

```

1  #include <iostream>
2  #include <vector>
3
4  using namespace std;
5
6  class GraphMatrix
7  {
8      int numVertices;
9      vector<vector<int>> matrix;
10
11 public:
12     GraphMatrix(int vertices)
13     {
14         numVertices = vertices;
15         matrix.resize(vertices, vector<int>(vertices, 0));
16     }
17
18     void addEdge(int src, int dest, bool directed = false)
19     {
20         matrix[src][dest] = 1;
21         if (!directed)
22         {
23             matrix[dest][src] = 1;
24         }
25     }
26
27     void printGraph()
28     {
29         for (int i = 0; i < numVertices; i++)
30         {
31             for (int j = 0; j < numVertices; j++)
32             {
33                 cout << matrix[i][j] << " ";
34             }
35             cout << endl;
36         }
37     }
38 };
39 #include <iostream>
40 #include <vector>
41 #include <list>
42 #include <queue>
43 #include <stack>
44
45 using namespace std;
46
47 class Graph
48 {
49     int numVertices;
50     vector<list<int>> adjList;
51
52 public:
53     Graph(int vertices)
54     {
55         numVertices = vertices;
56         adjList.resize(vertices);
57     }
58
59     void addEdge(int src, int dest, bool directed = false)
60     {
61         adjList[src].push_back(dest);
62         if (!directed)
63         {
64             adjList[dest].push_back(src);
65         }
66     }
67
68     void printGraph()
69     {
70         for (int i = 0; i < numVertices; i++)
71         {
72             cout << i << ":";
73             for (int neighbor : adjList[i])
74             {
75                 cout << " -> " << neighbor;
76             }
77             cout << endl;
78         }
79     }
80
81     void DFS(int startVertex)
82     {
83         vector<bool> visited(numVertices, false);
84         DFSUtil(startVertex, visited);
85         cout << endl;
86     }
87
88     void DFSUtil(int v, vector<bool> &visited)
89     {
90         visited[v] = true;
91         cout << v << " ";
92
93         for (int neighbor : adjList[v])
94         {
95             if (!visited[neighbor])
96             {
97                 DFSUtil(neighbor, visited);
98             }
99         }
100     }
101
102     void BFS(int startVertex)
103     {
104         vector<bool> visited(numVertices, false);
105         queue<int> q;
106
107         visited[startVertex] = true;
108         q.push(startVertex);
109
110         while (!q.empty())
111         {
112             int current = q.front();
113             q.pop();
114             cout << current << " ";
115
116             for (int neighbor : adjList[current])
117             {
118                 if (!visited[neighbor])
119                 {
120                     visited[neighbor] = true;
121                     q.push(neighbor);
122                 }
123             }
124             cout << endl;
125         }
126     }
127 };
128 int main()
129 {
130     Graph g(5);
131
132     g.addEdge(0, 1);
133     g.addEdge(0, 4);
134     g.addEdge(1, 2);
135     g.addEdge(1, 3);
136     g.addEdge(1, 4);
137     g.addEdge(2, 3);
138     g.addEdge(3, 4);
139
140     g.printGraph();
141
142     g.DFS(0);
143     g.BFS(0);
144
145     return 0;
146 }

```