CS & IT ENGINEERING

Algorithms

Divide & Conquer

Lecture No.- 03

By- Aditya sir

Topics to be Covered

Topic

Topic

Merge Sort

About Aditya Jain sir

1. Appeared for GATE during BTech and secured AIR 60 in GATE in very first attempt - City topper
2. Represented college as the first Google DSC Ambassador.
3. The only student from the batch to secure an internship at Amazon. (9+ CGPA)
4. Had offer from IIT Bombay and IISc Bangalore to join the Masters program
5. Joined IIT Bombay for my 2 year Masters program, specialization in Data Science
6. Published multiple research papers in well known conferences along with the team
7. Received the prestigious excellence in Research award from IIT Bombay for my Masters thesis
8. Completed my Masters with an overall GPA of 9.36/10
9. Joined Dream11 as a Data Scientist
10. Have mentored 12,000+ students & working professions in field of Data Science and Analytics
11. Have been mentoring & teaching GATE aspirants to secure a great rank in limited time
12. Have got around 27.5K followers on Linkedin where I share my insights and guide students and professionals.

Divide & Conquer

1) Min-Max
2) Binary Search.

Topic : Divide and Conquer

Merge Sort

(3) Merge sort

Divide & Conquer based sorting Technique

Merging Aglo → 60-70%

(Principle of Merging)

Apps
Greedy:
1) OMP
2) Huffman Enc

Topic : Divide and Conquer

Principle of Merging (Conquer)

Problem Statement:

