DP - 5

Content

- Matrix Multiplication Basics
- Matrix Chain Multiplication
- Length of longest increasing subsequence.
- Buy and sell stocks

Matrix Multiplication Basics

Rule

A[3 4] × B[4 2] = 3 × 2

A[2 5] × B[5 3] = 2 × 3

A[3 4] × B[5 2] = cannot be ✗

NOTE $M_1[R_1\ C_1]$ $M_2[R_2\ C_2]$

$C_1 == R_2$

$$\underset{3 \times 4}{\overset{M_1}{\begin{bmatrix} 2 & 3 & 4 & 9 \\ 1 & 7 & 6 & 4 \\ 2 & 3 & 1 & 0 \end{bmatrix}}} \quad \underset{4 \times 2}{\overset{M_2}{\begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \\ 0 & 7 \end{bmatrix}}} = \begin{bmatrix} C_{0,0} & \\ & \\ & \end{bmatrix} = 3 * 2$$

$C_{0,0}$ = multiply M_1 0th row with M_2 0th col

$\Longrightarrow \quad 2 \times 1 + 3 \times 2 + 4 \times 3 + 9 \times 0$ 4 multis for $C_{0,0}$

Total elements in resultant matrix $= 3 \times 2$

Per element how many multiplications $= 4$

Total multiplication $= 3 \times 2 \times 4 = 3 \times 4 \times 2$

$$\overset{M_1}{3 \times 5} \quad \overset{M_2}{5 \times 7} \quad \overset{M_3}{7 \times 4}$$

$(M_1 M_2) \times M_3$

$(3 \times 7) \quad (7 \times 4)$

$\Longrightarrow 3 \times 4$

$M_1 \times (M_2 M_3)$

$(3 \times 5) \quad (5 \times 4)$

$\Longrightarrow 3 \times 4$

$$\overset{M_1}{a_1 \times b_1} == \overset{M_2}{a_2 \times b_2} == \overset{M_3}{a_3 \times b_3}$$

$\Longrightarrow$ Resultant matrix $a_1 \; b_3$

Matrix Representation

3 $\quad$ 5 $\quad$ 7 $\quad$ 4

M_1 $\quad$ M_2 $\quad$ M_3

3×5 $\quad$ 5×7 $\quad$ 7×4

$\Rightarrow \quad A[\,] = \{3 \;\; 5 \;\; 7 \;\; 4\}.$

Extra Info

$$A[6] = \left\{ \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 2 & 6 & 5 \end{array} \right\}$$

→ mul all mat from $\boxed{[0 \;\; 3]}$ res mat dimensions $= 2 \times 2$

→ mul all mat from $[1 \;\; 4]$ res mat dimensions $= 3 \times 6$

→ mul all mat from $[0 \;\; 4]$ res mat dimensions $= 2 \times 6$

Generalize $\quad$ Mul of all mat from $[i - j]$

Res matrix size $= A[i] \times A[j]$

Matrix Chain Multiplication

Given $A[N]$. Find **min** cost to multiply all matrices.

$$\begin{array}{cccc} 0 & 1 & 2 & 3 \\ [3, & 5, & 7, & 4] \\ & M_1 & M_2 & M_3 \end{array}$$

105

$3 \times 5 \times 7$ $(M_1 M_2) \times M_3$

(3×7) (7×4)

$3 \times 7 \times 4 = 84$

Total cost/multi = 189

140

$M_1 \times (M_2 M_3)$ $5 \times 7 \times 4$

(3×5) (5×4)

$3 \times 5 \times 4 = 60$

Total cost = 200

$$\begin{array}{cccc} 0 & 1 & 2 & 3 \\ [3, & 5, & 7, & 4] \\ & M_1 & M_2 & M_3 \end{array}$$

DP state $f(0-3)$ = min cost of multiply such that resultant matrix order $A[0] \times A[3]$

