ARTIFICIAL INTELLIGENCE
ASSIGNMENT 4

U20CS005
BANSI MARAKANA

Implement Traveling Salesman problem using below algorithms in prolog. Compare the complexity of both algorithms. Which algorithm is best suited for implementing the Traveling Salesman problem and why?

BFS Code:

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

const int N = 20;
int n, graph[N][N], visited[1 << N][N], ans = INT_MAX;
struct node
{
    int removed, current, cost;
};

void bfs()
{
    queue<node> q;
    q.push({1, 0, 0});
    visited[1][0] = 1;
    while (!q.empty())
    {
        node nd = q.front();
        q.pop();
        if (nd.removed == (1 << n) - 1 && graph[nd.current][0])
            ans = min(ans, nd.cost + graph[nd.current][0]);
        for (int i = 0; i < n; i++)
        {
            if (!visited[nd.removed | (1 << i)][i] &&
graph[nd.current][i])
            {
                visited[nd.removed | (1 << i)][i] = 1;
                q.push({nd.removed | (1 << i), i, nd.cost +
graph[nd.current][i]});
            }
        }
    }
```

```
}

int main()
{
    cout << "\n*********Travelling Salesman Problem Using Breadth First
Serach*********\n";
    cout << "Enter the number of nodes: ";
    cin >> n;
    cout << "Enter adjacency matrix: \n";
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            cin >> graph[i][j];
    bfs();
    if (ans == INT_MAX)
        cout << "Cycle doesn't exist!!\n";
    else
        cout << "Minimum cost is: " << ans << "\n";
    return 0;
}
```

```
PS D:\BANSI MARAKANA\AI> g++ a4q1.cpp
PS D:\BANSI MARAKANA\AI> ./a

*********Travelling Salesman Problem Using Breadth First Serach*********
Enter the number of nodes: 7
Enter adjacency matrix:
0 12 10 0 0 0 0
12 0 8 12 0 0 0
10 8 0 11 3 0 9
0 12 11 0 11 6 0
0 0 3 11 0 6 7
0 0 0 6 6 0 9
0 0 9 0 7 9 0
Minimum cost is: 59
PS D:\BANSI MARAKANA\AI>
```

Time Complexity: $O(N^2)$
Space Complexity: $O(2^N)$

DFS Code:

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

vector<int> path;
vector<int> min_path;
int min_tsp = INT_MAX - 1;

void dfs(vector<vector<int>> &adj, int node, int cnt, vector<bool>
visited, int cost)
{
    if (cnt == adj.size() && adj[node][0] > 0)
    {
        if (min_tsp > cost + adj[node][0])
        {
            min_tsp = cost + adj[node][0];
            min_path = path;
        }
        return;
    }
    for (int i = 0; i < adj.size(); i++)
    {
        if (adj[node][i] <= 0 || visited[i])
            continue;
        visited[i] = true;
        path.push_back(i);
        dfs(adj, i, cnt + 1, visited, cost + adj[node][i]);
        path.pop_back();
        visited[i] = false;
    }
}

int main()
{
    cout << "\n*********Travelling Salesman Problem Using Depth First
Serach*********\n";
    int n;
    cout << "Enter the number of nodes: ";
    cin >> n;
    vector<vector<int>> adj(n, vector<int>(n, -1));
```

```
    cout << "\nEnter the adjacency matrix\n";
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            cin >> adj[i][j];
    vector<bool> visited(n, false);
    visited[0] = true;
    path.push_back(0);
    dfs(adj, 0, 1, visited, 0);
    cout << "Minimum cost path is: ";
    for (int i = 0; i < min_path.size(); i++)
        cout << min_path[i] << " -> ";
    cout << 0 << endl;
    cout << "Minimum Cost is: " << min_tsp << endl;
}
```

```
PS D:\BANSI MARAKANA\AI> g++ a4q2.cpp
PS D:\BANSI MARAKANA\AI> ./a

*********Travelling Salesman Problem Using Depth First Serach*********
Enter the number of nodes: 7

Enter the adjacency matrix
0 12 10 0 0 0 0
12 0 8 12 0 0 0
10 8 0 11 3 0 9
0 12 11 0 11 6 0
0 0 3 11 0 6 7
0 0 0 6 6 0 9
0 0 9 0 7 9 0
Minimum cost path is: 0 -> 1 -> 3 -> 5 -> 6 -> 4 -> 2 -> 0
Minimum Cost is: 59
```

Time Complexity: O(N!)
Space Complexity: $O(N^2)$

As shown above both algorithms are complete and we can conclude from time complexity of both the algorithms that BFS is faster than DFS,so BFS is a better algorithm to solve this problem.