Todays Content:

- → Minimum Failing Path Sum II
- → Intersectings chords in a Circle

Wed: Dp Iteration

Friday: Dp few problems

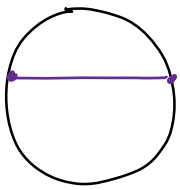
: Stock Dp

IQ) Given N pair of points in a Circle : Points = 2N

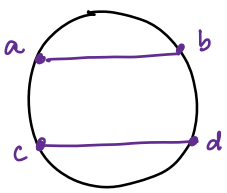
No: of ways to draw N chords such that no 2 chords intersect

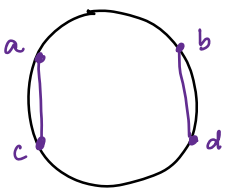
1 chord : have to drawn between 2 points

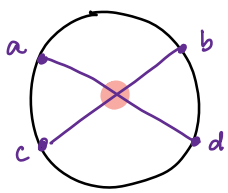
Eg: N = 1 : ch[1] = 1



N = 2 : ch[2] = 2

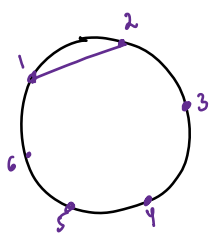




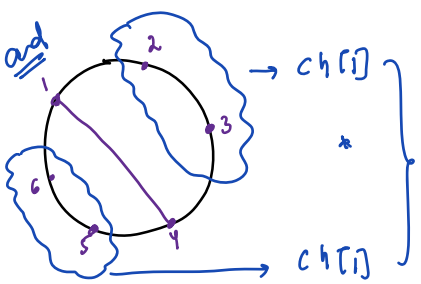


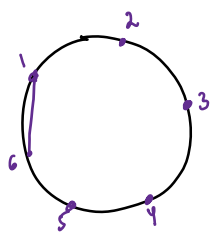
N = 3 : ch[3] = ch[2]*ch[0] + ch[1]*ch[1] + ch[0]*ch[2] = 5 ways

= 2 + 1*1 + 2 = 5

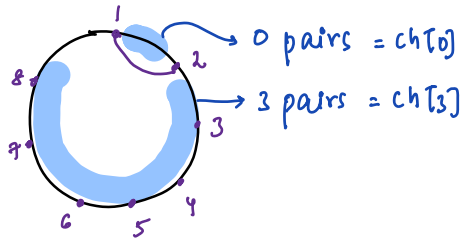


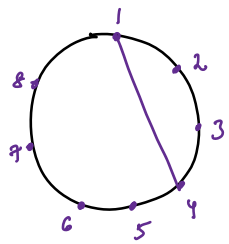
add

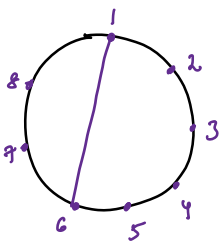


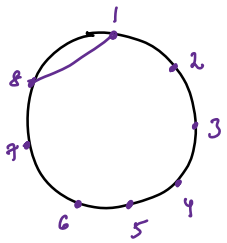


N = 4: ch[4] = ch[3]*ch[0] + ch[2]*ch[1] + ch[1]*ch[2] + ch[0]*ch[3] = 14

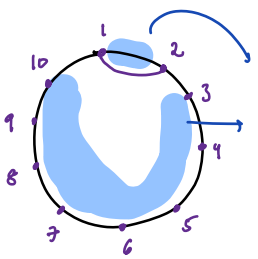


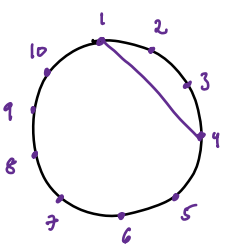


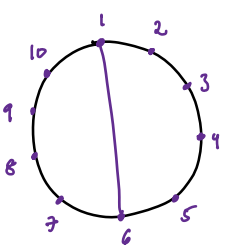


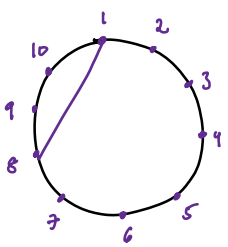


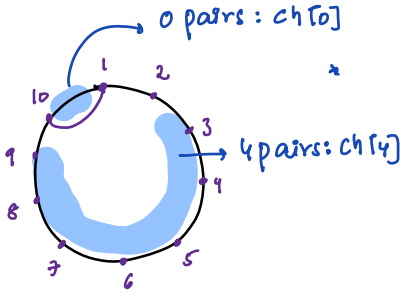
N = 5: ch[5] = ch[4]*ch[0] + ch[3]*ch[1] + ch[2]*ch[2] + ch[1]*ch[3] + ch[4]*ch[0] = 42



