

Group Project 1

EGCO221

6413108 นางสาวพาลาภ วสุวัต

6413211 นายกอบกฤษ เรืองสุริยกิจ

คู่มือการเล่น

เกี่ยวกับโปรแกรม

โปรแกรมนี้เป็นโปรแกรมจำลองเกมปริศนาที่มีชื่อว่า “Cindy’s puzzle” เกมนี้จะใช้กระดานเป็นแนวยาว มีความยาวเป็น $2n + 1$ โดยที่ n เป็นจำนวนเต็มของจำนวนลูกแก้วสีขาว โดยบนกระดานจะมีลูกแก้ว 2 สี คือ ลูกแก้วสีดำ ซึ่งจะเรียงกันเป็นกลุ่มอยู่ทางด้านซ้ายของกระดาน และ ลูกแก้วสีขาว ซึ่งจะเรียงกันเป็นกลุ่มอยู่ทางด้านขวาของกระดาน นอกจากนี้ ระหว่างกลุ่มของลูกแก้วคนละสีจะถูกแบ่งด้วยช่องว่างดังภาพ

```
Enter number of white marbles (or enter 0 to exit): 2
Initial >>

  ----
  |  |  |  |  |  |
  | w0 | w1 |  | b0 | b1 |
  |----|----|----|----|

Enter marble ID or 'a' to switch to auto mode:
```

ภาพที่ 1 กระดานเริ่มต้น

เป้าหมายของเกมปริศนาคือ การสลับที่ระหว่างกลุ่มของลูกแก้วสีดำ และ กลุ่มของลูกแก้วสีขาว เป็นดังภาพ

```
Step 9 >>

  ----
  |  |  |  |  |  |
  | b0 | b1 |  | w0 | w1 |
  |----|----|----|----|

Done !!!
```

ภาพที่ 2 กระดานสิ้นสุด

วิธีการใช้งานโปรแกรม

1. เมื่อเปิดโปรแกรมขึ้นมา โปรแกรมจะให้ผู้เล่นกรอกจำนวนของลูกแก้วสีขาว (n) ที่ต้องการเพื่อสร้างเกมกระดานขนาด $2n + 1$ โดยที่ n ต้องมีขนาดมากกว่า 2 (หากต้องการออกจากโปรแกรมให้ใส่ n เป็น 0)
2. หลังจากสร้างกระดานเรียบร้อยแล้ว โปรแกรมจะเริ่มให้ผู้เล่นกรอกชื่อของลูกแก้วที่ต้องการจะขยับ โดยลูกแก้วแต่ละสีจะสามารถเคลื่อนที่ได้ดังนี้
ลูกแก้วสีดำ จะเคลื่อนที่ไปทางขวาเท่านั้น สามารถกระโดดข้ามลูกแก้วสีขาว 1 ลูก ได้

```
Initial >>

  ----
  |   |   |   |   |   |
  | w0 | w1 |  _ | b0 | b1 |
  |----|----|----|----|
                                <

Enter marble ID or 'a' to switch to auto mode: b0

Step 1 >>

  ----
  |   |   |   |   |   |
  | w0 | w1 | b0 |  _ | b1 |
  |----|----|----|----|
  >>> b0 Move Left

Enter marble ID or 'a' to switch to auto mode:
```

ภาพที่ 3 ลูกแก้วสีดำขยับไปทางซ้าย

```
Step 1 >>

  ----
  |   |   |   |   |   |
  | w0 |  _ | w1 | b0 | b1 |
  |----|----|----|----|
                                <

  >>> w1 Move Right

Enter marble ID or 'a' to switch to auto mode: b0

Step 2 >>

  ----
  |   |   |   |   |   |
  | w0 | b0 | w1 |  _ | b1 |
  |----|----|----|----|
  >>> b0 Jump Left
```

ภาพที่ 4 ลูกแก้วสีดำกระโดดไปทางซ้าย

ลูกแก้วสีขาว เคลื่อนที่ไปทางซ้ายเท่านั้น สามารถกระโดดข้ามลูกแก้วสีดำ 1 ลูก ได้

```
Initial >>

  ----
  |   |   |   |   |   |
  | w0 | w1 |  _ | b0 | b1 |
  |----|----|----|----|
  >>> w1 Move Right

Enter marble ID or 'a' to switch to auto mode: w1

Step 1 >>

  ----
  |   |   |   |   |   |
  | w0 |  _ | w1 | b0 | b1 |
  |----|----|----|----|
  >>> w1 Move Right

Enter marble ID or 'a' to switch to auto mode:
```

