**Problem link**- [Triangle](https://www.codingninjas.com/codestudio/problems/triangle_1229398?source=youtube&campaign=striver_dp_videos&utm_source=youtube&utm_medium=affiliate&utm_campaign=striver_dp_videos)

\*\***1. recursive- solution: TC =** O(2^(total elements)), for every element we have 2 options to choose (bottom/ bottom right)

**//SC** = O(N) recursion stack space

**int minimumPathSum(vector<vector<int>>& triangle, int n){**

**return f(0, 0, triangle);**

**}**

**int f(int i, int j, vector<vector<int>>& triangle){**

**if(i== triangle.size()-1)**

**return triangle[i][j];**

**//bottom**

**int bottom = triangle[i][j] + f(i+1, j, triangle);**

**int bottomRight = triangle[i][j] + f(i+1, j+1, triangle);**

**//choose minimum**

**return min(bottom, bottomRight);**

**}**

\*\***2. DP- memoization solution: //tc =** O(N\*N) for all new calls

// **sc** = O(N\*N) + O(N) for recursion stack space

**int minimumPathSum(vector<vector<int>>& triangle, int n){**

**vector<vector<int>> dp(n, vector<int>(n, -1));**

**return f(0, 0, triangle, dp);**

**}**

**int f(int i, int j, vector<vector<int>>& triangle, vector<vector<int>>& dp){**

**if(i== triangle.size()-1)**

**return triangle[i][j];**

**if(dp[i][j] != -1) return dp[i][j];**

**//bottom**

**int bottom = triangle[i][j] + f(i+1, j, triangle, dp);**

**int bottomRight = triangle[i][j] + f(i+1, j+1, triangle, dp);**

**//choose minimum**

**return dp[i][j] = min(bottom, bottomRight);**

**}**

\*\***3. DP- tabulation: //TC = O(N\*N), SC = O(N\*N)**

**//DP\_tabulation (bottom- up approach)**

**int minimumPathSum(vector<vector<int>>& triangle, int n){**

**vector<vector<int>> dp(n, vector<int>(n, -1));**

**//builf base(fill last row)**

**for(int j=0; j<n; j++){**

**dp[n-1][j] = triangle[n-1][j];**

**}**

**//fill elements from row = n-2 to 0**

**for(int i=n-2; i>=0; i--){**

**for(int j=0; j<n; j++){**

**//bottom**

**int bottom = triangle[i][j] + dp[i+1][j];**

**int bottomRight = triangle[i][j] + dp[i+1][j+1];**

**//choose minimum**

**dp[i][j] = min(bottom, bottomRight);**

**}**

**}**

**//ans will be stored at dp[0][0]**

**return dp[0][0];**

**}**

\***\*4. DP- optimized space:** //TC = O(n\*n), SC = O(2n)

**//use a next[n] to keep track of [i+1][j] & [i+1][j+1] value and use curr[n] for storing current row**

**//DP\_tabulation (space- optimized)**

**int minimumPathSum(vector<vector<int>>& triangle, int n){**

**vector<int> next(n, -1);**

**//fill next with last row values of triangle**

**for(int j=0; j<n; j++){**

**next[j] = triangle[n-1][j];**

**}**

**//fill elements from row = n-2 to 0**

**for(int i=n-2; i>=0; i--){**

**vector<int> curr(n);**

**for(int j=0; j<n; j++){**

**//bottom**

**int bottom = triangle[i][j] + next[j];**

**int bottomRight = triangle[i][j] + next[j+1];**

**//choose minimum**

**curr[j] = min(bottom, bottomRight);**

**}**

**next = curr;**

**}**

**//ans will be stored in the last calculated row of size 1**

**return next[0];**

**}**