T&R Team of Algorithm Design
College of Computer Science, CQU

Algorithm Analysis & Design

Introduction to Algorithm

Chapter 4: Merge Sort and Recursion

Outline

- **2.1 Merge Sort**
- **2.2 Recursion Analyzing**

2.1 Merge Sort

Merging Sort

- **A typical algorithm based on divide-and-conquer**
  - **Divide: divide the given *n*-element-array into two sub arrays of about *n*/2 elements either**
  - **Conquer: sort the two sub arrays recursively**
  - **Merge: merge the two sorted sub arrays to generate the final output**

Merging Sort Pseudo Code

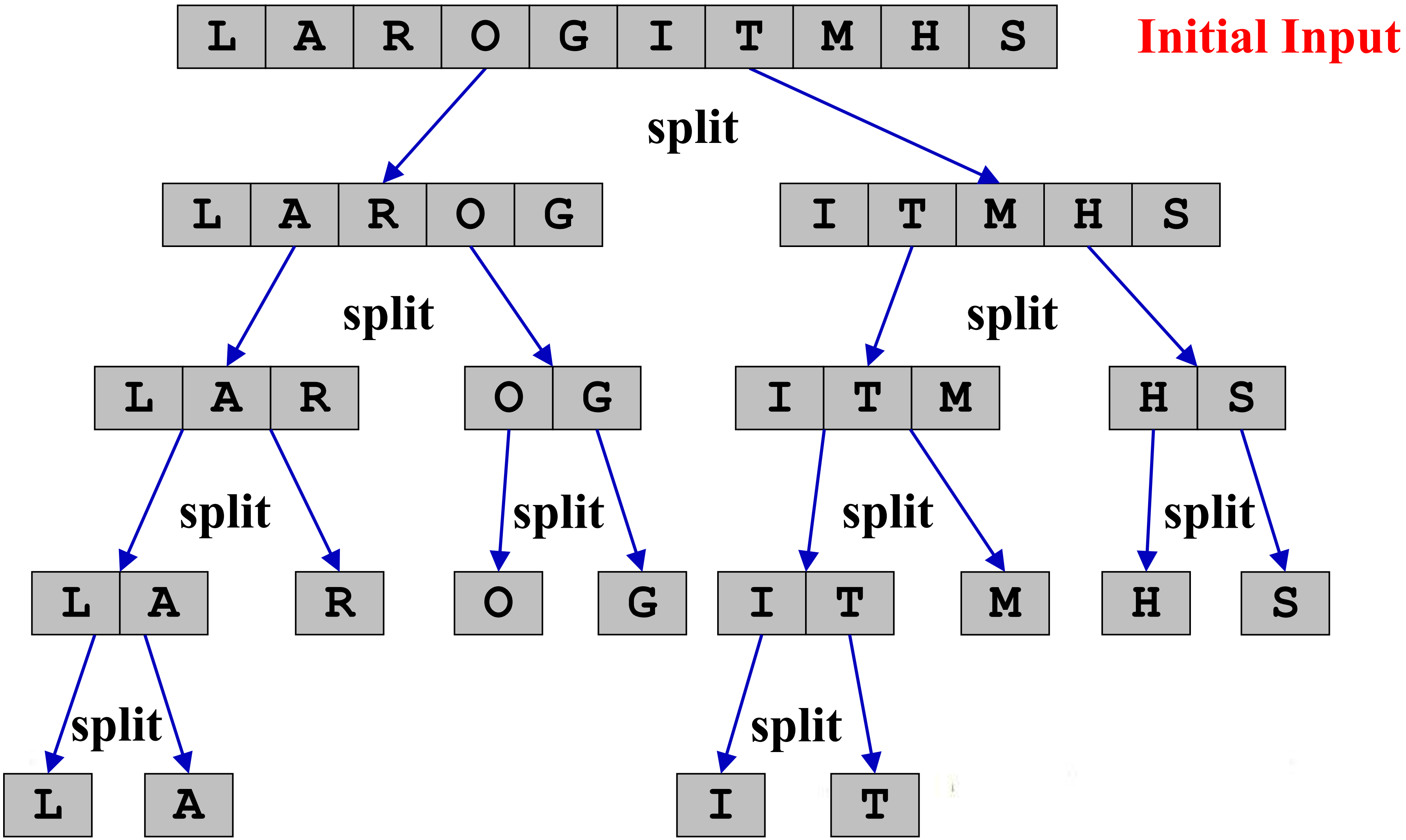
- **Input**: the unsorted array $A[p\ldots r]$
- **Output**: the sorted array A'

```
MERGE-SORT (A, p, r)
    if p < r
        then q ← ⌊(p + r) / 2⌋
            MERGE-SORT(A, p, q)
            MERGE-SORT(A, q + 1, r)
            MERGE(A, p, q, r)
```

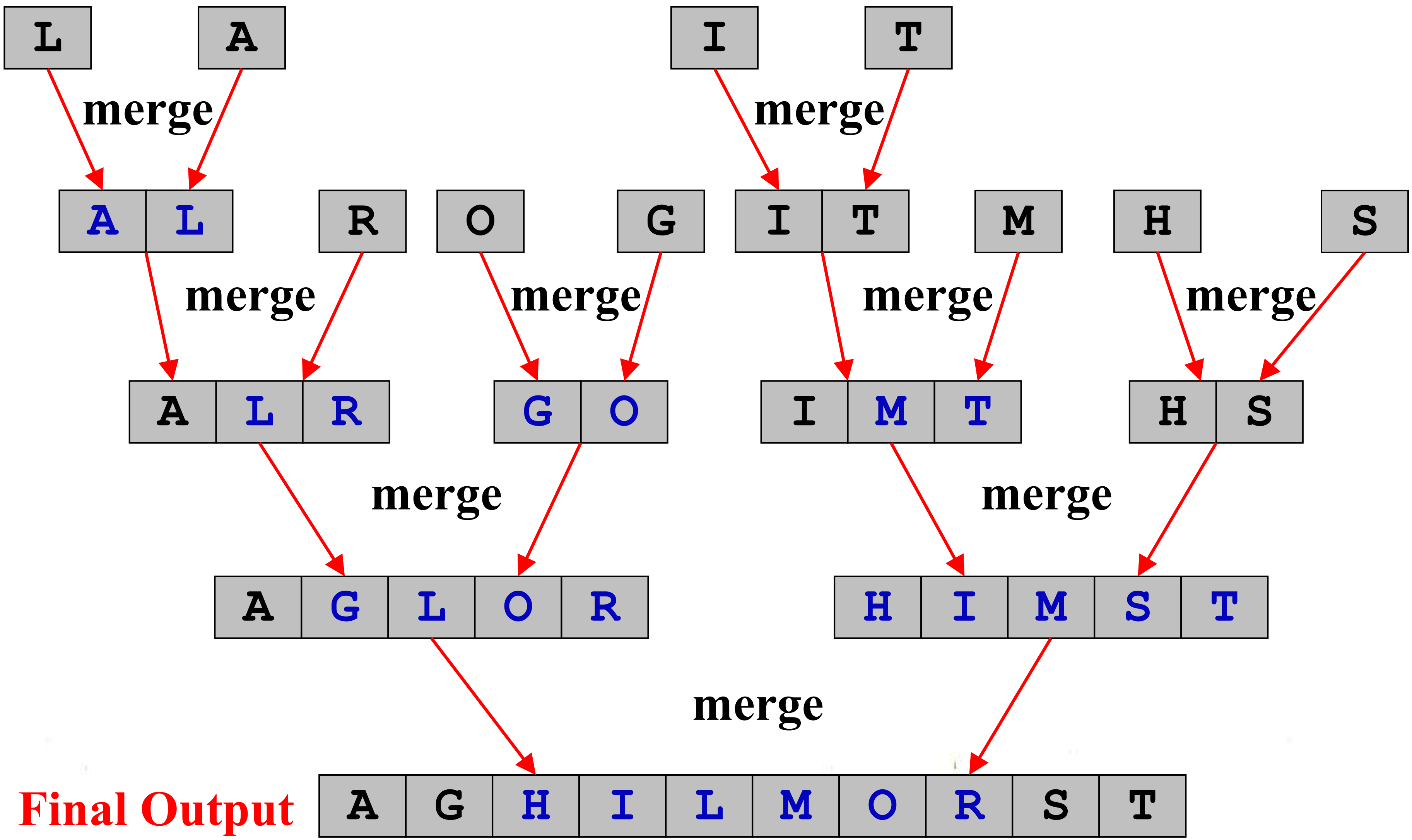
Pseudo Code of the Merge Procedure

```
MERGE (A, p, q, r)
    n1 ← q – p + 1
    n2 ← r – q
    create arrays L[1..n1 + 1] and R[1..n2 + 1]
    for i ← 1 to n1
            do L[i] ← A[p + i - 1]
    for j ← 1 to n2
            do R[j] ← A[q + j]
    L[n1 + 1] ← ∞
    R[n2 + 1] ← ∞
    i ← 1
    j ← 1
    for k ← p to r
            do if L[i] ⩽ R[j]
                    then A[k] ← L[i]
                         i ← i + 1
                    else A[k] ← R[j]
                         j ← j + 1
```

