Design and Analysis Algorithms

تصميم و تحليل خوارزميات

ITGS301

Lecture 7 : المحاضرة السابعة

Tripoli University - Faculty of Information Technology
Marwa Solla
m.solla@uot.edu.ly

The divide and conquer approach

divide and conquer is perhaps the most commonly used algorithm design technique in computer science. faced with a big problem P , divide it into smaller problems, solve these sub-problems, and combine their solutions into a solution for P.

- *Divide:* the problem into a number of sub problems.
- *Conquer:* the sub problems by solving them recursively.
- *Combine:* the solutions to the sub problems into the solution for the original problem.

Merge Sort

Merge sort is based on the *divide-and-conquer* paradigm.

- ***Divide:*** divide the n elements sequence to be stored into subsequences of n/2 element each.
- ***Conquer:*** sort the two subsequences recursively using merge sort.
- ***Combine:*** merge the two sorted subsequences to produce the sorted answer.

1. *Divide Step*

If a given array A has zero or one element, simply return; it is already sorted. Otherwise, split $A[p .. r]$ into two sub arrays $A[p .. q]$ and $A[q + 1 .. r]$, each containing about half of the elements of $A[p .. r]$. That is, q is the halfway point of $A[p .. r]$.

2. *Conquer Step*

Conquer by recursively sorting the two sub arrays $A[p .. q]$ and $A[q + 1 .. r]$.

3. Combine Step

Combine the elements back in $A[p .. r]$ by merging the two sorted sub arrays $A[p .. q]$ and $A[q + 1 .. r]$ into a sorted sequence. To accomplish this step, we will define a procedure MERGE (A, p, q, r).

Algorithm:

```
MERGE-SORT(A, p, r)
1.  if p < r
2.  then q ← (p + r)/2
3.          MERGE-SORT(A, p, q)
4.          MERGE-SORT(A, q + 1, r)
5.          MERGE(A, p, q, r)
```

- Divide
- Conquer
- Conquer
- Combine

NPUT: Array A and indices p, q, r such that $p \le q \le r$ and subarray A[p ..q] is sorted and subarray A[q + 1 ..r] is sorted. By restrictions on p, q, r, neither subarray is empty.

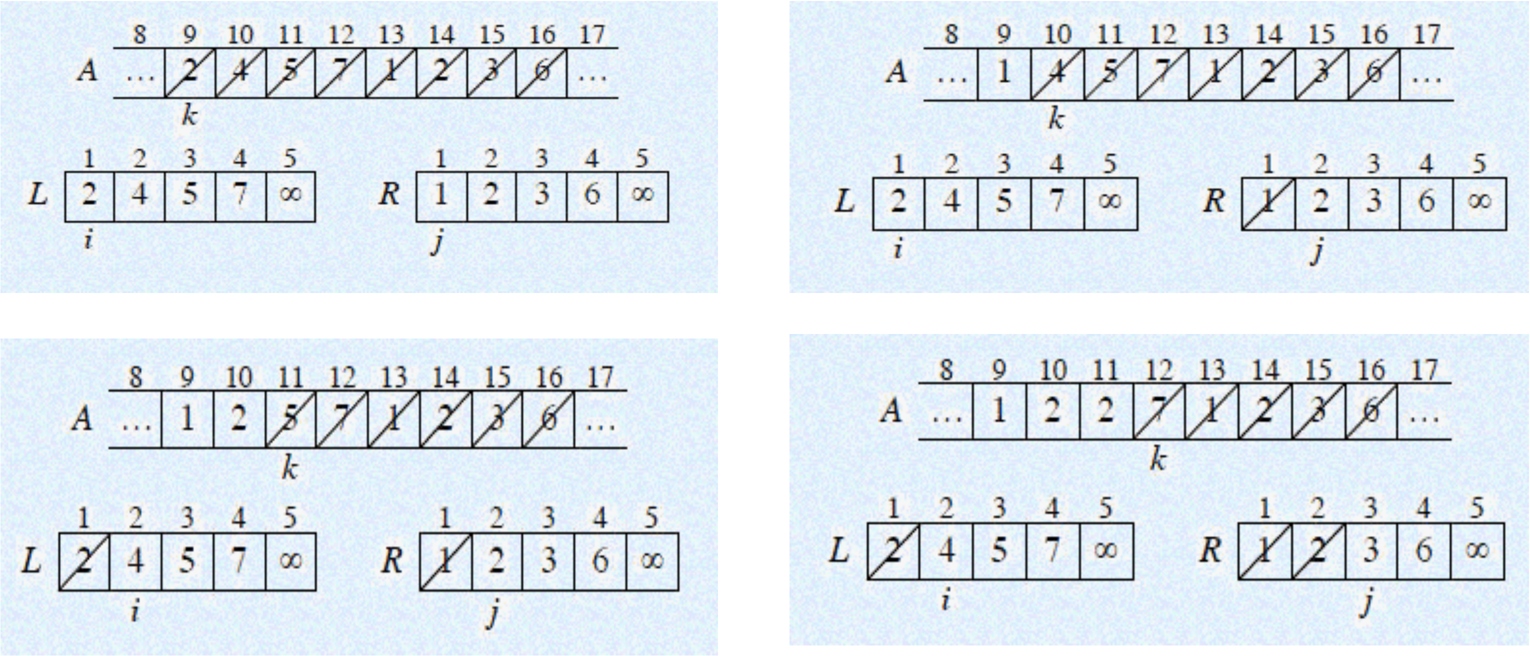
OUTPUT: The two subarrays are merged into a single sorted subarray in A[p ..r].

