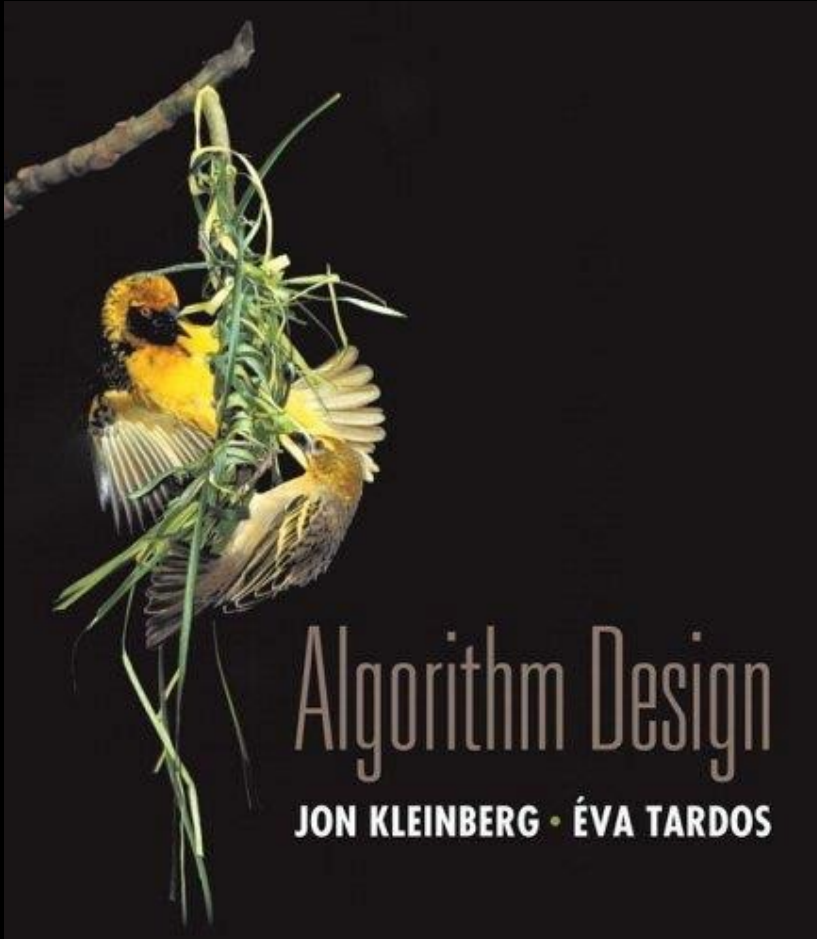


# CS 580: Algorithm Design and Analysis

---

# Chapter 5

## Divide and Conquer



Slides by Kevin Wayne.  
Copyright © 2005 Pearson-Addison Wesley.  
All rights reserved.

# Divide-and-Conquer

## 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  $\frac{1}{2}n$ .
- Solve two parts recursively.
- Combine two solutions into overall solution in **linear time**.

## Consequence.

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

Divide et impera.  
Veni, vidi, vici.  
- *Julius Caesar*

## 5.1 Mergesort

# Sorting

**Sorting.** Given  $n$  elements, rearrange in ascending order.

Obvious sorting applications.

- List files in a directory.

- Organize an MP3 library.

- List names in a phone book.

- Display Google PageRank results.

Problems become easier once sorted.

- Find the median.

- Find the closest pair.

- Binary search in a database.

- Identify statistical outliers.

- Find duplicates in a mailing list.

Non-obvious sorting applications.

- Data compression.

- Computer graphics.

- Interval scheduling.

- Computational biology.

- Minimum spanning tree.

- Supply chain management.

- Simulate a system of particles.

- Book recommendations on Amazon.

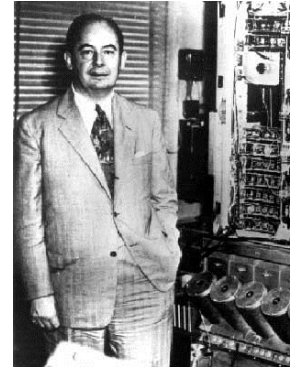
- Load balancing on a parallel computer.

- ...