Merge Sort - Split

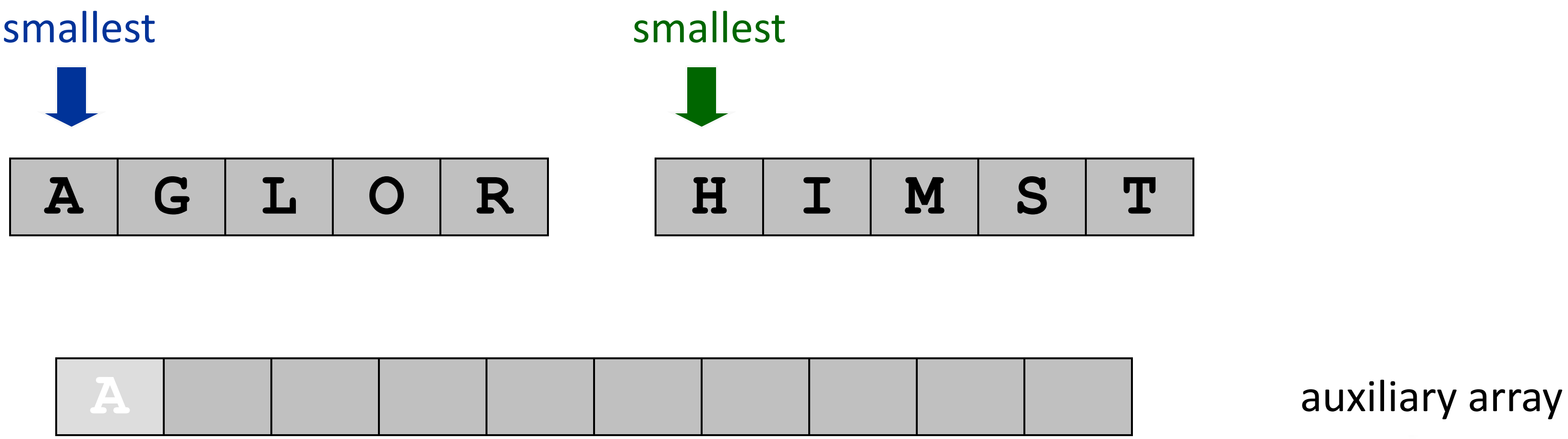


Merge Sort - Merge



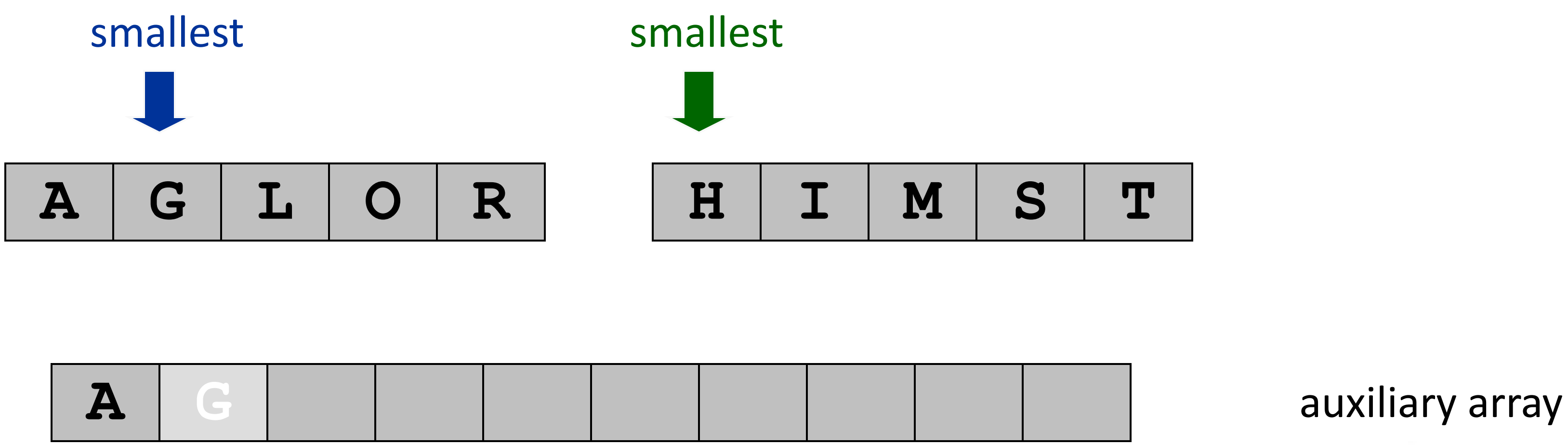
Merging Example

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



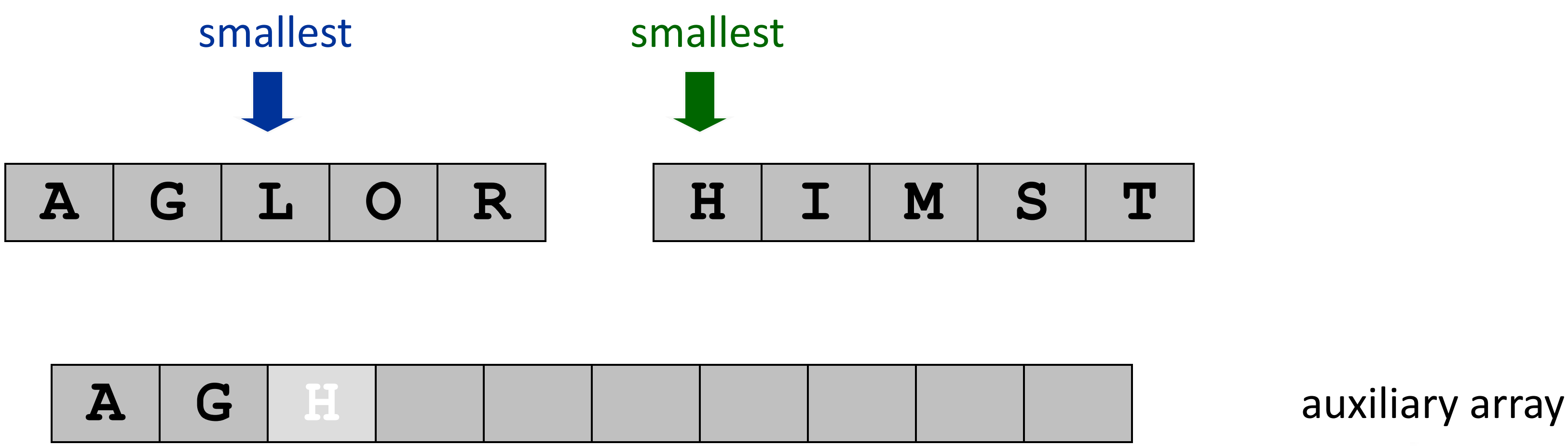
Merging Example

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



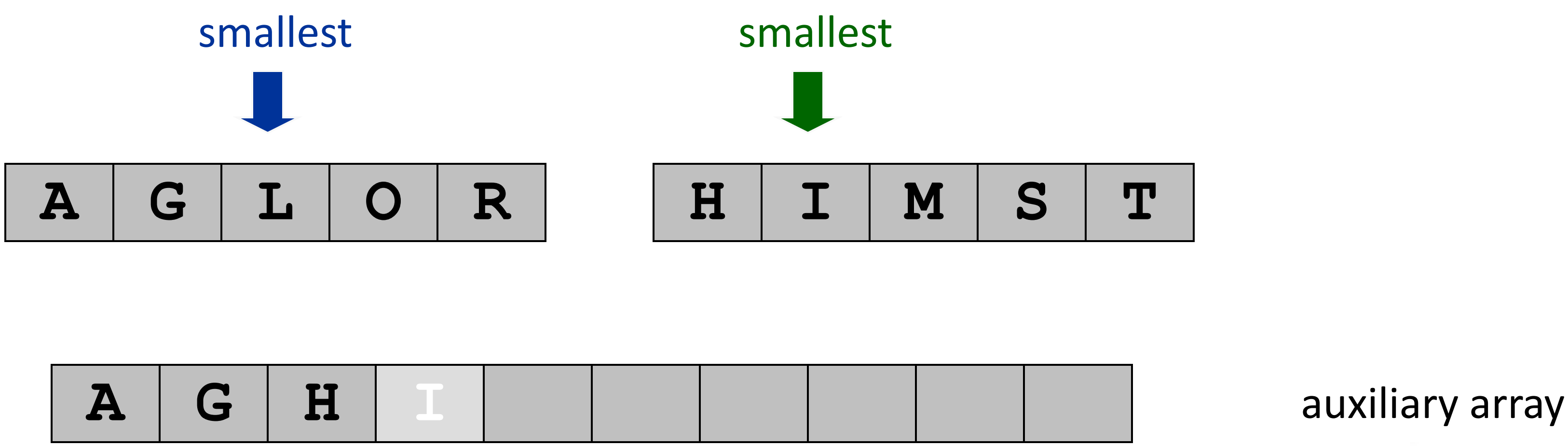
Merging Example

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



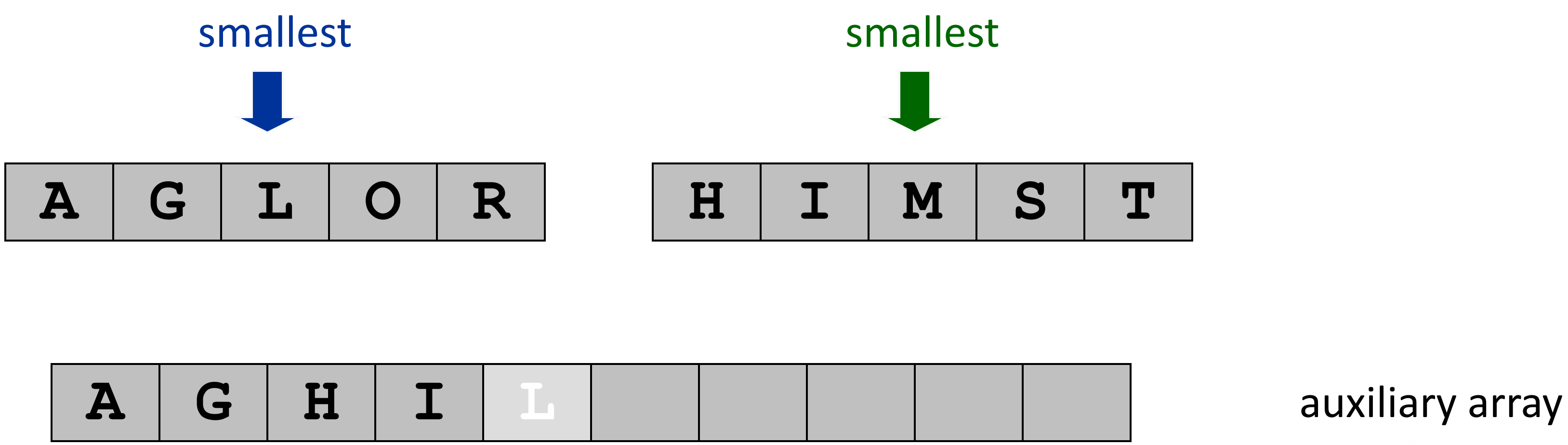
Merging Example

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



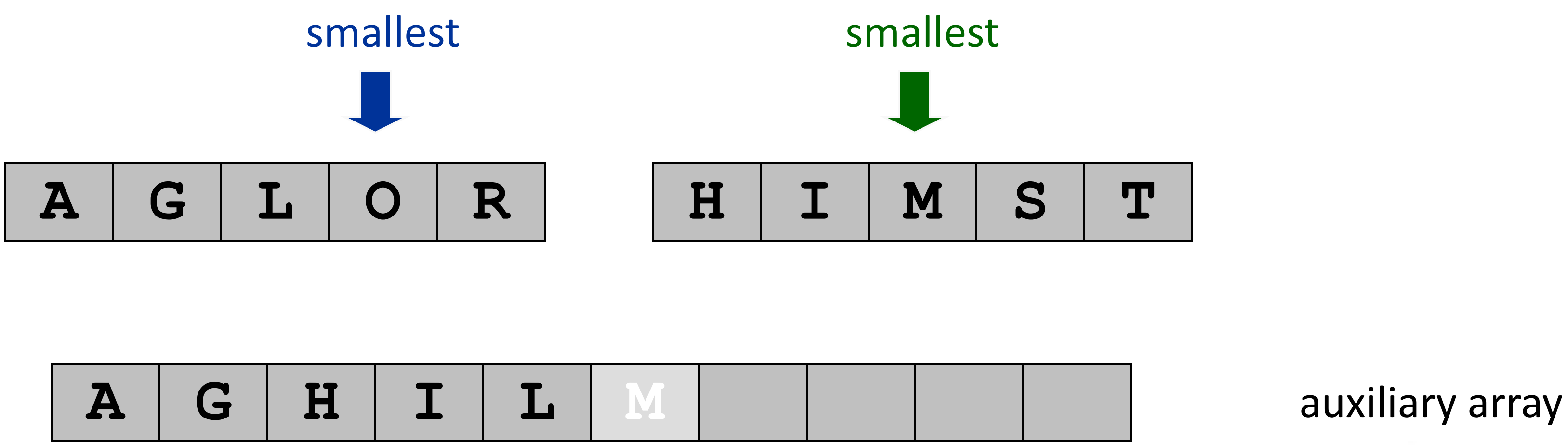
Merging Example

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



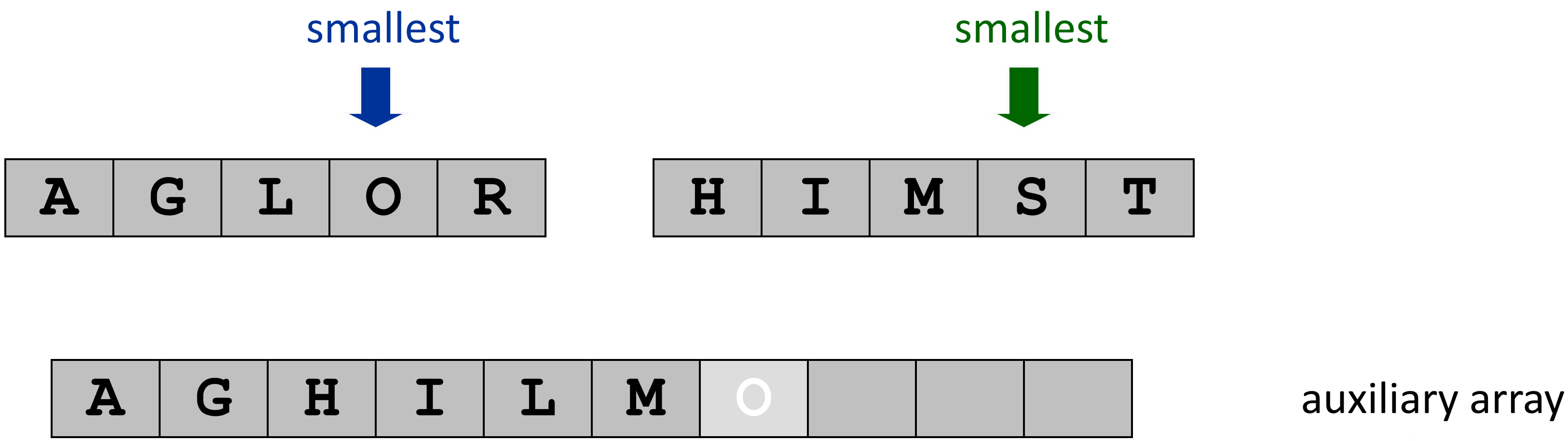
Merging Example

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



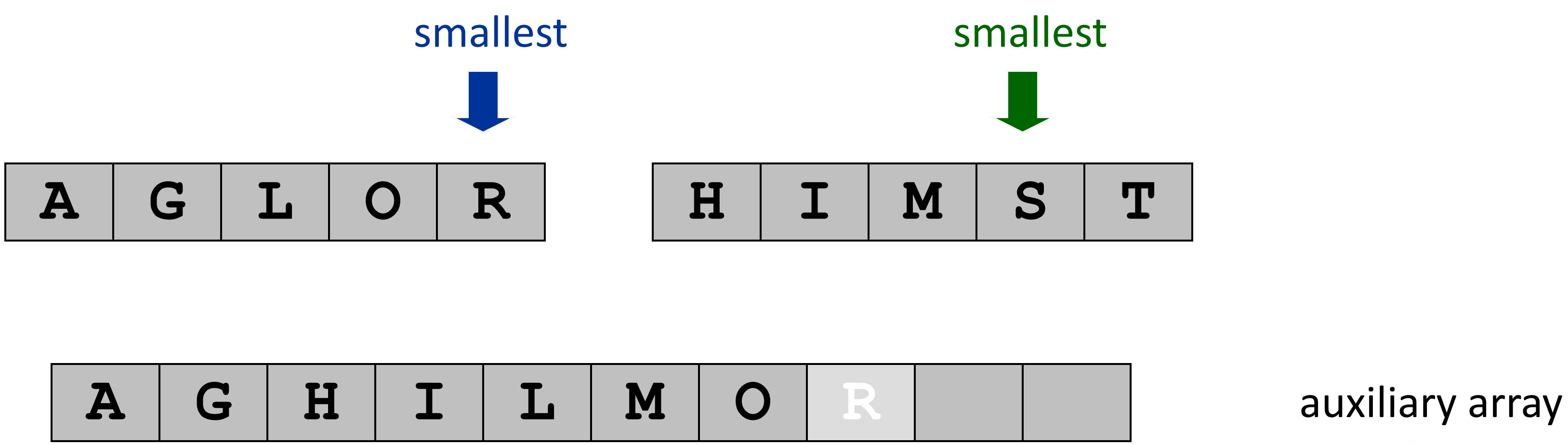
Merging Example

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



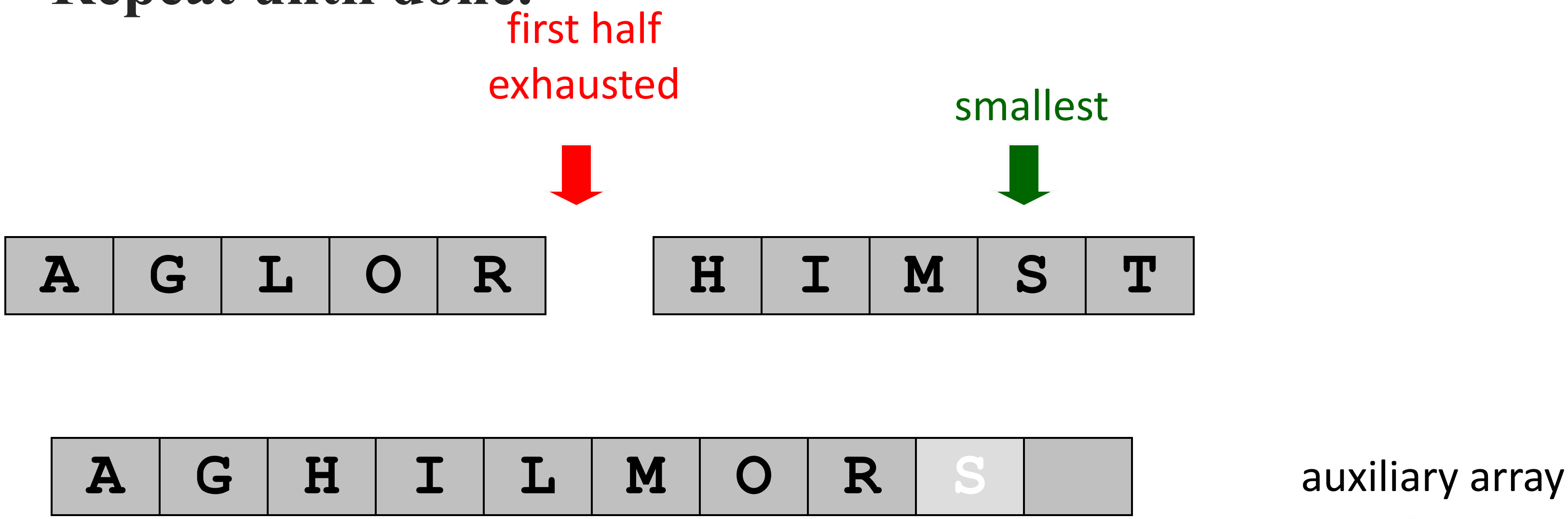
Merging Example

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



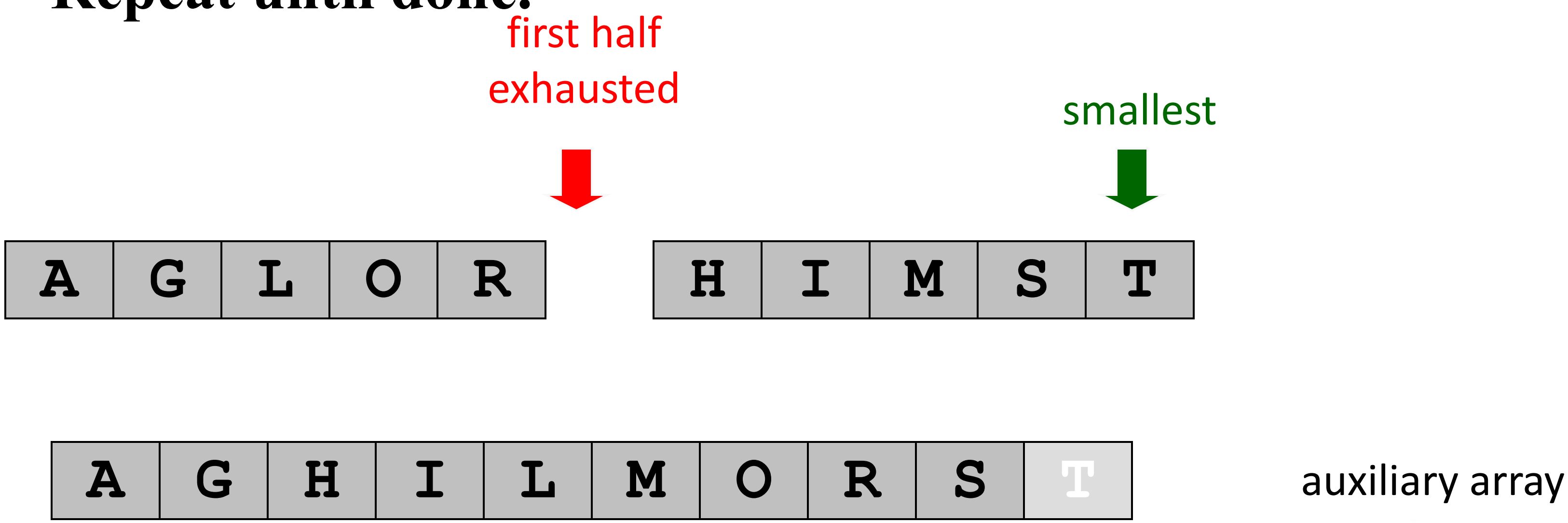
Merging Example

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



Merging Example

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



Merging Example

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

first half exhausted

second half exhausted

A	G	L	O	R

H	I	M	S	T

A	G	H	I	L	M	O	R	S	T

auxiliary array

Merge Sort – Combine with Insertion Sort

- **Consider the following is of unsorted array of 23 entries**

13	77	49	35	61	48	3	23	95	73	89	37	57	99	94	28	15	55	7	51	88	97	62

- **We will call insertion sort if the list being sorted is less than $n = 8$**

Merge Sort – Combine with Insertion Sort

- **Consider the following is of unsorted array of 23 entries**

13	77	49	35	61	48	3	23	95	73	89	37	57	99	94	28	15	55	7	51	88	97	62

- **The first and last entries are at indices first = 0 and last = 22**

- **We will split the list at midpoint = (0 + 22)/2, which equals 11 and recursively call merge sort on entries 0 through 11 and 12 through 22**

Merge Sort – Combine with Insertion Sort