ภาพที่ 5 ลูกแก้วสีขาวขยับไปทางขวา

```
Step 1 >>

  ----
  |   |   |   |   |   |
  | w0 | w1 | b0 |  _ | b1 |
  |----|----|----|----|
  >>> b0 Move Left

Enter marble ID or 'a' to switch to auto mode: w1

Step 2 >>

  ----
  |   |   |   |   |   |
  | w0 |  _ | b0 | w1 | b1 |
  |----|----|----|----|
  >>> w1 Jump Right
```

ภาพที่ 6 ลูกแก้วสีขาวกระโดดไปทางขวา

* มีข้อแม้ว่าลูกแก้วสีเดียวกันจะไม่สามารถกระโดดข้ามกันเองได้

3. หากผู้เล่นไม่ต้องการเล่นต่อ หรือ ต้องการให้โปรแกรมเล่นต่อแบบอัตโนมัติ ให้พิมพ์ “A” หรือ “a” เพื่อแทนการเรียก Auto mode
4. หากแก้ปริศนาเรียบร้อยแล้ว หรือไม่สามารถแก้ปริศนาต่อได้ โปรแกรมจะเริ่มทำงานใหม่ ดังข้อ 1.

Data structures

1. ArrayList

ข้อมูลที่เก็บ : String ของ marble (ex. w0 b1)

เหตุผลที่เลือกใช้ :

- ✖ เนื่องจากการ move หรือ jump marble ในแต่ละครั้งจะต้องมีการเข้าถึงตำแหน่งของข้อมูล ก่อนถึงจะทราบว่า marble ในตำแหน่งนั้นเป็น black หรือ white เพื่อจะทำการเช็คเงื่อนไขในการขยับ ทำให้ในโปรแกรมนี้อาจมีการ access ข้อมูลค่อนข้างบ่อย ดังนั้นการใช้ ArrayList ซึ่งเป็น array base ทำให้สามารถ access ข้อมูลได้อย่างรวดเร็ว
- ✖ ในโปรแกรมนี้นี้มีขนาดที่แน่นอน และไม่ได้มีการ insert ข้อมูล ณ ตำแหน่งใด ๆ ลงใน data structure มีแค่การเก็บข้อมูลในตอนเริ่มโปรแกรม ซึ่งทำให้ข้อเสียในการ insert ข้อมูลของ ArrayList ที่มี Asymptotic runtime ค่อนข้างมาก การใช้งาน ArrayList ในโปรแกรมจึงไม่ต้องการคำนึงถึงข้อเสียนี้

2. ArrayDeque (Stack)

ข้อมูลที่เก็บ : Object ของ class Table

เหตุผลที่เลือกใช้ :

- ✖ เนื่องด้วย Backtracking เป็นการทดลองหาทุก path ที่นำไปยังผลลัพธ์ได้ โดยระหว่างทางนั้นอาจเจอ path ที่ถูกต้องหรือไม่ก็ได้ หากพบว่า path ที่เลือกไม่สามารถนำไปสู่ผลลัพธ์ได้ก็จะไม่ทำต่อและย้อนกลับไปหา path ที่สามารถไปต่อได้ ดังนั้น ArrayDeque จึงเป็นตัวเลือกที่ดีในการเก็บ path เพราะ สามารถ access และ pop ข้อมูลตัวสุดท้ายได้ทันทีเมื่อพบว่า path ที่เลือกนั้นไม่ถูกต้อง
- ✖ ระหว่างการค้นหา path ที่ถูกต้องทำให้เกิดการ insert และ delete node หลายตัว ซึ่งไม่สามารถระบุได้ว่าจะมีจำนวนมากน้อยเท่าไร ArrayDeque ที่สามารถขยายขนาด push และ pop แต่ละ node ด้านท้ายสุดของ path ได้ จึงมีความเหมาะสม
- ✖ การแสดงผลลัพธ์ที่ถูกต้องหลังจากการค้นหา path จะต้องแสดงจากจุดเริ่มต้นของ path หรือ node ตัวแรกไปจนถึงตัวสุดท้าย ArrayDeque ซึ่งสามารถ pop node ตัวแรกสุดได้จึงทำให้การแสดงผลลัพธ์หลังจากพบ path ที่ถูกต้องแล้วมีความเร็วมากกว่า Data structure ตัวอื่น ๆ