Given two sorted lists/Arrays

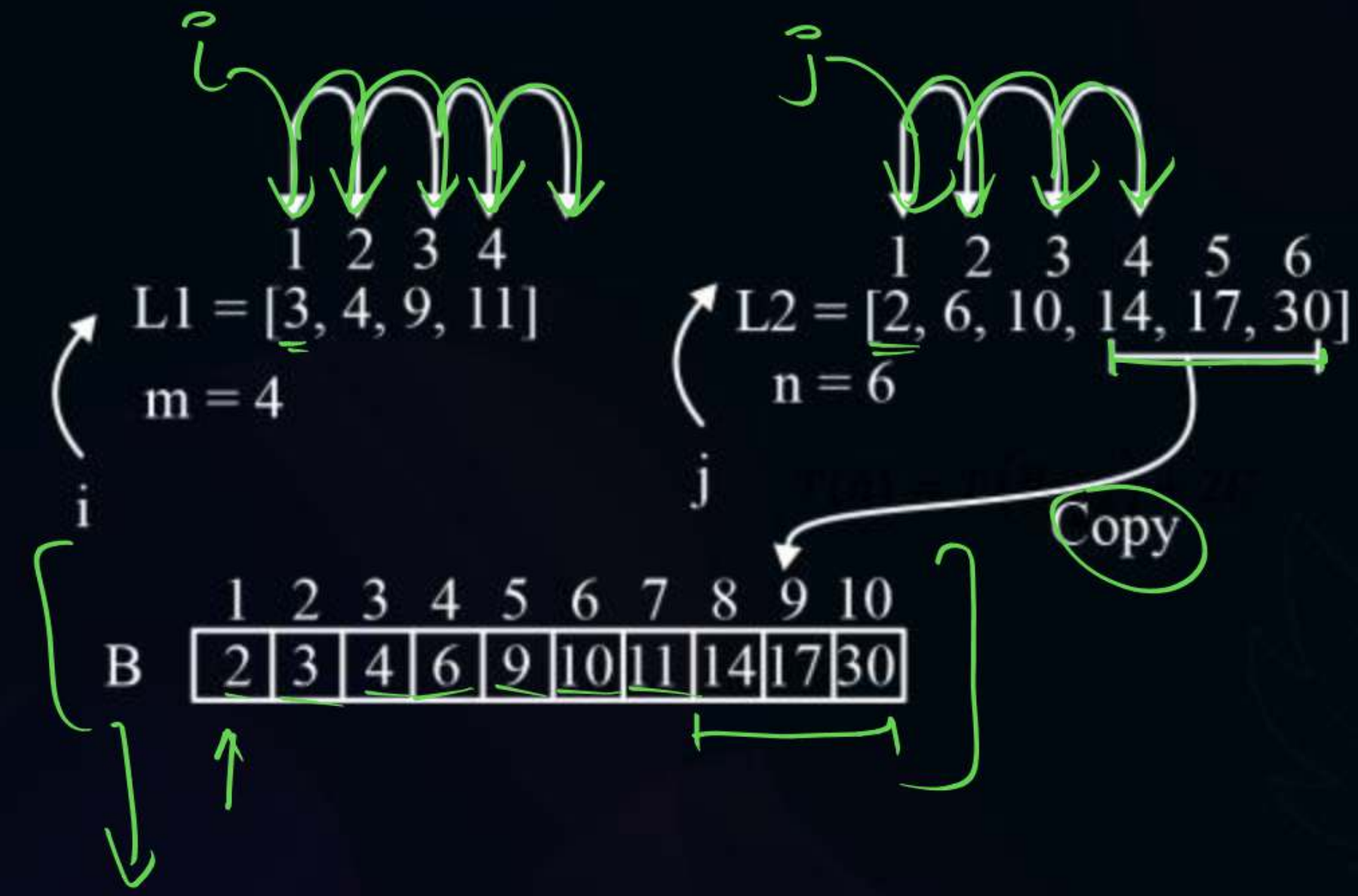
L1 → m & L2 → n such that m ≤ n,

Merge them into a single sorted list/Array → Size ⇒ (m + n)

Default → (2-way merging)

Topic : Divide and Conquer

Eg.



$m+n = 4+6$

$= 10$

Topic : Min-Max Problem

#Q. What are the min & max number of comparisons required in 2-way merging?

Sorted L1 → m, L2 → ~~m~~ n & m ≤ n

A. m, m + n

B. n, m+n–1

C. m, m + n–1

D. n, m + n

Topic : Divide and Conquer

(1) Min Comparisons:

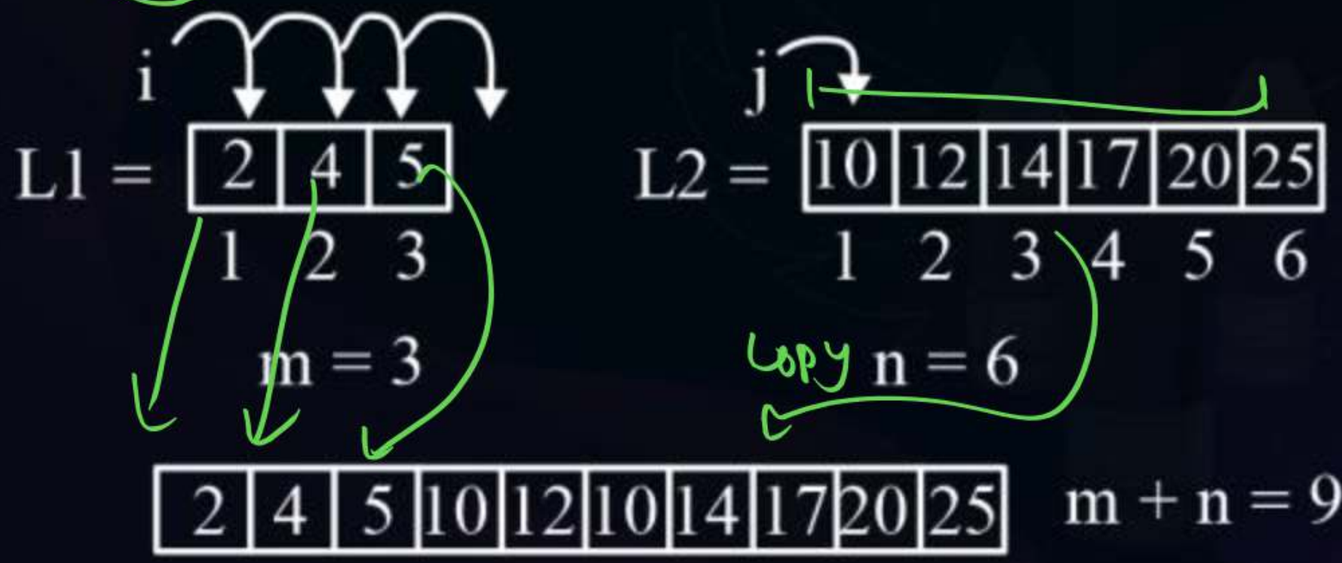
Input: All elements of the smaller list (L1) are less than all elements of the larger list (L2).

m n

OR

When last element of L1 < first element of L2

Comp = m



Topic : Divide and Conquer

(2) Max comparisons::

