M.Sc C.S - I
SEM I
E-Journal

Roll No.	027
Name	OJHA ABHISHEK DEVMANI
Subject	ARTIFICIAL INTELLIGENCE AND PROGRAMMING ROBOT

CERTIFICATE

This is here to certify that Mr. **OJHA ABHISHEK DEVMANI**, Seat Number **027** of M.Sc. I Computer Science, has satisfactorily completed the required number of experiments prescribed by the UNIVERSITY OF MUMBAI during the academic year 2021 – 2022.

Date:

Place: Mumbai

Teacher In-Charge   Head of Department

External Examiner

INDEX

Sr. No.	Practical Name	Date
1	Write a program to create a robot **(i)** With gear **(ii)** Without gear and move it forward, left, right	
2	Write a program to create a robot and add two motors to it, make it move forward, left and right.	
3	**3.A** Write a program to create a robot that moves in a square using a while loop. **3.B** Write a program to create a robot that moves in a square using a for loop.	
4	Write a program to create a robot with light sensors to follow a line.	
5	Write a program to create a robot that does a circle using 2 motors.	
6	Write a program to create a path following robot.	
7	Write a program to implement (BFS) algorithm for a given standard problem.	

ARTIFICIAL INTELLIGENCE AND PROGRAMMING ROBOT

PRACTICAL NO 1

027_Abhishek_Ojha

Practical No. 1.A

Aim:

Write a program to create a robot to move, forward, turn left and right without gears.

Theory:

Turtles are a class of educational robots designed originally in the late 1940s (largely under the auspices of researcher William Grey Walter)[citation needed] and used in computer science and mechanical engineering training. These devices are traditionally built low to the ground with a roughly hemispheric (sometimes transparent) shell and a power train capable of a very small turning radius. The robots are often equipped with sensor devices which aid in avoiding obstacles and, if the robot is sufficiently sophisticated, allow it some perception of its environment. Turtle robots are commercially available and are common projects for robotics hobbyists.

Code:

```java
package robotwithoutgear;

import ch.aplu.robotsim.*;

public class RobotWithoutGear {

    public RobotWithoutGear(){

        TurtleRobot robot=new TurtleRobot();
        robot.forward(100);
        robot.left(45);
        robot.forward(100);
        robot.right(90);
        robot.backward(100);
        robot.exit();
    }

    public static void main(String[] args) {
        // TODO code application logic here
        new RobotWithoutGear();

    }

}
```

Output:

Conclusion:

We successfully used the turtle robot to move the robot without the use of a gear.

Practical No. 1.B

Aim:

Write a program to create a robot to move, forward, turn left and right with gears.

Theory:

A gear is a wheel with evenly sized and spaced teeth machined or formed around its perimeter. Gears are used in rotating machinery not only to transmit motion from one point to another, but also for the mechanical advantage they offer. Two or more gears transmitting motion from one shaft to another is called a gear train, and gearing is a system of wheels or cylinders with meshing teeth. Gearing is chiefly used to transmit rotating motion but can also be adapted to translate reciprocating motion into rotating motion and vice versa.

Gears are versatile mechanical components capable of performing many different kinds of power transmission or motion control.

Examples of these are:
• Changing Rotational Speed
• Changing Rotational Direction
• Changing the angular orientation of rotational motion
• Multiplication or division of torque or magnitude of rotation
• Converting rotational to linear motion, and its reverse
• Offsetting or changing the location of rotating motion

Code:

```
package robotwithgear;
import ch.aplu.robotsim.*;
public class RobotWithGear {
    public RobotWithGear(){
        LegoRobot robot = new LegoRobot();
        Gear gear = new Gear();
        robot.addPart(gear);
        gear.forward(2000);
        gear.setSpeed(30);
        gear.left(200);
        gear.forward(2000);
        gear.right(200);
        gear.forward();
        Tools.delay(20);
        robot.exit();
    }
    public static void main(String[] args) {
        // TODO code application logic here
        new RobotWithGear();
    }
}
```

Output:
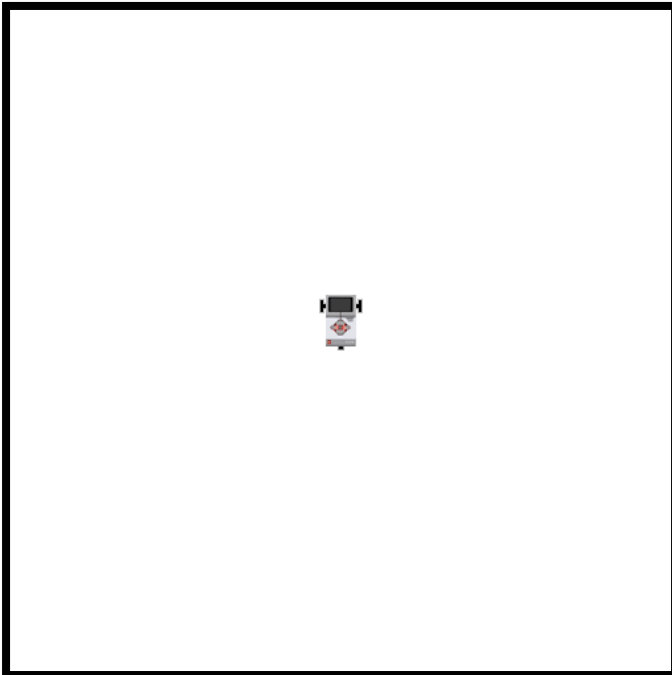
Conclusion: We successfully used lego robot and gear to allow movement in our robot.

