《数据结构与算法》课程组

重庆大学计算机学院

Data Structures & Algorithms

3 MERGE SORT AND RECURSION

Outline

3.1 Merge Sort

3.2 Recursion Analyzing

3.1 Merge Sort

Divide-and-Conquer

- Divide-and-conquer paradigm
  - **Divide** the problem into a number of subproblems.
  - **Conquer** the subproblems by solving them recursively. If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner.
  - **Combine** the solutions to the subproblems into the solution for the original problem.

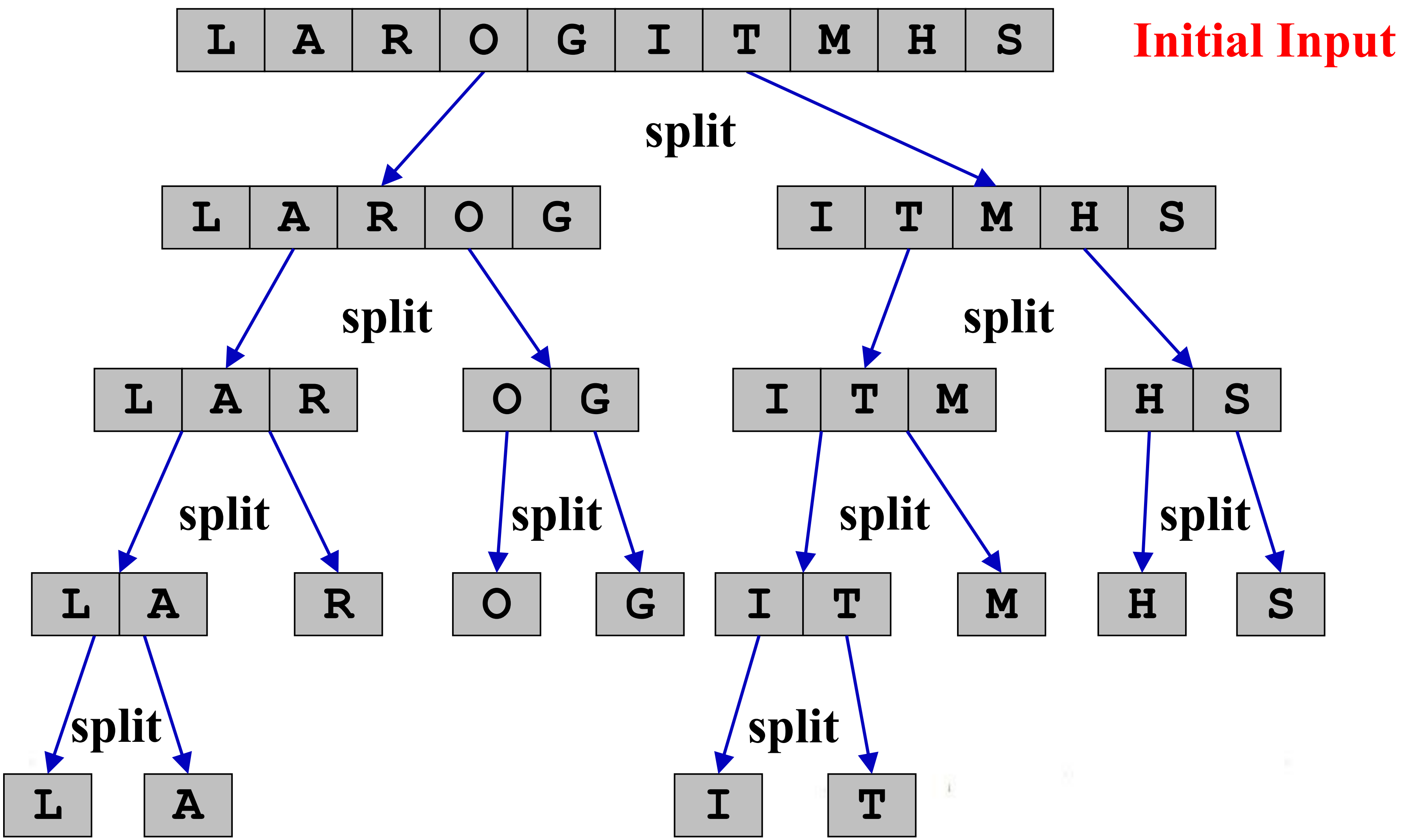
Merging Sort

- **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**

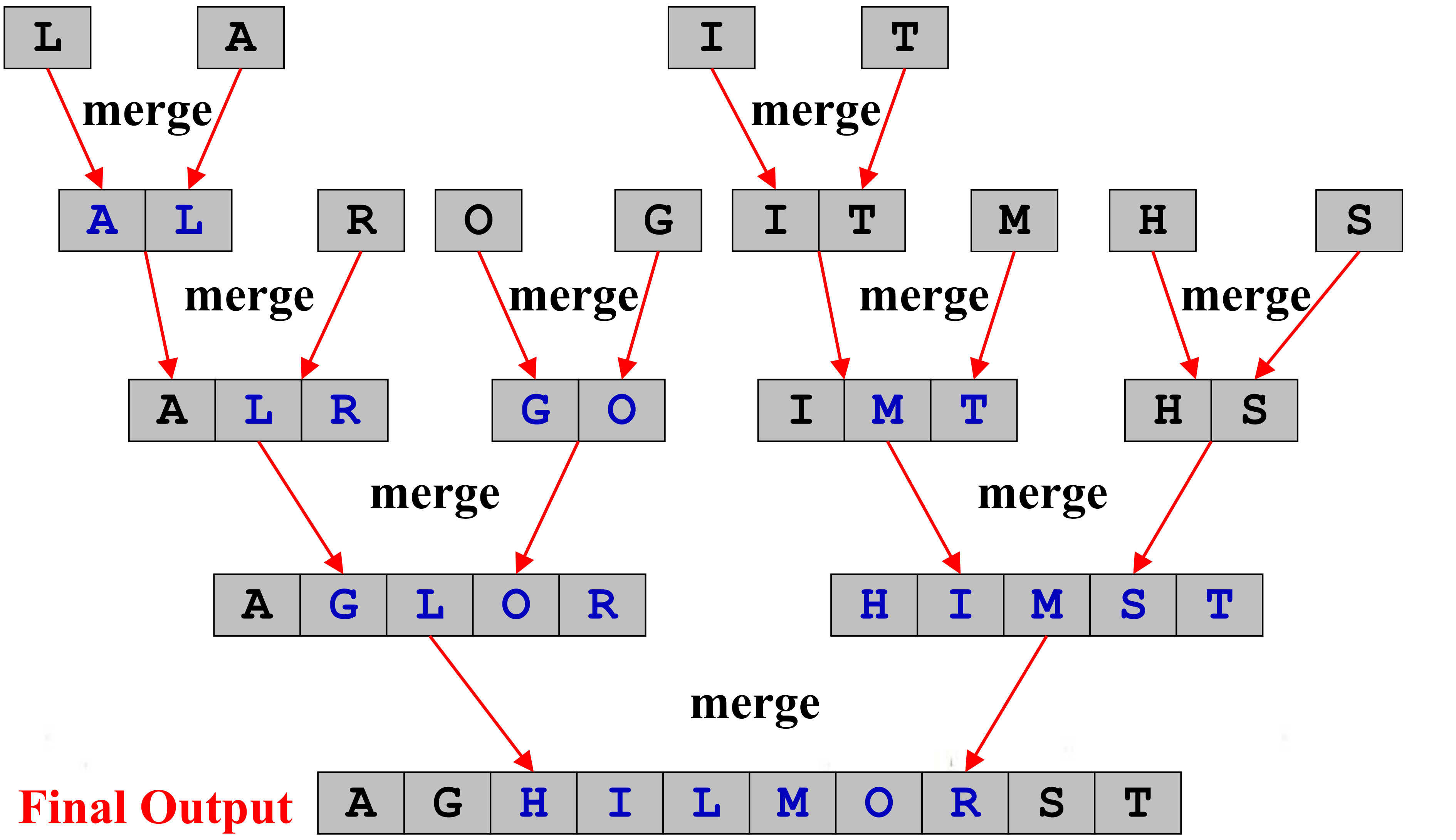
Merge Sort Algorithm: General Case

- Problem:
  - Input: A[l…r]
  - Output: A[l…r] in sorted order
- MERGE-SORT(A, l, r)
  - **if** $l < r$
    - **then** $m \leftarrow \lfloor (p + r)/2 \rfloor$