# Mergesort

## Mergesort.

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



Jon von Neumann (1945)

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| A | L | G | O | R | I | T | H | M | S |
|---|---|---|---|---|---|---|---|---|---|

|   |   |   |   |   |
|---|---|---|---|---|
| A | L | G | O | R |
|---|---|---|---|---|

|   |   |   |   |   |
|---|---|---|---|---|
| I | T | H | M | S |
|---|---|---|---|---|

divide  $O(1)$

|   |   |   |   |   |
|---|---|---|---|---|
| A | G | L | O | R |
|---|---|---|---|---|

|   |   |   |   |   |
|---|---|---|---|---|
| H | I | M | S | T |
|---|---|---|---|---|

sort  $2T(n/2)$

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| A | G | H | I | L | M | O | R | S | T |
|---|---|---|---|---|---|---|---|---|---|

merge  $O(n)$

# Merging

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

**How to merge efficiently?**



- Linear number of comparisons.
- Use temporary array.



**Challenge for the bored.** In-place merge. [Kronrud, 1969]

↑  
using only a constant amount of extra storage

# Merging

## Merge.

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

smallest



|   |   |   |   |   |
|---|---|---|---|---|
| A | G | L | O | R |
|---|---|---|---|---|

smallest



|   |   |   |   |   |
|---|---|---|---|---|
| H | I | M | S | T |
|---|---|---|---|---|

|   |  |  |  |  |  |  |  |  |  |
|---|--|--|--|--|--|--|--|--|--|
| A |  |  |  |  |  |  |  |  |  |
|---|--|--|--|--|--|--|--|--|--|

