

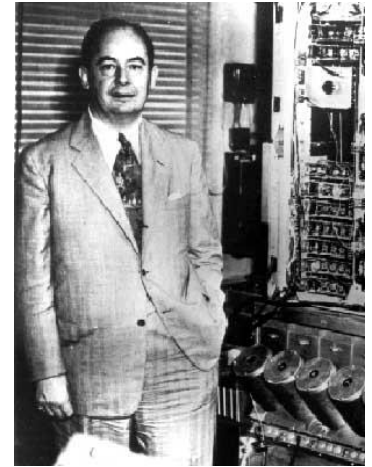
- Merge Sort
- Master Theorem
- Divide-and conquer paradigm
- Other algorithms by D&C



Merge Sort

Merge-Sort $A[1..n]$

1. If $n = 1$, done.
2. Recursively sort $A[1..n/2]$
and $A[n/2+1..n]$
3. “*Merge*” the 2 sorted lists.



Jon von Neumann (1945)



Merge Sort

MERGE-SORT(A, p, r)

```
1  if  $p < r$ 
2       $q = \lfloor (p + r) / 2 \rfloor$ 
3      MERGE-SORT( $A, p, q$ )
4      MERGE-SORT( $A, q + 1, r$ )
5      MERGE( $A, p, q, r$ )
```

Merge() takes two sorted subarrays of A and merges them into a single sorted subarray of A (how long should this take?)



MERGE(A, p, q, r)

MERGE(A, p, q, r)

```
1   $n_1 = q - p + 1$ 
2   $n_2 = r - q$ 
3  let  $L[1..n_1 + 1]$  and  $R[1..n_2 + 1]$  be new arrays
4  for  $i = 1$  to  $n_1$ 
5       $L[i] = A[p + i - 1]$ 
6  for  $j = 1$  to  $n_2$ 
7       $R[j] = A[q + j]$ 
8   $L[n_1 + 1] = \infty$ 
9   $R[n_2 + 1] = \infty$ 
10  $i = 1$ 
11  $j = 1$ 
12 for  $k = p$  to  $r$ 
13     if  $L[i] \leq R[j]$ 
14          $A[k] = L[i]$ 
15          $i = i + 1$ 
16     else  $A[k] = R[j]$ 
17          $j = j + 1$ 
```



		8	9	10	11	12	13	14	15	16	17	
A	...	2	4	5	7	1	2	3	6	...		
		k										
	1	2	3	4	5							
L	2	4	5	7	∞							
	i											
	1	2	3	4	5							
R	1	2	3	6	∞							
	j											

(a)

		8	9	10	11	12	13	14	15	16	17	
A	...	1	4	5	7	1	2	3	6	...		
		k										
	1	2	3	4	5							
L	2	4	5	7	∞							
	i											
	1	2	3	4	5							
R	1	2	3	6	∞							
	j											

(b)

	8	9	10	11	12	13	14	15	16	17
A	...	1	2	5	7	1	2	3	6	...
	k									
	1	2	3	4	5					
L	2	4	5	7	∞					
	i									
	1	2	3	4	5					
R	1	2	3	6	∞					
	j									

(c)

	8	9	10	11	12	13	14	15	16	17
A	...	1	2	2	7	1	2	3	6	...
	k									
	1	2	3	4	5					
L	2	4	5	7	∞					
	i									
	1	2	3	4	5					
R	1	2	3	6	∞					
	j									

(d)



	8	9	10	11	12	13	14	15	16	17	
A	...	1	2	2	3	1	2	3	6	...	
	k										
	1	2	3	4	5		1	2	3	4	5
L	2	4	5	7	∞		1	2	3	6	∞
	i						j				

(e)

	8	9	10	11	12	13	14	15	16	17	
A	...	1	2	2	3	4	2	3	6	...	
	k										
	1	2	3	4	5		1	2	3	4	5
L	2	4	5	7	∞		1	2	3	6	∞
	i						j				

(f)

	8	9	10	11	12	13	14	15	16	17	
A	...	1	2	2	3	4	5	3	6	...	
	k										
	1	2	3	4	5		1	2	3	4	5
L	2	4	5	7	∞		1	2	3	6	∞
	i						j				

(g)

	8	9	10	11	12	13	14	15	16	17	
A	...	1	2	2	3	4	5	6	6	...	
	k										
	1	2	3	4	5		1	2	3	4	5
L	2	4	5	7	∞		1	2	3	6	∞
	i						j				

(h)

	8	9	10	11	12	13	14	15	16	17	
A	...	1	2	2	3	4	5	6	7	...	
	k										
	1	2	3	4	5		1	2	3	4	5
L	2	4	5	7	∞		1	2	3	6	∞
	i						j				

(i)



Analysis of Merge Sort

Statement	Effort
MergeSort(A, left, right) {	$T(n)$
if (left < right) {	$\Theta(1)$
mid = floor((left + right) / 2);	$\Theta(1)$
MergeSort(A, left, mid);	$T(n/2)$
MergeSort(A, mid+1, right);	$T(n/2)$
Merge(A, left, mid, right);	$\Theta(n)$
}	
}	

So $T(n) = \begin{matrix} \Theta(1) & \text{when } n = 1, \text{ and} \\ 2T(n/2) + \Theta(n) & \text{when } n > 1 \end{matrix}$

So what (more succinctly) is $T(n)$?



