# CS5800 – ALGORITHMS

# MODULE 4. DIVIDE & CONQUER - II

# Lesson 1: Merge Sort

Ravi Sundaram

# Topics

- Divide & Conquer – recap
- Sorting – many applications
- Merge Sort
  - Algorithm
  - Efficient merging
- Analysis
  - Recursion Tree
- Summary

# Divide & Conquer – recap

- Divide and conquer.
  - Break up problem into several parts.
  - Solve each part recursively.
  - Combine solutions to sub-problems into overall solution.

- Most common usage.
  - Break up problem of size n into two equal parts of size ½n.
  - Solve two parts recursively.
  - Combine two solutions into overall solution in linear time.

- Consequence.
  - Brute force :- $n^2$.
  - Divide and conquer :- n log n.

# Sorting – many applications

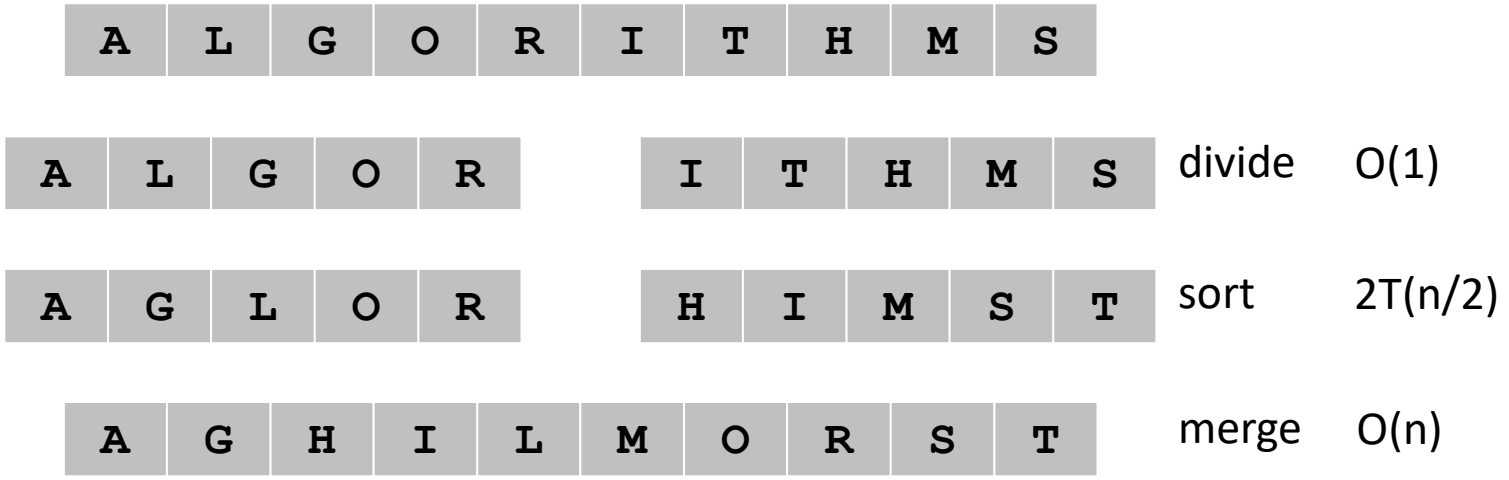
- Sorting.  Given an array of n elements, rearrange them in ascending order.

- Specific Applications:
  - Sort a list of names.
  - Organize an MP3 library.
  - Display Google PageRank results.
  - Find the median.
  - Find the closest pair.
  - Binary search in a database.
  - Identify statistical outliers.
  - Find duplicates in a mailing list.

- General areas:
  - Data compression.
  - Computer graphics.
  - Computational biology.
  - Supply chain management.
  - Book recommendations on Amazon.
  - Load balancing on a parallel computer.

# Merge Sort – algorithm

- Mergesort.
  - Divide array into two halves.
  - Recursively sort each half.
  - Merge two halves to make sorted whole.



