Aviral Raj(199302157)
IT-C

AI/ML LAB-1

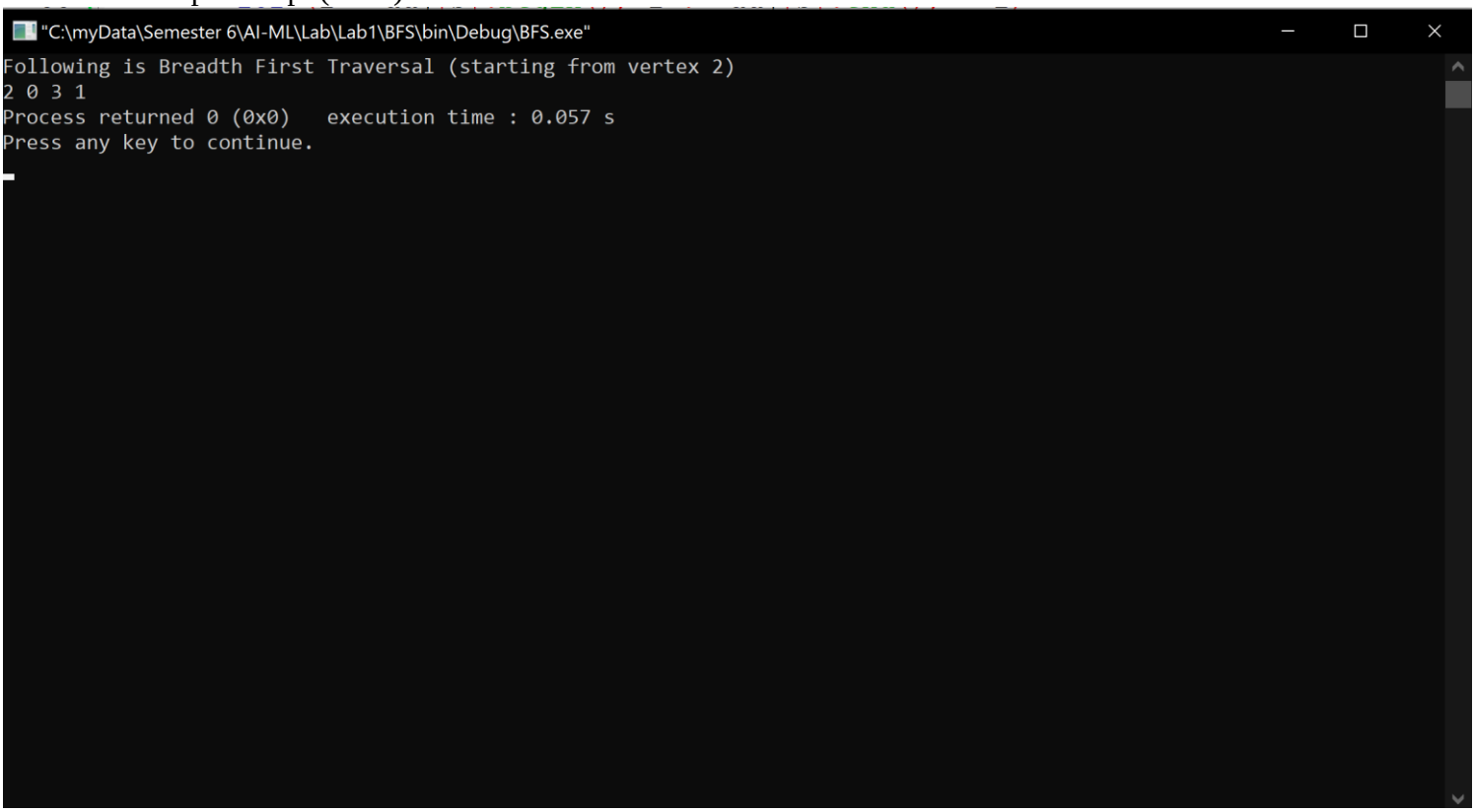
AIM: To perform BFS
Solution:

```
#include<iostream>
#include <list>
using namespace std;class Graph
{
	int V;
	list<int> *adj;
public:
	Graph(int V);
	void addEdge(int v, int w);
	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);
}
void Graph::BFS(int s)
{
	bool *visited = new bool[V];
	for(int i = 0; i < V; i++)
		visited[i] = false;
	list<int> queue;
	visited[s] = true;
	queue.push_back(s);
	list<int>::iterator i;
	while(!queue.empty())
	{
		s = queue.front();
		cout << s << " ";
		queue.pop_front();
		for (i = adj[s].begin(); i != adj[s].end(); ++i)
		{
			if (!visited[*i])
			{
				visited[*i] = true;
				queue.push_back(*i);
			}
```

```
                }
        }
}
int main()
{
        Graph g(4);
        g.addEdge(0, 1);
        g.addEdge(0, 2);
        g.addEdge(1, 2);
        g.addEdge(2, 0);
        g.addEdge(2, 3);
        g.addEdge(3, 3);
        cout << "Following is Breadth First Traversal "
                << "(starting from vertex 2) \n";
        g.BFS(2);
        return 0;
}
```

