

# CS 312: Algorithm Analysis



## Lecture #39: Parallel Merge Sort

# Announcements

- **Proj. #4 Graded and on BlackBoard**
  - Please check your early bonus count and your late day budget on BlackBoard
  - Estimate your grade
- **Complete the Course Evaluation for "1 HW"**
  - In sec. 002 & 003, this will mean that you will receive extra credit in the amount of the average value of a homework asst.
- **CS 312 Ultimate Frisbee**
  - 4/19, Wednesday, First Reading Day
  - 10am, Kiwanis Park
- **Thought**

# Objectives

### Lecture #38: Parallel Algorithms

- Explain "computational model"
- Compare P-RAM with Fixed Network model of parallel computation
- Try some examples
- Compute speedup and efficiency

### Lecture #39: Parallel Algorithm for Merge Sort

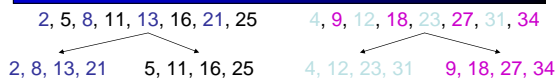
- Explain odd/even parallel merge
- Analyze odd/even merge
- Use odd/even merge in parallel sort
- Analyze odd/even sort

# Odd/Even Merge

2, 5, 8, 11, 13, 16, 21, 25      4, 9, 12, 18, 23, 27, 31, 34

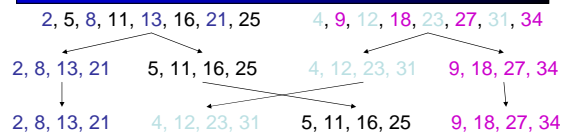
Split each list into two parts:  
 Odd indices in one part, even in the other  
 (Hence the name)

# Odd/Even Merge

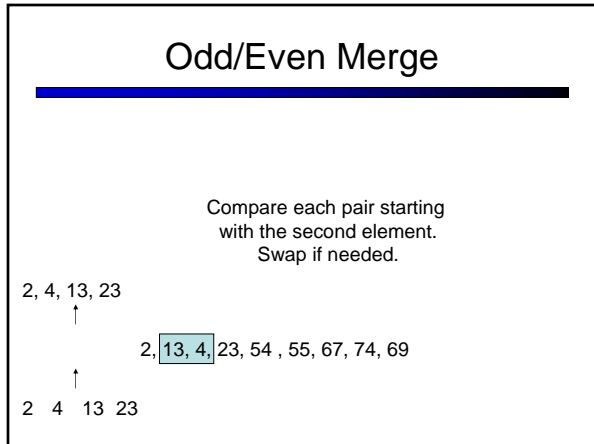
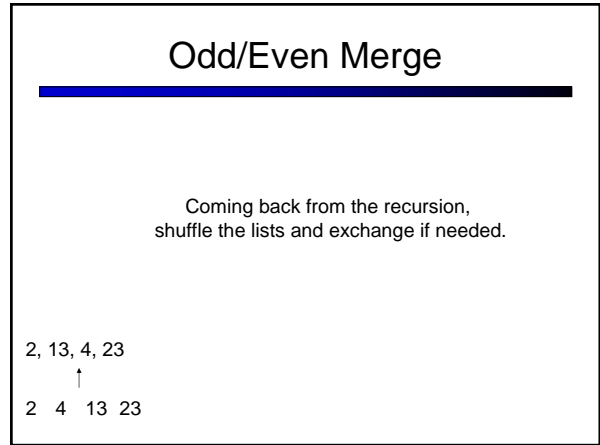
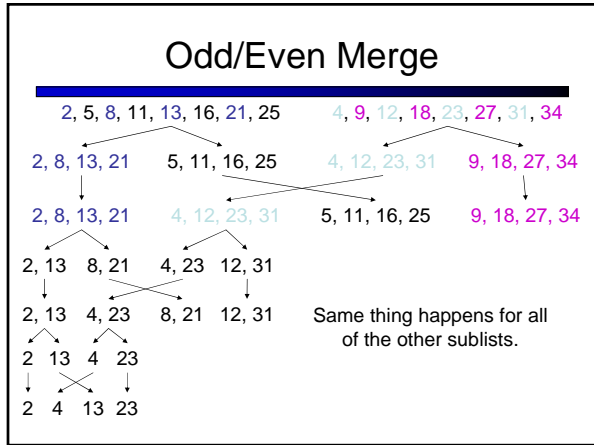
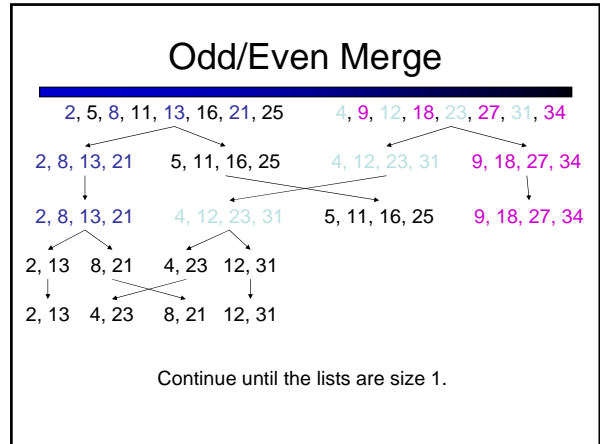
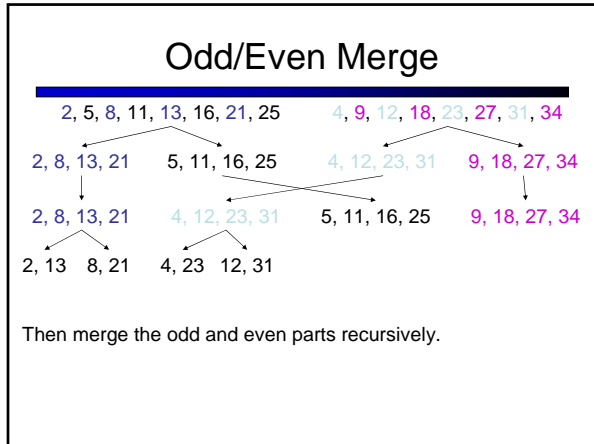


