# No of provinces in C++

```
#include <bits/stdc++.h>
using namespace std;

class Solution {
  private:
    // dfs traversal function
    void dfs(int node, vector<int> adjLs[], int vis[]) {
        // mark the more as visited
        vis[node] = 1;
        for(auto it: adjLs[node]) {
            if(!vis[it]) {
                dfs(it, adjLs, vis);
            }
        }
    }
  public:
    int numProvinces(vector<vector<int>> adj, int V) {
        vector<int> adjLs[V];

        // to change adjacency matrix to list
        for(int i = 0;i<V;i++) {
            for(int j = 0;j<V;j++) {
                // self nodes are not considered
                if(adj[i][j] == 1 && i != j) {
                    adjLs[i].push_back(j);
                    adjLs[j].push_back(i);
                }
            }
        }
        int vis[V] = {0};
        int cnt = 0;
        for(int i = 0;i<V;i++) {
            // if the node is not visited
            if(!vis[i]) {
                // counter to count the number of provinces
                cnt++;
               dfs(i, adjLs, vis);
            }
        }
        return cnt;

    }
};

int main() {

    vector<vector<int>> adj
    {
        {1, 0, 1},
        {0, 1, 0},
        {1, 0, 1}
    };

    Solution ob;
    cout << ob.numProvinces(adj,3) << endl;

    return 0;
}
```

**Dry Run:**

**Input:**

```
vector<vector<int>> adj = {
    {1, 0, 1},
    {0, 1, 0},
    {1, 0, 1}
};
```

**Adjacency Matrix to List Conversion:**

- adj[0] has a 1 at indices 0 and 2, so node 0 is connected to node 2.
- adj[1] has a 1 at index 1, so node 1 is connected to itself.
- adj[2] has a 1 at indices 0 and 2, so node 2 is connected to node 0.

**Adjacency List:**

```
adjLs = {
    {2},  // Node 0 is connected to 2
    {0},  // Node 1 is connected to 0
    {0}   // Node 2 is connected to 0
}
```

**DFS Execution:**

1. Start DFS from node 0. Mark node 0 as visited and visit node 2.
2. In DFS traversal, we will also mark node 2 as visited.
3. Start DFS from node 1. Since it is unvisited, we increment the province counter. Mark node 1 as visited.
4. Finally, return the total count of provinces (2).

**Output:-**

```
2
```