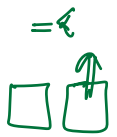
1) Longest Increasing subsequence — find the length of LIS.

10	3	12	7	2	9	11	20	11	13	6	8

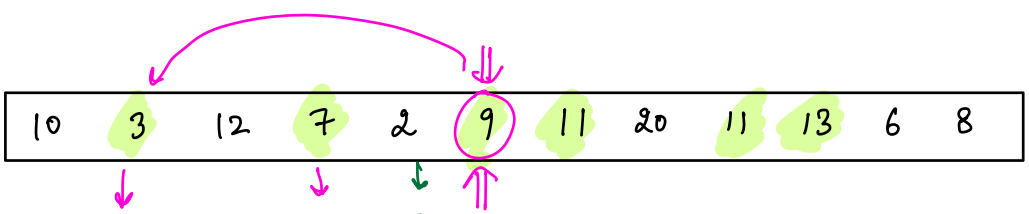
2 9 11

3 7 9 11 11 13



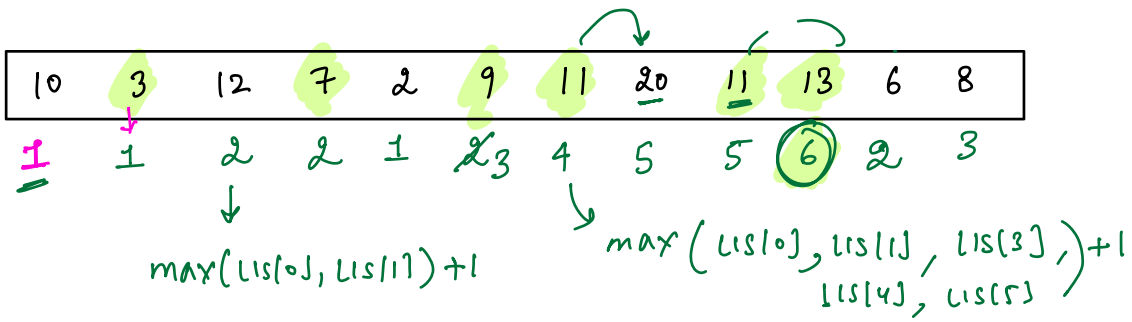
idea: Find all possible valid subsequences $\approx O(2^n)$

go to every element → find LIS ending at that element



$$\max\{x+1, y+1, z+1\}$$

{ LIS[i] = longest increasing subsequence ending at i }



$$LIS[i] = 1 + \max_{\substack{j<i,\ arr[j]<=arr[i] \\ j=0}} \{ LIS[j] \}$$

```
for( int i=0; i<n; i++)
{
    dp[i]=0;
    for( j=0; j<i; j++)
    {
        if( arr[j] <= arr[i])
        {
            dp[i] = max(dp[i], dp[j]);
        }
    }
    dp[i]++;
}

int ans=0;
for( i=0 → n)
{
    ans = max(ans, dp[i]);
}
```

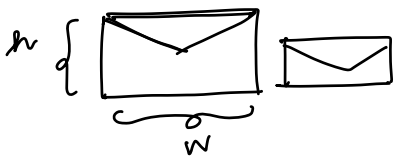
T.C: $O(n^2)$

S.C: $O(n)$

HW: Try finding out LIS not just the length!

Q Russian doll envelops

N diff envelopes.
height and width.



Find max count of envelope that can be put a side envelope.