- **We are now sorting positions 0 through 11, inclusive**

13	77	49	35	61	48	3	23	95	73	89	37	57	99	94	28	15	55	7	51	88	97	62

- **Again, we calculate midpoint = (0 + 11)/2, which equals 5 and recursively sort entries 0 through 5 and 6 through 11**

Merge Sort – Combine with Insertion Sort

- **This sub-list has only 6 entries, so we call insertion sort**

13	**77**	**49**	**35**	**61**	**48**	3	23	95	73	89	37	57	99	94	28	15	55	7	51	88	97	62

13	**35**	**48**	**49**	**61**	**77**	3	23	95	73	89	37	57	99	94	28	15	55	7	51	88	97	62

Merge Sort – Combine with Insertion Sort

- **This sub-list also has only 6 entries: call insertion sort**

13	35	48	49	61	77	**3**	**23**	**95**	**73**	**89**	**37**	57	99	94	28	15	55	7	51	88	97	62

13	35	48	49	61	77	**3**	**23**	**37**	**73**	**89**	**95**	57	99	94	28	15	55	7	51	88	97	62

Merge Sort – Combine with Insertion Sort

- These first two lists are now sorted, so we merge them:

13	35	48	49	61	77	3	23	37	73	89	95	57	99	94	28	15	55	7	51	88	97	62

3	13	23	35	37	48	49	61	73	77	89	95	57	99	94	28	15	55	7	51	88	97	62

Merge Sort – Combine with Insertion Sort

- We now proceed to the second half at positions 12 through 23

3	13	23	35	37	48	49	61	73	77	89	95	57	99	94	28	15	55	7	51	88	97	62

- The midpoint is at midpoint = (12 + 23)/2, which equals 17 and recursively call merge sort on entries 12 through 17 and 18 through 22

Merge Sort – Combine with Insertion Sort

- **The sub-list from 12 through 17 has 6 entries: call insertion sort**

3	13	23	35	37	48	49	61	73	77	89	95	**57**	**99**	**94**	**28**	**15**	**55**	7	51	88	97	62
3	13	23	35	37	48	49	61	73	77	89	95	**15**	**28**	**55**	**57**	**94**	**99**	7	51	88	97	62

Merge Sort – Combine with Insertion Sort

- **The sub-list from 18 through 22 has 5 entries: call insertion sort**

3	13	23	35	37	48	49	61	73	77	89	95	15	28	55	57	94	99	**7**	**51**	**88**	**97**	**62**

3	13	23	35	37	48	49	61	73	77	89	95	15	28	55	57	94	99	**7**	**51**	**62**	**88**	**97**

Merge Sort – Combine with Insertion Sort

- **Merge the two lists together:**

3	13	23	35	37	48	49	61	73	77	89	95	**15**	**28**	**55**	**57**	**94**	**99**	**7**	**51**	**62**	**88**	**97**