The Pseudocode of the MERGE procedure is as follow:

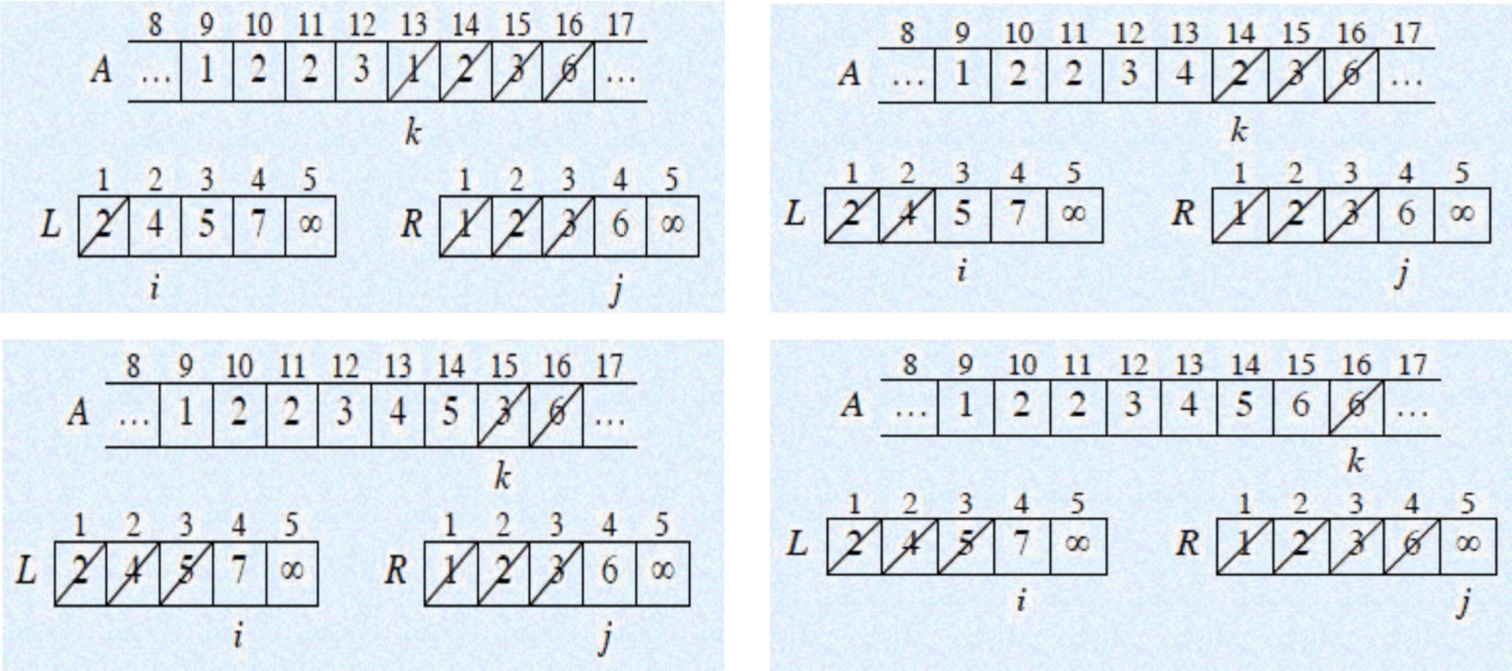
MERGE (A, p, q, r)

```
n1←q−p+1
n2←r−q
create arrays L[1..n1] and R[1..n2]
For i←1to n1
   Do L[i] ←A[p +i −1]
For j←1to n2
   Do R[j] ←A[q +j ]
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
```

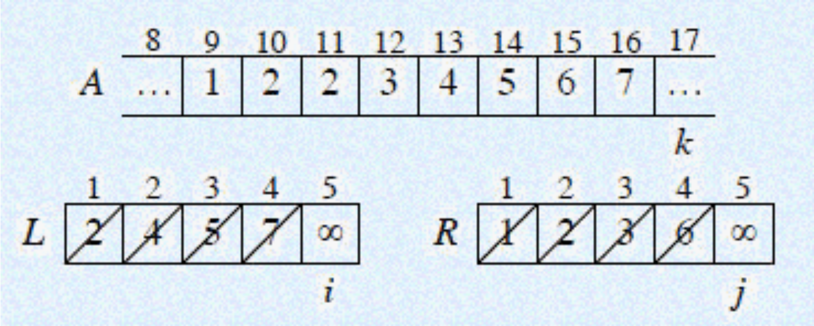
example:



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



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



Analyzing Merge Sort

For simplicity, assume that n is a power of 2 so that each divide step yields two sub problems, both of size exactly $n/2$.

The base case occurs when $n = 1$.

When $n \geq 2$, time for merge sort steps:

Divide: Just compute q as the average of p and r, which takes constant time i.e. Θ(1).

Conquer: Recursively solve 2 sub problems, each of size $n/2$, which is 2T($n/2$).

Combine: MERGE on an n-element sub array takes Θ(n) time.

Summed together they give a function that is linear in n, which is Θ(n). Therefore, the recurrence for merge sort running time is

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

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

- 2: # subproblems
- $n/2$: subproblem size
- $\Theta(n)$: work dividing and combining

Merge sort: $a = 2, b = 2 \Rightarrow n^{\log_b a} = n^{\log_2 2} = n$

CASE 2 $\Rightarrow T(n) = \Theta(n \lg n)$.

The End .