A B

$h_A < h_B$ } A can fit into B
$w_A < w_B$

N = 4

	h		w
1	5	*	6
2	6	*	4
3	6	*	7
4	4	*	3

4 → 1 } 3
1 → 3

sort by height

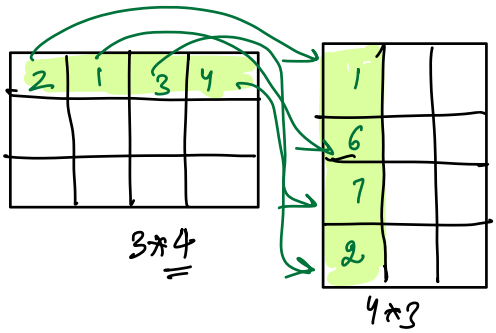
h	4	5	6	6
w	3	6	4	7

→ LIS

T.C. $n \log n + n^2$

MCM - Matrix chain multiplication

M_1	M_2	ans
$3*7$	$7*4$	$3*4$
$3*9$	$8*9$	Mul ✗

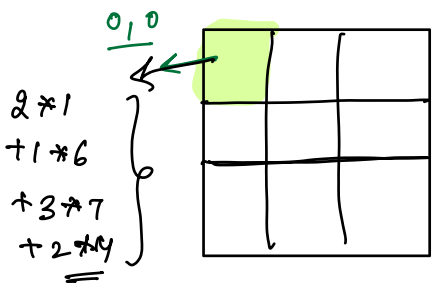


$a*b \quad b*c$

$\Downarrow$

$a*c * b$

↑ iterating b times for every cell



$$ans[i][j] = \sum_{k=0}^{k<b} A[i][k] * B[k][j]$$

M_1	M_2	M_3	
$3*5$	$5*4$	$4*7$	$\Rightarrow 3*7$

$(M_1 M_2) * M_3$

$(M_1 \; M_2)$ M_3

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

$3*4$

$3*5*4 + 3*4*7 = 144$

$M_1 * (M_2 M_3)$

$M_1 \quad (M_2 \; M_3)$

$3*5 \quad 5*7$

$5*4*7 + 3*5*7 = 245$

{ Find min cost to multiply a chain of matrices }

Array of N positive int

N = 4

N-1 matrices

3 5 7 4

M_1 M_2 M_3

Matrix

1 arr[0] * arr[1]

2 arr[1] * arr[2]

⋮

i arr[i-1] * arr[i]

1 * 2

arr[0] * arr[2]

arr[1] * arr[2]

size

arr[0] * arr[2]

3 → 5

4

arr[2] * arr[3]

arr[2] * arr[4]

arr[4] * arr[5]

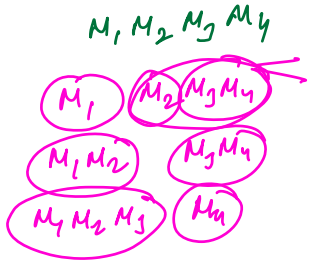
arr[2] * arr[5]

i — j

arr[i-1] * arr[j]

min cost to multiply (1 → 8)

1 2 3 4 5 6 7 8



min

[1-1] * [2-8] + arr[0] * arr[1] * arr[8]

[1-1] → c1 : arr[0] * arr[1]

[2-8] → c2 : arr[1] * arr[8]

[1-2] * [3-8] + arr[0] * arr[2] * arr[8]

[1-2] → c1

[3-8] → c2

[1-3]	[4-8]
[1-4]	[5-8]
[1-5]	[6-8]
[1-7]	[8-8]

[3-8]

[3-3]	[4-8]
[3-4]	[5-8]
[3-6]	[7-8]
[3-7]	[8-8]

mincost (i, j) = min cost to multiply matrix from (i, j)

min

[i, i] [i+1, j]
[i, i+1] [i+2, j]
[i, i+2] [i+3, j]
⋮
[i, j-1] [j, j]

(k = i, k+1 = i+1)

$$\text{mincost}(i,j) = \min_{k=i \text{ to } j-1} \begin{cases} \text{mincost}(i,k) + \\ \text{mincost}(k+1,j) + \\ arr[i-1] * arr[k] * arr[j] \end{cases}$$

(i - k) + (k+1 - j)

arr[i-1] * arr[k] * arr[j]

dp[N][N]

matrix
1 → N-1

```
int mncost(int arr[], int i, int j)=>
{
    if(i==j) return 0;
    if(dp[i][j]!=-1) return dp[i][j];
    dp[i][j]= INT_MAX;
    for(k=i; k<j; k++)
    {
    dp[i][j]= min(dp[i][j], mncost(i,k) +
                  mncost(k+1,j) +
                  arr[i-1]*arr[k]*arr[j]);
    }
    return dp[i][j];
```