      - MERGE-SORT(A, l, m)
      - MERGE-SORT(A, $m + 1$, r)
      - MERGE(A, l, m, r)

Merge Sort - Split



Merge Sort - Merge



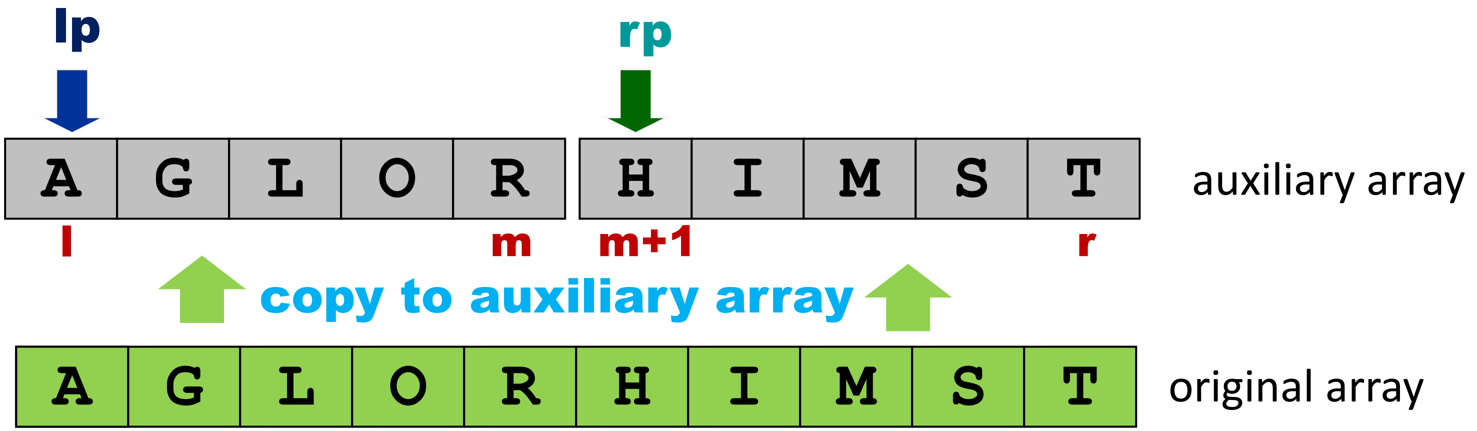
Pseudo Code of the Merge Procedure

```
MERGE(A, l, m, r)
Create array B[l..r]                        Θ(1)
for i ← l to r
        do  B[i] ← A[i]                     Θ(n)
lp ← l
rp ← m + 1                                  Θ(1)
for i ← l to r do
        if lp > m
                then A[i] ← B[rp]
                        rp ← rp + 1
        else if  rp > r
                then A[i] ← B[lp]
                        lp ← lp + 1                  Θ(n)
        else if comp::prior(B[lp], B[rp])
                then A[i] ← B[lp]
                        lp ← lp + 1
        else  A[i] ← B[rp]
                rp ← rp + 1
```

