N-Queens Problem

LinkedIn: Md. Ziaul Karim

Find me here:

51. N-Queens

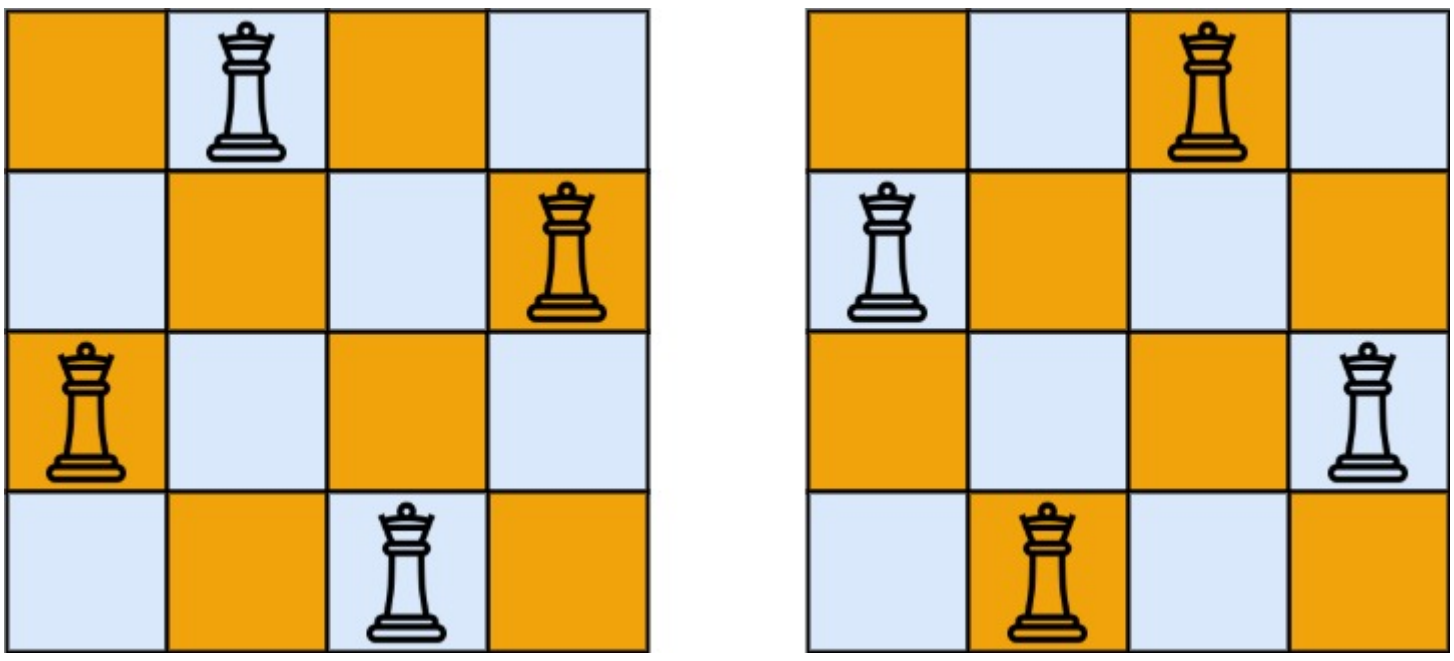
Hard

The **n-queens** puzzle is the problem of placing `n` queens on an `n x n` chessboard such that no two queens attack each other.

Given an integer `n`, return *all distinct solutions to the **n-queens puzzle***. You may return the answer in **any order**.

Each solution contains a distinct board configuration of the n-queens' placement, where `'Q'` and `'.'` both indicate a queen and an empty space, respectively.

Example 1:



Input: n = 4
Output: `[[".Q..","...Q","Q...","..Q."],["..Q.","Q...","...Q",".Q.."]]`
Explanation: There exist two distinct solutions to the 4-queens puzzle as shown above

Example 2:

Input: n = 1
Output: `[["Q"]]`

Code

```
import pprint
col = set()
posDiag = set()
negDiag = set()
n = int(input("Give me n: "))
res = []
board = [["."] * n for i in range(n)]
def backtrack(r):
        if  r==n:
                res.append(["".join(row) for row in board])
                return
        for c in range(n):
                if c in col or (r+c) in posDiag or (r-c) in negDiag:
                        continue
                col.add(c)
```

```
                posDiag.add(r + c)
                negDiag.add(r - c)
                board[r][c] = "Q"
                backtrack(r+1)

                col.remove(c)
                posDiag.remove(r + c)
                negDiag.remove(r - c)
                board[r][c] = "."
        return res
pprint.pprint(backtrack(0))
```