$[3, 5, 7, 4]$ M_1 M_2 M_3

$f(0-3)$

$[3, 5, 7, 4]$ $(M_1 \; M_2)(M_3)$

$[3, 5, 7, 4]$ M_1 M_2 M_3

$(M_1)(M_2 \; M_3)$

$f(0-2) + f(2-3)$

$f(0-1) + f(1-3)$

$[3, 5, 7, 4]$ M_1 M_2 M_3

$[3, 5, 7, 4]$ M_1 M_2 M_3

$f(0-1) + f(1-2)$

$f(1-2) + f(2-3)$

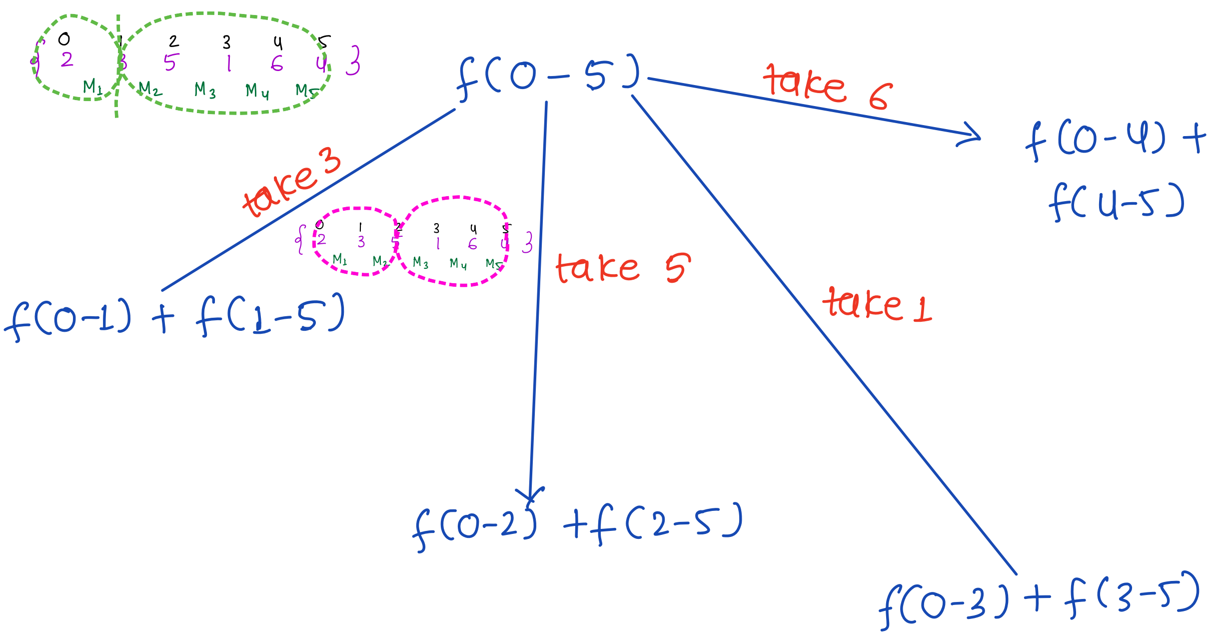
Overlapping Sub problems ✓

Optimal substructure ✓

0	1	2	3	4	5
2	3	5	1	6	4

$\{\ 2 \quad 3 \quad 5 \quad 1 \quad 6 \quad 4\ \}$

$M_1 \quad M_2 \quad M_3 \quad M_4 \quad M_5$



---

dp [i] [j] = min cost of multiply such that resultant matrix order A[i] x A[j]

$$\text{dp expression} = \min \begin{cases} i+1 = f(i,i+1) + f(i+1,j) \\ i+2 = f(i,i+2) + f(i+2,j) \\ \vdots \qquad\qquad \vdots \\ j-2 = \qquad \vdots \\ j-1 = f(i,j-1) + f(j-1,j) \end{cases}$$

$$\min_{\forall\, k=i+1}^{j-1} \underbrace{f(i,k)}_{A[i] \times A[k]} + \underbrace{f(k,j)}_{A[k] \times A[j]} + \underbrace{A[i] \times A[k] \times A[j]}_{\text{cost}}$$

