Project Report

เรื่อง

Rate in maize

เสนอ

รศ.ดร.รังสิพรรณ มฤคทัต

จัดทำโดย

ณัฐมน ว่องไววุฒิกุลเดช 6513165

นพรุจ ฤทธิ์เนติกุล 6513168

นิติวดี ลิมปยารยะ 6513169

แองเจลิน่า ชัยนิธิกรรณ 6513178

รายงานนี้เป็นส่วนหนึ่งของวิชา Structure and Algorithmภาคเรียนที่ 2 ปีการศึกษา 2565

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของวิชา Data Structure and Algorithm (EGCO 221) จัดทำขึ้นเพื่อใช้ประกอบอธิบายการทำงานโปรแกรมของ Rate in maize ประกอบไปด้วยส่วนของคู่มือการใช้งานโปรแกรมเบื้องต้น การสาธิตและการอธิบายในส่วนของ Algorithm รวมไปถึงข้อจำกัดต่างๆ ในการใช้งานโปรแกรม

ทางคณะผู้จัดทำขอขอบพระคุณ รศ.ดร.รังสิพรรณ มฤคทัต ผู้ให้ความรู้ และแนวทางการศึกษา สุดท้ายนี้ทางคณะผู้จัดทำหวังว่ารายงานฉบับนี้จะสามารถเป็นประโยชน์ไม่มากก็น้อยแก่ผู้อ่านทุกท่าน

ขอขอบพระคุณ

คณะผู้จัดทำ

สารบัญ

คู่มือการใช้งานโปรแกรม

```
--- exec:3.1.0:exec (default-cli) @ excercisecom1 ---
Please enter a new file name: maize_01.txt
File maize_01.txt is not found. Please enter a new file name: maize_1.txt
```

1.ให้ผู้ใช้งานป้อนชื่อไฟล์ที่จะใช้งานผ่าน keyboard

2. เมื่อป้อนชื่อไฟล์ที่ต้องการเรียบร้อยแล้วให้กดปุ่ม Enter

หมายเหตุ : หากป้อนชื่อไฟล์ไม่ตรงกับไฟล์ที่มีจะไม่สามารถเปิดได้ ต้องทำการพิมพ์ชื่อไฟล์ใหม่อีกครั้ง

	col_0	col_1	col_2	col_3	col_4
row_0	0	1	1	1	F
row_1	1	R	1	0	1
row_2	0	1	0	F	1
row_3	1	1	1	0	1
row_4	0	0	1	1	1

3. โปรแกรมจะแสดงรูปแบบของเขาวงกตในไฟล์ที่ผู้ใช้ทำการป้อนชื่อไฟล์มา โดยรูปแบบของเขาวงกตจะมี row คือแถวในแนวนอน และ column คือหลักในแนวตั้ง สิ่งที่อยู่ในเขาวงกตจะประกอบไปด้วย

- เลข 0 คือกำแพงที่หนูจะไม่สามารถเดินทางผ่านไปได้
- เลข 1 คือทางเดินที่หนูสามารถเดินทางผ่านไปได้
- ตัวอักษร R คือตัวหนู
- ตัวอักษร F คืออาหาร เป้าหมายของหนู เมื่อหนูสามารถเก็บอาหารในเขาวงกตได้ครบถือว่าจบเกมส์

```
============================HOW TO PLAY============================
--Key Inputs--
--Can be lowercase letter or uppercase letter
U = move rat up 1 row(from current position)
D = move rat down 1 row(from current position)
L = move rat left 1 column(from current position)
L = move rat right 1 column(from current position)
A = use auto mode to find all food in maze
E = exit
O = restart
===========================================================

User input >>( U D L R ) key to move
           >>(A) key to auto mode
           >>(E) key to exit
           >>(O) key to restart
```

4. นอกจากโปรแกรมจะแสดงรูปแบบของเขาวงกตในไฟล์ที่ผู้ใช้ทำการป้อนชื่อไฟล์มา โปรแกรมจะแสดงรูปแบบตัวอักษรในการออกคำสั่งต่างๆ เพื่อให้ผู้ใช้ได้เลือกใช้งานตามความต้องการโดยสามารถพิมพ์ได้ทั้งตัวอักษรพิมพ์เล็กและพิมพ์ใหญ่ ดังนี้

- พิมพ์ตัวอักษร U เมื่อต้องการขยับตัวหนูขึ้นด้านบน(column ก่อนหน้า)
- พิมพ์ตัวอักษร D เมื่อต้องการขยับตัวหนูลงด้านล่าง(column ถัดไป)
- พิมพ์ตัวอักษร L เมื่อต้องการขยับตัวหนูไปทางด้านซ้าย(row ก่อนหน้า)
- พิมพ์ตัวอักษร R เมื่อต้องการขยับตัวหนูไปทางด้านขวา(row ถัดไป)
- พิมพ์ตัวอักษร A เมื่อต้องการใช้งาน Auto mode เพื่อให้หนูค้นหาอาหารทั้งหมดอัตโนมัติ
- พิมพ์ตัวอักษร E เพื่อจบการทำงาน
- พิมพ์ตัวอักษร O เพื่อเริ่มต้นเกมส์ใหม่อีกครั้ง

