UNIT 2:
Divide and Conquer

Merge sort

Merge sort

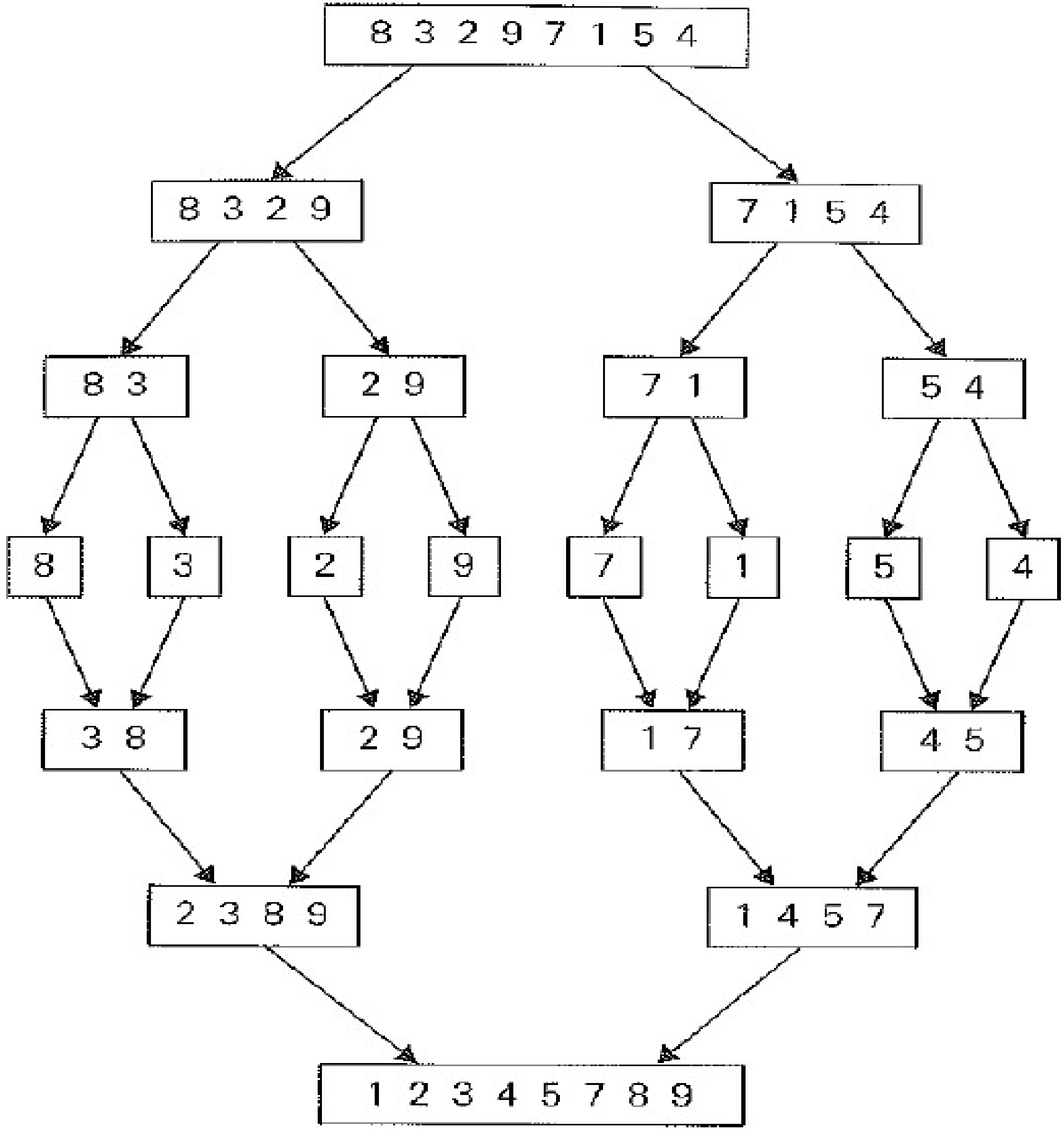
- invented by John von Neumann in 1945
- general-purpose, comparison-based sorting algorithm
- stable sorting algorithm
- uses divide and conquer strategy

Working of Merge sort

Conceptually, a merge sort works as follows:

- Divide the unsorted list into n sub-lists, each containing one element (a list of one element is considered sorted).

