



รายงาน

เรื่อง N-Queens

เสนอ

รศ.ดร.รังสีพรรณ มฤคทัต

จัดทำโดย

นายชนชล จันทร์สว่าง 6313123

นายกริชชัย เจริญเรืองเลิศ 6313171

นายธนกฤต ฉันท์พลากร 6313215

รายงานนี้เป็นส่วนหนึ่งของวิชา Structure and Algorithm

คำนำ

รายงานฉบับนี้เป็นส่วนหนึ่งของรายวิชา Data Structure and Algorithm (EGCO221) จัดทำขึ้นเพื่อใช้ประกอบการอธิบายการทำงานของโปรแกรม N-Queens ประกอบด้วย คู่มือการใช้งานโปรแกรมเบื้องต้น และการอธิบายในส่วนของ Algorithm รวมถึงข้อจำกัด ในการใช้งานโปรแกรม

คณะผู้จัดทำขอขอบพระคุณ รศ.ดร.รังสิพรรณ มฤคทัต ผู้ให้ความรู้ และแนวทางการศึกษา สุดท้ายนี้ทางคณะผู้จัดทำหวังว่ารายงานฉบับนี้จะสามารถเป็นประโยชน์กับผู้อ่านทุกท่าน

คณะผู้จัดทำ

การใช้งานโปรแกรมเบื้องต้น

ใส่เลข 4-8 เพื่อกำหนดขอบเขตของกระดาน

```
Enter N (at least 4) =
```

ถ้าใส่เลขที่เกินขอบเขตที่กำหนดจะให้ใส่ใหม่ไปเรื่อยๆจนกว่าจะใส่ถูก

```
Enter N (at least 4) =
```

```
-5
```

```
Enter N (at least 4) =
```

```
9
```

```
Enter N (at least 4) =
```

```
5
```

```
Board layout
```

	1	2	3	4	5
1	=	=	=	=	=
2	=	=	=	=	=
3	=	=	=	=	=
4	=	=	=	=	=
5	=	=	=	=	=

Place first queen manually (y for yes) ?

บรรทัดนี้ ถ้าเราใส่ค่าที่ไม่ใช่ y หรือ n จะหลุดออกจากการทำงานแล้วโปรแกรมจะถามว่าจะทำต่อหรือไม่

```
Board layout
```

	1	2	3	4	5
1	=	=	=	=	=
2	=	=	=	=	=
3	=	=	=	=	=
4	=	=	=	=	=
5	=	=	=	=	=

```
Place first queen manually (y for yes) ?
```

```
5
```

```
how dare you input wrong chioce?
```

```
Plese~~~~~input only (y for yes / n for no)
```

```
Continue (y for yes)
```

```
|
```

```
Place first queen manually (y for yes) ?
```

```
k;k
```

```
how dare you input wrong chioce?
```

```
Plese~~~~~input only (y for yes / n for no)
```

```
Continue (y for yes)
```

```
|
```

ถึงขั้นตอนที่เราจะใส่ค่าเพื่อที่จะบอกโปรแกรมว่าจะทำต่อหรือไม่
ซึ่งถ้าเราใส่ค่าอื่นที่ไม่ใช่ n หรือ y จะให้ใส่ใหม่ไปเรื่อยๆจนกว่าจะถูก

```
Continue (y for yes)
jghjj
Continue (y for yes)
5
Continue (y for yes)
y
Place first queen manually (y for yes) ?
|
```

ถ้าเราใส่ n โปรแกรมจะใส่ข้อมูลลงตารางให้เรียบร้อยเลย

```
Place first queen manually (y for yes) ?
n
Solution
    1    2    3    4
1    =    =    Q    =
2    Q    =    =    =
3    =    =    =    Q
4    =    Q    =    =
Continue (y for yes)
|
```

แต่ถ้าใส่ y แล้วเรากำหนดแถวกับคอลัมน์ที่ทำให้ไม่สามารถหาวิธีการได้จะขึ้นว่า No solution!!

```
Place first queen manually (y for yes) ?
y
Enter row of first queen =
3
Enter col of first queen =
2
    1    2    3    4    5
1    =    =    =    =    =
2    =    =    =    =    =
3    =    Q    =    =    =
4    =    =    =    =    =
5    =    =    =    =    =
No solutions!!
```

แต่ถ้าเรากำหนดแถวกับคอลัมน์ถูก โปรแกรมจะแสดงผลลัพธ์ที่ถูกออกมา

```
Place first queen manually (y for yes) ?
y
Enter row of first queen =
1
Enter col of first queen =
2
  1   2   3   4   5
1  =   Q   =   =   =
2  =   =   =   =   =
3  =   =   =   =   =
4  =   =   =   =   =
5  =   =   =   =   =
solution
  1   2   3   4   5
1  =   Q   =   =   =
2  =   =   =   Q   =
3  Q   =   =   =   =
4  =   =   Q   =   =
5  =   =   =   =   Q
```

แล้วตอนสุดท้ายใส่ n เพื่อจบโปรแกรม

```
Continue (y for yes)
n
-----
BUILD SUCCESS
```

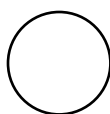
Idea ในการแก้ N-Queen Problem โดยใช้เรื่องของ Tree ซึ่งมีคุณสมบัติที่ความเป็น Recursive Structure เข้ามาช่วยแก้ปัญหา

