Q.1 Given an Array of size N, find the length of longest Increasing Subsequence (LIS)

Note :- Subsequence elements should be in strictly increasing order. $a_1 < a_2 < a_3 < \dots\dots < a_n$

$A : \{\overset{0}{9}, \overset{1}{2}, \overset{2}{4}, \overset{3}{3}, \overset{4}{10}\}$

$\{2, 4, 10\} \Rightarrow ③$

$\{2, 3, 10\} \Rightarrow ③$

$A : \{\overset{0}{2}, \overset{1}{-1}, \overset{2}{6}, \overset{3}{3}, \overset{4}{7}, \overset{5}{9}\} \Rightarrow \underline{\underline{4}}$

$\{2, 6, 7, 9\} \Rightarrow \underline{\underline{4}}$

$\{-1, 6, 7, 9\} \Rightarrow 4$

$\{2, 6, 7\} \Rightarrow \underline{\underline{3}}$

$\{2, 3, 7, 9\} \Rightarrow \underline{\underline{4}}$

Idea1 :-

For every subsequence, check if it is strictly increasing or not & get max length.

↙ ↘

Bit Masking

TC: $O(N \cdot 2^N)$

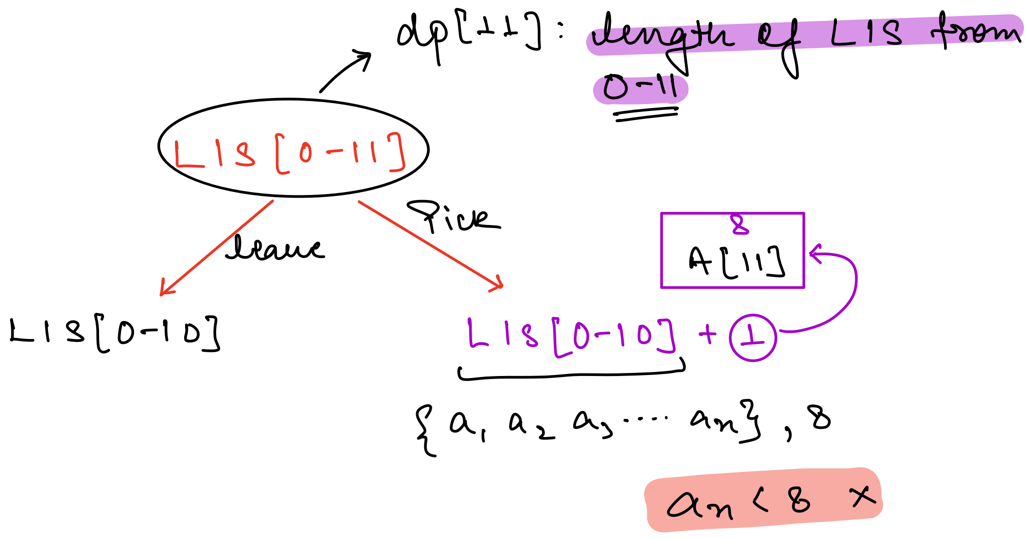
Backtracking

TC: $O(2^N)$

$1 <= N <= 10^3 \Rightarrow 2^{10^3} : \underline{\underline{2^{1000}}}$ ×

Note : If we have a backtracking sol^n with very high constraints, think about Dynamic Programming sol^n.

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



Issue :- We don't know the end of subsequence.

dp[i] : Length of LIS from [0-i] ending at ith index [A[i] is a part of this subsequence].

	0	1	2	3	4	5	6	7	8	9	10	11
A:	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
			{3,12} {10,12}	{3,7}	{2}	{3,7,9}	{3,7,9,11}	{3,7,9,11,20}	{3,7,9,11}	{3,7,9,11,13}	{3,6} {2,6}	

dp[i]: Length of LIS ending at i.

$$dp[i] = \max\left(\underset{\substack{j=0 \\ (A[j] < A[i])}}{\overset{j<i}{\forall}} dp[j] \right) + 1$$

including the ith index.

Base Case :

$$dp[0] = 1$$

Table :- int dp[N];

```
int LIS ( int a[], int N ){
        int dp[N];
        dp[0] = 1
        for ( i= 1; i< N; i++) {
                //dp[i]
                V = 0
                for ( j= 0; j< i ; j++){
                        if(a[j] < a[i]){
                                V= max(V, dp[j]);
                        }
                }
                dp[i] = V+1;
        }
        return max(dp Array);
}
```

TC: $\underbrace{\#\text{ of states}} \times$ TC of each state

$$O(N \times N) \Rightarrow \underline{O(N^2)}$$

SC : $O(N) \xrightarrow{\text{optimise ?}} \times$

$$O(N^2) \longrightarrow \boxed{\begin{array}{c} O(N \log N) \\ (DP + BS) \end{array}}$$

Problem Solving (Optimal)

Q. N Houses.

Given N houses & cost associated to paint each house in R/G/B. find the minimum cost to paint all the houses.

Note:- No 2 adjacent houses should have same color.