- Repeatedly merge sub-lists to produce new sorted sub-lists until there is only one sub-list remaining. This will be the sorted list.


8 3 2 9 7 1 5 4
8 3 2 9
7 1 5 4
8 3
2 9
7 1
5 4
8
3
2
9
7
1
5
4
3 8
2 9
1 7
4 5
2 3 8 9
1 4 5 7
1 2 3 4 5 7 8 9

Merge sort

ALGORITHM *Mergesort*($A[0..n-1]$)

//Sorts array $A[0..n-1]$ by recursive mergesort
//Input: An array $A[0..n-1]$ of orderable elements
//Output: Array $A[0..n-1]$ sorted in nondecreasing order
if $n > 1$
    copy $A[0..\lfloor n/2 \rfloor - 1]$ to $B[0..\lfloor n/2 \rfloor - 1]$
    copy $A[\lfloor n/2 \rfloor..n-1]$ to $C[0..\lceil n/2 \rceil - 1]$
    Mergesort($B[0..\lfloor n/2 \rfloor - 1]$)
    Mergesort($C[0..\lceil n/2 \rceil - 1]$)
    Merge(B, C, A)

Merge sort : Visualization

https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html

Merge sort…

ALGORITHM $Merge(B[0..p-1], C[0..q-1], A[0..p+q-1])$

```
//Merges two sorted arrays into one sorted array
//Input: Arrays B[0..p − 1] and C[0..q − 1] both sorted
//Output: Sorted array A[0..p + q − 1] of the elements of B and C
i ← 0; j ← 0; k ← 0
while i < p and j < q do
    if B[i] ≤ C[j]
        A[k] ← B[i]; i ← i + 1
    else A[k] ← C[j]; j ← j + 1
    k ← k + 1
if i = p
    copy C[j..q − 1] to A[k..p + q − 1]
else copy B[i..p − 1] to A[k..p + q − 1]
```

Merge sort algorithm analysis

1. input's size: **n –** number of elements to be sorted. (Assuming for simplicity that n is a power of 2)
2. basic operation: **comparison**
3. **No worst, average, and best cases**
4. Let T(n) = number of times the basic operation is executed.