$\boxed{\text{Time complexity: } \Theta(n)}$

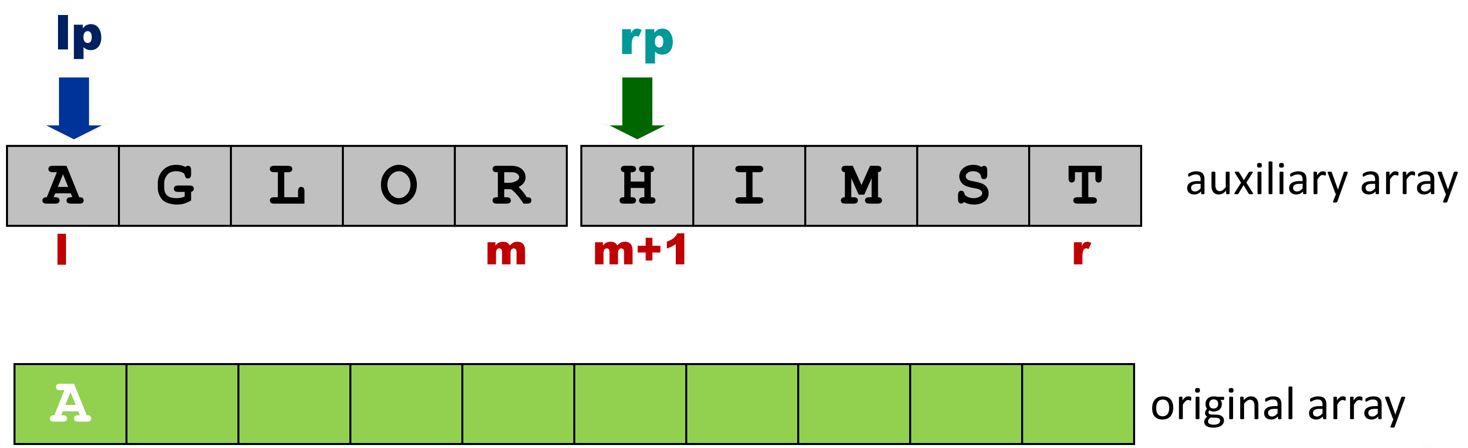
Merging Example

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



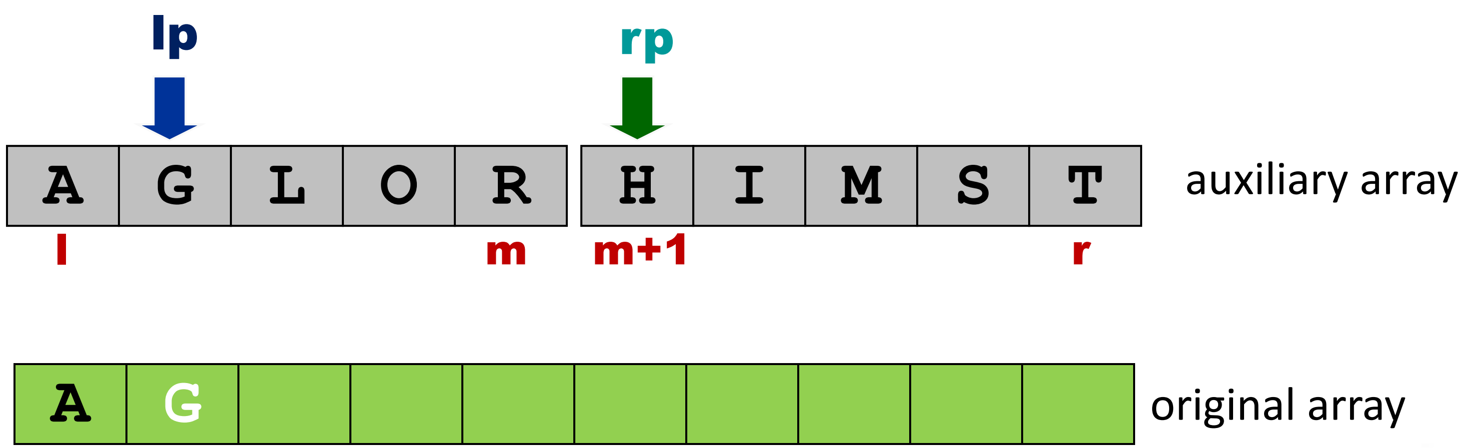
Merging Example

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



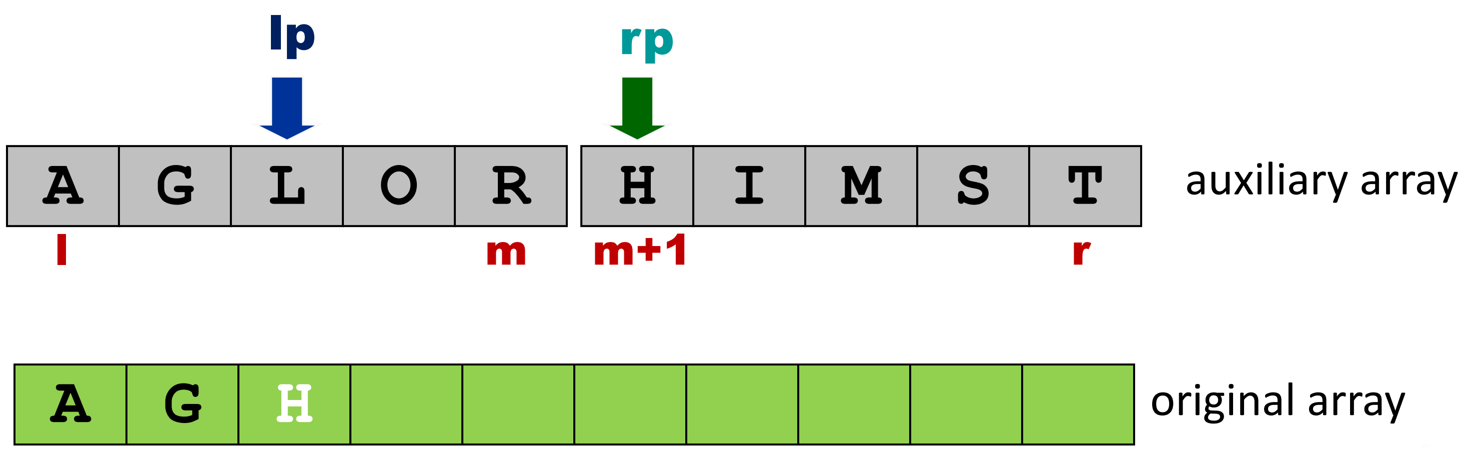
Merging Example

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



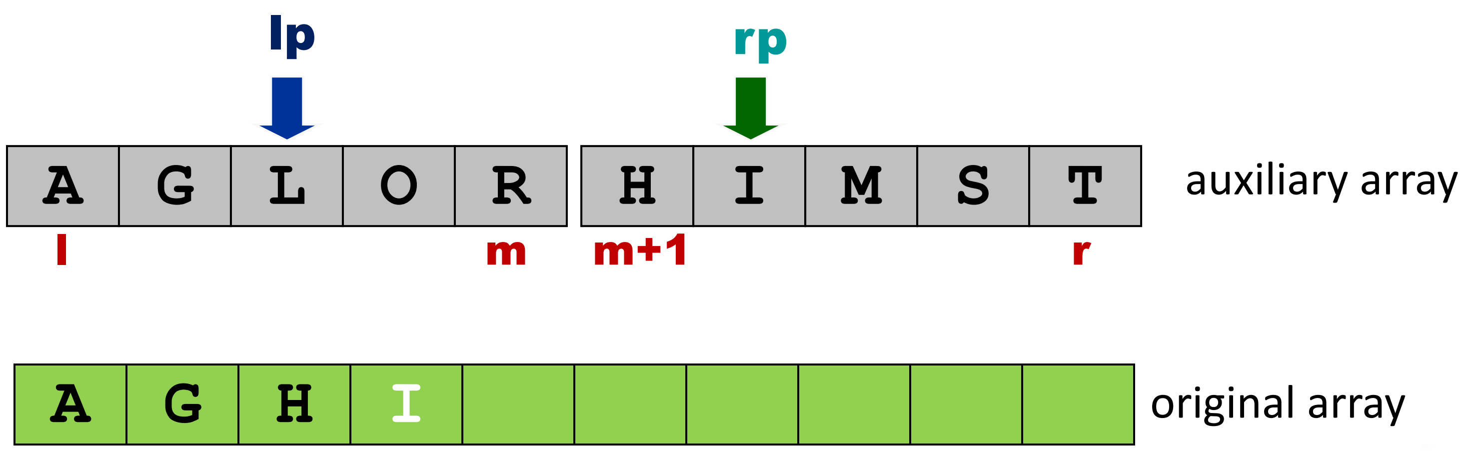
Merging Example

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



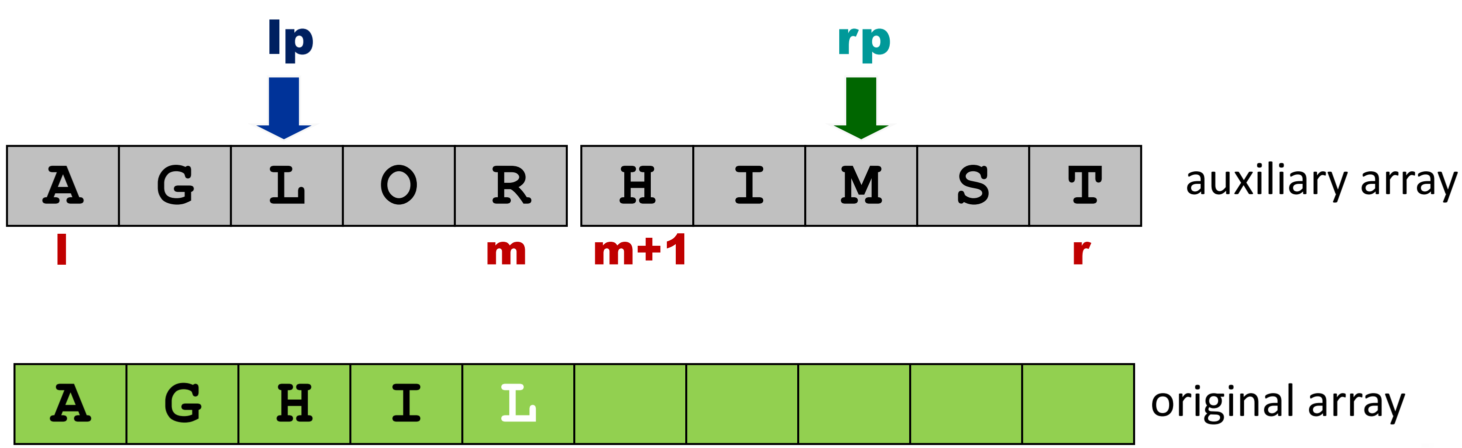
Merging Example

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



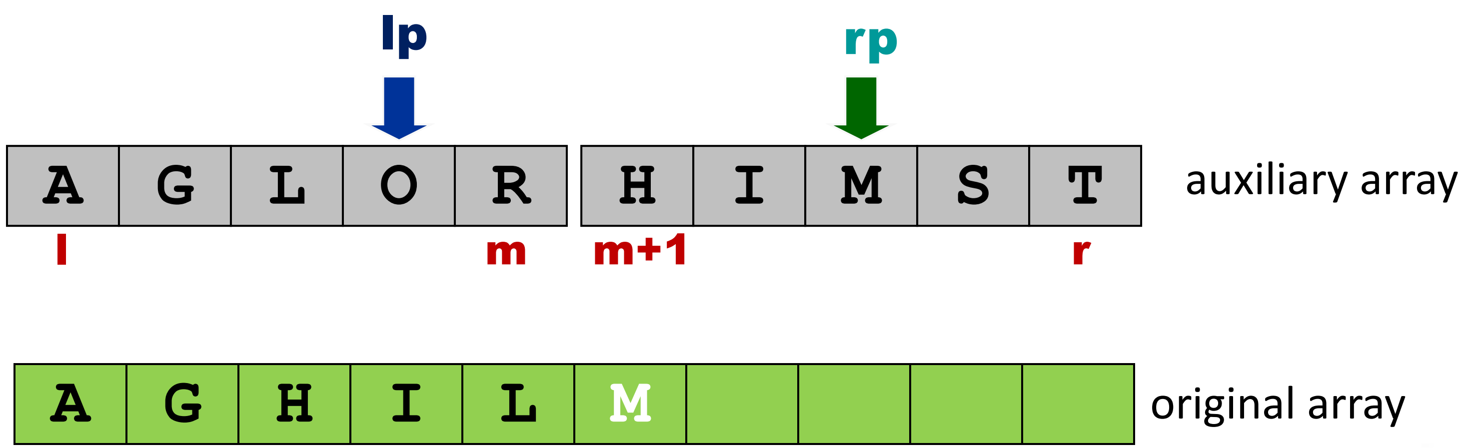
Merging Example

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



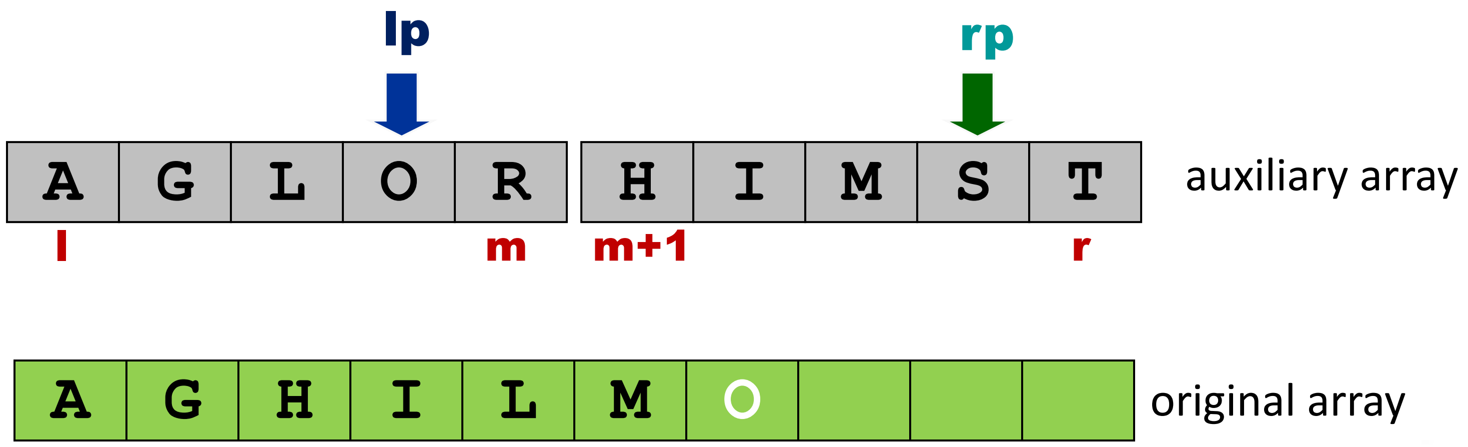
Merging Example

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



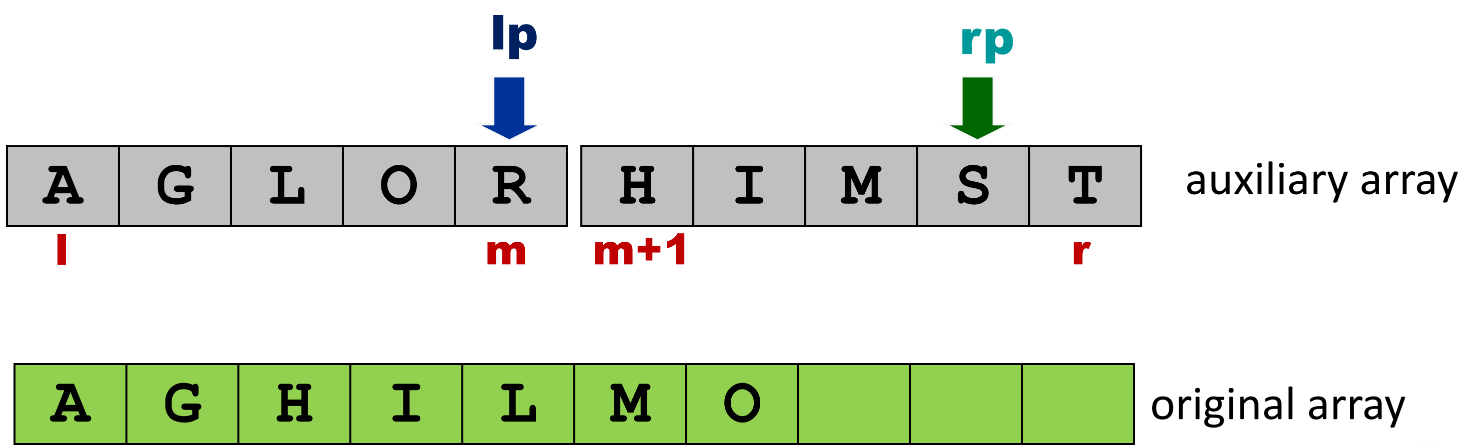
Merging Example

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



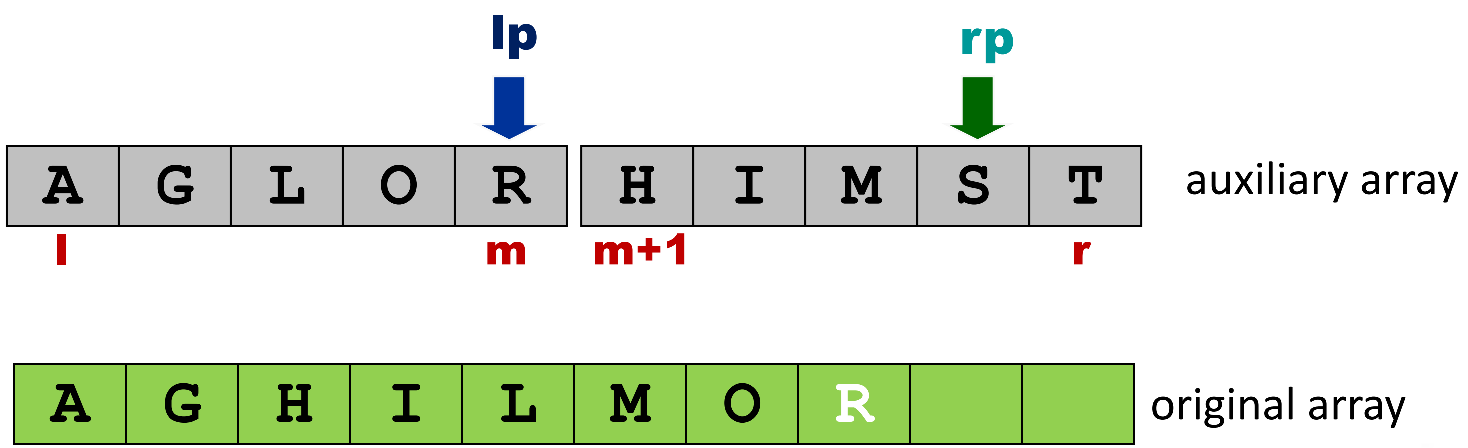
Merging Example

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



Merging Example

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



Merging Example

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

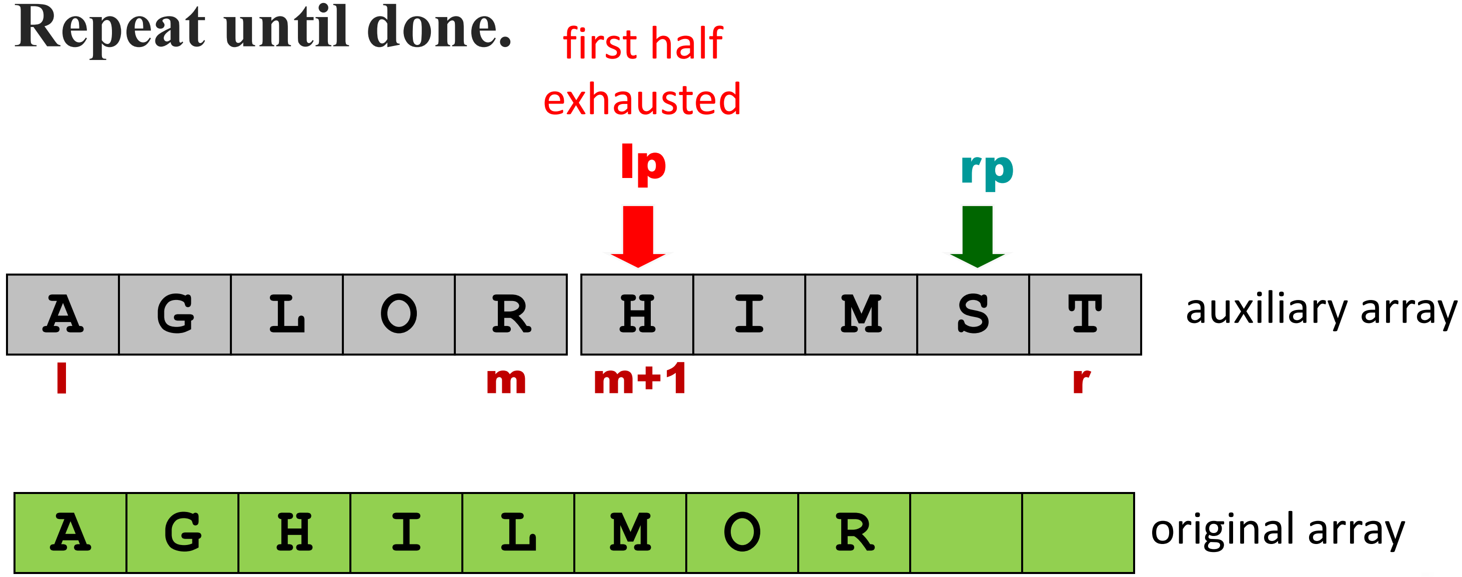
first half exhausted

								lp				rp		