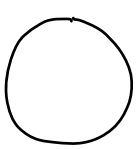








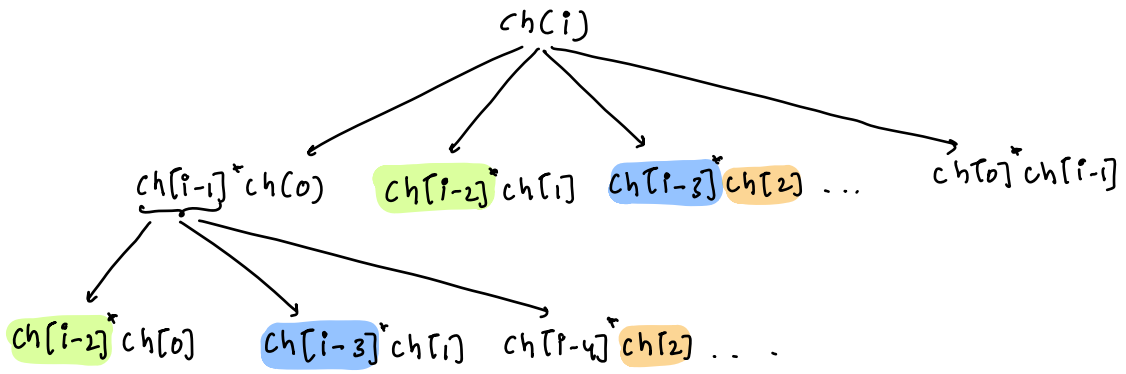
N = 0 ch[ ]



ch[3] = ch[2]*ch[0] + ch[1]*ch[1] + ch[0]*ch[2]

ch[4] = ch[3]*ch[0] + ch[2]*ch[1] + ch[1]*ch[2] + ch[0]*ch[3]

ch[5] = ch[4]*ch[0] + ch[3]*ch[1] + ch[2]*ch[2] + ch[1]*ch[3] + ch[0]*ch[4]



Dp Steps:

ch(i) = No: of ways to draw i chords using i pair of points

ch[i] = ch[i-1]*ch[0] + ch[i-2]*ch[1] + ch[i-3]*ch[2] ... ch[0]*ch[i-1]

Final ans: Given N pair of points, ways to draw N chords:
: ch[N]

#States * TC for each state

Dp Table: int dp[N+1] TC: O(N) * O(N) = O(N^2)

```
int dp[N+1] = INVALID/-1
int cords(int i){
    if(i==0){return 1;}
    if(dp[i]==-1){
        int val=0
        int j=i-1 k=0
        while(j>=0){
            val = val + cords(j)*cords(k)
            j=j-1  k=k+1
        }
        dp[i] = val
    }
    return dp[i]
}
```

→ ch[0] = 1
ch[1] = 1
ch[2] = 2
ch[3] = 5
ch[4] = 14
ch[5] = 42
⋮

Catalan Series :

$f_n = f_{n-1} * f_0 + f_{n-2} * f_1 + f_{n-3} * f_2 + \dots + f_0 * f_{n-1}$

↳ // If n^{th} number of a series can be written as above, in that case it's Catalan Series

Formula: $f_n = \frac{(2n)!}{(n)!\,(n+1)!}$

$f(3) = 5$ ✓
$f(4) = 14$ ✓

2Q) How many N pair of balanced paranthesis. // Catalyn Series.

