Dynamic Programming

DP is a technique to solve problems efficiently that have following properties :-

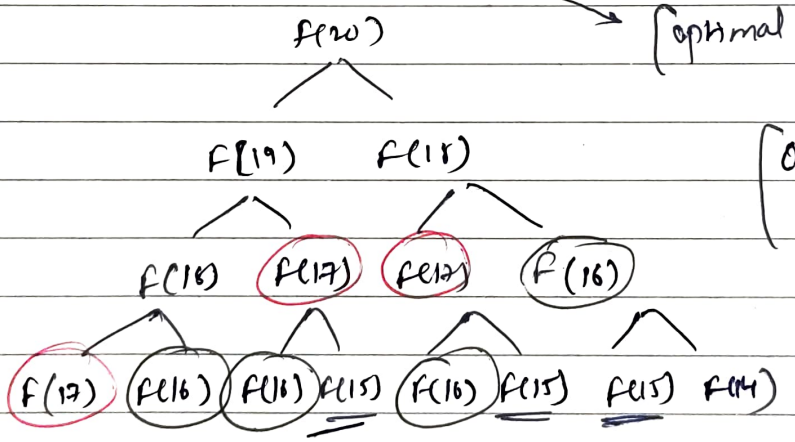
1) optimal substructure property
2) overlapping subproblems.

A problem is said to have optimal substructure if an optimal solution can be constructed from optimal solutions of its subproblems.

A problem is said to have overlapping subproblems if problem can be broken down into subproblems which are reused several times.

Ex :- $f(n) = f(n-1) + f(n-2)$ ⟷ fibbonacci

[optimal substructure].



[Overlapping subproblems]

Approach -

1) Break complex problem into smaller problems
2) Find optimal solution to these sub problems
3) Store the result of subproblem (memoization).
4) Reuse them when sub problem is calculated again
5) Finally, find result for complex problem.

0/1 Knapsack Problem

Given array of items with their weight and value. and a knapsack with weight W. Find max. value of items that can be included into knapsack.
[We either have to pick full (1) or no item (0)].

Way of thinking

for each item - We have 2 choices
1) Take it
2) Not take it

Let f(n, w) - denotes max. value of items for 'n' items and 'w' weight knapsack.

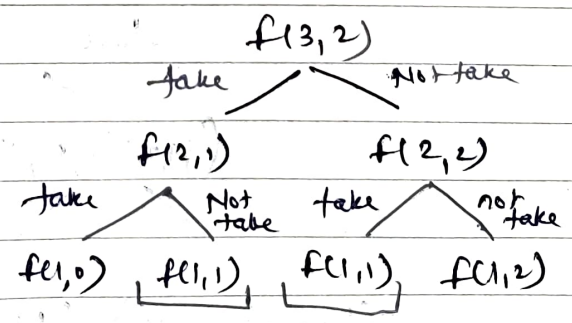
f(n, w) = f(n-1, w) ⟹ Do not take it
f(n, w) = v_i + f(n-1, w-w[i]) ⟹ Take it

It follows → optimal substructure property

for overlapping subproblems -

Ex.

1	1	1	W=2
5	10	20	



overlapping subproblem.

Refer separate PDF uploaded for detailed explanation of 0/1 knapsack

* Coin Change

Given a set of coins and a value 'V', find no. of ways we can make change of V.

Ex:- $S = \{1,2,3\}$ $V = 3$.

Possible ways - $\{3\}$ $\{2,1\}$ $\{1,1,1\}$.

