Week 4 Lecture 10

Theory

What's in this lecture?

- Reasoning about the Efficiency of Algorithms

Sorting

- Previous lectures: explored bubble sort, insertion sort, and merge sort
- Today: quantifying running time of algorithms
- For any algorithm, we seek to describe a recurrence relation that talks about its running time in terms of component operations

Recap: Merge Sort

- Input: array of size 1
  - Return the array itself (already sorted)
- Input: an array of size N
  - Split into 2 sub-arrays of size N/2
  - Recursively sort sub-arrays
  - Merge two sorted sub-arrays

Talking about Time

- T(1) means: "operations necessary for merge_sort of array of size 1"
  - Thus, T(1) = 1
- T(N) means: "operations necessary for merge_sort of array of size N"
  - T(N) = T(split(N)) + T(N/2) + T(N/2) + T(merge(N/2, N/2))

Digging In

- T(N) = T(split(N)) + T(N/2) + T(N/2)
  + T(merge(N/2, N/2))
  - T(split(N)) = N * copy from original array
  - T(N/2) = time of recursive merge_sort on array of size N/2
  - T(merge(N/2, N/2)) = N * copy from original to destination array

Simplifying Terms

- T(N) = T(split(N)) + T(N/2) + T(N/2) + T(merge(N/2, N/2))
- T(N) = N + T(N/2) + T(N/2) + N
- T(N) = 2N + 2T(N/2)

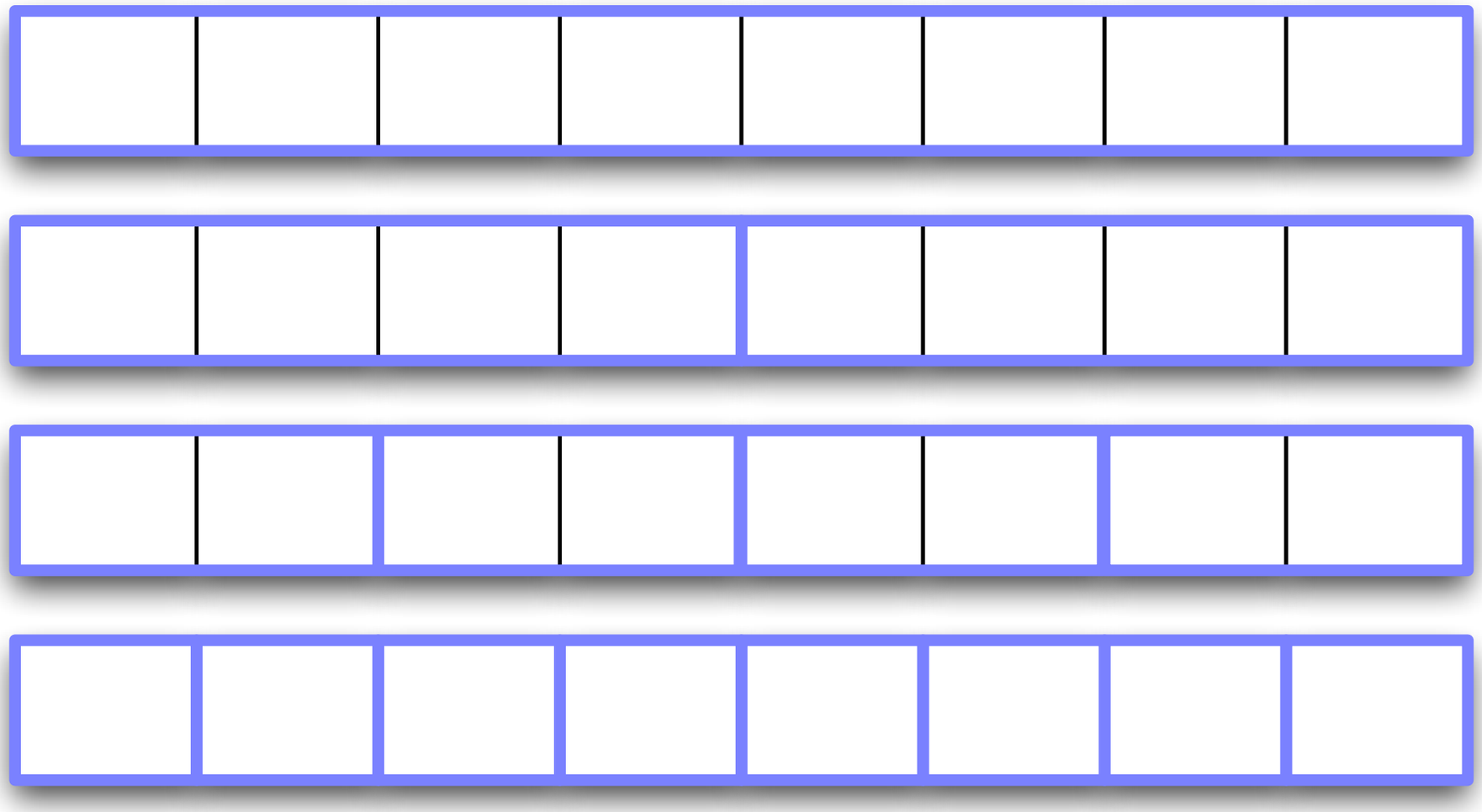
Question

- How many times can something be split in half before it is less than or equal to 1?

Answer

- The number of times is less than ceil(lg(N))
- Thus, merge sort does at most ceil(lg(N)) recursive calls

Recursive Calls



Working it Out

- $T(8) = 8 + 2T(4) + 8$
  $= 16 + 2(4 + 2T(2) + 4)$
  $= 32 + 4T(2)$
  $= 32 + 4(2 + 2T(1) + 2)$
  $= 48 + 8T(1)$
  $= 48 + 8$
  $= 56$

Formalizing

- For merge sort, we have:
  $T(N) = 2T(N/2) + 2N$
- Using the master theorem (not covered here), the running time of merge sort as N approaches infinity is proportional to the function: N * lg(N)
- Intuitively, there are lg N levels in the recursion tree, and the total work in each level is N

Take-Aways

- It is possible to analyze and compare algorithms based on recurrence relations
- Calculating concrete recurrences is beyond the scope of this class
- For many algorithms and data structures, the running times of operations are well known

Exercises

- Read wikipedia entries on Merge Sort, Insertion Sort, and Bubble Sort
- Re-read CLR sections which talk about running-time analysis of Merge Sort and Insertion Sort in a best-attempt to understand them