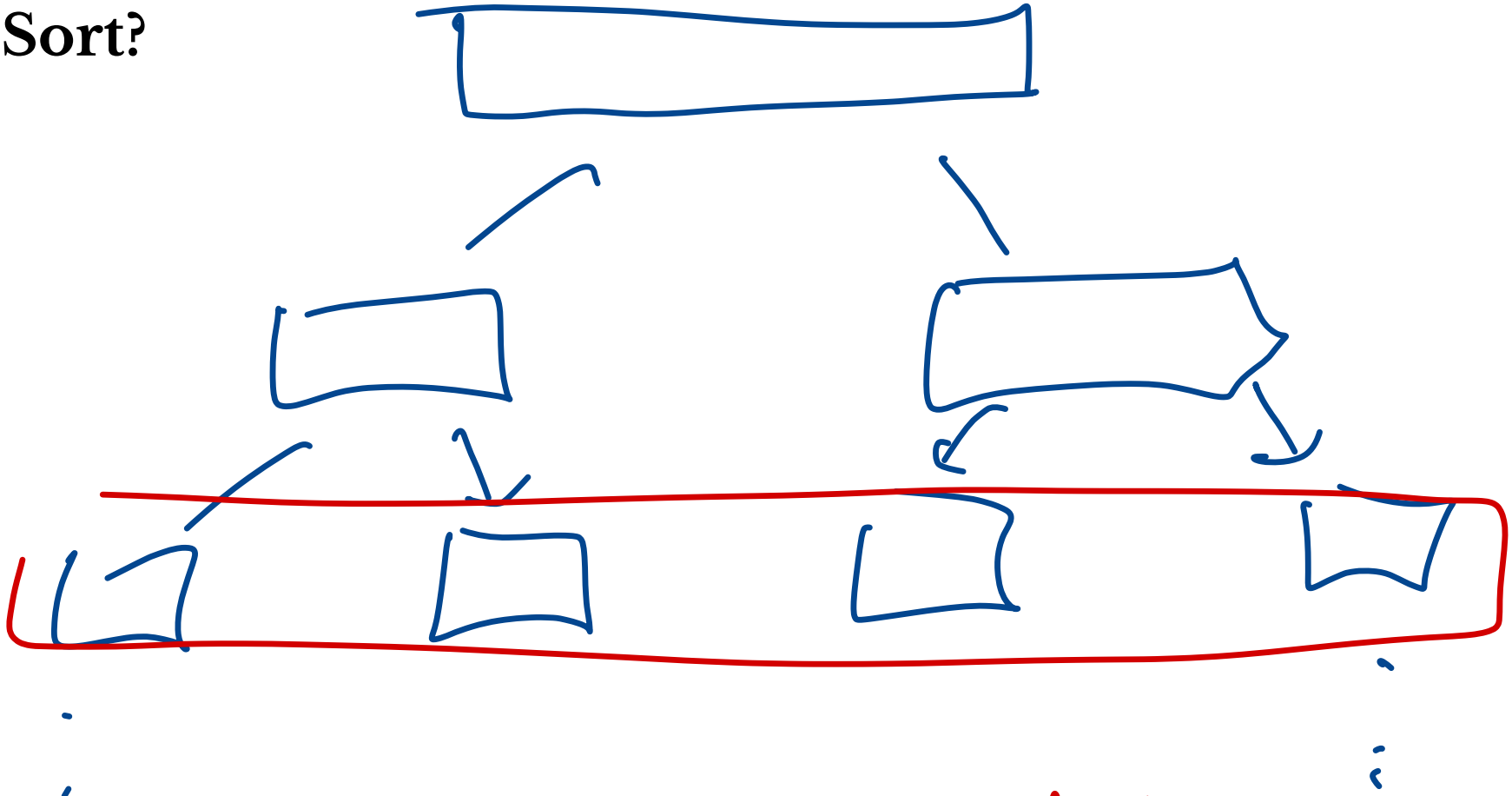
+ recursive sorts wait for no one!

- Partitioning is hard to parallelize

When to Stop Recursing?

array size n/M

Merge Sort?



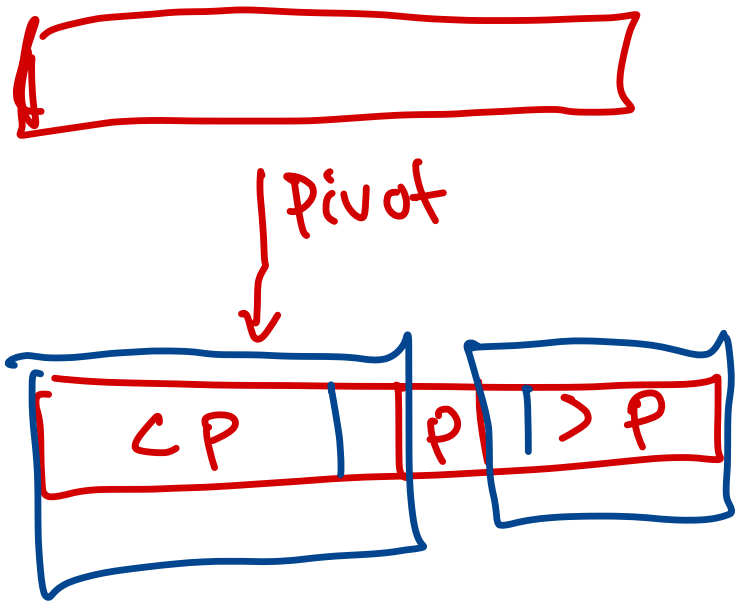
Stop @ size 2? - 500k tasks in 1 M?

Stop recursing w/ new threads when array is "small enough" length/#T.

When to Stop Recursing?

Merge Sort?

Quick Sort?



Stop making new threads for sufficiently small tasks?

Merge Operation

Question. How to `merge` two sorted arrays?

Speeding Up Merge?

Question. Can we improve performance of `merge` operation with SIMD instructions?

Partition Operation?

Question. How to `partition` an array given a `pivot`?

Speeding Up Partition?

Question. Can we improve performance of `partition` with SIMD instructions?

Next Week

1. Fork-join pools: thread pools tailored to recursive methods!
2. Sorting networks: the art of sorting small arrays