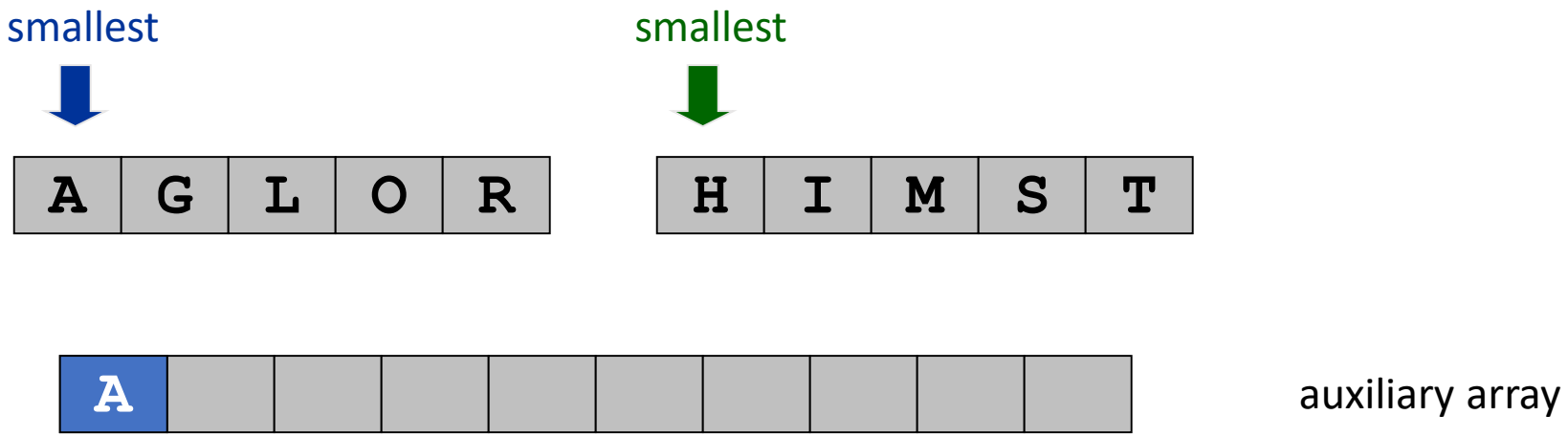
# Merge Sort – efficient merging

- Merging.  Combine two pre-sorted lists into a sorted whole.

- How to merge efficiently?
  - Linear number of comparisons.
  - Use temporary array.

