**Group Project 2**

**EGCO221**

**6413108 นางสาวพาลาภ วสุวัต**

**6413211 นายกอบกฤช เรืองสุริยกิจ**

**สารบัญ**

[คู่มือการใช้งาน 1](#_Toc133087635)

[เกี่ยวกับโปรแกรม 1](#_Toc133087636)

[วิธีการใช้งานโปรแกรม 2](#_Toc133087637)

[แนะนำ Class เบื้องต้น 3](#_Toc133087638)

[LightBoard 3](#_Toc133087639)

[SolutionPath 4](#_Toc133087640)

[Graph data structure 5](#_Toc133087641)

[Data structure อื่น ๆ 7](#_Toc133087642)

[Graph Algorithms 7](#_Toc133087643)

[Demo 3 9](#_Toc133087644)

[Demo 4 18](#_Toc133087645)

[Demo 5 27](#_Toc133087646)

[สรุป 33](#_Toc133087647)

[ข้อจำกัดของโปรแกรม 34](#_Toc133087648)

[อ้างอิง 35](#_Toc133087649)

**คู่มือการใช้งาน**

**เกี่ยวกับโปรแกรม**

โปรแกรมจำลองเกมที่มีชื่อว่า “Lights Out” เป็นเกมกระดานที่มีขนาด n x n โดยบนกระดานของโปรแกรมนี้จะประกอบไปด้วยตัวเลขซึ่งเป็นตัวแทนของไฟเปิด (1) ไฟปิด (0) และ อาจมีไฟเสีย 1 ดวง (1x หรือ 0x) อยู่บนกระดานด้วย

เป้าหมาย:

การปิดไฟทั้งหมดบนกระดาน หรือ กล่าวได้ว่าตัวเลขบนกระดานจะเป็นเลข 0 ทั้งหมด

สิ่งที่ผู้ใช้งานสามารถทำเพื่อบรรลุเป้าหมายดังกล่าว:

การกดสวิตช์เพื่อ toggle ไฟที่อยู่บนกระดาน

* หากไฟปกติเปิดถูก toggle จะกลายเป็นไฟปิด (1 -> 0) ในทางกลับกันหากไฟปิดถูก toggle จะกลายเป็นไฟเปิด (0 -> 1) ซึ่งการกดสวิตช์ตัวหนึ่งนั้นไม่เพียงจะ toggle ตัวมันเอง แต่จะ toggle ไฟโดยรอบที่อยู่ด้านบน ล่าง ซ้าย และ ขวา ของมันด้วย ดังภาพต่อไปนี้

A picture containing background pattern

Description automatically generated A picture containing background pattern

Description automatically generated

ไฟที่อยู่ตรงกลางถูก toggle ไฟที่อยู่มุมซ้ายบนถูก toggle

* หากไฟเสียถูก toggle จะ toggle ตัวมันเอง และ ไฟที่อยู่รอบ ๆ ตัวมันเองทางด้านซ้ายบน ขวาบน ซ้ายล่าง และ ขวาล่างด้วย ดังภาพต่อไปนี้

ไฟที่อยู่ตรงกลางถูก toggle ไฟเสียที่อยู่มุมซ้ายบนถูก toggle

**วิธีการใช้งานโปรแกรม**

1. เมื่อเปิดโปรแกรมขึ้นมา โปรแกรมจะให้ผู้ใช้งานกรอกขนาดของกระดาน (n) ที่ต้องการเพื่อสร้างเกมกระดานขนาด n x n โดยที่ n ต้องเป็นตัวเลขจำนวนเต็มที่มีค่าตั้งแต่ 2 ขึ้นไป (หากต้องการออกจากโปรแกรมให้ใส่ n เป็น 0)

Text

Description automatically generated

1. หลังจากสร้างกระดานเรียบร้อยแล้ว ผู้ใช้งานจะต้องกรอกสถานะของไฟเป็นตัวเลข 0 และ 1 รวมทั้งหมด n x n ตัว ตามขนาดของกระดาน เพื่อสร้างกระดานเริ่มต้น

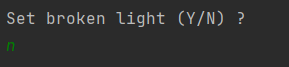
Text

Description automatically generated

1. ถัดมาโปรแกรมจะถามผู้ใช้งานว่าต้องการให้มีไฟเสียบนกระดานหรือไม่ หากต้องการโปรแกรมก็จะให้กรอกตำแหน่งของแถว และ หลัก ที่ต้องการต่อไป

Text

Description automatically generated



1. หลังจากนั้น โปรแกรมจะทำการค้นหาวิธีการปิดไฟทั้งกระดานที่ใช้จำนวนครั้งในการกดสวิตช์น้อยที่สุดออกมาให้ และ โปรแกรมจะเริ่มทำงานใหม่ ดังข้อ 1 ต่อไป

**แนะนำ Class เบื้องต้น**

**LightBoard**

ข้อมูลที่เก็บ

|  |  |
| --- | --- |
| String[ ][ ] board | กระดานปัจจุบัน |
| int movingRow | แถวที่ถูกกดสวิตช์ |
| int movingCol | หลักที่ถูกกดสวิตช์ |
| int[ ] brokenPosition | ตำแหน่งของไฟเสียบนกระดาน โดยตำแหน่งที่ 0 ของ array จะเก็บแถว และ ตำแหน่งที่ 1 ของ array จะเก็บหลัก |

Method ที่สำคัญ

|  |  |
| --- | --- |
| toggleLight(int row, int column) | Method เสมือนการกดสวิตช์ไฟ รับแถว และ หลัก ของไฟดวงที่ต้องการเข้าไป และ จะทำการตรวจสอบว่าตำแหน่งดังกล่าวใช่ไฟเสียหรือไม่ หากไม่จะ toggle ไฟตามเงื่อนไขปกติ แต่หากใช่จะ toggle ไฟตามเงื่อนไขของไฟเสีย |
| equals(Object o) | Override method สำหรับการเปรียบเทียบ object LightBoard ว่าเหมือนกันหรือไม่ โดยเปรียบเทียบจากกระดานปัจจุบันที่เก็บไว้ |
| hashCode() | Override method สำหรับการกำหนด id ของทุก ๆ object LightBoard ให้มีค่าเหมือนกัน ช่วยให้การเปรียบเทียบด้วย method equals ทำการเปรียบเทียบกระดานได้ว่ามีค่าเหมือนกันจริงโดยไม่ต้องสนใจ id |

