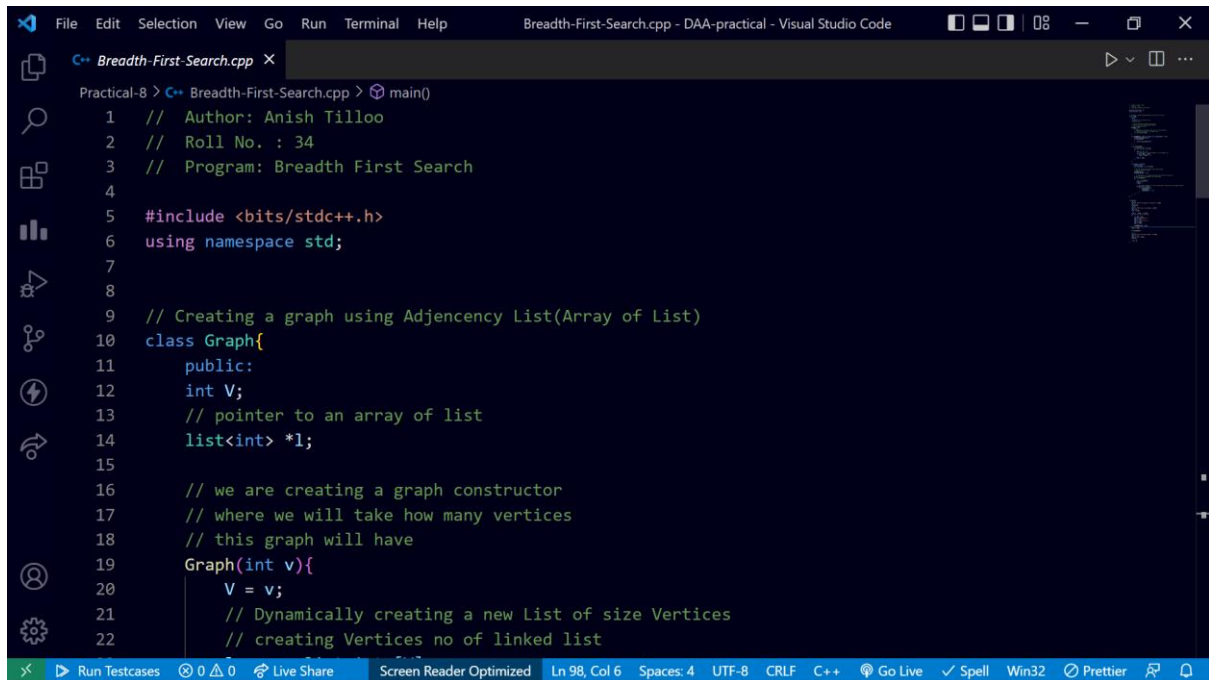
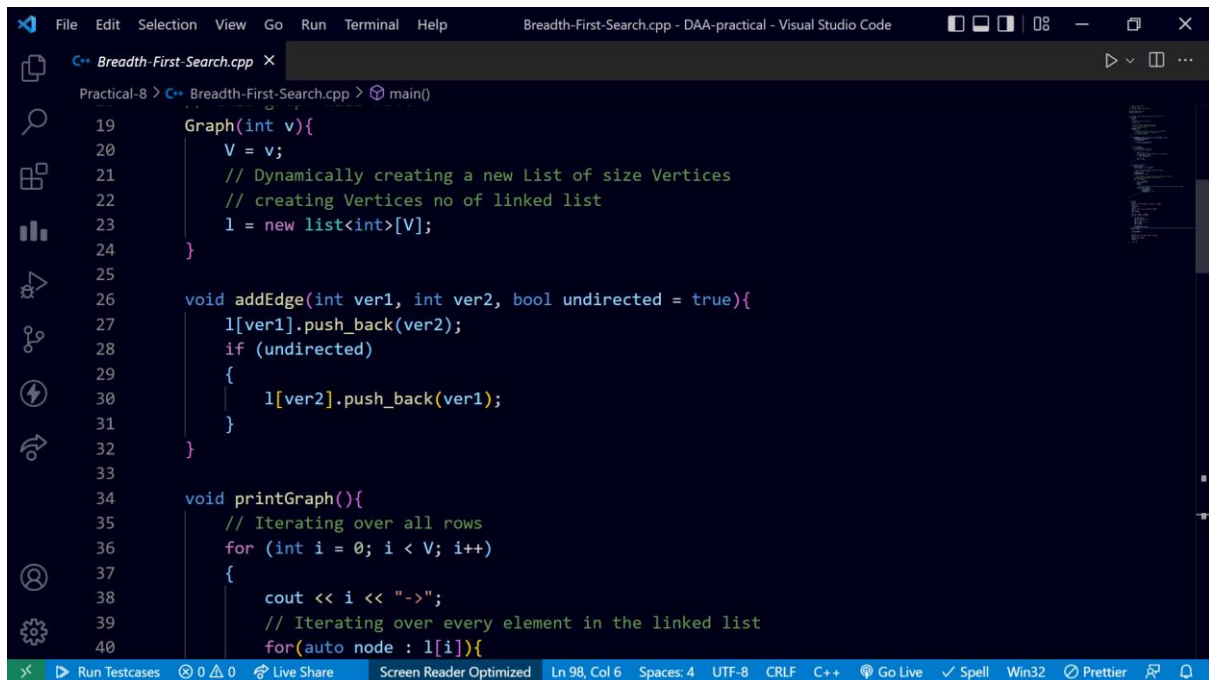


BFS



```
Practical-8 > C++ Breadth-First-Search.cpp > main()
1 // Author: Anish Tilloo
2 // Roll No. : 34
3 // Program: Breadth First Search
4
5 #include <bits/stdc++.h>
6 using namespace std;
7
8
9 // Creating a graph using Adjacency List(Array of List)
10 class Graph{
11 public:
12     int V;
13     // pointer to an array of list
14     list<int> *l;
15
16     // we are creating a graph constructor
17     // where we will take how many vertices
18     // this graph will have
19     Graph(int v){
20         V = v;
21         // Dynamically creating a new List of size Vertices
22         // creating Vertices no of linked list
```



```
19     Graph(int v){
20         V = v;
21         // Dynamically creating a new List of size Vertices
22         // creating Vertices no of linked list
23         l = new list<int>[V];
24     }
25
26     void addEdge(int ver1, int ver2, bool undirected = true){
27         l[ver1].push_back(ver2);
28         if (undirected)
29         {
30             l[ver2].push_back(ver1);
31         }
32     }
33
34     void printGraph(){
35         // Iterating over all rows
36         for (int i = 0; i < V; i++)
37         {
38             cout << i << "->";
39             // Iterating over every element in the linked list
40             for(auto node : l[i]){
```

```
File Edit Selection View Go Run Terminal Help Breadth-First-Search.cpp - DAA-practical - Visual Studio Code
Breadth-First-Search.cpp x
Practical-8 > C++ Breadth-First-Search.cpp > main()
33
34 void printGraph(){
35     // Iterating over all rows
36     for (int i = 0; i < V; i++)
37     {
38         cout << i << "->";
39         // Iterating over every element in the linked list
40         for(auto node : l[i]){
41             cout << node << ",";
42         }
43         cout << endl;
44     }
45 }
46
47
48 void bfs(int source){
49     queue<int> q;
50     bool *visited = new bool[V]{0};
51
52     // We are pushing the source node inside queue
53     // and it is also visited
54     q.push(source);
```

```
File Edit Selection View Go Run Terminal Help Breadth-First-Search.cpp - DAA-practical - Visual Studio Code
Breadth-First-Search.cpp x
Practical-8 > C++ Breadth-First-Search.cpp > main()
48 void bfs(int source){
49     queue<int> q;
50     bool *visited = new bool[V]{0};
51
52     // We are pushing the source node inside queue
53     // and it is also visited
54     q.push(source);
55     visited[source] = true;
56
57     // if the queue is not empty then pop out the node and print it
58     // and insert its neighbours into the queue