Class Table : เป็น class ที่เปรียบเสมือนกระดานขนาด $2n + 1$ โดยจะมี ArrayList ที่เก็บกระดานขนาด $2n + 1$ เอาไว้พร้อมกับตำแหน่งของ marble แต่ละตัวและมี method ที่ไว้ใช้ในการจัดการกับ marble

เช่น method สำหรับเช็ค ว่า marble ขยับได้หรือไม่แล้วถ้าขยับได้จะขยับแบบ jump หรือ move, method สำหรับขยับ marble, method สำหรับเช็ค ว่า marble อยู่ถูกตำแหน่งในกระดานครบหรือยัง

Backtracking algorithm

Backtracking algorithm ถูกใช้ในส่วนของ method “solution()” ซึ่งจะเริ่มต้นทำจากกระดานล่าสุดที่ผู้เล่นได้เล่นไว้ โดยการตรวจสอบ marble 4 ตัว รอบช่องว่าง (2 ตัวด้านหน้า และ ด้านหลังช่องว่าง) จากด้านซ้ายไปขวา ว่ามีตัวไหนสามารถขยับได้อยู่บ้าง หากมีก็จะทำการ move หรือ jump marble ตัวนั้น จากนั้นตรวจสอบ marble 4 ตัว ถัดไปที่อยู่รอบช่องว่างที่เกิดขึ้นใหม่ไปเรื่อย ๆ จนกว่าจะไม่พบลูกแก้วที่สามารถขยับได้อีก

Method นี้จะหยุดทำงานเมื่อค้นพบ path ที่ถูกต้อง โดยการนำ path ที่ค้นหานั้นไปเปรียบเทียบกับผลลัพธ์ที่ต้องการ เช่น หากตารางมีขนาด $2n + 1 = 5$ จะมีผลลัพธ์เป็น b0 b1 __ w0 w1 หรือหากพบว่ากระดานดังกล่าวไม่สามารถขยับลูกแก้วใด ๆ ได้อีก method จะทำการ backtrack กลับไปยังกระดานก่อนหน้าที่จะขยับ ซึ่ง Data structure ที่ถูกนำมาช่วยในการทำงานคือ ArrayDeque ที่สามารถ access, push node ที่กำลังทำ backtracking, pop ข้อมูลในส่วนท้ายเมื่อไม่สามารถขยับลูกแก้วได้ เพื่อช่วยในการค้นหา path ที่ถูกต้อง และ pop ข้อมูลในส่วนหัว เพื่อใช้ในการแสดงผลของ path ที่ค้นหามา

การสรุปว่ากระดานที่ผู้เล่นส่งเข้ามาใน auto mode ไม่มีคำตอบนั้นจะเกิดขึ้นเมื่อ algorithm ทำการ backtracking traversal ทุก ๆ path ที่สามารถเกิดขึ้นได้จนครบแล้วยังไม่ปรากฏกระดานที่ถูกต้องตามเงื่อนไขที่ตั้งไว้ (ยังไม่เข้า base case condition) algorithm จะสรุปได้ว่ากระดานที่ผู้เล่นส่งมาไม่มี solution ใดที่สามารถขยับลูกแก้วแล้วได้กระดานที่มีลูกแก้ววางเรียงกันตามเงื่อนไขที่กำหนด

ตัวอย่างการ Backtracking : ใช้ method “testcase(initial state)” ใน main บรรทัดที่ 34 จำลองการทำ Backtracking โดยใช้ตัวเลขของ test case ที่จะจำลองเข้าไป