N=3	1	2	3
R	5	8	4
G	2	1	5
B	6	9	7

G R G : 2+8+5 = 15

B R B : 21

R G R : 10

R G B : 13

$\underbrace{G\ G}_{\times}$ R : ×

Idea:- Try out all the combinations:-

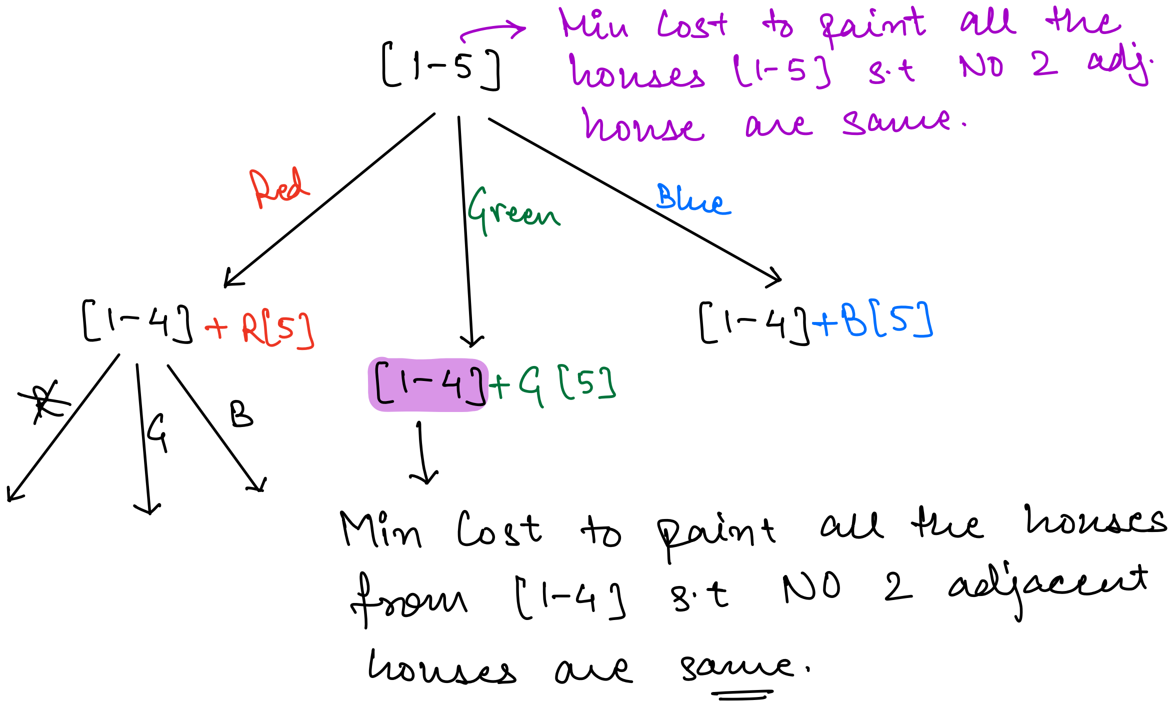
→ 3^N Combinations.

→ By neglecting few combinations & making sure that no two adjacent houses are same.

$3 \times 2^{N-1}$

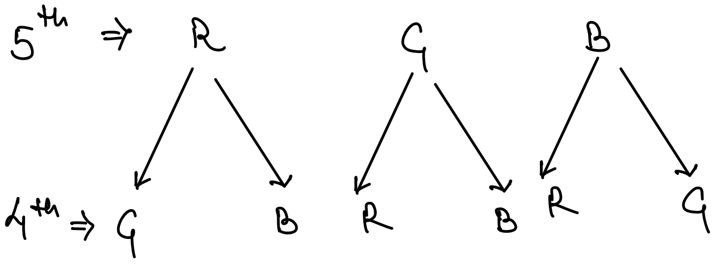
⇒ N=5

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



Min Cost to paint all the houses [1-5] s.t NO 2 adj. house are same.

Min Cost to paint all the houses from [1-4] s.t NO 2 adjacent houses are same.



⇒ We should also store the color of ith house.

$$\boxed{\begin{aligned} dp[i][R] &= \min(dp[i-1][G],\ dp[i-1][B]) + R[i] \\ dp[i][G] &= \min(dp[i-1][R],\ dp[i-1][B]) + G[i] \\ dp[i][B] &= \min(dp[i-1][R],\ dp[i-1][G]) + B[i] \end{aligned}}$$

$$\left.\begin{array}{l} R \rightarrow 0 \\ B \rightarrow 1 \\ G \rightarrow 2 \end{array}\right\}$$

<u>Base</u> <u>Condition</u> :-

$i == 0$

$dp[0][0] = dp[0][1] = dp[0][2] = 0$

<u>DP</u> <u>table</u>

int dp[N+1][3]

Code :-

R[i] → cost to color ith house in Red

G[i] → cost to color ith house in Green

B[i] → cost to color ith house in Blue

```
int dp[N+1][3]
dp[0][0] = dp[0][1] = dp[0][2] = 0
for( i = 1 ; i <= N ; i++ ) {
    dp[i][0] = R[i] + min(dp[i-1][1] , dp[i-1][2])
    dp[i][1] = B[i] + min(dp[i-1][0] , dp[i-1][2])
    dp[i][2] = G[i] + min(dp[i-1][0] , dp[i-1][1])
}
  return  min(dp[N][0] , dp[N][1] , dp[N][2]);
```

TC: 3×N

O(N)

SC: O(N)

Optimise ?

Yes.

We only need 6 dp state at any point of time.

⇒ SC: O(1)

N=3

		1	2	3
		0	1	2
0	R	5	8	4
1	G	2	1	5
2	B	6	9	7

dp[4][3]

	0	1	2
0	0	0	0
1	5	2	6
2	10	6	11
3	10	15	13