$$T(n) = 2T(n/2) + T_{divide\_merge}(n) \text{ for } n > 1,$$

$$T(1) = 0$$

Merge sort algorithm analysis…

Assume

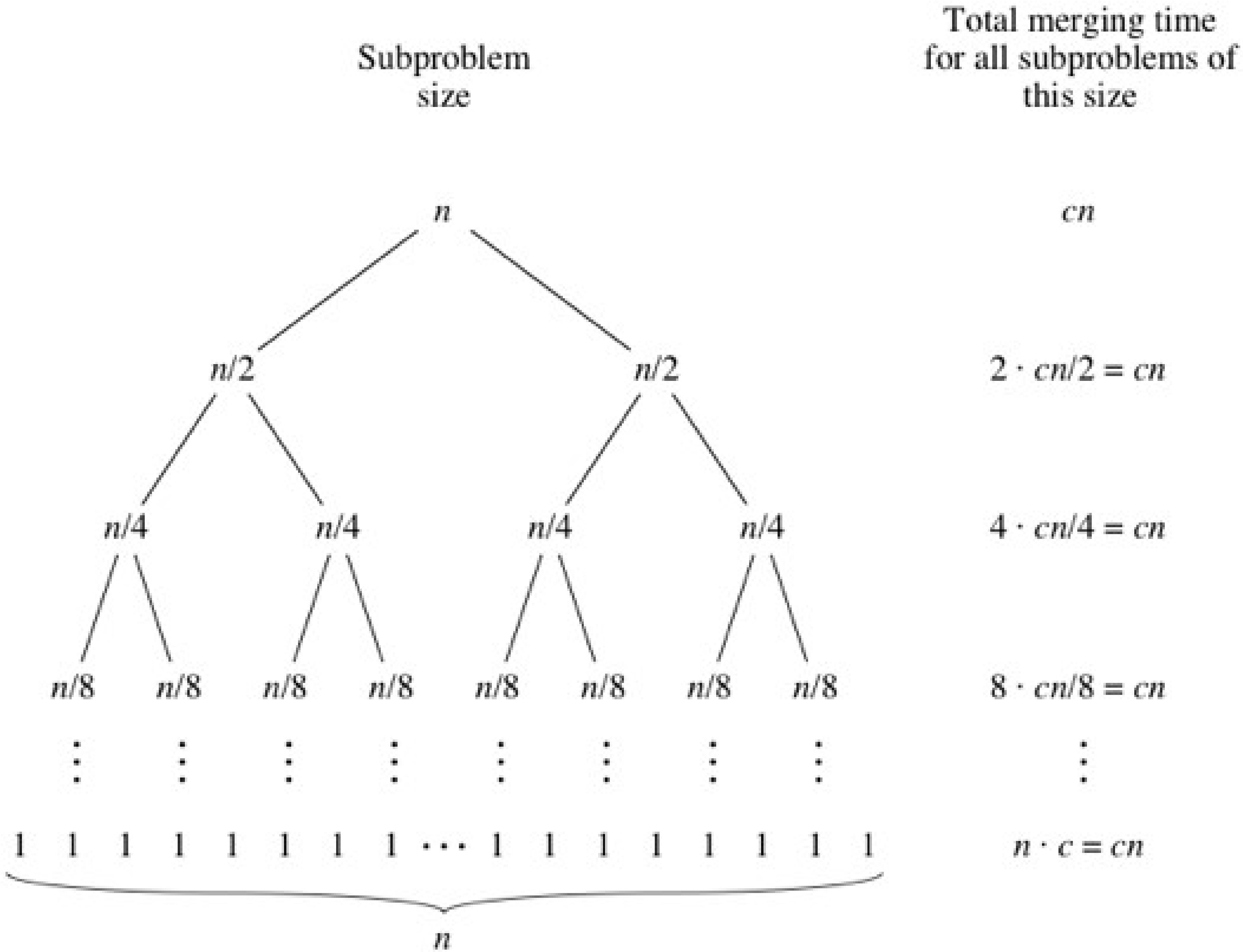
- array is of size n elements
- divide step takes constant time, regardless of the subarray size: $\Theta(1)$
- **conquer step, where we recursively sort two subarrays of approximately n/2 elements each, takes some amount of time.**
- combine step merges a total of n elements, taking $\Theta(n)$ time.

divide and combine steps together takes

$\Theta(1) + \Theta(n)$ running time

=> $\Theta(n)$

=> cn time for some constant c.



Subproblem size
Total merging time for all subproblems of this size
n
cn
n/2
n/2
2 · cn/2 = cn
n/4
n/4
n/4
n/4
4 · cn/4 = cn
n/8
n/8
n/8
n/8
n/8
n/8
n/8
n/8
8 · cn/8 = cn
1 1 1 1 1 1 1 1 · · · 1 1 1 1 1 1 1 1
n · c = cn
n

Merge sort algorithm analysis…

$T(n) = 2T(n/2) + cn$ for $n > 1$, $T(1) = 0$

$$T(n) \in \begin{cases} \Theta(n^d) & \text{if } a < b^d \\ \Theta(n^d \log n) & \text{if } a = b^d \\ \Theta(n^{\log_b a}) & \text{if } a > b^d \end{cases}$$

Using Master method:

$a = 2$, $b = 2$, $f(n) = n$, $d=1$

$$2 = 2^1$$

$$a = b^d$$

Case 3 of Master method holds good. Therefore

$$T(n) \in \Theta(n^d \log n) \quad \text{if } a = b^d$$

$T(n) = \Theta(n^1 \log n) = \Theta(n \log n)$

Merge sort : Applications

- sorting linked lists in O(n log n) time
- Inversion Count Problem
- Used in External Sorting

Merge sort : Drawbacks

- slower comparative to the other sort algorithms for smaller tasks.
- requires additional memory space of O(n) for the temporary array .
- It goes through the whole process even if the array is sorted.

Merge sort version and variants

- Top-down implementation:

  splits the list into sub-lists until sub-list size is 1, then merges those sub-lists to produce a sorted list.

- Bottom-up implementation:

  treats the list as an array of n sub-lists of size 1, and iteratively merges sub-lists

- Implementations using lists
- Natural merge sort
- Parallel merge sort
- Cascade merge sort, Oscillating merge sort, Polyphase merge sort

Let’s check our understanding

QUIZ time!!!

Attempt the quiz using the given link:

https://forms.gle/apdP6iUSruZnZ6F78

Time: 15 min

Marks: 10

No. of questions: 10

Next session…

Divide and Conquer:

Quicksort