LAB - 4

1. What is the principle of optimality? Explain its use in dynamic programming

→ The dynamic programming algorithm obtains the soln using principle of optimality.

→ The principle of optimality states that "in an optimal sequence of decisions or choices, each subsequence must also be optimal."

→ When it is not possible to apply the principle of optimality it is almost impossible to obtain the soln using dynamic programming approach.

→ The principle of optimality : "If k is a node on the shortest path from i to j, then the part of the path from i to k, and the part from k to j, must also be optimal."

→ It is used to solve various problem in dynamic programming.

<1> Calculating the binomial coefficient
<2> Making change problem
<3> Assembly line scheduling
<4> Knapsack problem
<5> Shortest path
<6> Matrix chain multiplication

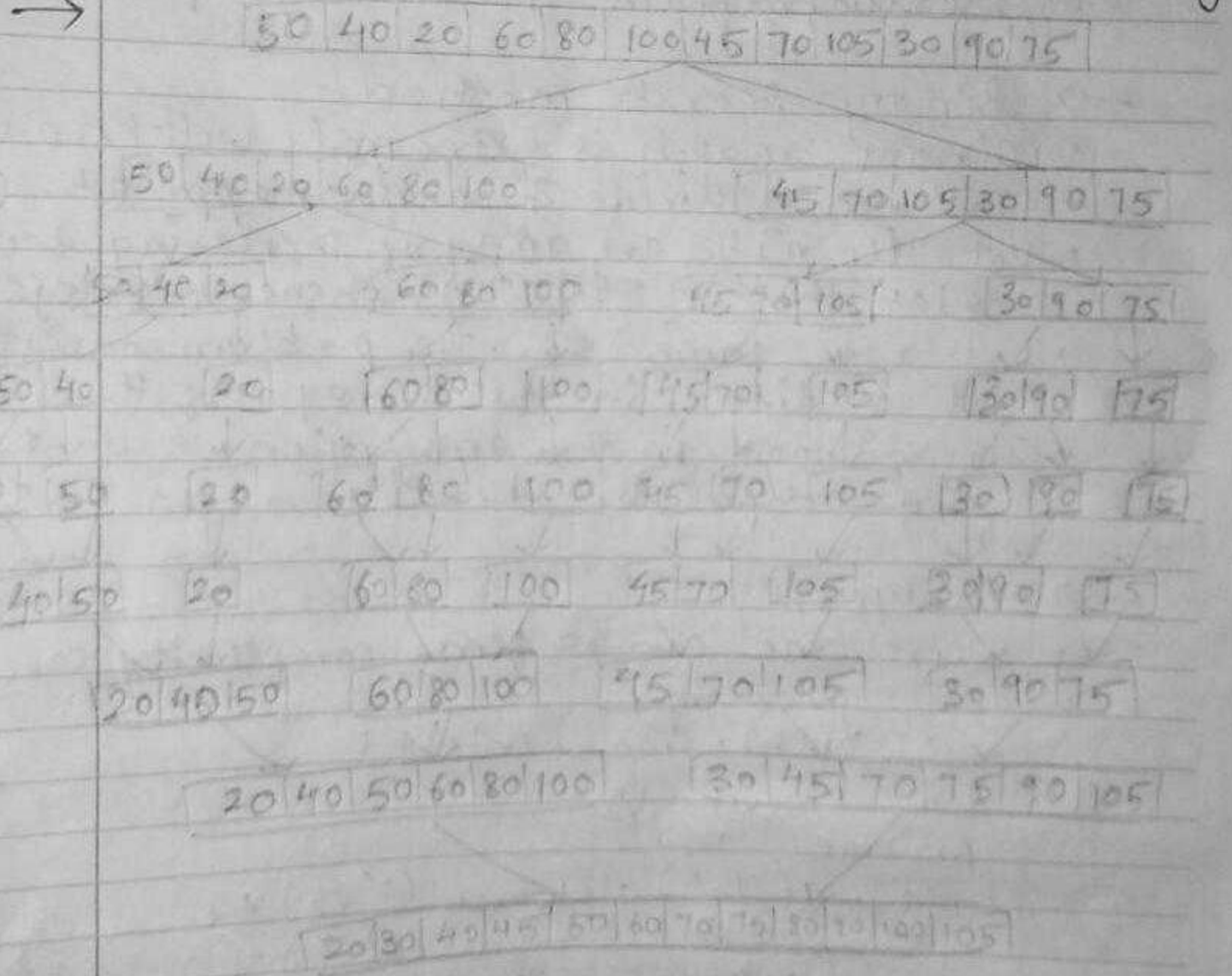
2. Differentiate betⁿ divide & conquer with dynamic programming.