จะยกตัวอย่างจากใน Demo2 เพื่อง่ายต่อการอธิบาย โดยจะขออธิบายในกรณีที่เราให้โปรแกรมของเราทำการ Auto Fill Queen ทั้งหมด n ตัว โดยเริ่มแรกจะให้ Node ตัวแรกที่เป็น Parent Node ของทุกๆ Node แทน Queen ที่ถูกเติมเข้าไปใน Column ที่ 0 Row ที่ 0 แต่ในความเป็นจริง Queen ที่เราต้องการเติมจะถูกเติมตั้งแต่ Column ที่ 1 ถึง Column ที่ n แต่เนื่องจากเราต้องการ Node ตัวแรกที่เป็นจุดเริ่มต้นในการทำ Recursive Function เพื่อให้ทำการสร้าง Node ลูกต่อไปเรื่อยๆจนกว่าจะเจอคำตอบ หรือจนกว่าจะสร้าง Node ลูกครบทุกตัว จึงขอเริ่มการเติม Queen แบบปลอมๆ ที่ Column ที่ 0 Row ที่ 0 แทน

Step การทำงานของ Program โดยใช้ Demo2 Input แบบ Auto Fill N-Queen

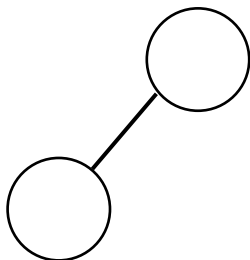
- เติม Queen ลงใน Column ที่ 0 Row ที่ 0

ได้ตารางเปล่ามาเพราะ ใน Board จะเริ่มนับที่ Index = 1



ต่อจากนี้จะมองว่า Depth ของ Tree จะเท่ากับหมายเลขของ Row ที่เราเติม Queen เข้าไป

- เติม Queen Column ที่ 1 Row ที่ 1



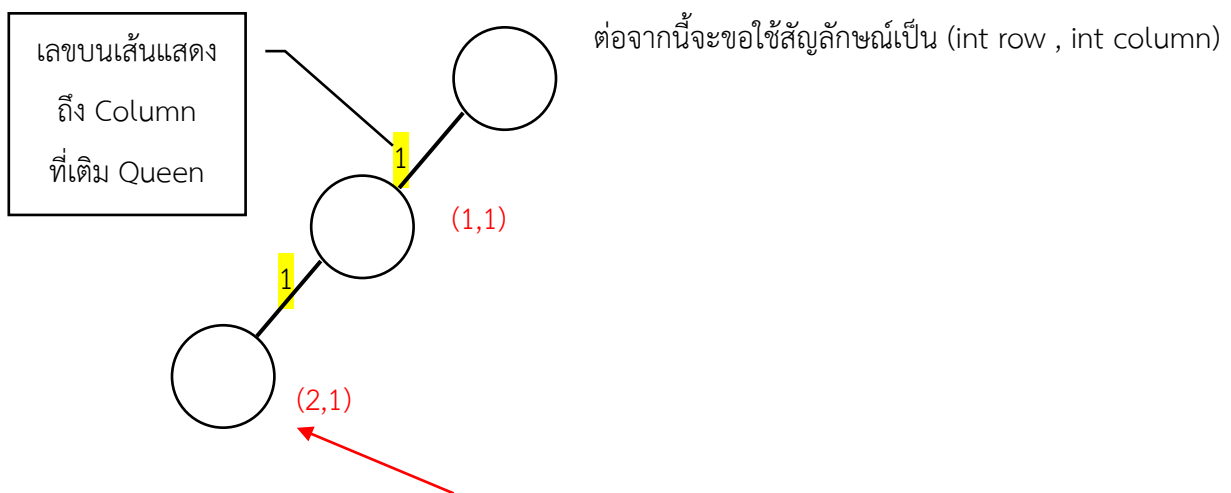
Q			

Step ในการเติม Queen ตัวใหม่ของเราก็คือ จะทำการเติม Queen ลงไปใน Board ที่ Row ถัดไปซึ่งเราจะทำการลองเติม Queen ลงไปในทุกๆ Column ที่ละ Column เพื่อดูว่า Queen ที่เราเติมไปนั้นมันกินกันเองหรือไม่

