รายงาน

คณะวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยมหิดล

รายวิชา

EGCO 221 – Data Structures and Algorithms

เสนอ

รศ.ดร. รังสิพรรณ มฤคทัต (Assoc. Prof. Rangsipan Marukatat, Ph.D.)

สมาชิก

นายชนาวีร์	เสถียรธีราภาพ	6313124
นายชยกร	จุลนีย์	6313125
นายปิยวัฒน์	วิโรจน์กิจไพบูลย์	6313132
นายสรวิชญ์	คูหาเกษมสิน	6313219

คู่มือโปรแกรม (User Manual)

1. เมื่อโปรแกรมเริ่มทำงาน โปรแกรมจะถามค่า N จากผู้ใช้ ซึ่งค่า N คือขนาดของกระดาน N x N ที่ใช้ในการวางราชินี โดยที่ N มีขนาดตั้งแต่ 4 x 4 ขึ้นไป เมื่อใส่ค่า N แล้ว โปรแกรมจะแสดงรูปแบบของตาราง N x N ขึ้นมาทางหน้าจอ

```
---( N-queen problem solutions )---
Enter N (at least 4) = 4
     1  2  3  4
  1  .  .  .  .
  2  .  .  .  .
  3  .  .  .  .
  4  .  .  .  .
```

2. จากนั้นโปรแกรมจะถามผู้ใช้ว่าต้องการระบุตำแหน่งของราชินีตัวแรกบนกระดานด้วยตนเองหรือไม่ กรณีต้องการวางราชินีตัวแรกบนกระดานด้วยตัวเอง ให้ใส่ y หรือ yes จากนั้น โปรแกรมจะแสดงให้กรอกแถวกับคอลัมน์ที่ต้องการวางราชินีตัวแรก จากนั้นโปรแกรมจึงแสดง Solutions ที่มีราชินีวางอยู่ที่ตำแหน่งที่ได้กำหนดไว้ทั้งหมดของกระดาน N x N

```
Place first queen manually? (y/n) y
Enter column of first queen (1-4) = 1
Enter row of first queen (1-4) = 2
Input: (x=1, y=2)
     1  2  3  4
  1  .  .  .  .
  2  Q  .  .  .
  3  .  .  .  .
  4  .  .  .  .

Solution 1:
     1  2  3  4
  1  .  .  Q  .
  2  Q  .  .  .
  3  .  .  .  Q
  4  .  Q  .  .

Continue? (y/n)
```

กรณีที่ผู้ใช้ไม่ต้องการระบุตำแหน่งของราชินีตัวแรกบนกระดานด้วยตัวเอง ให้ใส่ n หรือ no จากนั้น โปรแกรมจะแสดง Solutions ทั้งหมดของกระดาน N x N นั้น

```
Place first queen manually? (y/n) n
Solution 1:
     1  2  3  4
  1  .  Q  .  .
  2  .  .  .  Q
  3  Q  .  .  .
  4  .  .  Q  .

Solution 2:
     1  2  3  4
  1  .  .  Q  .
  2  Q  .  .  .
  3  .  .  .  Q
  4  .  Q  .  .

Continue? (y/n)
```

3. เมื่อเสร็จสิ้นกระบวนการ ผู้ใช้สามารถเลือกวนกลับไปเล่นใหม่ได้ เพื่อระบุขนาดของกระดาน หรือตำแหน่งของการวางราชินีตัวแรกใหม่โดยการระบุ y หรือ yes หรือเลือกที่จะจบการทำงานของโปรแกรมได้โดยการระบุ n หรือ no

โครงสร้างข้อมูลที่นำมาประยุกต์ใช้งาน (Applied Data Structures)