**SolutionPath**

ข้อมูลที่เก็บ

|  |  |
| --- | --- |
| Graph boardGraph | graph (SimpleGraph) |
| ArrayDeque processingQueue | Queue |
| LightBoard initBoard | กระดานเริ่มต้น (ผู้ใช้งานเป็นคนกำหนด) |
| LightBoard validBoard | กระดานที่ต้องการ (กระดานที่ปิดไฟหมดแล้ว) ใช้สำหรับการเปรียบเทียบว่ากระดานที่กำลังค้นหานั้นเป็นกระดานที่ถูกต้องแล้วหรือไม่ |

Method ที่สำคัญ

|  |  |
| --- | --- |
| solution() | Method สำหรับการค้นหาวิธีการปิดไฟทั้งกระดานด้วยจำนวนครั้งในการกดสวิตช์ที่น้อยยที่สุด |
| printPath(LightBoard endBoard) | Method สำหรับแสดงผล path ระหว่าง initBoard กับ กระดานที่ปิดไฟหมดแล้วที่ถูกส่งเข้ามาเป็น parameter |

**Graph data structure**

**SimpleGraph** :

Node – ใช้เก็บ Object LightBoard

  Edge – ใช้เป็นตัวเชื่อมระหว่าง state หรือกล่าวคือจะเชื่อมระหว่าง state ตั้งต้นก่อนที่จะกดสวิตช์ กับ state หลังจากดสวิตช์ไปแล้ว

  เงื่อนไขของการเพิ่ม edge - เมื่อกระดานใหม่ที่สร้างขึ้นมาไม่ซ้ำกับกระดานเดิมที่มีอยู่ใน graph แล้วจึงทำการ edge ที่เชื่อมระหว่างกระดานใหม่กับกระดานที่ก่อนที่จะทำการเปิด/ปิดไฟ

เหตุผลที่เลือกใช้ SimpleGraph เนื่องจากในตัว Graph data structure ที่ใช้สำหรับหาวิธีการปิดไฟทั้งหมดในกระดานโดยใช้จำนวนครั้งที่น้อยที่สุดนี้ไม่ได้สนใจน้ำหนักระหว่าง state และ Graph นี้จะเอาเฉพาะวิธีที่สั้นที่สุดจาก initial state (กระดานเริ่มต้น) ถึง final state (กระดานที่ปิดไฟหมดแล้ว) เท่านั้นจึงไม่จำเป็นที่ Graph นี้จะต้องระบุทิศทางอย่างชัดเจน ดังนั้นจึงเลือกใช้เป็น SimpleGraph

graph แสดง nodes และ edges ของ Demo 1

Diagram

Description automatically generated

graph แสดง nodes และ edges ของ Demo 2

Diagram

Description automatically generated

**Data structure อื่น ๆ**

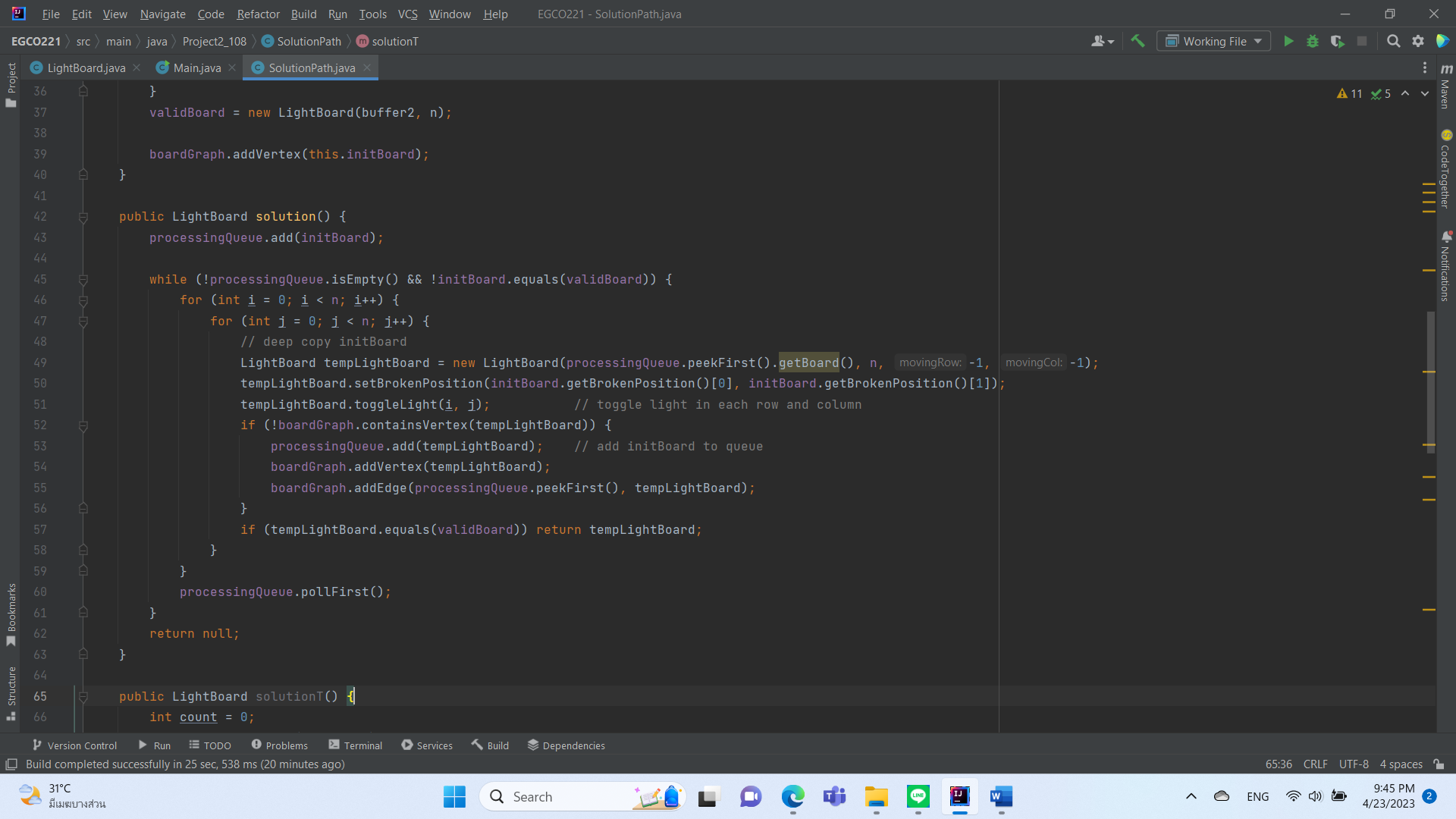
**ArrayDeque (Queue)**

ข้อมูลที่เก็บ :  Object ของ class LightBoard

เหตุผลที่เลือกใช้ :