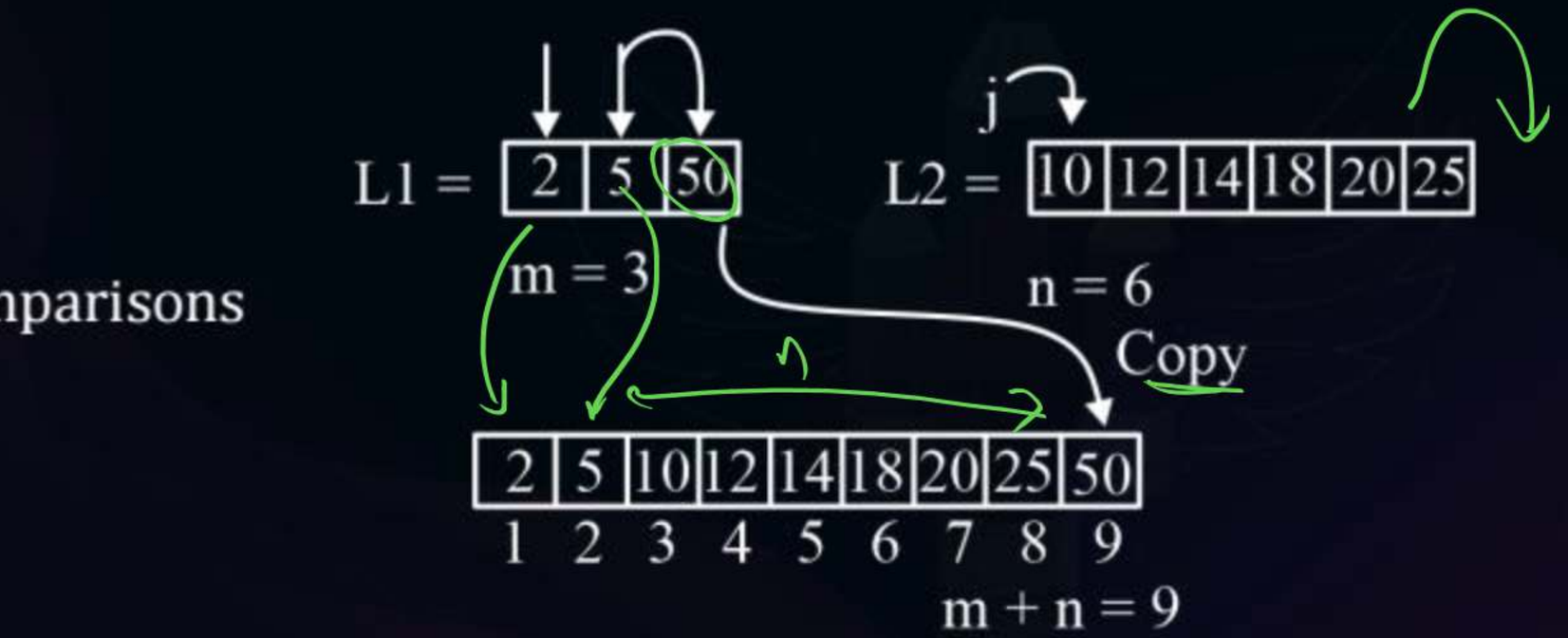
Input: Smaller List (L1): (1 → m – 1) < first element of larger list (L2)

and m^{th} element of L1 (last element) > all elements of L2 (last element of L2)

Total comparisons

= (m – 1) + n

= (m + n – 1) → max comparisons



Topic : Divide and Conquer

Summary:-

The number of elements comparisons required to merge two sorted list into a single sorted list range in between $m \leq \text{compaison} \leq (m + n - 1)$

Sorted

$L1 \rightarrow m$

$L2 \rightarrow n$

~~$L3 \leq n$~~

$s.t\ m \leq n$

$\min(m, n)$

Topic : Merge Sort

1. **Merging Algo:-**
2. Summary:

Sorted list(L_1) → m

~~Merge L_3: (n + m)~~

Sorted list(L_2) → n

[m ≤ number of element comparisons ≤ (m + n – 1)]