5. เมื่อป้อนคำสั่งที่ต้องการเรียบร้อยแล้วให้กดปุ่ม Enter

```
Game End. Do You want to play new game?(Y/N)
```

6.เมื่อผู้ใช้ชนะเกมส์ หรือสามารถหาอาหารได้ครบจำนวน โปรแกรมจะแสดงผลว่า Game End และถามว่าต้องการที่จะเล่นอีกหรือไม่ โดยจะรับค่าคำสั่งเป็นแค่ Y (y) และ N (n)

7.เมื่อป้อนคำสั่งที่ต้องการเรียบร้อยแล้วให้กดปุ่ม Enter

8.ถ้าผู้ใช้ป้อน Y โปรแกรมจะถามชื่อไฟล์อีกครั้งเหมือนในข้อที่1 แต่ ผู้ใช้ป้อน N โปรแกรมจะจบการทำงานทันที

ตัวอย่างการทำงานของโปรแกรมเมื่อพิมพ์คำสั่ง

```
User input >>( U D L R ) key to move
           >>(A) key to auto mode
           >>(E) key to exit
           >>(O) key to restart
u
        col_0  col_1  col_2  col_3  col_4
row_0       0      R      1      1      F
row_1       1      1      1      0      1
row_2       0      1      0      F      1
row_3       1      1      1      0      1
row_4       0      0      1      1      1

User input >>( U D L R ) key to move
           >>(A) key to auto mode
           >>(E) key to exit
           >>(O) key to restart
```

เมื่อผู้ใช้ทำการพิมพ์ตัวอักษร u โปรแกรมจะแสดงผลเป็นรูปแบบของเขาวงกตที่ตัวหนูมีการขยันขึ้นไป1 ตำแหน่ง และจะทำกการแสดงคำสั่งทั้งหมด พร้อมกับรับคำสั่งใหม่ทันที

```
User input >>( U D L R ) key to move
           >>(A) key to auto mode
           >>(E) key to exit
           >>(O) key to restart
u
Can't move
```

เมื่อผู้ใช้ทำการพิมพ์ตัวอักษร u อีกครั้ง จะเห็นได้ว่าไม่มี row ด้านบนตัวหนูเหลือแล้วทำให้ไม่สามารถขยับได้ โปรแกรมจะแสดงผลลัพธ์เป็น Can’t move

```
User input >>( U D L R ) key to move
          >>(A) key to auto mode
          >>(E) key to exit
          >>(O) key to restart
A
```

```
====================Finding Food 1====================
++++++++Food is eaten++++++++
        col_0  col_1  col_2  col_3  col_4
row_0       0      1      1      1      R
row_1       1      1      1      0      1
row_2       0      1      0      F      1
row_3       1      1      1      0      1
row_4       0      0      1      1      1

--------------------PATH FROM START--------------------
 start -> (row  0 ,col  1, R)
 right -> (row  0 ,col  2, 1)
 right -> (row  0 ,col  3, 1)
 right -> (row  0 ,col  4, F)
-------------------------------------------------------



====================Finding Food 2====================
++++++++Food is eaten++++++++
        col_0  col_1  col_2  col_3  col_4
row_0       0      1      1      1      1
row_1       1      1      1      0      1
row_2       0      1      0      R      1
row_3       1      1      1      0      1
row_4       0      0      1      1      1

--------------------PATH FROM START--------------------
 start -> (row  0 ,col  4, R)
  left -> (row  0 ,col  3, 1)
  left -> (row  0 ,col  2, 1)
  left -> (row  0 ,col  1, 1)
  down -> (row  1 ,col  1, 1)
  down -> (row  2 ,col  1, 1)
  down -> (row  3 ,col  1, 1)
 right -> (row  3 ,col  2, 1)
  down -> (row  4 ,col  2, 1)
 right -> (row  4 ,col  3, 1)
 right -> (row  4 ,col  4, 1)
    up -> (row  3 ,col  4, 1)
    up -> (row  2 ,col  4, 1)
  left -> (row  2 ,col  3, F)
-------------------------------------------------------
```