ARTIFICIAL INTELLIGENCE AND PROGRAMMING ROBOT

PRACTICAL NO 2

027_Abhishek_Ojha

Practical No. 2

Aim:

Write a program to create a robot and add two motors to it, make it move forward, left and right.

Theory:

Motors are one of the primary mechanisms by which robots move. Some motors can be attached to wheels that drive a robot around. Other motors might cause joints in a robot limb to move. Yet others might move the control surfaces of a robotic airplane or submarine. A robot might have many different kinds of effectors to perform specific tasks, but many of these effectors are being moved around by motors.

What motors do is convert the electrical energy that powers the robot into mechanical energy that allows the robot to do work. There are two measurements of a motor that are important for understanding how much work it can do.

Speed is what the maximum speed of the motor is. This is usually measured in revolutions per minute, or RPM. 1 RPM means that the axle of the motor will turn completely around a circle once in a minute, which is very slow. Even a very cheap DC motor will have a speed rating of at least 1000 RPM.

Torque is the measure of how much strength the motor outputs. A simple way to think about it is that if you attached an arm with a weight on it to the axis of the motor, what is the maximum amount of weight that the motor could lift.

Code:

```java
package movewithmotor;
import ch.aplu.robotsim.*;

public class MoveWithMotor {

    public MoveWithMotor() {

        LegoRobot robot = new LegoRobot();
        Motor motA = new Motor(MotorPort.A);
        Motor motB = new Motor(MotorPort.B);
        robot.addPart(motA);
        robot.addPart(motB);

        motA.forward();
        motB.forward();
        Tools.delay(2000);
```

```
    motA.stop();
    Tools.delay(2000);

    motB.stop();
    Tools.delay(2000);

    motA.backward();
    motB.forward();
    Tools.delay(2000);

    motB.backward();
    Tools.delay(2000);

    robot.exit();
  }
  public static void main(String[] args) {
    new MoveWithMotor();
  }
}
```
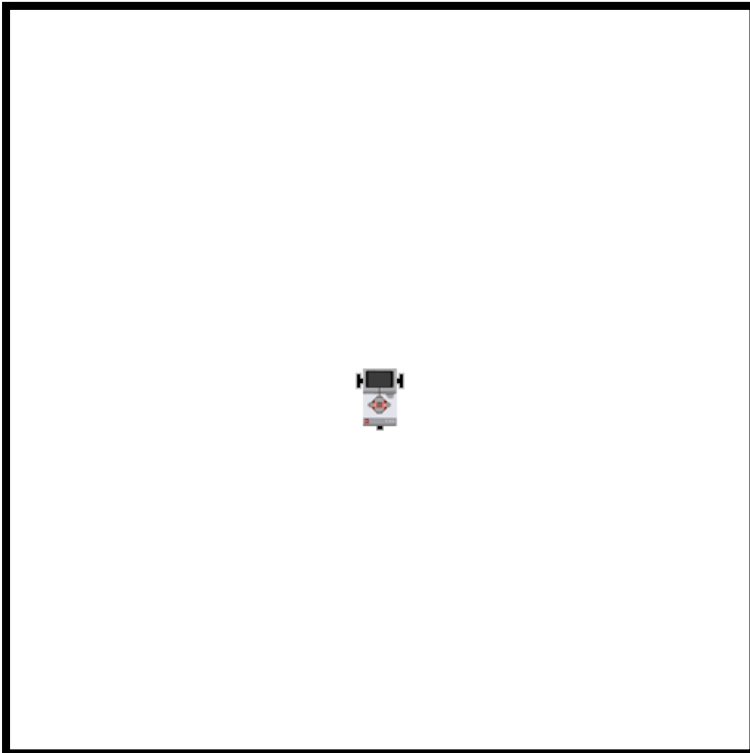
Output:

Conclusion: We successfully use two motor on a Lego robot to move the robot.

ARTIFICIAL INTELLIGENCE AND PROGRAMMING ROBOT

PRACTICAL NO 3

027_Abhishek_Ojha

Practical No. 3.A

Aim:

Write a program to create a robot that moves in a square using a while loop.

Theory:

A gear is a wheel with evenly sized and spaced teeth machined or formed around its perimeter. Gears are used in rotating machinery not only to transmit motion from one point to another, but also for the mechanical advantage they offer. Two or more gears transmitting motion from one shaft to another is called a gear train, and gearing is a system of wheels or cylinders with meshing teeth. Gearing is chiefly used to transmit rotating motion but can also be adapted to translate reciprocating motion into rotating motion and vice versa.

We use a while loop to continuously turn 90 degree after moving forward to move in a square path.