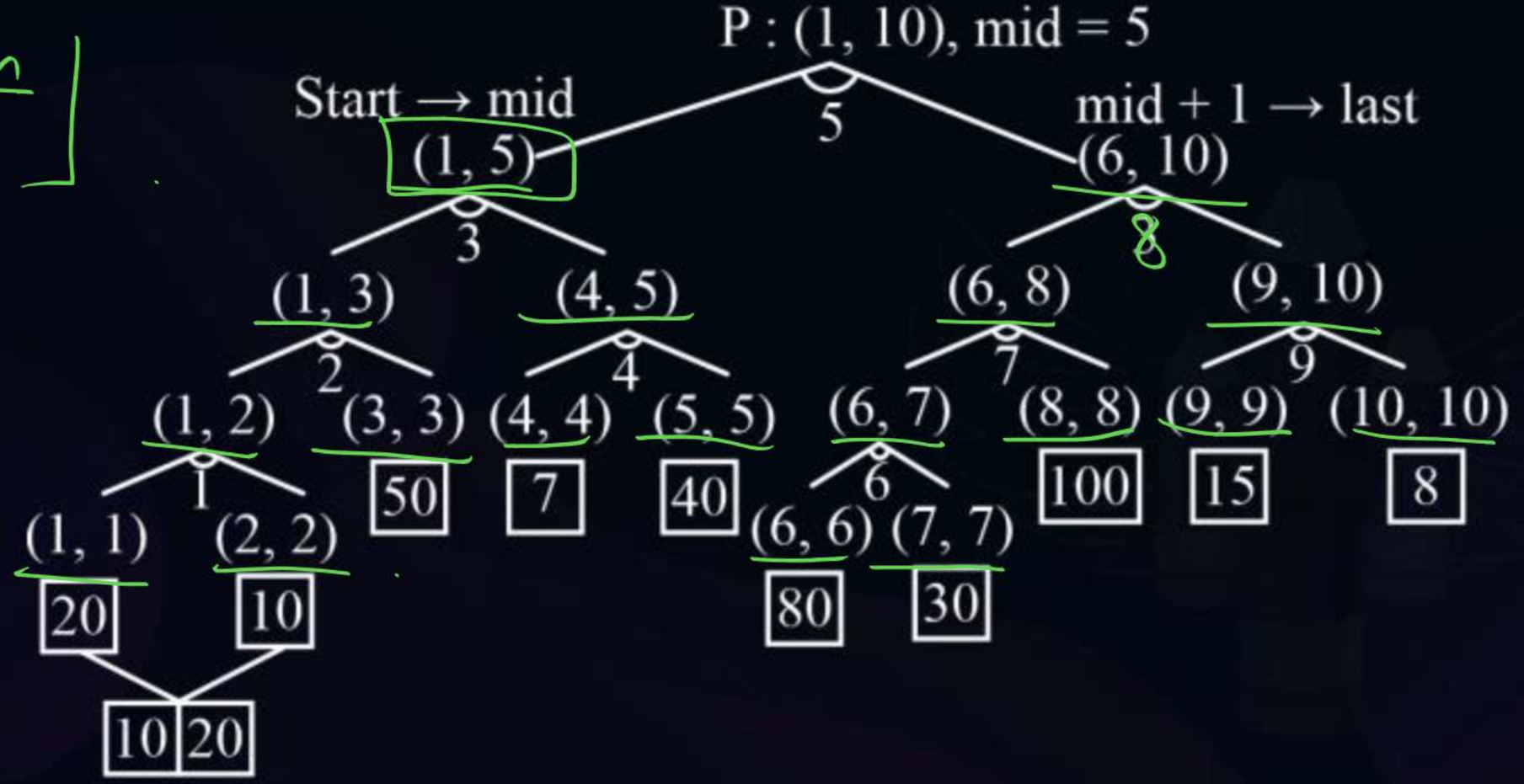
min(m, n)