Logic

we have two options -

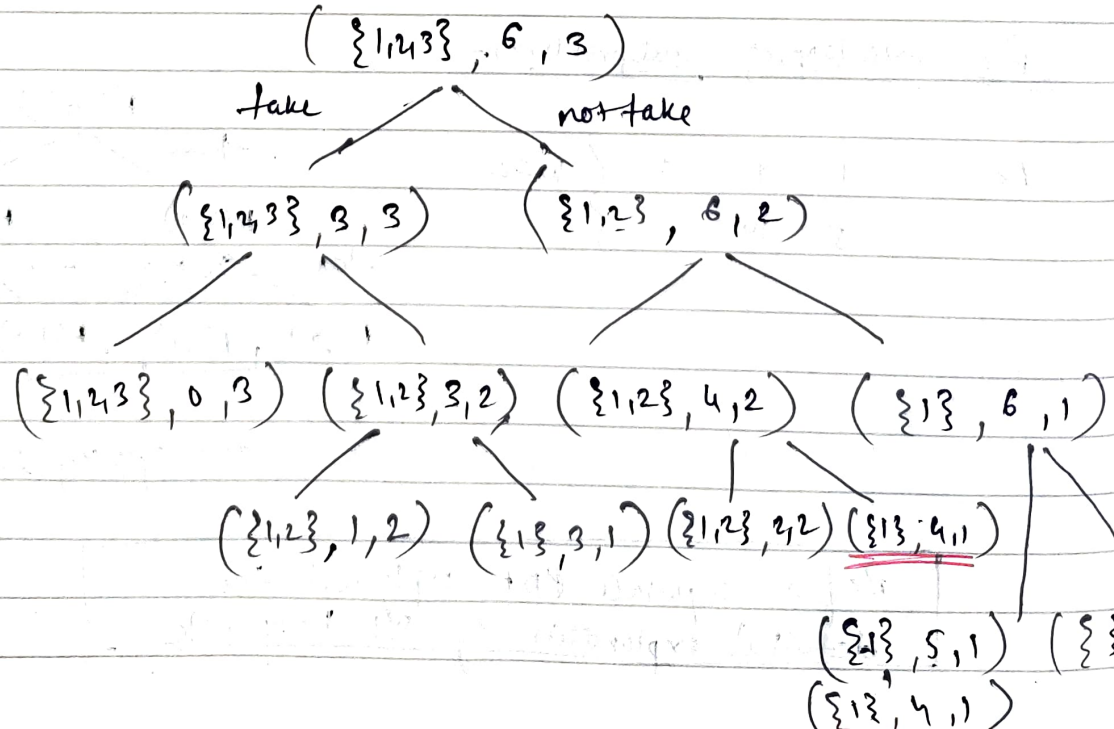
1) Make it
2) Not take it

$$count(s, v, n) = \underbrace{count(s, v-s_m, n)}_{\text{Take it}} + \underbrace{count(s, v, n-1)}_{\text{Not take it}}$$

Thus, it has optimal substructure property.

Overlapping subproblem

$\{1,2,3\}$ $v = 6$. — Make recursion tree



Examples

coin \ value	0	1	2	3	4	5	6
0	1	0	0	0	0	0	0
1	1	1	1	1	1	1	1
2	1	1	2	2	3	3	4
3	1	1	2	3	4	5	(7)

for each cell → 2 options →

Do Not Take coin = $v[i-1, j]$

Take coin = $v[i, j-c_i]$ } Add

Algo :-

```
for i=1 to n
    for j=1 to sum
        if j < ci
            v[i,j] = v[i-1, j]
        else
            v[i,j] = v[i-1, j] + v[i, j-ci]
```

Complexity = $O(n \cdot sum)$

Space = $O(n^2)$ → Can be improved to $O(n)$.

* Minimum No. of coins Required to make sum.

$C = \{1, 7, 10\}$

i	0	1	2	3	4	5	6	7	8	9	10
s[i]	0	1	2	3	4	5	6	1	2	3	1

Logic - If we select i^{th} item - $1 + s[n - c_i]$

ie.

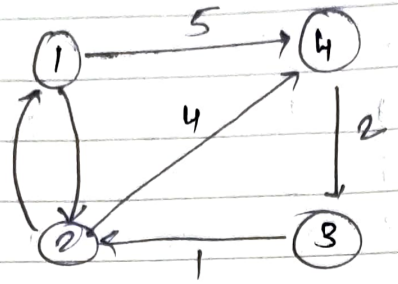
for 1p ⇒ $1 + s[n-1]$

7p ⇒ $1 + s[n-7]$

10p ⇒ $1 + s[n-10]$.

minimum coins = $1 + \min\{s[n-1], s[n-7], s[n-10]\}$.