Recursively merge odd lists and even lists.

# Odd/Even Merge



The recursion means that the smaller lists are divided into even and odd parts.



- ### Analysis of Odd/Even Merge
- 
- Given two lists of  $m = n/2$  keys each.
  - Subdivide the lists until length = 1.
  - Assign one processor to each sublist, *or we recurse.*
  - Partitioning:  $O(1)$
  - Recursive merge:  $f(m/2)$
  - Compare/exchange:  $O(1)$
  - Entire process:  $f(m) = f(m/2) + O(1) = O(\log m)$
- $= O(\log n)$

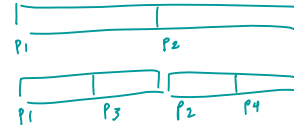
## Comparison with Seq. Merge

- Sequential merge is  $O(n)$ .
- Parallel merge is  $O(\log n)$  on  $n$  machines
- So efficiency is  $\frac{O(n)}{nO(\log n)} = \frac{1}{\log n} \leq 1$

And speedup is  $\frac{O(n)}{O(\log n)} = O\left(\frac{n}{\log n}\right) \geq 1$

## Parallel MergeSort

- Given a list to sort
- Split the list into halves
- Recursively parallel MergeSort the halves
- Merge the halves using Even/Odd merge at each level:  $\log n$  levels with  $\log n$  work at each.



## Analysis

- Parallel MergeSort is therefore  $O(\log^2 n)$  on  $n$  processors
- Optimal sequential algorithm is  $O(n \log n)$

What's the efficiency?  $\frac{O(n \log n)}{O(\log^2 n)} = O\left(\frac{n}{\log n}\right)$

What's the speedup?  $\frac{O(n \log n)}{O(\log^2 n)} = O\left(\frac{n}{\log n}\right)$

$p = n$

## Another Sorting Algorithm

- Suppose we have a  $O(\log n)$  sorting algorithm on  $n$  processors.
- Given that sequential algorithm is  $O(n \log n)$
- What is the speedup?
- What is the efficiency?

The End