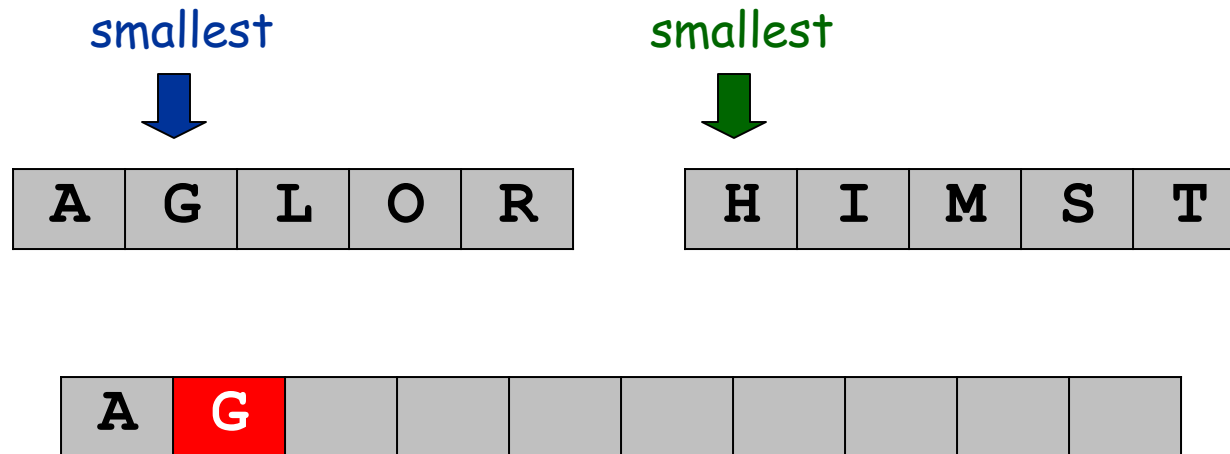
auxiliary array



# Merging

## Merge.

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

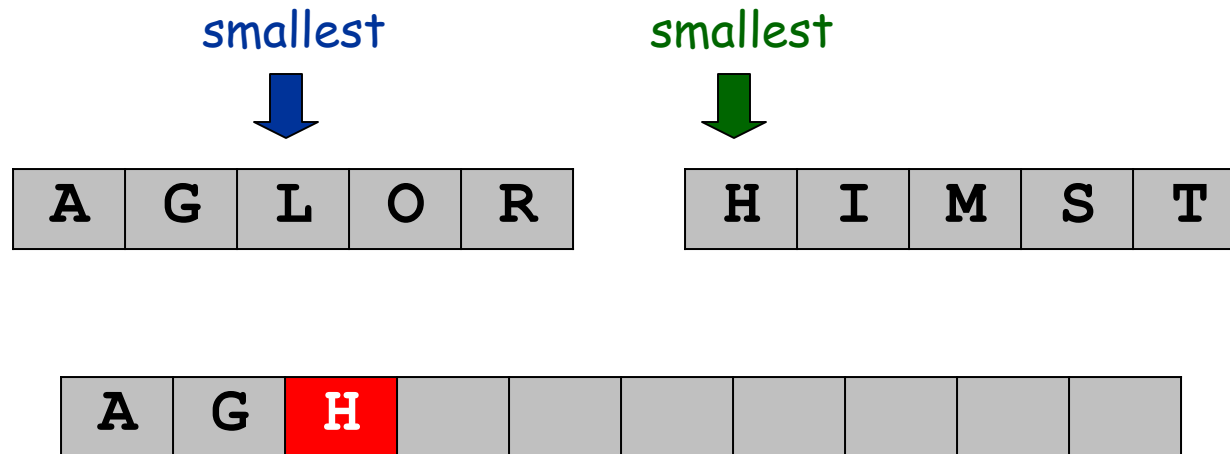


auxiliary array

# Merging

## Merge.

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

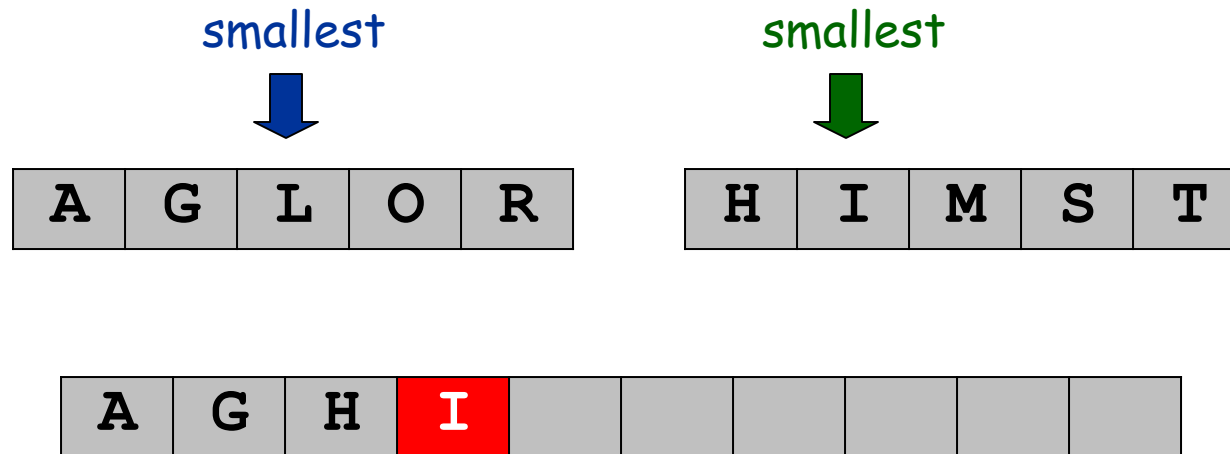


auxiliary array