Explnation

```
backtrack(0):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in set() or 0 in set() or 0 in set(): --> False
                set().add(0) >> {0}
                posDiag.add(0 + 0) >> {0}
                negDiag.add(0 - 0) >> {0}
                board[0][0] = 'Q'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(0+1) >> backtrack(1):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {0} or 1 in {0} or 1 in {0}: --> True
                        continue
                #iteration 2
                if 1 in {0} or 2 in {0} or 0 in {0}: --> True
                        continue
                #iteration 3
                if 2 in {0} or 3 in {0} or -1 in {0}: --> False
                {0}.add(2) >> {0, 2}
                posDiag.add(1 + 2) >> {0, 3}
                negDiag.add(1 - 2) >> {0, -1}
                board[1][2] = 'Q'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(1+1) >> backtrack(2):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {0, 2} or 2 in {0, 3} or 2 in {0, -1}: --> True
                        continue
                #iteration 2
                if 1 in {0, 2} or 3 in {0, 3} or 1 in {0, -1}: --> True
                        continue
                #iteration 3
```

```
                if 2 in {0, 2} or 4 in {0, 3} or 0 in {0, -1}: --> True
                        continue
                #iteration 4
                if 3 in {0, 2} or 5 in {0, 3} or -1 in {0, -1}: --> True
                        continue
                #iteration 5
                if 4 in {0, 2} or 6 in {0, 3} or -2 in {0, -1}: --> False
                {0, 2}.add(4) >> {0, 2, 4}
                posDiag.add(2 + 4) >> {0, 3, 6}
                negDiag.add(2 - 4) >> {0, -1, -2}
                board[2][4] = 'Q'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(2+1) >> backtrack(3):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {0, 2, 4} or 3 in {0, 3, 6} or 3 in {0, -1, -2}: --> True
                        continue
                #iteration 2
                if 1 in {0, 2, 4} or 4 in {0, 3, 6} or 2 in {0, -1, -2}: --> False
                {0, 2, 4}.add(1) >> {0, 1, 2, 4}
                posDiag.add(3 + 1) >> {0, 3, 4, 6}
                negDiag.add(3 - 1) >> {0, 2, -1, -2}
                board[3][1] = 'Q'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(3+1) >> backtrack(4):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {0, 1, 2, 4} or 4 in {0, 3, 4, 6} or 4 in {0, 2, -1, -2}: --> True
                        continue
                #iteration 2
                if 1 in {0, 1, 2, 4} or 5 in {0, 3, 4, 6} or 3 in {0, 2, -1, -2}: --> True
                        continue
                #iteration 3
                if 2 in {0, 1, 2, 4} or 6 in {0, 3, 4, 6} or 2 in {0, 2, -1, -2}: --> True
                        continue
                #iteration 4
                if 3 in {0, 1, 2, 4} or 7 in {0, 3, 4, 6} or 1 in {0, 2, -1, -2}: --> False
                {0, 1, 2, 4}.add(3) >> {0, 1, 2, 3, 4}
                posDiag.add(4 + 3) >> {0, 3, 4, 6, 7}
                negDiag.add(4 - 3) >> {0, 1, 2, -2, -1}
                board[4][3] = 'Q'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(4+1) >> backtrack(5):
        if  r==n: --> True
                🟡 #Base Case 🟡
                res.append(''.join(row) for row in board])
>>
```

```
[['Q....', '..Q..', '....Q', '.Q...', '...Q.']]
                return
                #Maximum recursive depth reached
                🔴 #End of recursive block🔴

                {0, 1, 2, 3, 4}.remove(3) >> {0, 1, 2, 4}
                posDiag.remove(4 + 3) >> {0, 3, 4, 6}
                negDiag.remove(4 - 3) >> {0, 2, -2, -1}
                board[4][3] = '.'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 5
                if 4 in {0, 1, 2, 4} or 8 in {0, 3, 4, 6} or 0 in {0, 2, -2, -1}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {0, 1, 2, 4}.remove(1) >> {0, 2, 4}
                posDiag.remove(3 + 1) >> {0, 3, 6}
                negDiag.remove(3 - 1) >> {0, -2, -1}
                board[3][1] = '.'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 3
                if 2 in {0, 2, 4} or 5 in {0, 3, 6} or 1 in {0, -2, -1}: --> True
                        continue
                #iteration 4
                if 3 in {0, 2, 4} or 6 in {0, 3, 6} or 0 in {0, -2, -1}: --> True
                        continue
                #iteration 5
                if 4 in {0, 2, 4} or 7 in {0, 3, 6} or -1 in {0, -2, -1}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {0, 2, 4}.remove(4) >> {0, 2}
                posDiag.remove(2 + 4) >> {0, 3}
                negDiag.remove(2 - 4) >> {0, -1}
                board[2][4] = '.'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {0, 2}.remove(2) >> {0}
                posDiag.remove(1 + 2) >> {0}
```

```
                negDiag.remove(1 - 2) >> {0}
                board[1][2] = '.'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 4
                if 3 in {0} or 4 in {0} or -2 in {0}: --> False
                {0}.add(3) >> {0, 3}
                posDiag.add(1 + 3) >> {0, 4}
                negDiag.add(1 - 3) >> {0, -2}
                board[1][3] = 'Q'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(1+1) >> backtrack(2):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {0, 3} or 2 in {0, 4} or 2 in {0, -2}: --> True
                        continue
                #iteration 2
                if 1 in {0, 3} or 3 in {0, 4} or 1 in {0, -2}: --> False
                {0, 3}.add(1) >> {0, 1, 3}
                posDiag.add(2 + 1) >> {0, 3, 4}
                negDiag.add(2 - 1) >> {0, 1, -2}
                board[2][1] = 'Q'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(2+1) >> backtrack(3):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {0, 1, 3} or 3 in {0, 3, 4} or 3 in {0, 1, -2}: --> True
                        continue
                #iteration 2
                if 1 in {0, 1, 3} or 4 in {0, 3, 4} or 2 in {0, 1, -2}: --> True
                        continue
                #iteration 3
                if 2 in {0, 1, 3} or 5 in {0, 3, 4} or 1 in {0, 1, -2}: --> True
                        continue
                #iteration 4
                if 3 in {0, 1, 3} or 6 in {0, 3, 4} or 0 in {0, 1, -2}: --> True
                        continue
                #iteration 5
                if 4 in {0, 1, 3} or 7 in {0, 3, 4} or -1 in {0, 1, -2}: --> False
                {0, 1, 3}.add(4) >> {0, 1, 3, 4}
                posDiag.add(3 + 4) >> {0, 3, 4, 7}
                negDiag.add(3 - 4) >> {0, 1, -2, -1}
                board[3][4] = 'Q'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.']]
```

```
                🟢 #Starting Recursive block 🟢
backtrack(3+1) >> backtrack(4):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {0, 1, 3, 4} or 4 in {0, 3, 4, 7} or 4 in {0, 1, -2, -1}: --> True
                        continue
                #iteration 2
                if 1 in {0, 1, 3, 4} or 5 in {0, 3, 4, 7} or 3 in {0, 1, -2, -1}: --> True
                        continue
                #iteration 3
                if 2 in {0, 1, 3, 4} or 6 in {0, 3, 4, 7} or 2 in {0, 1, -2, -1}: --> False
                {0, 1, 3, 4}.add(2) >> {0, 2, 1, 3, 4}
                posDiag.add(4 + 2) >> {0, 3, 4, 6, 7}
                negDiag.add(4 - 2) >> {0, 1, 2, -2, -1}
                board[4][2] = 'Q'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', 'Q', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(4+1) >> backtrack(5):
        if  r==n: --> True
                🟡 #Base Case 🟡
                res.append([''.join(row) for row in board])
>>
[['Q....', '..Q..', '....Q', '.Q...', '...Q.'],
 ['Q....', '...Q.', '.Q...', '....Q', '..Q..']]
                return
                #Maximum recursive depth reached
                🔴 #End of recursive block🔴

                {0, 2, 1, 3, 4}.remove(2) >> {0, 1, 3, 4}
                posDiag.remove(4 + 2) >> {0, 3, 4, 7}
                negDiag.remove(4 - 2) >> {0, 1, -2, -1}
                board[4][2] = '.'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.']]
                #iteration 4
                if 3 in {0, 1, 3, 4} or 7 in {0, 3, 4, 7} or 1 in {0, 1, -2, -1}: --> True
                        continue
                #iteration 5
                if 4 in {0, 1, 3, 4} or 8 in {0, 3, 4, 7} or 0 in {0, 1, -2, -1}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {0, 1, 3, 4}.remove(4) >> {0, 1, 3}
                posDiag.remove(3 + 4) >> {0, 3, 4}
                negDiag.remove(3 - 4) >> {0, 1, -2}
                board[3][4] = '.'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..']]
```

```
        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {0, 1, 3}.remove(1) >> {0, 3}
                posDiag.remove(2 + 1) >> {0, 4}
                negDiag.remove(2 - 1) >> {0, -2}
                board[2][1] = '.'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 3
                if 2 in {0, 3} or 4 in {0, 4} or 0 in {0, -2}: --> True
                        continue
                #iteration 4
                if 3 in {0, 3} or 5 in {0, 4} or -1 in {0, -2}: --> True
                        continue
                #iteration 5
                if 4 in {0, 3} or 6 in {0, 4} or -2 in {0, -2}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {0, 3}.remove(3) >> {0}
                posDiag.remove(1 + 3) >> {0}
                negDiag.remove(1 - 3) >> {0}
                board[1][3] = '.'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 5
