Bus Routes in C++

```cpp
#include <iostream>
#include <vector>
#include <unordered_map>
#include <queue>
#include <unordered_set>

using namespace std;

int numBusesToDestination(vector<vector<int>>& routes, int S, int T) {
    int n = routes.size();
    unordered_map<int, vector<int>> map;

    // Building a map of bus stops to their respective bus routes
    for (int i = 0; i < n; ++i) {
        for (int j = 0; j < routes[i].size(); ++j) {
            int busStopNo = routes[i][j];
            map[busStopNo].push_back(i);
        }
    }

    queue<int> q;
    unordered_set<int> busStopVisited;
    unordered_set<int> busVisited;
    int level = 0;
    q.push(S);
    busStopVisited.insert(S);

    // Performing BFS to find the minimum number of buses
    while (!q.empty()) {
        int size = q.size();
        while (size-- > 0) {
            int currentStop = q.front();
            q.pop();
            if (currentStop == T) {
                return level;
            }

            if (map.find(currentStop) != map.end()) {
                vector<int>& buses = map[currentStop];
                for (int bus : buses) {
                    if (busVisited.count(bus) > 0) {
                        continue;
                    }

                    vector<int>& busRoute = routes[bus];
                    for (int nextStop : busRoute) {
                        if (busStopVisited.count(nextStop) > 0) {
                            continue;
                        }

                        q.push(nextStop);
                        busStopVisited.insert(nextStop);
                    }
                    busVisited.insert(bus);
                }
            }
        }
        ++level;
```

Input:

- Bus routes:

```
routes = {
    {1, 2, 7},    // Bus 0
    {3, 6, 7}     // Bus 1
}
```

- Source bus stop (`S = 1`)
- Destination bus stop (`T = 6`)

Step 1: Build the Map

The program constructs a map where each bus stop points to the buses that stop there. The map is:

```
map = {
    1: {0},
    2: {0},
    7: {0, 1},
    3: {1},
    6: {1}
}
```

Here:

- `1` is served by bus `0`.
- `7` is served by buses `0` and `1`.
- `6` is served by bus `1`, etc.

Step 2: BFS Initialization

- Queue `q` is initialized with the **source stop (`S = 1`)**: `q = {1}`.
- Visited sets:
    - `busStopVisited = {1}` (to track visited bus stops).
    - `busVisited = {}` (to track visited buses).
- `level = 0` (tracks the number of buses taken).

Step 3: BFS Process

Level 0:

```
    }
    return -1; // If destination is not reachable
}

int main() {
    // Hardcoded input values
    vector<vector<int>> routes = {
        {1, 2, 7},
        {3, 6, 7}
    };
    int src = 1; // source bus stop
    int dest = 6; // destination bus stop

    cout << numBusesToDestination(routes, src, dest)
<< endl;

    return 0;
}
```

- Queue size = 1 (contains `1`).
- Process bus stop `1`:
    - Stops at `1` are served by bus `0` (from `map`).
    - Bus `0` is not visited, so:
        - Add all stops from bus `0` (`{1, 2, 7}`) to the queue:
            - Add `2` to `q`.
            - Add `7` to `q`.
            - Mark stops `2` and `7` as visited (`busStopVisited = {1, 2, 7}`).
        - Mark bus `0` as visited (`busVisited = {0}`).
- **End of level 0:**
    - Queue: `q = {2, 7}`.
    - Increment `level = 1`.

Level 1:

- Queue size = 2 (contains `2, 7`).
- Process bus stop `2`:
    - Stops at `2` are served by bus `0`, which is already visited (`busVisited = {0}`).
    - Skip further processing for stop `2`.
- Process bus stop `7`:
    - Stops at `7` are served by buses `0` and `1` (from `map`).
    - Bus `0` is already visited.
    - Bus `1` is not visited, so:
        - Add all stops from bus `1` (`{3, 6, 7}`) to the queue:
            - Add `3` to `q`.
            - Add `6` to `q`.
            - Mark stops `3` and `6` as visited (`busStopVisited = {1, 2, 3, 6, 7}`).
        - Mark bus `1` as visited (`busVisited = {0, 1}`).
- **End of level 1:**
    - Queue: `q = {3, 6}`.

    - Increment `level = 2`.

Level 2:

- Queue size = 2 (contains `3, 6`).
- Process bus stop `3`:
    - Stops at `3` are served by bus `1`, which is already visited (`busVisited = {0, 1}`).
    - Skip further processing for stop `3`.
- Process bus stop `6`:
    - `6` is the destination (`T = 6`).
    - **Return `level = 2`.**

Output:

The minimum number of buses required to travel from stop `1` to stop `6` is:

Output:-
2