Topic : Merge Sort

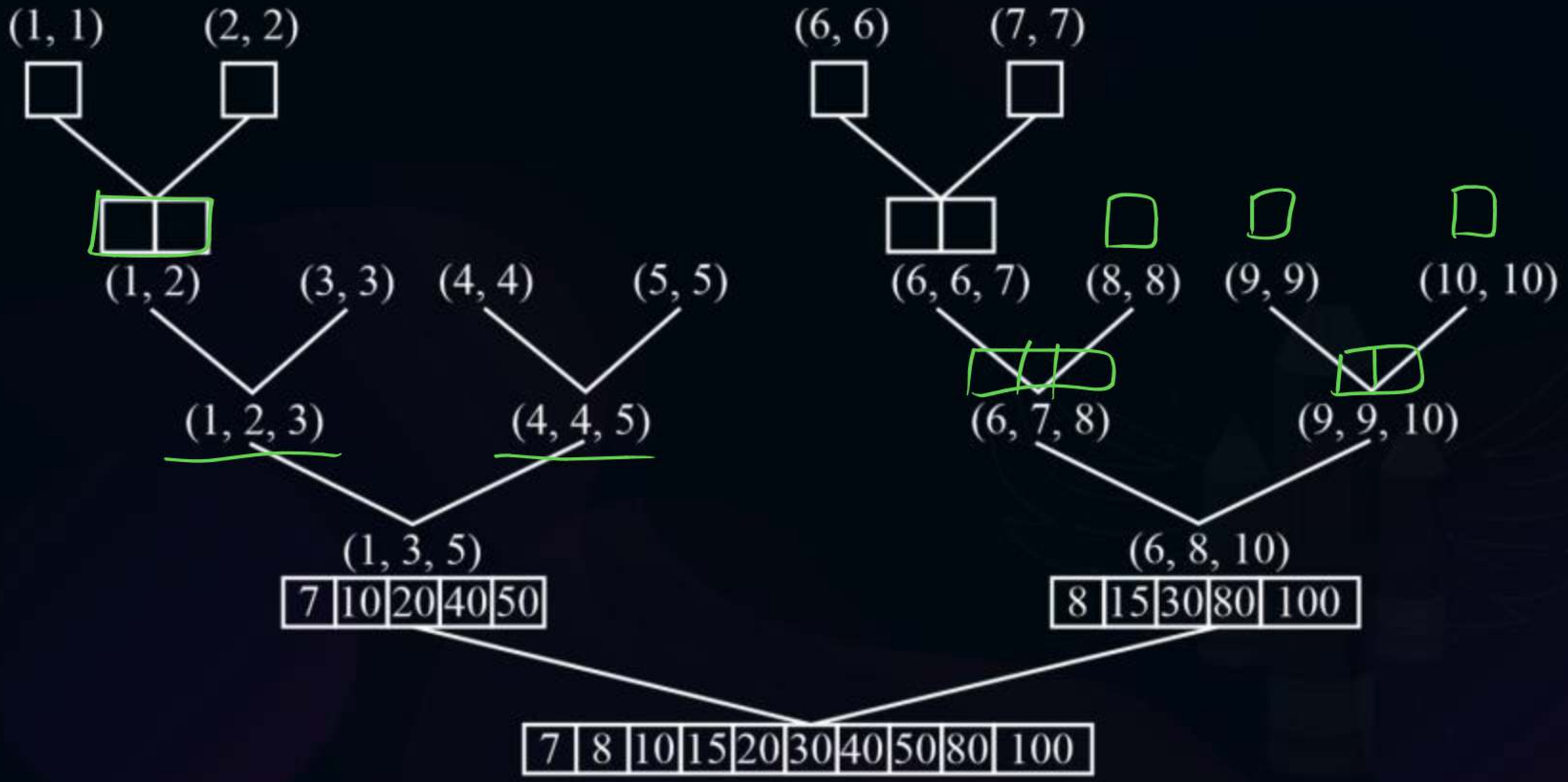
Divide and Conquer

A :	20	10	50	7	40	80	30	100	15	8
	1	2	3	4	5	6	7	8	9	10

$m = \left\lfloor \frac{l+h}{2} \right\rfloor$



Topic : Sorting Algorithms

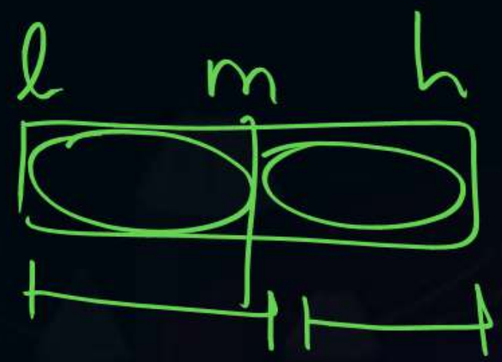


Topic : Divide and Conquer


```
1.  Algorithm MergeSort(low, high)
2.  //a[low: high] is a global array to be sorted.
3.  //Small(P) is true if there is only one element
4.  //to sort. In this case the list is already sorted.
5.  {
6.          if (low < high) then // If there are more than one element
7.          {
8.                  // Divide P into subproblems.
9.                  // Find where to split the set.
```

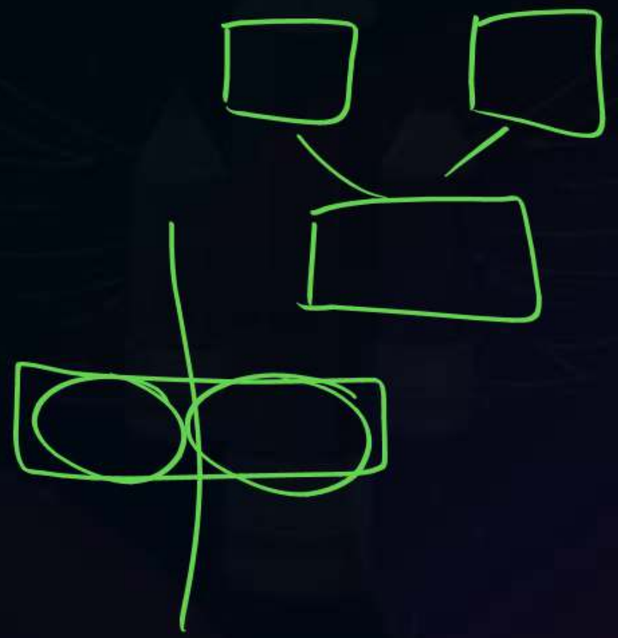
Topic : Divide and Conquer

```
10.                    mid: = [(low + high)/2];
11.                // Solve the subproblems.
12.                MergeSort(low, mid);
13.                MergeSort(mid + 1, high);
14.                // Combine the solutions.
15.                Merge(low, mid, high);
16.            }
17.  }
```



Topic : Divide and Conquer

```
1.      Algorithm Merge(low, mid, high)
2.  //a[low: high] is a global array containing two sorted
3.  // subsets in a [low: mid] and in a[mid +1: high]. The goal
4.  // is to merge these two sets into a single set residing
5.  // in a[low: high] b[ ] is an auxiliary global array.
6.  {
7.              h:= low; i := low; j := mid + 1;
8.          while ((h ≤ mid) and (j ≤ high)) do
9.              {
10.                 if (a[h]  ≤ a[j]) then
```