                if 4 in {0} or 5 in {0} or -3 in {0}: --> False
                {0}.add(4) >> {0, 4}
                posDiag.add(1 + 4) >> {0, 5}
                negDiag.add(1 - 4) >> {0, -3}
                board[1][4] = 'Q'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block ⏪
backtrack(1+1) >> backtrack(2):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {0, 4} or 2 in {0, 5} or 2 in {0, -3}: --> True
                        continue
                #iteration 2
                if 1 in {0, 4} or 3 in {0, 5} or 1 in {0, -3}: --> False
                {0, 4}.add(1) >> {0, 1, 4}
                posDiag.add(2 + 1) >> {0, 5, 3}
                negDiag.add(2 - 1) >> {0, 1, -3}
                board[2][1] = 'Q'
>>
[['Q', '.', '.', '.', '.'],
```

```
 ['.', '.', '.', '.', 'Q'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(2+1) >> backtrack(3):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {0, 1, 4} or 3 in {0, 5, 3} or 3 in {0, 1, -3}: --> True
                        continue
                #iteration 2
                if 1 in {0, 1, 4} or 4 in {0, 5, 3} or 2 in {0, 1, -3}: --> True
                        continue
                #iteration 3
                if 2 in {0, 1, 4} or 5 in {0, 5, 3} or 1 in {0, 1, -3}: --> True
                        continue
                #iteration 4
                if 3 in {0, 1, 4} or 6 in {0, 5, 3} or 0 in {0, 1, -3}: --> True
                        continue
                #iteration 5
                if 4 in {0, 1, 4} or 7 in {0, 5, 3} or -1 in {0, 1, -3}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {0, 1, 4}.remove(1) >> {0, 4}
                posDiag.remove(2 + 1) >> {0, 5}
                negDiag.remove(2 - 1) >> {0, -3}
                board[2][1] = '.'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 3
                if 2 in {0, 4} or 4 in {0, 5} or 0 in {0, -3}: --> True
                        continue
                #iteration 4
                if 3 in {0, 4} or 5 in {0, 5} or -1 in {0, -3}: --> True
                        continue
                #iteration 5
                if 4 in {0, 4} or 6 in {0, 5} or -2 in {0, -3}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {0, 4}.remove(4) >> {0}
                posDiag.remove(1 + 4) >> {0}
                negDiag.remove(1 - 4) >> {0}
                board[1][4] = '.'
>>
[['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁
```

```
                🔴 #End of recursive block🔴

                {0}.remove(0) >> set()
                posDiag.remove(0 + 0) >> set()
                negDiag.remove(0 - 0) >> set()
                board[0][0] = '.'
>>
[['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 2
                if 1 in set() or 1 in set() or -1 in set(): --> False
                set().add(1) >> {1}
                posDiag.add(0 + 1) >> {1}
                negDiag.add(0 - 1) >> {-1}
                board[0][1] = 'Q'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(0+1) >> backtrack(1):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {1} or 1 in {1} or 1 in {-1}: --> True
                        continue
                #iteration 2
                if 1 in {1} or 2 in {1} or 0 in {-1}: --> True
                        continue
                #iteration 3
                if 2 in {1} or 3 in {1} or -1 in {-1}: --> True
                        continue
                #iteration 4
                if 3 in {1} or 4 in {1} or -2 in {-1}: --> False
                {1}.add(3) >> {3, 1}
                posDiag.add(1 + 3) >> {1, 4}
                negDiag.add(1 - 3) >> {-1, -2}
                board[1][3] = 'Q'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(1+1) >> backtrack(2):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {3, 1} or 2 in {1, 4} or 2 in {-1, -2}: --> False
                {3, 1}.add(0) >> {0, 3, 1}
                posDiag.add(2 + 0) >> {1, 2, 4}
                negDiag.add(2 - 0) >> {2, -1, -2}
                board[2][0] = 'Q'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
```

```
backtrack(2+1) >> backtrack(3):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {0, 3, 1} or 3 in {1, 2, 4} or 3 in {2, -1, -2}: --> True
                        continue
                #iteration 2
                if 1 in {0, 3, 1} or 4 in {1, 2, 4} or 2 in {2, -1, -2}: --> True
                        continue
                #iteration 3
                if 2 in {0, 3, 1} or 5 in {1, 2, 4} or 1 in {2, -1, -2}: --> False
                {0, 3, 1}.add(2) >> {2, 0, 3, 1}
                posDiag.add(3 + 2) >> {1, 2, 5, 4}
                negDiag.add(3 - 2) >> {1, 2, -1, -2}
                board[3][2] = 'Q'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(3+1) >> backtrack(4):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {2, 0, 3, 1} or 4 in {1, 2, 5, 4} or 4 in {1, 2, -1, -2}: --> True
                        continue
                #iteration 2
                if 1 in {2, 0, 3, 1} or 5 in {1, 2, 5, 4} or 3 in {1, 2, -1, -2}: --> True
                        continue
                #iteration 3
                if 2 in {2, 0, 3, 1} or 6 in {1, 2, 5, 4} or 2 in {1, 2, -1, -2}: --> True
                        continue
                #iteration 4
                if 3 in {2, 0, 3, 1} or 7 in {1, 2, 5, 4} or 1 in {1, 2, -1, -2}: --> True
                        continue
                #iteration 5
                if 4 in {2, 0, 3, 1} or 8 in {1, 2, 5, 4} or 0 in {1, 2, -1, -2}: --> False
                {2, 0, 3, 1}.add(4) >> {2, 0, 3, 1, 4}
                posDiag.add(4 + 4) >> {1, 2, 5, 4, 8}
                negDiag.add(4 - 4) >> {0, 1, 2, -1, -2}
                board[4][4] = 'Q'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', 'Q']]
                🟢 #Starting Recursive block 🟢
backtrack(4+1) >> backtrack(5):
        if  r==n: --> True
                🟡 #Base Case 🟡
                res.append([''.join(row) for row in board])
>>
[['Q....', '..Q..', '....Q', '.Q...', '...Q.'],
 ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
 ['.Q...', '...Q.', 'Q....', '..Q..', '....Q']]
                return
                #Maximum recursive depth reached
                🔴 #End of recursive block🔴

                {2, 0, 3, 1, 4}.remove(4) >> {2, 0, 3, 1}
                posDiag.remove(4 + 4) >> {1, 2, 5, 4}
                negDiag.remove(4 - 4) >> {1, 2, -1, -2}
                board[4][4] = '.'
>>
```

```
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', '.']]
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {2, 0, 3, 1}.remove(2) >> {0, 3, 1}
                posDiag.remove(3 + 2) >> {1, 2, 4}
                negDiag.remove(3 - 2) >> {2, -1, -2}
                board[3][2] = '.'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 4
                if 3 in {0, 3, 1} or 6 in {1, 2, 4} or 0 in {2, -1, -2}: --> True
                        continue
                #iteration 5
                if 4 in {0, 3, 1} or 7 in {1, 2, 4} or -1 in {2, -1, -2}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {0, 3, 1}.remove(0) >> {3, 1}
                posDiag.remove(2 + 0) >> {1, 4}
                negDiag.remove(2 - 0) >> {-1, -2}
                board[2][0] = '.'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 2
                if 1 in {3, 1} or 3 in {1, 4} or 1 in {-1, -2}: --> True
                        continue
                #iteration 3
                if 2 in {3, 1} or 4 in {1, 4} or 0 in {-1, -2}: --> True
                        continue
                #iteration 4
                if 3 in {3, 1} or 5 in {1, 4} or -1 in {-1, -2}: --> True
                        continue
                #iteration 5
                if 4 in {3, 1} or 6 in {1, 4} or -2 in {-1, -2}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {3, 1}.remove(3) >> {1}
                posDiag.remove(1 + 3) >> {1}
                negDiag.remove(1 - 3) >> {-1}
```

```
                board[1][3] = '.'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 5
                if 4 in {1} or 5 in {1} or -3 in {-1}: --> False
                {1}.add(4) >> {1, 4}
                posDiag.add(1 + 4) >> {1, 5}
                negDiag.add(1 - 4) >> {-1, -3}
                board[1][4] = 'Q'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(1+1) >> backtrack(2):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {1, 4} or 2 in {1, 5} or 2 in {-1, -3}: --> False
                {1, 4}.add(0) >> {0, 1, 4}
                posDiag.add(2 + 0) >> {1, 2, 5}
                negDiag.add(2 - 0) >> {2, -1, -3}
                board[2][0] = 'Q'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(2+1) >> backtrack(3):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {0, 1, 4} or 3 in {1, 2, 5} or 3 in {2, -1, -3}: --> True
                        continue
                #iteration 2
                if 1 in {0, 1, 4} or 4 in {1, 2, 5} or 2 in {2, -1, -3}: --> True
                        continue
                #iteration 3
                if 2 in {0, 1, 4} or 5 in {1, 2, 5} or 1 in {2, -1, -3}: --> True
                        continue
                #iteration 4
                if 3 in {0, 1, 4} or 6 in {1, 2, 5} or 0 in {2, -1, -3}: --> False
                {0, 1, 4}.add(3) >> {3, 0, 1, 4}
                posDiag.add(3 + 3) >> {1, 6, 2, 5}
                negDiag.add(3 - 3) >> {0, 2, -1, -3}
                board[3][3] = 'Q'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(3+1) >> backtrack(4):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {3, 0, 1, 4} or 4 in {1, 6, 2, 5} or 4 in {0, 2, -1, -3}: --> True
                        continue