# Merging

## Merge.

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

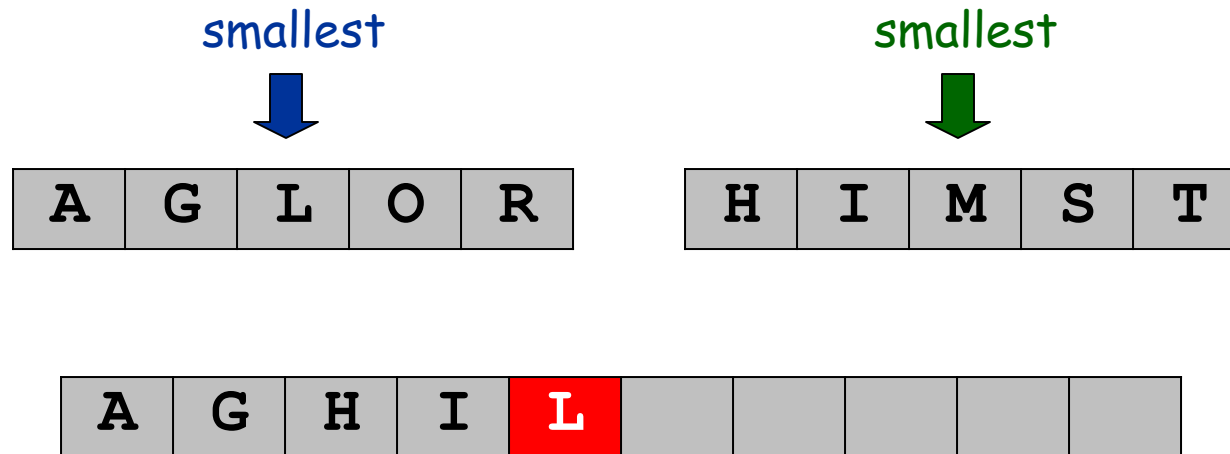


auxiliary array

# Merging

## Merge.

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

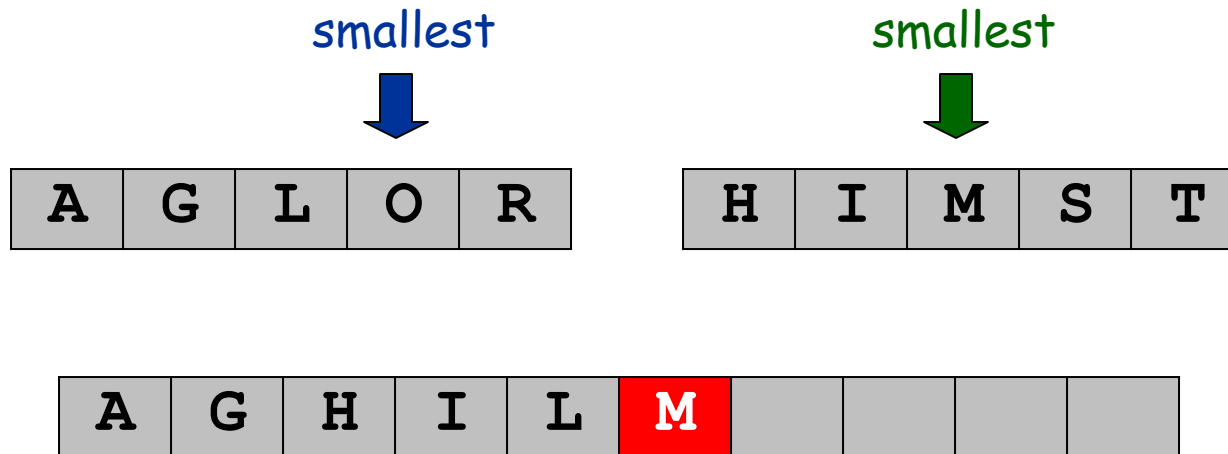


auxiliary array

# Merging

## Merge.

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

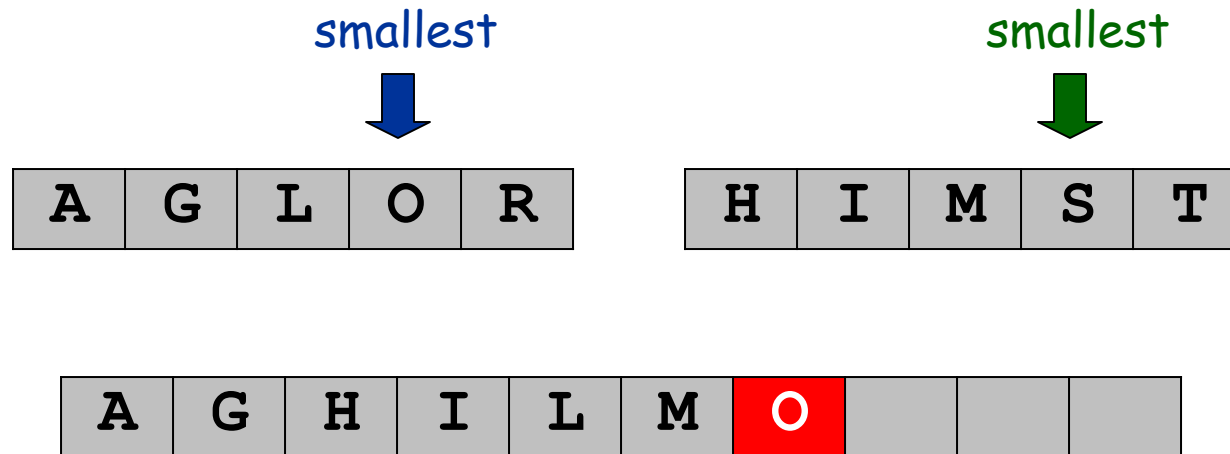