เมื่อผู้ใช้ทำการพิมพ์ตัวอักษร A จะแสดงผลลัพธ์จาการเดินทางของหนูไปสู่อาหารทั้งหมด(หาอาหารครั้งละหนึ่งตำแหน่ง)ในเขาวงกต โดยจะแสดงรูปแบบเขาวงกตล่าสุดเมื่ออาหารถูกหนูกินไปแล้ว และหนูจะไปอยู่ตำแหน่งของอาหารอันที่ถูกหนูกินไป รวมถึงแสดงผลว่าหนูเดินผ่านเส้นทางไหนบ้าง ซ้าย,ขวา,บน,ล่าง รวมไปถึงระบุ row และ column นั้นๆที่หนูเดินผ่าน ในกรณีตัวอย่างมีอาหารทั้งหมดสองตำแหน่งเมื่อหนูเสร็จสิ้นการหาอาหารตำแหน่งที่หนึ่งและแสดงผลลัพธ์ หนูจะไปหาอาหารอันที่สองต่อแสดงผลลัพธ์การหาอาหารในตำแหน่งที่สอง

```
User input >>( U D L R ) key to move
           >>(A) key to auto mode
           >>(E) key to exit
           >>(O) key to restart
E
------------------------------------------------------------------------
BUILD SUCCESS
------------------------------------------------------------------------
```

เมื่อผู้ใช้ทำการพิมพ์ตัวอักษร E โปรแกรมจะจบการทำงานทันที

```
User input >>( U D L R ) key to move
           >>(A) key to auto mode
           >>(E) key to exit
           >>(O) key to restart
o
Please enter a new file name:
```

เมื่อผู้ใช้ทำการพิมพ์ตัวอักษร O โปรแกรมจะเริ่มต้นใหม่โดยให้ผู้ใช้งานป้อนชื่อไฟล์ที่ต้องการใหม่อีกครั้ง

```
User input >>( U D L R ) key to move
           >>(A) key to auto mode
           >>(E) key to exit
           >>(O) key to restart
g
Incorrect input!!
```

เมื่อผู้ใช้ทำการพิมพ์ตัวอักษรนอกเหนือจากที่ระบุไว้ ตัวโปรแกรมจะแจ้งและให้พิมพ์คำสั่งใหม่อีกครั้ง

Data structures + classes

Class RatInMaze

```
12  public class RatInMaze {
13
14      private static final String FILE_NAME = null;
15      private static final String PATH = "src/main/Java/Project1/";
16
17      public static void main(String[] args) {...202 lines }
219
220 }
```

Class RatInMaze เป็น main class ของโปรแกรมประกอบไปด้วย การอ่านข้อมูลจากไฟล์เพื่อนำมาสร้าง maze โดยมีประกอบเป็นขนาดของ row และ column รวมไปถึงตำแหน่งของหนูและอาหาร ซึ่งข้อมูลใน maze เหล่านี้จะถูกเก็บข้อมูลในรูปแบบของ 2D Array อาหารนั้นจะถูกเก็บข้อมูลทั้งจำนวนและตำแหน่งและเก็บใน Class Food ในรูปแบบของ Array เพื่อในข้อมูลจำนวนของอาหารที่มีไปใช้ต่อ มีการพิมพ์กติกาการเล่นและวิธีเล่นให้ user ทั้งคลาสนี้ยังเป็นส่วนที่รับ input จาก user เพื่อเลื่อนตำแหน่งของหนูหรือใช้คำสั่งที่กำหนดไว้ เช่น U (ขึ้น), D (ลง), L (ซ้าย), R (ขวา), A (Auto), E (ออก), O (เริ่มเกมใหม่)

- **มีการใช้ Array**

Array คือ โครงสร้างข้อมูลแบบเชิงเส้นที่ใช้เก็บข้อมูลประเภทเดียวกันแบบเป็นลำดับค่าที่เก็บใน Array ได้แก่

Object ของ Food: แต่ละ element ของ array เก็บ object ของ Class Food

ข้อมูลตำแหน่ง: แต่ละอ็อบเจกต์ Food มีข้อมูลเกี่ยวกับตำแหน่งของอาหารภายใน maze

ข้อมูลสถานะ: แต่ละอ็อบเจกต์ Food ยังมี Attributes boolean เพื่อระบุว่าอาหารถูกกินหรือยัง

Class Food

```
693    class Food {
694
695        private int col, row;
           private boolean is_alive = true;
697
698        public Food(int r, int c) {...4 lines }
702
703        public int getfrow() {...3 lines }
706
707        public int getfcol() {...3 lines }
710
711        public void setcol(int x) {...3 lines }
714
715        public void setrow(int x) {...3 lines }
718
719        public void eat() {...3 lines }
722
723    }
```

การออกแบบคลาส Food มีลักษณะดังนี้

Attributes:

row: จำนวนเต็มที่แทนตำแหน่งแถว (row) ของอาหารใน maze

col: จำนวนเต็มที่แทนตำแหน่งหลัก (column) ของอาหารใน maze

is_alive: ตัวแปร boolean ที่ระบุว่าอาหารยังไม่ถูกกิน (true) หรือถูกกินแล้ว (false)