ใช้เป็น Queue ช่วยในการทำ Breadth First Search เพื่อหาวิธีการปิดไฟทั้งหมดบนกระดาน โดยมีการกดสวิตช์ที่น้อยที่สุด

**Graph Algorithms**



**อธิบาย algorithm เบื้องต้น**

  algorithm นี้จะเริ่มทำงานเมื่อทำการรับ input เสร็จหมดแล้วในบรรทัดที่ 67 ใน main.java

แล้วทำการสร้าง graph แบบ Breadth First Search โดยจะทำการนำกระดานที่ต้องวิเคระห์เพิ่มลงใน queue

แล้วดึงออกมาวิเคระห์ผลตามลำดับซึ่งจะทำให้สามารถตรวจสอบได้ว่าในการกดสวิตช์ไฟแต่ละครั้งจะมีรูปแบบกระดานแบบไหนได้บ้าง จากนั้นจึงทำการเช็คว่าในการกดสวิตช์ไฟครั้งนั้นสามารถเกิดกระดานที่ปิดไฟทั้งหมดได้หรือไม่ ก่อนที่จะทำการไปวิเคราะห์การกดสวิตช์ไฟครั้งต่อไป (state ต่อไป) และใช้ graph เข้ามาช่วยในการเก็บขั้นตอนการปิดไฟโดยจะทำการเพิ่ม path ระหว่างกระดานที่กำลังวิเคระห์อยู่กับผลลัพธ์ของการกดสวิตช์ไฟในแต่ละตำแหน่งเพื่อจะใช้ในการบอกเส้นทางระหว่างกระดานที่ปิดไฟแล้วกับกระดานเริ่มต้น (initial state) แต่จะมีเงื่อนไขในการเพิ่มกระดานลง graph และ queue อยู่คือ ใน graph ต้องไม่มีกระดานที่จะเพิ่มอยู่

เนื่องจากกระดานนี้สามารถเกิดได้จากการกดสวิตช์ไฟด้วยวิธีอื่นที่มีจำนวนครั้งในการกดสวิตช์น้อยกว่าหรือเท่ากันแน่นอน ดังนั้นกระดานนี้จึงเป็นกระดานที่อาจจะถูกวิเคราะห์ไปแล้วจึงไม่มีความจำเป็นที่จะต้องเอาไปวิเคราะห์ซ้ำหรือเก็บลง graph อีก

  โดย algorithm นี้จะหยุดก็ต่อเมื่อพบกระดานที่เป็นคำตอบที่ถูกต้อง ซึ่งเป็นกระดานที่ปิดไฟทั้งหมดทุกตัว (เลข 0 ทั้งกระดาน) โดย method จะทำการ return object ของกระดานที่ถูกต้องกลับไปเพื่อใช้ในการหา path  หรือ ขั้นตอนในการปิดไฟทั้งหมดจาก graph ที่สร้างไว้ต่อไป แต่หากทำไปเรื่อย ๆ แล้ว ไม่มีกระดานอยู่ใน processing queue แล้วนั่นหมายความว่าทุกๆรูปแบบความเป็นไปได้ของกระดานที่จะเกิดขึ้นได้อยู่ใน graph ทั้งหมดแล้วแต่ยังไม่มีกระดานที่ถูกปิดไฟทั้งหมดดังนั้น initial state นี้จึงไม่มีความเป็นไปได้เลยที่จะปิดไฟทั้งหมดได้จึงส่งค่า null กลับไป

  จากนั้นจึงนำกระดานที่ได้จาก method solution() ไปทำการเช็คว่ามีกระดานที่ปิดไฟทั้งหมดแล้วหรือไม่ในบรรทัดที่ 68 ใน main.java ถ้ามีจึงทำการ print ขั้นตอนในการปิดไฟออกมา หากไม่มีให้ print ว่าไม่สามารถปิดไฟทั้งหมดได้

**ขั้นตอนการทำงาน**

1. เพิ่ม initial state ที่รับ input เข้ามาลงใน processing queue (บรรทัดที่ 43 ใน SolutionPath.java)
2. นำกระดานที่อยู่ใน queue ออกมาทำการไล่กดแต่ละสวิตช์ (บรรทัดที่ 49 ใน SolutionPath.java) โดยถ้าไม่มีกระดานอยู่ใน queue แล้วจะทำการส่ง null กลับไป (บรรทัดที่ 45 ใน SolutionPath.java)
3. เช็คว่ากระดานที่เกิดจากการกดสวิตช์แต่ละตำแหน่งไม่ซ้ำกับกระดานที่มีอยู่ใน graph แล้วจึงทำการเพิ่มกระดานนั้นลง queue และ graph (บรรทัดที่ 52 ใน SolutionPath.java)
4. เช็คว่ากระดานที่เกิดจากการกดสวิตช์แต่ละตำแหน่งนั้นเป็นกระดานที่ปิดไฟหมดแล้วหรือไม่ ถ้าใช่ทำการส่งกระดานนั้นกลับไป ถ้าไม่ใช่ algorithm จะวนกลับมาทำงานต่อในขั้นตอนที่ 2 (บรรทัดที่ 57 ใน SolutionPath.java)
5. pop กระดานที่ทำการดึงออกมาจาก queue ในตอนแรก (บรรทัดที่ 60 ใน SolutionPath.java)