auxiliary array

# Merging

## Merge.

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

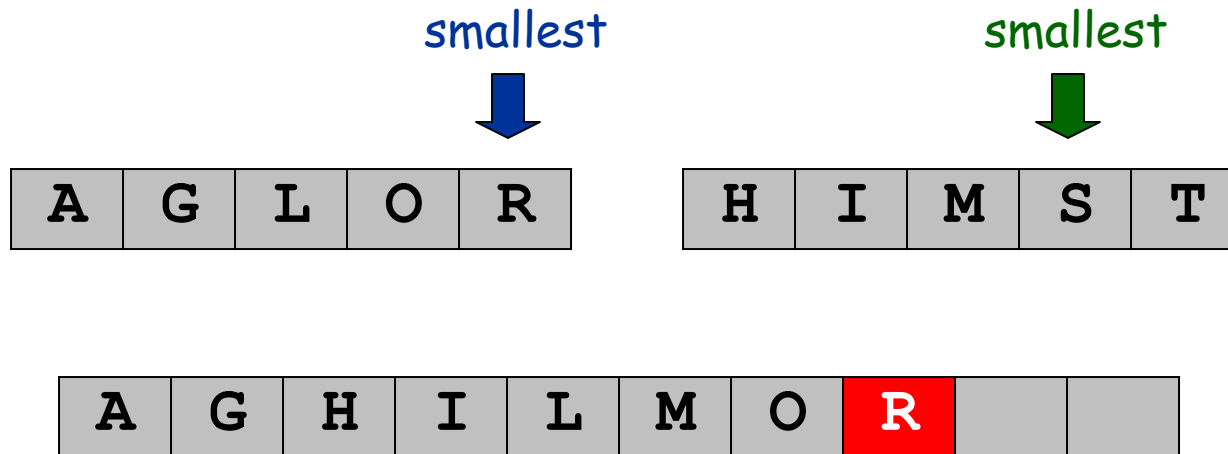


auxiliary array

# Merging

## Merge.

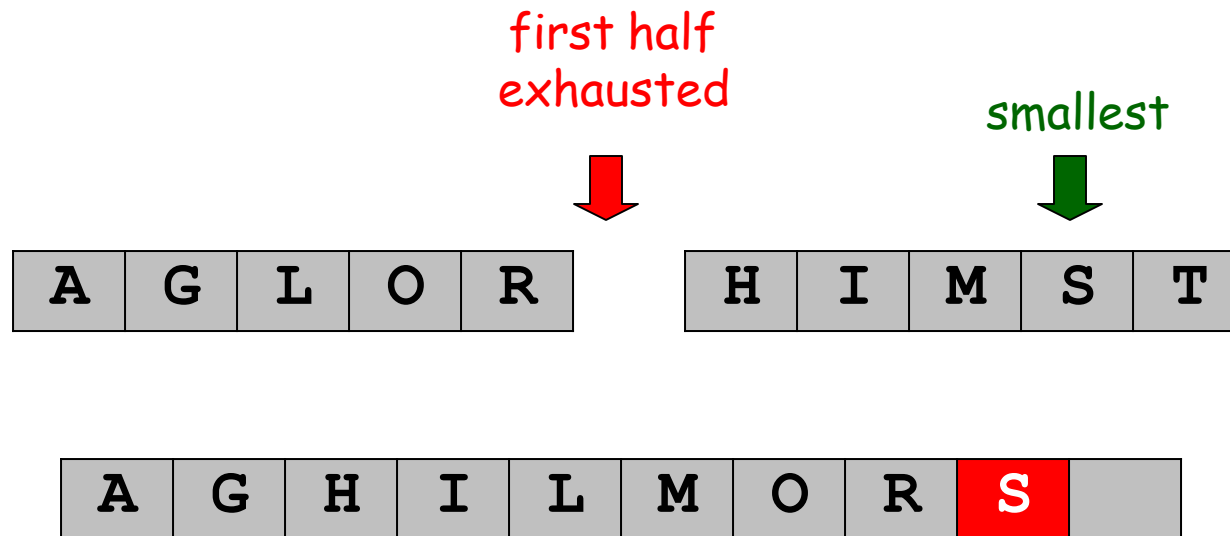
- Keep track of smallest element in each sorted half.
- Insert smallest of two elements into auxiliary array.
- Repeat until done.