Constructor: กำหนดค่าให้กับ Attributes row และ col ตามพารามิเตอร์ที่ให้มา

Methods:

getfrow( ): คืนค่าตำแหน่งแถว (row) ของอาหาร

getfcol( ): คืนค่าตำแหน่งหลัก (column) ของอาหาร

eat( ): กำหนดว่าอาหารถูกกิน โดยตั้งค่า is_alive เป็น false

เหตุผลที่เลือกใช้ Array เพราะการใช้ Array ทำให้ง่ายต่อการจัดเก็บข้อมูลของ object หลายตัวในโครงสร้างข้อมูลเดียวกัน ทำให้การจัดการข้อมูลเป็นไปอย่างเรียบง่าย

- **มีการใช้ 2D Array**

2D Array คือ โครงสร้างข้อมูลที่ใช้เก็บข้อมูลแบบตารางสองมิติ ประกอบด้วยแถว (row) และหลัก (column)

ซึ่งมีตัวแปรที่ใช้ในการเก็บค่าคือ myarray, temp_array, และ B ซึ่งเก็บค่าดังนี้

myarray: เป็นตารางหลักที่เก็บข้อมูลเกี่ยวกับสถานะของแผนที่ในเกม โดยแต่ละช่องจะมีค่าที่ระบุสถานะต่าง ๆ คือ 0 แทนกำแพงหรือทางเดินที่ไม่สามารถเดินผ่านได้, 1 แทนทางเดินที่สามารถเดินผ่านได้, 3 แทนตำแหน่งอาหาร, 5 แทนตำแหน่งปัจจุบันของหนู

temp_array: เป็นตัวแปรที่ใช้ในการจำลองหรือเก็บค่าแบบชั่วคราวของ myarray เพื่อใช้ในการคำนวณหรือการดำเนินการที่ไม่ต้องการเปลี่ยนแปลงค่าใน myarray โดยทั่วไปจะใช้ในการค้นหาเส้นทางหรือการประมวลผลเกี่ยวกับแผนที่

B: เป็นตัวแปรที่ใช้เก็บค่าเริ่มต้นของแผนที่จากไฟล์ข้อมูลที่ถูกโหลดเข้ามา โดยจะถูกใช้ในการสร้าง myarray และ temp_array โดยมีจุดประสงค์เพื่อเตรียมข้อมูลสำหรับการใช้งานในโปรแกรมหลักต่อไป

เหตุผลที่เลือกใช้ 2D Array เพราะ

2D Array ช่วยให้สามารถเก็บข้อมูลของตารางได้อย่างสะดวกและเรียกใช้ข้อมูลในตารางได้ง่าย โดยใช้คู่ index (row, column) เพื่อเข้าถึงข้อมูลในตาราง และช่วยให้โปรแกรมสามารถจำลองสภาพแวดล้อมในเกมได้ โดยใช้ค่าต่าง ๆ ในแต่ละช่องของ Array เพื่อแทนสถานะของแต่ละส่วนของแผนที่ เช่น ช่องที่หนูสามารถเดินผ่านได้ หรือตำแหน่งของอาหารที่ต้องหา หรือตำแหน่งปัจจุบันของหนูในสภาพแวดล้อม

Class Arraytmatrix

```
223  class Arraytmatrix {
224
225      private int[][] myarray;
226      private int[][] temp_array;
227      private int ratrow;
228      private int ratcol;
229      private int tratrow;
230      private int tratcol;
231      private boolean move = false;
232      private boolean END = false;
         private boolean sol = false;
234      private int eatenfood = 0;
235      private boolean found = false;
236      int[][] tarray = null;
         private Food[] myfood;
238
239      public Arraytmatrix(int[][] arg, Food[] f, int[][] t, int rowr, int colr) {...7 lines }
246
247      public void auto(int rowr, int colr, Food[] f) {...217 lines }
464
465      public int[][] up(int rowr, int colr, Food[] f) {...40 lines }
505
506      public int[][] down(int rowr, int colr, Food[] f) {...39 lines }
545
546      public int[][] left(int rowr, int colr, Food[] f) {...40 lines }
586
587      public int[][] right(int rowr, int colr, Food[] f) {...41 lines }
628
629      public void printMat() {...23 lines }
652
653      public void printMattemp() {...22 lines }
675
676      public int getrow() {...3 lines }
679
680      public boolean getend() {...3 lines }
683
684      public int getcol() {...3 lines }
687
688      public boolean getmove() {...3 lines }
691  }
```

Class Arraytmatrix เป็น object ที่เก็บค่า maze จาก main ที่อ่านค่ามาได้ดังที่กล่าวไป และประกอบไปด้วยmethod เพื่อควบคุมหนูให้เป็นไปตามที่ user ต้องการ ในโหมดปกตินั้นประกอบไปด้วย up( ), down( ), left( ), right( ) ซึ่งควบคุมการขยับของหนูให้เป็นไปตามทิศทางที่ user ใส่เข้ามา โดยทำงานร่วมกับ getrow( ), getend( ), getcol( ), getmove( ) เพื่อส่งข้อมูลนั้น ๆ ใน maze นำไปใช้งานในการอัพเดตตำแหน่งของหนู และยังมี printMat( ) เพื่อพิมพ์ maze ล่าสุดออกมาให้ดูว่าหน้าตาเป็นอย่างไร

