Q.1 Longest Increasing Subsequence (LIS)

Given an array, find the length of longest increasing subsequence (LIS)

Note:- Increasing subsequence means strictly increasing.

↳ Order matters.

A: {9 2 4 3 10} ⇒ 2, 4, 10
2, 3, 10

LIS ⇒ 3.

A: {2 -1 6 3 7 9}

↳ {2 3 7 9}
{2 6 7 9}
{-1 6 7 9}
{-1 3 7 9}

LIS = 4.

No. of subsequences = 2^N

$a_0 \quad a_1 \quad a_2 \quad - - - - \quad a_{n-1}$

I E I E

⇒ 2^N

⇒ Empty sequence is also a subsequence.

⇒ Archive : Intermediate DSA : Subsequences & subset.

Brute force

1) Backtracking $\Rightarrow O(2^N)$
2) Bit Masking $\Rightarrow O(N \cdot 2^N)$

$\Rightarrow$ $\boxed{N <= 10^4}$

NOTE : If we have a recursive / backtracking sol^n and the constraints are tigh then that problem is a D.P. problem.

$\Rightarrow$

A[12] : {

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

}

LIS[0-i] $\Rightarrow$ length of LIS from index 0 to i

LIS[0-11]

inc $\rightarrow$ LIS[0-10] + 1

exc $\rightarrow$ LIS[0-10]

$\{a_1, a_2 \cdots a_x\}$ 8

$a_x < 8$

$\Rightarrow$ here, we don't know the ending of a subseq.

LIS[i] : length of longest Increasing Subsequence from index 0 to i ending at index i.
{A[i]}

A[12] : { 10 3 12 7 2 9 11 20 11 13 6 8 }

index	0	1	2	3	4	5	6	7	8	9	10	11
A[12]	10	3	12	7	2	9	11	20	11	13	6	8
LIS[12]	1	1	2	2	1	3	4	5	4	5	2	3
	{10}	{3}	{10, 12} {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}	{2, 6} {3, 6}	{2, 6, 8}

final ans ⇒ MAX of LIS[].

dp[i] = length of LIS ending at index i.

dp Expression :

$$\boxed{dp[i] = \underset{A[i] > A[j]}{Max}\left\{ \forall_{j=i-1}^{0} dp[j] \right\} + 1}$$

dp[0] = 1

int dp[N];

int LIS (int arr[ ], N){

```
int LIS (int arr[ ], N){
    int dp[N];
    dp[0] = 1;
    for( i= 1; i<N; i++){
        s = 0
        for( j= i-1; j>=0; j--){
            if (A[i] > A[j]) {
                s = max(s, dp[j]);
            }
        }
        dp[i] = s+1;
    }
    return max(dp[ ]);
}
```

Instead of max(), we can maintain max variable duting loop

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

Optimise

$O(N \log N)$

Q.2 N Houses.

Given N Houses & cost associated to color each house with R/G/B. Find min cost to color all the houses s.t no two adjacent houses have same color

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

G R G $\Rightarrow$ 15 (2+8+5)

G B R $\Rightarrow$ 15

B G B $\Rightarrow$ 14

R G R $\Rightarrow$ $\boxed{10}$ * ans.

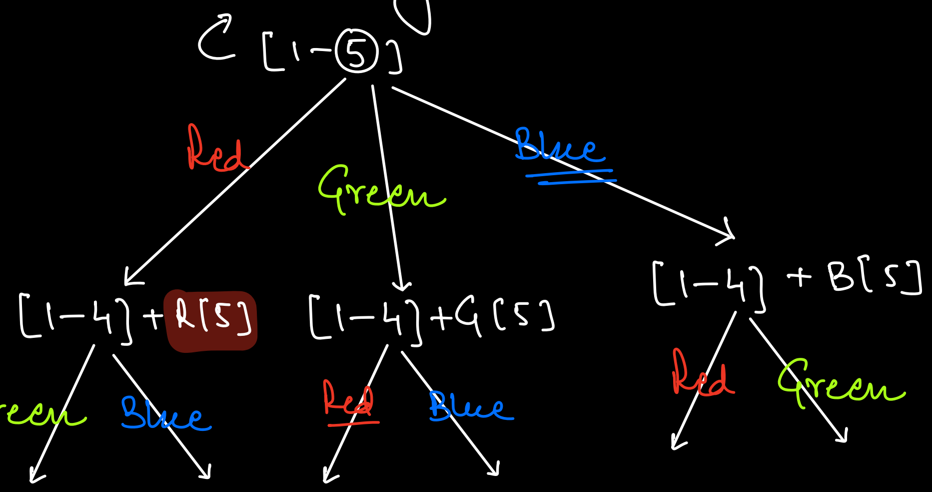
Try out all the possibilities.