Code:

```java
package squarewithwhile;

import ch.aplu.robotsim.*;

public class SquareWithWhile {

    static {
        RobotContext.setStartDirection(90);
        RobotContext.setStartPosition(100, 100);
    }
    public SquareWithWhile() {

        LegoRobot robot = new LegoRobot();
        Gear gear = new Gear();
        robot.addPart(gear);
        gear.setSpeed(100);
        while (true) {
            gear.forward(1000);
            gear.left(90);
        }
    }
public static void main(String[] args) {
        new SquareWithWhile();
    }
}
```
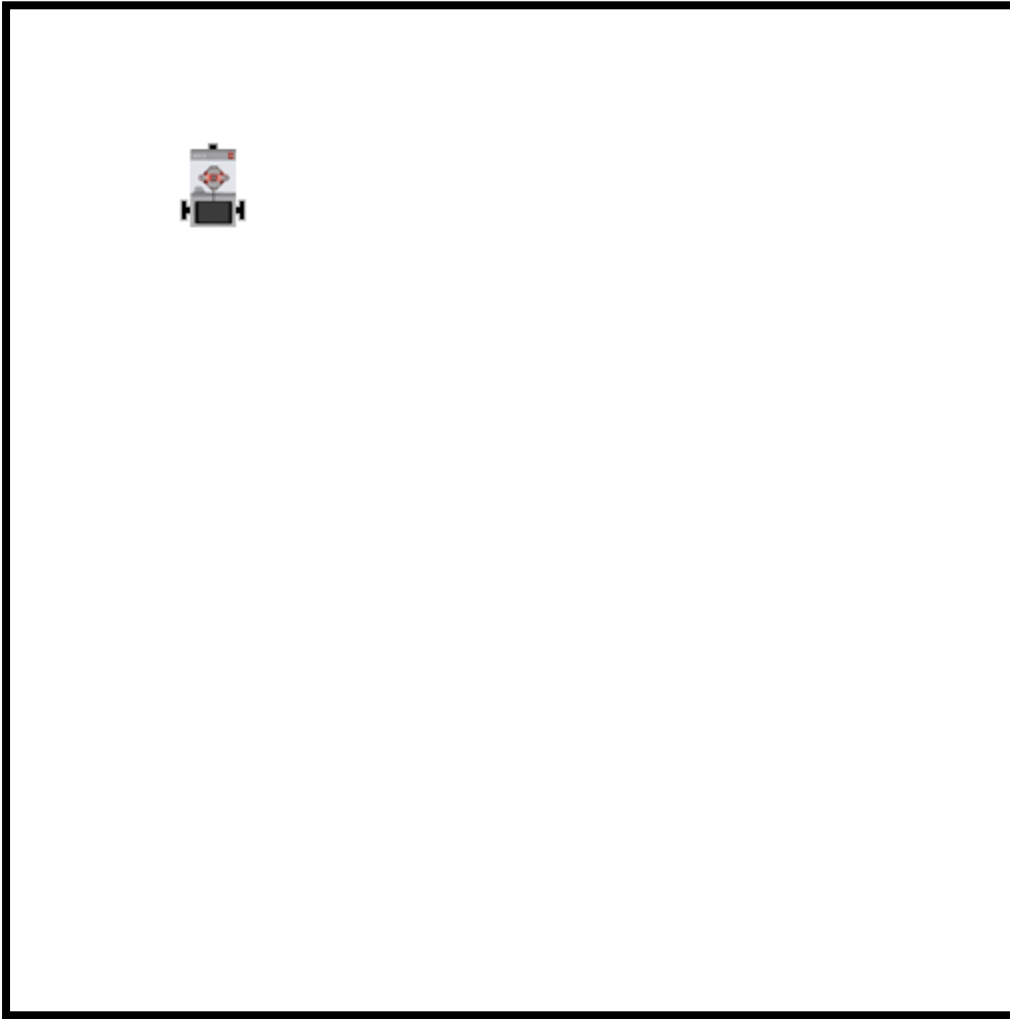
Output:

Conclusion:

   We successfully used gear and a while loop to simulate movement in a square path.

Practical No. 3.B

Aim:

Write a program to create a robot that moves in a square using a for loop.

Theory:

A gear is a wheel with evenly sized and spaced teeth machined or formed around its perimeter. Gears are used in rotating machinery not only to transmit motion from one point to another, but also for the mechanical advantage they offer. Two or more gears transmitting motion from one shaft to another is called a gear train, and gearing is a system of wheels or cylinders with meshing teeth. Gearing is chiefly used to transmit rotating motion but can also be adapted to translate reciprocating motion into rotating motion and vice versa.

We use a for loop to continuously turn 90 degree after moving forward to move in a square path.

Code:

```
package stepswithfor;

import ch.aplu.robotsim.*;

public class StepsWithFor {

   static {
      RobotContext.setStartDirection(90);
      RobotContext.setStartPosition(100, 100);
   }

   public StepsWithFor() {

      LegoRobot robot = new LegoRobot();
      Gear gear = new Gear();
      robot.addPart(gear);
      gear.setSpeed(100);
      for (int i = 1; i <= 4; i++) {
         gear.forward(1000);
         gear.left(90);
      }
      Tools.delay(2000);
      robot.exit();
```

```
    }
  public static void main(String[] args) {
    new StepsWithFor();
  }
}
```

Output:

Conclusion:

We successfully used gear and a for loop to simulate movement in a square path.

