Fall - 2022

Ans. to the Q. No-01 (a)

$$T(n) = 3T\left(\frac{n}{4}\right) + O(n \log n)$$

$a = 3$

$b = 4$

$K = 1$

$P = 1$

$a < b^K$, $P \geq 0$,

$$T(n) = \theta(n^K \log^P n)$$

$$= \theta(n \log n)$$

$$= O(n \log n)$$

Ans. to the Q.No - 1 (b)

$$T(n) = 3T(n/3) + O(1)$$

$a = 3$

$b = 3$

$K = 0$

$P = 0$

$a > b^K$

$$\therefore \quad T(n) = \theta(n^{\log_b a})$$

$$= O(n)$$

Ans. to the Q.No- 02 (a)

	rice	Salt	Saffron Powder	Sugar
Cost:	800	800	2000	500
weight:	10	10	8	4
Cost/weight:	80	89	250	125

$\frac{10}{9}$

Knapsack = 8 kg

Thief 1:

Saffron

8 kg

knapsack

→ Profit → 2000 taka [Max Possible]

Thief 2:

Salt

4 kg

→ 356 tk

Sugar

4 kg

→ 500 tk

knapsack

profit : 500 + 356 = 856 taka

Thier 3:

Knapsack	
Rice 2kg	→ 160 tk
Salt 6kg	→ 534 tk

Knapsack

∴ Max probit = 534 + 160
= 694 taka

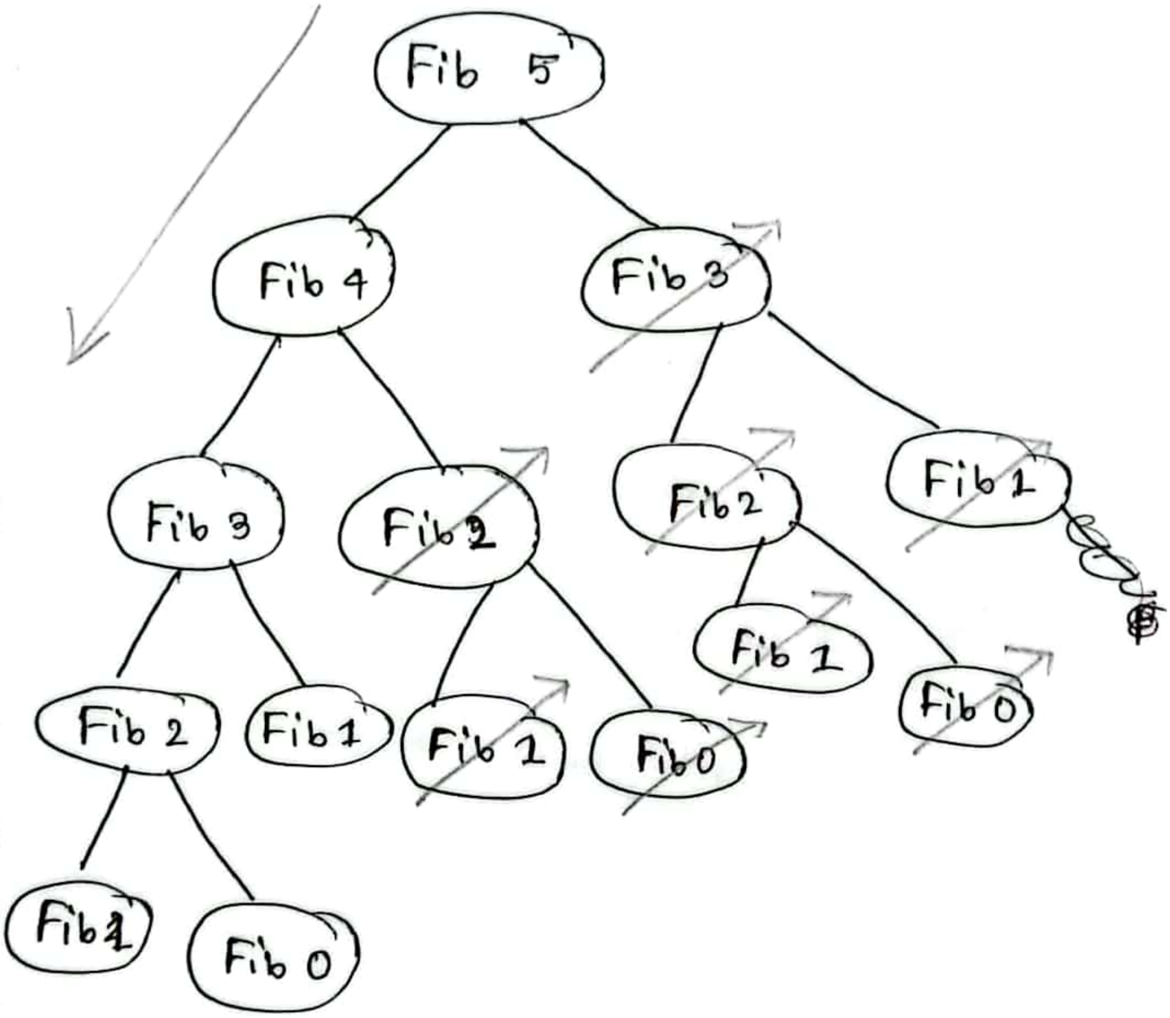
Thief 4:

