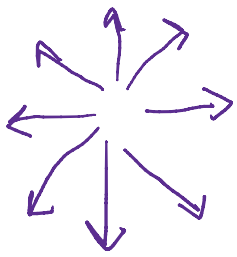
4-Jan-2021

04 January 2022 20:57

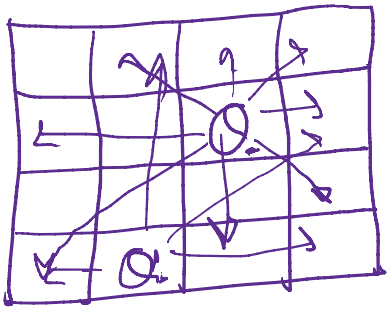
N-Queens :-

Directions

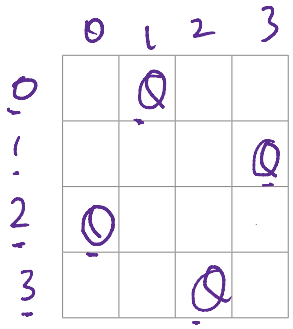


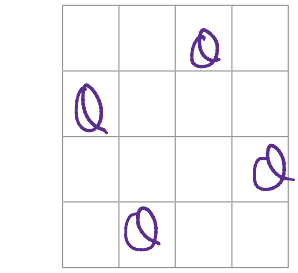
Square Chess board

n x n = n queens

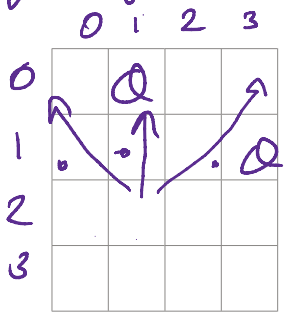


Placement of Queens

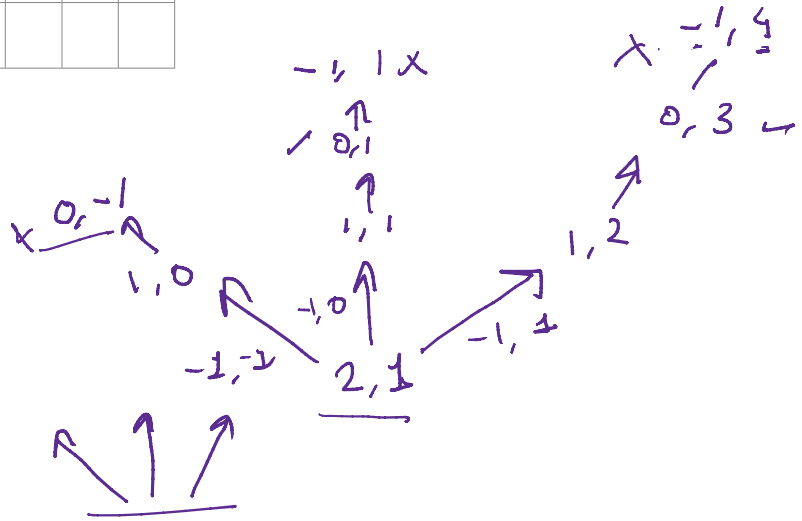


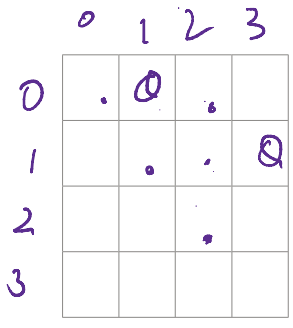


q, p, s, f = 0 x 2 3



(2, 1)





cols

```java
public static boolean canQueenBePlaced(boolean[][] chess, int r, int c) {
    for (int i = 0; i < queenDirections.length; i++) {
        int dr = queenDirections[i][0];
        int dc = queenDirections[i][1];
        int nr = r;
        int nc = c;
        while (nr + dr >= 0 && nc + dc >= 0 && nc + dc < chess.length) {
            nr += dr;
            nc += dc;
            if (chess[nr][nc]) return false;
        }
    }
    return true;
}
```

```java
queenDirections = {{-1, -1}, {-1, 0}, {-1, 1}};
```

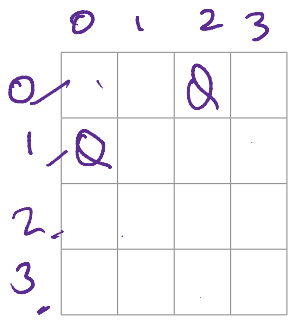
r = 2
c = 2

i = 0 x 2

dr = -1 -1 -1

dc = -1 0 1

nr = 2 x 0 2 + 0 x 1

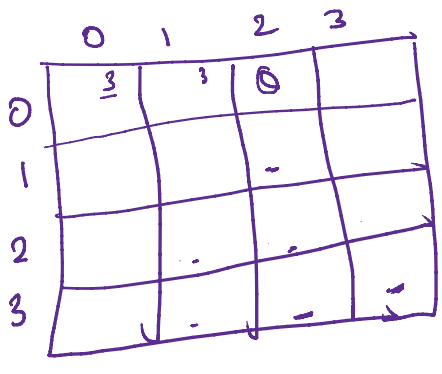


```
nQueens(chess, r, c, qpsf) {
  if(r == chess.length) { found ans }

  queenbcPlaced = canBePlaced(chess, r, c);

  if(qbp) {
    chess[r][c] = true;
    nQueens(chess, r+1, 0, qpsf+1)
    chess[r][c] = false;
  }

  if(c+1 < chess.length) {
    nQueens(chess, r, c+1, qpsf)
  }
```



```java
public static void nQueens(boolean[][] chess, int r,
                           int c, ArrayList<Integer> qpsf) {
    if (r == chess.length) {
        ArrayList<Integer> res = new ArrayList<>(qpsf);
        ans.add(res);
        return;
    }
    boolean canBePlaced = canQueenBePlaced(chess, r, c);
    if (canBePlaced) {
        chess[r][c] = true;
        qpsf.add(c + 1);
        nQueens(chess, r + 1, 0, qpsf);
```

qpsf[

ans: [res [2,4,1,3]]

```
        chess[r][c] = true;
        qpsf.add(c + 1);
2.5     nQueens(chess, r: r + 1, c: 0, qpsf);
        chess[r][c] = false;
        qpsf.remove( index: qpsf.size() - 1);
    }
3   if (c + 1 < chess.length) {
        nQueens(chess, r, c: c + 1, qpsf);
    }
}
```