```

```
                #iteration 2
                if 1 in {3, 0, 1, 4} or 5 in {1, 6, 2, 5} or 3 in {0, 2, -1, -3}: --> True
                        continue
                #iteration 3
                if 2 in {3, 0, 1, 4} or 6 in {1, 6, 2, 5} or 2 in {0, 2, -1, -3}: --> True
                        continue
                #iteration 4
                if 3 in {3, 0, 1, 4} or 7 in {1, 6, 2, 5} or 1 in {0, 2, -1, -3}: --> True
                        continue
                #iteration 5
                if 4 in {3, 0, 1, 4} or 8 in {1, 6, 2, 5} or 0 in {0, 2, -1, -3}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {3, 0, 1, 4}.remove(3) >> {0, 1, 4}
                posDiag.remove(3 + 3) >> {1, 2, 5}
                negDiag.remove(3 - 3) >> {2, -1, -3}
                board[3][3] = '.'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 5
                if 4 in {0, 1, 4} or 7 in {1, 2, 5} or -1 in {2, -1, -3}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {0, 1, 4}.remove(0) >> {1, 4}
                posDiag.remove(2 + 0) >> {1, 5}
                negDiag.remove(2 - 0) >> {-1, -3}
                board[2][0] = '.'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 2
                if 1 in {1, 4} or 3 in {1, 5} or 1 in {-1, -3}: --> True
                        continue
                #iteration 3
                if 2 in {1, 4} or 4 in {1, 5} or 0 in {-1, -3}: --> False
                {1, 4}.add(2) >> {2, 1, 4}
                posDiag.add(2 + 2) >> {1, 4, 5}
                negDiag.add(2 - 2) >> {0, -1, -3}
                board[2][2] = 'Q'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(2+1) >> backtrack(3):
        if  r==n: --> False
```

```
        for c in range(5):
                #iteration 1
                if 0 in {2, 1, 4} or 3 in {1, 4, 5} or 3 in {0, -1, -3}: --> False
                {2, 1, 4}.add(0) >> {0, 2, 1, 4}
                posDiag.add(3 + 0) >> {1, 3, 4, 5}
                negDiag.add(3 - 0) >> {0, 3, -1, -3}
                board[3][0] = 'Q'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', 'Q', '.', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(3+1) >> backtrack(4):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {0, 2, 1, 4} or 4 in {1, 3, 4, 5} or 4 in {0, 3, -1, -3}: --> True
                        continue
                #iteration 2
                if 1 in {0, 2, 1, 4} or 5 in {1, 3, 4, 5} or 3 in {0, 3, -1, -3}: --> True
                        continue
                #iteration 3
                if 2 in {0, 2, 1, 4} or 6 in {1, 3, 4, 5} or 2 in {0, 3, -1, -3}: --> True
                        continue
                #iteration 4
                if 3 in {0, 2, 1, 4} or 7 in {1, 3, 4, 5} or 1 in {0, 3, -1, -3}: --> False
                {0, 2, 1, 4}.add(3) >> {0, 2, 1, 4, 3}
                posDiag.add(4 + 3) >> {1, 3, 4, 5, 7}
                negDiag.add(4 - 3) >> {1, 0, 3, -1, -3}
                board[4][3] = 'Q'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', 'Q', '.', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(4+1) >> backtrack(5):
        if  r==n: --> True
                🟡 #Base Case 🟡
                res.append([''.join(row) for row in board])
>>
[['Q....', '..Q..', '....Q', '.Q...', '...Q.'],
 ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
 ['.Q...', '...Q.', 'Q....', '..Q..', '....Q'],
 ['.Q...', '....Q', '..Q..', 'Q....', '...Q.']]
                return
                #Maximum recursive depth reached
                🔴 #End of recursive block🔴

                {0, 2, 1, 4, 3}.remove(3) >> {0, 2, 1, 4}
                posDiag.remove(4 + 3) >> {1, 3, 4, 5}
                negDiag.remove(4 - 3) >> {0, 3, -1, -3}
                board[4][3] = '.'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', 'Q', '.', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 5
                if 4 in {0, 2, 1, 4} or 8 in {1, 3, 4, 5} or 0 in {0, 3, -1, -3}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.']]
```

```
        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {0, 2, 1, 4}.remove(0) >> {2, 1, 4}
                posDiag.remove(3 + 0) >> {1, 4, 5}
                negDiag.remove(3 - 0) >> {0, -1, -3}
                board[3][0] = '.'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 2
                if 1 in {2, 1, 4} or 4 in {1, 4, 5} or 2 in {0, -1, -3}: --> True
                        continue
                #iteration 3
                if 2 in {2, 1, 4} or 5 in {1, 4, 5} or 1 in {0, -1, -3}: --> True
                        continue
                #iteration 4
                if 3 in {2, 1, 4} or 6 in {1, 4, 5} or 0 in {0, -1, -3}: --> True
                        continue
                #iteration 5
                if 4 in {2, 1, 4} or 7 in {1, 4, 5} or -1 in {0, -1, -3}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {2, 1, 4}.remove(2) >> {1, 4}
                posDiag.remove(2 + 2) >> {1, 5}
                negDiag.remove(2 - 2) >> {-1, -3}
                board[2][2] = '.'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 4
                if 3 in {1, 4} or 5 in {1, 5} or -1 in {-1, -3}: --> True
                        continue
                #iteration 5
                if 4 in {1, 4} or 6 in {1, 5} or -2 in {-1, -3}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {1, 4}.remove(4) >> {1}
                posDiag.remove(1 + 4) >> {1}
                negDiag.remove(1 - 4) >> {-1}
                board[1][4] = '.'
>>
[['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
```

```
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {1}.remove(1) >> set()
                posDiag.remove(0 + 1) >> set()
                negDiag.remove(0 - 1) >> set()
                board[0][1] = '.'
>>
[['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 3
                if 2 in set() or 2 in set() or -2 in set(): --> False
                set().add(2) >> {2}
                posDiag.add(0 + 2) >> {2}
                negDiag.add(0 - 2) >> {-2}
                board[0][2] = 'Q'
>>
[['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(0+1) >> backtrack(1):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {2} or 1 in {2} or 1 in {-2}: --> False
                {2}.add(0) >> {0, 2}
                posDiag.add(1 + 0) >> {1, 2}
                negDiag.add(1 - 0) >> {1, -2}
                board[1][0] = 'Q'
>>
[['.', '.', 'Q', '.', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(1+1) >> backtrack(2):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {0, 2} or 2 in {1, 2} or 2 in {1, -2}: --> True
                        continue
                #iteration 2
                if 1 in {0, 2} or 3 in {1, 2} or 1 in {1, -2}: --> True
                        continue
                #iteration 3
                if 2 in {0, 2} or 4 in {1, 2} or 0 in {1, -2}: --> True
                        continue
                #iteration 4
                if 3 in {0, 2} or 5 in {1, 2} or -1 in {1, -2}: --> False
                {0, 2}.add(3) >> {3, 0, 2}
                posDiag.add(2 + 3) >> {5, 1, 2}
                negDiag.add(2 - 3) >> {1, -2, -1}
                board[2][3] = 'Q'
>>
[['.', '.', 'Q', '.', '.'],
 ['Q', '.', '.', '.', '.'],
```

```
 ['.', '.', '.', 'Q', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(2+1) >> backtrack(3):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {3, 0, 2} or 3 in {5, 1, 2} or 3 in {1, -2, -1}: --> True
                        continue
                #iteration 2
                if 1 in {3, 0, 2} or 4 in {5, 1, 2} or 2 in {1, -2, -1}: --> False
                {3, 0, 2}.add(1) >> {1, 3, 0, 2}
                posDiag.add(3 + 1) >> {4, 5, 1, 2}
                negDiag.add(3 - 1) >> {2, 1, -2, -1}
                board[3][1] = 'Q'
>>
[['.', '.', 'Q', '.', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(3+1) >> backtrack(4):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {1, 3, 0, 2} or 4 in {4, 5, 1, 2} or 4 in {2, 1, -2, -1}: --> True
                        continue
                #iteration 2
                if 1 in {1, 3, 0, 2} or 5 in {4, 5, 1, 2} or 3 in {2, 1, -2, -1}: --> True
                        continue
                #iteration 3
                if 2 in {1, 3, 0, 2} or 6 in {4, 5, 1, 2} or 2 in {2, 1, -2, -1}: --> True
                        continue
                #iteration 4
                if 3 in {1, 3, 0, 2} or 7 in {4, 5, 1, 2} or 1 in {2, 1, -2, -1}: --> True
                        continue
                #iteration 5
                if 4 in {1, 3, 0, 2} or 8 in {4, 5, 1, 2} or 0 in {2, 1, -2, -1}: --> False
                {1, 3, 0, 2}.add(4) >> {1, 3, 0, 2, 4}
                posDiag.add(4 + 4) >> {4, 5, 1, 2, 8}
                negDiag.add(4 - 4) >> {0, 2, 1, -2, -1}
                board[4][4] = 'Q'
>>
[['.', '.', 'Q', '.', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q']]
                🟢 #Starting Recursive block 🟢
backtrack(4+1) >> backtrack(5):
        if  r==n: --> True
                🟡 #Base Case 🟡
                res.append([''.join(row) for row in board])
>>
[['Q....', '..Q..', '....Q', '.Q...', '...Q.'],
 ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
 ['.Q...', '...Q.', 'Q....', '..Q..', '....Q'],
 ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'],
 ['..Q..', 'Q....', '...Q.', '.Q...', '....Q']]
                return
                #Maximum recursive depth reached
                🔴 #End of recursive block🔴

                {1, 3, 0, 2, 4}.remove(4) >> {1, 3, 0, 2}
                posDiag.remove(4 + 4) >> {4, 5, 1, 2}
```

```
