|  |  |
| --- | --- |
| **No of Distinct Island in C++** | |
| #include <iostream>  #include <vector>  #include <unordered\_set>  using namespace std;  // Function prototypes  void dfs(vector<vector<int>>& arr, int row, int col, string& psf);  int numDistinctIslands(vector<vector<int>>& arr);  // Depth-first search to mark all connected land cells of an island  void dfs(vector<vector<int>>& arr, int row, int col, string& psf) {  arr[row][col] = 0; // Marking current cell as visited  int n = arr.size();  int m = arr[0].size();    // Directions: up, right, down, left  vector<pair<int, int>> dirs = {{-1, 0}, {0, 1}, {1, 0}, {0, -1}};  string dirStr = "urdl"; // Corresponding directions characters  for (int i = 0; i < 4; ++i) {  int newRow = row + dirs[i].first;  int newCol = col + dirs[i].second;  if (newRow >= 0 && newRow < n && newCol >= 0 && newCol < m && arr[newRow][newCol] == 1) {  psf += dirStr[i]; // Append direction character to path string  dfs(arr, newRow, newCol, psf);  }  }  psf += "a"; // Append anchor to indicate end of island path  }  // Function to find number of distinct islands  int numDistinctIslands(vector<vector<int>>& arr) {  int n = arr.size();  if (n == 0) return 0;  int m = arr[0].size();  unordered\_set<string> islands; // Set to store distinct island paths  for (int i = 0; i < n; ++i) {  for (int j = 0; j < m; ++j) {  if (arr[i][j] == 1) {  string psf = "x"; // Starting character to represent new island  dfs(arr, i, j, psf);  islands.insert(psf); // Insert island path into set  }  }  }  return islands.size(); // Return the number of distinct islands  }  int main() {  // Hardcoded input  vector<vector<int>> arr = {  {1, 0, 0},  {0, 1, 0},  {1, 1, 1}  };  // Calculating number of distinct islands  cout << numDistinctIslands(arr) << endl;  return 0;  } | **Key Concepts:**   * An **island** is a group of 1s connected **horizontally or vertically**. * Each island is converted into a **path string** (psf) using DFS with directional encoding (u, r, d, l, and a for backtracking). * The unordered\_set stores these path strings to count **unique island shapes**.   **📥 Input Grid:**  1 0 0  0 1 0  1 1 1  **Key for DFS path string (psf):**   * x → Start of island * u → Up * r → Right * d → Down * l → Left * a → Backtrack (anchor)   **📊 Dry Run Table:**   | **Island #** | **Starting Cell** | **DFS Path (psf)** | **Shape Description** | **Is Unique?** | | --- | --- | --- | --- | --- | | 1 | (0, 0) | xa | Single cell | ✅ Yes | | 2 | (1, 1) | xa | Single cell | ❌ No | | 3 | (2, 0) | xrraa | Horizontal chain (L-shape) | ✅ Yes |   **🧮 Final Set of Unique Island Shapes:**   | **Shape Path** | | --- | | xa | | xrraa |   **✅ Output:**  2 |
| Output:- 2 | |