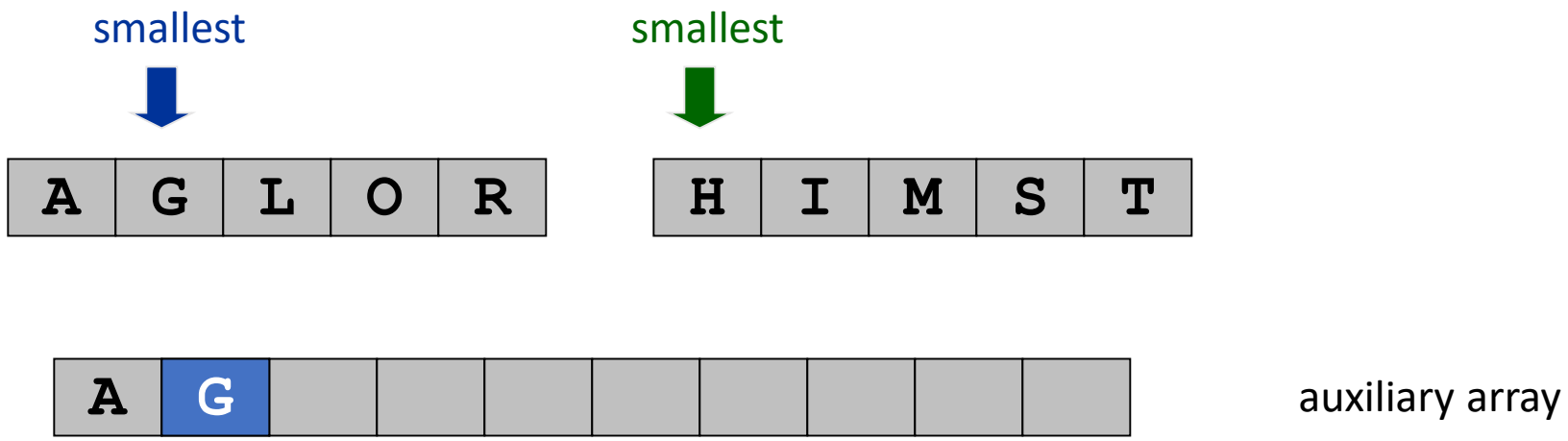
# Merge Sort – efficient merging

- Merge.
  - Keep track of smallest element in each sorted half.
  - Insert smallest of two elements into auxiliary array.
  - Repeat until done.



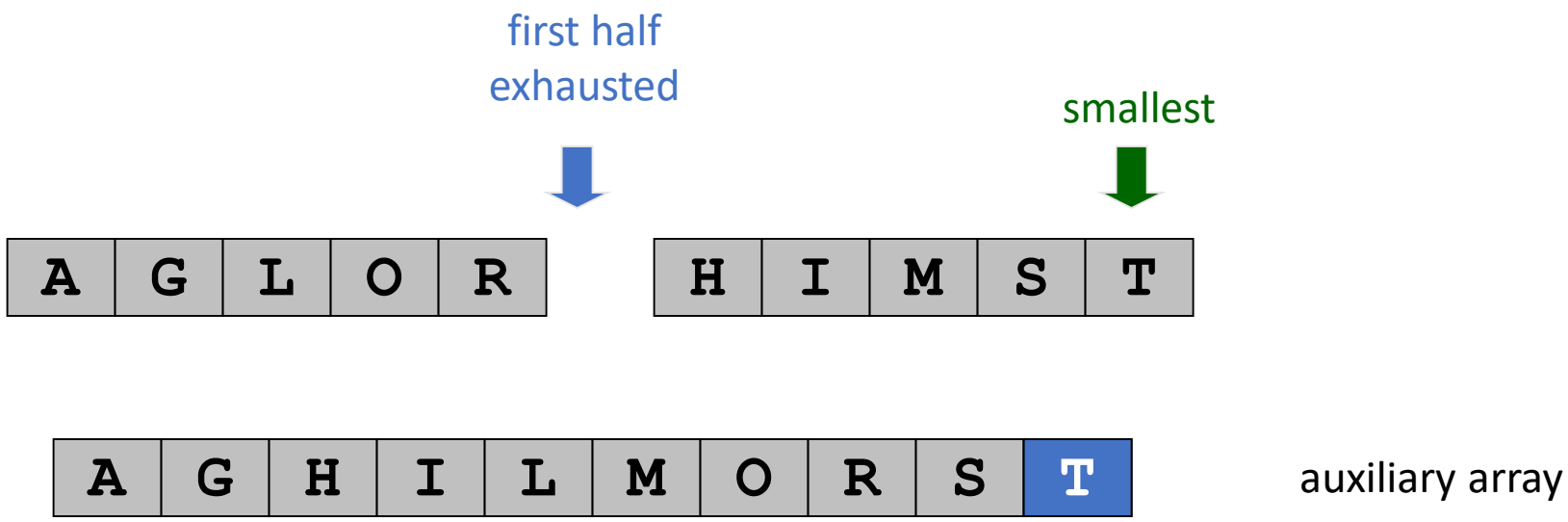
# Merge Sort – efficient merging

- Merge.
  - Keep track of smallest element in each sorted half.
  - Insert smallest of two elements into auxiliary array.
  - Repeat until done.