1. Java Array ประเภท Integer จำนวน 1 ตัว ขนาด N แทนกระดานขนาด NxN ซึ่งทำหน้าที่เก็บตำแหน่งของหมากราชินีแต่ละตัว โดยใช้ index ของ array เป็นแถว (y) และค่าใน array เป็นคอลัมน์ (x) โดยหากในแถวใดไม่มีหมากราชินีอยู่จะให้ค่าเป็น -1 เช่น ในกระดานขนาด 4x4 มี array {-1, 3, 0, -1} หมายถึงมีหมากราชินีอยู่ที่ตำแหน่ง (x,y) = (3,1) และ (0,2) ในกระดาน

Array
-1
3
0
-1

	0	1	2	3
0				
1				Q
2	Q			
3				

2. Runtime stack สำหรับการ backtracking ในการเรียก recursive method โดยจะบันทึกตำแหน่งของเมธอดต่างๆ ที่ถูกเรียกใช้

อัลกอริทึมการย้อนรอย (Backtracking algorithm)

เริ่มต้นด้วยการเลือกว่าจะระบุตำแหน่งของราชินีตัวแรกที่จะวางหรือไม่

โดยใช้ `NQueensProblem(int n, int sx, int sy)` เมื่อต้องการที่จะระบุตำแหน่งของราชินีตัวแรก

และ `NQueensProblem(int n)` เมื่อต้องการให้ตำแหน่งแรกของราชินีเป็นตำแหน่งใดก็ได้

พารามิเตอร์ `n` หมายถึง ขนาดของกระดาน n × n

`sx` หมายถึง คอลัมน์ของราชินีตัวแรก

`sy` หมายถึง แถวของราชินีตัวแรก

```
public NQueensProblem(int n, boolean showEmptyOnly) {
    N = n;
    createBoard();
    if (showEmptyOnly) { display(); return; }
    placeQueen(0); //start an attempt to place queens
    if ( foundSolution < 1 ) System.out.println("No solution!");
}
public NQueensProblem(int n, int sx, int sy) {
    N = n; startX = sx; startY = sy;
    createBoard();
    board[startY] = startX; //place first queen
    System.out.printf("Input: (x=%d, y=%d)\n", startX+1, startY+1);
    display();
    placeQueen(1); //start an attempt to place remaining queens after a queen is placed
    if ( foundSolution < 1 ) System.out.println("No solution!");
}
public NQueensProblem(int n) { this(n, false); }
```

จากนั้นจึงเรียกเมธอด `placeQueen(int placed)` ซึ่งเป็นเมธอดที่ใช้ในการประมวลผลเพื่อหาผลลัพธ์ในการวางราชินี

```
private void placeQueen(int placed) {
    if (placed == N) { //N queens are placed successfully
        System.out.printf("Solution %d:\n", foundSolution+1);
        display();
        foundSolution++;
    } else {
        for (int i = 0; i < N; i++) {
            int x = (startX + i)%N;
            int y = (startY + placed)%N;
            if ( isSafe(x,y) ) {
                board[y] = x; //place queen on this row
                placeQueen(placed+1); //move to the next row
                board[y] = -1; // remove placed queen from this row
            }
        }
    }
}
```

ตัวแปร placed คือตัวแปรที่ใช้อ้างอิงถึงจำนวนราชินีที่วางไปแล้วในกระดาน ถ้าหากราชินีตัวแรกถูกทำการวางไปเรียบร้อยแล้ว เริ่มต้นจะให้ตัวแปร placed มีค่าเป็น 1 แต่หากยังไม่ได้ถูกวางจะให้ placed มีค่าเป็น 0

เมธอด `placedQueen(int placed)` เริ่มต้นโดยการตรวจสอบว่า บนกระดานมีจำนวนราชินีครบตามที่กำหนดหรือไม่ หากมีจำนวนครบ ให้ทำการแสดงผลกระดานนั้นออกมาแล้วจึงจบเมธอด แต่หากยังไม่ครบจะทำการวนลูปเพื่อหาตำแหน่งให้กับราชินีตัวถัดไป

```
for (int i = 0; i < N; i++) {
    int x = (startX + i)%N;
    int y = (startY + placed)%N;
    if ( isSafe(x,y) ) {
        board[y] = x; //place queen on this row
        placeQueen(placed+1); //move to the next row
        board[y] = -1; // remove placed queen from this row
    }
}
```