Knapsack	
Rice 8kg	→ 640 tk [Max]

Knapsack

Ans. to the Q. No- 03 (a)

(i)



Here we can see that Fib 4 contains Fib 3, Fib 2, Fib 1, Fib 0 and Fib 3 also contains Fib 2, Fib 1, Fib 0 and Fib 5 contains Fib 3.

So this is the overlapping subproblem property.

(16)

(ii)

Dynamic programming improves the running time than an obvious recursive algorithm, by avoiding redundant calculations ~~tro~~ through the use of memorization or ~~at~~ tabulation. It does this by storing and reusing solutions to overlapping subproblems, reducing the time complexity from exponential to polynomial.

Example:

Time complexity of fibonacci series using

brute force, $T(n) = O(2^n)$

using dp, $T(n) = O(n)$.

Ans. to the Q.NO-03 (b)

The optimal Substructure property means that the optimal solution to a problem can be Constructed from optimal solutions to it's subproblems. Dynamic Programming efficiently addresses this property by storing and reusing solutions to overlapping subproblems. In contrast, divide and conquer typically solved subproblems independently without explicitly considering overlapping subproblems, potentially leading to redundant calculations.

Ans. to the Q. NO - 03 (c)

Cost $= [\{50 + (50 \times 3\%)\} \times 2]$ taka

$= 103$ taka

I gave $= 110$ taka

$\therefore$ I will get $= 110 - 103 = 7$ taka

value	1	2	3	4	5	6	7
min	1	1	2	2	1	2	2
	1		1+1	2+1		1+1	2+1
	2		1+1	1+1		2+1	1+1
	5					1+1	1+1

For 7, The cashier will give me minimum 2 coins from the combination.

Ans. to the Q. NO-04 (a)

Worst Case :

$$T(n) = 1 + n+1 + \frac{2n^3+3n^2+n}{6} + \frac{2n^3+3n^2+n}{6} - 1 + m + m - 1 + m - 1 + 1$$

$$= O(n^3 + n^2 + n + m)$$

$$= O(n^3) \quad [n \geq 1,\ m = 0]$$

$$= O(m) \quad [m \geq 1,\ n = 0]$$

Best case : Constant

Ans. to the Q. NO-04 (b)

Cost calculation :

$$T(n) = 1 + \log_2 n + \{(\log_2 n - 1)(\tfrac{n}{2} + 1)\} + \{\log_5 n (\log_2 n - 1)(\tfrac{n}{2})\} + \{\log_5 n (\log_2 n - 1)(\tfrac{n}{2}) - 1\} + 1$$

$$= O(n \log_2 n \log_5 n)$$

[Proved]