OEIS A326166

RICHARD J. MATHAR

Abstract. This is a Java program that implements Fröhlich's heterosexual variant of Langton's ant moves in sequence A326166 of the Online Encyclopedia of Integer Sequences.

1. Rules

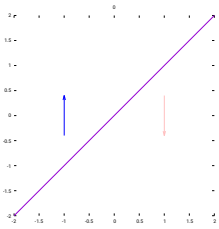
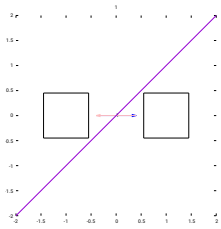
Ants of two sexes move on the simple square lattice [1, A326166]. The male ants are colored blue, the female ants pink. Ants have headings in one of the four major compass directions and occupy squares. The square lattice has white and black squares. The initial state (generation 0) consists of a male ant heading N and a female ant heading S at a horizontal distance of 2; all squares are white. At each move/generation

(1) If an ant is on a black square, its heading changes by 90° clockwise, and it moves one square into that new direction.
(2) If an ant is on a white square, its heading changes by 90° counter-clockwise, and it moves one square into that new direction.
(3) If ants leave a square, the square's color is flipped.
(4) For each pair of ants of different sex on the same square, an egg is left behind as they leave. This fertility is one egg per pair, which means the number of eggs added to the square is the minimum of the number of males and females in that square. If only ants of the same sex are on the square, no such changes of egg numbers happen.
(5) An egg hatches (is transformed into an ant), when in the forthcoming move no ants from any of the four neighboring directions enter the cell. (According to the rules the ants in the cell of the eggs are leaving.)
(6) In each cell at most one egg can become an ant per iteration.
(7) The gender of the hatched ant is male if the square on which it hatches is white at the time of hatching, female if the square is black.
(8) The hatched ant has the initial heading N.

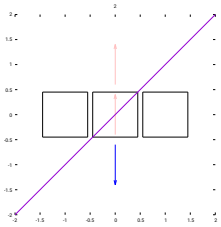
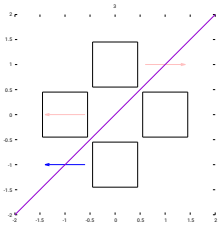
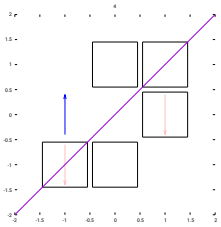
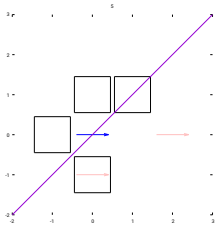
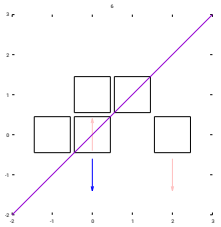
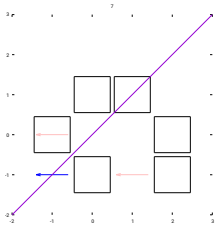
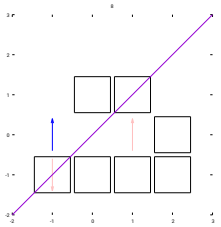
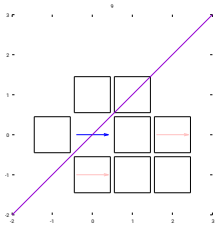
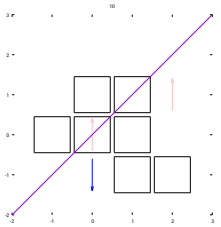
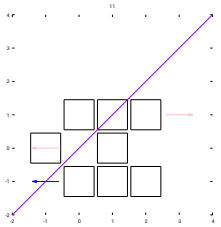
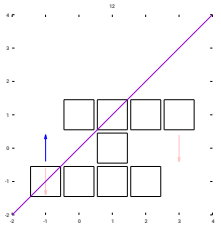
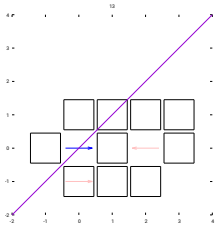
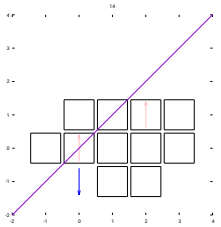
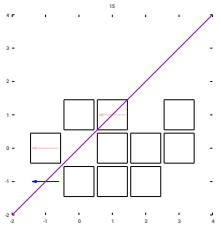
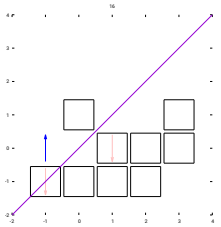
2. Illustrations

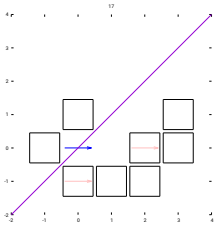
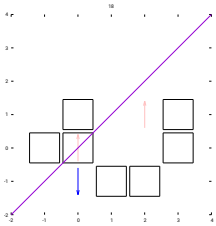
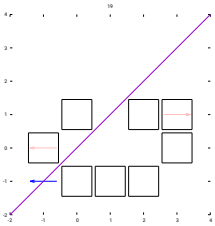
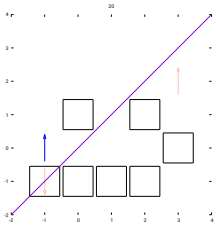
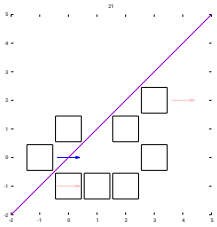
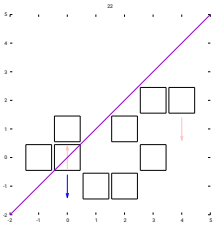
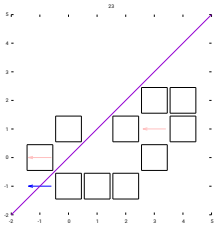
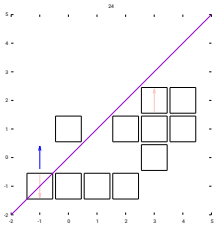
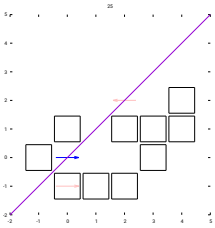
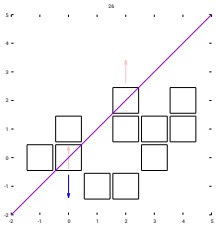
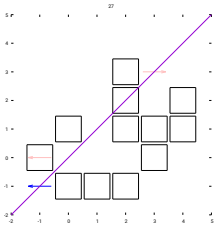
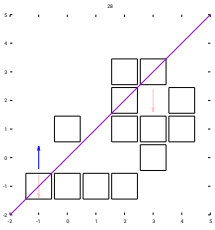
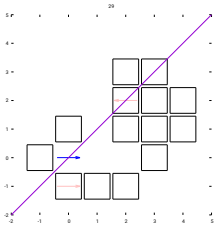
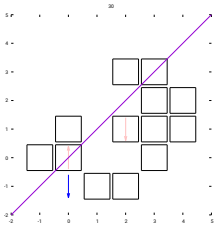
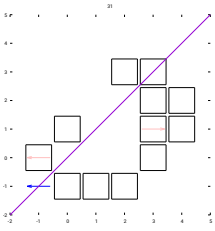
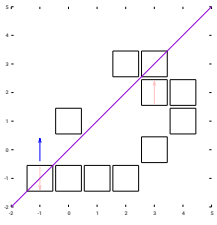
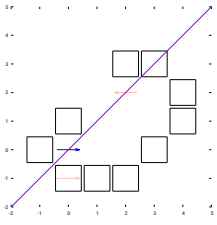
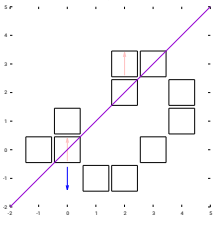
In the illustrations black squares are indicated by a small solid outline, white squares have no such explicit frame. The number of eggs in a cell is the positive integer number in the illustrations; no such number if printed for zero eggs. As a guideline to the eye a diagonal bluish line is drawn through the lattice; the bias of the initial heading of hatched ants means that the populations grows rather northwards than southwards.

