**1458. Max Dot Product of Two Subsequences: -**

**Hard Accepted: 44.4K Submissions: 77.8K Acceptance Rate: 57.0%**

Given two arrays nums1 and nums2.

Return the maximum dot product between **non-empty** subsequences of nums1 and nums2 with the same length.

A subsequence of a array is a new array which is formed from the original array by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters. (ie, [2,3,5] is a subsequence of [1,2,3,4,5] while [1,5,3] is not).

**Example 1:**

**Input:** nums1 = [2,1,-2,5], nums2 = [3,0,-6]

**Output:** 18

**Explanation:** Take subsequence [2,-2] from nums1 and subsequence [3,-6] from nums2.

Their dot product is (2\*3 + (-2)\*(-6)) = 18.

**Example 2:**

**Input:** nums1 = [3,-2], nums2 = [2,-6,7]

**Output:** 21

**Explanation:** Take subsequence [3] from nums1 and subsequence [7] from nums2.

Their dot product is (3\*7) = 21.

**Example 3:**

**Input:** nums1 = [-1,-1], nums2 = [1,1]

**Output:** -1

**Explanation:** Take subsequence [-1] from nums1 and subsequence [1] from nums2.

Their dot product is -1.

**Constraints:**

* 1 <= nums1.length, nums2.length <= 500
* -1000 <= nums1[i], nums2[i] <= 1000

**Code: -**

class Solution {

public:

    int maxDotProduct(vector<int>& nums1, vector<int>& nums2) {

        int m = nums1.size(), n = nums2.size();

        vector<vector<int>> dp(m, vector<int>(n));

        for(int i = m-1; i>=0; --i){

            for(int j=n-1; j>=0; --j){

                int finalans = INT\_MIN;

                int single = nums1[i] \* nums2[j];

                // for diagonal

                int nextrow = i + 1;

                int nextcol = j + 1;

                if(0<=nextrow and nextrow<m and 0<=nextcol and nextcol<n)

                    finalans = max(finalans, single + dp[nextrow][nextcol]);

                finalans = max(finalans, single);

                // for right exist

                if(j+1 < n)

                    finalans = max(finalans, dp[i][j+1]);

                // for down exist

                if(i+1 < m)

                    finalans = max(finalans, dp[i+1][j]);

                dp[i][j] = finalans;

            }

        }

        return dp[0][0];

    }

};

**T.C: - O(M \* N)**

**S.C: - O(M \* N)**

**M = nums1.length; N = nums2.length;**