Today's Content :

→ N Houses }

→ Longest Increasing SubSequence }

→ MCM

→ { → }

N Houses:

Given N houses & Cost associated to Colour each house in R/G/B
find min Cost to paint all houses

Note: No 2 Adjacent house should have same colour

→ Cost → R[], G[], B[]

N:	1	2	3	4	5
R	5	8	4	2	1
G	2	1	5	6	7
B	6	5	7	4	5

R G R - 10 → min cost
B G R - 11
G B R - 11

Idea: Greedy logic of picking min at every house wont work

1) optimal Substructure
2) overlapping Subproblems

5:R → min{ 4:G , 4:B }

5:G → min{ 4:B , 4:R }

5:B → min{ 4:R , 4:G }

4:G → min Cost to paint all house [1-4] such that 4^{th}: G
4:B → min Cost to paint all house [1-4] such that 4 : B
4:R → min Cost to paint all house [1-4] such that 4 : R
5:R → min Cost to paint all house [1-5] such that 5^{th}: R

// _ _ _ _ _ _ (i^{th})

$dp[i, R]$ = Min cost to paint all house $[1-i]$ such that i^{th} colour = Red

$dp[i, G]$ = Min cost to paint all house $[1-i]$ such that i^{th} colour = Green

$dp[i, B]$ = Min cost to paint all house $[1-i]$ such that i^{th} colour = Blue

// dp Expression

cost of it is cost to fill i^{th} building in Red

$$dp[i, R] = R[i] + \min(dp[i-1, G], dp[i-1, B])$$

$$dp[i, B] = B[i] + \min(dp[i-1, G], dp[i-1, R])$$

$$dp[i, G] = G[i] + \min(dp[i-1, B], dp[i-1, R])$$

// Table:

dpR[N+1], dpG[N+1], dpB[N+1] → dp[N+1][3] →

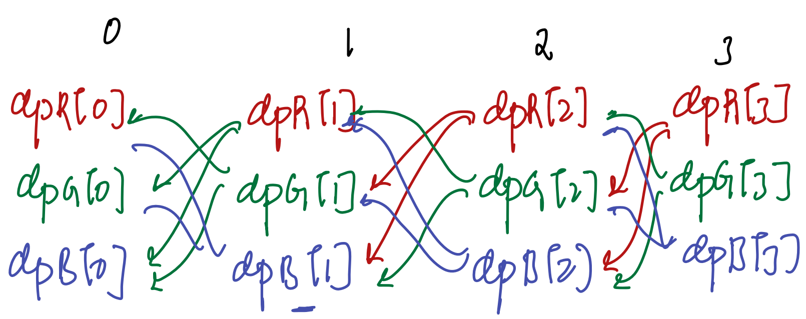
- dp[N+1][0] → R
- dp[N+1][1] → B
- dp[N+1][2] → G

// Base Condition:

dpR[0] = dpG[0] = dpB[0] = 0

→ 0 indicate no house hence no cost

// N=4: { Fill All dpR[], dpG[], dpB[], for a i^{th} house & goto next house }



```
i = 1; i <= N; i++){
    dpR[i] = R[i] + min( dpG[i-1], dpB[i-1])
    dpB[i] = B[i] + min( dpR[i-1], dpG[i-1])
    dpG[i] = G[i] + min( dpR[i-1], dpB[i-1])
}

return min( dpR[N], dpB[N], dpG[N])
```

TC: $O(N)$

SC: $O(N)$ $\xrightarrow{\text{We can optimize}}$ 6 Variables → $O(1)$

N:	1	2	3	4
R	5	8	4	2
G	2	1	5	6
B	6	5	7	4

_ _ B   R

g   + 4

[i j]

[i, N-1]

	0	1	2	3	4
dpR :	0	5	10	10	14
dpG :	0	2	6	12	16
dpB :	0	6	7	13	14

number of ways to have fun.

i^{th} → pizza, tv, sleep

Try it out with 3 1 Dp Expressions

2Q) Given ar[N], find length of longest increasing Subsequence?

order based on indexing

	0	1	2	3	4	5	6	7	8	9	10	11
ar[ ]:	10	3	12	7	2	9	11	20	11	13	6	8
dp[ ]:												

index: len

5 6 9 : 3

3 5 8 10 :

// BF: TC: { 2^N } → bit masking, Back tracking

→ generate all subsequence & for every subsequence check wether it's increasing or not

[0 11] → max inc subsequence from [0 11]

[0 11] → $*$ [0 10]

[0 11] → Pick → { ar[11] } → [0 10] + 1 → max inc subseque

{ for max inc subsequence from [0 10], what will be last Element? } → is not possible

obs: We need to know element we are picking?

dp[i] = { Max len inc subsequence from [0 i], ending at i^{th} index }

→ should contain i^{th} index

				[1]	[ ]	[1, 3, 4]	[0, 1, 3, 4, 5]	[ ]	[1, 3, 5]	[ ]		
	0	1	2	3	4	5	6	7	8	9	10	11
ar[ ] :	10	3	12	7	2	9	11	20	11	13	6	8
dp[ ] :	1	1	2	2	1	3	4	5	4	5	2	3

↳ final ans → overall max

// dp[N]

dp[i] = { max len in subseq from [0 - i] containing i^{th} element }

$$dp[i] = \boxed{\max_{\forall\, j=0 \ldots i-1 \;\&\&\; A[j] < A[i]} (dp[j])} + 1$$

dp[0] = 1;