ARTIFICIAL INTELLIGENCE AND PROGRAMMING ROBOT

PRACTICAL NO 4

027_Abhishek_Ojha

Practical No. 4

Aim:

Write a program to create a robot with light sensors to follow a line.

Theory:

Light sensor is a transducer used for detecting light and creates a voltage difference equivalent to the light intensity fall on a light sensor.

The two main light sensors used in robots are **Photovoltaic cells and Photo resistor.** Other kind of light sensors like phototransistors, phototubes are rarely used.

The type of light sensors used in robotics are:

- **Photo resistor –**

    It is a type of resistor used for detecting the light. In photo resistor resistance varies with change in light intensity. The light falls on photo resistor is inversely proportional to the resistance of the photo resistor. In general photo resistor is also called as Light Dependent Resistor (LDR).

Consider the circuit diagram of Photo resistor sensor:
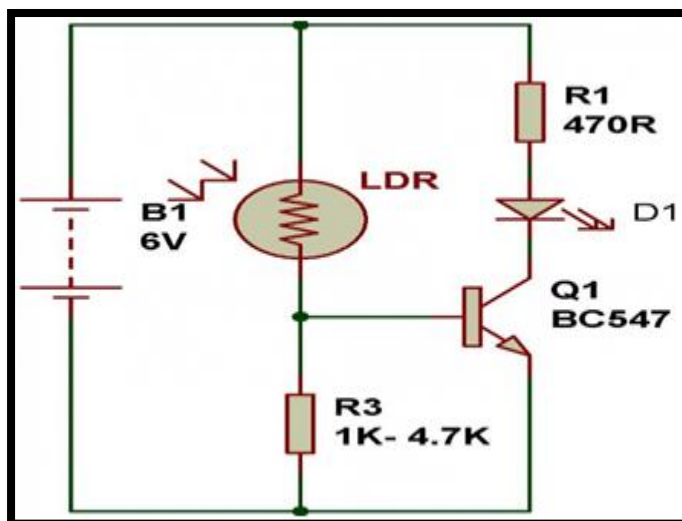
- **Photovoltaic Cells –**

    Photovoltaic cells are energy conversion device used to convert solar radiation into electrical electric energy. It is used if we are planning to build a solar robot. Individually photovoltaic cells are considered as an energy source, an implementation combined with capacitors and transistors can convert this into a sensor.

Consider the circuit diagram of photovoltaic cell is,
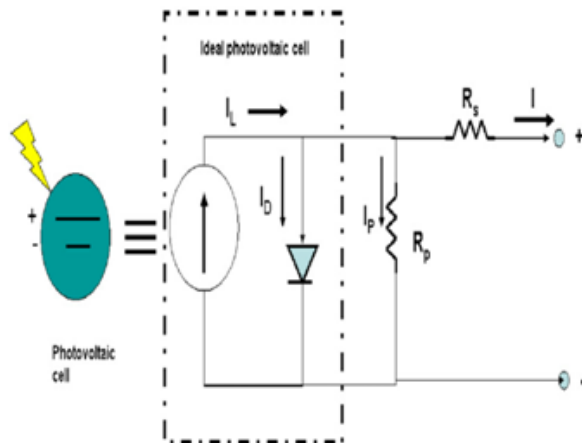
Code:

```java
package robotwithlightsensor;
import ch.aplu.robotsim.*;
public class RobotWithLighSensor {
    static {
        RobotContext.setStartDirection(90);
        RobotContext.setStartPosition(250, 10);
        RobotContext.useBackground("sprites/black_white.gif");
    }
//   Making gear global to be used in handlers
    private Gear gear = new Gear();
// initiate a legorobot with lighsensor and gear
    public RobotWithLighSensor() {
        LegoRobot robot = new LegoRobot();
        LightSensor ls = new LightSensor(SensorPort.S3);
        robot.addPart(gear);
        robot.addPart(ls);
        gear.forward();
        while (true) {
            if (ls.getValue() > 500) {
                gear.leftArc(0.1);
            } else {
                gear.rightArc(0.1);
            }
        }
    }
    public static void main(String[] args) {
        new RobotWithLighSensor();
    }
}
```

Output:

Conclusion:

We successfully made use of light sensor to make a Lego robot follow a line/path.

ARTIFICIAL INTELLIGENCE AND PROGRAMMING ROBOT

PRACTICAL NO 5

027_Abhishek_Ojha

Practical No. 5

Aim:

Write a program to create a robot that does a circle using 2 motors.

Theory:

Motors are one of the primary mechanisms by which robots move. Some motors can be attached to wheels that drive a robot around. Other motors might cause joints in a robot limb to move. Yet others might move the control surfaces of a robotic airplane or submarine. A robot might have many different kinds of effectors to perform specific tasks, but many of these effectors are being moved around by motors.

To make a robot go in circle using two motors we set one of the motors at lower speed then the other.