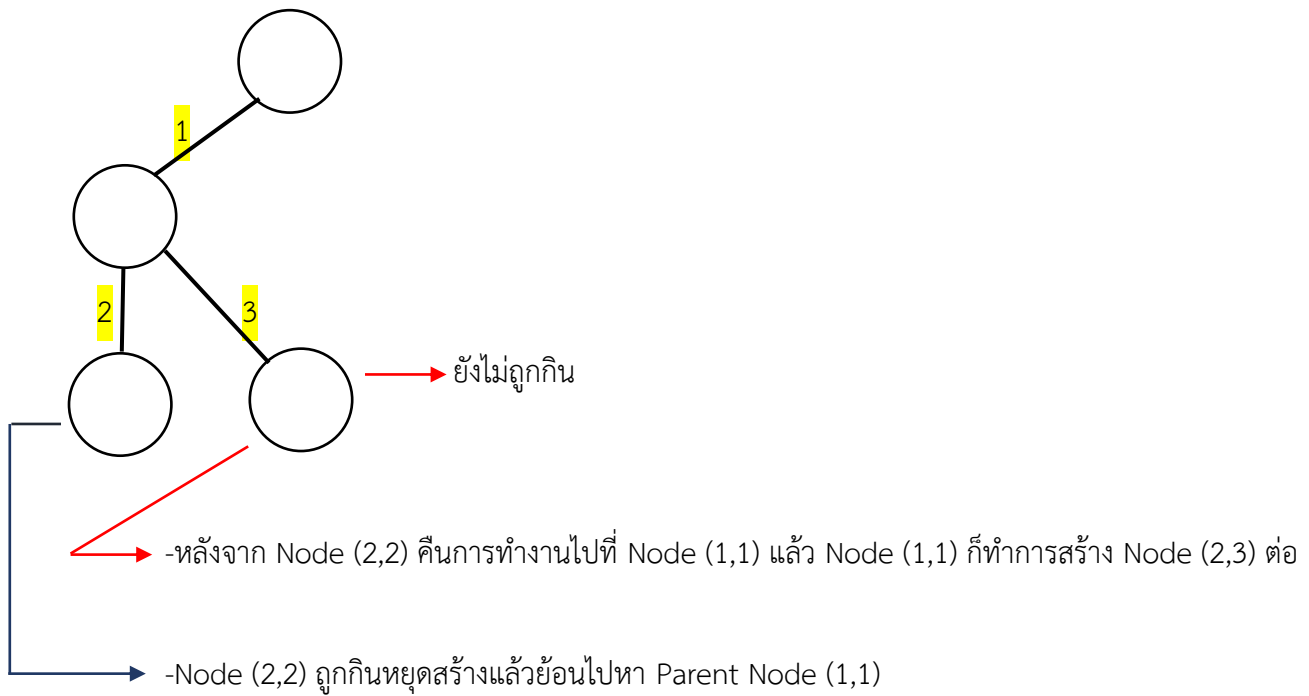
- ลองเติม Queen ทุก Column ใน Row ที่ 2 จนกว่าจะเจอตัวที่ไม่ถูกกิน

Q			
Q	Q	Q	
ถูกกิน			

ถ้า Queen ที่เราใช้ไปใน Board ถูกกินก็ให้เอา Queen ตัวนั้นออกจาก Board ไปด้วย



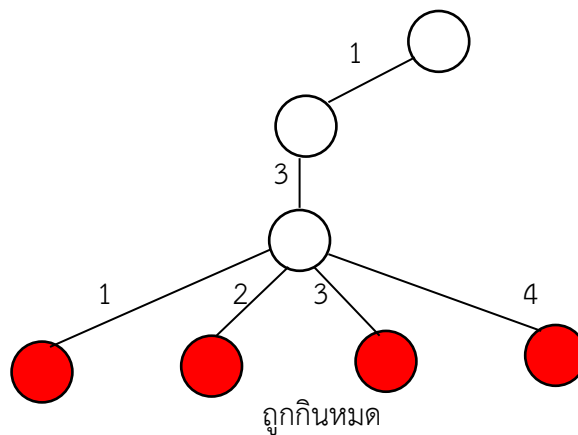
ถูกกินให้หยุดสร้าง Child Node ตรงนี้จากนั้นให้โปรแกรมย้อนกลับไปทำงานที่ Parent Node ที่เป็นคนสร้าง Node (2,1)



ตอนนี้เราเติม Queen ได้ 2 ตัวแล้ว Step ต่อมาจะทำการลองเติม Queen ตัวใหม่ในแถวถัดไป

Q			
		Q	
Q	Q	Q	Q

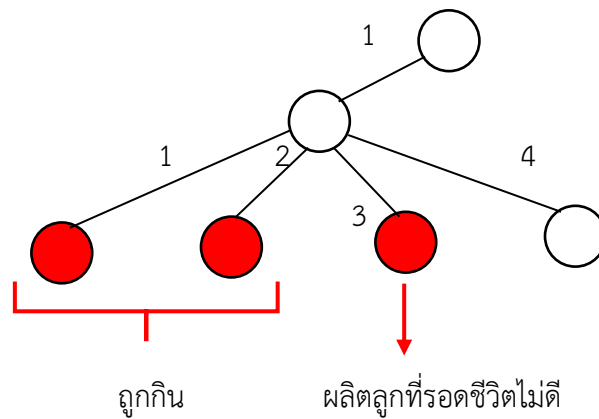
Queen ที่เติมเข้ามาใหม่จะโดนกินหมด และ Algorithm ที่แสดงการถูกกินจะขอลงรายละเอียดทีหลัง



จากภาพตัวนี้ทำให้เราทราบว่า Child Node จะ Return การทำงานไปหา Parent Node ก็ต่อเมื่อ

1. Node ที่สร้างถูก Node Parent กิน
2. Child Node ไม่สามารถผลิตลูกที่ไม่ถูก Parent กินได้โดย Max ของลูกที่สามารถผลิตได้จะเท่ากับ n ตัว แต่ในตัวอย่างนี้ $n=4$ ตัว ดังนั้นจำนวน Child Node มากที่สุดที่ถูกสร้างจาก Parent Node อันเดียวกันเท่ากับ 4 Node