3	13	23	35	37	48	49	61	73	77	89	95	**7**	**15**	**28**	**51**	**55**	**57**	**62**	**88**	**94**	**97**	**99**

Merge Sort – Combine with Insertion Sort

- **Finally, merge both lists together:**

3	13	23	35	37	48	49	61	73	77	89	95	7	15	28	51	55	57	62	88	94	97	99

3	7	13	15	23	28	35	37	48	49	51	55	57	61	62	73	77	88	89	94	95	97	99

Exercise

- **1. Write the pseudo code of the merge-insertion sort described above.**

- **2. What is the computational complexity of merge-insertion sort?**

2.2 Recursion Analyzing

Analyzing Merge Sort

$T(n)$	**MERGE-SORT** $A[1 \,..\, n]$
$\Theta(1)$	1. If $n = 1$, done.
$2T(n/2)$	2. Recursively sort $A[\,1 \,..\, \lceil n/2 \rceil\,]$ and $A[\,\lceil n/2 \rceil + 1 \,..\, n\,]$.
$\Theta(n)$	3. ***"Merge"*** the 2 sorted lists

Cons.

Note: Should be $T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor)$,
but it turns out not to matter asymptotically.

Recurrence for Merge Sort

$$T(n) = \begin{cases} \Theta(1) \text{ if } n = 1; \\ 2T(n/2) + \Theta(n) \text{ if } n > 1. \end{cases}$$

- **Solution: the asymptotic running time of Merge Sort is $T(n) = \Theta(n \log n)$.**

- **We have several ways to prove this recurrence.**

Contents

- **2.2.1 Expansion**
- **2.2.2 Substitution**
- **2.2.3 Recursion Tree**

2.2.1 Expansion

Expansion Method

- **Sometimes the expression of the recurrence is very simple.**

- **Thus, we can expand the recurrence expression by replacing the current term with the decreasing-input-terms directly.**

Expansion Method

- **E.g., given the following recurrence :**
- **$T(n) = T(n-1) + \Theta(n)$, $T(1) = \Theta(1)$**

$$
\begin{aligned}
T(n) &= T(n-1) + c_1 n \\
&= \big(T(n-2) + c_1(n-1)\big) + c_1 n \\
&= T(n-2) + c_1 n + c_1(n-1) \\
&= T(n-3) + c_1 n + c_1(n-1) + c_1(n-2) \\
&\ \vdots \\
&= T(1) + c_1 \sum_{i=2}^{n} i = c_2 + c_1 \sum_{i=2}^{n} i \\
&= c_1 \sum_{i=1}^{n} i + (c_2 - c_1) = c_1 \frac{n(n+1)}{2} + (c_2 - c_1) \Rightarrow T(n) = \Theta(n^2)
\end{aligned}
$$

Expansion Method

- What if
  T(n) = T(n – 1) + O(n), T(1) = O(1)?