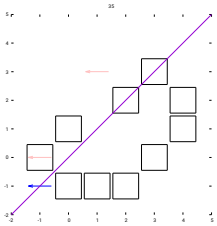
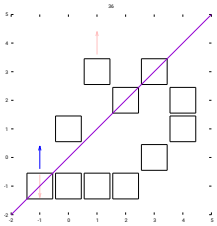
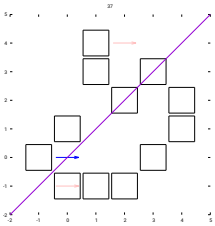
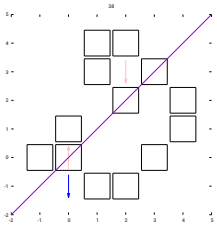
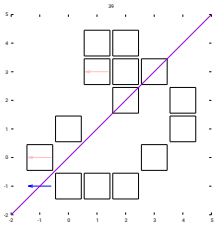
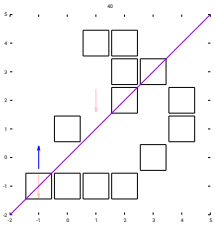
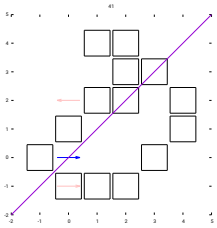
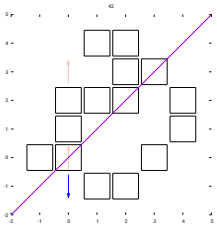
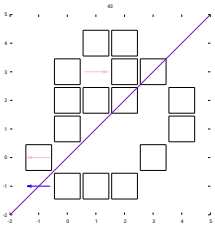
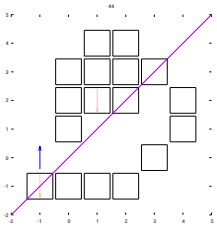
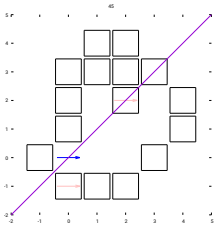
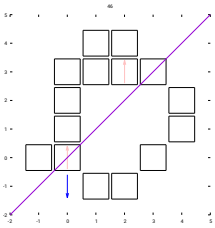
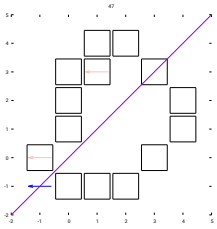
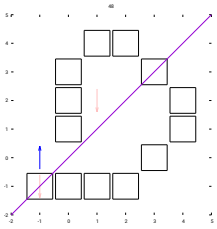
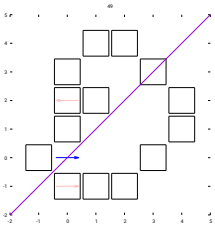
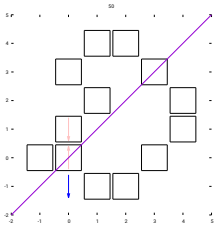
Date: November 24, 2023.
2020 *Mathematics Subject Classification.* Primary 37B15; Secondary 92D25.