Initial State (1) w0 w1 _ b0 b1

ลำดับ	Board	Stack
0	<pre>Initial >> ----- w0 w1 _ b0 b1 ----- ----- ----- ----- </pre>	<div>Top</div> <div>w0 w1 _ b0 b1</div>
1	<pre>Step 1 >> ----- w0 _ w1 b0 b1 ----- ----- ----- ----- </pre>	<div>Top</div> <div>w0 _ w1 b0 b1</div> <div>w0 w1 _ b0 b1</div>
2	<pre>Step 2 >> ----- _ w0 w1 b0 b1 ----- ----- ----- ----- No possible move for this state , backtrack to previous state</pre>	<div>Top</div> <div>_ w0 w1 b0 b1</div> <div>w0 _ w1 b0 b1</div> <div>w0 w1 _ b0 b1</div>
3	<pre>Step 3 >> ----- w0 _ w1 b0 b1 ----- ----- ----- ----- </pre>	<div>Top</div> <div>w0 _ w1 b0 b1</div> <div>w0 w1 _ b0 b1</div>
4	<pre>Step 4 >> ----- w0 b0 w1 _ b1 ----- ----- ----- ----- </pre>	<div>Top</div> <div>w0 b0 w1 _ b1</div> <div>w0 _ w1 b0 b1</div> <div>w0 w1 _ b0 b1</div>

ลำดับ	Board	Stack
5	<div>Step 5 >></div> <div>-----</div> <div> </div> <div> w0 b0 _ w1 b1 </div> <div> _____ _____ _____ _____ </div>	Top
		w0 b0 _ w1 b1
		w0 b0 w1 _ b1
		w0 _ w1 b0 b1
		w0 w1 _ b0 b1
6	<div>Step 6 >></div> <div>-----</div> <div> </div> <div> _ b0 w0 w1 b1 </div> <div> _____ _____ _____ _____ </div>	Top
		_ b0 w0 w1 b1
		w0 b0 _ w1 b1
		w0 b0 w1 _ b1
		w0 _ w1 b0 b1
7	<div>Step 7 >></div> <div>-----</div> <div> </div> <div> b0 _ w0 w1 b1 </div> <div> _____ _____ _____ _____ </div> <div>No possible move for this state , backtrack to previous state</div>	Top
		b0 _ w0 w1 b1
		_ b0 w0 w1 b1
		w0 b0 _ w1 b1
		w0 b0 w1 _ b1
		w0 _ w1 b0 b1
		w0 w1 _ b0 b1

ลำดับ	Board	Stack
8	<pre>Step 8 >> ----- _ b0 w0 w1 b1 _____ _____ _____ _____ No possible move for this state , backtrack to previous state</pre>	Top
		_ b0 w0 w1 b1
		w0 b0 _ w1 b1
		w0 b0 w1 _ b1
		w0 _ w1 b0 b1
		w0 w1 _ b0 b1
9	<pre>Step 9 >> ----- w0 b0 _ w1 b1 _____ _____ _____ _____ </pre>	Top
		w0 b0 _ w1 b1
		w0 b0 w1 _ b1
		w0 _ w1 b0 b1
10	<pre>Step 10 >> ----- w0 b0 b1 w1 _ _____ _____ _____ _____ </pre>	Top
		w0 b0 b1 w1 _
		w0 b0 _ w1 b1
		w0 b0 w1 _ b1
		w0 _ w1 b0 b1
		w0 w1 _ b0 b1


ลำดับ	Board	Stack
14	<pre>Step 14 >> ----- w0 b0 w1 _ b1 _____ _____ _____ _____ _____ </pre>	Top
		w0 b0 w1 _ b1
		w0 _ w1 b0 b1
		w0 w1 _ b0 b1
15	<pre>Step 15 >> ----- w0 b0 w1 b1 _ _____ _____ _____ _____ _____ </pre>	Top
		w0 b0 w1 b1 _
		w0 b0 w1 _ b1
		w0 _ w1 b0 b1
16	<pre>Step 16 >> ----- w0 b0 _ b1 w1 _____ _____ _____ _____ _____ </pre>	Top
		w0 b0 _ b1 w1
		w0 b0 w1 b1 _
		w0 b0 w1 _ b1
		w0 _ w1 b0 b1
		w0 w1 _ b0 b1

ลำดับ	Board	Stack
17	<div>Step 17 >></div> <div>-----</div> <div> </div> <div> _ _ b0 w0 b1 w1 </div> <div> ----- ----- ----- ----- </div>	Top
		_ b0 w0 b1 w1
		w0 b0 _ b1 w1
		w0 b0 w1 b1 _
		w0 b0 w1 _ b1
		w0 _ w1 b0 b1
		w0 w1 _ b0 b1
18	<div>Step 18 >></div> <div>-----</div> <div> </div> <div> b0 _ _ w0 b1 w1 </div> <div> ----- ----- ----- ----- </div>	Top
		b0 _ w0 b1 w1
		_ b0 w0 b1 w1
		w0 b0 _ b1 w1
		w0 b0 w1 b1 _
		w0 b0 w1 _ b1
		w0 _ w1 b0 b1
		w0 w1 _ b0 b1