1. Breadth First Search (BFS) :

  เหตุผลที่ใช้ - เนื่องจากต้องการวิเคระห์การกดสวิตช์ไฟในการกดแต่ละครั้งก่อนที่จะไปการกดครั้งต่อไปดังที่ได้อธิบายไปข้างต้น การใช้ BFS เหมาะที่จะนำมาประยุกต์ใช้ในการวิเคราะห์กระดาน และสร้าง graph เพราะจะทำให้สามารถสร้าง graph ที่ค่อย ๆ วิเคราะห์การกดแต่ละครั้งให้หมดทุกรูปแบบในการกดสวิตช์ไฟก่อนที่จะไปการกดสวิตช์ครั้งต่อไปได้

2. Dijkstra algorithm :

เหตุผลที่ใช้ - ใช้เพื่อดึง graph path ของกระดานเริ่มต้นกับกระดานที่ถูกปิดไฟทั้งหมด จากตัว graph ที่เก็บรูปแบบการกดสวิตซ์ไฟทั้งหมดเอาไว้ (แต่เนื่องจาก algorithms ทำให้ graph ไม่มีกระดานที่ซ้ำกันจึงทำให้ graph path ของกระดานเริ่มต้นกับกระดานที่ถูกปิดไฟทั้งหมดมีแค่เส้นทางเดียว ดังนั้นจึงไม่จำเป็นต้องหาทางที่สั้นที่สุด)

**Noted**

จากที่เคยกล่าวไว้ว่า Node ของ graph จะถูกเก็บเป็น Object ของ class LightBoard ซึ่งภายใน class LightBoard ก็จะเก็บกระดานซึ่งเป็น array 2 มิติ ของ string ในการอธิบายต่อไปนี้จะแทนเป็นชุดของตัวเลข 9 ตัว จากซ้ายไปขวา ทีละแถวจากบนลงล่างของตาราง ดังตัวอย่าง

100 110 101

|  |  |  |
| --- | --- | --- |
| 1 | 0 | 0 |
| 1 | 1 | 0 |
| 1 | 0 | 1 |

นอกจากนี้ภายใน class ยังมีตำแหน่งของสวิตช์ที่ถูกกดไป และ ตำแหน่งของไฟเสียซึ่งในการอธิบายจะบอกตำแหน่งของสวิตช์ที่ถูกกดในรูป (แถว, หลัก) เช่น (1, 0) คือ กดสวิตช์ที่แถว 1 หลัก 0 ส่วนไฟเสียจะกล่าวถึงแค่ตอนต้นหลังจากนั้นจะละไว้ เนื่องจาก เป็นค่าคงที่ ไม่ถูกเปลี่ยนแปลง

\* ในตัวอย่างที่จะแสดงด้านล่างหากต้องการดูขั้นตอนทั้งหมดอย่างละเอียดสามารถทำได้โดยการเปลี่ยนจาก method solution() เป็น method solutionT() ได้ในบรรทัดที่ 67 ใน main.java

**Demo 3**

กระดานเริ่มต้น 000 101 010

ตำแหน่งไฟเสีย ไม่มี

**Step 0:**

Add node ของกระดานเริ่มต้นเข้าไปใน graph และ add กระดานเริ่มต้นนี้เข้าไปใน queue

|  |
| --- |
| 000 101 010 |

Queue ปัจจุบัน:

**Step 1:**

Visit object ตัวแรกที่อยู่ใน queue (000 101 010) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ทั้งหมด 9 ตัว

* 110 001 010 กดสวิตช์ (0, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 111 111 010 กดสวิตช์ (0, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 011 100 010 กดสวิตช์ (0, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 100 011 110 กดสวิตช์ (1, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 010 010 000 กดสวิตช์ (1, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 001 110 011 กดสวิตช์ (1, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 000 001 100 กดสวิตช์ (2, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 000 111 000 กดสวิตช์ (2, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 000 100 011 กดสวิตช์ (2, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph

หลังจากนั้น pop object ที่ visited ออก

Cont.

Queue ปัจจุบัน:

First

000 101 010 Visited (pop)

110 001 010 -> New

111 111 010 -> New

011 100 010 -> New

100 011 110 -> New

010 010 000 -> New

001 110 011 -> New

000 001 100 -> New

000 111 101 -> New

000 100 001 -> New

Last

**Step 2:**

Visit object ตัวแรกที่อยู่ใน queue (110 001 010) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ ดังนี้

* 000 101 010 กดสวิตช์ (0, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 001 010 010 กดสวิตช์ (0, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 101 000 010 กดสวิตช์ (0, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 010 111 110 กดสวิตช์ (1, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 100 110 000 กดสวิตช์ (1, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 111 010 011 กดสวิตช์ (1, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 110 101 100 กดสวิตช์ (2, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 110 011 101 กดสวิตช์ (2, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 110 000 001 กดสวิตช์ (2, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph

หลังจากนั้น pop object ที่ visited ออก

Cont.

Queue ปัจจุบัน:

First

110 001 010 Visited (pop)

111 111 010

011 100 010

100 011 110

010 010 000

001 110 011

000 001 100

000 111 101

000 100 001

001 011 010 -> New

101 000 010 -> New

010 111 110 -> New

100 110 000 -> New

111 010 011 -> New

110 101 100 -> New

110 011 101 -> New

110 000 001 -> New

Last

**Step 3:**

Visit object ตัวแรกที่อยู่ใน queue (111 111 010) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ ดังนี้

* 001 011 010 กดสวิตช์ (0, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 000 101 010 กดสวิตช์ (0, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 100 110 010 กดสวิตช์ (0, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 011 001 110 กดสวิตช์ (1, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 101 000 000 กดสวิตช์ (1, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 110 100 011 กดสวิตช์ (1, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 111 011 100 กดสวิตช์ (2, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 111 101 101 กดสวิตช์ (2, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 111 110 001 กดสวิตช์ (2, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph

หลังจากนั้น pop object ที่ visited ออก

Cont.