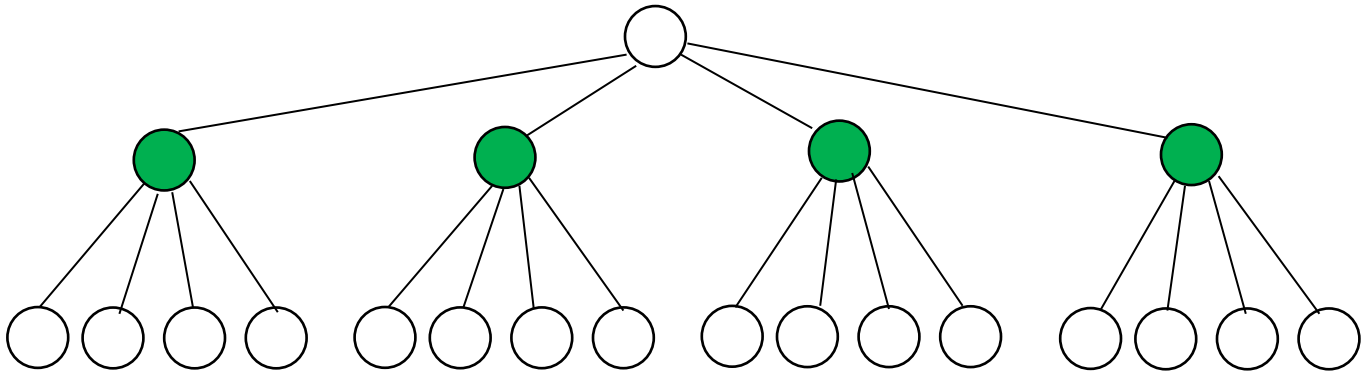
Step ถัดมา เนื่องจาก Node (2,3) ไม่สามารถ ผลิตลูกๆที่ไม่ถูกกินได้ดังนั้น เราจึงต้องถอด Queen (2,3) ออกจาก Board ด้วย



Q			
			Q

จะขอพูดถึงการเติม Queen ถอด Queen ในส่วนของรายละเอียดที่หลัง

จากการลองวาดรูป Tree เพื่อจำลองการทำงานของโปรแกรมทำให้เราทราบ Property ของ Tree ได้ว่า



ในต้นไม้ต้นใหญ่ก็ยังมรต้นไม้ต้นเล็กๆอยู่โดยที่แต่ละต้นก็จะมี Child Node 4 Node เหมือนกัน (ไม่นับ Node ที่อยู่ข้างล่างสุด)

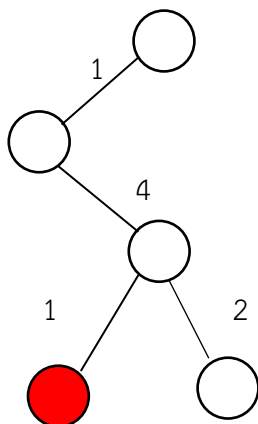
จากที่บรรยายมาข้างต้นทำให้เรารู้ว่า Diagram ที่แสดงการทำงานของ Program นี้มี Property ของ Recursive Structure ซึ่งสามารถใช้การเขียน Program แบบ Recursive Function มาใช้แก้ปัญหาได้

Step ถัดมาลองเติม Queen ที่แถวถัดไป

Q			
			Q
Q	Q		

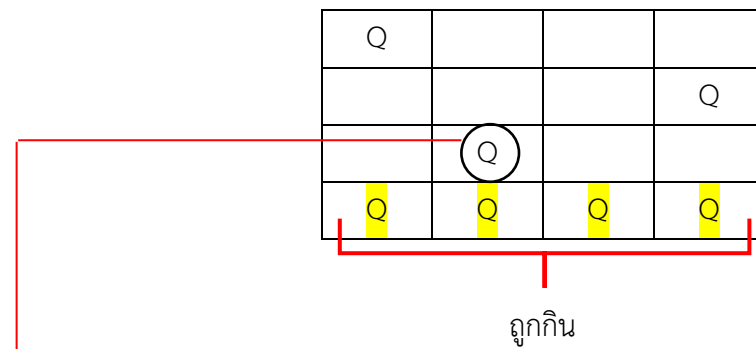
ถูกกิน(ไม่ใช่)

ถ้ารอดให้ใส่ลงไปใน Board

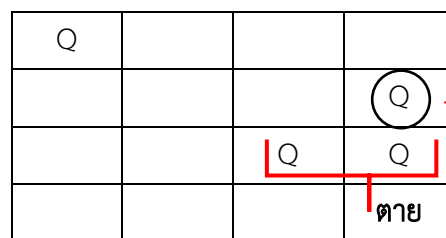
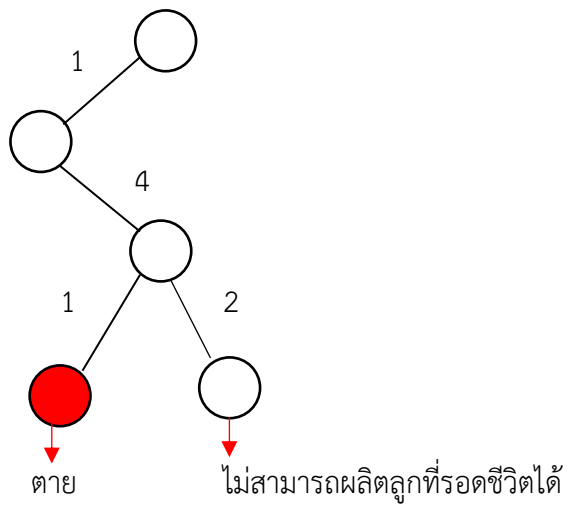