ส่วนในโหมด autoนั้น ประกอบไปด้วย auto( ) ทำการค้นหาตำแหน่งของอาหารและวิธีการเดินของหนูเพื่อไปหาอาหารเหล่านั้นโดยไม่ต้องอาศัยการช่วยเหลือจาก user มีการใช้ ArrayDeque เพื่อจัดเก็บตำแหน่งและทิศทางที่เดินไปยังอาหารได้ โดยใช้ทั้งหมด 2 ArrayDeque ArrayDeque ตัวแรกใช้เก็บเส้นทางที่หนูเดิน ส่วน ArrayDeque ตัวที่ 2 ทำงานจากการที่เช็คไปทั้ง 4 ทิศว่าสามารเดินไปทางไหนได้บ้างหากเจอว่ามีทางเดินให้เลือกมากกว่าหนึ่งทางจะทำการบันทึกตำแหน่งปัจจุบันไว้เพื่อใช้ในการถอยกลับในกรณีที่ทางไม่สามารถไปต่อได้และไม่พบอาหาร และทำงานร่วมกับ printMattemp( ) ในการพิมพ์ผลลัพธ์สุดท้ายของการหาอาหารแต่ละตำแหน่ง

- **มีการใช้ ArrayDeque**

ArrayDeque คือโครงสร้างข้อมูลแบบ Deque (Double Ended Queue) ซึ่งหมายความว่าสามารถเพิ่มหรือลบข้อมูลได้ทั้งจากด้านหน้าและด้านหลังของคิว โดยใช้ array ที่ปรับขนาดได้ (resizable array) ในการเก็บข้อมูล

ในโปรเจคนี้ใช้ ArrayDeque ในการเก็บตัวแปรประเภท ‘One’ โดย method ที่นำมาใช้ ได้แก่

1. addFirst( ): ใช้ในการเพิ่มตำแหน่งของหนูลงใน ArrayDeque เมื่อมีการเริ่มต้นค้นหาอาหารใหม่ โดยเพิ่มตำแหน่งที่หนูอยู่ในขณะนั้น (ตำแหน่งเริ่มต้น) ลงไปใน ArrayDeque ที่ชื่อ dq โดยใช้ addFirst(new One(tratrow,tratcol,5,"start")) โดยที่ One เป็นชื่อคลาสที่ใช้เก็บข้อมูลของตำแหน่ง และ 5 แสดงถึงประเภทของตำแหน่งที่เป็นตำแหน่งเริ่มต้นของ Rat

2. add( ): ใช้ในการเพิ่มตำแหน่งหนูที่สามารถเคลื่อนที่ไปได้ใน ArrayDeque โดยใช้ dq.add(new One(ratrow,ratcol,1,"down")) เมื่อหนูสามารถเคลื่อนที่ไปได้ในทิศทางด้านล่าง (down) โดยที่ 1 แสดงถึงประเภทของตำแหน่งที่หนูสามารถเคลื่อนที่ไปได้

3. pollLast( ): ใช้ในการดึงและลบตำแหน่งที่หนูจะถูกนำไปใช้ต่อในการเคลื่อนที่ออกจาก ArrayDeque โดยใช้ temp_mark = dq_mark.pollLast( ) เมื่อตำแหน่งนั้นไม่สามารถเคลื่อนที่ไปในทิศทางใดได้ เพื่อให้หนูสามารถย้อนกลับไปยังตำแหน่งก่อนหน้านี้ที่เคยผ่านได้

4. peekLast( ): ใช้ในการดูตำแหน่งที่หนูจะเคลื่อนที่ไปต่อไปซึ่งยังไม่ลบออกจาก ArrayDeque โดยใช้ temp_dq = dq.peekLast( ) เพื่อดูตำแหน่งที่หนูจะไปต่อไปหลังจากเคลื่อนที่ในทิศทางต่างๆ

เหตุผลที่เลือกใช้ ArrayDeque เพราะ

- ArrayDeque มีความสามารถในการเพิ่มและลบข้อมูลจากด้านหน้าและด้านหลัง ซึ่งเหมาะสำหรับการใช้งานในสถานการณ์ที่ต้องการจัดการกับข้อมูลเป็นลำดับเมื่อมีการเพิ่มหรือลบตำแหน่งไปมาเรื่อย ๆ เช่น การเคลื่อนที่ของหนูในการค้นหาอาหาร และการ Backtracking