- Thus, we can only get the upper bound.

$$T(n) \le T(n-1) + c_1 n$$

$$\le c_1 \frac{n(n+1)}{2} + (c_2 - c_1) \Rightarrow T(n) = O(n^2)$$

Apply Expansion to Merge Sort

$$T(n) = \begin{cases} \Theta(1) \text{ if } n = 1; \\ 2T(n/2) + \Theta(n) \text{ if } n > 1. \end{cases}$$

$$\begin{aligned}
\mathrm{T}(n) &= 2\,\mathrm{T}(\frac{n}{2}) + c_1 n \\
&= 4\,\mathrm{T}(\frac{n}{4}) + c_1 \frac{n}{2} \times 2 + c_1 n \\
&= 8\,\mathrm{T}(\frac{n}{8}) + c_1 \frac{n}{4} \times 4 + c_1 \frac{n}{2} \times 2 + c_1 n \\
&= \ldots \\
&= 2^k\,\mathrm{T}(\frac{n}{2^k}) + c_1 \left(\frac{n}{2^{k-1}} \times 2^{k-1} + \ldots + \frac{n}{2} \times 2 + n \right)
\end{aligned}$$

Apply Expansion to Merge Sort

$$\textbf{if } \frac{n}{2^k} = 1 \textbf{ then}$$

$$\mathrm{T}(n) = c_2 2^k + c_1\left(\frac{n}{2^{k-1}} \times 2^{k-1} + \ldots + \frac{n}{2} \times 2 + n\right)$$

$$\frac{n}{2^k} = 1$$

$$\Rightarrow k = \log n$$

$$\Rightarrow 2^{\log n} c_2 + \left(\underbrace{\frac{n}{2^{k-1}} \times 2^{k-1} + \ldots + \frac{n}{2} \times 2 + n}_{n \log n}\right) c_1$$

$$\Rightarrow T(n) = \Theta(c_2 n + c_1 n \log n) = \Theta(n \log n)$$

Exercise in Class

- $T(n) = 4T(n/2) + \Theta(n)$, $\mathrm{T}(1) = \Theta(1)$
- **Solve the above recurrence through expansion.**

2.2.2 Substitution

Substitution method

- **The most general method.**

    1. ***Guess*** **the form of the solution.**

    2. ***Verify*** **by induction.**

    3. ***Solve*** **for constants.**

Example of substitution

- **Example:** $T(n) = 4T(n/2) + \Theta(n)$, $T(1) = \Theta(1)$
  - **Guess** $O(n^3)$.
  - **Assume that** $T(n) \leq c_1 n^3$ **for** $n \geq n_0$.
  - **Prove** $T(n) \leq c_1 n^3$ **by induction.**

Example of substitution

$$\begin{aligned}
\mathrm{T}(n) &= 4\,\mathrm{T}(n/2) + c_2 n \\
&\le 4c_1 (n/2)^3 + c_2 n \\
&= (c_1/2) n^3 + c_2 n \\
&= c_1 n^3 + (c_2 n - (c_1/2) n^3)
\end{aligned}$$

$$\textit{If}\ \mathrm{T}(n) \le c_1 n^3$$

$$\textit{then}\ (c_2 n - (c_1/2) n^3) \le 0$$

$$\Rightarrow \text{holds for } n^2 \ge \frac{2c_2}{c_1}, \text{e.g., } c_2 = 1, c_1 = 2 \text{ and } n \ge 1$$

- ***This is not a tight bound: We cannot prove the tightness!***

Example of substitution

- $T(n) = 4T(n/2) + \Theta(n)$, $T(1) = \Theta(1)$
  - $O(n^3)$ is proven.
  - How about we want to prove $\Theta(n^3)$?
  - We need to prove $\Omega(n^3)$ and $O(n^3)$
  - Prove $T(n) \leq c_1 n^3$ and $T(n) \geq c_3 n^3$ for $n \geq n_0$ simultaneously.

Example of substitution

$$
\begin{aligned}
\mathrm{T}(n) &= 4\,\mathrm{T}(n/2) + c_2 n \\
&\geq 4c_1\ (n/2)^3 + c_2 n \\
&= (c_1/2)n^3 + c_2 n \\
&= c_1 n^3 + (c_2 n - (c_3/2)n^3)
\end{aligned}
$$

$$\boldsymbol{\mathit{If}\ \mathrm{T}(n) \geq c_1 n^3}$$

$$\boldsymbol{\mathit{then}\ (c_2 n - (c_3/2)n^3) \geq 0}$$

$$\textbf{thus } \boldsymbol{n^2 \leq \frac{2c_2}{c_1}}$$

$$\boldsymbol{\Rightarrow \textbf{can not hold since } n \to \infty \textbf{ and } 0 < c_1 < \infty}$$

Example of substitution

- Then for $T(n) = 4T(n/2) + \Theta(n)$, $T(1) = \Theta(1)$
  - Can we prove $T(n) = \Theta(n^2)$?
  - Then we should prove $T(n) = O(n^2)$ and $T(n) = \Omega(n^2)$ for $n \geq n_0$
  - We firstly prove $T(n) = O(n^2)$, and we choose to prove $T(n) \leq cn^2$

Example of substitution

$$
\begin{aligned}
\mathrm{T}(n) &= 4\,\mathrm{T}(n/2) + dn \\
&\le 4c(n/2)^2 + dn \\
&= cn^2 + dn
\end{aligned}
$$

- **Can we say that we have proven our inductive hypothesis (I.H.) which is denoted by $\mathrm{T}(n) \le cn^2$?**
- **NO, WE CANNOT**
- **Since we have to prove the EXACT form of the I.H!**
- **Thus, the above proof fails!**

Example of substitution

- **Idea: strengthen the inductive hypothesis,** ***by subtracting*** **a low-order term.**

  I.H.: $T(n) \le c_1 n^2 - c_2 n$ for $n \ge n_0$.

  Proof **:**

$$
\begin{aligned}
\mathrm{T}(n) &= 4\,\mathrm{T}(n/2) + dn \\
&\le 4\left(c_1 (n/2)^2 - c_2 (n/2)\right) + dn \\
&= c_1 n^2 - 2c_2 n + dn \\
&= c_1 n^2 - c_2 n + (d - c_2) n
\end{aligned}
$$

$\Rightarrow$ *If* $\mathrm{T}(n) \le c_1 n^2 - c_2 n$

then $(d - c_2) < 0$

$\Rightarrow$ holds for $d < c_2$

Example of substitution

- **Then for $T(n) = 4T(n/2) + \Theta(n)$, $\mathrm{T}(1) = \Theta(1)$**
  - **We prove $\mathrm{T}(n) = \Omega(n^2)$ by proving**
    $T(n) \geq c_3 n^2 - c_4 n$ for $n \geq n_0$

$$
\begin{aligned}
\mathrm{T}(n) &= 4\,\mathrm{T}(n/2) + dn \\
&\geq 4\left(c_3 (n/2)^2 - c_4 (n/2)\right) + dn \\
&= c_3 n^2 - 2c_4 n + dn \\
&= c_3 n^2 - c_4 n + (d - c_4)n
\end{aligned}
$$

$If\ \mathrm{T}(n) \geq c_3 n^2 - c_4 n$

$\Rightarrow$ $then\ (d - c_4) > 0$

$\Rightarrow$ holds for $d > c_4$

Example of substitution

- **Thus, for $T(n) = 4T(n/2) + \Theta(n)$, $T(1) = \Theta(1)$,**
  - **We achieve that $T(n) = \Theta(n^2)$**

Apply Substitution to Merge Sort

$$T(n) = \begin{cases} \Theta(1) \text{ if } n = 1; \\ 2T(n/2) + \Theta(n) \text{ if } n > 1. \end{cases}$$

- Guess $\Theta(n \log n)$.
- Assume that
  $T(n) \le c_1 \cdot n \log n$ and $T(n) \ge c_2 \cdot n \log n$
  for $n \ge n_0$.
- Prove
  $T(n) \le c_1 \cdot n \log n$ and $T(n) \ge c_2 \cdot n \log n$
  by induction.