$$3 \times 3 \times 3 \cdots 3 \Rightarrow 3^N$$

$$\underset{\substack{R\\G\\B\\ \Downarrow \\ 3}}{\_} \quad \underset{2}{\_} \quad \underset{2}{\_} \quad \underset{2}{\_} \quad \_ \cdots \_ \quad \underset{2}{\_}$$

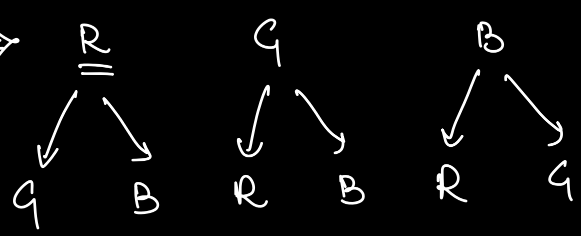
$$\Rightarrow 3 \times 2^{N-1}$$

Min Cost of painting the houses from [1-5]



[1-5] 5^{th} ⇒

dp[5]



Note : We need to know the color of each house along with min cost.

R → 0 , G → 1 , B → 2

dp[i][R] ⇒ Min cost of painting 1 to i houses if i^{th} house is painted with Red.

dp[i][G] ⇒ Min cost of painting [1-i] houses if i^{th} house is painted with Green

dp[i][B] ⇒ Min cost of painting [1-i] houses if i^{th} house is painted with Blue

$$\begin{cases} dp[i][0] = R[i] + \min(dp[i-1][1], dp[i-1][2]) \\ dp[i][1] = G[i] + \min(dp[i-1][0], dp[i-1][2]) \\ dp[i][2] = B[i] + \min(dp[i-1][0], dp[i-1][1]) \end{cases}$$

Minimum

* **Base Case** ⇒ $i = 0$

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

* **dp table size**

`int dp[N+1][3];`

Code

R[ ], G[ ], B[ ] } 1 based indexing

```
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] = G[i] + min(dp[i-1][0], dp[i-1][2]);
    dp[i][2] = B[i] + min(dp[i-1][0], dp[i-1][1]);
}
return min(dp[n][0], dp[n][1], dp[n][2]);
```

row

TC : O(N)

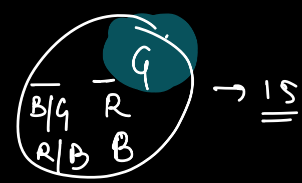
SC : O(N)

Todo

SC can be optimised, as we only need 6 variables at a time.

N=3

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



dp[4][3]

	R (0)	G (1)	B (2)
0	0	0	0
1	5	2	6
2	10	6	11
3	10	15	13

$dp[1][0] = \underbrace{R[1]}_{5} + \min(dp[0][1], dp[0,2])$

$dp[2][0] = \underbrace{R[2]}_{8} + \min(dp[1][1], dp[1][2])$

$dp[2][1] = \underbrace{G[2]}_{1} + \min(dp[1][0], dp[1][2])$

$$dp[2][2] = B[2] + \min(\underline{dp[1][0]}, dp[1][1])$$

$$dp[3][0] = \underset{4}{\underline{R[3]}} + \min(dp[2][1], dp[2][2])$$

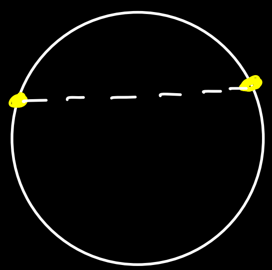
$$dp[3][1] = \underset{5}{\underline{G[3]}} + \min(dp[2][0], dp[2][2])$$

$$dp[3][2] = \underset{7}{\underline{B[3]}} + \min(dp[2][0], dp[2][1])$$

Q.3 Interesting Chords.

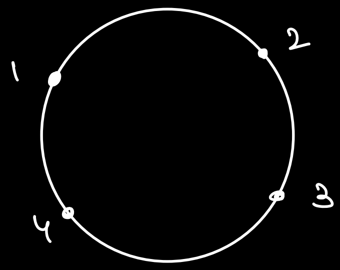
Given 2A no. of points on circle, find no. of ways we can draw A Chords in the circle from 2A points.

Note : No 2 chords should intersect.

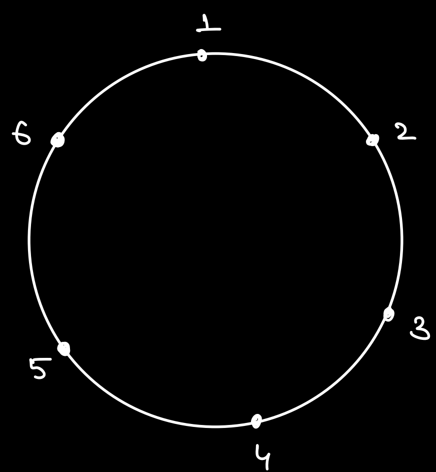


$f(1) = 1$

1 pair of points.