                negDiag.remove(4 - 4) >> {2, 1, -2, -1}
                board[4][4] = '.'
>>
[['.', '.', 'Q', '.', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q']]

        ▩▩▩▩ #End of a level ▩▩▩▩

                🔴 #End of recursive block🔴

                {1, 3, 0, 2}.remove(1) >> {3, 0, 2}
                posDiag.remove(3 + 1) >> {5, 1, 2}
                negDiag.remove(3 - 1) >> {1, -2, -1}
                board[3][1] = '.'
>>
[['.', '.', 'Q', '.', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 3
                if 2 in {3, 0, 2} or 5 in {5, 1, 2} or 1 in {1, -2, -1}: --> True
                        continue
                #iteration 4
                if 3 in {3, 0, 2} or 6 in {5, 1, 2} or 0 in {1, -2, -1}: --> True
                        continue
                #iteration 5
                if 4 in {3, 0, 2} or 7 in {5, 1, 2} or -1 in {1, -2, -1}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q']]

        ▩▩◆◆▩ #End of a level ▩▩▩▩

                🔴 #End of recursive block🔴

                {3, 0, 2}.remove(3) >> {0, 2}
                posDiag.remove(2 + 3) >> {1, 2}
                negDiag.remove(2 - 3) >> {1, -2}
                board[2][3] = '.'
>>
[['.', '.', 'Q', '.', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 5
                if 4 in {0, 2} or 6 in {1, 2} or -2 in {1, -2}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q']]

        ▩▩▩▩ #End of a level ▩▩▩▩

                🔴 #End of recursive block🔴

                {0, 2}.remove(0) >> {2}
                posDiag.remove(1 + 0) >> {2}
                negDiag.remove(1 - 0) >> {-2}
```

```
                board[1][0] = '.'
>>
[['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 2
                if 1 in {2} or 2 in {2} or 0 in {-2}: --> True
                        continue
                #iteration 3
                if 2 in {2} or 3 in {2} or -1 in {-2}: --> True
                        continue
                #iteration 4
                if 3 in {2} or 4 in {2} or -2 in {-2}: --> True
                        continue
                #iteration 5
                if 4 in {2} or 5 in {2} or -3 in {-2}: --> False
                {2}.add(4) >> {2, 4}
                posDiag.add(1 + 4) >> {2, 5}
                negDiag.add(1 - 4) >> {-2, -3}
                board[1][4] = 'Q'
>>
[['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(1+1) >> backtrack(2):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {2, 4} or 2 in {2, 5} or 2 in {-2, -3}: --> True
                        continue
                #iteration 2
                if 1 in {2, 4} or 3 in {2, 5} or 1 in {-2, -3}: --> False
                {2, 4}.add(1) >> {1, 2, 4}
                posDiag.add(2 + 1) >> {3, 2, 5}
                negDiag.add(2 - 1) >> {1, -2, -3}
                board[2][1] = 'Q'
>>
[['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(2+1) >> backtrack(3):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {1, 2, 4} or 3 in {3, 2, 5} or 3 in {1, -2, -3}: --> True
                        continue
                #iteration 2
                if 1 in {1, 2, 4} or 4 in {3, 2, 5} or 2 in {1, -2, -3}: --> True
                        continue
                #iteration 3
                if 2 in {1, 2, 4} or 5 in {3, 2, 5} or 1 in {1, -2, -3}: --> True
                        continue
                #iteration 4
                if 3 in {1, 2, 4} or 6 in {3, 2, 5} or 0 in {1, -2, -3}: --> False
                {1, 2, 4}.add(3) >> {3, 1, 2, 4}
                posDiag.add(3 + 3) >> {6, 3, 2, 5}
                negDiag.add(3 - 3) >> {0, 1, -2, -3}
                board[3][3] = 'Q'
>>
```

```
[['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(3+1) >> backtrack(4):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {3, 1, 2, 4} or 4 in {6, 3, 2, 5} or 4 in {0, 1, -2, -3}: --> False
                {3, 1, 2, 4}.add(0) >> {0, 3, 1, 2, 4}
                posDiag.add(4 + 0) >> {4, 6, 3, 2, 5}
                negDiag.add(4 - 0) >> {0, 1, 4, -2, -3}
                board[4][0] = 'Q'
>>
[['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(4+1) >> backtrack(5):
        if  r==n: --> True
                ❓❓❓ #Base Case 🟡
                res.append([''.join(row) for row in board])
>>
[['Q....', '..Q..', '....Q', '.Q...', '...Q.'],
 ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
 ['.Q...', '...Q.', 'Q....', '..Q..', '....Q'],
 ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'],
 ['..Q..', 'Q....', '...Q.', '.Q...', '....Q'],
 ['..Q..', '....Q', '.Q...', '...Q.', 'Q....']]
                return
                #Maximum recursive depth reached
                🔴 #End of recursive block🔴

                {0, 3, 1, 2, 4}.remove(0) >> {3, 1, 2, 4}
                posDiag.remove(4 + 0) >> {6, 3, 2, 5}
                negDiag.remove(4 - 0) >> {0, 1, -2, -3}
                board[4][0] = '.'
>>
[['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 2
                if 1 in {3, 1, 2, 4} or 5 in {6, 3, 2, 5} or 3 in {0, 1, -2, -3}: --> True
                        continue
                #iteration 3
                if 2 in {3, 1, 2, 4} or 6 in {6, 3, 2, 5} or 2 in {0, 1, -2, -3}: --> True
                        continue
                #iteration 4
                if 3 in {3, 1, 2, 4} or 7 in {6, 3, 2, 5} or 1 in {0, 1, -2, -3}: --> True
                        continue
                #iteration 5
                if 4 in {3, 1, 2, 4} or 8 in {6, 3, 2, 5} or 0 in {0, 1, -2, -3}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴
```

```
                {3, 1, 2, 4}.remove(3) >> {1, 2, 4}
                posDiag.remove(3 + 3) >> {3, 2, 5}
                negDiag.remove(3 - 3) >> {1, -2, -3}
                board[3][3] = '.'
>>
[['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 5
                if 4 in {1, 2, 4} or 7 in {3, 2, 5} or -1 in {1, -2, -3}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔁 #End of recursive block🔴

                {1, 2, 4}.remove(1) >> {2, 4}
                posDiag.remove(2 + 1) >> {2, 5}
                negDiag.remove(2 - 1) >> {-2, -3}
                board[2][1] = '.'
>>
[['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 3
                if 2 in {2, 4} or 4 in {2, 5} or 0 in {-2, -3}: --> True
                        continue
                #iteration 4
                if 3 in {2, 4} or 5 in {2, 5} or -1 in {-2, -3}: --> True
                        continue
                #iteration 5
                if 4 in {2, 4} or 6 in {2, 5} or -2 in {-2, -3}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {2, 4}.remove(4) >> {2}
                posDiag.remove(1 + 4) >> {2}
                negDiag.remove(1 - 4) >> {-2}
                board[1][4] = '.'
>>
[['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {2}.remove(2) >> set()
```

```
                posDiag.remove(0 + 2) >> set()
                negDiag.remove(0 - 2) >> set()
                board[0][2] = '.'
>>
[['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 4
                if 3 in set() or 3 in set() or -3 in set(): --> False
                set().add(3) >> {3}
                posDiag.add(0 + 3) >> {3}
                negDiag.add(0 - 3) >> {-3}
                board[0][3] = 'Q'
>>
[['.', '.', '.', 'Q', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                ?? #Starting Recursive block 🟢
backtrack(0+1) >> backtrack(1):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {3} or 1 in {3} or 1 in {-3}: --> False
                {3}.add(0) >> {0, 3}
                posDiag.add(1 + 0) >> {1, 3}
                negDiag.add(1 - 0) >> {1, -3}
                board[1][0] = 'Q'
>>
[['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(1+1) >> backtrack(2):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {0, 3} or 2 in {1, 3} or 2 in {1, -3}: --> True
                        continue
                #iteration 2
                if 1 in {0, 3} or 3 in {1, 3} or 1 in {1, -3}: --> True
                        continue
                #iteration 3
                if 2 in {0, 3} or 4 in {1, 3} or 0 in {1, -3}: --> False
                {0, 3}.add(2) >> {2, 0, 3}
                posDiag.add(2 + 2) >> {4, 1, 3}
                negDiag.add(2 - 2) >> {0, 1, -3}
                board[2][2] = 'Q'
>>
[['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(2+1) >> backtrack(3):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {2, 0, 3} or 3 in {4, 1, 3} or 3 in {0, 1, -3}: --> True
                        continue
                #iteration 2
```

```
                if 1 in {2, 0, 3} or 4 in {4, 1, 3} or 2 in {0, 1, -3}: --> True
                        continue
                #iteration 3
                if 2 in {2, 0, 3} or 5 in {4, 1, 3} or 1 in {0, 1, -3}: --> True
                        continue
                #iteration 4
                if 3 in {2, 0, 3} or 6 in {4, 1, 3} or 0 in {0, 1, -3}: --> True
                        continue
                #iteration 5
                if 4 in {2, 0, 3} or 7 in {4, 1, 3} or -1 in {0, 1, -3}: --> False
                {2, 0, 3}.add(4) >> {4, 2, 0, 3}
                posDiag.add(3 + 4) >> {7, 4, 1, 3}
                negDiag.add(3 - 4) >> {0, 1, -1, -3}
                board[3][4] = 'Q'
>>
[['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(3+1) >> backtrack(4):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {4, 2, 0, 3} or 4 in {7, 4, 1, 3} or 4 in {0, 1, -1, -3}: --> True
                        continue
                #iteration 2
                if 1 in {4, 2, 0, 3} or 5 in {7, 4, 1, 3} or 3 in {0, 1, -1, -3}: --> False