Queue ปัจจุบัน:

First

111 111 010 Visited (pop)

011 100 010

100 011 110

010 010 000

001 110 011

000 001 100

000 111 101

000 100 001

001 011 010

101 000 010

010 111 110

100 110 000

111 010 011

110 101 100

110 011 101

110 000 001

100 110 010 -> New

011 001 110 -> New

101 000 000 -> New

110 100 011 -> New

111 011 100 -> New

111 101 101 -> New

111 110 001 -> New

Last

**Step 4:**

Visit object ตัวแรกที่อยู่ใน queue (011 100 010) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ ดังนี้

* 101 000 010 กดสวิตช์ (0, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 100 110 010 กดสวิตช์ (0, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 000 101 010 กดสวิตช์ (0, 2) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 111 010 110 กดสวิตช์ (1, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 001 011 000 กดสวิตช์ (1, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 010 111 011 กดสวิตช์ (1, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 011 000 100 กดสวิตช์ (2, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 011 110 101 กดสวิตช์ (2, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 011 101 001 กดสวิตช์ (2, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph

หลังจากนั้น pop object ที่ visited ออก

Cont.

Queue ปัจจุบัน:

First

011 100 010 Visited (pop)

100 011 110

010 010 000

001 110 011

000 001 100

000 111 101

000 100 001

001 011 010

101 000 010

010 111 110

100 110 000

111 010 011

110 101 100

110 011 101

110 000 001

100 110 010

011 001 110

101 000 000

110 100 011

111 011 100

111 101 101

111 110 001

111 010 110 -> New

001 011 000 -> New

010 111 011 -> New

011 000 100 -> New

011 110 101 -> New

011 101 001 -> New

Last

Cont.

**Step 5:**

Visit object ตัวแรกที่อยู่ใน queue (100 011 110) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ ดังนี้

* 010 111 110 กดสวิตช์ (0, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 011 001 110 กดสวิตช์ (0, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 111 010 110 กดสวิตช์ (0, 2) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 000 101 010 กดสวิตช์ (1, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 110 100 100 กดสวิตช์ (1, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 101 000 111 กดสวิตช์ (1, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 100 111 000 กดสวิตช์ (2, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 100 001 001 กดสวิตช์ (2, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 100 010 101 กดสวิตช์ (2, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph

หลังจากนั้น pop object ที่ visited ออก

Cont.

Queue ปัจจุบัน:

First

100 011 110 Visited (pop)

010 010 000

001 110 011

000 001 100

000 111 101

000 100 001

001 011 010

101 000 010

010 111 110

100 110 000

111 010 011

110 101 100

110 011 101

110 000 001

100 110 010

011 001 110

101 000 000

110 100 011

111 011 100

111 101 101

111 110 001

111 010 110

001 011 000

010 111 011

011 000 100

011 110 101

011 101 001

110 100 100 -> New

101 000 111 -> New

Cont.

100 111 000 -> New

100 001 001 -> New

100 010 101 -> New

Last

Process keeps running.

∙

∙

∙

**Step 347:**

Visit object ตัวแรกที่อยู่ใน queue (000 001 011) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ ดังนี้

* 110 101 011 กดสวิตช์ (0, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 111 011 011 กดสวิตช์ (0, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 011 000 011 กดสวิตช์ (0, 2) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 100 111 111 กดสวิตช์ (1, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 010 110 001 กดสวิตช์ (1, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 001 010 010 กดสวิตช์ (1, 2) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 000 101 101 กดสวิตช์ (2, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 000 011 100 กดสวิตช์ (2, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 000 000 000 กดสวิตช์ (2, 2) ⟶ กระดานที่ถูกต้อง add object ลง graph

หลังจากนั้น pop object ที่ visited ออก

Cont.

Queue ปัจจุบัน:

First

000 001 011 Visited (pop)

000 111 010

000 100 110

001 000 101

001 011 001

001 101 000

010 100 110

010 111 010

010 001 011

011 110 100

100 101 000

100 110 100

100 000 101

101 111 010

110 011 001

100 010 011

100 100 010

100 111 110

101 011 101

101 000 001

101 110 000

110 111 110

110 100 010

110 010 011

111 101 100

000 110 000

000 101 100

000 011 101

001 100 010

010 000 001

111 001 100

111 010 000

111 100 001

110 011 110

101 111 101

011 110 011

101 000 101

100 111 010

100 001 011

100 010 111

Cont.

111 011 001

111 101 000

111 110 100

110 010 111

110 001 011

110 111 010

001 010 111

001 100 110

001 111 010

000 011 001

000 000 101

000 110 100

011 111 010

011 100 110

011 010 111

010 101 000

110 101 011

110 011 010

110 000 110

111 100 101

111 111 001

111 001 000

100 000 110

100 011 010

100 101 011

101 010 100

010 001 000

010 010 100

010 100 101

011 011 010

000 111 001

010 110 011

010 000 010

010 011 110

011 111 101

011 100 001

011 010 000

000 011 110

000 000 010

000 110 011

001 001 100

110 010 000

110 001 100

110 111 101

111 000 010

100 100 001

001 101 100

001 110 000

001 000 001

000 111 110

011 011 101

Cont.

101 010 011

011 000 011

011 110 010

011 101 110

010 001 101

010 010 001

010 100 000

001 101 110

001 110 010

001 000 011

000 111 100

111 100 000

111 111 100

111 001 101

110 110 010

101 010 001

000 011 100 -> New

000 000 000 -> New

Last

Method พบกระดานที่ถูกต้องแล้วจะหยุดทำงาน และ return object ของกระดานดังกล่าวกลับไป

**Demo 4**

กระดานเริ่มต้น 000 101 010

ตำแหน่งไฟเสีย แถว 2 หลัก 2

**Step 0:**

Add node ของกระดานเริ่มต้นเข้าไปใน graph และ add กระดานเริ่มต้นนี้เข้าไปใน queue

|  |
| --- |
| 000 101 010 |

Queue ปัจจุบัน:

**Step 1:**

Visit object ตัวแรกที่อยู่ใน queue (000 101 010) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ทั้งหมด 9 ตัว

* 110 001 010 กดสวิตช์ (0, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 111 111 010 กดสวิตช์ (0, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 011 100 010 กดสวิตช์ (0, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 100 011 110 กดสวิตช์ (1, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 010 010 000 กดสวิตช์ (1, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 001 110 011 กดสวิตช์ (1, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 000 001 100 กดสวิตช์ (2, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 000 111 101 กดสวิตช์ (2, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 000 111 011 กดสวิตช์ (2, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph

หลังจากนั้น pop object ที่ visited ออก

Cont.