Apply Substitution to Merge Sort

- ***Proof* :**

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

$$\leq 2c_1 \cdot (n/2) \cdot \log(n/2) + dn$$

$$= c_1 n \cdot (\log n - 1) + dn$$

$$= c_1 n \log n - (c_1 - d)n$$

$\rightarrow$ *if* $T(n) \leq c_1 n \log n$ *for* $n \geq n_0$

then $(c_1 - d)n \geq 0$

$\rightarrow$ *holds for* $c_1 \geq d$

$\rightarrow$ $T(n) = O(n \log n)$ *is proven.*

Apply Substitution to Merge Sort

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

$$\geq 2c_2 \cdot (n/2) \cdot \log(n/2) + dn$$

$$= c_2 n \cdot (\log n\text{-}1) + dn$$

$$= c_2 n \log n - (c_2 - d)n$$

$\rightarrow$ *if* $T(n) \geq c_2 n \log n$ *for* $n \geq n_0$

then $(c_2 - d)n \leq 0$

$\rightarrow$ *holds for* $c_2 \leq d$

$\rightarrow$ $T(n) = \Omega(n \log n)$ *is proven.*

Thus, we have achieved that $\mathrm{T}(n) = \Theta(n \log n)$

Exercise in Class

- For $T(n) = 4T(n/2) + \Theta(n)$, $T(1) = \Theta(1)$
  - Can we prove that $T(n) = O(n)$?

2.2.3 Recursion Tree

Recursion-tree Method

- Sometimes a good I.H. is intractable through guessing.
- Fortunately, we can draw the recursion tree to help us obtain the I.H.
- However, after achieving the I.H., we still need to prove the correctness of this I.H. by substitution.