auxiliary array	A	G	L	O	R	H	I	M	S	T
	l				m	m+1				r

original array	A	G	H	I	L	M	O	R	S	

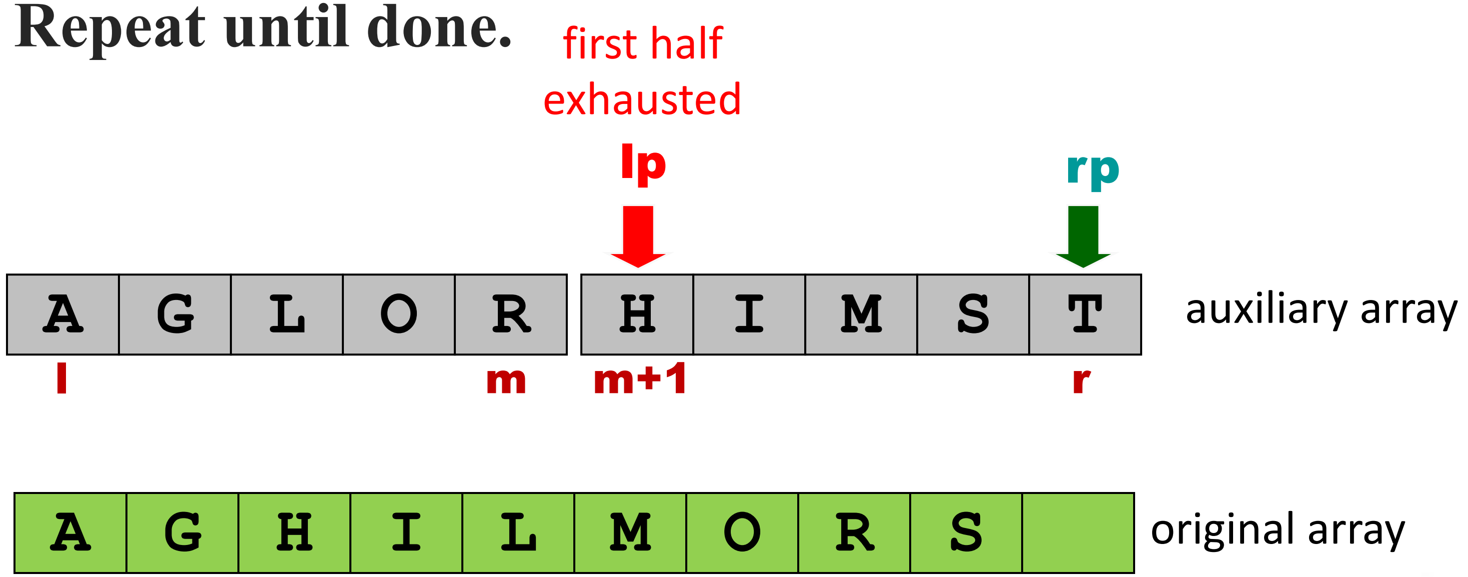
Merging Example

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



Merging Example

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



Merging Example

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

first half exhausted

lp

second half exhausted

rp

auxiliary array

A	G	L	O	R	H	I	M	S	T
l				m	m+1				r

original array

A	G	H	I	L	M	O	R	S	T

经典面试题

- 长度为n和m的两个数组(n>m)。第一个数组的前面连续排放n-m个整数且从小到大排序,其后是长度为m的空置空间(null space)。第二个数组中的m个整数也按升序排列。现将第二个数组中的元素合并至第一个数组并构成升序序列，且合并过程不使用额外的数组，即空间复杂度O(1)。

<1,2,4,6,7,_,_,_,_>
<2,4,7,9>

<1,2,2,4,4,6,7,7,9>

经典面试题

在一个排列中，如果一对数的前后位置与大小顺序相反，它们就称为一个逆序(inversion)。一个排列中逆序的总数就称为这个排列的逆序数。

- 定义：数组$a[0..n-1]$，数对$a[i]$和$a[j]$是逆序的充分必要条件是 $0 \le i < j < n \land a[i] > a[j]$
- 比如序列<2, 4, 3, 1>中，数据对<2,1>，<4,3>，<4,1>，<3,1>是逆序，逆序数是4。

方法1. 枚举所有数据对并判断是否逆序

时间复杂度：$\Theta(n^2)$

经典面试题

方法2. 归并排序　　时间复杂度：$\Theta(n\log(n))$

- 对子数组$a\left[0..\frac{n}{2}-1\right]$ 和$a\left[\frac{n}{2}..n-1\right]$ 排序并求各自的逆序数

- Merge $\begin{cases} a[0], a[1], \ldots, a[p-1], a[p], a[p+1], \ldots, a[\frac{n}{2}-1] & p\in\left[0,\frac{n}{2}\right) \\ a\left[\frac{n}{2}\right], a\left[\frac{n}{2}+1\right], \ldots\ldots, a[q], \ldots\ldots, a[n-1] & q\in\left[\frac{n}{2}, n\right) \end{cases}$