Flyod Warshall



$$D_0 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 5 \\ 2 & 0 & \infty & 4 \\ \infty & 1 & 0 & \infty \\ \infty & \infty & 2 & 0 \end{bmatrix} \end{array}$$

for 2-3 $\Rightarrow$ Direct 2-3 $= \infty$

via (1) $2\text{-}1 + 1\text{-}3 = 2 + \infty = \infty$ } min

$= \infty$

Similarly others :-

$$D_1 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 5 \\ 2 & 0 & \infty & 4 \\ \infty & 1 & 0 & 8 \\ \infty & \infty & 2 & 0 \end{bmatrix} \end{array} \qquad D_2 = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & 3 & 4 \end{array} \\ \begin{bmatrix} 0 & 3 & \infty & 5 \\ 2 & 0 & \infty & 4 \\ 3 & 1 & 0 & 5 \\ \infty & \infty & 2 & 0 \end{bmatrix} \end{array}$$

$$D_3 = \begin{bmatrix} 0 & 3 & \infty & 5 \\ 2 & 0 & \infty & 4 \\ 3 & 1 & 0 & 5 \\ 5 & 3 & 2 & 0 \end{bmatrix} \qquad D_4 = \begin{bmatrix} 0 & 3 & 7 & 5 \\ 2 & 0 & 6 & 4 \\ 3 & 1 & 0 & 5 \\ 5 & 3 & 2 & 0 \end{bmatrix}$$

```
for k = 1 to n
    for i = 1 to n
        for j = 1 to n
            D_k(i,j) = min { D_{k-1}(i,j),
                             D_{k-1}(i,k) + D_{k-1}(k,j) }
return D.
```

Complexity $= O(n^3)$.

Matrix Chain Multiplication

Given :- • 'n' matrices

Goal :- Multiply them such that total no. of operations are minimum.

Ex :- A = 1×2 B = 2×3 C = 3×4. {1, 2, 3, 4}

A*(BC) ⇒ A = 1×2 BC = 2×4

2×3×4 operation. = 24

A(BC) = 1×4.

1×2×4 = 8 operations.

Total = 24+8 = 32 operations.

AB = 1×3 C = 3×4.

1×2×3 = 6 operations.

(AB) C = 1×4

1×3×4 = 12 operations.

Total = 6+12 = 18 operations.

Similarly :- (A) (BCD).
(AB) (CD).
(ABC) D.

} minimum of these 3.

We can write recurrence -

$f(ABCD) = \min [f(A|BCD), f(AB|CD), f(ABC|D)]$

Thus, it follows substructure property.

Here, dimensions are given in the form of array.

Dimension of i^{th} matrix is - $a[i-1] \times a[i]$.

Ex :- $M_1 = a[0] \times a[1]$
$M_2 = a[1] \times a[2]$
$M_3 = a[2] \times a[3]$.

Dimension of matrix for multiplication of M_i to M_j
$= a[i-1] \times a[j]$

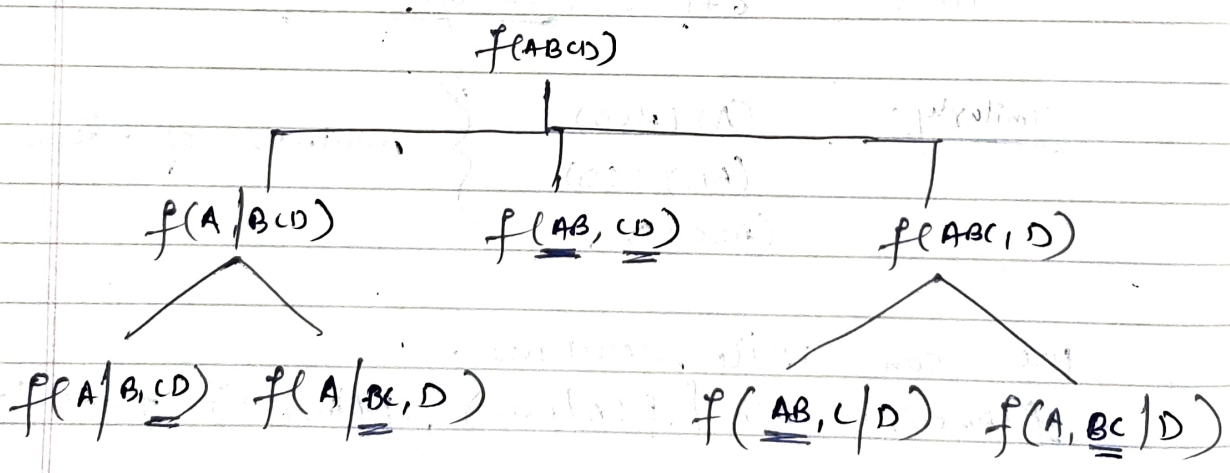
Ex :- $M_1 M_2 M_3 = a[0] \times a[3]$

Recurrence relation becomes -

$$f(M_1 M_2 \dots M_n) = \min \left(f(M_1 \dots M_k) + f(M_{k+1} \dots M_n) + a[0] \times a[k] \times a[N] \right)$$

where $1 < K < N-1$

Checking overlapping Sub problem -



Computation of CD, BC, AB ... repeated
Hence overlapping.