Topic : Divide and Conquer

```
11.             {
12.                     b[i] := a[h];h := h + 1;
13.             }
14.     else
15.             {
16.                     b[i] := a[j]; j := j + 1 ;
17.             }
18.             i := i + 1;
19.     }
20.     if (h > mid) then
```

Topic : Divide and Conquer

```
21.  for k := j to high do
22.  {
23.              b[i] := a[k]; i := i + 1;
24.  }
25.  else
26.      for k := h to mid do
27.  {
28.              b[i] := a[k]; i :=i + 1;
29.      }
```

Topic : Divide and Conquer

```
30.         for k := low to high do a[k] := b[k];
31.     }
```

Tc : O(m+n)

Sc : O(m+n)

b[ ]

Topic : Divide and Conquer

#Q. For merging two sorted lists of size m and n into a sorted list of size m + n, we require comparisons of

A $O(m)$

B $O(n)$

C $O(m + n)$

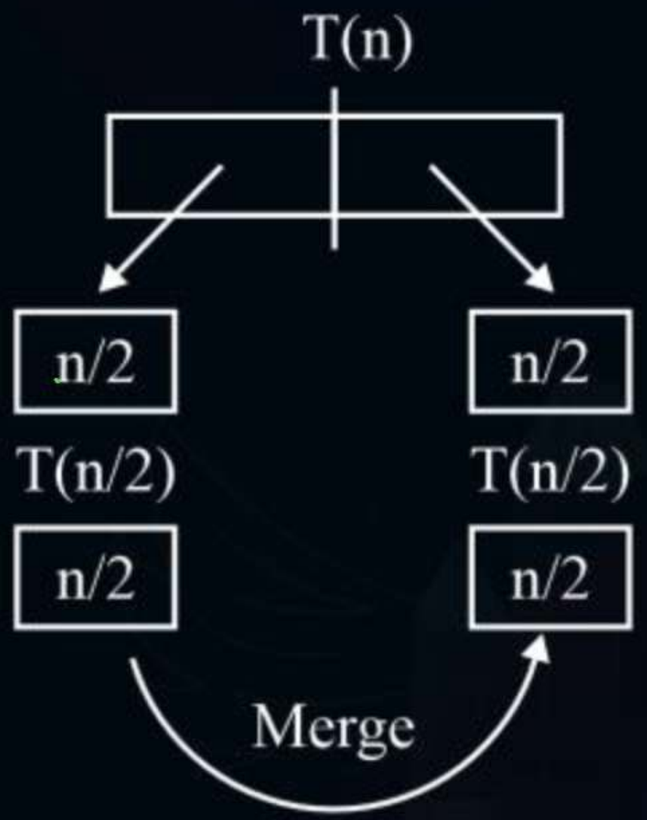
D $O(\log m + \log n)$

Topic : Performance of Merge Sort

Time complexity: Merging

Best Case: $(n/2) \rightarrow O(n)$

Worst Case: $(n/2 + n/2 - 1) \rightarrow O(n)$



Topic : Performance of Merge Sort

Let T(n) represent time Complexity of Merge Sort on A[1 : n]

$T(n) = C, n = 1$ (Small problem)

$T(n) = 2T(n/2) + b \times n, n > 1$

Solution:-

$T(n) = 2T(n/2) + b * n,\ n > 1$

$$T(n) = 2\left(2T\left(\frac{n}{2^2}\right) + \frac{bn}{2}\right) + b \times n$$

$$= 2^2T\left(\frac{n}{2^2}\right) + bn + bn$$

$$= 2^2T\left(\frac{n}{2^2}\right) + 2b \times n$$

$$= 2^3T\left(\frac{n}{2^3}\right) + 3b \times n$$

Step3:- T(n) = O(n logn) Every case

O(nlogn)