Recurrences

- The expression:

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\frac{n}{2}\right) + cn & n > 1 \end{cases}$$

is a *recurrence*.

- **Recurrence**: an equation that describes a function in terms of its value on smaller functions

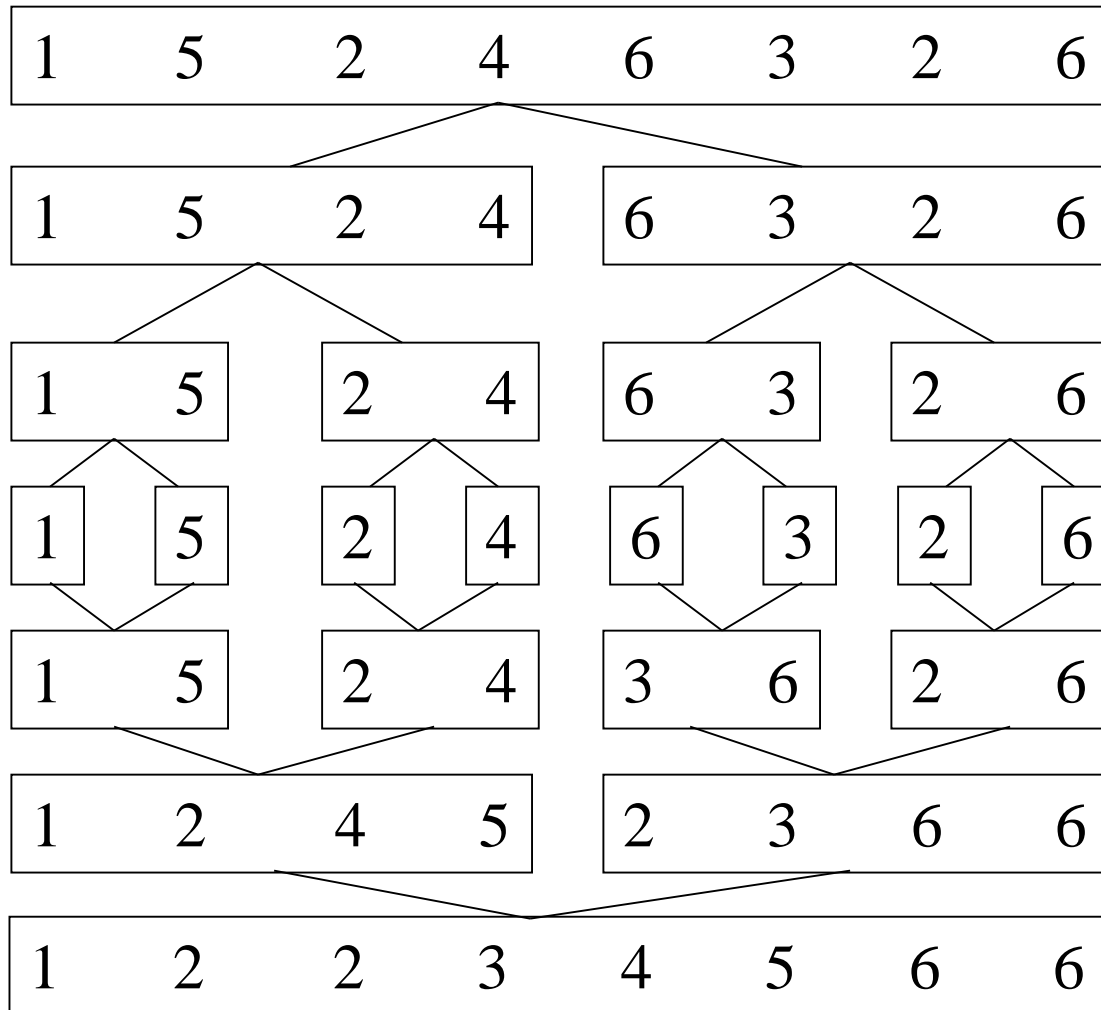


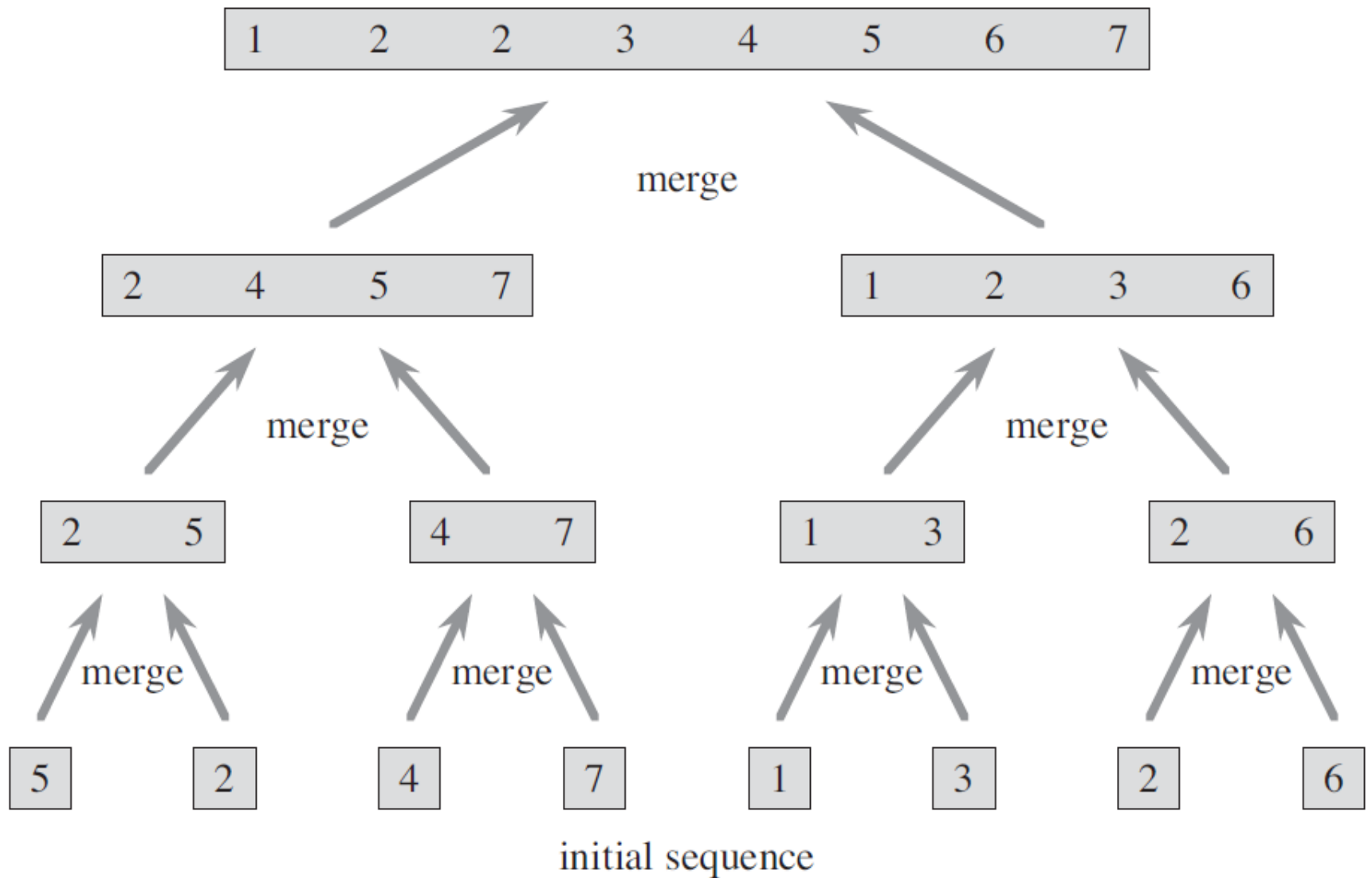
Structure of merge sort algorithm

1. **break** problem into similar (smaller) subproblems
2. **recursively solve** subproblems
3. **combine** solutions to produce final answer



Example of Merge Sort





Divide-and-conquer paradigm

1. *Divide* problem into subproblems.
2. *Conquer* subproblems by solving recursively.
3. *Combine* subproblem solutions.



Example of D&C Paradigm

Merge sort as Divide-and-conquer algorithm

1. Divide: Divide n -array into two $n/2$ -subarrays.
2. Conquer: Sort the two subarrays recursively.
3. Combine: Linear-time merge.



Recurrence for Merge sort

$$T(n) = \underset{\text{\#subproblems}}{2} \overset{\text{subproblem size}}{T(n/2)} + \underset{\text{work dividing \& combining}}{\Theta(n)}$$

$$T(n) = 2T(n/2) + \Theta(n)$$



A Useful Recurrence Relation

- Mergesort recurrence.

