|  |  |
| --- | --- |
| **Largest submatrix C++** | |
| #include <iostream>  #include <algorithm>  using namespace std;  // Define the maximum size for the grid (you can adjust this as needed)  const int MAX\_ROWS = 100;  const int MAX\_COLS = 100;  // Function to find the largest square submatrix  int largestSquareSubmatrix(const int arr[MAX\_ROWS][MAX\_COLS], int rows, int cols) {      int dp[MAX\_ROWS][MAX\_COLS] = {0}; // DP table      int largestSide = 0;      // Fill the dp array      for (int i = rows - 1; i >= 0; i--) {          for (int j = cols - 1; j >= 0; j--) {              if (i == rows - 1 || j == cols - 1) {                  dp[i][j] = arr[i][j];              } else {                  if (arr[i][j] == 0) {                      dp[i][j] = 0;                  } else {                      int minSide = min(dp[i][j + 1], min(dp[i + 1][j], dp[i + 1][j + 1]));                      dp[i][j] = minSide + 1;                  }              }              if (dp[i][j] > largestSide) {                  largestSide = dp[i][j];              }          }      }      return largestSide; // Return the side length of the largest square submatrix  }  int main() {      // Define the array and its dimensions      const int arr[MAX\_ROWS][MAX\_COLS] = {          {0, 1, 0, 1, 0, 1},          {1, 0, 1, 0, 1, 0},          {0, 1, 1, 1, 1, 0},          {0, 0, 1, 1, 1, 0},          {1, 1, 1, 1, 1, 1}      };      int rows = 5;      int cols = 6;      cout << largestSquareSubmatrix(arr, rows, cols) << endl;      return 0;  } | ****Step 2.1: Given Matrix (****arr****)**** 0 1 0 1 0 1  1 0 1 0 1 0  0 1 1 1 1 0  0 0 1 1 1 0  1 1 1 1 1 1 ****Step 2.2: DP Table Construction********Step 2.2.1: Initialize**** dp[][] ****(Same as**** arr[][] ****for last row & last column)**** 0 1 0 1 0 1  1 0 1 0 1 0  0 1 1 1 1 0  0 0 1 1 1 0  1 1 1 1 1 1 <- (Same as `arr` because it's the last row) ****Step 2.2.2: Fill the**** dp[][] ****Table Bottom-Up****  | **i, j** | **arr[i][j]** | **Formula Applied** | **dp[i][j]** | | --- | --- | --- | --- | | (3,4) | 1 | 1 + min(1, 1, 1) | **2** | | (3,3) | 1 | 1 + min(1, 1, 2) | **2** | | (3,2) | 1 | 1 + min(1, 2, 1) | **2** | | (2,4) | 1 | 1 + min(2, 1, 1) | **2** | | (2,3) | 1 | 1 + min(2, 2, 1) | **2** | | (2,2) | 1 | 1 + min(2, 2, 2) | **3** (Largest Square Found) |   **Final dp[][] Matrix**  0 1 0 1 0 1  1 0 1 0 1 0  0 1 2 2 2 0  0 0 2 2 2 0  1 1 1 1 1 1 **Step 3: Final Answer** Largest Square Side = 3 |
| Output:- 3 | |