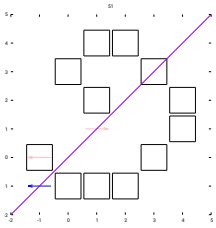



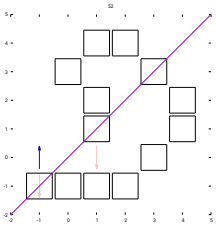
2.1. **Moves 0–50.**

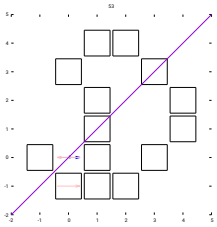



















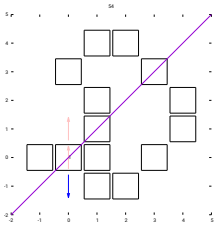

















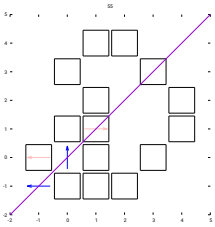


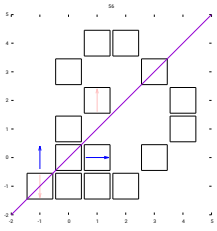


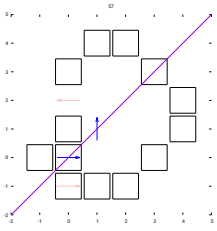
2.2. **Moves 51–100.**

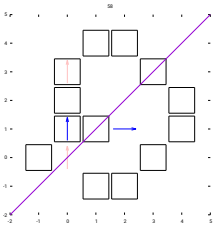


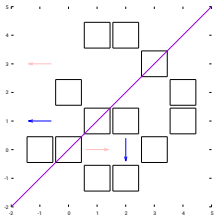


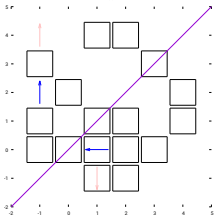


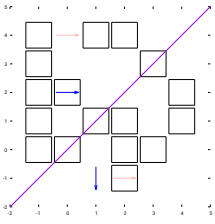


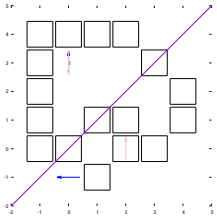


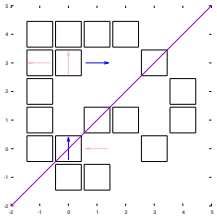


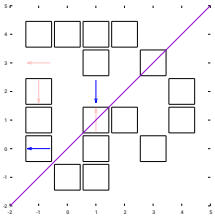


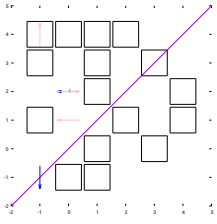


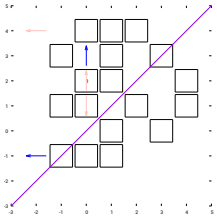


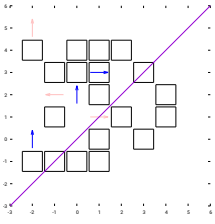


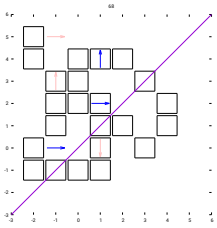
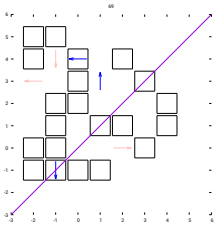
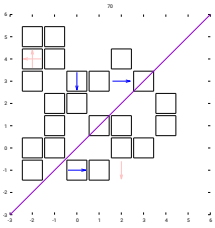
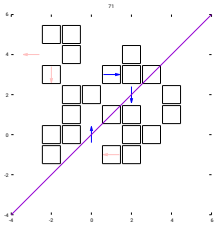
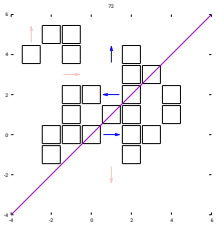
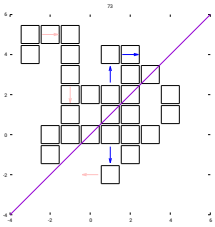
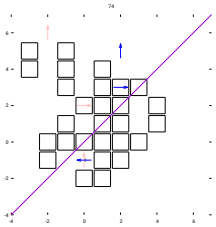
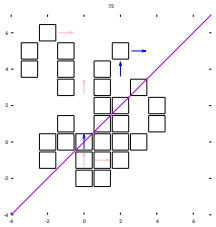
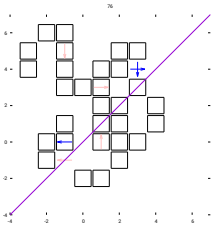
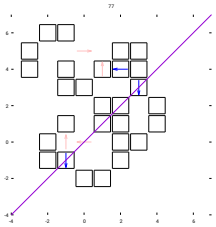
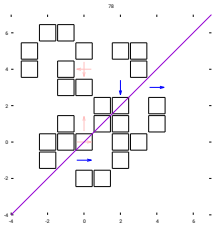
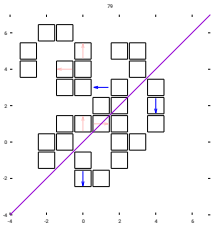
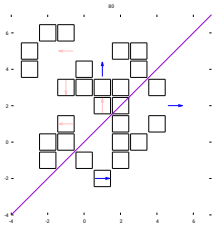
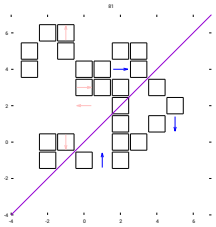
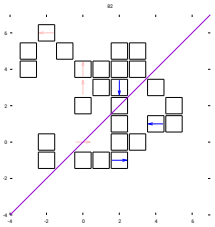
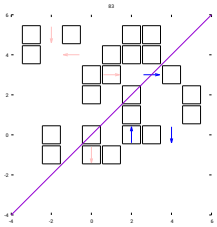
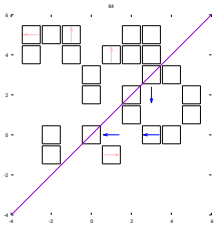
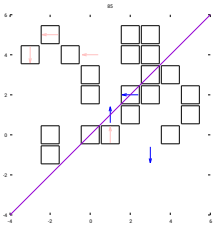


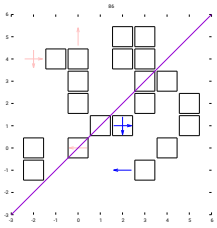
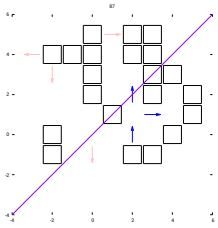
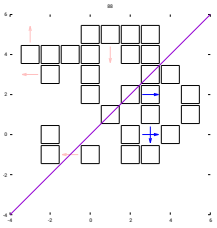
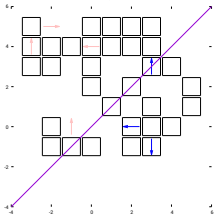
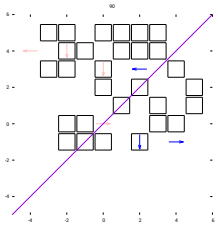
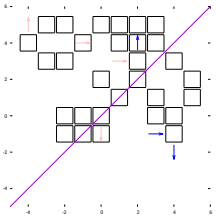
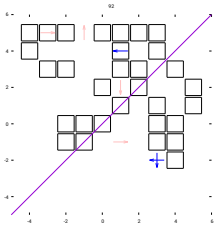
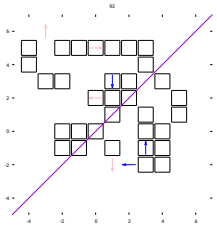
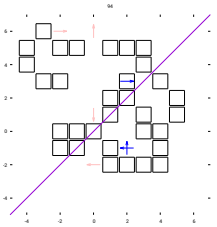
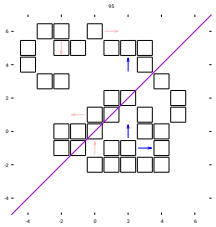
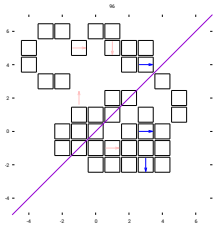
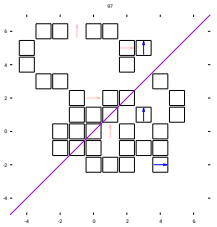
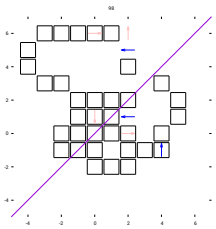
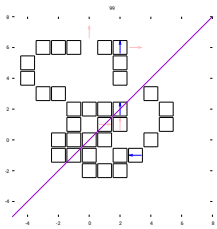
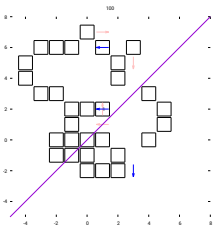


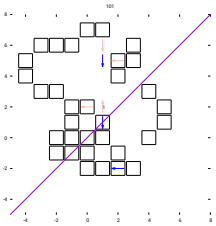
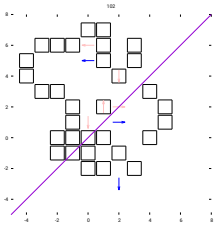