Q1: 如果合并过程中发生$a[p]$和$a[q]$的比较，表明什么？

A1: $a[0]\le\cdots\le a[p-1]\le a[q]$

Q2: 如果$a[p]>a[q]$，　$a\left[0..\frac{n}{2}-1\right]$中多少元素大于$a[q]$？

A2: $a\left[\frac{n}{2}-1\right]\ge\cdots\ge a[p]>a[q]$　即 $\frac{n}{2}-p$ 个逆序

3.2 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 $A[\,\lceil n/2\rceil+1\,..\,n\,]$.
Cons.		
	$\Theta(n)$	3. ***"Merge"*** the 2 sorted lists

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

3.2.1 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.**

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

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

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

- **Solve the above recurrence through expansion.**

3.2.2 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 $T(n) \le c_1 n^3$ for $n \ge n_0$.
  - Prove $T(n) \le c_1 n^3$ by induction.

Example of substitution

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

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

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

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

思考题：
数值A,B,C，无论A和B是什么值，要使下列两个不等式同时成立，C应该满足什么条件？

(1) A < B
(2) A < B + C

A < B+C ≤ B

C ≤ 0

- *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 $T(n) \le c_1 n^3$ and $T(n) \ge c_3 n^3$ for $n \ge n_0$ simultaneously.

Example of substitution

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

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

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

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

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

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

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

- Can we say that we have proven our inductive hypothesis (I.H.) which is denoted by $T(n) \le cn^2$?
- 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 **:**

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

$\Rightarrow$ **If $0 < d < c_2$ Then $T(n) \le c_1 n^2 - c_2 n$ holds**

Example of substitution

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

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

⇨ **If $d > c_4$ Then $T(n) \geq c_3 n^2 - c_4 n$ holds**

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*** **:**

$$
\begin{aligned}
T(n) &= 2T(n/2) + dn \\
&\le 2c_1 \cdot (n/2) \cdot \log(n/2) + dn \\
&= c_1 n \cdot (\log n - 1) + dn \\
&= c_1 n \log n + (d - c_1) n
\end{aligned}
$$

➢ $T(n) \le c_1 n \log n$ *for* $n \ge n_0$ holds by assumption

➢ if $c_1 \ge d$ then $T(n) \le c_1 n \log n$ holds by induction

Thus, $T(n) = O(n \log n)$ is proven.