ในการหาตำแหน่งของราชินีจะเริ่มต้นที่แถว startY และคอลัมน์ startX โดยจะขยับตำแหน่งที่จะทดลองวางราชินีจากซ้ายไปขวาที่ละหนึ่งคอลัมน์จนครบ N คอลัมน์ โดยในแต่ละตำแหน่งที่จะทดลองวางราชินีนี้ จะทำการตรวจสอบเงื่อนไขในการวางด้วยเมธอด `boolean isSafe(int x, int y)` ซึ่งคืนค่า `true` เมื่อราชินีที่จะทำการวางเพิ่มในตำแหน่ง (x, y) ไม่กินกันเองกับราชินีตัวอื่นที่อยู่บนกระดานที่มีตำแหน่งในแถวเดียวกัน คอลัมน์เดียวกัน หรือในแนวเส้นทแยงมุมเดียวกัน

```
private boolean isSafe(int x, int y) {
    for (int i = 0; i < N; i++) {
        if (board[i] == -1) continue;
        // placed_x = board[i], placed_y = i
        // position (x,y) is in the same row or column with placed queen
        if (board[i] == x || i == y) return false;
        // position (x,y) is on the same diagonal line with placed queen
        if (Math.abs(board[i] - x) == Math.abs(i - y)) return false;
    }
    return true;
}
```

ต่อมาหากตำแหน่งใดสามารถวางราชินีได้ ให้ทำการวางหมากราชินีที่ตำแหน่งนั้น แล้วจึงทำการเรียกเมธอดนี้ซ้ำซึ่งเป็นการเรียกใช้ recursive call โดยจะใส่อาร์กิวเมนต์เป็น placed+1 เพื่อทำการขยับแถวไปยังแถวถัดไป

```
placeQueen( place+1 );
```

เมื่อเรียกเมธอดใดๆ runtime จะทำการบันทึก (push) ตำแหน่งของคำสั่งที่เรียกเมธอดนี้ลงใน runtime stack เพื่อที่จะสามารถกลับมาทำคำสั่งถัดไปที่อยู่ต่อจากเมธอดนี้ได้ ทำให้ placeQueen เป็น recursive method ที่สามารถขยับแถวขึ้นลงได้โดยการ recursive call

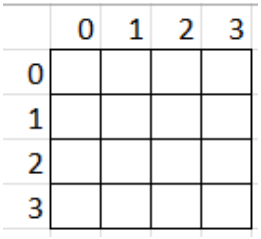
จากนั้นในแต่ละแถวเมื่อขยับครบ N คอลัมน์แล้วจึงจบเมธอด โดยหากแถวปัจจุบันเป็นแถวที่เกิดจากการ recursive call เมื่อจบเมธอด runtime จะพากลับไปยังคำสั่งที่แถวก่อนหน้านี้ (backtracking) โดยการลบ (pop) ตำแหน่งคำสั่งของแถวปัจจุบันออกจาก runtime stack ทำให้สามารถอ่านค่าที่ด้านบนสุดของ

runtime stack ซึ่งเป็นตำแหน่งของคำสั่งที่แถวก่อนหน้านี้ได้ เมื่อกลับมาแล้วอัลกอริทึมจะหยิบตัวราชินีในแถวนั้นออก แล้วจึงขยับไปยังคอลัมน์ถัดไป

ในการขยับตำแหน่ง หากขยับจนสุดขอบกระดานด้านขวาแล้วจะวนกลับมายังขอบกระดานด้านซ้าย ซึ่งสามารถทำได้โดยการหาเศษเหลือจากการนำพิกัด x ในปัจจุบันไปหารด้วยขนาดของกระดาน หรือ (x mod N) ในทางเดียวกันเมื่อขยับตำแหน่งไปยังแถวถัดไป หากขยับจนสุดขอบกระดานด้านล่างแล้ว ก็วนกลับมาที่ขอบกระดานด้านบนเช่นกัน