//DFS

```
#include<iostream>
#include <list>
using namespace std;class Graph
{
        int V;
        list<int> *adj;
public:
        Graph(int V);
        void addEdge(int v, int w);
        void BFS(int s);
};
Graph::Graph(int V)
```

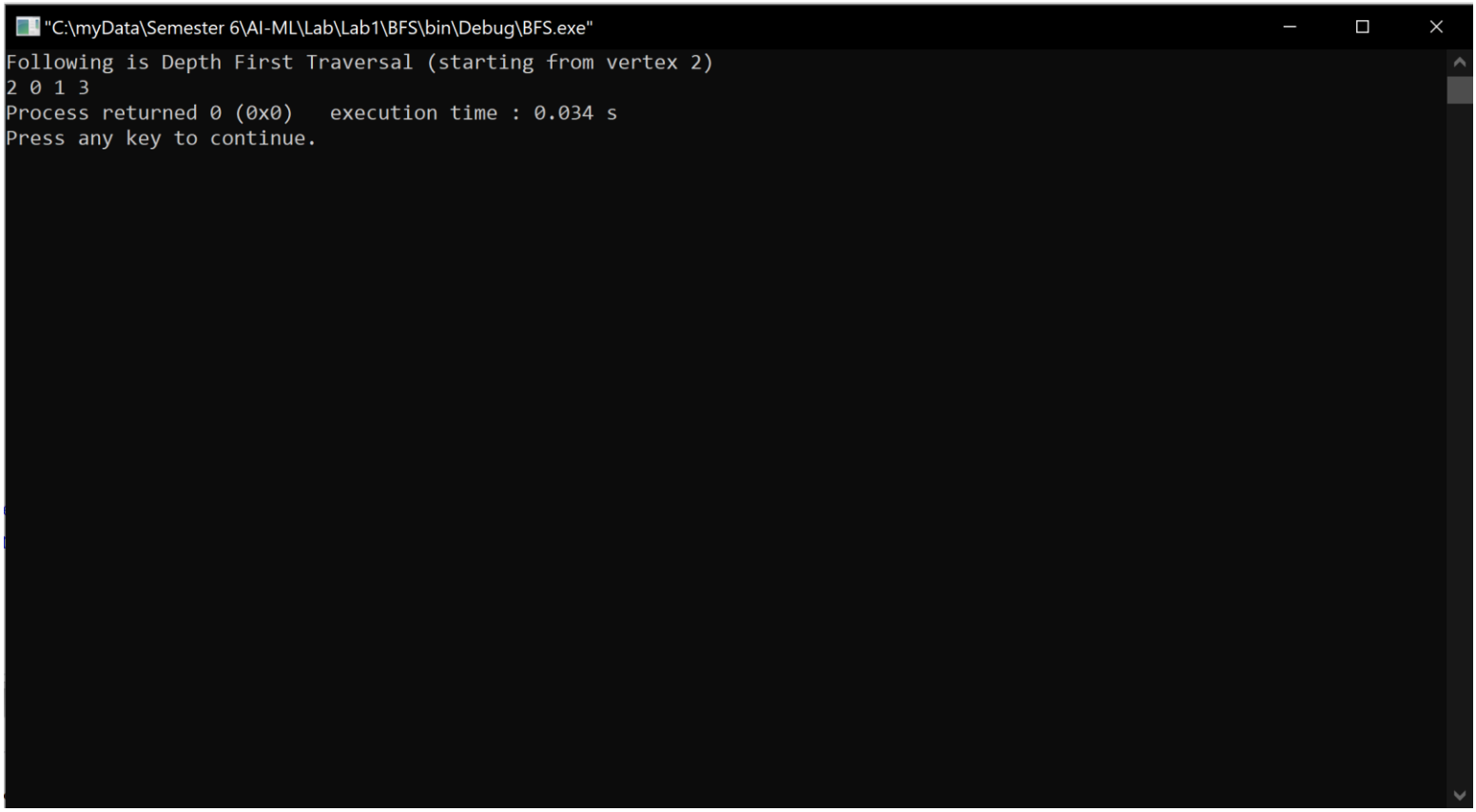


```
{
	this->V = V;
	adj = new list<int>[V];
}
void Graph::addEdge(int v, int w)
{
	adj[v].push_back(w);
}
void Graph::BFS(int s)
{
	bool *visited = new bool[V];
	for(int i = 0; i < V; i++)
		visited[i] = false;
	list<int> queue;
	visited[s] = true;
	queue.push_back(s);
	list<int>::iterator i;
	while(!queue.empty())
	{
		s = queue.front();
		cout << s << " ";
		queue.pop_front();
		for (i = adj[s].begin(); i != adj[s].end(); ++i)
		{
			if (!visited[*i])
			{
				visited[*i] = true;
				queue.push_back(*i);
			}
		}
	}
}
int main()
{
	Graph g(4);
	g.addEdge(0, 1);
	g.addEdge(0, 2);
	g.addEdge(1, 2);
	g.addEdge(2, 0);
	g.addEdge(2, 3);
	g.addEdge(3, 3);
	cout << "Following is Breadth First Traversal "
		<< "(starting from vertex 2) \n";
	g.BFS(2);
	return 0;
```

```
}
```