Apply Substitution to Merge Sort

$$
\begin{aligned}
T(n) &= 2T(n/2) + dn \\
&\geq 2c_2 \cdot (n/2) \cdot \log(n/2) + dn \\
&= c_2 n \cdot (\log n - 1) + dn \\
&= c_2 n \log n + (d - c_2) n
\end{aligned}
$$

- $T(n) \geq c_2 n \log n$ *for* $n \geq n_0$ holds by assumption
- if $c_2 \leq d$ then $T(n) \geq c_2 n \log n$ holds by induction

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

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

Exercise in Class

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

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

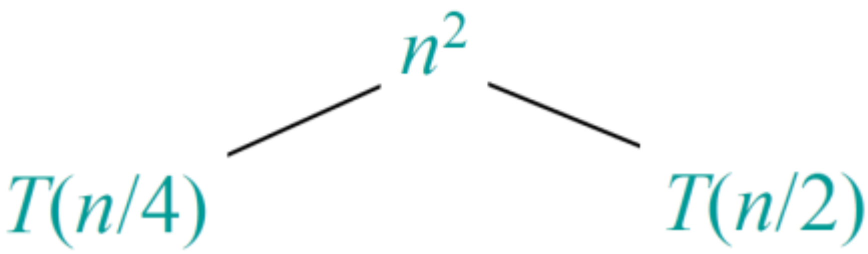
Example of Recursion-tree

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

$$T(n)$$

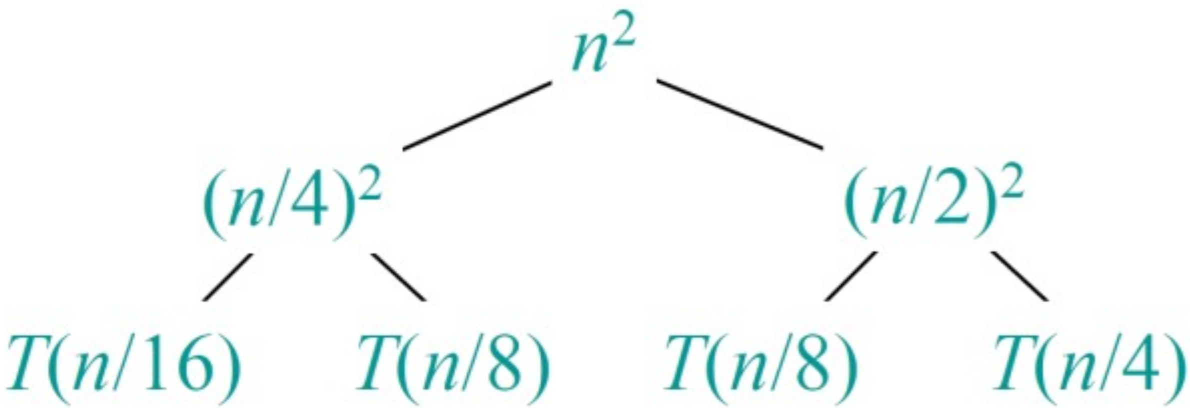
Example of Recursion-tree

