**Problem link**: [Ninja's Training](https://www.codingninjas.com/codestudio/problems/ninja-s-training_3621003?source=youtube&campaign=striver_dp_videos&utm_source=youtube&utm_medium=affiliate&utm_campaign=striver_dp_videos)

\*\***1. recursive- solution: //tc =** O(3\*N)\*3, sc = O(N)

**int f(int day, int last, vector<vector<int>> &points){**

**int maxi = 0;**

**if(day==0){ //selct max from remaining task(0, 0)-> max(task[1], task[2])**

**for(int i=0; i<3; i++){**

**if(i != last){**

**maxi = max(maxi, points[0][i]);**

**}**

**}**

**return maxi;**

**}**

**//for rest of the days**

**for(int i=0; i<3; i++){**

**if(i != last){**

**int point = points[day][i] + f(day-1, i, points);**

**maxi = max(maxi, point);**

**}**

**}**

**return maxi;**

**}**

**int ninjaTraining(int n, vector<vector<int>> &points)**

**{**

**return f(n-1, 3, points);**

**}**

\*\***2. DP- memoization solution: //**tc **=** O(3\*N)\*3, sc = O(4\*N) + O(N)

**int f(int day, int last, vector<vector<int>> &points, vector<vector<int>> &dp){**

**int maxi = 0;**

**if(day==0){ //selct max from remaining task(0, 0)-> max(task[1], task[2])**

**for(int i=0; i<3; i++){**

**if(i != last){**

**maxi = max(maxi, points[0][i]);**

**}**

**}**

**return maxi;**

**}**

**if(dp[day][last] != -1) return dp[day][last];**

**//for rest of the days**

**for(int i=0; i<3; i++){**

**if(i != last){**

**int point = points[day][i] + f(day-1, i, points, dp);**

**maxi = max(maxi, point);**

**}**

**}**

**dp[day][last] = maxi;**

**}**

**int ninjaTraining(int n, vector<vector<int>> &points)**

**{**

**vector<vector<int>> dp(n, vector<int>(4, -1));**

**return f(n-1, 3, points, dp);**

**}**

\*\***3. DP- tabulation: //TC = O(4N)\*3, SC = O(4N)**

**//DP- tabulation: bottom- up approach**

**int ninjaTraining(int n, vector<vector<int>> &points)**

**{**

**//dp[n][4]**

**vector<vector<int>> dp(n, vector<int>(4, -1));**

**//filling day[0][i] values**

**dp[0][0] = max(points[0][1], points[0][2]);**

**dp[0][1] = max(points[0][0], points[0][2]);**

**dp[0][2] = max(points[0][0], points[0][1]);**

**dp[0][3] = max(points[0][0], max(points[0][1], points[0][2]));**

**//day = 1 to n-1**

**for (int day = 1; day < n; day++) {**

**for (int last = 0; last < 4; last++) {**

**for (int task = 0; task <= 2; task++) {**

**if (task != last) {**

**dp[day][last] = max(dp[day][last], points[day][task] + dp[day - 1][task]);**

**}**

**}**

**}**

**}**

**return dp[n-1][3];**

**}**

\***\*4. DP- optimized space:** //TC = O(4N)\*3, SC = O(4)

**//DP- tabulation: bottom- up approach**

**int ninjaTraining(int n, vector<vector<int>> &points)**

**{**

**//dp[n][4]**

**vector<int> prev(4, -1);**

**//store previous 4 values that's all we need to compute current day points**

**prev[0] = max(points[0][1], points[0][2]);**

**prev[1] = max(points[0][0], points[0][2]);**

**prev[2] = max(points[0][0], points[0][1]);**

**prev[3] = max(points[0][0], max(points[0][1], points[0][2]));**

**//day = 1 to n-1**

**vector<int> dp(4, -1); //stores current day points**

**for (int day = 1; day < n; day++) {**

**for (int last = 0; last < 4; last++) {**

**for (int task = 0; task <= 2; task++) {**

**if (task != last) {**

**dp[last] = max(dp[last], points[day][task] + prev[task]);**

**}**

**}**

**}**

**prev = dp; //use current values in prev after computing current day points**

**}**

**return dp[3]; //value stored at this index is our ans**

**}**