                {4, 2, 0, 3}.add(1) >> {1, 4, 2, 0, 3}
                posDiag.add(4 + 1) >> {5, 7, 4, 1, 3}
                negDiag.add(4 - 1) >> {3, 0, 1, -1, -3}
                board[4][1] = 'Q'
>>
[['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', 'Q', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(4+1) >> backtrack(5):
        if  r==n: --> True
                🟡 #Base Case 🟡
                res.append([''.join(row) for row in board])
>>
[['Q....', '..Q..', '....Q', '.Q...', '...Q.'],
 ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
 ['.Q...', '...Q.', 'Q....', '..Q..', '....Q'],
 ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'],
 ['..Q..', 'Q....', '...Q.', '.Q...', '....Q'],
 ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'],
 ['...Q.', 'Q....', '..Q..', '....Q', '.Q...']]
                return
                #Maximum recursive depth reached
                🔴 #End of recursive block🔴

                {1, 4, 2, 0, 3}.remove(1) >> {4, 2, 0, 3}
                posDiag.remove(4 + 1) >> {7, 4, 1, 3}
                negDiag.remove(4 - 1) >> {0, 1, -1, -3}
                board[4][1] = '.'
>>
[['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.']]
                #iteration 3
```

```
                if 2 in {4, 2, 0, 3} or 6 in {7, 4, 1, 3} or 2 in {0, 1, -1, -3}: --> True
                        continue
                #iteration 4
                if 3 in {4, 2, 0, 3} or 7 in {7, 4, 1, 3} or 1 in {0, 1, -1, -3}: --> True
                        continue
                #iteration 5
                if 4 in {4, 2, 0, 3} or 8 in {7, 4, 1, 3} or 0 in {0, 1, -1, -3}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {4, 2, 0, 3}.remove(4) >> {2, 0, 3}
                posDiag.remove(3 + 4) >> {4, 1, 3}
                negDiag.remove(3 - 4) >> {0, 1, -3}
                board[3][4] = '.'
>>
[['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {2, 0, 3}.remove(2) >> {0, 3}
                posDiag.remove(2 + 2) >> {1, 3}
                negDiag.remove(2 - 2) >> {1, -3}
                board[2][2] = '.'
>>
[['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 4
                if 3 in {0, 3} or 5 in {1, 3} or -1 in {1, -3}: --> True
                        continue
                #iteration 5
                if 4 in {0, 3} or 6 in {1, 3} or -2 in {1, -3}: --> False
                {0, 3}.add(4) >> {4, 0, 3}
                posDiag.add(2 + 4) >> {6, 1, 3}
                negDiag.add(2 - 4) >> {1, -2, -3}
                board[2][4] = 'Q'
>>
[['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(2+1) >> backtrack(3):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {4, 0, 3} or 3 in {6, 1, 3} or 3 in {1, -2, -3}: --> True
```

```
                        continue
                #iteration 2
                if 1 in {4, 0, 3} or 4 in {6, 1, 3} or 2 in {1, -2, -3}: --> False
                {4, 0, 3}.add(1) >> {1, 4, 0, 3}
                posDiag.add(3 + 1) >> {4, 6, 1, 3}
                negDiag.add(3 - 1) >> {2, 1, -2, -3}
                board[3][1] = 'Q'
>>
[['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(3+1) >> backtrack(4):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {1, 4, 0, 3} or 4 in {4, 6, 1, 3} or 4 in {2, 1, -2, -3}: --> True
                        continue
                #iteration 2
                if 1 in {1, 4, 0, 3} or 5 in {4, 6, 1, 3} or 3 in {2, 1, -2, -3}: --> True
                        continue
                #iteration 3
                if 2 in {1, 4, 0, 3} or 6 in {4, 6, 1, 3} or 2 in {2, 1, -2, -3}: --> True
                        continue
                #iteration 4
                if 3 in {1, 4, 0, 3} or 7 in {4, 6, 1, 3} or 1 in {2, 1, -2, -3}: --> True
                        continue
                #iteration 5
                if 4 in {1, 4, 0, 3} or 8 in {4, 6, 1, 3} or 0 in {2, 1, -2, -3}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {1, 4, 0, 3}.remove(1) >> {4, 0, 3}
                posDiag.remove(3 + 1) >> {6, 1, 3}
                negDiag.remove(3 - 1) >> {1, -2, -3}
                board[3][1] = '.'
>>
[['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 3
                if 2 in {4, 0, 3} or 5 in {6, 1, 3} or 1 in {1, -2, -3}: --> True
                        continue
                #iteration 4
                if 3 in {4, 0, 3} or 6 in {6, 1, 3} or 0 in {1, -2, -3}: --> True
                        continue
                #iteration 5
                if 4 in {4, 0, 3} or 7 in {6, 1, 3} or -1 in {1, -2, -3}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁
```

```
                🔴 #End of recursive block🔴

                {4, 0, 3}.remove(4) >> {0, 3}
                posDiag.remove(2 + 4) >> {1, 3}
                negDiag.remove(2 - 4) >> {1, -3}
                board[2][4] = '.'
>>
[['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {0, 3}.remove(0) >> {3}
                posDiag.remove(1 + 0) >> {3}
                negDiag.remove(1 - 0) >> {-3}
                board[1][0] = '.'
>>
[['.', '.', '.', 'Q', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 2
                if 1 in {3} or 2 in {3} or 0 in {-3}: --> False
                {3}.add(1) >> {1, 3}
                posDiag.add(1 + 1) >> {2, 3}
                negDiag.add(1 - 1) >> {0, -3}
                board[1][1] = 'Q'
>>
[['.', '.', '.', 'Q', '.'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(1+1) >> backtrack(2):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {1, 3} or 2 in {2, 3} or 2 in {0, -3}: --> True
                        continue
                #iteration 2
                if 1 in {1, 3} or 3 in {2, 3} or 1 in {0, -3}: --> True
                        continue
                #iteration 3
                if 2 in {1, 3} or 4 in {2, 3} or 0 in {0, -3}: --> True
                        continue
                #iteration 4
                if 3 in {1, 3} or 5 in {2, 3} or -1 in {0, -3}: --> True
                        continue
                #iteration 5
                if 4 in {1, 3} or 6 in {2, 3} or -2 in {0, -3}: --> False
                {1, 3}.add(4) >> {4, 1, 3}
                posDiag.add(2 + 4) >> {6, 2, 3}
                negDiag.add(2 - 4) >> {0, -2, -3}
                board[2][4] = 'Q'
>>
[['.', '.', '.', 'Q', '.'],
```

```
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(2+1) >> backtrack(3):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {4, 1, 3} or 3 in {6, 2, 3} or 3 in {0, -2, -3}: --> True
                        continue
                #iteration 2
                if 1 in {4, 1, 3} or 4 in {6, 2, 3} or 2 in {0, -2, -3}: --> True
                        continue
                #iteration 3
                if 2 in {4, 1, 3} or 5 in {6, 2, 3} or 1 in {0, -2, -3}: --> False
                {4, 1, 3}.add(2) >> {2, 4, 1, 3}
                posDiag.add(3 + 2) >> {5, 6, 2, 3}
                negDiag.add(3 - 2) >> {1, 0, -2, -3}
                board[3][2] = 'Q'
>>
[['.', '.', '.', 'Q', '.'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(3+1) >> backtrack(4):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {2, 4, 1, 3} or 4 in {5, 6, 2, 3} or 4 in {1, 0, -2, -3}: --> False
                {2, 4, 1, 3}.add(0) >> {0, 2, 4, 1, 3}
                posDiag.add(4 + 0) >> {4, 5, 6, 2, 3}
                negDiag.add(4 - 0) >> {1, 0, 4, -2, -3}
                board[4][0] = 'Q'
>>
[['.', '.', '.', 'Q', '.'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', 'Q', '.', '.'],
 ['Q', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(4+1) >> backtrack(5):
        if  r==n: --> True
                🟡 #Base Case 🟡
                res.append([''.join(row) for row in board])
>>
[['Q....', '..Q..', '....Q', '.Q...', '...Q.'],
 ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
 ['.Q...', '...Q.', 'Q....', '..Q..', '....Q'],
 ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'],
 ['..Q..', 'Q....', '...Q.', '.Q...', '....Q'],
 ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'],
 ['...Q.', 'Q....', '..Q..', '....Q', '.Q...'],
 ['...Q.', '.Q...', '....Q', '..Q..', 'Q....']]
                return
                #Maximum recursive depth reached
                🔴 #End of recursive block🔴

                {0, 2, 4, 1, 3}.remove(0) >> {2, 4, 1, 3}
                posDiag.remove(4 + 0) >> {5, 6, 2, 3}
                negDiag.remove(4 - 0) >> {1, 0, -2, -3}
                board[4][0] = '.'
>>
[['.', '.', '.', 'Q', '.'],
 ['.', 'Q', '.', '.', '.'],
```

```
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 2
                if 1 in {2, 4, 1, 3} or 5 in {5, 6, 2, 3} or 3 in {1, 0, -2, -3}: --> True
                        continue
                #iteration 3
                if 2 in {2, 4, 1, 3} or 6 in {5, 6, 2, 3} or 2 in {1, 0, -2, -3}: --> True
                        continue
                #iteration 4
                if 3 in {2, 4, 1, 3} or 7 in {5, 6, 2, 3} or 1 in {1, 0, -2, -3}: --> True
                        continue
                #iteration 5
                if 4 in {2, 4, 1, 3} or 8 in {5, 6, 2, 3} or 0 in {1, 0, -2, -3}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...'], ['...Q.', '.Q...', '....Q', '..Q..', 'Q....']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🔴🔴🏁

                🔴 #End of recursive block🔴

                {2, 4, 1, 3}.remove(2) >> {4, 1, 3}
                posDiag.remove(3 + 2) >> {6, 2, 3}
                negDiag.remove(3 - 2) >> {0, -2, -3}
                board[3][2] = '.'
>>
[['.', '.', '.', 'Q', '.'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 4