Class One

```
725     class One {
726
727         private int col, row;
            private String direc;
            private int type;
730
731         public One(int r, int c, int t, String d) {...6 lines }
737
738         public int gettype() {...3 lines }
741
742         public String getdirect() {...3 lines }
745
746         public int getfrow() {...3 lines }
749
750         public int getfcol() {...3 lines }
753
754         public void setcol(int x) {...3 lines }
757
758         public void setrow(int x) {...3 lines }
761
            public boolean equals(Object o) {...10 lines }
772
773     }
```

Class One ถูกใช้เพื่อเป็นคลาสที่มีหน้าที่จัดการกับข้อมูลของตำแหน่งในตาราง maze ที่เราต้องการจะบันทึกตำแหน่งไว้เมื่อตำแหน่งนั้นสามารถเลือกเดินได้หลายทางโดยเก็บข้อมูลคือ พิกัดที่มีแถวและหลัก รวมถึงประเภทของตำแหน่งนั้นว่าเป็นทางเดิน กำแพง หรืออาหาร และทิศทางที่สามารถเดินไปได้

Algorithm

```
if (ratrow - 1 < 0) {
    move = false;
} else {
    if (myarray[ratrow - 1][ratcol] == 1) {
        move = true;
    } else if (myarray[ratrow - 1][ratcol] == 3) {
        move = true;
    } else {
        move = false;
    }
}
if (move == true) {
    path++;//System.out.printf("path1 %d\n",path);
}
if (ratrow + 1 >= myarray.length) {
    move = false;
} else {
    if (myarray[ratrow + 1][ratcol] == 1) {
        move = true;
    } else if (myarray[ratrow + 1][ratcol] == 3) {
        move = true;
    } else {
        move = false;
    }
}
if (move == true) {
    path++;//System.out.printf("path2 %d\n",path);
}
if (ratcol - 1 < 0) {
    move = false;
} else {
    if (myarray[ratrow][ratcol - 1] == 1) {
        move = true;
    } else if (myarray[ratrow][ratcol - 1] == 3) {
        move = true;
    } else {
        move = false;
    }
}
if (move == true) {
    path++;//System.out.printf("path3 %d\n",path);
}

if (ratcol + 1 >= myarray[1].length) {
    move = false;
} else {
    if (myarray[ratrow][ratcol + 1] == 1) {
        move = true;
    } else if (myarray[ratrow][ratcol + 1] == 3) {
        move = true;
    } else {
        move = false;
    }
}
if (move == true) {
    path++;    }
```

```
        left(..., ratrow, ..., ratcol, f);
        if (move == false) {
            up(..., ratrow, ..., ratcol, f);
            if (move == false) {
                right(..., ratrow, ..., ratcol, f);
                if (move == false) {
                    down(..., ratrow, ..., ratcol, f);
                    if (move == false) {
                        myarray[ratrow][ratcol] = 0;// back set R to zero
                        //back here

                        temp_mark = dq_mark.peekLast();
                        if (temp_mark != null)//deque mark condition
                        {
                            ratrow = temp_mark.getfrow();
                            ratcol = temp_mark.getfcol();
                            //System.out.printf("pollmark row %d col %d\n",temp_mar
                        }
                        temp_mark = dq_mark.pollLast();
                        mark--;
                        //System.out.printf("mark %d\n",mark);
                        if (mark < 0) {
                            System.out.printf(..., "Food not found(No solution)\n")
                            eatenfood = f.length;
                            END = true;
                            break;
                        }

                        do {
                            temp_dq = dq.peekLast();
                            if (!temp_dq.equals(... temp_mark)) {
                                temp_dq = dq.pollLast();
                            }

                            if (temp_dq != null) {
                                // System.out.printf("polldq row %d col %d\n",temp_
                            }
                        } while (temp_dq != null && !temp_dq.equals(... temp_mark));

                    } else {
                        myarray[ratrow - 1][ratcol] = 0;
                        dq.add(new One(... ratrow, ... ratcol, ... 1, ... "down"));/*System
                    }
                } else {
                    myarray[ratrow][ratcol - 1] = 0;
                    dq.add(new One(... ratrow, ... ratcol, ... 1, ... "right"));/*System.ou
                }
            } else {
                myarray[ratrow + 1][ratcol] = 0;
                dq.add(new One(... ratrow, ... ratcol, ... 1, ... "up"));/*System.out.print
            }
        } else {
            myarray[ratrow][ratcol + 1] = 0;
            dq.add(new One(... ratrow, ... ratcol, ... 1, ... "left"));/*System.out.printf(
        }
```

Forwarding step: ลำดับในการหาเดินทางไปหาอาหารของหนู? และใช้อัลกอริทึมอะไรในการทำแต่ละขั้นตอน