# Merging

## Merge.

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



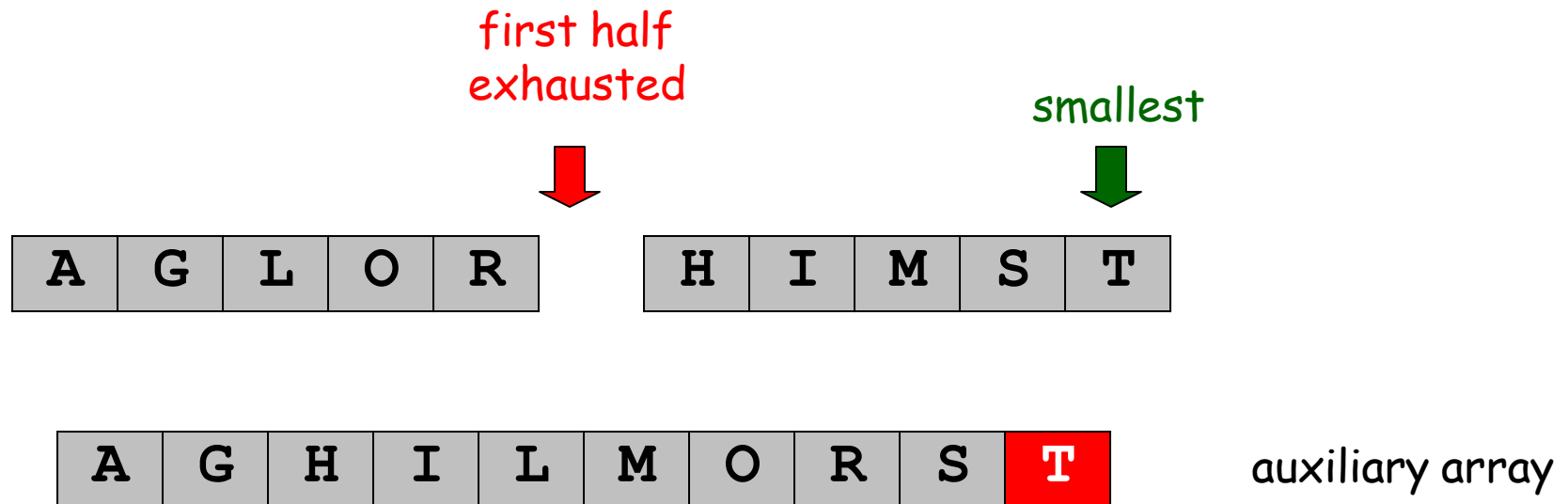
auxiliary array



# Merging

## Merge.

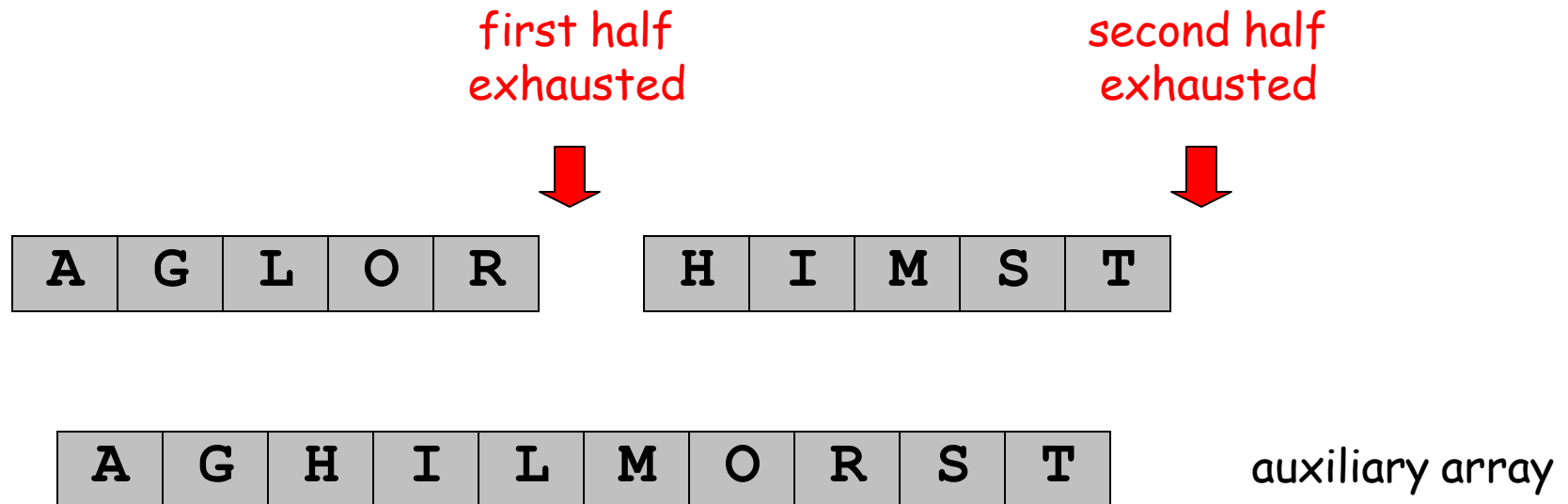
- Keep track of smallest element in each sorted half.
- Insert smallest of two elements into auxiliary array.
- Repeat until done.



# Merging

## Merge.

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



## A Useful Recurrence Relation

Def.  $T(n)$  = number of comparisons to mergesort an input of size  $n$ .

Mergesort recurrence.

**Exercise:** Write recurrence (i.e. inequality to upper bound  $T(n)$ ).

## A Useful Recurrence Relation

Def.  $T(n)$  = number of comparisons to mergesort an input of size  $n$ .