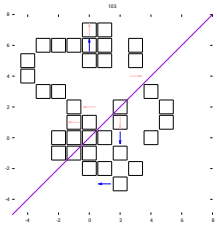
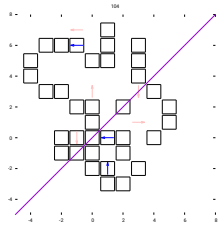
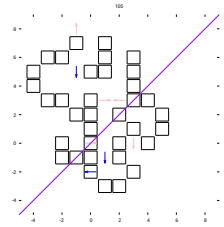
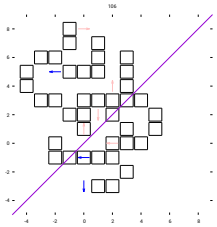
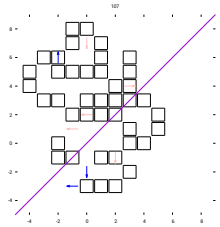
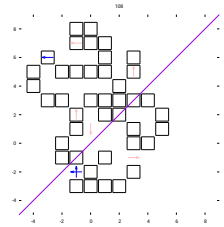
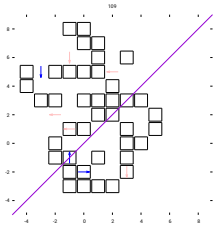
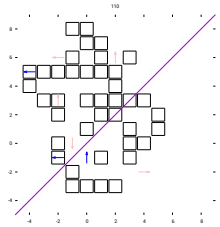
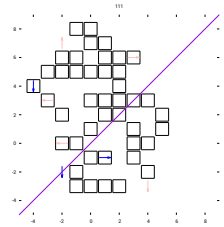
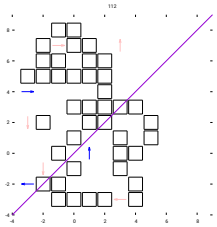
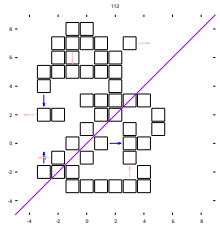
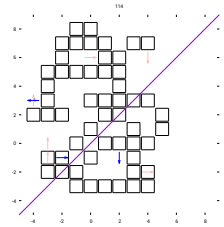
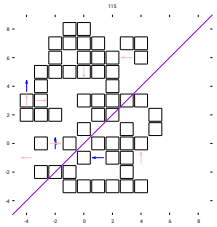
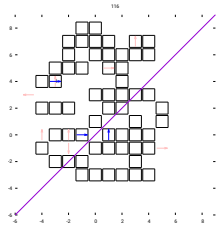
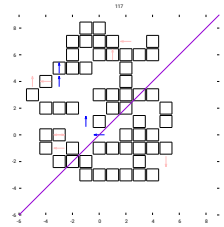


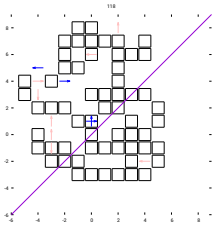
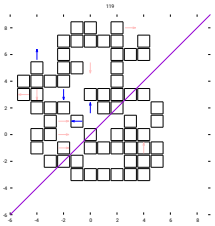
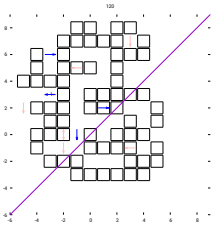
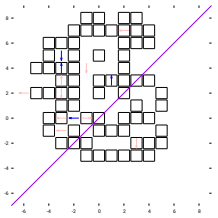
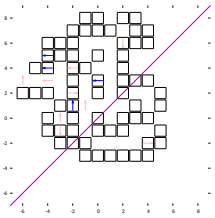
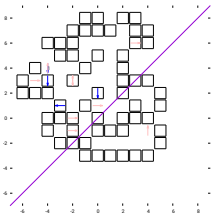
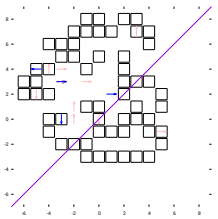
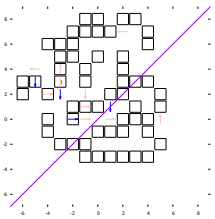
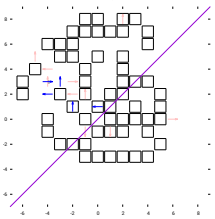
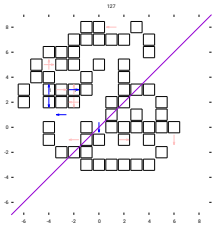
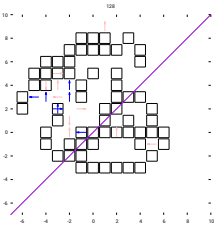
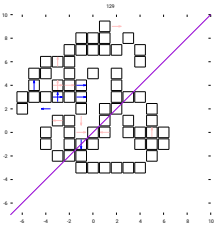
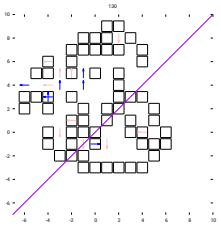
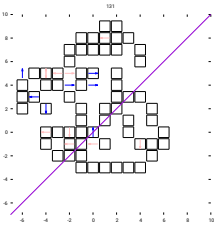
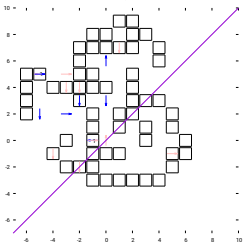
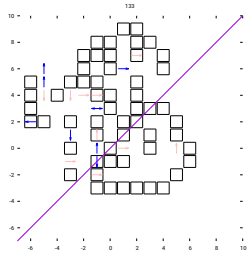
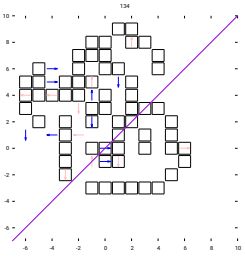
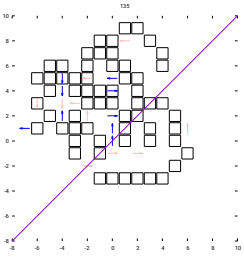



