N=1: ( ) : 1 pair

N=2: (( ))
( ) ( ) : 2 pairs

N=3: ( ) ( ) ( ) : 5 pairs
( ) (( ))
(( )) ( )
(( ) ( ))
((( )))

N=4
Start ↑ (3 pairs) 0 pairs = $f_3 * f_0$ +
Start ↑ (2 pairs) 1 pair = $f_2 * f_1$ +
Start ↑ (1 pair) 2 pairs = $f_1 * f_2$ +
Start ↑ (0 pair) 3 pairs = $f_0 * f_3$
= f_4

TODO: Question generate catalan Series
: Binary Trees
: Binary Search Trees
: Matrices
: Diagnols

Minimum Falling Path Sum II

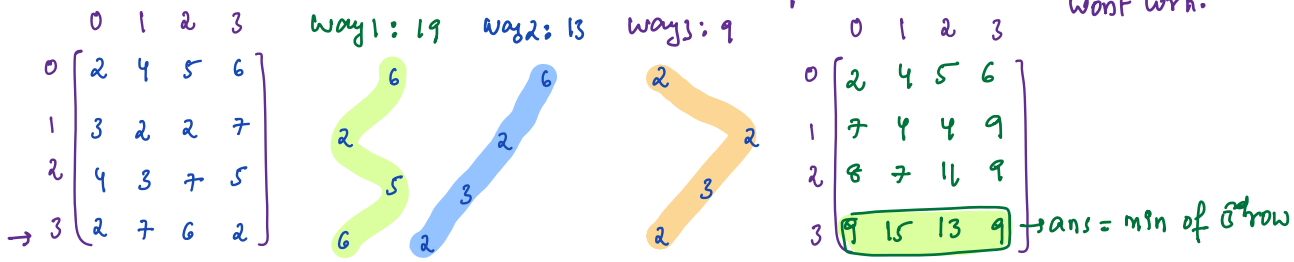
Given N*N matrix find **min sum** we can get such that

a) In Every row, we should pick **1 element**

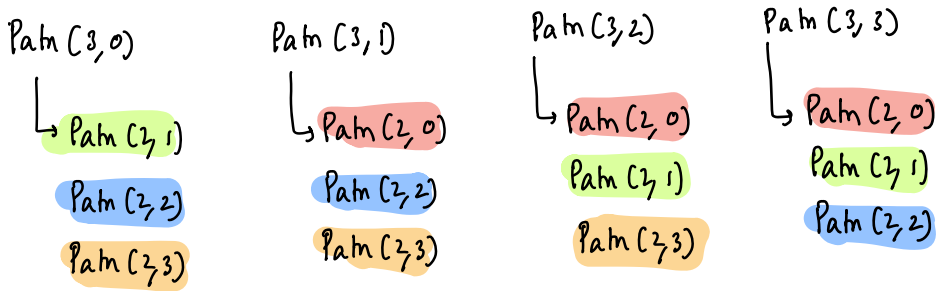
b) No 2 elements chosed in adjacent rows, should be in same column

Ex: Mat[4][4] : Pick 4 ele, 1 ele in each row.

Ideal: Greedily picking in every row wont work.



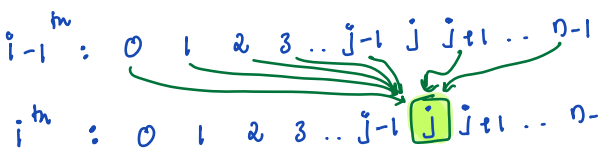
min Path Cost ending at (3, 0) such that, no 2 adjacent rows same column Picked.



Dp Steps:

dpState : dp(i,j) = min Path Cost ending at (i,j) no 2 adjacent rows same column Picked.

dpExpression: dp(i,j) =



$$dp(i,j) = \min\begin{bmatrix} n-1 \\ \forall\, dp(i-1,k) \\ k=0 \\ k\neq j \end{bmatrix} + mat[i][j]$$

Final ans: Min cost to reach last row

: min value of last row

: min{ dp[n-1][0] dp[n-1][1] dp[n-1][2] ... dp[n-1][3] }

states * TC for each state

Dp Table: dp[N][N] TC: O(N²) * O(N)

Code:

```
int dp[N][N] = INVALID / Assume all are +ve / -1

int PathCost( int i, int j, int mat[][]){
    if(i<0) { invalid row, return 0}
    if( dp[i][j] == -1){
        int val = INT_MAX
        k=0; k<N; k++){
            if(k!=j){
                val = min(val, PathCost(i-1, k, mat)
            }
        }
        dp[i][j] = val + mat[i][j]
    }
    return dp[i][j]
}

main(){
    Given mat[N][N]
    // Final ans is on last row => Make function calls for all rows in last row
    int ans = INT_MAX
    j=0; j<n; j++){
        ans = min{ ans, PathCost(N-1, j, mat[][]) }
    }
    return ans;
}
```