Code:

```java
package circlewithmotor;
/**
 *
 * @author STreK
 */
import ch.aplu.robotsim.*;
public class CircleWithMotor {
   public CircleWithMotor(){

 LegoRobot robot = new LegoRobot();
   Motor mot1 = new Motor(MotorPort.A);
   Motor mot2 = new Motor(MotorPort.B);
   robot.addPart(mot1);
   robot.addPart(mot2);
  try{
        Thread.sleep(5000);
     } catch (InterruptedException ex) {
        ex.printStackTrace();
     }
   while(true){
   mot1.forward();
   mot1.setSpeed(100);
   mot2.forward();
   mot2.setSpeed(50);
   }
 }

   public static void main(String[] args) {
```

```
      new CircleWithMotor();
  }

}
```

Output:
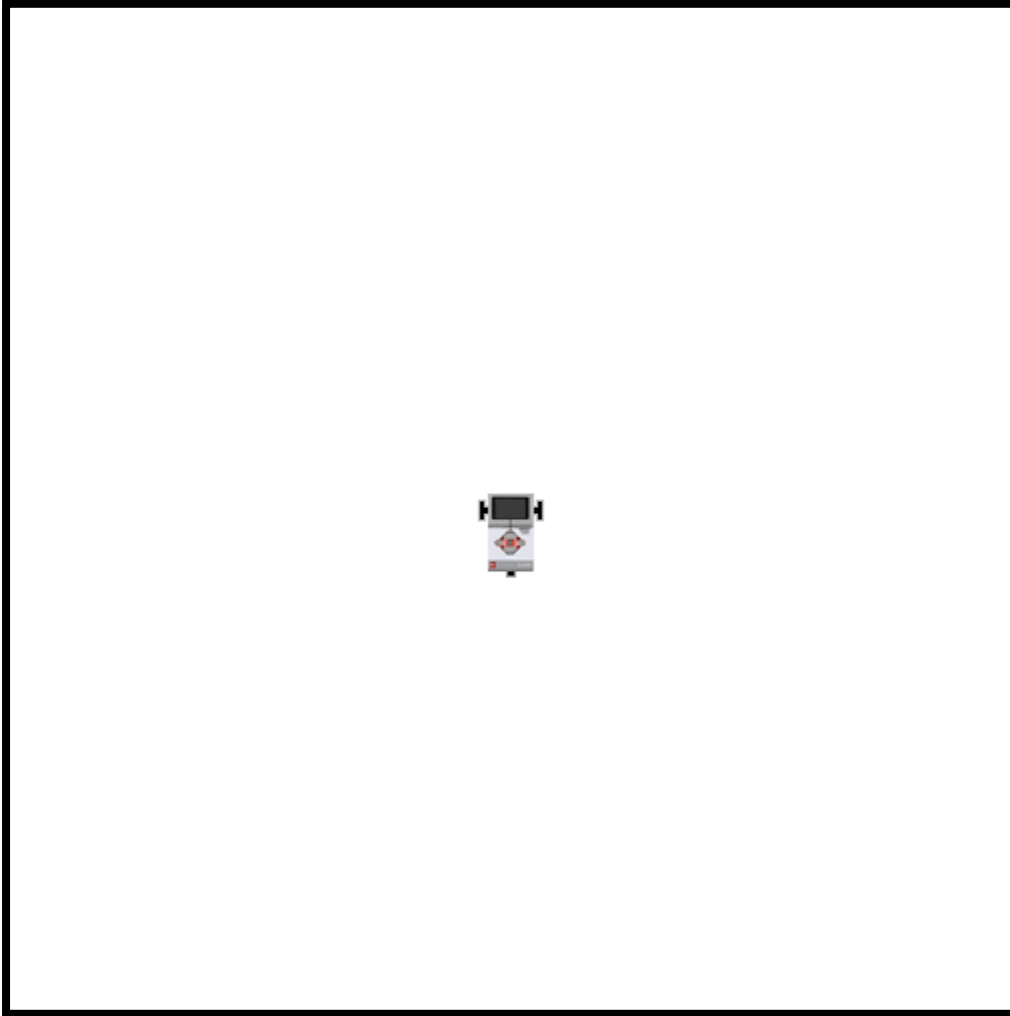
Conclusion:

We successfully used two motors one with lower speed then other to make the Lego robot go in a circle.

ARTIFICIAL INTELLIGENCE AND PROGRAMMING ROBOT

PRACTICAL NO 6

027_Abhishek_Ojha

Practical No. 6

Aim:

Write a program to create a path following robot.

Theory:

Light sensor is a transducer used for detecting light and creates a voltage difference equivalent to the light intensity fall on a light sensor.

The two main light sensors used in robots are **Photovoltaic cells and Photo resistor.** Other kind of light sensors like phototransistors, phototubes are rarely used.

We make a path using game grid where the path is denoted with a dark color and anything outside is not the path, following this assumption we use two light sensor to make the robot stay on the dark path only and turn it towards the dark path when the senor does not sense the path.