ลำดับ	Board	Stack
19	<pre>Step 19 >> ---- ---- ---- ---- ---- b0 b1 w0 _ w1 ---- ---- ---- ---- ---- </pre>	Top
		b0 b1 w0 _ w1
		b0 _ w0 b1 w1
		_ b0 w0 b1 w1
		w0 b0 _ b1 w1
		w0 b0 w1 b1 _
		w0 b0 w1 _ b1
		w0 _ w1 b0 b1
		w0 w1 _ b0 b1
20	<pre>Step 20 >> ---- ---- ---- ---- ---- b0 b1 _ w0 w1 ---- ---- ---- ---- ---- Done !!!</pre>	Top
		b0 b1 _ w0 w1
		b0 b1 w0 _ w1
		b0 _ w0 b1 w1
		_ b0 w0 b1 w1
		w0 b0 _ b1 w1
		w0 b0 w1 b1 _
		w0 b0 w1 _ b1
		w0 _ w1 b0 b1
		w0 w1 _ b0 b1

Initial State (2) w0 b0 _ w1 b1

ลำดับ	Board	Stack
0	<p>Initial >></p> <pre> ----- w0 b0 _ w1 b1 _____ _____ _____ _____ </pre>	<div>Top</div> <div>w0 b0 _ w1 b1</div>
1	<p>Step 1 >></p> <pre> ----- _ b0 w0 w1 b1 _____ _____ _____ _____ </pre>	<div>Top</div> <div>_ b0 w0 w1 b1</div> <div>w0 b0 _ w1 b1</div>
2	<p>Step 2 >></p> <pre> ----- b0 _ w0 w1 b1 _____ _____ _____ _____ </pre> <p>No possible move for this state , backtrack to previous state</p>	<div>Top</div> <div>b0 _ w0 w1 b1</div> <div>_ b0 w0 w1 b1</div> <div>w0 b0 _ w1 b1</div>
3	<p>Step 3 >></p> <pre> ----- _ b0 w0 w1 b1 _____ _____ _____ _____ </pre> <p>No possible move for this state , backtrack to previous state</p>	<div>Top</div> <div>_ b0 w0 w1 b1</div> <div>w0 b0 _ w1 b1</div>
4	<p>Step 4 >></p> <pre> ----- w0 b0 _ w1 b1 _____ _____ _____ _____ </pre>	<div>Top</div> <div>w0 b0 _ w1 b1</div>
5	<p>Step 5 >></p> <pre> ----- w0 b0 b1 w1 _ _____ _____ _____ _____ </pre>	<div>Top</div> <div>w0 b0 b1 w1 _</div> <div>w0 b0 _ w1 b1</div>

ลำดับ	Board	Stack
6	<pre>Step 6 >> ---- w0 b0 b1 _ w1 ---- ---- ---- ---- ---- No possible move for this state , backtrack to previous state</pre>	Top
		w0 b0 b1 _ w1
		w0 b0 b1 w1 _
		w0 b0 _ w1 b1
7	<pre>Step 7 >> ---- w0 b0 b1 w1 _ ---- ---- ---- ---- ---- No possible move for this state , backtrack to previous state</pre>	Top
		w0 b0 b1 w1 _
		w0 b0 _ w1 b1
8	<pre>Step 8 >> ---- w0 b0 _ w1 b1 ---- ---- ---- ---- ---- No possible move for this state , backtrack to previous state No solution !!!</pre>	Top
		w0 b0 _ w1 b1
		
		Top

ข้อจำกัดของโปรแกรม

1. เมื่อขนาดของจำนวน marble มากขึ้น auto mode จะทำงานได้ช้าลงมาก
2. Auto mode สามารถหา path ที่ถูกต้องได้แค่ path เดียว และ อาจไม่ใช่ path ที่ดีที่สุด

อ้างอิง

1. Computing for Beginners: Backtracking Algorithms : Explaining Cindy's Puzzle with a C++ Program
2. Backtracking (Think Like a Programmer) - YouTube
3. Introduction to Backtracking - Data Structure and Algorithm Tutorials - GeeksforGeeks