$f(2) = \left.\begin{matrix} \{1-2,\ 3-4\} \\ \{1-4,\ 2-3\} \end{matrix}\right\}$ 2

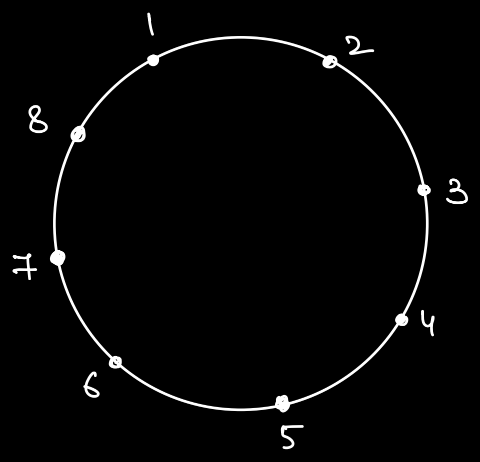


$f(3) =\ 1-2 \left\{\begin{matrix} 3-6,\ 4-5 \\ 5-6,\ 3-4 \end{matrix}\right\}$

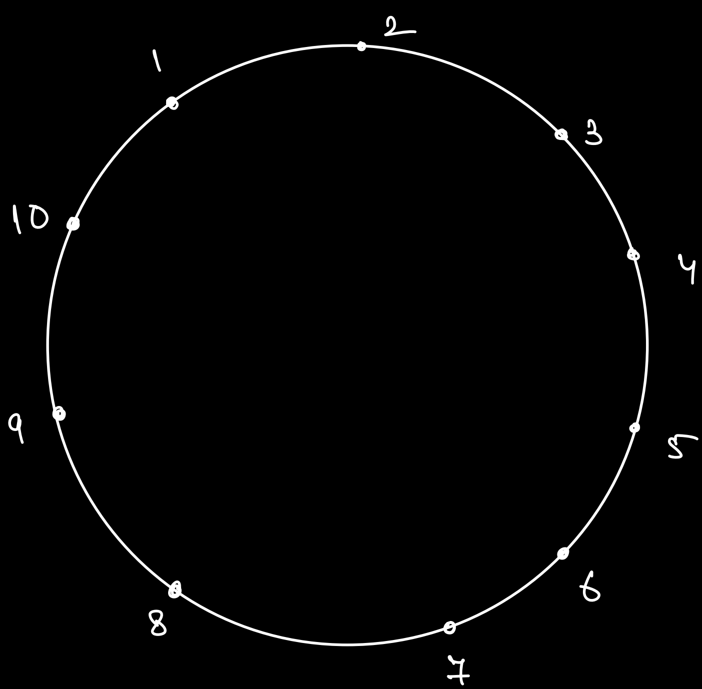
$1-4\ \{2-3,\ 5-6\}$

$1-6 \left\{\begin{matrix} 2-3,\ 4-5 \\ 3-4,\ 2-5 \end{matrix}\right\}$

$f(3) = 5$

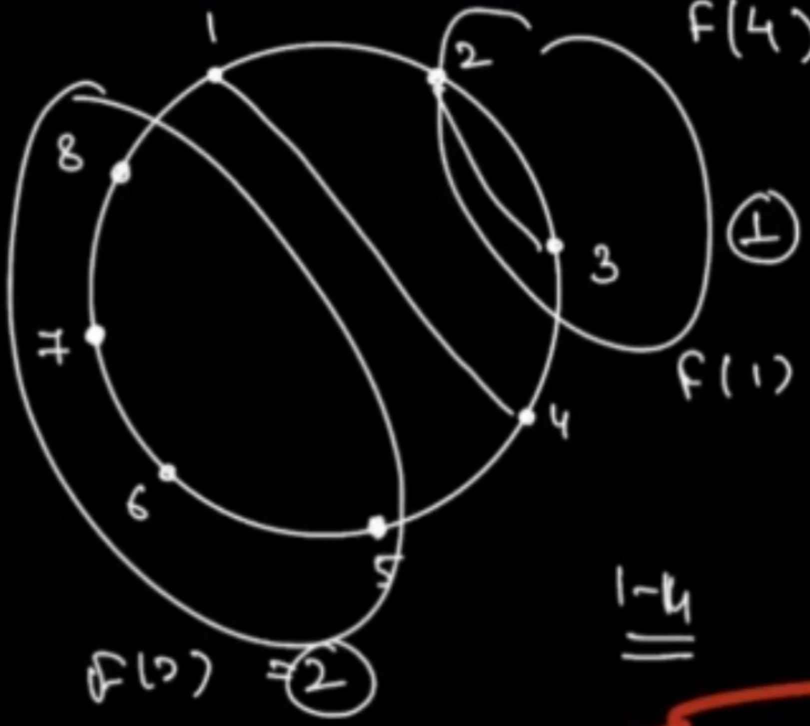


$$f(4) = \underbrace{f(3)}_{1-2} + \underbrace{f(2)\times f(1)}_{1-4} + \underbrace{f(1)\times f(2)}_{1-6} + \underbrace{f(3)}_{1-8}$$