Code:

```java
package pathfollowingrobot;

import ch.aplu.robotsim.*;
import ch.aplu.jgamegrid.*;
import java.awt.*;

public class PathFollowingRobot {

  static {
    RobotContext.setStartDirection(10);
  }

  public PathFollowingRobot() {

    LegoRobot robot = new LegoRobot();
    Gear gear = new Gear();
    LightSensor ls1 = new LightSensor(SensorPort.S1);
    LightSensor ls2 = new LightSensor(SensorPort.S2);
  try{
      Thread.sleep(5000);
    } catch (InterruptedException ex) {
      ex.printStackTrace();
    }
//    Adding Parts
    robot.addPart(gear);
    robot.addPart(ls1);
    robot.addPart(ls2);
```

```java
//    initial movement
    gear.forward();

//    Trigger Level
    int intensity = 500;
//    turning arch
    double arch = 0.1;

    while (true) {
//        get values of light sensors
        int ls1v = ls1.getValue();
        int ls2v = ls2.getValue();

//        Driver Logic for movement
//        if lighsensor1 and lighsensor2 are on track move forward
        if (ls1v < intensity && ls2v < intensity) {
            gear.forward();
        }
//        if lighsensor1 is on track and lighsensor2 is not on track turn right by arch
        if (ls1v < intensity && ls2v > intensity) {
            gear.rightArc(arch);
        }
//        if  lighsensor1 is not on track and lighsensor2 is on track turn left by arch
        if (ls1v > intensity && ls2v < intensity) {
            gear.leftArc(arch);
        }
//        if lighsensor1 and lighsensor2 are not on track move backward
        if (ls1v > intensity && ls2v > intensity) {
            gear.backward();
        }
    }
}

//    Creating a track
    private static void _init(GameGrid gg) {
        GGBackground bg = gg.getBg();
//    Tracks
        bg.setPaintColor(Color.black);
        bg.fillArc(new Point(250, 150), 120, 0, 360);
        bg.fillArc(new Point(250, 350), 120, 0, 360);
        bg.fillArc(new Point(150, 250), 120, 0, 360);
        bg.fillArc(new Point(350, 250), 120, 0, 360);
//    Gaps
        bg.setPaintColor(Color.white);
        bg.fillArc(new Point(250, 350), 60, 0, 360);
```
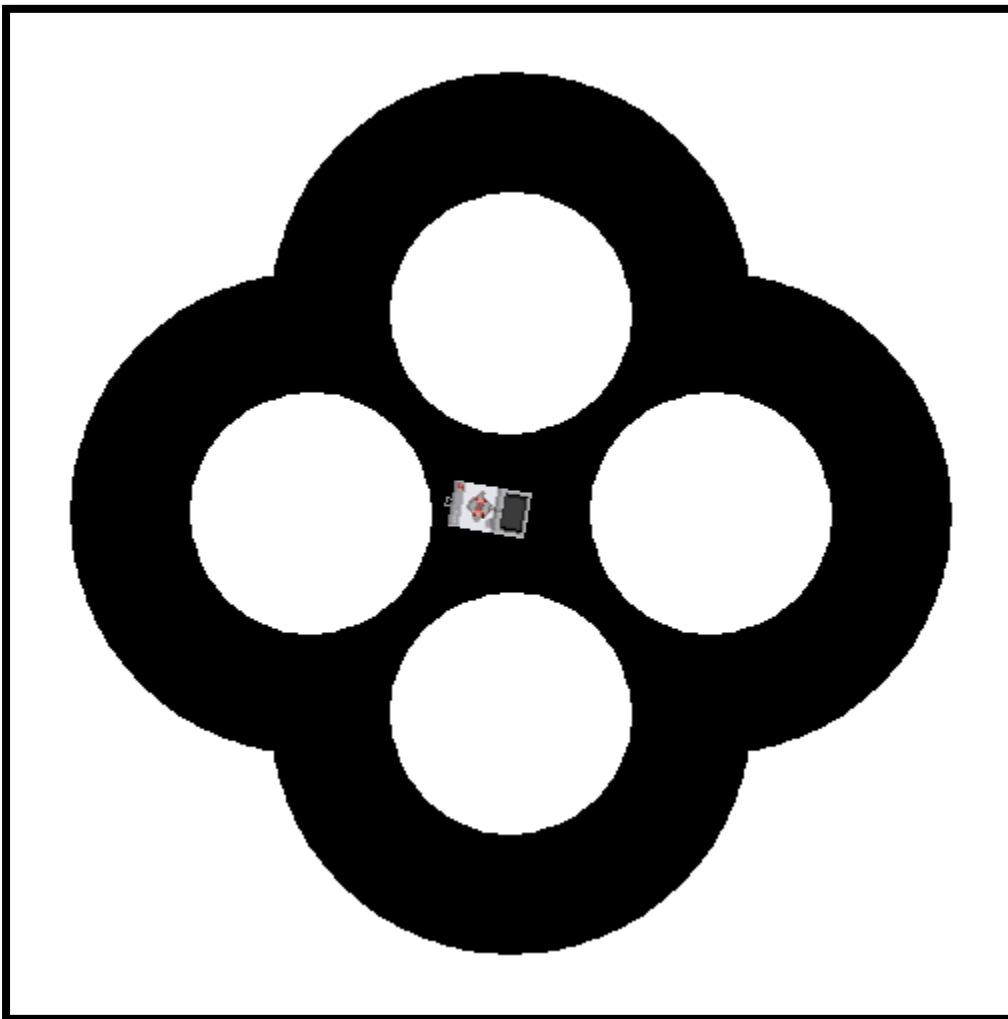
```java
        bg.fillArc(new Point(250, 150), 60, 0, 360);
        bg.fillArc(new Point(150, 250), 60, 0, 360);
        bg.fillArc(new Point(350, 250), 60, 0, 360);
    }

    public static void main(String[] args) {
        new PathFollowingRobot();
    }

}
```