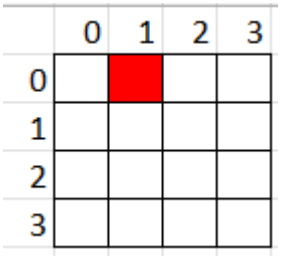
ตัวอย่างการทำงานของการย้อนลอย (Backtracking Demonstration)

เริ่มต้นด้วยการสร้างกระดานขนาด 4 x 4 โดยที่กระดานเริ่มต้นคือกระดานที่ไม่มีราชินีวางอยู่




```
board = { -1, -1, -1, -1 };
```

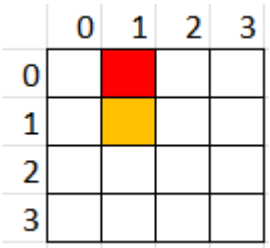
วางราชินีตัวแรกในตำแหน่งที่เราต้องการ `board[row] = col;` เช่น row = 0 และ col = 1 ดังนั้น




```
Board[0] = 1;
```

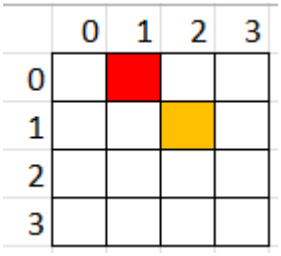
- **กรณีพบ Solutions**

ตำแหน่งถัดไปที่จะวางคือ x = (1 + 0)%4 = 1 , y = (0+1)%4 = 1 , ตอนนี้ i = 0 และ placed = 1



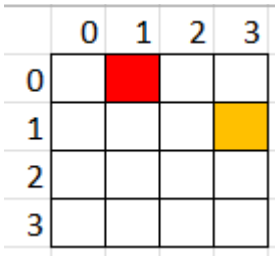
จากรูปจะเห็นว่าที่ตำแหน่ง (1,1) ไม่สามารถวางได้เนื่องจากอยู่ในคอลัมน์เดียวกัน จึงขยับไปยังตำแหน่งถัดไป

ตำแหน่งถัดไปที่จะวางคือ x = (1 + 1)%4 = 2 , y = (0+1)%4 = 1 , ตอนนี้ i = 1 และ placed = 1



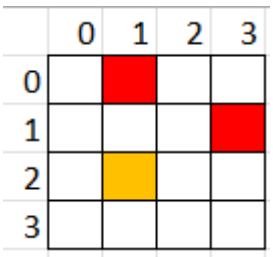
จากรูปจะเห็นว่าที่ตำแหน่ง (2,1) ไม่สามารถวางได้เนื่องจาก อยู่เส้นทแยงมุมของราชินีตัวก่อนหน้า จึงขยับไปยังตำแหน่งถัดไป

ตำแหน่งถัดไปที่จะวางคือ x = (1 + 2)%4 = 3 , y = (0+1)%4 = 1 , ตอนนี้ i = 2 และ placed = 1



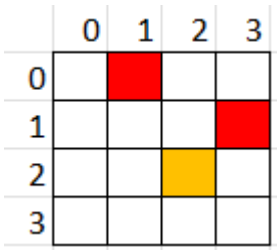
จากรูปจะเห็นว่าที่ตำแหน่ง (3,1) สามารถวางราชินีตัวที่ 2 ได้ จากนั้นจึงใช้ recursive call เพื่อวางราชินีตัวถัดไป placeQueen(placed +1); ตอนนี้ placed = 2

ตำแหน่งถัดไปที่เราจะวาง x = (1 + 0)%4 = 1 , y = (0+2)%4 = 2 , ตอนนี้ i = 0 และ placed = 2



