Breadth First Search:

BFS explores all the vertices of a graph in breadth-first manner, i.e., it visits all the vertices at distance **k** from the source before visiting the vertices at distance **k+1**.

Algorithm:

1. Enqueue the source vertex and mark it as visited.
2. While the queue is not empty, dequeue a vertex, visit it, and enqueue its adjacent unvisited vertices.

Code:

```cpp
#include <iostream>
#include <queue>
#include <vector>
#include <algorithm>
using namespace std;

class Graph {
  int V;
  vector<int> *adj;

public:
  Graph(int V) {
    this->V = V;
    adj = new vector<int>[V];
  }

  void addEdge(int v, int w) {
    adj[v].push_back(w);
  }

  void BFS(int s) {
    bool *visited = new bool[V];
    fill(visited, visited + V, false);
    queue<int> q;
    visited[s] = true;
    q.push(s);

    while (!q.empty()) {
      int v = q.front();
      cout << v << " ";
      q.pop();
      for (int i : adj[v]) {
```

```
            if (!visited[i]) {
                visited[i] = true;
                q.push(i);
            }
        }
    }
  }
};

int main() {
    Graph g(7);
    g.addEdge(0, 1);
    g.addEdge(0, 2);
    g.addEdge(1, 3);
    g.addEdge(1, 4);
    g.addEdge(2, 5);
    g.addEdge(2, 6);
    cout << "BFS starting from vertex 0: ";
    g.BFS(0);
    return 0;
}
```

Output:

Space complexity: _____

Justification:_____
_____

Time complexity:

Best case time complexity: _____

Justification:_____
_____

Worst case time complexity: _____

Justification:_____