Mergesort recurrence.

$$T(n) \leq \begin{cases} 0 & \text{if } n = 1 \\ \underbrace{T(\lceil n/2 \rceil)}_{\text{solve left half}} + \underbrace{T(\lfloor n/2 \rfloor)}_{\text{solve right half}} + \underbrace{n}_{\text{merging}} & \text{otherwise} \end{cases}$$

# A Useful Recurrence Relation

Def.  $T(n)$  = number of comparisons to mergesort an input of size  $n$ .

Mergesort recurrence.

$$T(n) \leq \begin{cases} 0 & \text{if } n = 1 \\ \underbrace{T(\lceil n/2 \rceil)}_{\text{solve left half}} + \underbrace{T(\lfloor n/2 \rfloor)}_{\text{solve right half}} + \underbrace{n}_{\text{merging}} & \text{otherwise} \end{cases}$$

What is the solution?

## A Useful Recurrence Relation

**Def.**  $T(n)$  = number of comparisons to mergesort an input of size  $n$ .

Mergesort recurrence.

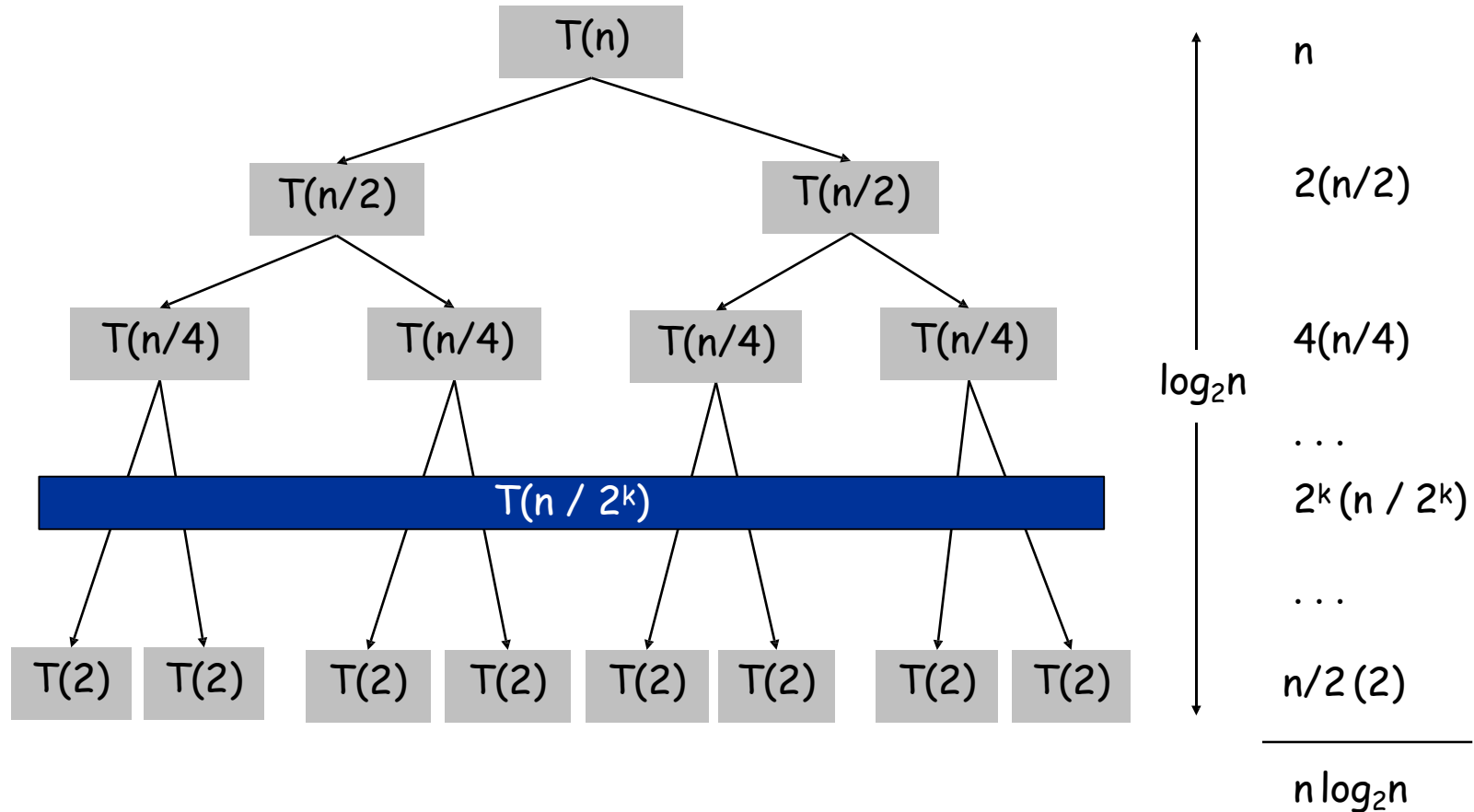
$$T(n) \leq \begin{cases} 0 & \text{if } n = 1 \\ \underbrace{T(\lceil n/2 \rceil)}_{\text{solve left half}} + \underbrace{T(\lfloor n/2 \rfloor)}_{\text{solve right half}} + \underbrace{n}_{\text{merging}} & \text{otherwise} \end{cases}$$

