





**Conclusion:**

#include <iostream>

#include <list>

#include <queue>

using namespace std;

class Graph {

int V;

list<int>\* adj;

public:

Graph(int V);

void addEdge(int v, int w);

void DFS(int v, bool visited[]);

void BFS(int s);

};

Graph::Graph(int V) {

this->V = V;

adj = new list<int>[V];

}

void Graph::addEdge(int v, int w) {

adj[v].push\_back(w);

adj[w].push\_back(v); // for undirected graph

}

void Graph::DFS(int v, bool visited[]) {

visited[v] = true;

cout << v << " ";

for (int u : adj[v])

if (!visited[u])

DFS(u, visited);

}

void Graph::BFS(int s) {

bool\* visited = new bool[V]{false};

queue<int> q;

visited[s] = true;

q.push(s);

while (!q.empty()) {

int v = q.front();

q.pop();

cout << v << " ";

for (int u : adj[v])

if (!visited[u]) {

visited[u] = true;

q.push(u);

}

}

}

int main() {

Graph g(4);

g.addEdge(0, 1);

g.addEdge(0, 2);

g.addEdge(1, 2);

g.addEdge(2, 3);

cout << "DFS starting from vertex 2:\n";

bool visited[4] = {false};

g.DFS(2, visited);

cout << "\nBFS starting from vertex 2:\n";

g.BFS(2);

return 0;

}

Thus, we have implemented the A\* algorithm to find the most cost-effective path in a graph. it combines the benefits of both greedy and dijkstra’s algorithms by considering both the actual cost and the estimated cost to reach the goal.

Thus, we have implemented the depth-first search (dfs) algorithm using an adjacency list in c++. it efficiently explores graphs, marking nodes to avoid revisiting them.

Thus, we have implemented the greedy search algorithm for kruskal's minimum spanning tree. it efficiently finds the optimal spanning tree by selecting edges with the least weight, ensuring no cycles are formed.

Thus, we have implemented the n-queen problem using the branch and bound technique. This method efficiently explores possible solutions by eliminating invalid configurations, ensuring an optimal arrangement of queens on the chessboard.

Thus, we have developed an elementary Chatbot for customer interaction applications. it effectively handles basic queries and responses, providing a foundation for more advanced conversational systems.

Thus, we have developed a student information management system in C++ that efficiently stores and manages student data.