โปรแกรมจะใช้อัลกอริทึม Depth-First Search (DFS) ในการค้นหาเส้นทางไปยังอาหารแต่ละชิ้น โดยที่หนูจะเคลื่อนที่ไปยังทิศทางต่างๆ ได้แก่ ซ้าย (Left), ขึ้น (Up), ขวา (Right) ,ลง (Down)ซึ่งจะไปเรื่อย ๆ จนกว่าจะเจออาหาร หรือไม่สามารถเคลื่อนที่ได้ต่อไปแล้ว แล้วจึงจะเริ่มการค้นหาใหม่โดยเริ่มจากตำแหน่งปัจจุบันของหนู ดังนั้นลำดับขั้นตอนในการหาเดินทางไปหาอาหารของหนูจะเป็นดังนี้:

1. ตรวจสอบว่าหนูสามารถเคลื่อนที่ไปยังทิศทางข้างซ้าย (Left) ได้หรือไม่ หากสามารถเคลื่อนที่ไปได้ ให้เลือกเคลื่อนที่ซ้าย (Left) และตรวจสอบตำแหน่งใหม่ของหนู

2.ตรวจสอบว่าหนูสามารถเคลื่อนที่ไปยังทิศทางข้างบน (Up) ได้หรือไม่ หากสามารถเคลื่อนที่ไปได้ ให้เลือกเคลื่อนที่ขึ้น (Up) และตรวจสอบตำแหน่งใหม่ของหนู

3.ตรวจสอบว่าหนูสามารถเคลื่อนที่ไปยังทิศทางข้างขวา (Right) ได้หรือไม่ หากสามารถเคลื่อนที่ไปได้ ให้เลือกเคลื่อนที่ขวา (Right) และตรวจสอบตำแหน่งใหม่ของหนู

4. ตรวจสอบว่าหนูสามารถเคลื่อนที่ไปยังทิศทางข้างล่าง (Down) ได้หรือไม่ หากสามารถเคลื่อนที่ไปได้ ให้เลือกเคลื่อนที่ลง (Down) และตรวจสอบตำแหน่งใหม่ของหนู

5. หากมีเส้นทางที่เป็นไปได้มากกว่า 1 เส้นทาง จะทำการบันทึกตำแหน่งปัจจุบันไว้เพื่อใช้ในการถอยกลับ

6. ทำการเคลื่อนที่ไปยังตำแหน่งที่เป็นไปได้ และทำการตรวจสอบว่าตำแหน่งนั้นมีอาหารหรือไม่

7. หากพบอาหาร จะทำการเคลื่อนที่ไปยังตำแหน่งอาหารและทำเครื่องหมายว่าเส้นทางนั้นเป็นเส้นทางที่ถูกต้อง

8.หากไม่พบอาหาร จะทำการย้อนกลับไปยังตำแหน่งที่มีเส้นทางที่เป็นไปได้ก่อนหน้า

โดยเราจะเก็บค่าตัวแปร move เป็น Boolean เพื่อเก็บว่าหนูสามารถเดินไปในเส้นทางต่างๆที่เรากำหนดได้หรือไม่ และเราจะนำค่า move ไปตรวจสอบถ้าหนูสามารถเดินทางไม่ไปในทิศทางตามคำสั่งได้ค่าในตำแหน่งที่หนูต้องการเคลื่อนที่จะถูกกำหนดให้มีค่าเป็น 0 เพื่อทำเคลื่อนที่ในทิศทางดังกล่าว และตำแหน่งใหม่ที่หนูเคลื่อนที่ไป (ตำแหน่งใหม่หลังจากการเคลื่อนที่) จะถูกเพิ่มเข้าไปใน ArrayDeque โดยใช้ object One เพื่อเก็บข้อมูลเกี่ยวกับตำแหน่งและทิศทางการเคลื่อนที่ของหนูในขั้นตอนต่อไปของการค้นหาเส้นทาง

Backtracking step: เงื่อนไขที่ทำให้เกิดการ Backtracking และอัลกอริทึมที่ใช้ในแต่ละขั้นตอน

เงื่อนไขที่ทำให้เกิด Backtracking step คือเมื่อไม่สามารถเคลื่อนที่ได้ อัลกอริทึมคือ postorder traversal (ทำ node ลูกให้หมดก่อน) และอัลกอริทึมจะทำตามขั้นตอนต่อไปนี้:

1. ลบตำแหน่ง(obj.)ใน dq ทีละ 1 ตำแหน่งจนเท่ากับตำแหน่ง ที่ mark ไว้ล่าสุด ใน dq_mark และลบตำแหน่งนั้นออกจาก dq_mark

2.วนลูปเพื่อหาอาหารต่อไป

3. การย้อนกลับจะเกิดขึ้นไปเรื่อย ๆ จนกว่าจะพบอาหารหรือไม่สามารถหาทางออกได้โดยอัลกอริทึมจะสิ้นสุดการทำงานโดยไม่พบทางออก

Finding next Food: ขั้นตอนที่อัลกอริทึมทำก่อนเริ่มการค้นหาFoodถัดไปคืออะไร?