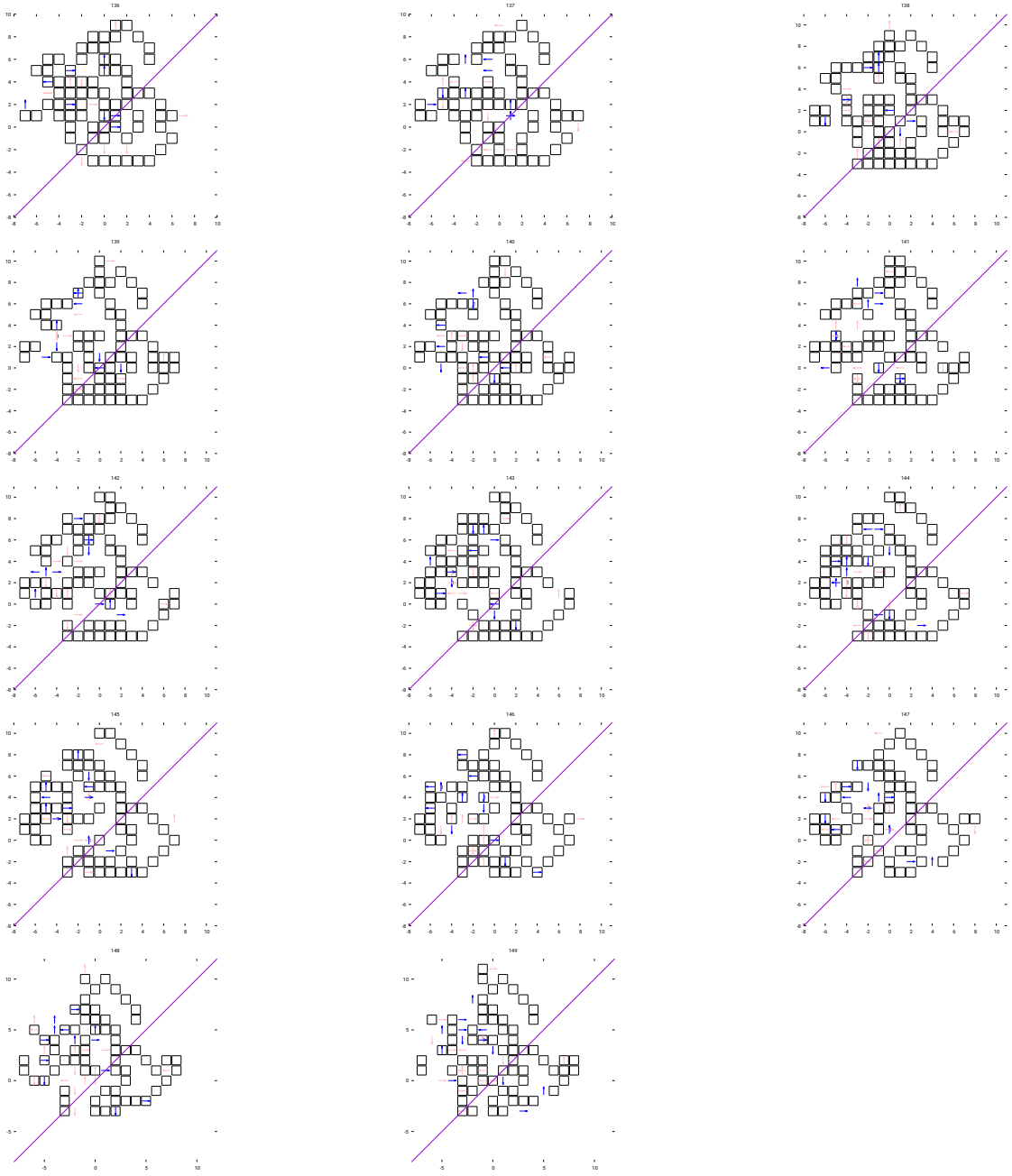
















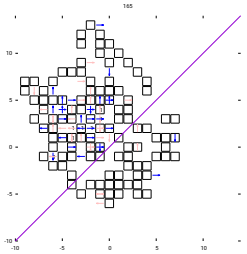
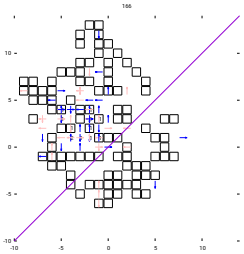
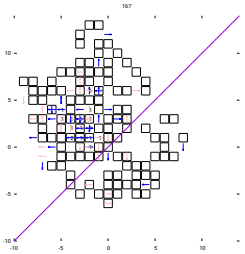
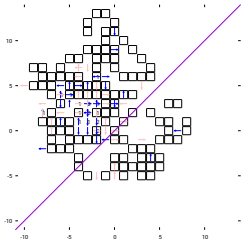
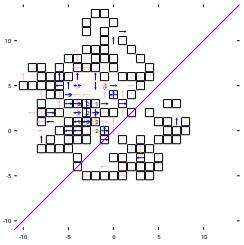
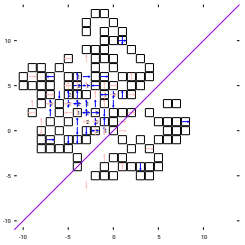
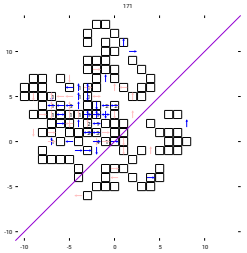
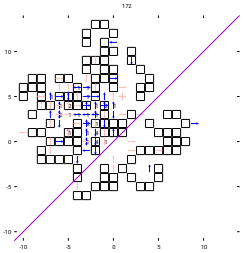
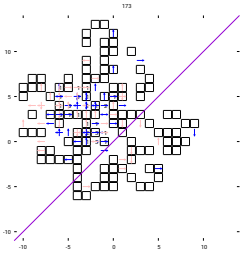
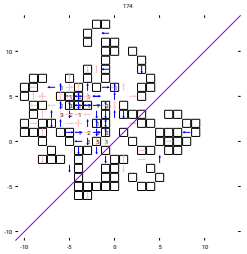
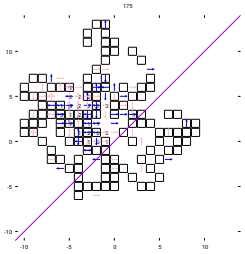
2.3. **Moves 101–150.**

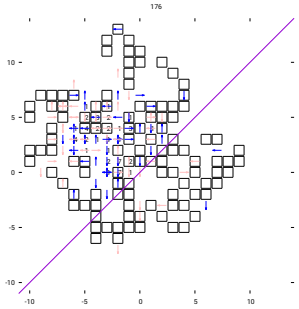
















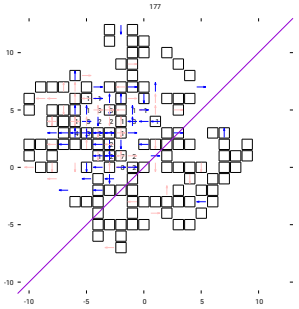





















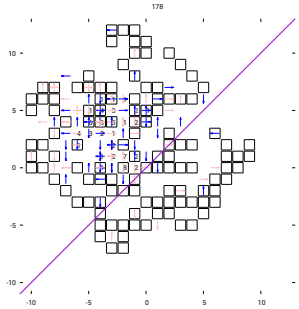
2.4. **Moves 150–175.** Near move number 160 the number of eggs slightly to the NW of the origin that cannot hatch grows rapidly, because these squares start to be almost always occupied by wandering ants.