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



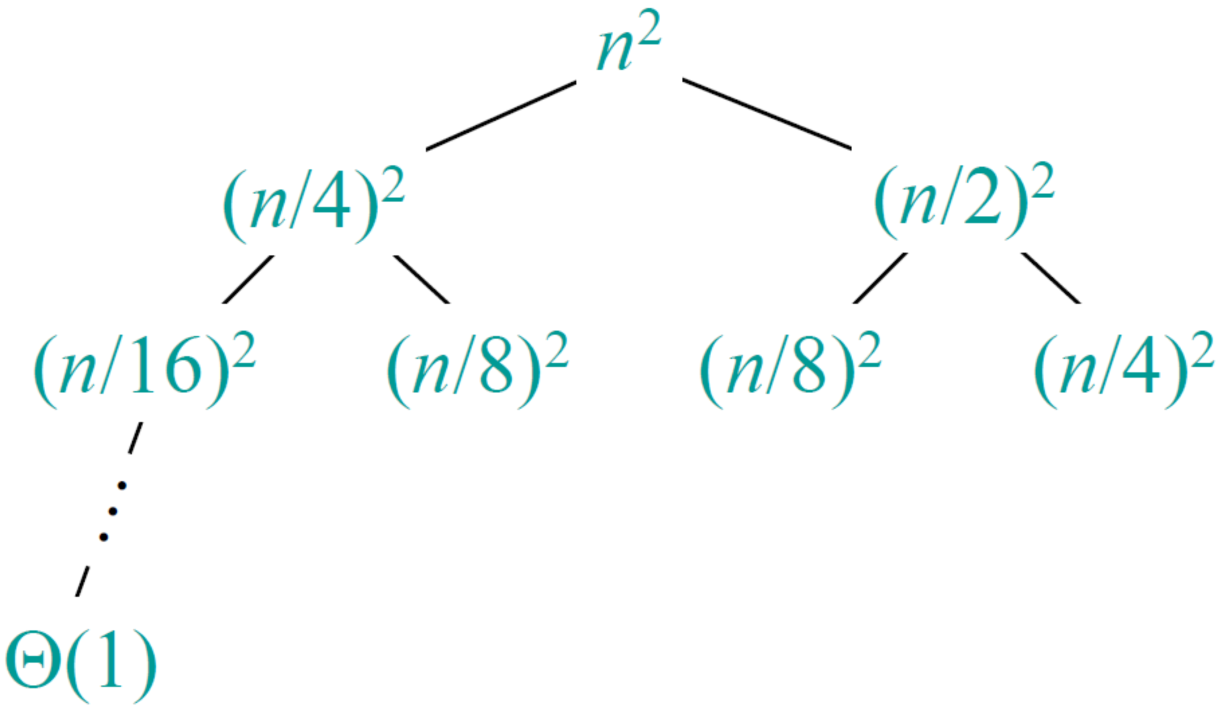
Example of Recursion-tree

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



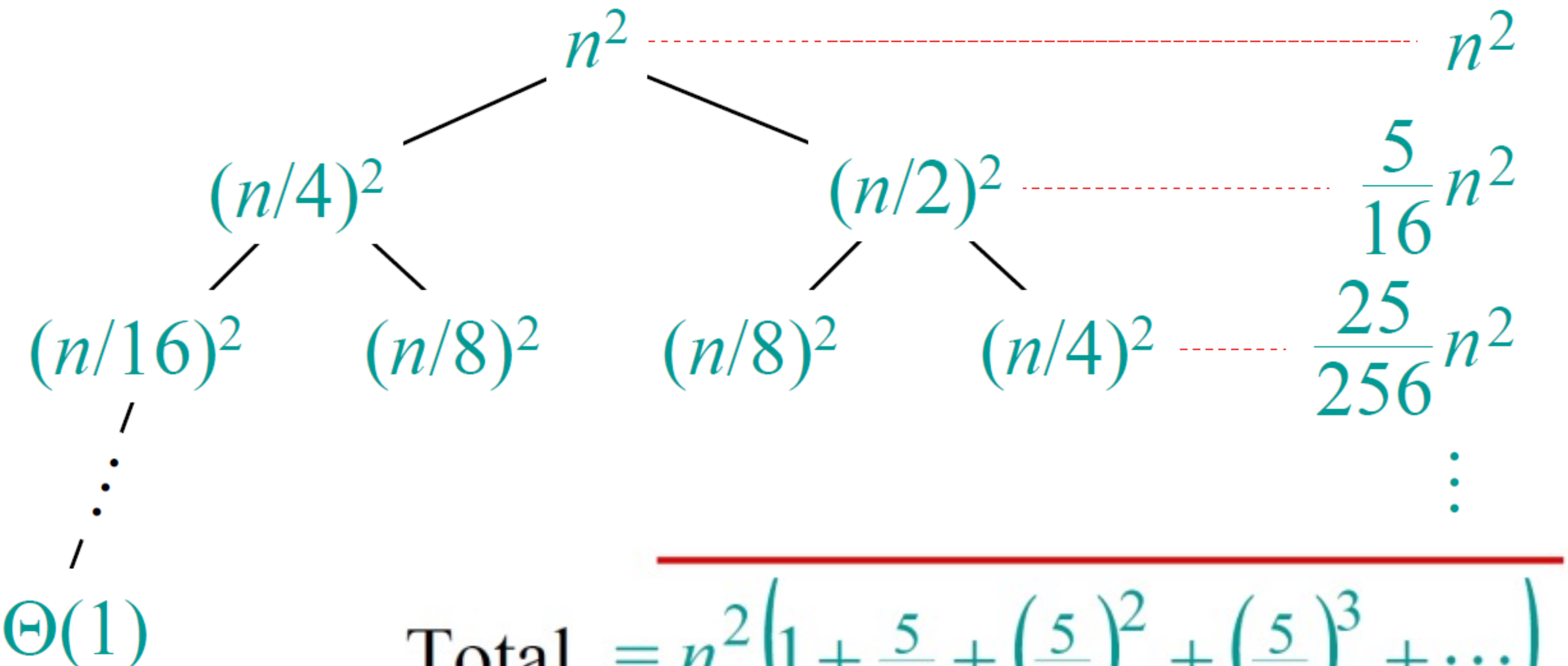
Example of Recursion-tree

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



Example of Recursion-tree

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

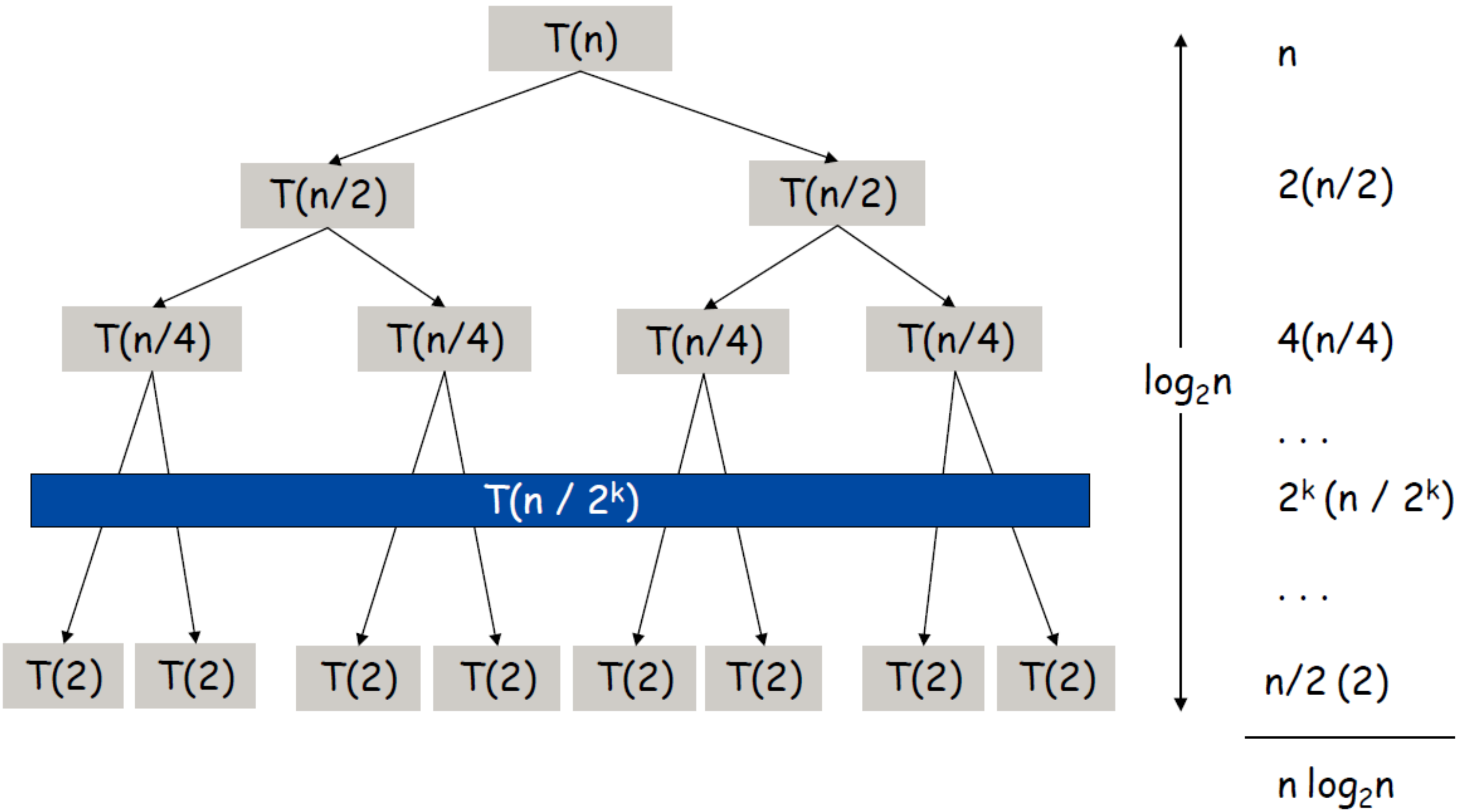


$$\text{Total} = n^2\left(1 + \frac{5}{16} + \left(\frac{5}{16}\right)^2 + \left(\frac{5}{16}\right)^3 + \cdots\right)$$

$$= \Theta(n^2)$$ *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

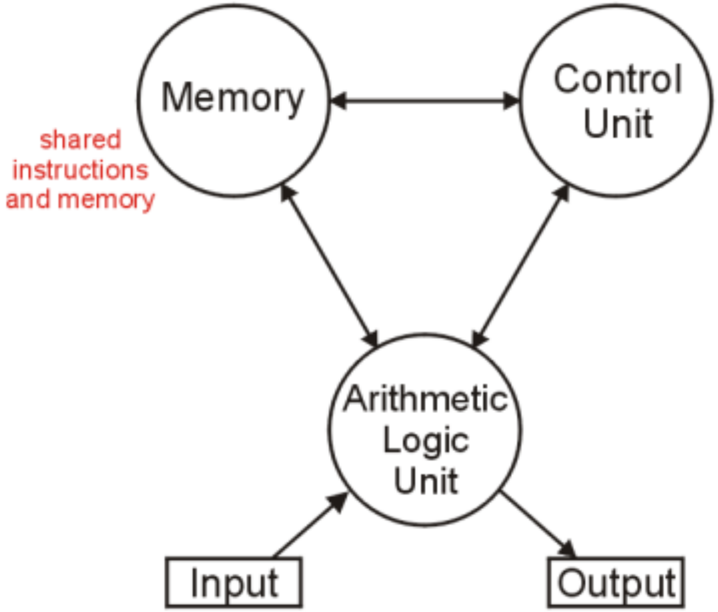
- **The following table summarizes the run-times of merge sort**

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.

《数据结构与算法》课程组

重庆大学计算机学院

End of Section.