ก่อนเริ่มการค้นหาอาหารถัดไป อัลกอริทึมจะดำเนินการตามขั้นตอนต่อไปนี้เพื่อรีเซ็ตสถานะ:

1. ปรับตำแหน่งล่าสุดของหนู (tratrow และ tratcol) ให้อยู่ที่ตำแหน่งของอาหาร ซึ่งเป็นตำแหน่งเริ่มต้นที่ตั้งของหนูในด่านต่อไป

2. รีเซ็ตอาร์เรย์ชั่วคราว (temp_array) เป็นการกำหนดค่าเริ่มต้นใหม่ (myarray) ซึ่งเป็นการกำหนดค่าเริ่มต้นของด่านที่ผ่านมา

3. ล้างรายการที่เหลืออยู่ใน dq (ArrayDeque ที่ใช้สำหรับการค้นหาเส้นทาง) และ dq_mark (ArrayDeque ที่ใช้สำหรับทำเครื่องหมายเส้นทางหลายเส้น)

4. รีเซ็ตตัวแปรอื่นๆเช่น found, move, และ END ไปสู่สถานะเริ่มต้น

อัลกอริทึมจะสรุปได้อย่างไรว่า no solution?

อัลกอริทึมจะเก็บการเคลื่อนที่ที่ทำไว้โดยใช้ deque เมื่อมันพบว่าถึงทางตันหรือมาถึงตำแหน่งที่ไม่สามารถไปต่อได้ มันจะย้อนกลับโดยการลบการเคลื่อนที่ล่าสุดออกจาก deque และสำรวจทิศทางอื่น ๆ ซึ่งตัวแปรที่นับจุดที่มีทางแยกคือ mark เมื่อ dq_mark ไม่มีค่าอะไรเก็บอยู่ (ไม่มีทางเหลือแล้ว) mark จะเท่ากับ 0 ตอน poll dq_mark ครั้งถัดไป mark จะน้อยกว่า 0 เป็นเงื่อนไขที่ทำให้หลุดจากการค้นหา (no solution)

Demos

maize_1.txt

```
====================Finding Food 1====================
++++++++Food is eaten++++++++
        col_0  col_1  col_2  col_3  col_4
row_0       0      1      1      1      R
row_1       1      1      1      0      1
row_2       0      1      0      F      1
row_3       1      1      1      0      1
row_4       0      0      1      1      1

---------------------PATH FROM START---------------------
 start -> (row  1 ,col  1, R)
    up -> (row  0 ,col  1, 1)
 right -> (row  0 ,col  2, 1)
 right -> (row  0 ,col  3, 1)
 right -> (row  0 ,col  4, F)
---------------------------------------------------------
```

```
====================Finding Food 2====================
++++++++Food is eaten++++++++
        col_0  col_1  col_2  col_3  col_4
row_0       0      1      1      1      1
row_1       1      1      1      0      1
row_2       0      1      0      R      1
row_3       1      1      1      0      1
row_4       0      0      1      1      1

---------------------PATH FROM START---------------------
 start -> (row  0 ,col  4, R)
  left -> (row  0 ,col  3, 1)
  left -> (row  0 ,col  2, 1)
  left -> (row  0 ,col  1, 1)
  down -> (row  1 ,col  1, 1)
  down -> (row  2 ,col  1, 1)
  down -> (row  3 ,col  1, 1)
 right -> (row  3 ,col  2, 1)
  down -> (row  4 ,col  2, 1)
 right -> (row  4 ,col  3, 1)
 right -> (row  4 ,col  4, 1)
    up -> (row  3 ,col  4, 1)
    up -> (row  2 ,col  4, 1)
  left -> (row  2 ,col  3, F)
---------------------------------------------------------


Game End. Do You want to play new game?(Y/N)
N
Exit Game
```

maize_2.txt

```
====================Finding Food 1====================
++++++++Food is eaten++++++++
        col_0  col_1  col_2  col_3  col_4  col_5
row_0       0      0      1      1      1      1
row_1       1      1      0      1      0      1
row_2       0      F      1      0      1      1
row_3       1      1      0      1      R      1

---------------------PATH FROM START--------------------
 start -> (row  0 ,col  5, R)
  down -> (row  1 ,col  5, 1)
  down -> (row  2 ,col  5, 1)
  left -> (row  2 ,col  4, 1)
  down -> (row  3 ,col  4, F)
------------------------------------------------------------


====================Finding Food 2====================
Food not found(No solution)
Game End. Do You want to play new game?(Y/N)
N
Exit Game
```

Limitation

1.ถ้า row หรือ col มากกว่าเท่ากับ 100 array 2 มินิที่ปริ้นออกมformat จะเพี้ยน

2. maze file ที่อ่านต้องไม่มีบรรทัดว่าง

บรรณานุกรม

https://www.geeksforgeeks.org/java-program-for-rat-in-a-maze-backtracking-2/

(sol[x][y] = 0; return false; เมื่อเจอทางตันและ path ที่เดินผ่านไปแล้วจะเปลี่ยนค่าให้เป็น 0)