TC → O(N) × (N) → O(N²)

SC → O(N)

→ 10:58 PM

```
i = 1; i < N; i++){
    c = 0
    j = 0; j < i; j++){
        if( A[j] < A[i] ) { c = max (c, dp[j])
    }
    dp[i] = c + 1
}
return max (dp[])
```

// Note: We cannot carry max value c, because each state will depend on different subproblems

38) Matrix Chain Multiplication :

$\underline{M_1}$ $3 * 4$ $\underline{M_2}$ $4 * 2$ $\underline{Ans}$ $3 * 2 \longrightarrow 3 * 2 * \{4\}$ Single Elem

$$
\begin{array}{c|cccc} & 0 & 1 & 2 & 3 \\ \hline 0 & 2 & 1 & 2 & 3 \\ 1 & 1 & 4 & 2 & 6 \\ 2 & 3 & 2 & 9 & 8 \end{array}
*
\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & 1 \\ 1 & 1 & -1 \\ 2 & -1 & 2 \\ 3 & 2 & 0 \end{array}
=
\begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 5 & 5 \\ 1 & - & - \\ 2 & - & 19 \end{array}
$$

$\underline{M_1}$ $a * b$ $*$ $\underline{M_2}$ $b * c$ $\underline{M_3}$ $a * c$ } operations: $a * c * \{b\}$ $\rightarrow$ Total operations.

3 Matrices ?

M_1	M_2	M_3	Ans
$3 * 5$	$5 * 7$	$7 * 4$	$3 * 4$

Case-I : $M_1 * \{M_2 * M_3\}$

: Cost $M_2 M_3$ + Cost $M_1 * \{M_2 M_3\}$ ($3*5$, $5*4$)

$140 + 60 = 200$

Case-II : $(M_1 * M_2) * M_3$

: Cost $M_1 M_2$ + Cost $\{M_1 M_2\} * M_3$ ($\{3 * 7\}$, $\{7 * 4\}$)

$105 + 84 = 189$

// final matrix is same ✓

i) operations are different

} // given N matrices find min cost to multiply all of them $\Rightarrow$ {MCM}

// N = 4:

M_1		M_2		M_3		M_4	
a_1	b_1	a_2	b_2	a_3	b_3	a_4	b_4
1	2		3		4		5

N = 5 : value →
: 6 value

N = n : value → {n + 1}

// N = 5:

	M_1					
	[0	1]	2	3	4	5
val[6] :	3	4	2	5	6	2

M_1 : [0 1]
M_2 : [1 2]
M_3 : [2 3]
M_4 : [3 4]
M_5 : [4 5]

M_i : [v[i-1], v[i]]

i^{th} matrix dimensions

// Say we mult all [i - j] = val[i-1] * val[j]

	M_1					
	[0	1]	2	3	4	5
val[6] :	3	4	2	5	6	2

final dimension

M_1 : [0 1]
M_2 : [1 2]
M_3 : [2 3]
M_4 : [3 4]
M_5 : [4 5]

// Say we mult all [1 - 3] = val[0] val[3] → {3*5}

1: val[0] val[1]
2: val[1] val[2]
3: val[2] val[3]

// Say we mult all [2 - 5] = val[1] * val[5] → {4*2}

// Say we mult all [i - j] = val[i-1] * val[j]

// N = 5 : { we need min cost to mul [1 - 5] }

{ min cost to multiply all mat 1 - 5]

[1 - 5] ⇒ [V_0 V_5]

Min cost to mult all [2 - 5]

[1] * [2 - 5]

c_1 c_2

[1 2] * [3 5]

[1 3] * [4 5]

[1 4] * [5]

[V_0 V_2] [V_2 V_5] ⇒ $V_0 V_2 V_5$

→ $c_1 + c_2 + V_0 V_2 V_5$

[2 2] * [3 5]

[2 3] [4 5]

[2 4] [5]

// dp[i j] = Min cost to mul all mat from [i - j]

// dp[i j] = min { k = i; k < j; k++) { dp[i k] + dp[k+1, j] + v[i-1] * v[k] * v[j] } }

dp[i k]: cost to multiply all [i k]

dp[k+1, j]: cost to multiply all [k+1 j]

v[i-1] * v[k] * v[j]: cost to multiply 2 resultant matrices

dp[i j]: Result v[i-1], v[j]

Resultant [v[i-1], v[k]] * Result [v[k], v[j]]

// dp[N+1][N+1] = {-1}   {min cost to mult all mat [i-j]}

```
int mincost( int v[], int i, int j ){
    if (i==j) { return 0 }
    if (dp[i][j] == -1) {          // it is getting called first time
        ans = INT_MAX
        k=i; k<j; k++) {
            int l = mincost (v, i, k)      = dp[i k]
            int r = mincost( v, k+1, j)    = dp[k+1, j]
            int c = v[i-1] * v[k] * v[j]
            ans = min( ans, l+r+c)
        }
        dp[i][j] = ans;
    }
    return dp[i][j]
}
```

// Final Ans :

→ min cost to mult all mat from 1, N => dp[1][N]

TC: $O(N^2) * \{N\} \Rightarrow O(N^3)$ SC => $O(N^2)$

```
main() {
    mincorr(V, 1, N)
}
```