**Solution.**  $T(n) = O(n \log_2 n)$ .

**Assorted proofs.** We describe several ways to prove this recurrence. Initially we assume  $n$  is a power of 2 and replace  $\leq$  with  $=$ .

# Visualization of the recursion tree

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ \underbrace{2T(n/2)}_{\text{sorting both halves}} + \underbrace{n}_{\text{merging}} & \text{otherwise} \end{cases}$$



## Proof sketch

**Claim.** If  $T(n)$  satisfies this recurrence, then  $T(n) = n \log_2 n$ .

↑  
assumes  $n$  is a power of 2

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ \underbrace{2T(n/2)}_{\text{sorting both halves}} + \underbrace{n}_{\text{merging}} & \text{otherwise} \end{cases}$$

**Pf.** For  $n > 1$ :

$$\begin{aligned} \frac{T(n)}{n} &= \frac{2T(n/2)}{n} + 1 \\ &= \frac{T(n/2)}{n/2} + 1 \\ &= \frac{T(n/4)}{n/4} + 1 + 1 \\ &\dots \\ &= \frac{T(n/n)}{n/n} + \underbrace{1 + \dots + 1}_{\log_2 n} \\ &= \log_2 n \end{aligned}$$



# Proof by Induction

**Claim.** If  $T(n)$  satisfies this recurrence, then  $T(n) = n \log_2 n$ .

↑  
assumes  $n$  is a power of 2

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ \underbrace{2T(n/2)}_{\text{sorting both halves}} + \underbrace{n}_{\text{merging}} & \text{otherwise} \end{cases}$$

**Pf.** (by induction on  $n$ )

- Base case:  $n = 1$ .
- Inductive hypothesis:  $T(n) = n \log_2 n$ .
- Goal: show that  $T(2n) = 2n \log_2 (2n)$ .

$$\begin{aligned} T(2n) &= 2T(n) + 2n \\ &= 2n \log_2 n + 2n \\ &= 2n(\log_2 (2n) - 1) + 2n \\ &= 2n \log_2 (2n) \end{aligned}$$

# Analysis of Mergesort Recurrence

**Claim.** If  $T(n)$  satisfies the following recurrence, then  $T(n) \leq n \lceil \lg n \rceil$ .

$$T(n) \leq \begin{cases} 0 & \text{if } n = 1 \\ \underbrace{T(\lceil n/2 \rceil)}_{\text{solve left half}} + \underbrace{T(\lfloor n/2 \rfloor)}_{\text{solve right half}} + \underbrace{n}_{\text{merging}} & \text{otherwise} \end{cases}$$

$\uparrow$   
 $\lg n$

**Pf.** (by induction on  $n$ )

- Base case:  $n = 1$ .
- Define  $n_1 = \lfloor n / 2 \rfloor$ ,  $n_2 = \lceil n / 2 \rceil$ .
- Induction step: assume true for  $1, 2, \dots, n-1$ .

$$\begin{aligned} T(n) &\leq T(n_1) + T(n_2) + n \\ &\leq n_1 \lceil \lg n_1 \rceil + n_2 \lceil \lg n_2 \rceil + n \\ &\leq n_1 \lceil \lg n_2 \rceil + n_2 \lceil \lg n_2 \rceil + n \\ &= n \lceil \lg n_2 \rceil + n \\ &\leq n(\lceil \lg n \rceil - 1) + n \\ &= n \lceil \lg n \rceil \end{aligned}$$

$$\begin{aligned} n_2 &= \lceil n/2 \rceil \\ &\leq \lceil 2^{\lceil \lg n \rceil / 2} \rceil \\ &= 2^{\lceil \lg n \rceil / 2} \\ \Rightarrow \lg n_2 &\leq \lceil \lg n \rceil - 1 \end{aligned}$$