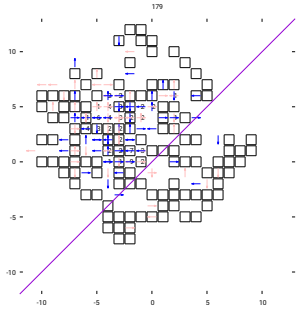












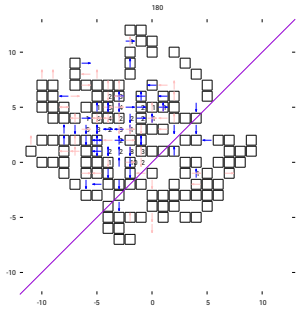


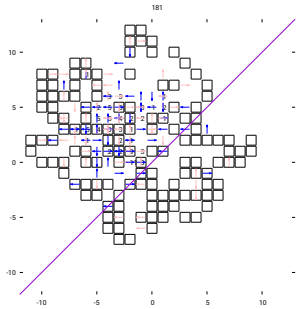


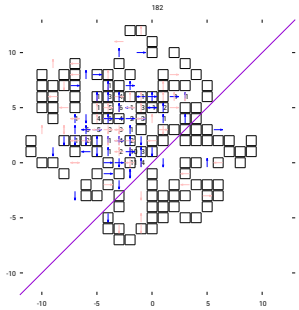
2.5. **Moves 176–199.**

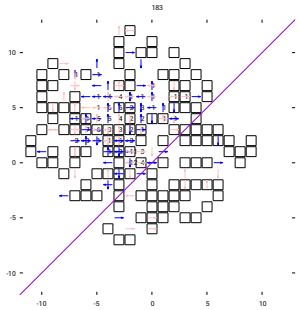


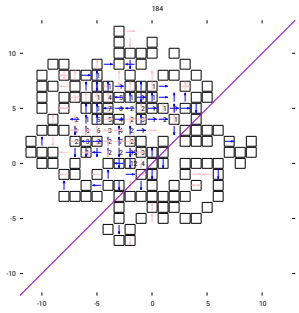


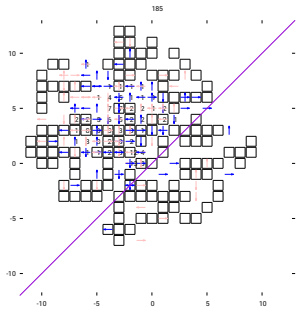


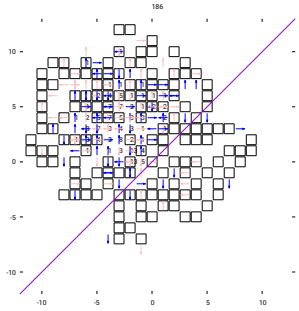


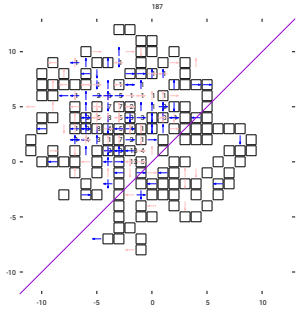


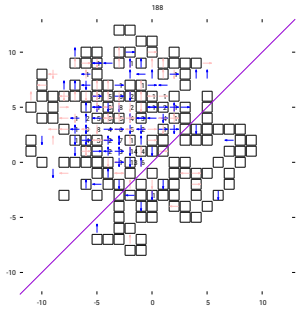


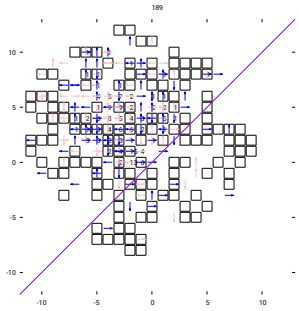

184

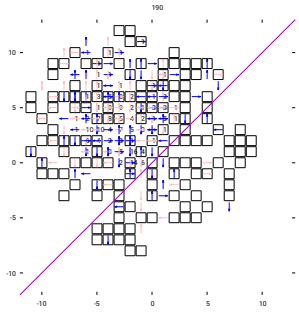

185

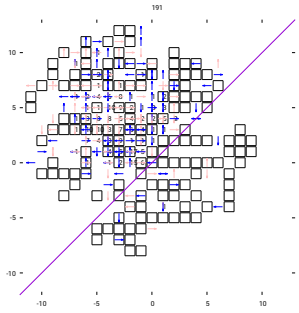

186

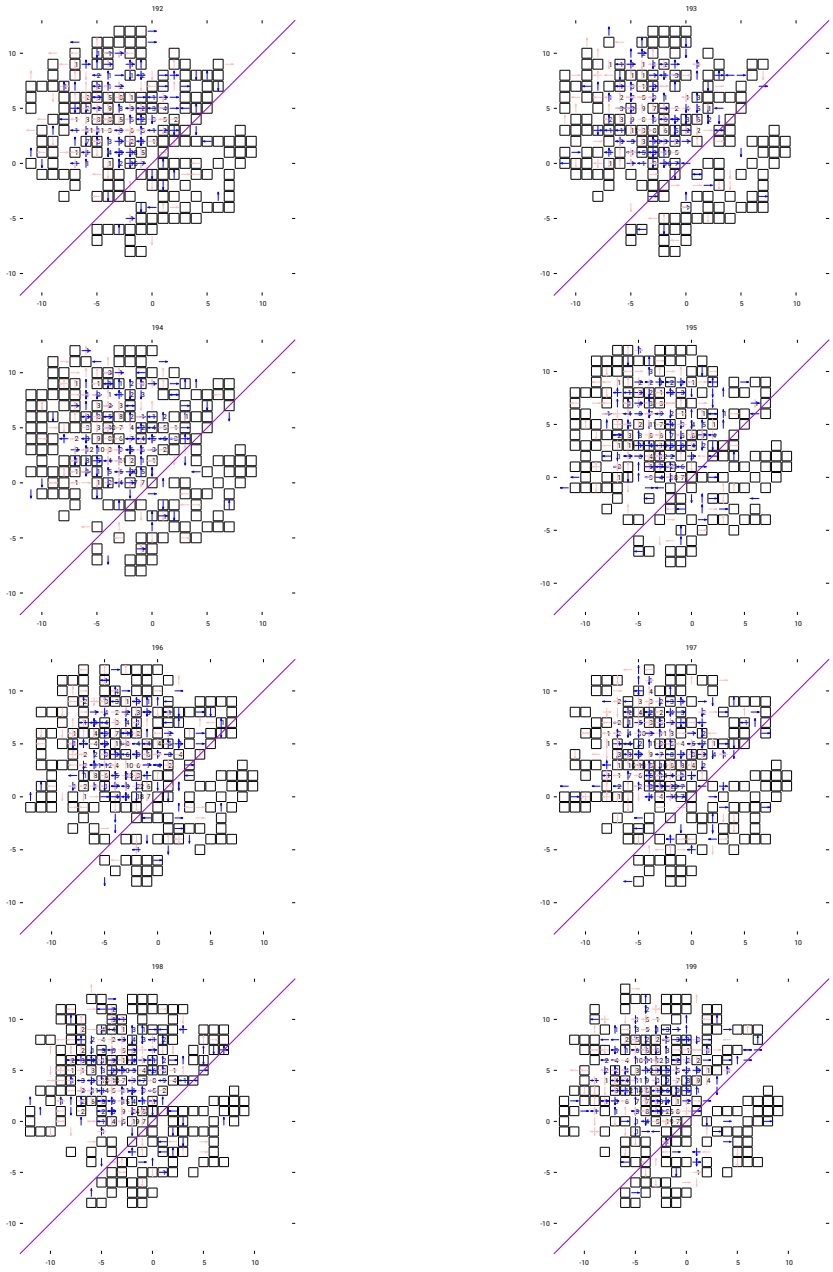

187


188


189


190


191



3. Java Program

3.1. File Ant.java.

```
public class Ant
{
    /* the x and y coordinate and color of its cell
    */
    Cell loc ;
```

```
    /* the sex as a boolean: male=true, female=false
    */
    boolean male ;

    /* direction of the current heading : 0=N, 1=E, 2=S, 3=W
    */
    int heading ;

    /* generate an ant at a specific coordinate
    * @param where The cell
    * @param isMale if true male, else female
    */
    Ant(Cell where, boolean ismale, int dir)
    {
        loc = where ;
        male = ismale ;
        /* wrap into correct mod 4 range
        */
        heading = (dir + 4)% 4 ;
    }
} /* Ant */
```

3.2. File Cell.java.

```
public class Cell
{
    /* horizontal coordinate
    */
    int x ;
    /* vertical coordinate
    */
    int y ;

    /* true of black, false if white
    */
    boolean black ;

    /* number of eggs at this position
    */
    int noEggs ;

    /* generate a cell at a Cartesian coordinate
    * @param E the x coordinate (East+)
    * @param N the y coordinate (North+)
    * @param isblack if true a black cell, else a white cell
    */
    Cell(int E, int N, boolean isblack)
    {
        x=E ;
        y=N ;
        black = isblack ;
        noEggs =0 ;
    }
```

```
    /* generate a cell at a Cartesian coordinate
    * @param E the x coordinate (East+)
    * @param N the y coordinate (North+)
    * @param isblack if true a black cell, else a white cell
    * @param Neggs numer of eggs
    */
    Cell(int E, int N, boolean isblack, int Neggs)
    {
        x=E ;
        y=N ;
        black = isblack ;
        noEggs = Neggs ;
    }

    /* compare two cells by location
    * (since location and color are the same information, comparison by color is useless here)
    */
    @Override
    public boolean equals(Object oth)
    {
        if ( ! (oth instanceof Cell) )
            return false;
        return ( ( x == ((Cell)(oth)).x ) && (y == ((Cell)(oth)).y) ) ;
    } /* equals */

} /* Cell */
```