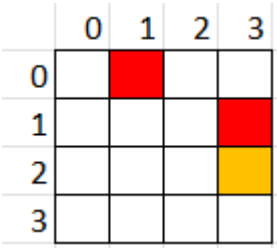
จากรูปจะเห็นว่าที่ตำแหน่ง (1,2) ไม่สามารถวางได้เนื่องจาก อยู่ในคอลัมน์เดียวกันกับราชินีตัวที่ 1 จึงขยับไปยังตำแหน่งถัดไป

ตำแหน่งถัดไปที่เราจะวาง x = (1 + 1)%4 = 2 , y = (0+2)%4 = 2 , ตอนนี้ i = 1 และ placed = 2



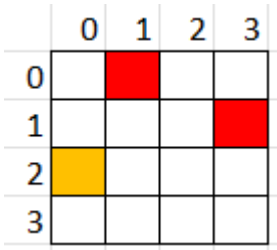
จากรูปจะเห็นว่าที่ตำแหน่ง (2,2) ไม่สามารถวางได้เนื่องจาก อยู่ในแนวทแยงเดียวกันกับราชินีตัวที่ 2 จึงขยับไปยังตำแหน่งถัดไป

ตำแหน่งถัดไปที่เราจะวาง x = (1 + 2)%4 = 3 , y = (0+2)%4 = 2 , ตอนนี้ i = 2 และ placed = 2



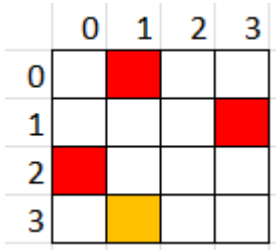
จากรูปจะเห็นว่าที่ตำแหน่ง (3,2) ไม่สามารถวางได้เนื่องจาก อยู่ในแนวทแยงมุมเดียวกับราชินีตัวที่ 1 และคอลัมน์เดียวกับราชินีตัวที่ 2 จึงขยับไปยังตำแหน่งถัดไป

ตำแหน่งถัดไปที่เราจะวาง x = (1 + 3)%4 = 0 , y = (0+2)%4 = 2 , ตอนนี้ i = 3 และ placed = 2



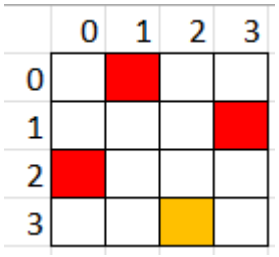
จากรูปจะเห็นว่าที่ตำแหน่ง (0,2) สามารถวางราชินีตัวที่ 3 ได้ จากนั้นจึงใช้ recursive call เพื่อวางราชินีตัวถัดไป placeQueen(placed +1); ตอนนี้ placed = 3

ตำแหน่งถัดไปที่เราจะวาง x = (1 + 0)%4 = 1 , y = (0+3)%4 = 3 , ตอนนี้ i = 0 และ placed = 3



จากรูปจะเห็นว่าที่ตำแหน่ง (1,3) ไม่สามารถวางได้เนื่องจาก อยู่ในคอลัมน์เดียวกันกับราชินีตัวที่ 1 และเส้นทแยงมุมเดียวกันกับราชินีตัวที่ 3 จึงขยับไปยังตำแหน่งถัดไป

ตำแหน่งถัดไปที่เราจะวาง x = (1 + 1)%4 = 2 , y = (0+3)%4 = 3 , ตอนนี้ i = 1 และ placed = 3

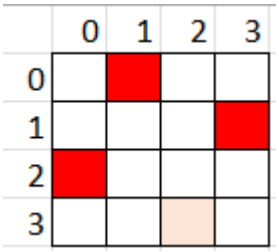


จากรูปจะเห็นว่าที่ตำแหน่ง (2,3) สามารถวางราชินีตัวที่ 4 ได้

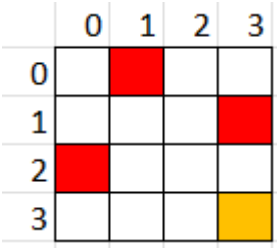
จากนั้นจึงใช้ recursive call เพื่อวางราชินีตัวถัดไป placeQueen(placed +1); ตอนนี้ placed = 4

