BÁO CÁO THỰC HÀNH NHẬP MÔN TRÍ TUỆ NHÂN TẠO

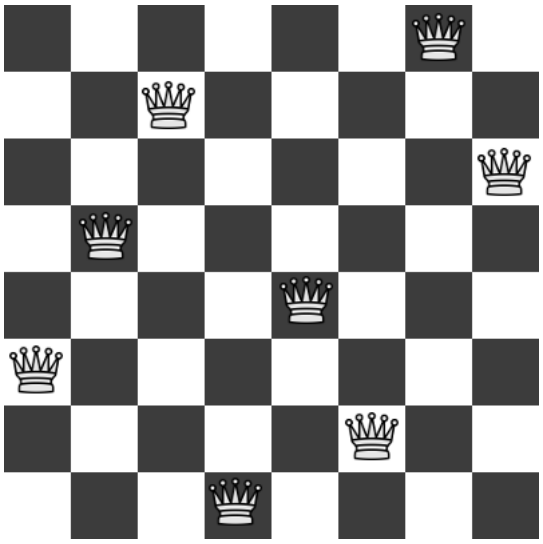
Tuần 6

Bài toán. Viết chương trình cài đặt thuật toán xếp tám quân hậu trên bàn cờ kích thước 8×8 sao cho không có quân hậu nào có thể ăn những quân hậu còn lại, nghĩa là không có hai quân nào đứng trên cùng hàng, hoặc cùng cột hoặc cùng đường chéo.

1 Ý tưởng thực hiện bài toán

Ý tưởng chung của bài toán này là ta sẽ quét từng hàng của bàn cờ và đặt một quân hậu trên mỗi cột, đồng thời kiểm tra ở mỗi bước, xem không có hai quân hậu nào nằm trong đường tấn công của quân kia.

Để tối ưu bài toán, ta có thể biểu diễn bàn cờ dưới dạng một mảng gồm tám phần tử, trong đó mỗi phần tử đại diện cho vị trí cột của quân hậu từ một hàng cụ thể ứng với chỉ số của mảng đó. Lấy một ví dụ cho cách biểu diễn trên như sau



Các vị trí quân hậu trên bàn cờ ở trên, có thể được biểu diễn dưới dạng các vị trí đã chiếm của mảng 8x8 hai chiều: [0, 6], [1, 2], [2, 7], [3, 1], [4, 4], [5, 0], [6, 5], [7, 3]. Tuy nhiên, như đã mô tả ở trên, chúng ta có thể sử dụng mảng 8 phần tử một chiều: [6, 2, 7, 1, 4, 0, 5, 3]. Bằng cách tạo ra tất cả các hoán vị có thể có của một mảng tám số, [0, 1, 2, 3, 4, 5, 6, 7] và loại bỏ các trạng thái không hợp lệ (các trạng thái mà trong đó hai quân hậu bất kỳ có thể tấn công nhau) ta sẽ có được giải pháp cần tìm.

2 Xây dựng class NQueens

```
class NQueens:
    """Generate all valid solutions for the n queens puzzle"""
    def __init__(self, size):
        # Store the puzzle (problem) size and the number of valid solutions
        self.size = size
        self.solutions = 0
        self.solve()

    def solve(self):
        """Solve the n queens puzzle and print the number of solutions"""
        positions = [-1] * self.size
        self.put_queen(positions, 0)
        print("Found", self.solutions, "solutions.")

    def put_queen(self, positions, target_row):
        """
        Try to place a queen on target_row by checking all N possible cases.
        If a valid place is found the function calls itself trying
        to place a queen on the next row until all N queens are
        placed on the NxN board.
        """
        # Base (stop) case - all N rows are occupied
        if target_row == self.size:
            self.show_full_board(positions)
            self.solutions += 1
        else:
            # For all N columns positions try to place a queen
            for column in range(self.size):
                # Reject all invalid positions
                if self.check_place(positions, target_row, column):
                    positions[target_row] = column
                    self.put_queen(positions, target_row + 1)
```

```
    def check_place(self, positions, ocuppied_rows, column):
        """
        Check if a given position is under attack from any of
        the previously placed queens (check column and diagonal positions)
        """
        for i in range(ocuppied_rows):
            if positions[i] == column or \
                positions[i] - i == column - ocuppied_rows or \
                positions[i] + i == column + ocuppied_rows:

                return False
        return True

    def show_full_board(self, positions):
        """Show the full NxN board"""
        for row in range(self.size + 1):
            line = ""
            if row == 0:
                line += "  "
                for c in ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h']:
                    line = line + c + " "
            else:
                line = str(row) + " "
                for column in range(self.size):
                    if positions[row - 1] == column:
                        line += "Q "
                    else:
                        line += ". "
            print(line)
        print("\n")
```

Trong đó

- Hàm **__init__** là hàm khởi tạo các giá trị cho class NQueens với **size** là số quân hậu, đồng thời là kích thước một cạnh bàn cờ, **solutions** cho biết số giải pháp của bài

toán, **solve()** là hàm thực thi bài toán.

- Phương thức **solve** : giải bài toán n-quân hậu và in ra kết quả và số đáp án.
- Phương thức **put_queen** : Đặt một quân hậu trên target_row bằng cách kiểm tra tất cả N trường hợp có thể xảy ra. Nếu tìm thấy một vị trí hợp lệ, hàm tự gọi lại nó và đặt một quân hậu ở hàng tiếp theo cho đến khi tất cả N quân hậu được đặt trên bảng NxN.
- Phương thức **check_place** : Kiểm tra xem một vị trí nhất định có bị tấn công từ bất kỳ quân hậu đã đặt trước đó hay không (kiểm tra vị trí cột và đường chéo)
- Phương thức **show_full_board** : hiển thị bàn cờ với vị trí của các quân hậu sau khi được đặt.

3 Kết quả thực thi

Hiển thị một vài kết quả sau khi thực thi và hiển thị số trường hợp có để thể đặt 8 quân hậu thoả mãn điều kiện đề bài.

```
  a b c d e f g h
1 . . . . . . . Q
2 . Q . . . . . .
3 . . . . Q . . .
4 . . Q . . . . .
5 Q . . . . . . .
6 . . . . . . Q .
7 . . . Q . . . .
8 . . . . . Q . .


  a b c d e f g h
1 . . . . . . . Q
2 . . Q . . . . .
3 Q . . . . . . .
4 . . . . . Q . .
5 . Q . . . . . .
```

```
6 . . . . Q . . .
7 . . . . . . Q .
8 . . . Q . . . .



  a b c d e f g h
1 . . . . . . . Q
2 . . . Q . . . .
3 Q . . . . . . .
4 . . Q . . . . .
5 . . . . . Q . .
6 . Q . . . . . .
7 . . . . . . Q .
8 . . . . Q . . .


Found 92 solutions.
```