3.3. File Colony.java.

```
/* @file Ant colony with growth as in OEIS A326166
* @author Richard J. Mathar
* @since 2023-11-24
*/
import java.util.Vector ;
import java.io.PrintWriter ;

public class Colony
{
    /* the finite list of distinc cells which are black.
    * The cells in here just represent x,y coordinates, whre
    * the eggs and black attributes may be wrong...
    */
    Vector<Cell> blackCells ;

    Vector<Ant> ants ;
    Vector<Cell> cellsWeggs ;

    /* generate the initial state of a male and a female at horizontal distance of 2
    * No black cells nor eggs yet.
    */
    Colony()
    {
        Cell W = new Cell(-1,0,false,0) ;
```

```
        Cell E = new Cell(1,0,false,0) ;
        Ant adam = new Ant(W,true,0) ;
        Ant eve = new Ant(E,false,2) ;
        ants = new Vector<Ant>() ;
        ants.add(adam) ;
        ants.add(eve) ;
        cellsWeggs = new Vector<Cell>() ;
        blackCells = new Vector<Cell>() ;
    } /* ctor */

    /* copy constructor with a list of colors, ants and eggs
    */
    Colony(Vector<Cell> blacks, Vector<Ant> a, Vector<Cell> e)
    {
        blackCells = blacks ;
        ants = a ;
        cellsWeggs= e ;
    } /* ctor */

    /** depict the color of cell x and y
    * @param x int horizontal coordinate
    * @param y int vertical coordinate
    * @return true if black
    */
    boolean isBlack(int x, int y)
    {
        /* search through the finite list of cells marked black
        */
        for( Cell b : blackCells)
        {
            if ( b.x == x && b.y ==y)
                return true ;
        }
        return false ;
    }

    /** move all ants by one step, hatch eggs, generate eggs
    * @return the state of the colony after one move of each ant
    */
    Colony move()
    {
        /* clone the current list of black cells
        */
        Vector<Cell> newblacC = new Vector(blackCells) ;

        /* the rule is that a cell changes color when
        * there were one or more ants on it, independent of
        * how many ants.
        */
        for( Ant a : ants)
        {
            if ( isBlack(a.loc.x, a.loc.y) )
            {
```

```
        /* cells which are left by ants and were black turn white,
        * i.e.,  are removed from the black list (even if
        * new ants enter)
        */
        for(int i= newblacC.size()-1 ; i >= 0 ;i--)
        {
            if ( newblacC.elementAt(i).equals(a.loc) )
                newblacC.remove(i) ;
        }
    }
    else
    {
        /* cells which are left by ants and were white turn black,
        * i.e. are added to the black list;toggling their color
        * is not done, because they are implicitly black by being
        * in the list. Because the list of ants may have more
        * than 1 ant in the same cell, we add the cell at most once.
        */
        if ( ! newblacC.contains(a.loc) )
        {
            newblacC.add(a.loc) ;
        }
    }
}


Vector<Ant> newants = new Vector<Ant>() ;
/* the existing ants move on
*/
for( Ant a: ants)
{
    Cell targcell ;
    int newdir ;

    if ( isBlack(a.loc.x,a.loc.y) )
        /* if on black squares turn 90 deg ccw, 270 deg cw
        */
        newdir = (a.heading+3) % 4 ;
    else
        /* if on black squares turn 90 deg cw
        */
        newdir = (a.heading+1) % 4 ;

    /* newx, newy = new coordinate N,E,S,W
    */
    int newx = a.loc.x ;
    int newy = a.loc.y ;
    switch (newdir)
    {
    case 0 :
        newy++ ;
        break ;
    case 1 :
```

```
            newx++ ;
            break ;
        case 2 :
            newy-- ;
            break ;
        default :
            newx-- ;
            break ;
        }

        /* new cell of the ant assumed to have no egg.
        * i.e., we never can get the number of eggs of a cell
        * by dereferencing the ant's cell
        */
        if ( isBlack(newx, newy) )
            targcell = new Cell(newx,newy,true,0) ;
        else
            targcell = new Cell(newx,newy,false,0) ;

        Ant newa = new Ant(targcell, a.male, newdir) ;
        newants.add(newa) ;
    }

    /* hatching: the cellswith eggs are reduced by one egg
    * (possibly removed from the list altogether if hatch is possible)
    * and the hatching is determined by the number of ants in
    * the newants list
    */
    Vector<Cell> newecellsl = new Vector<Cell>() ;
    for( Cell c : cellsWeggs)
    {
        boolean canhatch = true ;
        /* can only hatch if the ants (already moved!) have all left the cell.
        */
        for( Ant a : newants)
        {
            if ( a.loc.equals(c) )
                canhatch = false ;
        }

        if ( canhatch )
        {
            /* new ant, looking north, gender depnding on cell color
            * (color after the ants have left, at the time of hatching, not when laid!)
            */
            boolean leftblack =  newblacC.contains(c)  ;
            Ant newa = new Ant(c, !leftblack, 0) ;
            newants.add(newa) ;
            /* reduce number of eggs  in that cell, if down to zero
            * don't add to new eggs list
            */
            if ( c.noEggs > 1)
            {
```

```
            Cell residegs = new Cell(c.x, c.y, leftblack, c.noEggs-1) ;
            newecellsl.add (residegs) ;
        }
    }
    else
    {
        /* leave number of eggs in that cell as is */
        newecellsl.add (c) ;
    }
}

/* new/additional eggs by pairs of ants; assume that many pairs
* can create many eggs at the same time.
* Couple all pairs of distinct ants at shared location of differnt sex
* e.g. 5 males and 3 females generate 3 eggs.
*/
Vector<Cell> cellswPairs = new Vector<Cell>() ;
/* first scan: generate all cells where at least one couple
* exists
*/
for(int a1 =0 ; a1 < newants.size()-1 ; a1++)
{
    Ant ant1 = newants.elementAt(a1) ;
    for(int a2 = a1+1 ; a2 < newants.size() ; a2++)
    {
        Ant ant2 = newants.elementAt(a2) ;
        if ( ant1.loc.equals(ant2.loc) && ( ant1.male != ant2.male) )
        {
            if ( ! cellswPairs.contains(ant1.loc) )
                cellswPairs.add(ant1.loc) ;
        }
    }
}
/* second scan: in all cells count male and female before they leave
*/
for( Cell c : cellswPairs)
{
    int male =0 ;
    int fem =0 ;
    for(Ant a : newants)
    {
        if ( a.loc.equals(c) )
        {
            if ( a.male)
                male++ ;
            else
                fem++ ;
        }
    }
    int newlaid = Math.min(male,fem) ;
    /* these new eggs may or may not yet be in the newecellssl
    */
    int haveE = newecellsl.indexOf(c) ;
```

```
            if ( haveE >=0)
            {
                /* add the new eggs to a cell already in the list
                */
                newecellsl.elementAt(haveE).noEggs += newlaid ;