เมื่อ placed = 4 = N หมายความว่า ตอนนี้เราได้ทำการวางราชินีครบจำนวน N ตัวแล้ว ดังนั้น จึงทำการแสดง Solution นั้นออกมา ได้ Solution#1

จากนั้นเมื่อ placeQueen(4) ทำหน้าที่เสร็จเรียบร้อยแล้ว จึง backtrack กลับมาที่ placeQueen(3) และทำการลบราชินีตัวที่ 4 ออกจากกระดาน

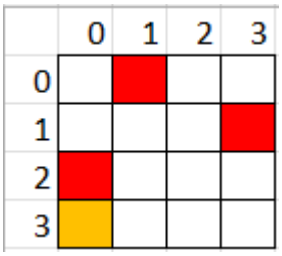


ตำแหน่งถัดไปที่เราจะวาง x = (1 + 2)%4 = 3 , y = (0+3)%4 = 3 , ตอนนี้ i = 2 และ placed = 3



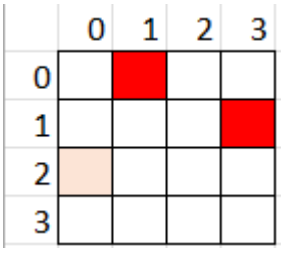
จากรูปจะเห็นว่าที่ตำแหน่ง (3,3) ไม่สามารถวางได้เนื่องจาก อยู่ในคอลัมน์เดียวกันกับราชินีตัวที่ 2 จึงขยับไปยังตำแหน่งถัดไป

ตำแหน่งถัดไปที่เราจะวาง x = (1 + 3)%4 = 0 , y = (0+3)%4 = 3 , ตอนนี้ i = 3 และ placed = 3

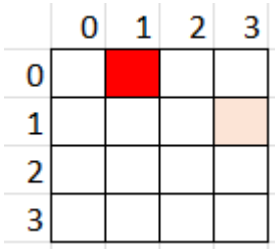


จากรูปจะเห็นว่าที่ตำแหน่ง (0,3) ไม่สามารถวางได้เนื่องจาก อยู่ในคอลัมน์เดียวกันกับราชินีตัวที่ 3

จากนั้นเมื่อ placeQueen(3) ทำหน้าที่เสร็จเรียบร้อยแล้ว จึง backtrack กลับมาที่ placeQueen(2) และทำการลบราชินีตัวที่ 3 ออกจากกระดาน

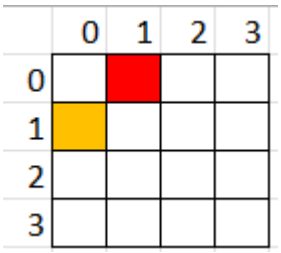


จากรูปจะเห็นว่า ที่ตำแหน่ง (0,2) placeQueen(2) ทำการประมวลผลลูปสุดท้ายเสร็จแล้ว (i = 3) เมื่อ placeQueen(2) ทำหน้าที่เสร็จเรียบร้อยแล้ว จึง backtrack กลับมาที่ placeQueen(1) และทำการลบราชินีตัวที่ 2 ออกจากกระดาน



ตอนนี้ placeQueen(1) ทำการประมวลผลลูป (i = 2)

ตำแหน่งถัดไปที่เราจะวาง x = (1 + 3)%4 = 0 , y = (0+1)%4 = 1 , ตอนนี้ i = 3 และ placed = 1



จากรูปจะเห็นว่าที่ตำแหน่ง (0,1) ไม่สามารถวางได้เนื่องจาก อยู่ในแนวทแยงมุมเดียวกันกับราชินีตัวที่ 1