General:-

$$T(n) = 2^KT\left(\frac{n}{2^K}\right) + b \times K \times n$$

For B.C,

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

$n = 2^K \rightarrow K = \log_2{}^n$

$T(n) = n \times T(1) + b \times n \times \log_2{}^n$

$T(n) = n \times C + b \times n\log_2{}^n$

Topic : Merge Sort

TC

Best Case : $\Omega(n \log_2 n)$

Worst Case : $O(n \log_2 n)$

$\theta(n \log 2n)$

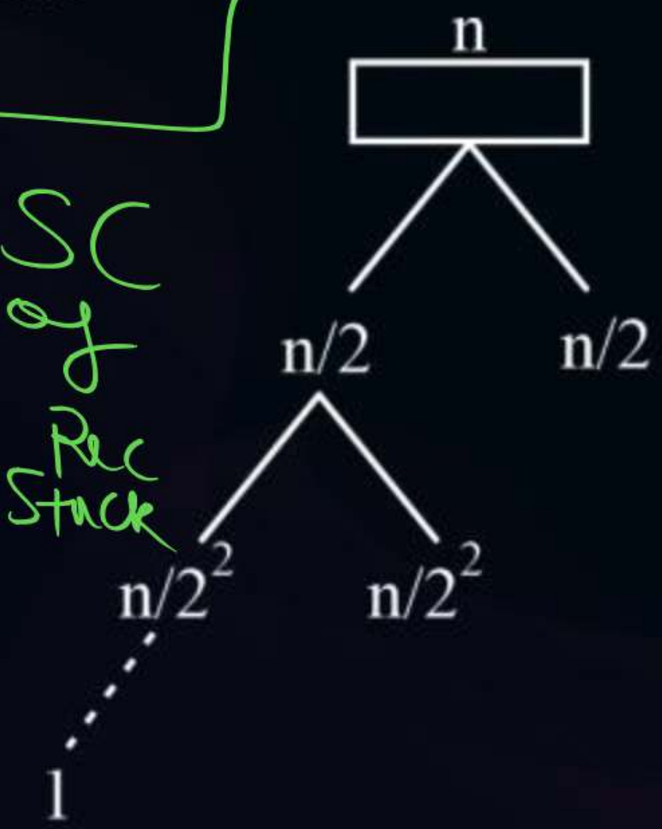
Max height of Recursion stack

~~Print~~ ~~daring~~ MS(1, n)

Will be during

$\log_2 n$

SC of Rec Stack



$n/2^K = 1$

$2^K = n$

$K = \log_2^n$

Topic : Space Complexity Analysis

(1) Temporary/Auxiliary

Array 'B' used for merging

$A = [1 \rightarrow n]$

$B = [1 \rightarrow n]$

$O(n)$

(2) Recursion Stack

$\Rightarrow \quad O(\log_2^n)$

Over all Space Complexity = $O(n + \log_2^n)$

$= O(n) \Rightarrow$ not inplace

Topic : Space Complexity Analysis

Performance Analysis of Merge Sort

→ Summary

Time Complexity ⇒ O(n log n) {Time efficient}

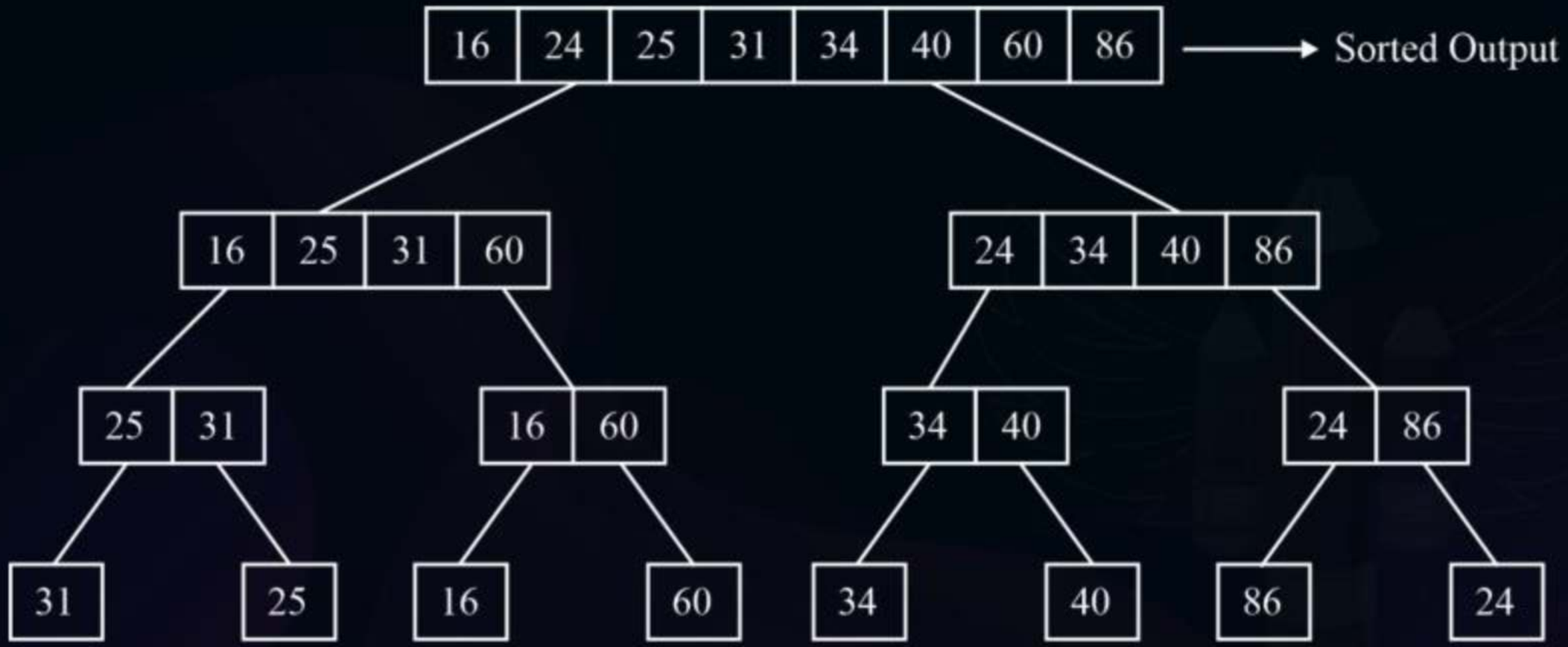
But Space Complexity ⇒ O(n) {Space inefficient}