Queue ปัจจุบัน:

First

000 101 010 Visited (pop)

110 001 010 -> New

111 111 010 -> New

011 100 010 -> New

100 011 110 -> New

010 010 000 -> New

001 110 011 -> New

000 001 100 -> New

000 111 101 -> New

000 111 011 -> New

Last

**Step 2:**

Visit object ตัวแรกที่อยู่ใน queue (110 001 010) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ ดังนี้

* 000 101 010 กดสวิตช์ (0, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 001 011 010 กดสวิตช์ (0, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 101 000 010 กดสวิตช์ (0, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 010 111 110 กดสวิตช์ (1, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 100 110 000 กดสวิตช์ (1, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 111 010 011 กดสวิตช์ (1, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 110 101 100 กดสวิตช์ (2, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 110 011 101 กดสวิตช์ (2, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 110 011 011 กดสวิตช์ (2, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph

หลังจากนั้น pop object ที่ visited ออก

Cont.

Queue ปัจจุบัน:

First

110 001 010 Visited (pop)

111 111 010

011 100 010

100 011 110

010 010 000

001 110 011

000 001 100

000 111 101

000 111 011

001 011 010 -> New

101 000 010 -> New

010 111 110 -> New

100 110 000 -> New

111 010 011 -> New

110 101 100 -> New

110 011 101 -> New

110 011 011 -> New

Last

**Step 3:**

Visit object ตัวแรกที่อยู่ใน queue (111 111 010) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ ดังนี้

* 001 011 010 กดสวิตช์ (0, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 000 101 010 กดสวิตช์ (0, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 100 110 010 กดสวิตช์ (0, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 011 001 110 กดสวิตช์ (1, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 101 000 000 กดสวิตช์ (1, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 110 100 011 กดสวิตช์ (1, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 111 011 100 กดสวิตช์ (2, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 111 101 101 กดสวิตช์ (2, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 111 101 011 กดสวิตช์ (2, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph

หลังจากนั้น pop object ที่ visited ออก

Cont.

Queue ปัจจุบัน:

First

111 111 010 Visited (pop)

011 100 010

100 011 110

010 010 000

001 110 011

000 001 100

000 111 101

000 111 011

001 011 010

101 000 010

010 111 110

100 110 000

111 010 011

110 101 100

110 011 101

110 011 011

100 110 010 -> New

011 001 110 -> New

101 000 000 -> New

110 100 011 -> New

111 011 100 -> New

111 101 101 -> New

111 101 011 -> New

Last

**Step 4:**

Visit object ตัวแรกที่อยู่ใน queue (011 100 010) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ ดังนี้

* 101 000 010 กดสวิตช์ (0, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 100 110 010 กดสวิตช์ (0, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 000 101 010 กดสวิตช์ (0, 2) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 111 010 110 กดสวิตช์ (1, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 001 011 000 กดสวิตช์ (1, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 010 111 011 กดสวิตช์ (1, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 011 000 100 กดสวิตช์ (2, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 011 110 101 กดสวิตช์ (2, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 011 110 011 กดสวิตช์ (2, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph

หลังจากนั้น pop object ที่ visited ออก

Cont.

Queue ปัจจุบัน:

First

011 100 010 Visited (pop)

100 011 110

010 010 000

001 110 011

000 001 100

000 111 101

000 111 011

001 011 010

101 000 010

010 111 110

100 110 000

111 010 011

110 101 100

110 011 101

110 011 011

100 110 010

011 001 110

101 000 000

110 100 011

111 011 100

111 101 101

111 101 011

111 010 110 -> New

001 011 000 -> New

010 111 011 -> New

011 000 100 -> New

011 110 101 -> New

011 110 011 -> New

Last

Cont.

**Step 5:**

Visit object ตัวแรกที่อยู่ใน queue (100 011 110) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ ดังนี้

* 010 111 110 กดสวิตช์ (0, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 011 001 110 กดสวิตช์ (0, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 111 010 110 กดสวิตช์ (0, 2) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 000 101 010 กดสวิตช์ (1, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 110 100 100 กดสวิตช์ (1, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 101 000 111 กดสวิตช์ (1, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 100 111 000 กดสวิตช์ (2, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 100 001 001 กดสวิตช์ (2, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 100 001 111 กดสวิตช์ (2, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph

หลังจากนั้น pop object ที่ visited ออก

Cont.

Queue ปัจจุบัน:

First

100 011 110 Visited (pop)

010 010 000

001 110 011

000 001 100

000 111 101

000 111 011

001 011 010

101 000 010

010 111 110

100 110 000

111 010 011

110 101 100

110 011 101

110 011 011

100 110 010

011 001 110

101 000 000

110 100 011

111 011 100

111 101 101

111 101 011

111 010 110

001 011 000

010 111 011

011 000 100

011 110 101

011 110 011

110 100 100 -> New

101 000 111 -> New

Cont.

100 111 000 -> New

100 001 001 -> New

100 001 111 -> New

Last

Process keeps running.

∙

∙

∙

**Step 190:**

Visit object ตัวแรกที่อยู่ใน queue (000 010 001) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ ดังนี้

* 110 110 001 กดสวิตช์ (0, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 111 000 001 กดสวิตช์ (0, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 011 011 001 กดสวิตช์ (0, 2) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 100 100 101 กดสวิตช์ (1, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 010 101 011 กดสวิตช์ (1, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 001 001 000 กดสวิตช์ (1, 2) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 000 110 111 กดสวิตช์ (2, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 000 000 110 กดสวิตช์ (2, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 000 000 000 กดสวิตช์ (2, 2) ⟶ กระดานที่ถูกต้อง add object ลง graph

หลังจากนั้น pop object ที่ visited ออก

Queue ปัจจุบัน:

Cont.

First

000 010 001 Visited (pop)

000 010 111

111 010 001

110 101 110

110 011 111

110 011 001

101 001 101

101 111 100

101 111 010

100 000 011

100 000 101

100 110 100

000 101 101

001 010 010

001 100 011

001 100 101

010 110 001

010 000 000

010 000 110

011 111 111

011 111 001

011 001 000

100 111 111

100 001 110

100 001 000

101 110 001

101 110 111

101 000 110

110 010 010

110 010 100

110 100 101

111 011 010

100 110 101

101 001 010

101 111 011

101 111 101

110 101 001

110 011 000

110 011 110

111 100 111

Cont.