→

	Divide & conquer	Dynamic programming
<1>	The problem is divided into small subproblems. These problems are solved independently. Finally all the solⁿ of subproblems are collected together to get the solⁿ to given problem.	In dynamic programming many decision sequences are generated and all the overlapping are considered.
<2>	In this method duplication are neglected.	In dynamic computing duplications in solⁿ is avoided totally
<3>	Divide & conquer is less efficient because of rework on solⁿ	Dynamic programming is efficient than divide & conquer strategy.
<4>	The divide & conquer uses top down approach of problem solving	The dynamic programming uses bottom up approach of problem solving.
<5>	Divide & conquer splits its input at specific deterministic points usually in the middle	Dynamic programming splits its input at every possible split points rather than at a particular point. After trying all split points it determines which split point is optimal

3 Sort the foll. using merge sort algorithm:
50, 40, 20, 60, 80, 100, 45, 70, 105, 30, 90, 75
Also discuss worst & best case of merge sort algo

→



→ Here, $T(n) = T(n/2) + T(n/2) + g(n)$
$= 2T(n/2) + \Theta(n)$
$a=2, b=2, k=1$ (∵ Applying general case)
∴ $T(n) = \Theta(n \log n)$ for both worst & best case

Date Page No.

Explain the use of divide & conquer technique for Binary search method. What is complexity of Binary search method? Explain it with example.

→ Binary search method :-

- Binary search is extremely well-known instance of divide & conquer approach.
- Let T[1..n] be an array of increasing sorted order that is T[i] ≤ T[j] whenever 1 ≤ i ≤ j ≤ n.
- Let x be some no. the problem consists of finding x in the array T if it is there.
- If x is not in the array, then we want to find the position where it might be inserted

→ Analysis :-

- Worst case can be time complexity can be given by;

$C_{worst}(n) = C_{worst}(n/2) + 1$

Also $C_{worst}(1) = 1$

Assume $n = 2^k$

$C_{worst}(2^k) = C_{worst}(2^k/2) + 1$ — (1)

$C_{worst}(2^k) = C_{worst}(2^{k-1}) + 1$ — (2)

Using backward substitution;

$C_{worst}(2^{k-1}) = C_{worst}(2^{k-2}) + 1$ — (3)

Put eqn (3) in (2)

$C_{worst}(2^k) = C_{worst}(2^{k-2}) + 2$

$C_{worst}(2^k) = C_{worst}(1) + k$

$\therefore C_{worst}(n) = 1 + \log n$

$\therefore C_{worst}(n) \simeq \log n$

- Therefore the time complexity for worst case & average case is $O(\log n)$
- For best case it is $O(1)$.

→ Example:- Search the element 60 from the list 10, 20, 30, 40, 50, 60

- Consider a list of array

10	20	30	40	50	60	70
low						high

- Key element to be searched is 60.
- Middle element (m) = (low+high)/2 = (0+6)/2 = 3

40 = 60 ? NO ⇒ 40 < 60

- So it will check the right sublist.

... | 50 | 60 | 70 |

- Again m = (4+6)/2 = 5
- Hence the element 5 is the required search of our list.

60 = 60 ? Yes

- i.e the no. present in the list.