(1) ถูกกินให้คืนการทำงานไปที่ Node (2,4) ให้สร้างลูกตัวใหม่ให้

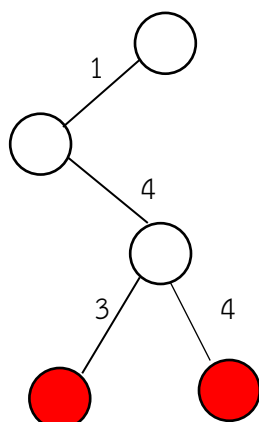
Step ถัดมาเติม Queen ลงในแถวถัดไป



ถูกถอดออกมาเพราะลูกตายหมด

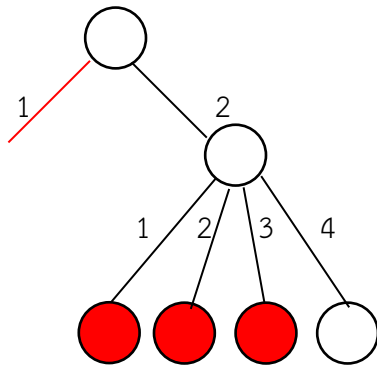


ถูกถอดออกมาเพราะลูกตายหมด



(3) และ (4) ตาย

Step ถัดมาเนื่องจาก Node (1,1) ไม่สามารถผลิตลูกที่รอดชีวิตได้ดังนั้นจึงต้องคืนการทำงานไปที่ Node (0,0) พร้อมทั้งถอด Queen (1,1) ออกจาก Board



	Q		
Q	Q	Q	Q
	ตาย		รอด

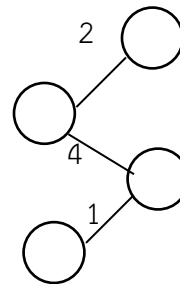
Queen ที่ลองวางแบบหลอกๆ ถ้าถูกกินเราจะไม่เติมลงใน Board

Step ถัดมาลองวาง Queen ที่แถวที่ 3

	Q		
			Q
Q			

วางครั้งแรกแล้วไม่รอดก็ให้เริ่มวาง

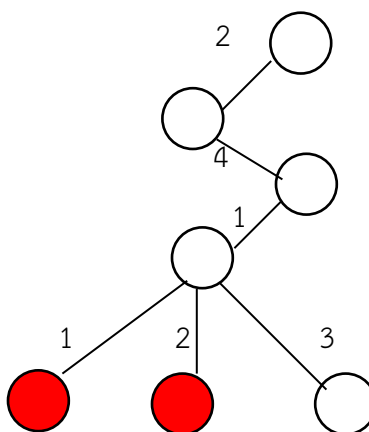
Queen ตัวใหม่ที่แถวถัดไป



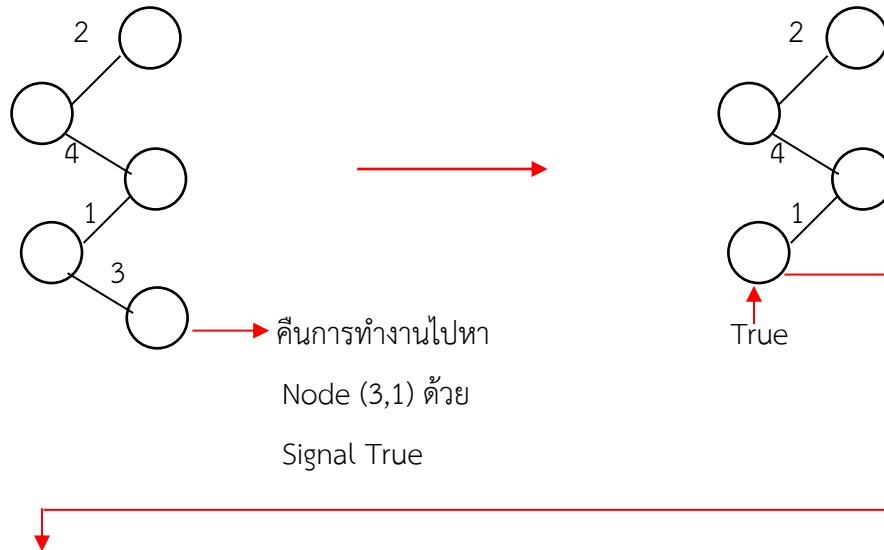
	Q		
			Q
Q			
Q	Q	Q	

โดนกิน

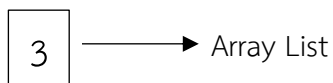
รอด



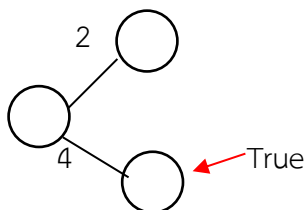
เมื่อ Queen ที่เราเติมไปใน Board มีจำนวนเท่ากับขนาดของ Column หรือ Row ของ Board แล้วเราก็จะหยุดการเติม Queen แล้วให้ Node ตัวสุดท้ายส่ง Signal True ไปหา Parent Node ของมัน



Node (3,1) ไปรับสัญญาณ True ก็ให้ Add เลข Column จาก Child Node ลง Array List มีความยืดหยุ่นในเรื่องของ Size ของ Data Structure เพื่อให้สอดคล้องกับโจทย์ที่ขนาดของปัญหาเท่ากับ n โดยให้ Add เลข 3 ลงไปในตัวท้ายสุด



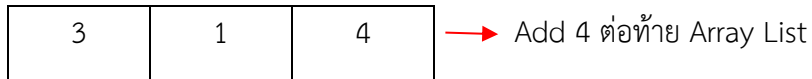
เมื่อ Node (3,1) Add 3 ลงไปใน Array List ก็ให้ Node (3,1) Return ไปหา Parent Node ของมัน นั่นคือ Node (2,4)