TC : No. of states X TC per state

i $O(N)$ X j $O(N)$

$O(N^2)$ $\qquad$ $O(N)$

$= O(N^3)$

SC : $O(N^2)$

Which state will store my final ans dp[0][n-1]

Break till 9:00 am

Pseudo

```
int MCM(int A[], int i, int j, dp) {
    // Base condition
    if (j == i+1) { // single matrix
        return 0
    }
    if (dp[i][j] != -1) {
        return dp[i][j]
    }
    mincost = INT_MAX
    // take logic
    for (k -> i+1 to j-1) {
        cost = A[i] x A[k] x A[j]
        left = MCM(A, i, k, dp)
        right = MCM(A, k, j, dp)
        overall = left + right + cost
        mincost = min(mincost, overall)
    }

    dp[i][j] = mincost.

    return mincost
}
```

Longest Increasing Subsequence

Given an array find the length of the longest strictly increasing subsequence

Eg: A[5] = {9 2 4 3 10} ans = 3

2 3 10

9 10

~~4 3 10~~

Eg: A[6] = {2 -1 6 3 7 9} ans = 4

2 3 7 9

-1 3 7 9

2 6 7 9

$\longrightarrow$ 2 3 8 1 3

2 3 8 }   3

$\longrightarrow$ 2 2 8 8 4 4 6 6 9 $\dashrightarrow$ 4

length = 4

Bruteforce : Generate all subsequences
for each sub check if they are inc ↗
⇒ keep track of max length.

TC : $O(2^n \times n)$

$f(0-n)$

$$A = \left\{ \begin{array}{ccccccccccc} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 10 & 3 & 12 & 7 & 2 & 9 & 11 & 20 & 11 & 13 & 8 \end{array} \right\}$$

take ↙ ↘ dont

$1 + f(1-n)$ $\quad\quad$ $f(1-n)$

TODO

$f(0-10, -\infty)$

t ↙ ↘ d

$1 + f(1-10, 10)$ $\quad\quad$ $f(1-10, -\infty)$

dp(i) = length of the longest subsequence that ends at ith index

	0	1	2	3	4	5	6	7	8	9	10
A =	10	3	12	7	2	9	11	20	11	13	8
dp	1	1	2	2	1	3	4	5	4	5	3
sub	10	3	3 12	3 7	2	3 7 9	3 7 9 11	3 7 9 11 20	3 7 9 11	3 7 9 11 13	3 7 8

How to find the answer. max(dp)

TC : $O(N^2)$
SC : $O(N)$

Pseudo code

```
int LIS ( A[] ) {

    dp[N] = {-1}    // Init

    for ( i -> 0 to n-1 ) {
        length = 0
        for ( j -> 0 to i-1 ) {

            if( A[j] < A[i]) {
                length = max (length , dp [j])
            }

        }

        dp [i] = length +1 .
    }

    return max (dp)
```

Q 3> Given stock prices over N days.

- You can buy on any day.
- You can sell on any day after buying

You can only do it atmost once.

Find max profit ?

A = { 3 5 2 1 4 5 2 }

you have to sell here

idea — at each and every index find min of everything towards the left.

$TC = O(N^2)$

```
int profit ( A[] ) {

    maxProfit = 0

    for ( i -> 0 to n-1 ) {
            minPrice = ∞
        for ( j -> 0 to i-1 ) {
            minPrice = min(minPrice , A[j])
        }
        p = A[i] - minPrice
        maxProfit = max(maxProfit , p)
    }

    return maxProfit
}
```

}

```
int profit ( A[]) {                    O(N)

    maxProfit = 0
    minPrice = ∞
    for( i ⟶ 0 to n-1) {

        minPrice = min(minPrice , A[i])
        p = A[i] - minPrice
        maxProfit = max (maxProfit , p)

    }

    return maxProfit
}
```

Doubt session

25 min