Map

```
-1  -9  -1  -1  -1  -1  -1  -9  -9  -1
-1  -1  -1  -1  -1  -1  -9  -9  -1  -1
-1  -1  -1  -1  -1  -1  -1  -1  -9  -1
-1  -1  -1  -1  -1  -9  -1  -1  -9  -1
-1  -1  -9  -9  -1  -1  -9  -1  -1  -1
-9  -1  -9  -1  -9  -1  -9  -9  -1  -9
-1  -1  -1  -1  -1  -1  -1  -1  -9  -1
-1  -1  -1  -1  -9  -1  -1  -1  -1  -1
-1  -1  -9  -1  -9  -9  -1  -1  -9  -1
-1  -1  -1  -1  -1  -1  -9  -1  -1  -1
```

Find Path queue

```
 0  -9   4   5   6   7   8  -9  -9  17
 1   2   3   4   5   6  -9  -9  17  16
 2   3   4   5   6   7   8   9  -9  15
 3   4   5   6   7  -9   9  10  -9  14
 4   5  -9  -9   8   9  -9  11  12  13
-9   6  -9  10  -9  10  -9  -9  13  -9
 8   7   8   9  10  11  12  13  -9  17
 9   8   9  10  -9  12  13  14  15  16
10   9  -9  11  -9  -9  14  15  -9  17
11  10  11  12  13  14  -9  16  17  18
Loop : 74 time(s)
```

Find Path Stack

```
 0  -9   4   5   6   7   8  -9  -9  17
 1   2   3   4   5   6  -9  -9  17  16
 2   3   4   5   6   7   8   9  -9  15
-1  -1  -1  -1  -1  -9   9  10  -9  14
-1  -1  -9  -9  -1  -1  -9  11  12  13
-9  -1  -9  -1  -9  -1  -9  -9  13  -9
-1  -1  -1  -1  -1  -1  -1  -1  -9  -1
-1  -1  -1  -1  -9  -1  -1  -1  -1  -1
-1  -1  -9  -1  -9  -9  -1  -1  -9  -1
-1  -1  -1  -1  -1  -1  -9  -1  -1  -1
Loop : 21 time(s)
```

ความแตกต่าง

1. จำนวนการทำซ้ำจนกระทั่งไปถึงเป้าหมาย สังเกตได้ว่าในการหาทางจากตำแหน่ง 0,0 ไปยัง 0,9 นั้น การใช้ Stack ใช้จำนวนครั้งในการทำซ้ำ 21 ครั้ง น้อยกว่าอย่างเห็นได้ชัดเมื่อเทียบกับการใช้ Queue ซึ่งใช้จำนวนครั้งในการทำซ้ำ 74 ครั้ง
2. ผลลัพธ์ที่ได้จากการใช้ Stack คือ path ที่ต้องเดินไปและระยะทางของ path นั้น
3. ผลลัพธ์ที่ได้จากการใช้ Queue คือ ระยะทางที่สั้นที่สุดไปยังตำแหน่งต่างๆ บน array ที่มีการ expand ผ่าน
4. การใช้ Stack นั้นใช้ได้ในกรณีที่ตำแหน่งทางออกของ Maze นั้นอยู่บริเวณสุดขอบทางด้านขวาเท่านั้น หากมีการเปลี่ยนแปลงตำแหน่งทางออกของ Maze Lee Algorithm จะไม่สามารถใช้งานได้และจะมีการทำงานผิดพลาด ตัวอย่างและสาเหตุจะอยู่ในหน้าถัดไป

สาเหตุที่ได้เหตุผลแบบนี้คือ

การ Traverse ไปใน Array 2 มิติโดยการใช้ stack เป็นการ Traverse แบบ Depth-First Search คือจะพยายามหาทางโดยใช้ตำแหน่งที่ได้มาล่าสุดก่อน สังเกตได้จาก `Find Path Stack` ว่าการหาทางจะพยายามไปให้ได้ลึกที่สุดก่อนและเมื่อหากไปต่อไม่ได้แล้วจะย้อนกลับมาที่ตำแหน่งที่สามารถไปได้ต่อได้ โดยผลลัพธ์ที่ได้คือ path (ทางเดิน) ในการเดินทางไปยังปลายทางที่ต้องการ ซึ่งเป็น shortest path รวมถึงแสดงระยะทางของ shortest path นั้น นอกจากนี้ยังแสดง path ที่ method ได้เคยลองไปแต่ไม่สำเร็จ

แต่การ Traverse ไปใน Array 2 มิติโดยการใช้ queue เป็นการ Traverse แบบ Breadth-First Search สังเกตได้จาก `Find Path queue` ว่าการหาทางจะพยายามไปไปในทุกทิศทางที่สามารถไปได้ก่อนทั้ง 4 ทิศทางแล้วจึงค่อย ๆ ขยับไปเรื่อย ๆ จนครบทั้ง array ผลลัพธ์ที่ได้คือระยะทางที่สั้นที่สุดไปยังตำแหน่งต่างๆ บน array ที่มีการ expand ผ่าน

```
Map
-1  -1  -9  -1  -1
-1  -1  -1  -1  -9
-1  -1  -1  -1  -1
-1  -1  -1  -9  -1
-9  -1  -1  -1  -1
```

```
Find Path queue
   0   1  -9  -1  -1
   1   2   3   4  -9
   2   3   4  -1  -1
   3   4  -1  -9  -1
   4  -1  -1  -1  -1
Loop : 9 time(s)

From (0,0) to (0,0)
From (0,0) to (1,0)
From (0,0) to (0,1)
From (1,0) to (2,0)
From (1,0) to (1,1)
From (2,0) to (3,0)
From (2,0) to (2,1)
From (1,1) to (1,2)
From (3,0) to (4,0)
```

```
Find Path Stack
   0   1  -9   5   6
   1   2   3   4  -9
  -1   3   4   5   6
  -1  12  11  -9   7
  12  11  10   9   8
Loop : 15 time(s)

From (0,0) to (0,0)
From (0,0) to (0,1)
From (0,1) to (1,1)
From (1,1) to (1,2)
From (1,2) to (1,3)
From (1,3) to (0,3)
From (0,3) to (0,4)
From (1,3) to (2,3)
From (2,3) to (2,4)
From (2,4) to (3,4)
From (3,4) to (4,4)
From (4,4) to (4,3)
From (4,3) to (4,2)
From (4,2) to (4,1)
From (4,1) to (4,0)
```

สาเหตุที่ได้เหตุผลแบบนี้คือ

Lee Algorithm มีข้อจำกัดของตำแหน่งทางออกของ Maze โดยทางออกจะต้องอยู่ทางขวาสุดเพื่อที่จะให้ Lee Algorithm นั้นทำงานได้อย่างถูกต้อง จากตัวอย่างลองปรับเปลี่ยน Destination ของ Maze จาก 0,9 เป็น 9,0 ซึ่งเป็นบริเวณมุมล่างซ้ายของ Maze พบว่า การใช้ Stack Path ที่ได้จะพยายามไปด้านขวาก่อน เนื่องจากการ expand และ push การเดินลงไปใน Stack จะทำให้ทิศทางการเดินไปทางขวาของตำแหน่งก่อนหน้าอยู่บนสุดของ Stack และเมื่อ pop stack ออกมาจะได้การเดินที่พยายามไปทางขวา path ที่ได้จึงมีลักษณะที่มีการอ้อมไปทางขวา