AIM: To perform Water Jug Problem
Solution:

```
#include <bits/stdc++.h>
using namespace std;
typedef pair<int,int> pii;
void printpath(map<pii,pii>mp ,pii u)
{
if(u.first==0 &&u.second==0)
        {
        cout<<0<<" "<<0<<endl;
        return ;
        }
printpath(mp,mp[u]);
        cout<<u.first<<" "<<u.second<<endl;
}
void BFS(int a ,int b, int target)
{
map<pii, int>m;
bool isSolvable =false;
vector<tuple<int ,int ,int>>path;
```



```
map<pii, pii>mp;
queue<pii>q;
q.push(make_pair(0,0));
while(!q.empty())
{
		auto u =q.front();
	q.pop();
	if(m[u]==1)
		continue;
	if ((u.first > a || u.second > b || u.first < 0 || u.second < 0))
		continue;
	m[{u.first,u.second}]=1;
	if(u.first == target || u.second==target)
	{
		isSolvable = true;
		printpath(mp,u);
			if (u.first == target) {
			if (u.second != 0)
				cout<<u.first<<" "<<0<<endl;
		}
		else {
			if (u.first != 0)
				cout<<0<<" "<<u.second<<endl;
		}
			return;
	}
	if(m[{u.first,b}]!=1)
	{q.push({u.first,b});
	mp[{u.first,b}]=u;}
	if(m[{a,u.second}]!=1)
	{ q.push({a,u.second});
	mp[{a,u.second}]=u;}
	int d = b - u.second;
	if(u.first >= d)
	{
		int c = u.first - d;
		if(m[{c,b}]!=1)
		{q.push({c,b});
		mp[{c,b}]=u;}
	}
	else
	{
		int c = u.first + u.second;
		if(m[{0,c}]!=1)
		{q.push({0,c});
		mp[{0,c}]=u;}
```

```
        }
        d = a - u.first;
        if(u.second >= d)
        {
                int c = u.second - d;
                if(m[{a,c}]!=1)
                {q.push({a,c});
                mp[{a,c}]=u;}
        }
        else
        {
                int c = u.first + u.second;
                if(m[{c,0}]!=1)
                {q.push({c,0});
                mp[{c,0}]=u;}
        }
        if(m[{u.first,0}]!=1)
                { q.push({u.first,0});
                mp[{u.first,0}]=u;}
        if(m[{0,u.second}]!=1)
                {q.push({0,u.second});
                mp[{0,u.second}]=u;}
}
if (!isSolvable)
                cout << "No solution";
}
int main()
{
        int Jug1 = 5, Jug2 = 7, target = 3;
        cout << "Path from initial state "
                        "to solution state ::\n";
        BFS(Jug1, Jug2, target);
        return 0;
}
```

"C:\myData\Semester 6\AI-ML\Lab\Lab1\BFS\bin\Debug\BFS.exe"

```
Path from initial state to solution state ::
0 0
5 0
0 5
5 5
3 7
3 0

Process returned 0 (0x0)   execution time : 0.035 s
Press any key to continue.
```