Example of Recursion-tree

- **Solve** $T(n) = T(n/4) + T(n/2) + \Theta(n^2)$, $T(1) = \Theta(1)$

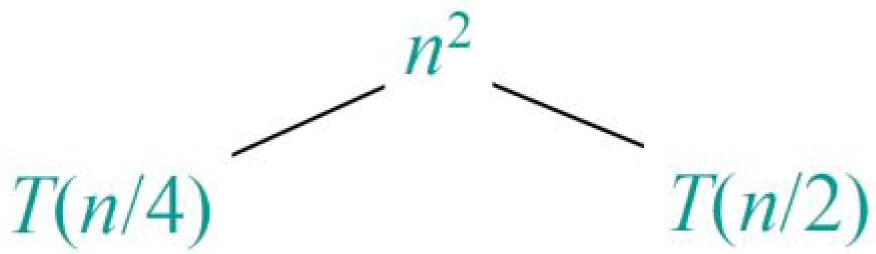
Example of Recursion-tree

- **Solve** $T(n) = T(n/4) + T(n/2) + \Theta(n^2)$, $T(1) = \Theta(1)$

$$T(n)$$

Example of Recursion-tree

- **Solve** $T(n) = T(n/4) + T(n/2) + \Theta(n^2)$, $T(1) = \Theta(1)$



Example of Recursion-tree

- **Solve** $T(n) = T(n/4) + T(n/2) + \Theta(n^2)$, $T(1) = \Theta(1)$

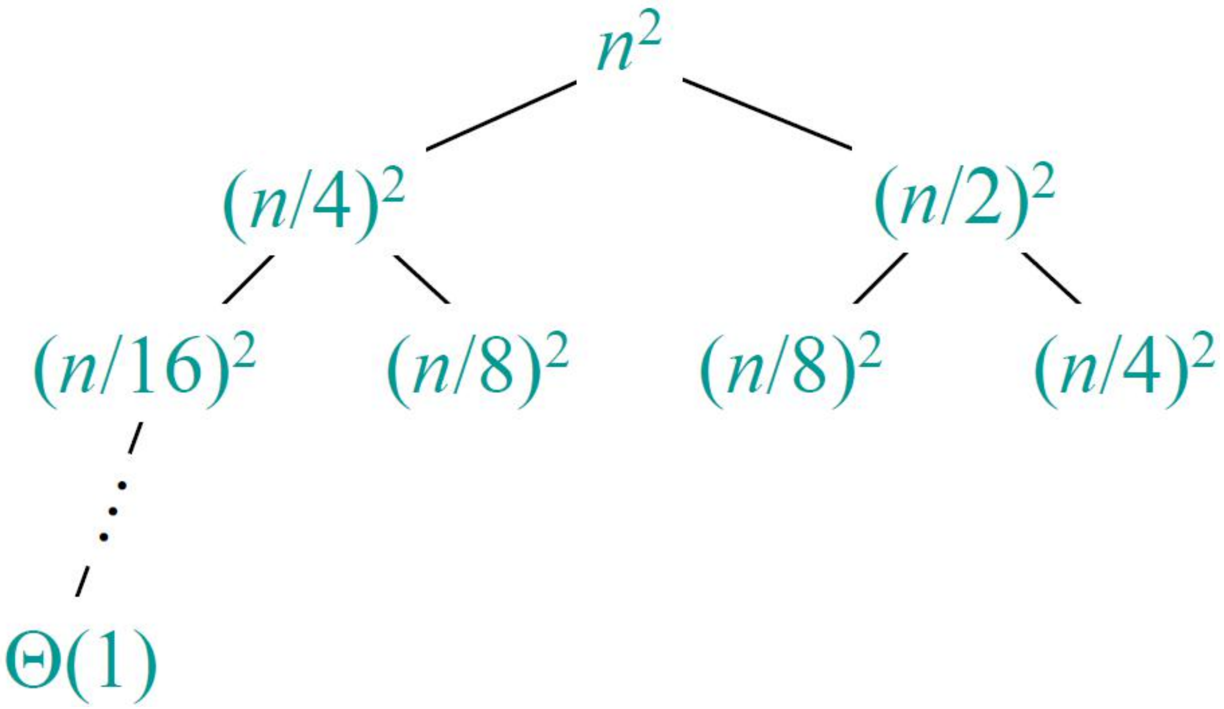
$$n^2$$

$$(n/4)^2 \qquad\qquad (n/2)^2$$

$$T(n/16) \quad T(n/8) \qquad T(n/8) \quad T(n/4)$$

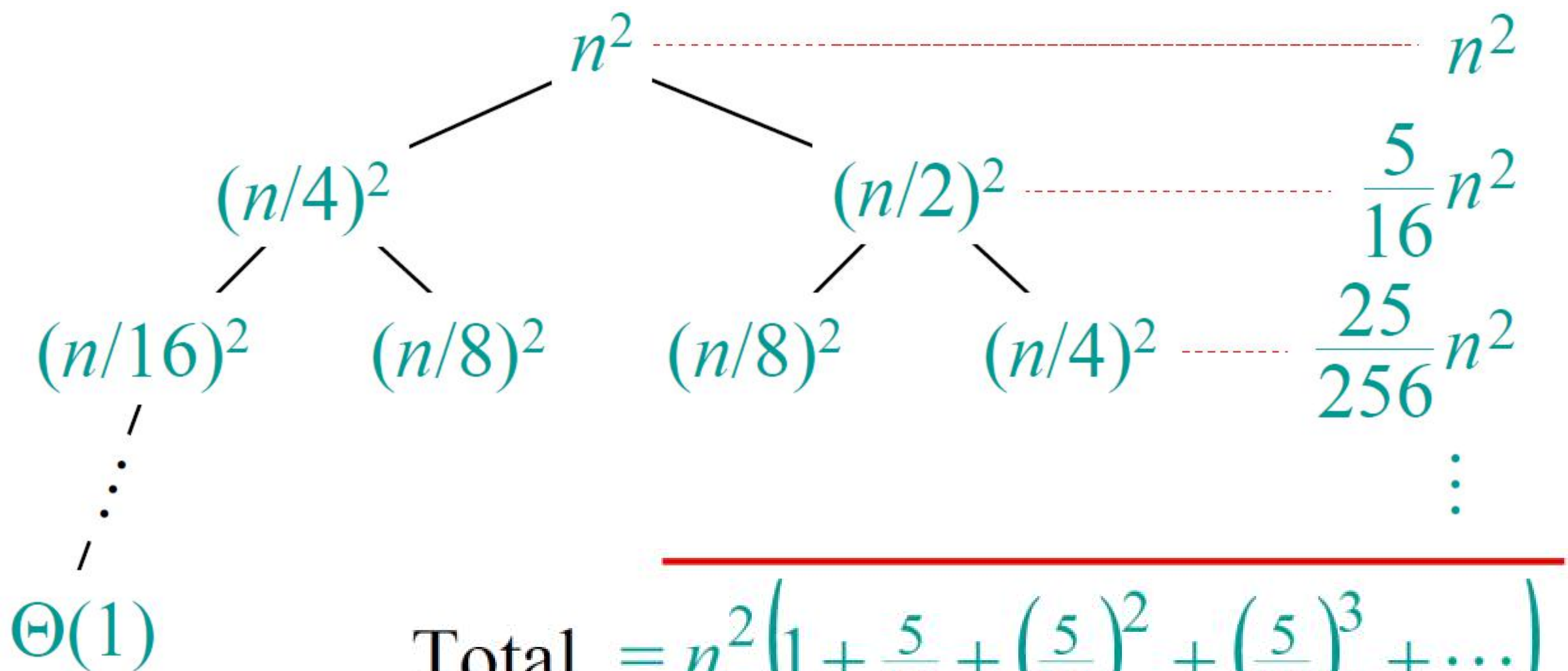
Example of Recursion-tree

- **Solve** $T(n) = T(n/4) + T(n/2) + \Theta(n^2)$, $T(1) = \Theta(1)$



Example of Recursion-tree

- **Solve** $T(n) = T(n/4) + T(n/2) + \Theta(n^2)$, $T(1) = \Theta(1)$

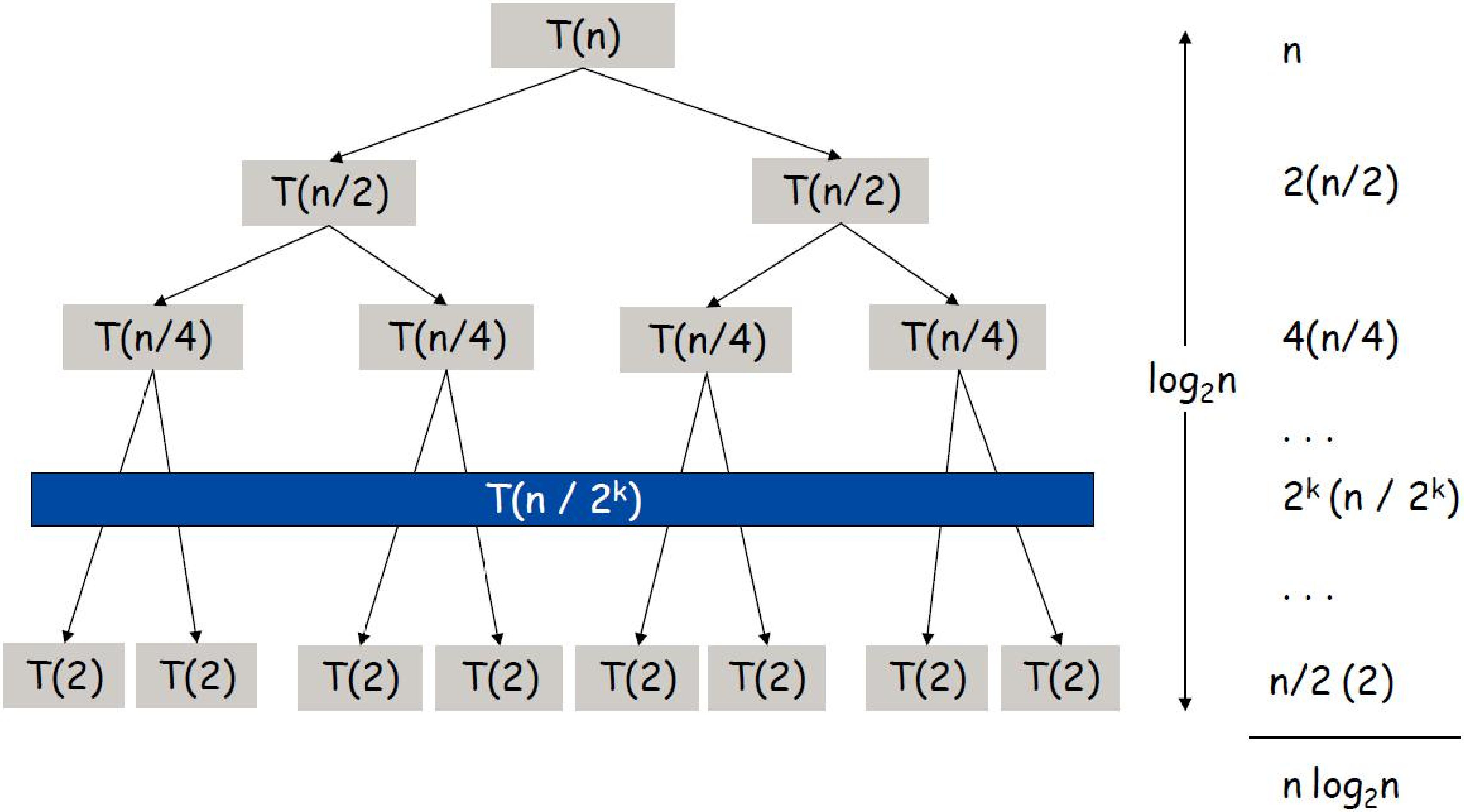


$$\text{Total} = n^2\left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^3 + \cdots\right)$$