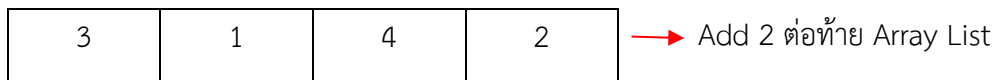
Node (2,4) ทำแบบเดียวกับ Node (3,1)



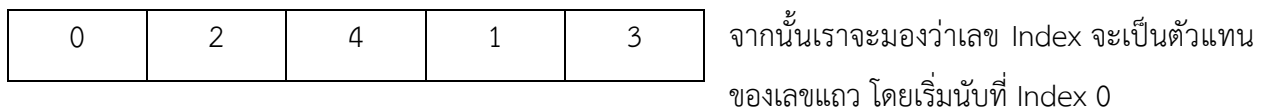
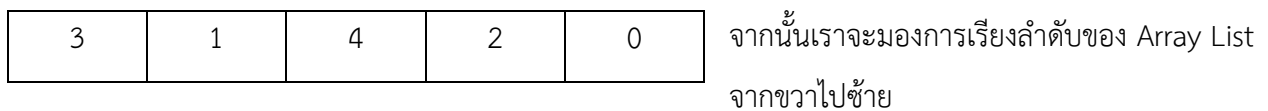
Node (1,2) ทำแบบเดียวกับ Node (2,4)



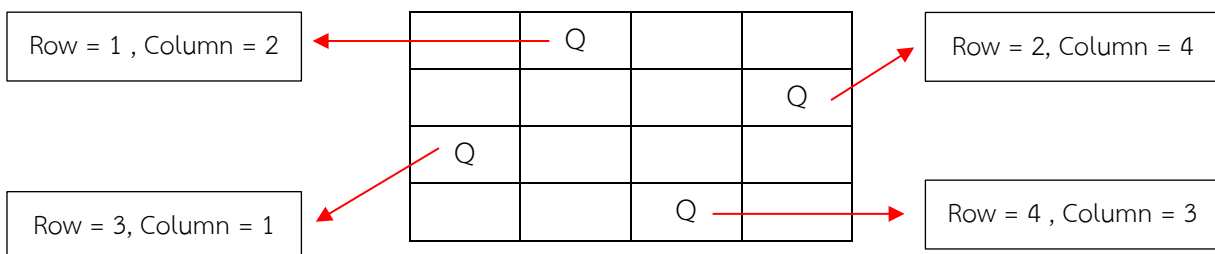
Node (0,0) ทำแบบเดียวกับ Node (1,2)



หลังจากนั้นก็ใส่เงื่อนไขไปว่าถ้า Node ที่ได้รับสัญญาณ True เป็น Node ตัวแรกสุดที่ถูกสร้างหลังจาก
เติมเลข Column ลง Array List ก็ให้เติมเลข Column ของตัวเองต่อท้าย Array List ด้วย



เมื่อเราลองนำเซตคำตอบมาแทน Queen ลงในตารางจะพบว่า



** ในกรณีที่ไมเกิดการ Back Truck เกิดขึ้นหรือก็คือ Size ของ Result Array List เท่ากับ 0 ก็จะเข้าใจว่าไม่มี Solution เกิดขึ้น

จะขอกลับมาถึงการ Add Queen & Remove Queen ลงไปในตารางอีกครั้ง

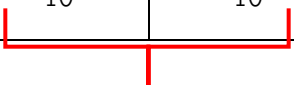
ในตอนนี้เราจะทำการสร้าง Array 2 ตัว ที่คอยเก็บว่า มี Queen Row ไหนบ้างที่ถูกเก็บ และมี Queen Column ไหนบ้างที่ถูกเก็บ โดยเราจะทำการสร้าง Array แบบคู่ขนาน โดย Array ทั้งสองตัวมีขนาดเท่ากับ n

ตัวอย่าง

Q			
		Q	

Column

1	3	-10	-10
---	---	-----	-----



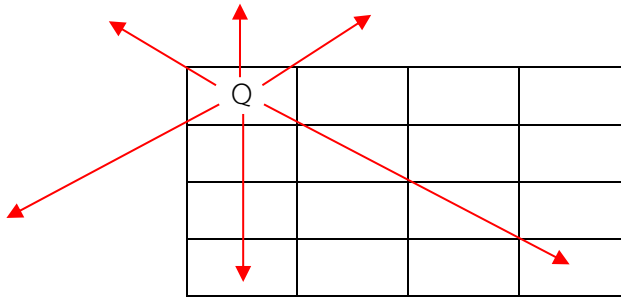
“Column ไหนที่ไม่ถูกเติมให้เติม -10 ไปก่อน”

Row

1	2	-10	-10
---	---	-----	-----

จากรูปก็จะทำให้เราเข้าใจว่า Queen (1,1) และ Queen (2,3) ถูกเก็บลงใน Board

จะขอ Repeat Step จาก Demo 2 อีกครั้ง



Queen (1,1) ไม่ได้ซ้ำ Column กับใดแล้วก็ไม่ได้ใครในแนวทแยง ดังนั้นเติม Queen (1,1) ลงไปในตารางได้