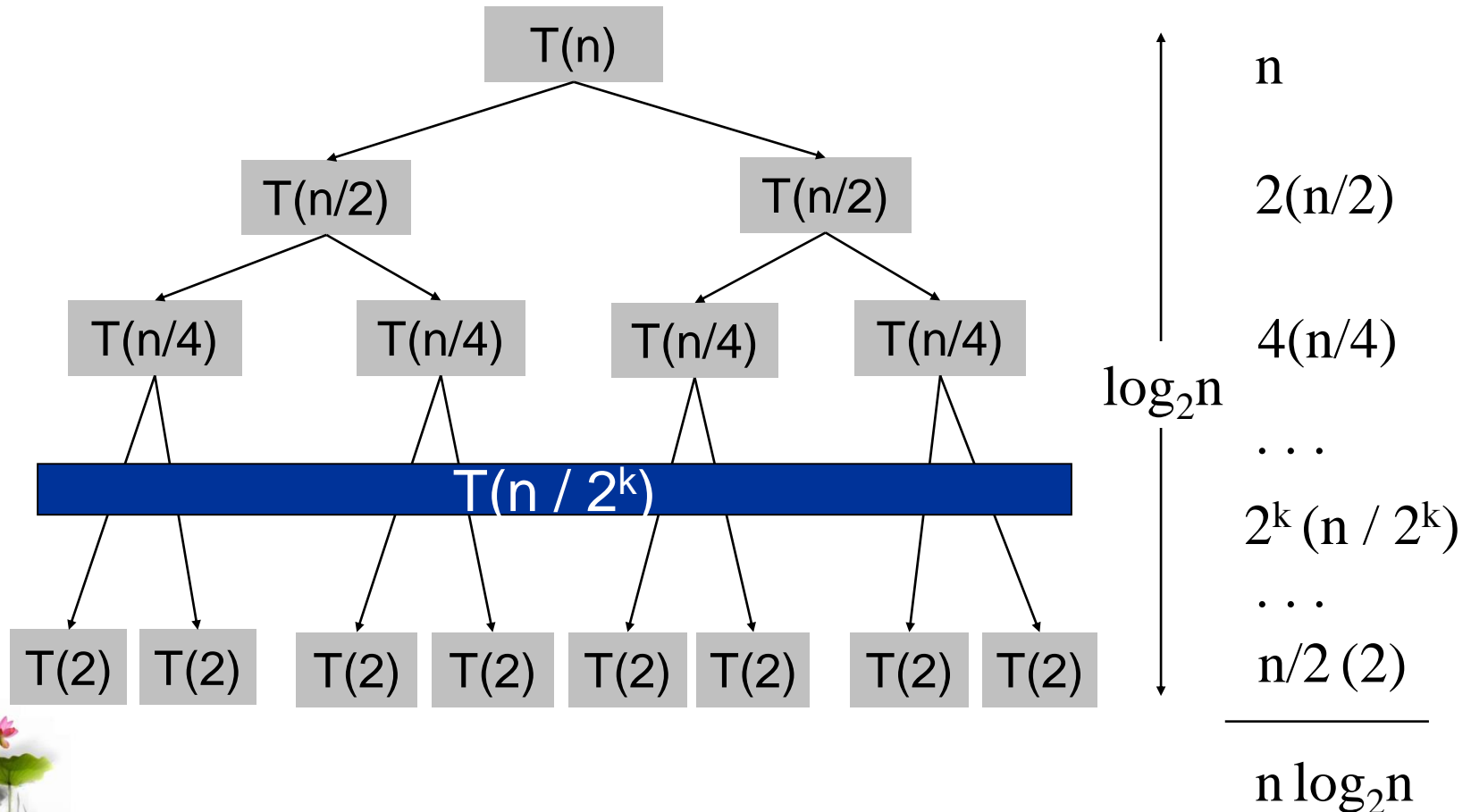
$$T(n) \leq \begin{cases} 0 & \text{if } n = 1 \\ \underbrace{T(\lceil n/2 \rceil)}_{\text{solve left half}} + \underbrace{T(\lfloor n/2 \rfloor)}_{\text{solve right half}} + \underbrace{n}_{\text{merging}} & \text{otherwise} \end{cases}$$

- 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 $=$.



Proof by 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 by Telescoping

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

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

↑
assumes n is a power of 2

- 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 $T(n) = n \log_2 n$.

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

↑
assumes n is a power of 2

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

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



Proof by Induction

- Claim. If $T(n)$ satisfies this recurrence, then $T(n) = n \lceil \lg n \rceil$.

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

↑
assumes n is a power of 2

- 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 $1, 2, \dots, n-1$.

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

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

Is there a Simple method?