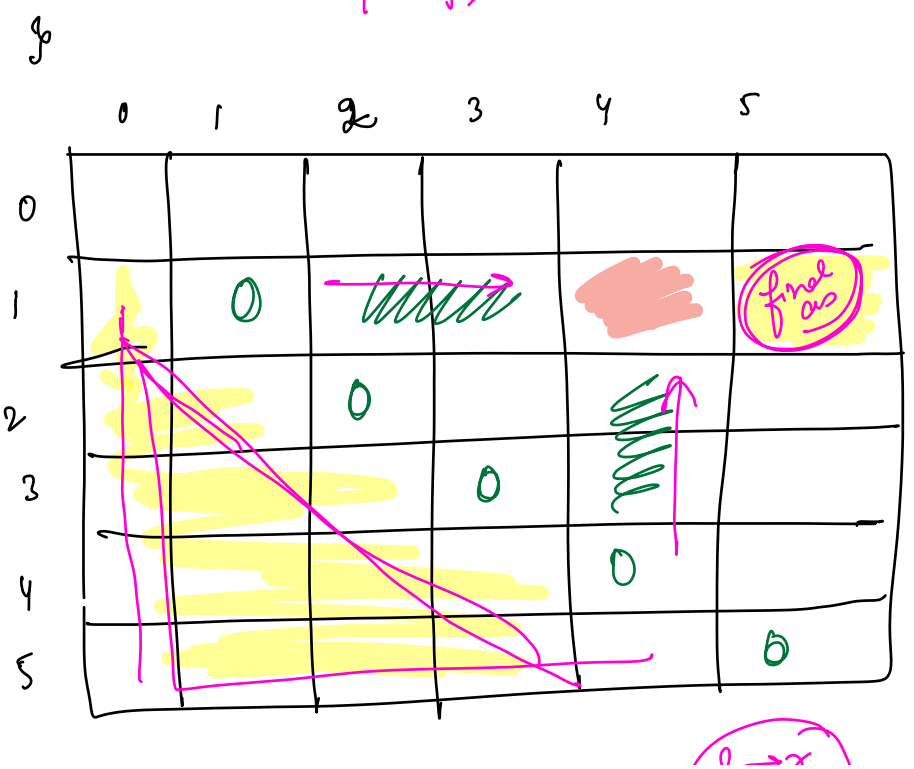
final ans
dp[1][N-1]

every problem is solved via a loop

T.C.: n*n*n
= O(n^3)

N=6

dp[i][i]=0



1→4
1→1, 2→4

dp[1][4]
= dp[1][1] dp[2][4]
1,2
2,4
1,2
4,4

0 → n

b → r

```
for( i = n-1; i >= 1; i--)
{
    for( j = i ; j < n; j++)
    {
        if( i == j) dp[i][j] = 0;
        for( k = i; k < j; k++)
        {
            recurrence
        }
    }
}
```

Q longest palindromic subsequence

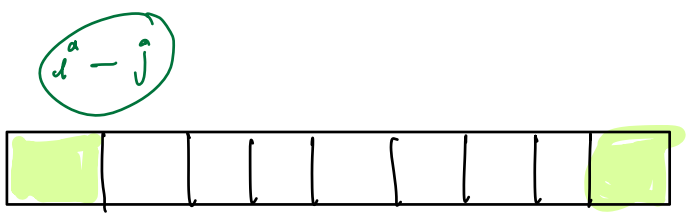
e b g a h j c k e a b

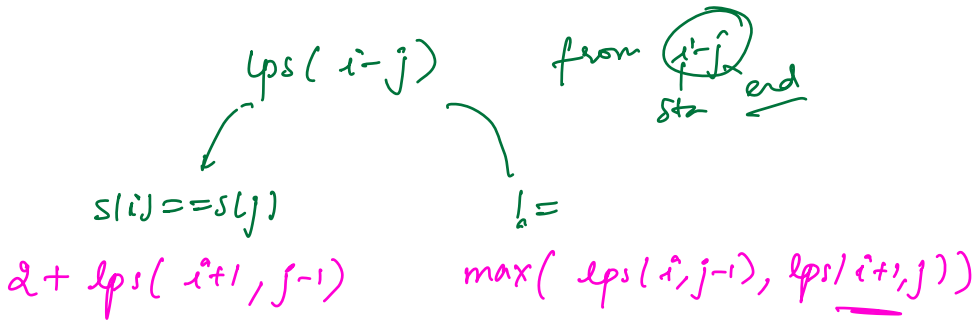
b a e k c j h a g b e

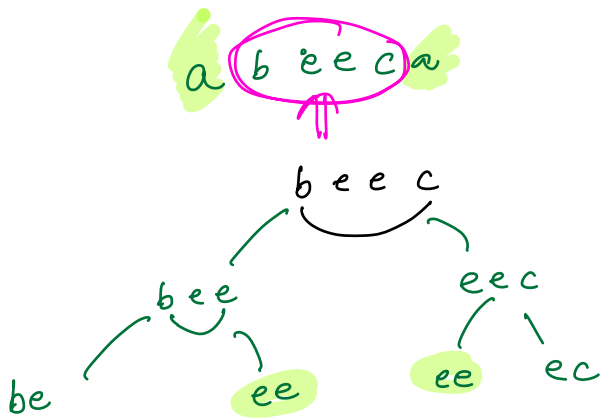
palindrome

s = rev(s)

lcs of (s, rev(s))




lps(i-j)
from
i-j
str
end
s[i]==s[j]
!=
2 + lps(i+1, j-1)
max(lps(i,j-1), lps(i+1,j))


a b e e c a
b e e c
b e e
e e c
be
ee
ee
ec