Column

1	-10	-10	-10
---	-----	-----	-----

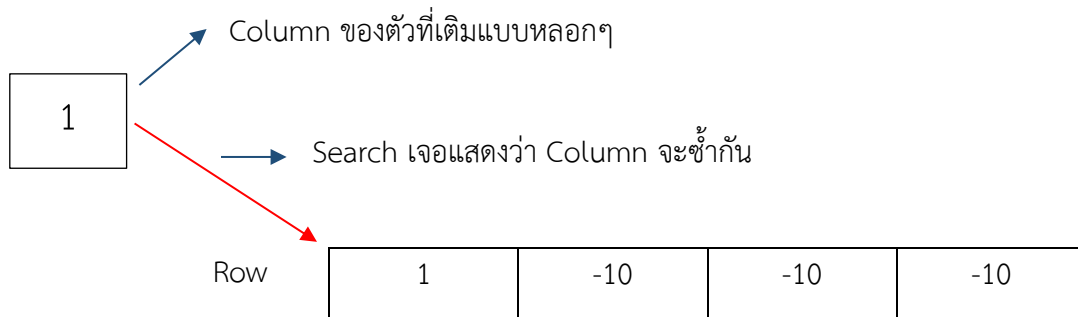
Row

1	-10	-10	-10
---	-----	-----	-----

ไม่มีความจำเป็นต้อง Check ว่า Queen ที่เติมไปซ้ำแถวกันหรือไม่เพราะ Queen ตัวใหม่ที่เราเติมลงในตารางจะเป็นแถวถัดไปเสมอ

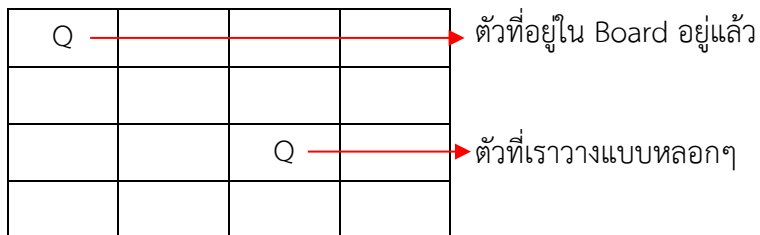
Q			
Q			

การซ้ำ Column จะถูก Check โดยเอาเลข Column ของ Queen ที่เติมแบบหลอกๆไป Check กับสมาชิกของ Array Row ทีละตัวๆ เพื่อ Check ว่า Queen จะถูกกินโดยซ้ำ Column กันไหม

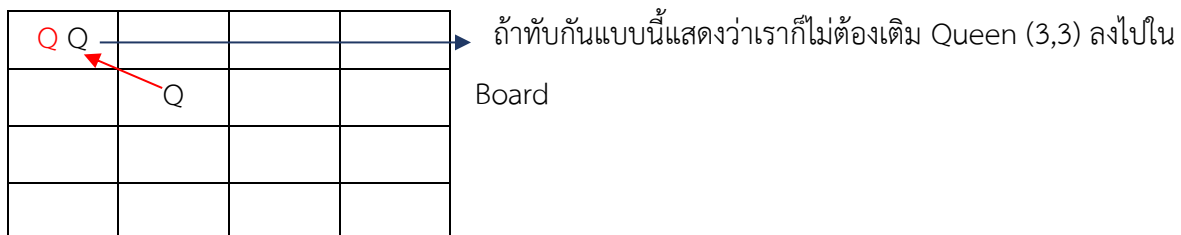
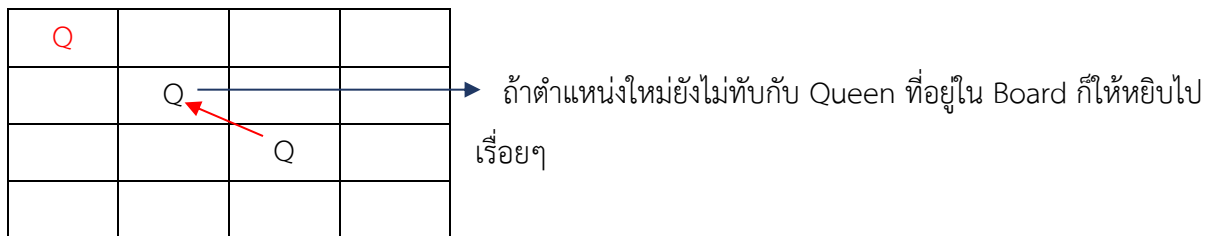


ถ้า Search เจอก็จะไม่ Add Queen ตัวนั้นลงตาราง

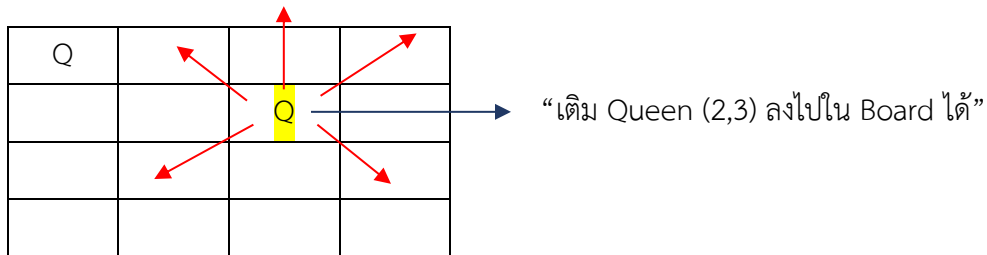
การซ้ำแนวทแยง



โดย Step ของเราคือ จะให้ตัวที่วางแบบหลอกๆ เคลื่อนที่ในแนวทแยงไปที่ละแนวดังนี้



เราจะหยุดขยับ Queen ก็ต่อเมื่อเราขยับ Queen ไปในแนวทแยงครบทั้ง 4 ทิศ แล้วในแต่ละทิศ Queen ของเราจะเคลื่อนที่ไปจนมุมแล้วแต่ก็ยัง不去พบกับ Queen ตัวอื่นที่เติมลงใน Board