⇒ Merge Sort is stable. (not inplace)

Topic : Space Complexity Analysis

Version-2 :- Bottom-up Merge Sort / 2-way (Special logic)

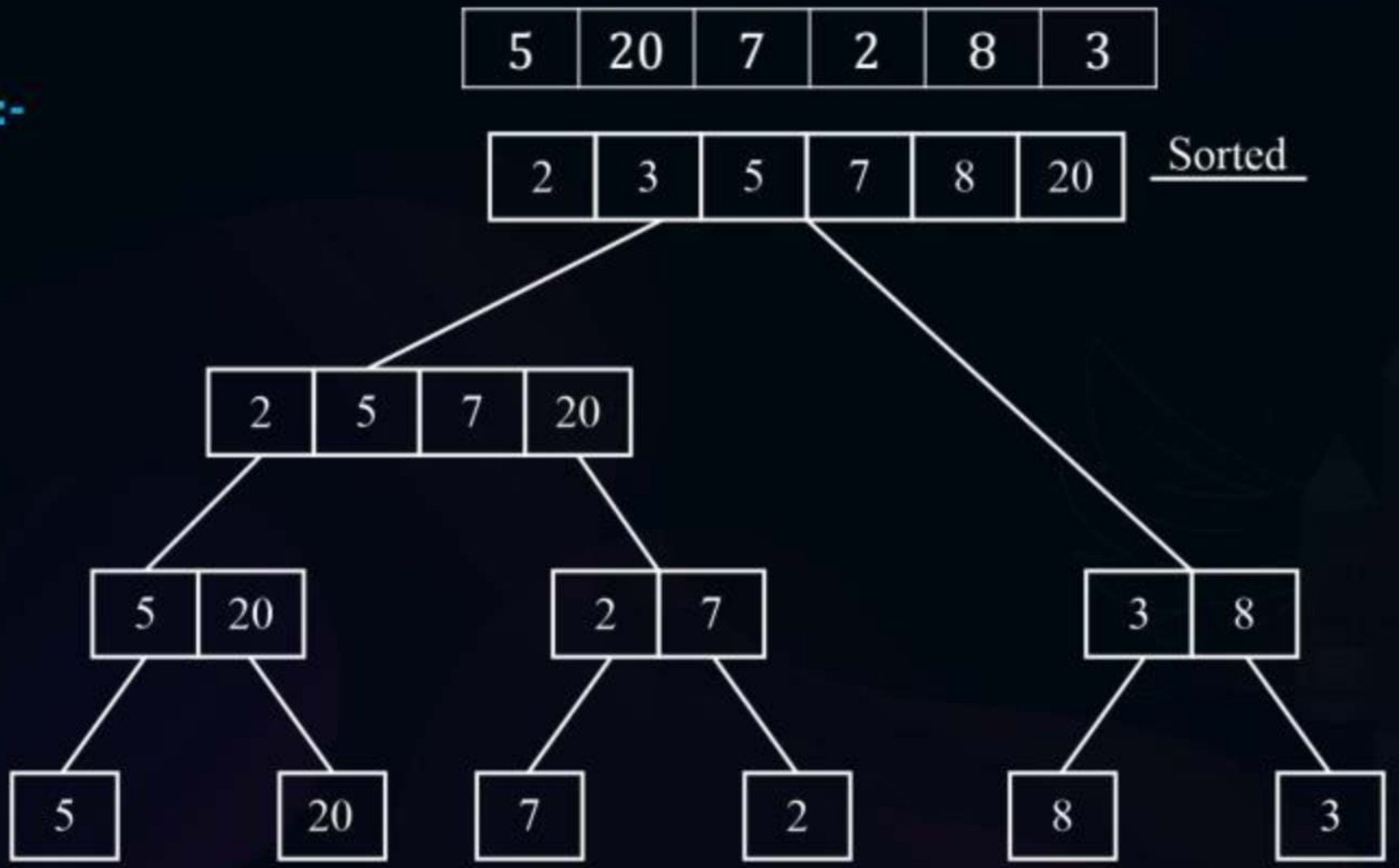
A = [31, 25, 16, 60, 34, 40, 86, 24]



Topic : Space Complexity Analysis

If equal – pairing not possible

Ex:-



THANK - YOU