$$f[i,j] = \min_{i \le k < j} \{ f[i,k] + f[k,j] + a_{i-1} a_k a_j \}$$

Top Down Approach

$H_1\ M_2\ M_3\ M_4$

$(a_0 a_1)\ (a_1 a_2)\ (a_2 a_3)\ (a_3 a_4)$ ⟵ size of Matrices.

$$f[1,4] = \min_{k=1,2,3} \begin{cases} f(1,1) + f(2,4) + a_0 a_1 a_4 \\ f(1,2) + f(3,4) + a_0 a_2 a_4 \\ f(1,3) + f(4,4) + a_0 a_3 a_4 \end{cases}$$

$0 + 24 + 10 = 34$

$6 + 60 + 15 = 81$

$18 + 0 + 20 = 38$

$$f[2,4] = \min_{k=2,3} \begin{cases} f(2,2) + f(3,4) + a_1 a_2 a_4 \\ f(2,3) + f(4,4) + a_1 a_3 a_4 \end{cases}$$

$0 + 60 + 30$

$24 + 0 + 40$

$$f(1,2) = \min_{k=1} \{ f(1,1) + f(2,2) + a_0 a_1 a_2 \}$$

Ex:- $\{1, 2, 3, 4, 5\}$.

A B C D

$1\times2 \mid 2\times3 \mid 3\times4 \mid 4\times5$

$a_0\ a_1\ a_2\ a_3\ a_4$

$1\ \ 2\ \ 3\ \ 4\ \ 5$

	1	2	3	4
1	0	6	18	38
2	x	0	24	64
3	x	x	0	60
4	x	x	x	0

cost

	1	2	3	4
1	0	1	2	3
2	x	0	2	3
3	x	x	0	3
4	x	x	x	0

"k"

A B C D

$[(A\times B)\times C]\times D$

$f(1,2) = \min_{k=1} \{ f(1,1) + f(2,2) + a_0 a_1 a_2 \} = 0 + 0 + 6 = 6$

$f(2,3) = \min_{k=2} \{ f(2,2) + f(3,3) + a_1 a_2 a_3 \} = 0 + 0 + 24 = 24$

$f(3,4) = \min_{k=3} \{ f(3,3) + f(4,4) + a_2 a_3 a_4 \} = 0 + 0 + 60 = 60$

$$f(1,3) = \min_{k=1,2} \begin{cases} f(1,1) + f(2,3) + a_0 a_1 a_3 \\ f(1,2) + f(3,3) + a_0 a_2 a_3 \end{cases}$$

$0 + 24 + 8 = 32$

$6 + 0 + 12 = 18$

* Assembly Line Scheduling

Given :- 'n' stations with their processing times on assembly lines, transition time to transfer product from one line to other line, entry & exit time.

Goal :- find minimum production cost

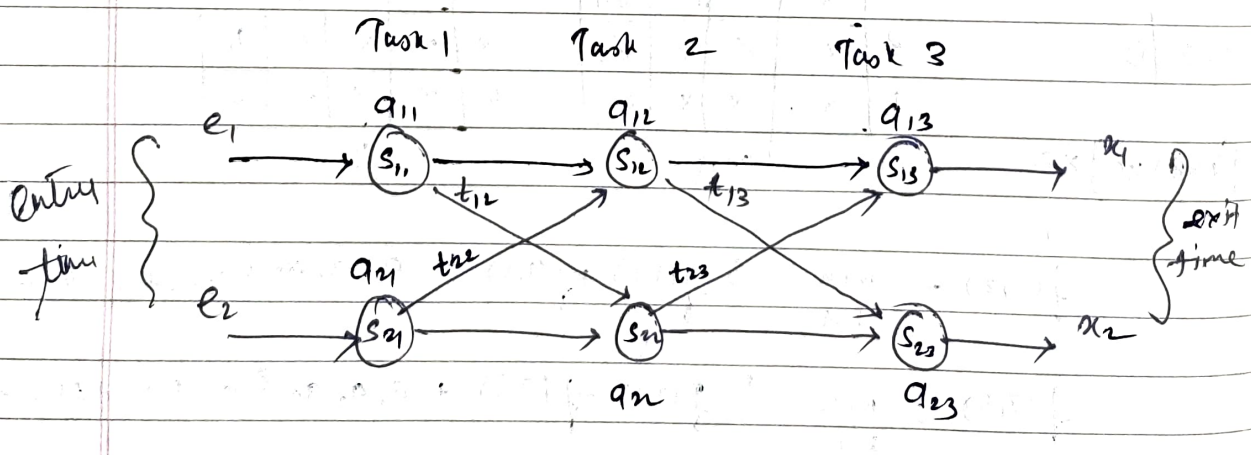
⇒ A factory has 2 assembly lines - each with 'n' stations

