Bus Routes in C++

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

using namespace std;

int
numBusesToDestination(vector<vector<i
nt>>& 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;
                  }
```

Input:

```
routes = {
    {1, 2, 7},
    {3, 6, 7}
};
src = 1;
dest = 6;
```

🧠 High-Level Idea:

The code builds a graph where each **bus stop** connects to **bus routes**, then performs **BFS** starting from the source stop to find the **minimum number of buses** needed to reach the destination.

🔁 Dry Run Table (Iterative BFS)

Iteration	Level	Queue Contents	Current Stop	Bus Routes from Stop	New Stops Added to Queue	Bus Visited	Comments
Init	0	[1]	-	-	-	-	Start from stop 1
1	0	[1]	1	[0]	[2, 7]	{0}	Stop 1 is in route 0; enqueue 2, 7
2	1	[2, 7]	2	[0]	-	{0}	Bus 0 already visited
3	1	[7]	7	[0, 1]	[3, 6]	{0, 1}	Route 1 has 6 (destination !)
4	2	[3, 6]	3	[1]	-	{0, 1}	Already visited bus 1
5	2	[6]	6	[1]	-	{0, 1}	🎯 Destination reached

```
                q.push(nextStop);

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

✅ Result:

The `level` when we reach stop `6` is **2**, but since levels are **incremented after each BFS layer**, and the first bus was taken at level 0:

☞ **Minimum buses required = 2**

🔚 Final Output:

```
2
```

Output:-
2