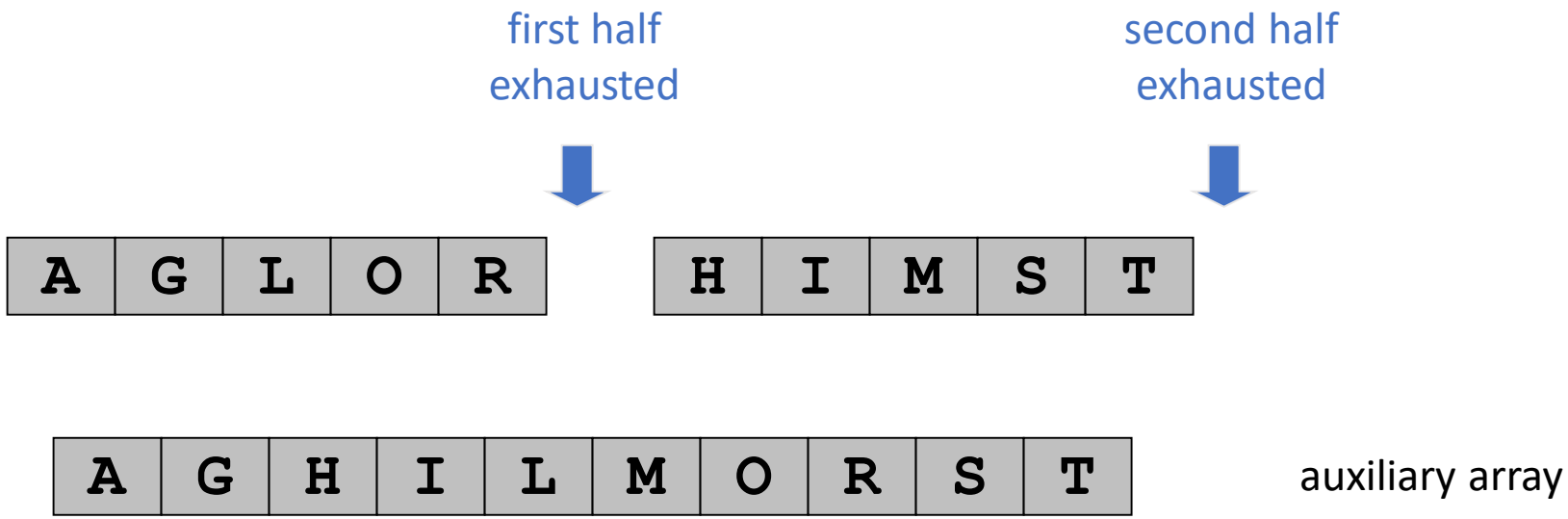
# Merge Sort – efficient merging

- Merge.
  - Keep track of smallest element in each sorted half.
  - Insert smallest of two elements into auxiliary array.
  - Repeat until done.