$S(i,j)$ ⇒ station 'j' on assembly line 'i'.

$a(i,j)$ ⇒ time taken to process at station 'j' on assembly line 'i'.

$t(i,j)$ ⇒ transition time to transfer the product from line 'j' to j^{th} station on other line.

Example :-



Logic =>

Find solution to smaller problem and use it for longer problem.

If $S(i, j-1)$ ⇒ is known → minimum time taken to leave station $j-1$ from line 'i'

then $S(i,j)$ can be easily calculated using $a(i,j)$ & $t(i,j)$

Let $T1(j)$ ⇒ minimum time taken by ~~car~~ product to leave station 'j' on line 1.

$T2(j)$ ⇒ min. time taken by product to leave station 'j' on line 2.

$$T1(1) \Rightarrow e_1 + a(1,1)$$

Similarly, $T2(1) \Rightarrow e_2 + a(2,1)$

Now for second station on any line has 2 scenarios-

Case 1 :- Product can come from same line

$$T1(j) = T1(j-1) + a(1,j)$$
$$T2(j) = T2(j-1) + a(2,j)$$

Case 2 : Product can come from other line

$$T1(j) = T2(j-1) + t(2,j) + a(1,j)$$

Similarly -

$$T2(j) = T1(j-1) + t(1,j) + a(2,j)$$

We need minimum time -

$$T_1(j) = \min \{ (T_1(j-1) + a(1,j)), (T_2(j-1) + t(2,j) + a(1,j)) \}$$

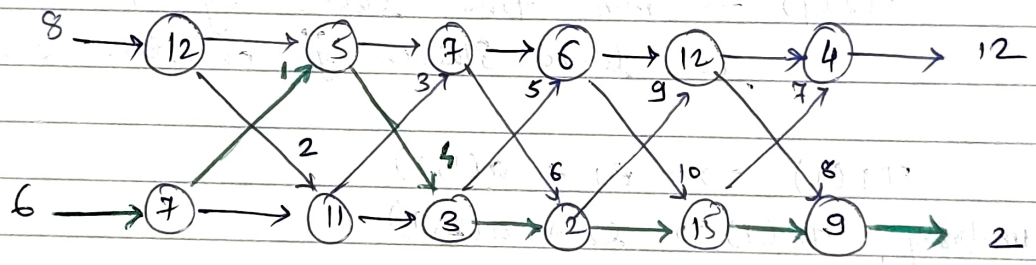
Similarly

$$T_2(j) = \min \{ [T_2(j-1) + a(2,j)], [T_1(j-1) + t(1,j) + a(2,j)] \}$$

Final time

$$T = \min \{ T_1(n) + x_1, T_2(n) + x_2 \}.$$

Example



	1	2	3	4	5	6	
$T_1(j)$	20	19	26	32	44	48+12=60	min
$T_2(j)$	13	24	26	28	43	52+2=54	

	2	3	4	5	6
Assembly line 1	2	1	1	1	1
Assembly line 2	2	1	2	2	2

$T_1(1) = 8+12 = 20$

$T_2(1) = 6+7 = 13$

$$T_1(2) = \min \{ T_1(1) + a(1,2), T_2(1) + t(2,2) + a(1,2) \}$$
$$= \min \{ 20+5, 13+1+5 \}$$
$$= \min \{ 25, 19 \}$$
$$= 19$$

Similarly, fill other entries.