                if 3 in {4, 1, 3} or 6 in {6, 2, 3} or 0 in {0, -2, -3}: --> True
                        continue
                #iteration 5
                if 4 in {4, 1, 3} or 7 in {6, 2, 3} or -1 in {0, -2, -3}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...'], ['...Q.', '.Q...', '....Q', '..Q..', 'Q....']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🔴🔴🏁

                🔴 #End of recursive block🔴

                {4, 1, 3}.remove(4) >> {1, 3}
                posDiag.remove(2 + 4) >> {2, 3}
                negDiag.remove(2 - 4) >> {0, -3}
                board[2][4] = '.'
>>
[['.', '.', '.', 'Q', '.'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...'], ['...Q.', '.Q...', '....Q', '..Q..', 'Q....']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴
```

```
                {1, 3}.remove(1) >> {3}
                posDiag.remove(1 + 1) >> {3}
                negDiag.remove(1 - 1) >> {-3}
                board[1][1] = '.'
>>
[['.', '.', '.', 'Q', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 3
                if 2 in {3} or 3 in {3} or -1 in {-3}: --> True
                        continue
                #iteration 4
                if 3 in {3} or 4 in {3} or -2 in {-3}: --> True
                        continue
                #iteration 5
                if 4 in {3} or 5 in {3} or -3 in {-3}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...'], ['...Q.', '.Q...', '....Q', '..Q..', 'Q....']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {3}.remove(3) >> set()
                posDiag.remove(0 + 3) >> set()
                negDiag.remove(0 - 3) >> set()
                board[0][3] = '.'
>>
[['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 5
                if 4 in set() or 4 in set() or -4 in set(): --> False
                set().add(4) >> {4}
                posDiag.add(0 + 4) >> {4}
                negDiag.add(0 - 4) >> {-4}
                board[0][4] = 'Q'
>>
[['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(0+1) >> backtrack(1):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {4} or 1 in {4} or 1 in {-4}: --> False
                {4}.add(0) >> {0, 4}
                posDiag.add(1 + 0) >> {1, 4}
                negDiag.add(1 - 0) >> {1, -4}
                board[1][0] = 'Q'
>>
[['.', '.', '.', '.', 'Q'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
```

```
                🟢 #Starting Recursive block 🟢
backtrack(1+1) >> backtrack(2):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {0, 4} or 2 in {1, 4} or 2 in {1, -4}: --> True
                        continue
                #iteration 2
                if 1 in {0, 4} or 3 in {1, 4} or 1 in {1, -4}: --> True
                        continue
                #iteration 3
                if 2 in {0, 4} or 4 in {1, 4} or 0 in {1, -4}: --> True
                        continue
                #iteration 4
                if 3 in {0, 4} or 5 in {1, 4} or -1 in {1, -4}: --> False
                {0, 4}.add(3) >> {3, 0, 4}
                posDiag.add(2 + 3) >> {5, 1, 4}
                negDiag.add(2 - 3) >> {1, -1, -4}
                board[2][3] = 'Q'
>>
[['.', '.', '.', '.', 'Q'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(2+1) >> backtrack(3):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {3, 0, 4} or 3 in {5, 1, 4} or 3 in {1, -1, -4}: --> True
                        continue
                #iteration 2
                if 1 in {3, 0, 4} or 4 in {5, 1, 4} or 2 in {1, -1, -4}: --> True
                        continue
                #iteration 3
                if 2 in {3, 0, 4} or 5 in {5, 1, 4} or 1 in {1, -1, -4}: --> True
                        continue
                #iteration 4
                if 3 in {3, 0, 4} or 6 in {5, 1, 4} or 0 in {1, -1, -4}: --> True
                        continue
                #iteration 5
                if 4 in {3, 0, 4} or 7 in {5, 1, 4} or -1 in {1, -1, -4}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...'], ['...Q.', '.Q...', '....Q', '..Q..', 'Q....']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {3, 0, 4}.remove(3) >> {0, 4}
                posDiag.remove(2 + 3) >> {1, 4}
                negDiag.remove(2 - 3) >> {1, -4}
                board[2][3] = '.'
>>
[['.', '.', '.', '.', 'Q'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 5
                if 4 in {0, 4} or 6 in {1, 4} or -2 in {1, -4}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
```

```
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...'], ['...Q.', '.Q...', '....Q', '..Q..', 'Q....']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {0, 4}.remove(0) >> {4}
                posDiag.remove(1 + 0) >> {4}
                negDiag.remove(1 - 0) >> {-4}
                board[1][0] = '.'
>>
[['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 2
                if 1 in {4} or 2 in {4} or 0 in {-4}: --> False
                {4}.add(1) >> {1, 4}
                posDiag.add(1 + 1) >> {2, 4}
                negDiag.add(1 - 1) >> {0, -4}
                board[1][1] = 'Q'
>>
[['.', '.', '.', '.', 'Q'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(1+1) >> backtrack(2):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {1, 4} or 2 in {2, 4} or 2 in {0, -4}: --> True
                        continue
                #iteration 2
                if 1 in {1, 4} or 3 in {2, 4} or 1 in {0, -4}: --> True
                        continue
                #iteration 3
                if 2 in {1, 4} or 4 in {2, 4} or 0 in {0, -4}: --> True
                        continue
                #iteration 4
                if 3 in {1, 4} or 5 in {2, 4} or -1 in {0, -4}: --> False
                {1, 4}.add(3) >> {3, 1, 4}
                posDiag.add(2 + 3) >> {5, 2, 4}
                negDiag.add(2 - 3) >> {0, -1, -4}
                board[2][3] = 'Q'
>>
[['.', '.', '.', '.', 'Q'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(2+1) >> backtrack(3):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {3, 1, 4} or 3 in {5, 2, 4} or 3 in {0, -1, -4}: --> False
                {3, 1, 4}.add(0) >> {0, 3, 1, 4}
                posDiag.add(3 + 0) >> {3, 5, 2, 4}
                negDiag.add(3 - 0) >> {3, 0, -1, -4}
                board[3][0] = 'Q'
>>
[['.', '.', '.', '.', 'Q'],
```

```
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(3+1) >> backtrack(4):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {0, 3, 1, 4} or 4 in {3, 5, 2, 4} or 4 in {3, 0, -1, -4}: --> True
                        continue
                #iteration 2
                if 1 in {0, 3, 1, 4} or 5 in {3, 5, 2, 4} or 3 in {3, 0, -1, -4}: --> True
                        continue
                #iteration 3
                if 2 in {0, 3, 1, 4} or 6 in {3, 5, 2, 4} or 2 in {3, 0, -1, -4}: --> False
                {0, 3, 1, 4}.add(2) >> {2, 0, 3, 1, 4}
                posDiag.add(4 + 2) >> {6, 3, 5, 2, 4}
                negDiag.add(4 - 2) >> {2, 3, 0, -1, -4}
                board[4][2] = 'Q'
>>
[['.', '.', '.', '.', 'Q'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', 'Q', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(4+1) >> backtrack(5):
        if  r==n: --> True
                🟡 #Base Case 🟡
                res.append([''.join(row) for row in board])
>>
[['Q....', '..Q..', '....Q', '.Q...', '...Q.'],
 ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
 ['.Q...', '...Q.', 'Q....', '..Q..', '....Q'],
 ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'],
 ['..Q..', 'Q....', '...Q.', '.Q...', '....Q'],
 ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'],
 ['...Q.', 'Q....', '..Q..', '....Q', '.Q...'],
 ['...Q.', '.Q...', '....Q', '..Q..', 'Q....'],
 ['....Q', '.Q...', '...Q.', 'Q....', '..Q..']]
                return
                #Maximum recursive depth reached
                🔴 #End of recursive block🔴

                {2, 0, 3, 1, 4}.remove(2) >> {0, 3, 1, 4}
                posDiag.remove(4 + 2) >> {3, 5, 2, 4}
                negDiag.remove(4 - 2) >> {3, 0, -1, -4}
                board[4][2] = '.'
>>
[['.', '.', '.', '.', 'Q'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 4
                if 3 in {0, 3, 1, 4} or 7 in {3, 5, 2, 4} or 1 in {3, 0, -1, -4}: --> True
                        continue
                #iteration 5
                if 4 in {0, 3, 1, 4} or 8 in {3, 5, 2, 4} or 0 in {3, 0, -1, -4}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...'], ['...Q.', '.Q...', '....Q', '..Q..', 'Q....'], ['....Q', '.Q...', '...Q.', 'Q....', '..Q..']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁
```

```
                🔴 #End of recursive block🔴

                {0, 3, 1, 4}.remove(0) >> {3, 1, 4}
                posDiag.remove(3 + 0) >> {5, 2, 4}
                negDiag.remove(3 - 0) >> {0, -1, -4}
                board[3][0] = '.'
>>
[['.', '.', '.', '.', 'Q'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 2
                if 1 in {3, 1, 4} or 4 in {5, 2, 4} or 2 in {0, -1, -4}: --> True
                        continue
                #iteration 3
                if 2 in {3, 1, 4} or 5 in {5, 2, 4} or 1 in {0, -1, -4}: --> True
                        continue
                #iteration 4
                if 3 in {3, 1, 4} or 6 in {5, 2, 4} or 0 in {0, -1, -4}: --> True
                        continue
                #iteration 5
                if 4 in {3, 1, 4} or 7 in {5, 2, 4} or -1 in {0, -1, -4}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...'], ['...Q.', '.Q...', '....Q', '..Q..', 'Q....'], ['....Q', '.Q...', '...Q.', 'Q....', '..Q..']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {3, 1, 4}.remove(3) >> {1, 4}