เมื่อ placeQueen(1) ทำหน้าที่เสร็จเรียบร้อยแล้ว จึงจบอัลกอริทึม

- **กรณีไม่พบ Solutions**

อัลกอริทึมการทำงานจะเหมือนกับกรณีพบ solutions ที่เริ่มประมวลผลจาก (คอลัมน์, แถว) ที่ผู้ใช้วางราชินีตัวแรก แล้วทำการวนซ้ำเพื่อตรวจสอบคอลัมน์ทีละคอลัมน์ในแถวเดียวกัน จนกว่าจะพบตำแหน่งที่สามารถวางราชินีได้

	0	1	2	3
0	Q			
1				
2				
3				

เมื่อวางราชินีในแถวนั้นได้แล้วจึงใช้ Recursive call เพื่อตรวจสอบแถวถัดไป

	0	1	2	3
0	Q			
1				
2				
3				

เมื่อตรวจสอบแล้วพบแถวที่ไม่สามารถวางราชินีในคอลัมน์ใดๆ ได้เลย

	0	1	2	3
0	Q			
1			Q	
2				
3				

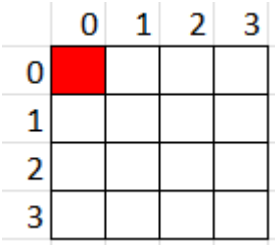
จึงทำการ Backtrack กลับมา แล้วลบราชินีที่วางล่าสุดเพื่อตรวจสอบคอลัมน์ถัดไป

	0	1	2	3
0	Q			
1			Q	
2				
3				

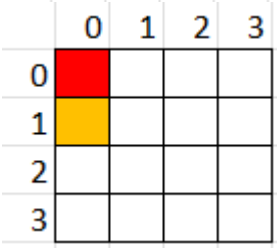
ดังนั้นเมื่อตรวจสอบทุกความเป็นไปได้แล้ว ไม่พบ Solutions เลย จึงแสดง No Solution! ออกทางหน้าจอ

ตัวอย่าง วางราชินีตัวแรก (0, 0)