Output:

Conclusion: We successfully made a Lego robot follow a path using light sensors

ARTIFICIAL INTELLIGENCE AND PROGRAMMING ROBOT

PRACTICAL NO 7

027_Abhishek_Ojha

Practical No. 7
Aim:

Write a program to implement bfs algorithm for a given standard problem.

Theory:
Breadth First Traversal or Breadth First Search is a recursive algorithm for searching all the vertices of a graph or tree data structure.

A standard BFS implementation puts each vertex of the graph into one of two categories:
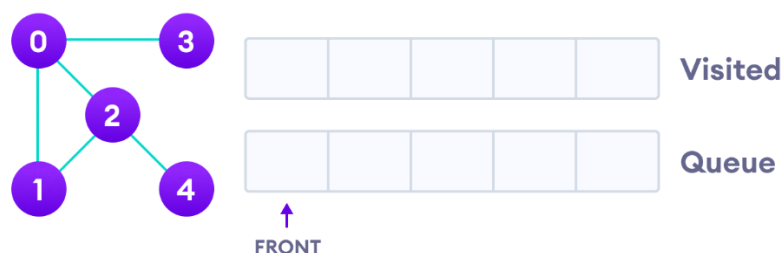
1. Visited
2. Not Visited

The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

The algorithm works as follows:

1. Start by putting any one of the graph's vertices at the back of a queue.
2. Take the front item of the queue and add it to the visited list.
3. Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the back of the queue.
4. Keep repeating steps 2 and 3 until the queue is empty.

The graph might have two different disconnected parts so to make sure that we cover every vertex, we can also run the BFS algorithm on every node.

Let's see how the Breadth First Search algorithm works with an example. We use an undirected graph with 5 vertices.
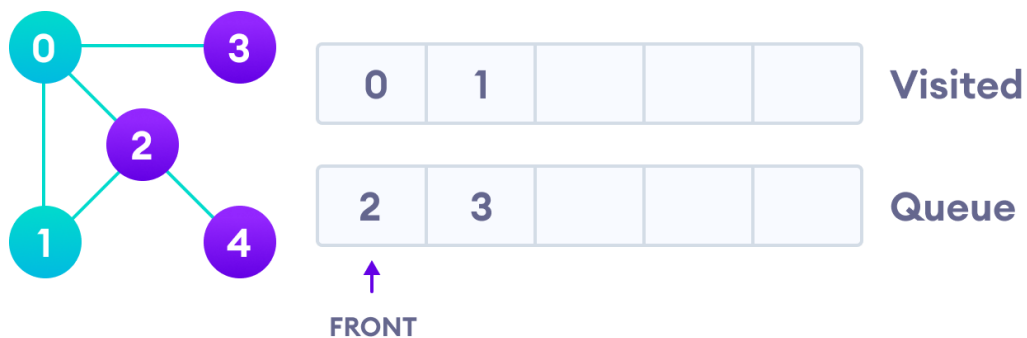

Undirected graph with 5 vertices

We start from vertex 0, the BFS algorithm starts by putting it in the Visited list and putting all its adjacent vertices in the stack.
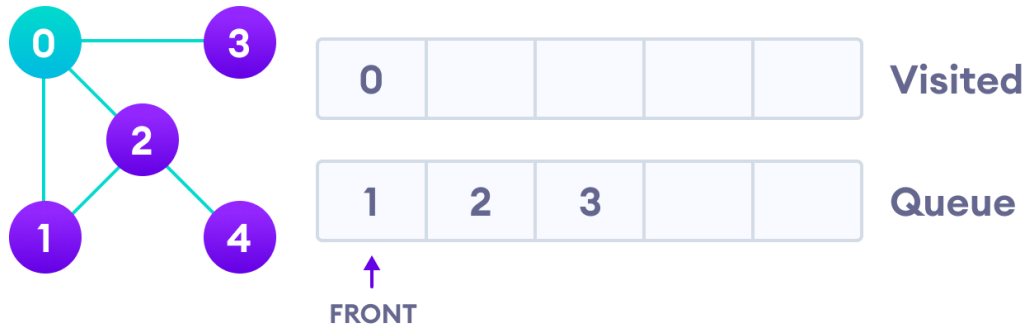
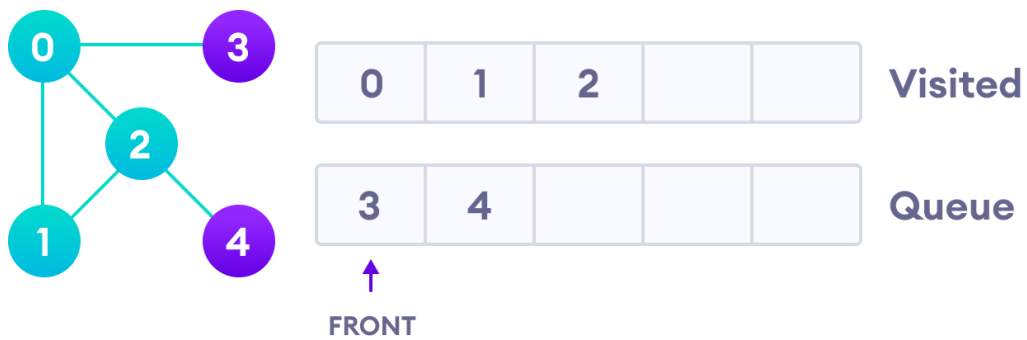Visit start vertex and add its adjacent vertices to queue