$$f(\underset{\substack{\downarrow \\ N}}{5}) = f(4) + f(3)\times f(1) + f(2)\times f(2) + f(1)\times f(3) + f(4)$$

$$\boxed{\begin{aligned} f(N) = f(N-1) + f(N-2)\cdot f(1) + f(N-2)\cdot f(2) + \ldots\ldots \\ f(N-1) \end{aligned}}$$

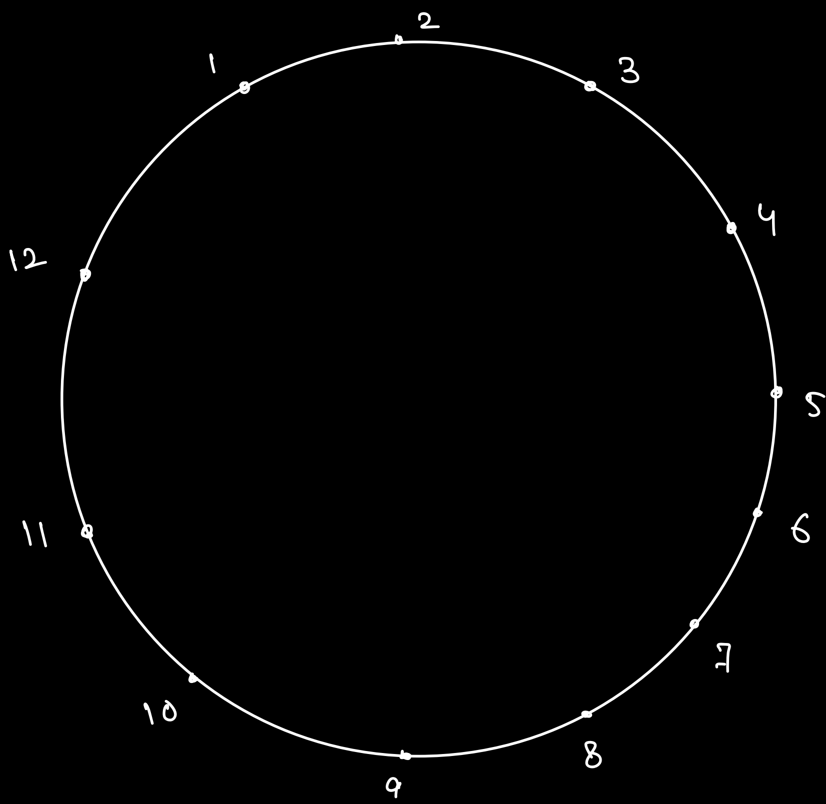


$$f(4) = f(3) + \underline{f(2) \times f(1)}$$

2-3

$\underline{\underline{1\text{-}4}}$

$$1\text{-}4 \; \left[2\text{-}3 \right] \begin{Bmatrix} 8\text{-}7, & 6\text{-}5 \\ \underline{8\text{-}5} & \underline{6\text{-}7} \end{Bmatrix} \Rightarrow 2$$



$$f(6) = f(5) + f(4)\cdot f(1) + f(3)\cdot f(2) + f(2)\cdot f(3) + f(1)\cdot f(4) + f(5) \times$$

$$f(6) = f(6-1) + f(6-2)\cdot f(1) + f(6-3)\cdot f(2) + f(6-4)\cdot f(3) + f(6-5)\cdot f(4) + f(5)$$

$$f(6) = f(5)f(0) + f(4)\cdot f(1) + f(3)\cdot f(2) + f(2)\cdot f(3) + f(1)\cdot f(4) + f(5)f(0)$$

$$\boxed{f(0) = 1}$$

int dp[A+1] ;

→ no. of pairs

dp[i] = # of ways to draw i chords using i pair of points.

$$dp[i] = dp[i-1] \times dp[0] + dp[i-2] \times dp[1] + dp[i-3] \times dp[2] + \ldots + dp[0] \times dp[i-1]$$

```
int dp[A+1];
dp[0] = 1
for( i= 1; i <= A; i++ ) {
    k=0, s=0
    for( j= i-1 ; j>=0; j-- ) {
        s += dp[j]×dp[k]
        k++
    }
    dp[i] = s;
}
return dp[A]
```

TC: $O(A^2)$

SC: $O(A)$

——— * ———