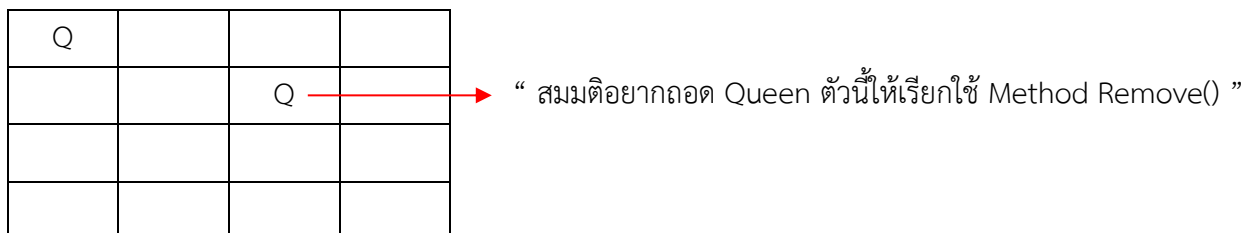
Row

1	2	-10	-10
---	---	-----	-----

Column

1	3	-10	-10
---	---	-----	-----

การถอด Queen ออกจาก Board



โดย Method Remove จะทำการเปลี่ยนแปลงค่าใน Array ของ Row และ Array ของ Column ในตัวขาวสุดที่ไม่ใช่ -10 ให้กลายเป็น -10

Q			

Row

1	-10	-10	-10
---	-----	-----	-----

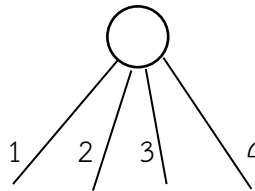
Column

1	-10	-10	-10
---	-----	-----	-----

“ Queen ถูกนำออกไปแล้ว ”

จากการ Diagram เราจะพบว่า ถึงแม้ว่า Data Structure ที่ใช้ในการเก็บข้อมูลของ Queen ใน Board จะเป็น Array ธรรมดา แต่ขั้นตอนในการ Add และ Delete จะมีคุณสมบัติเป็น Last in First out ซึ่งเป็น Property ของ Stack

การหาคำตอบในกรณีที่ User เลือกวาง Queen ตัวแรก



เราจะหาคำตอบจากทุกๆขาของ Node แล้วนำคำตอบที่ได้มาต่อท้ายใน Array List

เมื่อจบการทำงานของขา 1 ขา ก็สั่งให้ Board ของเราทำการ Clear ค่าที่เก็บ Queen โดยการวนจากตัวที่ 2 ถึงตัวสุดท้ายแล้วเปลี่ยนเป็น -10

ตัวอย่าง Input Row = 1 และ Column = 1

Arraylist result	3	1	4	2	0	2	4	1	3	0	Column
	4	3	2	1	0	4	3	2	1	0	Row

“ใช้การ Mod 5 เพื่อให้ได้เซตของคำตอบที่เราต้องการ”

จากนั้นนำ Input เข้าไป Check ทีละตัวกับ Result เพื่อดูว่าเลข Row กับเลข Column ตรงกันหรือไม่ ถ้าไม่ตรงกันทุกตัวก็ทำให้เราสรุปได้ว่า No Solution แต่ถ้า Search เจอเราก็ทำการเลือกเซตของคำตอบซึ่งมีสมาชิก 1 ตัวที่ตรงกับ Input แล้วนำไปแสดงผล

เพิ่มเติม

เหตุผลที่ใช้ Array List ในการเก็บเซตของคำตอบเนื่องจากตอนที่เรากำลังทำการเปรียบเทียบค่าของ Input กับค่าใน Array List จะต้องมีการเข้าถึงสมาชิกของ Array List ผ่าน Index ซึ่งเวลาในการทำงานเป็นเวลาที่เรากำลังต้องการเข้าถึงข้อมูล 1 ตัว เท่ากับ $O(1)$

เหตุผลที่ใช้ Array ในการเก็บข้อมูลของ Queen แทนที่จะใช้ Array Deque เนื่องจาก Java Api ไม่มีบริการที่คอยเข้าถึงสมาชิกทุกตัวใน Array Deque เวลาเราทำการ Search ข้อมูลใน Array ตอนที่เราวาง Queen แบบหลอกๆ เราจำเป็นต้องเข้าถึงสมาชิกทุกตัวใน Array เพื่อคอย Check ว่า Queen ตัวใหม่ที่ถูกรวบรวมแบบหลอกๆ นั้นจะถูกกินหรือไม่ เราจึงเลือกใช้ Array ธรรมดาแล้วสร้าง Method ที่ทำการ Add กับ Remove ข้อมูลที่มี Property แบบเดียวกับ Stack

ข้อจำกัดของโปรแกรม

การใส่ข้อมูลจะถูกกำหนดช่วงเอาไว้ผู้ที่เข้ามาใช้งานจะต้องกรอกข้อมูลตามช่วงที่กำหนดไว้เท่านั้น โปรแกรมถึงจะทำงานต่อไปได้

อ้างอิง

https://www.youtube.com/watch?v=xFv_HL4B83A (สืบค้นเมื่อ 23/02/2565)