111 100 001

111 010 000

000 100 111

000 010 110

000 010 000

001 101 001

001 101 111

001 011 110

010 001 010

010 001 100

010 111 101

011 000 010

111 011 011

111 101 010

111 101 100

110 010 101

110 010 011

110 100 010

101 110 110

101 110 000

101 000 001

100 111 110

011 111 000

011 111 110

011 001 111

010 110 000

001 010 011

100 011 100

111 111 111

110 000 000

110 110 001

110 110 111

001 110 001

000 001 110

000 111 111

000 111 001

011 101 101

011 011 100

011 011 010

010 100 011

010 100 101

010 010 100

110 001 101

111 110 010

111 000 011

111 000 101

100 010 001

100 100 000

100 100 110

101 011 111

101 011 001

101 101 000

010 011 111

010 101 110

010 101 000

011 010 001

011 010 111

011 100 110

000 110 010

000 110 100

Cont.

000 000 101

001 111 010

010 010 101

011 101 010

011 011 011

011 011 101

000 001 001

000 111 000

000 111 110

001 000 111

001 000 001

001 110 000

110 000 111

110 110 110

110 110 000

111 001 001

111 001 111

111 111 110

100 101 010

100 101 100

100 011 101

101 100 010

001 111 011

001 001 010

001 001 100

000 110 101

000 110 011

000 000 010

011 010 110

011 010 000

011 100 001

010 011 110

101 011 000

101 011 110

101 101 111

100 010 000

111 110 011

011 100 101

010 011 010

010 101 011

Cont.

010 101 101

001 111 001

001 001 000

001 001 110

000 110 111

000 110 001

000 000 000 -> New

Last

Method พบกระดานที่ถูกต้องแล้วจะหยุดทำงาน และ return object ของกระดานดังกล่าวกลับไป

**Demo 5**

กระดานเริ่มต้น 000 101 010

ตำแหน่งไฟเสีย แถว 0 หลัก 1

**Step 0:**

Add node ของกระดานเริ่มต้นเข้าไปใน graph และ add กระดานเริ่มต้นนี้เข้าไปใน queue

|  |
| --- |
| 000 101 010 |

Queue ปัจจุบัน:

**Step 1:**

Visit object ตัวแรกที่อยู่ใน queue (000 101 010) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ทั้งหมด 9 ตัว

* 110 001 010 กดสวิตช์ (0, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 010 000 010 กดสวิตช์ (0, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 011 100 010 กดสวิตช์ (0, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 100 011 110 กดสวิตช์ (1, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 010 010 000 กดสวิตช์ (1, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 001 110 011 กดสวิตช์ (1, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 000 001 100 กดสวิตช์ (2, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 000 111 101 กดสวิตช์ (2, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 000 100 001 กดสวิตช์ (2, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph

หลังจากนั้น pop object ที่ visited ออก

Cont.

Queue ปัจจุบัน:

First

000 101 010 Visited (pop)

110 001 010 -> New

010 000 010 -> New

011 100 010 -> New

100 011 110 -> New

010 010 000 -> New

001 110 011 -> New

000 001 100 -> New

000 111 101 -> New

000 100 001 -> New

Last

**Step 2:**

Visit object ตัวแรกที่อยู่ใน queue (110 001 010) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ ดังนี้

* 000 101 010 กดสวิตช์ (0, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 100 100 010 กดสวิตช์ (0, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 101 000 010 กดสวิตช์ (0, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 010 111 110 กดสวิตช์ (1, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 100 110 000 กดสวิตช์ (1, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 111 010 011 กดสวิตช์ (1, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 110 101 100 กดสวิตช์ (2, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 110 011 101 กดสวิตช์ (2, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 110 000 001 กดสวิตช์ (2, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph

หลังจากนั้น pop object ที่ visited ออก

Cont.

Queue ปัจจุบัน:

First

110 001 010 Visited (pop)

010 000 010

011 100 010

100 011 110

010 010 000

001 110 011

000 001 100

000 111 101

000 100 001

100 100 010 -> New

101 000 010 -> New

010 111 110 -> New

100 110 000 -> New

111 010 011 -> New

110 101 100 -> New

110 011 101 -> New

110 000 001 -> New

Last

**Step 3:**

Visit object ตัวแรกที่อยู่ใน queue (010 000 010) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ ดังนี้

* 100 100 010 กดสวิตช์ (0, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 000 101 010 กดสวิตช์ (0, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 001 001 010 กดสวิตช์ (0, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 110 110 110 กดสวิตช์ (1, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 000 111 000 กดสวิตช์ (1, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 011 011 011 กดสวิตช์ (1, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 010 100 100 กดสวิตช์ (2, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 010 010 101 กดสวิตช์ (2, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 010 001 001 กดสวิตช์ (2, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph

หลังจากนั้น pop object ที่ visited ออก

Cont.

Queue ปัจจุบัน:

First

010 000 010 Visited (pop)

011 100 010

100 011 110

010 010 000

001 110 011

000 001 100

000 111 101

000 100 001

100 100 010

101 000 010

010 111 110

100 110 000

111 010 011

110 101 100

110 011 101

110 000 001

001 001 010 -> New

110 110 110 -> New

000 111 000 -> New

011 011 011 -> New

010 100 100 -> New

010 010 101 -> New

010 001 001 -> New

Last

**Step 4:**

Visit object ตัวแรกที่อยู่ใน queue (011 100 010) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ ดังนี้

* 101 000 010 กดสวิตช์ (0, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 001 001 010 กดสวิตช์ (0, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 000 101 010 กดสวิตช์ (0, 2) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 111 010 110 กดสวิตช์ (1, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 001 011 000 กดสวิตช์ (1, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 010 111 011 กดสวิตช์ (1, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 011 000 100 กดสวิตช์ (2, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 011 110 101 กดสวิตช์ (2, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 011 101 001 กดสวิตช์ (2, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph

หลังจากนั้น pop object ที่ visited ออก

Cont.