            }
            else
            {
                /* generate a new cell for the list and don't bother
                * to get its color right
                */
                Cell newe = new Cell(c.x, c.y, true, newlaid) ;
                newecellsl.add(newe) ;
            }
        }

        return new Colony(newblacC, newants, newecellsl) ;
    } /* move */

    /** Generate a string with line breaks suitable for a gnuplot
    * Ants are printed as arrows, blue the males, red the females
    * The number of eggs in cells are integer numbers printed in the cells
    * Black cells are in a square box, white cells are not indicated.
    * standard input .
    * @param fna The file name basis (which is "g" plus the number of the generation
    * @return a length string
    */
    String gnuplot(int fna)
    {
        /* bounding box coordinates lower left, lower right...
        */
        int bbox[] = new int[2] ;
        int bboy[] = new int[2] ;
        bbox[0]=bbox[1] =0 ;
        bboy[0]=bboy[1] =0 ;

        String rep= new String("set terminal \"pdfcairo\" font \"Roman,7\"\n") ;
        rep += "set output \"g" + fna + ".pdf\"\n" ;
        rep += "set size 0.7,1\n" ;
        rep += "set size ratio 1\n" ;
        rep += "set border 0\n" ;
        rep += "set title \"" + fna +"\"\n" ;
        // rep += "set border 15\n" ;
        //rep += "set notics\n" ;
        for(Cell c : blackCells)
        {
            /* represent black cell by 4 smaller lines as a box
            */
            rep += "set arrow from " + (c.x-0.45) +"," + (c.y-0.45)
                + " to " + (c.x+0.45) +"," + (c.y-0.45) +" nohead lw 0.3 \n";
            rep += "set arrow from " + (c.x+0.45) +"," + (c.y-0.45)
                + " to " + (c.x+0.45) +"," + (c.y+0.45) +" nohead  lw 0.3\n";
            rep += "set arrow from " + (c.x+0.45) +"," + (c.y+0.45)
```

```
        + " to " + (c.x-0.45) +"," + (c.y+0.45) +" nohead  lw 0.3\n";
    rep += "set arrow from " + (c.x-0.45) +"," + (c.y+0.45)
        + " to " + (c.x-0.45) +"," + (c.y-0.45) +" nohead  lw 0.3\n";
    bbox[0] = Math.min(c.x-1,bbox[0]) ;
    bbox[1] = Math.max(c.x+1,bbox[1]) ;
    bboy[0] = Math.min(c.y-1,bboy[0]) ;
    bboy[1] = Math.max(c.y+1,bboy[1]) ;
}

for(Ant a : ants)
{
    switch (a.heading)
    {
    case 0 :
        rep += "set arrow from " + a.loc.x +"," + (a.loc.y-0.4)
        + " to " + a.loc.x +"," + (a.loc.y+0.4) +" head lw 0.8";
        break ;
    case 1 :
        rep += "set arrow from " + (a.loc.x-0.4) +"," + a.loc.y
        + " to " + (a.loc.x+0.4) +"," + a.loc.y +" head lw 0.8";
        break ;
    case 2 :
        rep += "set arrow from " + a.loc.x +"," + (a.loc.y+0.4)
        + " to " + a.loc.x +"," + (a.loc.y-0.4) +" head lw 0.8";
        break ;
    default :
        rep += "set arrow from " + (a.loc.x+0.4) +"," + a.loc.y
        + " to " + (a.loc.x-0.4) +"," + a.loc.y +" head lw 0.8";
        break ;

    }
    if ( a.male)
        rep += " lc rgb \"blue\"" ;
    else
        rep += " lc rgb \"pink\"" ;
    rep += "\n" ;
    bbox[0] = Math.min(a.loc.x-1,bbox[0]) ;
    bbox[1] = Math.max(a.loc.x+1,bbox[1]) ;
    bboy[0] = Math.min(a.loc.y-1,bboy[0]) ;
    bboy[1] = Math.max(a.loc.y+1,bboy[1]) ;

}

int blow = Math.min(bbox[0],bboy[0]) ;
int bhigh = Math.max(bbox[1],bboy[1]) ;
for(Cell e : cellsWeggs)
{
    rep += "set label \"" + e.noEggs +"\" at " + e.x+"," + e.y + "\n";
}
rep += "set xrange [" + blow +":" + bhigh + "]\n" ;
rep += "set yrange [" + blow +":" + bhigh + "]\n" ;
rep += "plot x notitle\n" ;
return rep ;
```

```
} /* gnuplot */

String status(int verb)
{
    String rep= new String("black c ") ;
    rep += blackCells.size()  ;
    if ( verb > 0)
    {
        rep += " at " ;
        for(Cell c : blackCells)
            rep += " (" + c.x + ","+ c.y + ")" ;
    }
    rep += " ants " + ants.size() ;
    if ( verb > 0)
    {
        rep += " at " ;
        for(Ant a : ants)
        {
            rep += " (" + a.loc.x + ","+ a.loc.y + ")" ;
            if ( a.male)
                rep += "m" ;
            else
                rep += "f" ;

            switch (a.heading)
            {
            case 0 :
                rep += "N" ;
                break ;
            case 1 :
                rep += "E" ;
                break ;
            case 2 :
                rep += "S" ;
                break ;
            default :
                rep += "W" ;
                break ;

            }

        }
    }
    rep += " cells w Eggs " + cellsWeggs.size() ;
    if ( verb > 0)
    {
        rep += " at " ;
        for(Cell e : cellsWeggs)
        {
            rep += " (" + e.x + ","+ e.y + ")" ;
            rep += e.noEggs ;

        }
```

```
    }
    return rep ;

} /* status */

/** Start with a pair of ants and run through the moves
* javac -cp . *.java
* Start with a pair and move them (one move per ant)
* java -cp . Colony [-n Nmoves] [-v] [-g]
*   Switch -n followed by the number of moves (default 20)
*   Switch -v means that the ants positions, eggs positions, lists of black squares are pritned
*   Switch -g means that gnuplot load files are genrated.
*
* Generate b326166.txt:
* java -cp . Colony -n 200 -v 0 |& grep -F "gen" | awk '{print $2,$7}'
*
* Generate illustrations in files g0.pdf, g1.pdf etc:
* java -cp . Colony -n 200 -v 0 -g
* for i in {0..199} ; do gnuplot < g${i}.gp ; done
*/
public static void main(String args[])
{
    int nMoves= 20 ;
    int verb = 0 ;
    boolean gnuplot =false ;

    for(int optind=0 ; optind < args.length ; optind++)
    {
        if ( args[optind].equals("-n") )
        {
            nMoves = Integer.parseInt(args[optind+1]) ;
            optind++ ;
        }
        else if ( args[optind].equals("-v") )
        {
            verb = Integer.parseInt(args[optind+1]) ;
            optind++ ;
        }
        else if ( args[optind].equals("-g") )
        {
            gnuplot =true ;
        }
    }

    Colony mose = new Colony() ;
    for(int gen =0 ; gen < nMoves ; gen++)
    {
        System.out.println("gen " + gen + " "+ mose.status(verb)) ;
        if ( gnuplot )
        {
            String fna = "g"+gen ;
            String gp=mose.gnuplot(gen) ;
            try
```

```
                {
                    PrintWriter f = new PrintWriter(fna+".gp") ;
                    f.print(gp) ;
                    f.close() ;
                }
                catch(Exception ex)
                {
                }
            }
            mose = mose.move() ;
        }
    }
} /* Colonony */
```

References

1. O. E. I. S. Foundation Inc., *The On-Line Encyclopedia Of Integer Sequences*, (2023), https://oeis.org/. MR 3822822

URL: https://www.mpia-hd.mpg.de/homes/mathar

Hoeschstr. 7, 52372 Kreuzau, Germany