59     while (!q.empty())
60     {
61         int f = q.front();
62         cout << f << " ";
63         q.pop();
64
65         // push the neighbours of the current node if they are not already visited
66         for(auto nbr : l[f]){
67             if (!visited[nbr]){
68                 q.push(nbr);
69                 visited[nbr] = true;
```

```
File Edit Selection View Go Run Terminal Help Breadth-First-Search.cpp - DAA-practical - Visual Studio Code
Breadth-First-Search.cpp x
Practical-8 > C++ Breadth-First-Search.cpp > main()
68         q.push(nbr);
69         visited[nbr] = true;
70     }
71 }
72 }
73 }
74 };
75
76
77 int main(){
78     int n;
79     cout << "Enter the number of vertices" << endl;
80     cin >> n;
81     Graph g(n);
82     int m;
83     cout << "Enter the no of edges" << endl;
84     cin >> m;
85     cout << endl;
86
87     cout << "Edges " << endl;
88     for (int i = 0; i < m; i++)
89     {
```

```
File Edit Selection View Go Run Terminal Help Breadth-First-Search.cpp - DAA-practical - Visual Studio Code
Breadth-First-Search.cpp x
Practical-8 > C++ Breadth-First-Search.cpp > main()
87     cout << "Edges " << endl;
88     for (int i = 0; i < m; i++)
89     {
90         int vex1, vex2;
91         cout << "Vertex One: ";
92         cin >> vex1;
93         cout << "Vertex Two: ";
94         cin >> vex2;
95         cout << endl;
96
97         g.addEdge(vex1, vex2);
98     }
99     cout << endl;
100
101     g.printGraph();
102
103     int s;
104     cout << "Enter the source Node " << endl;
105     cin >> s;
106     cout << "BFS" << endl;
107     g.bfs(s);
108 }
```

```
File Edit Selection View Go Run Terminal Help Breadth-First-Search.cpp - DAA-practical - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Enter the number of vertices
7
Enter the no of edges
6
Edges
Vertex One: 0
vertex Two: 1
Vertex One: 1
vertex Two: 2
Vertex One: 3
vertex Two: 5
Vertex One: 5
vertex Two: 6
Vertex One: 4
vertex Two: 5
Vertex One: 0
vertex Two: 4
0->1,4,
1->0,2,
2->1,
3->5,
Run Testcases 0 0 Live Share Screen Reader Optimized Ln 98, Col 6 Spaces: 4 UTF-8 CRLF C++ Go Live Spell Win32 Prettier
```

```
File Edit Selection View Go Run Terminal Help Breadth-First-Search.cpp - DAA-practical - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Vertex One: 4
vertex Two: 5
Vertex One: 0
vertex Two: 4
0->1,4,
1->0,2,
2->1,
3->5,
4->5,0,
5->3,6,4,
6->5,
Enter the source Node
0
BFS
0 1 4 2 5 3 6
PS E:\College-Work\Sixth-Semester\DAA-practical\Practical-8> |
Run Testcases 0 0 Live Share Screen Reader Optimized Ln 98, Col 6 Spaces: 4 UTF-8 CRLF C++ Go Live Spell Win32 Prettier