Queue ปัจจุบัน:

First

011 100 010 Visited (pop)

100 011 110

010 010 000

001 110 011

000 001 100

000 111 101

000 100 001

100 100 010

101 000 010

010 111 110

100 110 000

111 010 011

110 101 100

110 011 101

110 000 001

001 001 010

110 110 110

000 111 000

011 011 011

010 100 100

010 010 101

010 001 001

111 010 110 -> New

001 011 000 -> New

010 111 011 -> New

011 000 100 -> New

011 110 101 -> New

011 101 001 -> New

Last

Cont.

**Step 5:**

Visit object ตัวแรกที่อยู่ใน queue (100 011 110) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ ดังนี้

* 010 111 110 กดสวิตช์ (0, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 110 110 110 กดสวิตช์ (0, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 111 010 110 กดสวิตช์ (0, 2) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 000 101 010 กดสวิตช์ (1, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 110 100 100 กดสวิตช์ (1, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 101 000 111 กดสวิตช์ (1, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 100 111 000 กดสวิตช์ (2, 0) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 100 001 001 กดสวิตช์ (2, 1) ⟶ กระดานไม่ซ้ำ add object ลง graph
* 100 010 101 กดสวิตช์ (2, 2) ⟶ กระดานไม่ซ้ำ add object ลง graph

หลังจากนั้น pop object ที่ visited ออก

Cont.

Queue ปัจจุบัน:

First

100 011 110 Visited (pop)

010 010 000

001 110 011

000 001 100

000 111 101

000 100 001

100 100 010

101 000 010

010 111 110

100 110 000

111 010 011

110 101 100

110 011 101

110 000 001

001 001 010

110 110 110

000 111 000

011 011 011

010 100 100

010 010 101

010 001 001

111 010 110

001 011 000

010 111 011

011 000 100

011 110 101

011 101 001

110 100 100 -> New

101 000 111 -> New

Cont.

100 111 000 -> New

100 001 001 -> New

100 010 101 -> New

Last

Process keeps running.

∙

∙

∙

**Step 255:**

Visit object ตัวแรกที่อยู่ใน queue (101 111 101) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ ดังนี้

* 011 011 101 กดสวิตช์ (0, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 111 010 101 กดสวิตช์ (0, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 110 110 101 กดสวิตช์ (0, 2) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 001 001 001 กดสวิตช์ (1, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 111 000 111 กดสวิตช์ (1, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 100 100 100 กดสวิตช์ (1, 2) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 101 011 011 กดสวิตช์ (2, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 101 101 010 กดสวิตช์ (2, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 101 110 110 กดสวิตช์ (2, 2) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue

หลังจากนั้น pop object ที่ visited ออก

Queue ปัจจุบัน:

First

101 111 101 Visited (pop)

011 110 011

Last

**Step 256:**

Visit object ตัวแรกที่อยู่ใน queue (011 110 011) และ ทดลองกดสวิตช์ของมันทีละ 1 ตัว จะทำให้เกิด object ตัวใหม่ ดังนี้

* 101 010 011 กดสวิตช์ (0, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 001 011 011 กดสวิตช์ (0, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 000 111 011 กดสวิตช์ (0, 2) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 111 000 111 กดสวิตช์ (1, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 001 001 001 กดสวิตช์ (1, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 010 101 010 กดสวิตช์ (1, 2) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 011 010 101 กดสวิตช์ (2, 0) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 011 100 100 กดสวิตช์ (2, 1) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue
* 011 111 000 กดสวิตช์ (2, 2) ⟶ กระดานซ้ำ ไม่ add object ลง graph และ queue

หลังจากนั้น pop object ที่ visited ออก

Queue ปัจจุบัน:

011 110 011 Visited (pop)

Method ยังไม่พบกระดานที่ถูกต้อง แต่ว่าภายใน queue ไม่เหลือ object อยู่ภายในแล้ว method จึง return null กลับไป

**สรุป**

จาก demo 5 จะสังเกตเห็นได้ว่าไม่มีวิธีการปิดไฟทุกดวงในกระดาน เพราะว่าเมื่อทำการ run algorithms ไปเรื่อย ๆ แล้ว processing queue ไม่มีกระดานอยู่เลย ตามที่กล่าวในตอนอธิบาย algorithm เบื้องต้น การที่ processing queue ไม่มีกระดานอยู่เลยหมายความว่า algorithm ได้ลองทำทุกรูปแบบของการกดสวิตช์ไฟที่เป็นไปได้แล้วไม่พบวิธีที่จะปิดไฟทั้งหมดได้ จึงสรุปได้ว่าไม่มีวิธีปิดไฟทั้งหมดของกระดานเริ่มต้นใน demo 5 ซึ่งหากเปรียบเทียบกับ demo 3 , demo 4 แล้ว จะเห็นว่าทั้งสอง demo จะเจอวิธีที่จะปิดไฟทั้งหมดก่อนที่ processing queue จะไม่มีกระดานหรือจะกล่าวว่า ค้นพบวิธีปิดไฟทั้งหมดก่อนที่จะลองทุกวิธีที่เป็นไปได้จนหมด

**ข้อจำกัดของโปรแกรม**

1. เมื่อขนาดของกระดาน (n) มีขนาดใหญ่มากขึ้น method “solution()” ของโปรแกรมจะทำงานได้ช้าลงมาก
2. จากข้อที่ 1. มีโอกาสที่โปรแกรมอาจไม่สามารถหาคำตอบได้ เนื่องจาก การทำงานที่ใช้พื้นที่ในการค้นหามาก ยิ่ง n มาก ยิ่งใช้พื้นที่เยอะมากขึ้นไปอีก จึงทำให้โปรแกรมที่พื้นที่ไม่เพียงพอต่อการค้นหาคำตอบที่ถูกต้อง

**อ้างอิง**

โปรแกรมจำลองเกม Lights Out นี้ถูกเขียนด้วยตัวผู้จัดทำเอง โดยมีการศึกษา Graph data structure, วิธีการทำ Breadth First Search, และ การตกแต่ง output ดังนี้

1. <https://www.youtube.com/watch?v=oDqjPvD54Ss>
2. <https://www.geeksforgeeks.org/breadth-first-search-or-bfs-for-a-graph/>
3. <https://www.javatpoint.com/java-graph>
4. [How to Print Colored Text in Java Console? - GeeksforGeeks](https://www.geeksforgeeks.org/how-to-print-colored-text-in-java-console/)