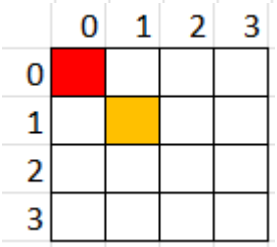
1. Start



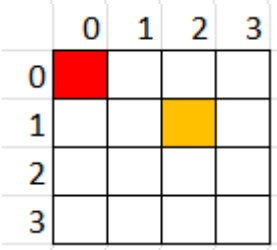
2. Recursive call



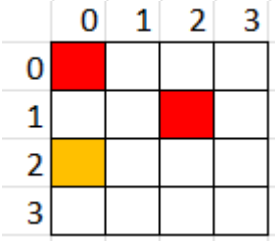
3.Find next



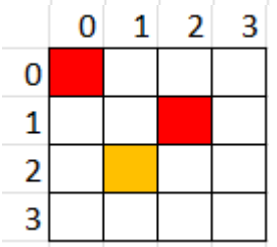
4. Find next



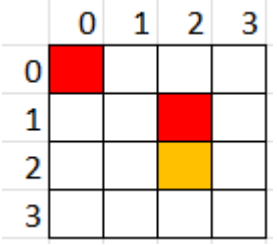
5. Recursive call



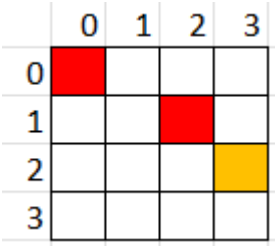
6. Find next



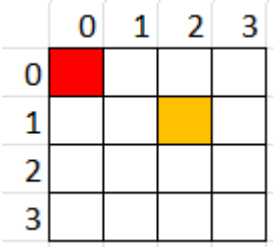
7. Find next



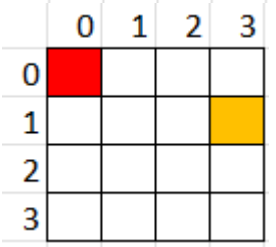
8. Find next



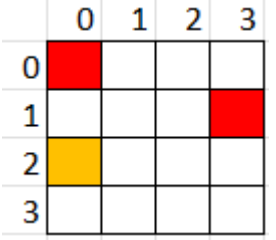
9. Backtrack



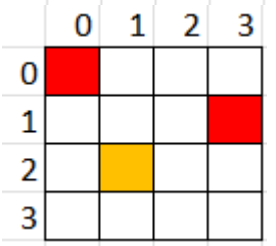
10. Find next



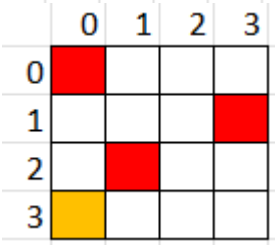
11.Recursive Call



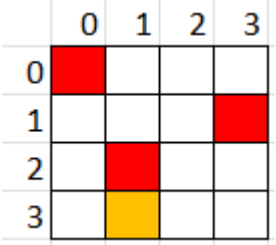
12. Find next



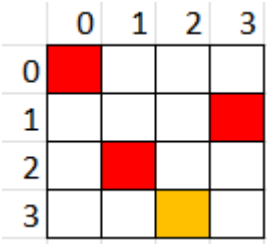
13. Recursive Call



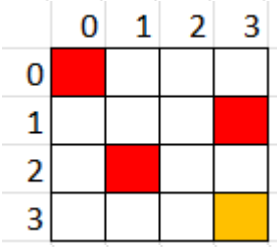
14. Find next



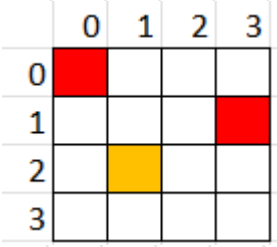
15. Find next



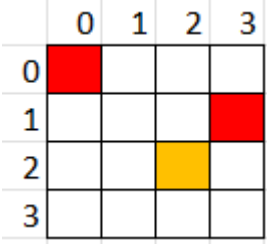
16. Find next



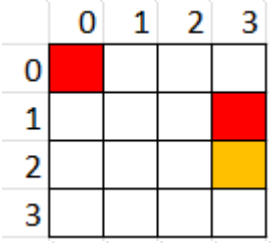
17. Backtrack



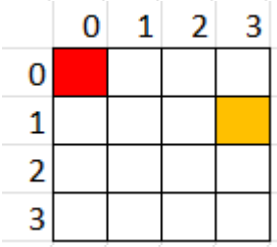
18. Find next



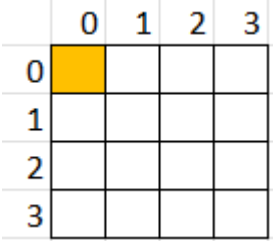
19. Find next



20. Backtrack



21. Backtrack



ดังนั้น No Solution!

ข้อจำกัดของโปรแกรม (Program Limitations)

โปรแกรมสามารถคำนวณเพื่อหาคำตอบของกระดานขนาดใดก็ได้ แต่ใช้เวลาในการคำนวณเพิ่มขึ้นตามขนาดกระดาน และโปรแกรมสามารถทำงานได้แค่บนระบบปฏิบัติการที่มีการติดตั้ง Java ไว้เท่านั้น

ความเป็นต้นฉบับและอ้างอิง (Originality and Reference)

คณะผู้จัดทำได้ทำการศึกษาอัลกอริทึมจากทางอินเทอร์เน็ต โดยใช้ข้อมูลและอัลกอริทึมจากเว็บเพจด้านใต้มาเรียบเรียงใหม่ และดัดแปลงอัลกอริทึมให้สามารถระบุจุดเริ่มต้นในการวางราชินีได้โดยการใช้ modulo ร่วมในการขยับแถวและคอลัมน์

https://sadakurapati.wordpress.com/2013/12/10/n-queens-backtracking-algorithm/