$$= \Theta(n^2) \qquad \textit{geometric series}$$

Apply Recursion-tree to Merge Sort

$$T(n) = \begin{cases} \Theta(1) \text{ if } n = 1; \\ 2T(n/2) + \Theta(n) \text{ if } n > 1. \end{cases}$$



Run-time Summary of Merge Sort

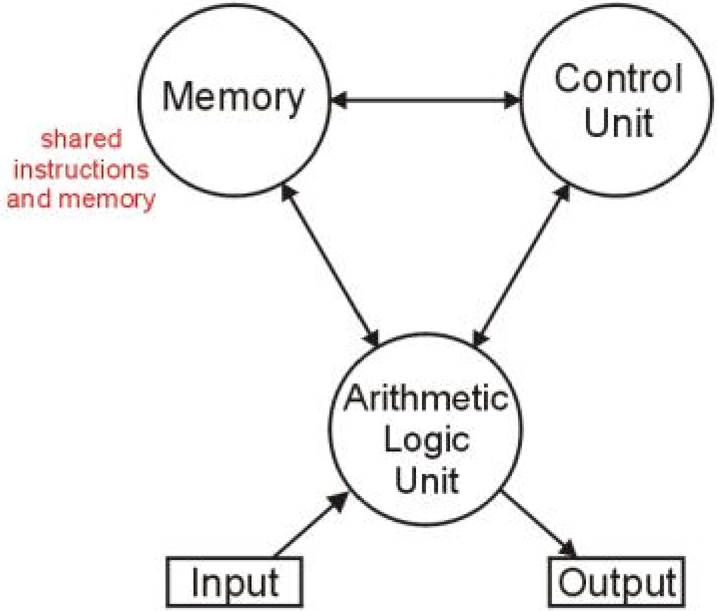
- The following table summarizes the run-times of merge sort

Case	Run Time	Comments
Worst	$\Theta(n \log(n))$	No worst case
Average	$\Theta(n \log(n))$	
Best	$\Theta(n \log(n))$	No best case

- How can merge sort always have the computational complexity at $\Theta(1)$?

Aside

- **The (likely) first implementation of merge sort was on the ENIAC in 1945 by John von Neumann**
- **The creator of the von Neumann architecture used by all modern computers:**



Exercise in Class

For

$$T(n) = T(n/4) + T(n/2) + \Theta(n^2),$$
$$T(1) = \Theta(1)$$

Prove that $T(n) = \Theta(n^2)$ through substitution.

Exercises

- CLRS 4.2-2
- CLRS 4.2-5

算法分析课程组

重庆大学计算机学院

End of Section.