```

DFS

```
File Edit Selection View Go Run Terminal Help Depth-First-Search-Algo.cpp - DAA-practical - Visual Studio Code
Depth-First-Search-Algo.cpp
Practical-8 > C++ Depth-First-Search-Algo.cpp > main()
1 // Author: Anish Tilloo
2 // Roll No. : 34
3 // Program: Depth First Search
4
5 #include <bits/stdc++.h>
6 using namespace std;
7
8
9 // Creating a graph using Adjacency List(Array of List)
10 class Graph{
11 public:
12     int V;
13     // pointer to an array of list
14     list<int> *l;
15
16     // we are creating a graph constructor
17     // where we will take how many vertices
18     // this graph will have
19     Graph(int v){
20         V = v;
21         // Dynamically creating a new List of size Vertices
22         // creating Vertices no of linked list
```

```
File Edit Selection View Go Run Terminal Help Depth-First-Search-Algo.cpp - DAA-practical - Visual Studio Code
Depth-First-Search-Algo.cpp
Practical-8 > C++ Depth-First-Search-Algo.cpp > main()
19     Graph(int v){
20         V = v;
21         // Dynamically creating a new List of size Vertices
22         // creating Vertices no of linked list
23         l = new list<int>[V];
24     }
25
26     void addEdge(int ver1, int ver2, bool undirected = true){
27         l[ver1].push_back(ver2);
28         if (undirected)
29         {
30             l[ver2].push_back(ver1);
31         }
32     }
33
34     void printGraph(){
35         // Iterating over all rows
36         for (int i = 0; i < V; i++)
37         {
38             cout << i << "->";
39             // Iterating over every element in the linked list
40             for(auto node : l[i]){
```

```
File Edit Selection View Go Run Terminal Help Depth-First-Search-Algo.cpp - DAA-practical - Visual Studio Code
Depth-First-Search-Algo.cpp X
Practical-8 > C++ Depth-First-Search-Algo.cpp > main()
34 void printGraph(){
35     // Iterating over all rows
36     for (int i = 0; i < V; i++)
37     {
38         cout << i << "->";
39         // Iterating over every element in the linked list
40         for(auto node : l[i]){
41             cout << node << ",";
42         }
43         cout << endl;
44     }
45 }
46
47
48 void dfsHelper(int node, bool *visited){
49     visited[node] = true;
50     cout << node << ",";
51     // make a dfs call on all its unvisited neighbours
52     for(auto nbr : l[node]){
53         if (!visited[nbr])
54         {
55             dfsHelper(nbr, visited);
56         }
57     }
58     return;
59 }
60
61 void dfs(int source){
62     bool *visited = new bool [V]{0};
63     dfsHelper(source, visited);
64 }
65
66 };
67
68 int main()
69 {
70     // ...