                posDiag.remove(2 + 3) >> {2, 4}
                negDiag.remove(2 - 3) >> {0, -4}
                board[2][3] = '.'
>>
[['.', '.', '.', '.', 'Q'],
 ['.', 'Q', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 5
                if 4 in {1, 4} or 6 in {2, 4} or -2 in {0, -4}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...'], ['...Q.', '.Q...', '....Q', '..Q..', 'Q....'], ['....Q', '.Q...', '...Q.', 'Q....', '..Q..']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴

                {1, 4}.remove(1) >> {4}
                posDiag.remove(1 + 1) >> {4}
                negDiag.remove(1 - 1) >> {-4}
                board[1][1] = '.'
>>
[['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 3
```

```
                if 2 in {4} or 3 in {4} or -1 in {-4}: --> False
                {4}.add(2) >> {2, 4}
                posDiag.add(1 + 2) >> {3, 4}
                negDiag.add(1 - 2) >> {-1, -4}
                board[1][2] = 'Q'
>>
[['.', '.', '.', '.', 'Q'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(1+1) >> backtrack(2):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {2, 4} or 2 in {3, 4} or 2 in {-1, -4}: --> False
                {2, 4}.add(0) >> {0, 2, 4}
                posDiag.add(2 + 0) >> {2, 3, 4}
                negDiag.add(2 - 0) >> {2, -1, -4}
                board[2][0] = 'Q'
>>
[['.', '.', '.', '.', 'Q'],
 ['.', '.', 'Q', '.', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(2+1) >> backtrack(3):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {0, 2, 4} or 3 in {2, 3, 4} or 3 in {2, -1, -4}: --> True
                        continue
                #iteration 2
                if 1 in {0, 2, 4} or 4 in {2, 3, 4} or 2 in {2, -1, -4}: --> True
                        continue
                #iteration 3
                if 2 in {0, 2, 4} or 5 in {2, 3, 4} or 1 in {2, -1, -4}: --> True
                        continue
                #iteration 4
                if 3 in {0, 2, 4} or 6 in {2, 3, 4} or 0 in {2, -1, -4}: --> False
                {0, 2, 4}.add(3) >> {3, 0, 2, 4}
                posDiag.add(3 + 3) >> {6, 2, 3, 4}
                negDiag.add(3 - 3) >> {0, 2, -1, -4}
                board[3][3] = 'Q'
>>
[['.', '.', '.', '.', 'Q'],
 ['.', '.', 'Q', '.', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', '.', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(3+1) >> backtrack(4):
        if  r==n: --> False
        for c in range(5):
                #iteration 1
                if 0 in {3, 0, 2, 4} or 4 in {6, 2, 3, 4} or 4 in {0, 2, -1, -4}: --> True
                        continue
                #iteration 2
                if 1 in {3, 0, 2, 4} or 5 in {6, 2, 3, 4} or 3 in {0, 2, -1, -4}: --> False
                {3, 0, 2, 4}.add(1) >> {1, 3, 0, 2, 4}
                posDiag.add(4 + 1) >> {5, 6, 2, 3, 4}
                negDiag.add(4 - 1) >> {3, 0, 2, -1, -4}
                board[4][1] = 'Q'
>>
[['.', '.', '.', '.', 'Q'],
```

```
 ['.', '.', 'Q', '.', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', 'Q', '.', '.', '.']]
                🟢 #Starting Recursive block 🟢
backtrack(4+1) >> backtrack(5):
        if  r==n: --> True
                🟡 #Base Case 🟡
                res.append(['.join(row) for row in board])
>>
[['Q....', '..Q..', '....Q', '.Q...', '...Q.'],
 ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
 ['.Q...', '...Q.', 'Q....', '..Q..', '....Q'],
 ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'],
 ['..Q..', 'Q....', '...Q.', '.Q...', '....Q'],
 ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'],
 ['...Q.', 'Q....', '..Q..', '....Q', '.Q...'],
 ['...Q.', '.Q...', '....Q', '..Q..', 'Q....'],
 ['....Q', '.Q...', '...Q.', 'Q....', '..Q..'],
 ['....Q', '..Q..', 'Q....', '...Q.', '.Q...']]
                return
                #Maximum recursive depth reached
                🔴 #End of recursive block🔴

                {1, 3, 0, 2, 4}.remove(1) >> {3, 0, 2, 4}
                posDiag.remove(4 + 1) >> {6, 2, 3, 4}
                negDiag.remove(4 - 1) >> {0, 2, -1, -4}
                board[4][1] = '.'
>>
[['.', '.', '.', '.', 'Q'],
 ['.', '.', 'Q', '.', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', 'Q', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 3
                if 2 in {3, 0, 2, 4} or 6 in {6, 2, 3, 4} or 2 in {0, 2, -1, -4}: --> True
                        continue
                #iteration 4
                if 3 in {3, 0, 2, 4} or 7 in {6, 2, 3, 4} or 1 in {0, 2, -1, -4}: --> True
                        continue
                #iteration 5
                if 4 in {3, 0, 2, 4} or 8 in {6, 2, 3, 4} or 0 in {0, 2, -1, -4}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...'], ['...Q.', '.Q...', '....Q', '..Q..', 'Q....'], ['....Q', '.Q...', '...Q.', 'Q....', '..Q..'], ['....Q',
'..Q..', 'Q....', '...Q.', '.Q...']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔲🔲

                {3, 0, 2, 4}.remove(3) >> {0, 2, 4}
                posDiag.remove(3 + 3) >> {2, 3, 4}
                negDiag.remove(3 - 3) >> {2, -1, -4}
                board[3][3] = '.'
>>
[['.', '.', '.', '.', 'Q'],
 ['.', '.', 'Q', '.', '.'],
 ['Q', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 5
                if 4 in {0, 2, 4} or 7 in {2, 3, 4} or -1 in {2, -1, -4}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
```

```
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...'], ['...Q.', '.Q...', '....Q', '..Q..', 'Q....'], ['....Q', '.Q...', '...Q.', 'Q....', '..Q..'], ['....Q',
'..Q..', 'Q....', '...Q.', '.Q...']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴🔴

                {0, 2, 4}.remove(0) >> {2, 4}
                posDiag.remove(2 + 0) >> {3, 4}
                negDiag.remove(2 - 0) >> {-1, -4}
                board[2][0] = '.'
>>
[['.', '.', '.', '.', 'Q'],
 ['.', '.', 'Q', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 2
                if 1 in {2, 4} or 3 in {3, 4} or 1 in {-1, -4}: --> True
                        continue
                #iteration 3
                if 2 in {2, 4} or 4 in {3, 4} or 0 in {-1, -4}: --> True
                        continue
                #iteration 4
                if 3 in {2, 4} or 5 in {3, 4} or -1 in {-1, -4}: --> True
                        continue
                #iteration 5
                if 4 in {2, 4} or 6 in {3, 4} or -2 in {-1, -4}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...'], ['...Q.', '.Q...', '....Q', '..Q..', 'Q....'], ['....Q', '.Q...', '...Q.', 'Q....', '..Q..'], ['....Q',
'..Q..', 'Q....', '...Q.', '.Q...']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴🔴

                {2, 4}.remove(2) >> {4}
                posDiag.remove(1 + 2) >> {4}
                negDiag.remove(1 - 2) >> {-4}
                board[1][2] = '.'
>>
[['.', '.', '.', '.', 'Q'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
                #iteration 4
                if 3 in {4} or 4 in {4} or -2 in {-4}: --> True
                        continue
                #iteration 5
                if 4 in {4} or 5 in {4} or -3 in {-4}: --> True
                        continue
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...'], ['...Q.', '.Q...', '....Q', '..Q..', 'Q....'], ['....Q', '.Q...', '...Q.', 'Q....', '..Q..'], ['....Q',
'..Q..', 'Q....', '...Q.', '.Q...']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

                🔴 #End of recursive block🔴
```

```
                {4}.remove(4) >> set()
                posDiag.remove(0 + 4) >> set()
                negDiag.remove(0 - 4) >> set()
                board[0][4] = '.'
>>
[['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.'],
 ['.', '.', '.', '.', '.']]
        return [['Q....', '..Q..', '....Q', '.Q...', '...Q.'], ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
['.Q...', '...Q.', 'Q....', '..Q..', '....Q'], ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'], ['..Q..', 'Q....',
'...Q.', '.Q...', '....Q'], ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'], ['...Q.', 'Q....', '..Q..', '....Q',
'.Q...'], ['...Q.', '.Q...', '....Q', '..Q..', 'Q....'], ['....Q', '.Q...', '...Q.', 'Q....', '..Q..'], ['....Q',
'..Q..', 'Q....', '...Q.', '.Q...']]

        🏁🏁🏁🏁 #End of a level 🏁🏁🏁🏁

# final output
[['Q....', '..Q..', '....Q', '.Q...', '...Q.'],
 ['Q....', '...Q.', '.Q...', '....Q', '..Q..'],
 ['.Q...', '...Q.', 'Q....', '..Q..', '....Q'],
 ['.Q...', '....Q', '..Q..', 'Q....', '...Q.'],
 ['..Q..', 'Q....', '...Q.', '.Q...', '....Q'],
 ['..Q..', '....Q', '.Q...', '...Q.', 'Q....'],
 ['...Q.', 'Q....', '..Q..', '....Q', '.Q...'],
 ['...Q.', '.Q...', '....Q', '..Q..', 'Q....'],
 ['....Q', '.Q...', '...Q.', 'Q....', '..Q..'],
 ['....Q', '..Q..', 'Q....', '...Q.', '.Q...']]
```

10. Two Sum - II

You are given a list of array and a target value, you have to find the indices of pairs that sum up to the target.

```
def twoSum(nums, target):
    def backtrack(start, target, path):
        if target == 0 and len(path) == 2:
            result.append(list(path))
            return
        if len(path) == 2:
            return
        for i in range(start, len(nums)):
            if i > start and nums[i] == nums[i - 1]:
                continue  # Skip duplicates to avoid duplicate results
            path.append(i)
            backtrack(i + 1, target - nums[i], path)
            path.pop()

    result = []
    nums.sort()
    backtrack(0, target, [])
    return result

# Example usage:
nums = [1, 2, 2, 3, 4, 5, 5]
target = 7
print(twoSum(nums, target))
```