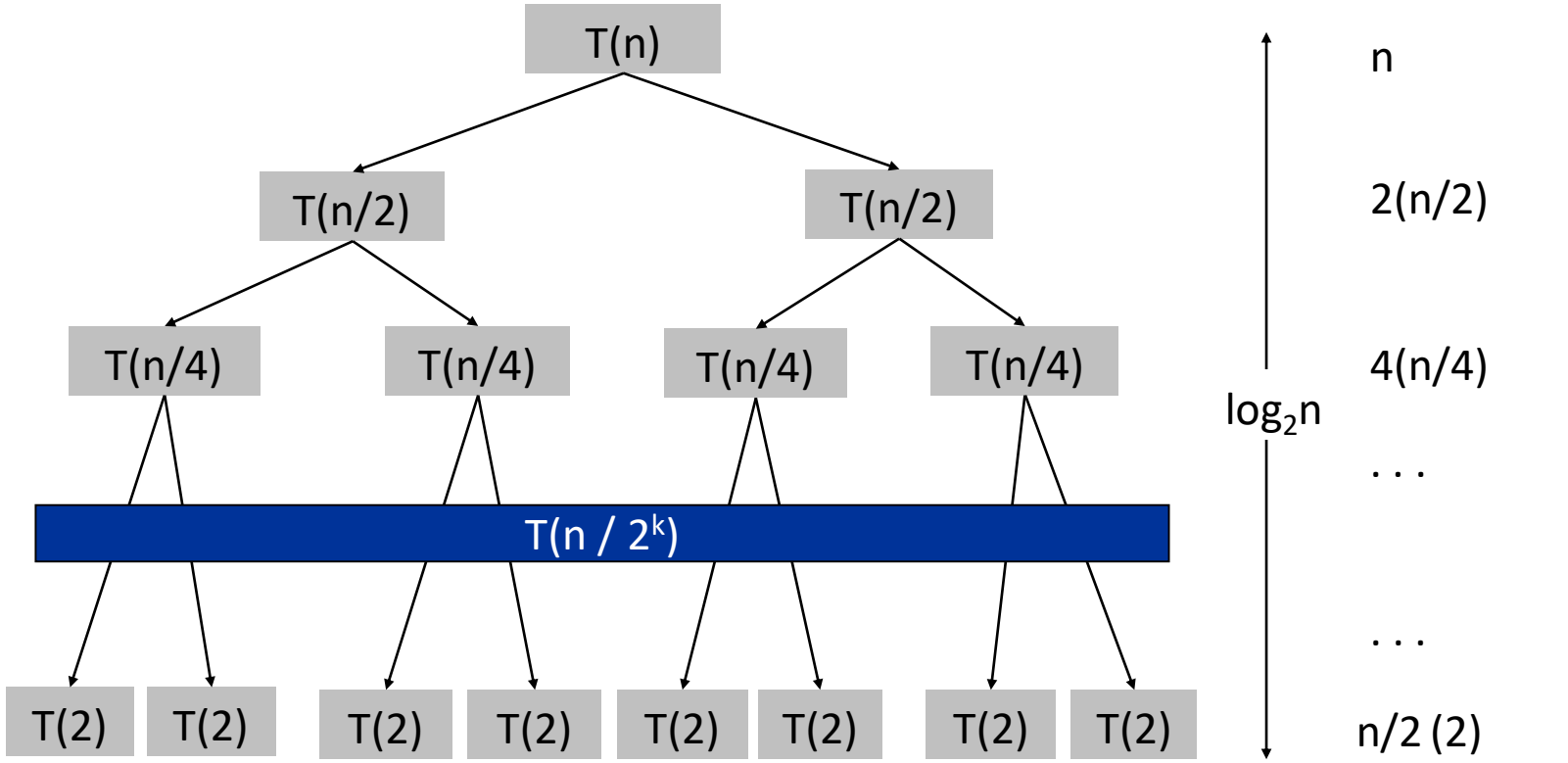
# Merge Sort – efficient merging

- Merge.
  - Keep track of smallest element in each sorted half.
  - Insert smallest of two elements into auxiliary array.
  - Repeat until done.



# Analysis – recursion tree

Solve $T(n) = 2T(n/2) + n$



# Analysis – Master Theorem

- $T(n) = aT(n/b) + f(n)$
- For Merge Sort we have $a = b = 2$ and $f(n) = n$
- By Master Theorem

  $f(n) = n = \Theta(n^{\log_b a}) \Rightarrow$

  $T(n) = \Theta(n \log n)$

# Summary

- The recursive approach of Divide & Conquer allows us to assume we can solve subproblems and then combine the answers to get the solution to the original larger problem
- Applied to sorting, a fundamental problem with many application, the Divide & Conquer approach gives rise to Merge Sort
- The main cleverness in Merge Sort is in combining the solutions to the sub problems, i.e. merging the two sorted halves
- With an extra array merging can be done efficiently in linear time giving rise to complexity $T(n) = 2T(n/2) + n = \Theta(n\log n)$