Next, we visit the element at the front of queue i.e. 1 and go to its adjacent nodes. Since 0 has already been visited, we visit 2 instead.
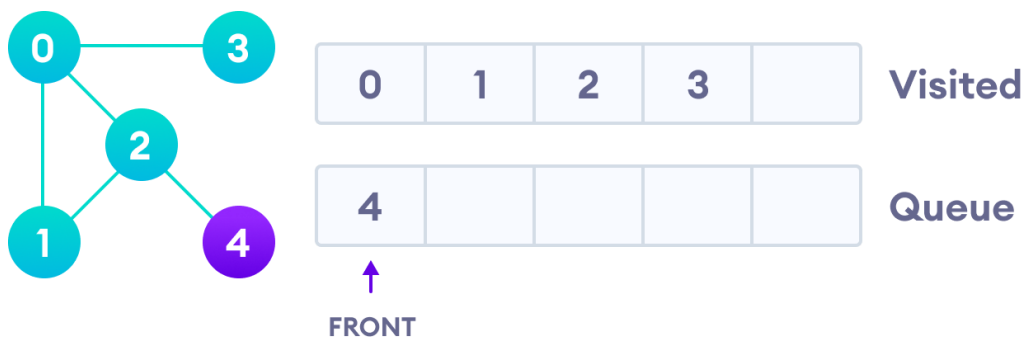

Visit the first neighbour of start node 0, which is 1

Vertex 2 has an unvisited adjacent vertex in 4, so we add that to the back of the queue and visit 3, which is at the front of the queue.

Visit 2 which was added to queue earlier to add its neighbours

4 remains in the queue

Only 4 remains in the queue since the only adjacent node of 3 i.e. 0 is already visited. We visit it.

Visit last remaining item in the stack to check if it has unvisited neighbors

Since the queue is empty, we have completed the Breadth First Traversal of the graph.

Code:

```java
package bfs;

import java.util.ArrayList;
import java.util.LinkedList;
import java.util.List;
import java.util.Queue;

public class BFS {

    private final Queue<Node> queue;
    static ArrayList<Node> nodes = new ArrayList<Node>();

    static class Node {

        int data;
        boolean visited;
        List<Node> neighbours;

        Node(int data) {
            this.data = data;
            this.neighbours = new ArrayList<>();
        }

        public void addneighbours(Node neighbourNode) {
            this.neighbours.add(neighbourNode);
        }

        public List<Node> getNeighbours() {
            return neighbours;
        }

        public void setNeighbours(List<Node> neighbours) {
            this.neighbours = neighbours;
        }
    }

//   constructor
    public BFS() {
        queue = new LinkedList<>();
    }

    public void bfs(Node node) {
        queue.add(node);
        node.visited = true;
```

```java
        while (!queue.isEmpty()) {
            Node element = queue.remove();
            System.out.print(element.data + "\t");
            List<Node> neighbours = element.getNeighbours();
            for (int i = 0; i < neighbours.size(); i++) {
                Node n = neighbours.get(i);
                if (n != null && !n.visited) {
                    queue.add(n);
                    n.visited = true;

                }
            }

        }
    }

    public static void main(String[] args) {
        Node node40 = new Node(40);
        Node node10 = new Node(10);
        Node node20 = new Node(20);
        Node node30 = new Node(30);
        Node node60 = new Node(60);
        Node node50 = new Node(50);
        Node node70 = new Node(70);

        node40.addneighbours(node10);
        node40.addneighbours(node20);
        node10.addneighbours(node30);
        node20.addneighbours(node10);
        node20.addneighbours(node30);
        node20.addneighbours(node60);
        node20.addneighbours(node50);
        node30.addneighbours(node60);
        node60.addneighbours(node70);
        node50.addneighbours(node70);
        System.out.println("The BFS traversal of the graph is ");
        BFS bfsExample = new BFS();
        bfsExample.bfs(node40);
    }

}
```

Output:

```
The BFS traversal of the graph is
40      10      20      30      60      50      70
```

Conclusion: We successfully implemented Breath first search.