71 }
```

```
File Edit Selection View Go Run Terminal Help Depth-First-Search-Algo.cpp - DAA-practical - Visual Studio Code
Depth-First-Search-Algo.cpp X
Practical-8 > C++ Depth-First-Search-Algo.cpp > main()
47
48 void dfsHelper(int node, bool *visited){
49     visited[node] = true;
50     cout << node << ",";
51     // make a dfs call on all its unvisited neighbours
52     for(auto nbr : l[node]){
53         if (!visited[nbr])
54         {
55             dfsHelper(nbr, visited);
56         }
57     }
58     return;
59 }
60
61 void dfs(int source){
62     bool *visited = new bool [V]{0};
63     dfsHelper(source, visited);
64 }
65
66 };
67
68 int main()
69 {
70     // ...
71 }
```

```
File Edit Selection View Go Run Terminal Help Depth-First-Search-Algo.cpp - DAA-practical - Visual Studio Code
Depth-First-Search-Algo.cpp X
Practical-8 > C++ Depth-First-Search-Algo.cpp > main()
69 int main(){
70     int n;
71     cout << "Enter the number of vertices" << endl;
72     cin >> n;
73     Graph g(n);
74     int m;
75     cout << "Enter the no of edges" << endl;
76     cin >> m;
77     cout << endl;
78
79     cout << "Edges " << endl;
80     for (int i = 0; i < m; i++)
81     {
82         int vex1, vex2;
83         cout << "Vertex One: ";
84         cin >> vex1;
85         cout << "vertex Two: ";
86         cin >> vex2;
87         cout << endl;
88
89         g.addEdge(vex1, vex2);
90     }
```

```
File Edit Selection View Go Run Terminal Help Depth-First-Search-Algo.cpp - DAA-practical - Visual Studio Code
Depth-First-Search-Algo.cpp X
Practical-8 > C++ Depth-First-Search-Algo.cpp > main()
84         cin >> vex1;
85         cout << "vertex Two: ";
86         cin >> vex2;
87         cout << endl;
88
89         g.addEdge(vex1, vex2);
90     }
91     cout << endl;
92
93     g.printGraph();
94
95     int s;
96     cout << "Enter the source Node " << endl;
97     cin >> s;
98     cout << "DFS" << endl;
99     g.dfs(s);
100
101     return 0;
102 }
```

```
File Edit Selection View Go Run Terminal Help Depth-First-Search-Algo.cpp - DAA-practical - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
($?) { g++ Depth-First-Search-Algo.cpp -o Depth-First-Search-Algo } ; if ($?) { .\Depth-First-Search-Algo }
Enter the number of vertices
7
Enter the no of edges
7

Edges
Vertex One: 0
vertex Two: 1

Vertex One: 1
vertex Two: 2

Vertex One: 3
vertex Two: 5

Vertex One: 5
vertex Two: 6

Vertex One: 4
vertex Two: 5

Vertex One: 0
vertex Two: 4

Vertex One: 3
vertex Two: 4
```

```
File Edit Selection View Go Run ... Depth-First-Search-Algo.cpp - DAA-practical - Visua...
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Vertex One: 4
vertex Two: 5

Vertex One: 0
vertex Two: 4

Vertex One: 3
vertex Two: 4

0->1,4,
1->0,2,
2->1,
3->5,4,
4->5,0,3,
5->3,6,4,
6->5,
Enter the source Node
1
DFS
1,0,4,5,3,6,2,
PS E:\College-Work\Sixth-Semester\DAA-practical\Practical-8> |
```