## Name: Vaishnavi Ingole

## Mail:vaishnavingole54@gmail.com

## Day 16 and 17:

Task 1: The Knight's Tour Problem Create a function bool SolveKnightsTour(int[,] board, int moveX, int moveY, int moveCount, int[] xMove, int[] yMove) that attempts to solve the Knight's Tour problem using backtracking. The function should return true if a solution exists and false otherwise. The board represents the chessboard, moveX and moveY are the current coordinates of the knight, moveCount is the current move count, and xMove[], yMove[] are the possible next moves for the knight. Fill the chessboard such that the knight visits every square exactly once. Keep the chessboard size to 8x8.

```
package com.wipro.algo;
public class KnightsTourAlgo {
     // Possible moves of a Knight
     int[] pathRow = { 2, 2, 1, 1, -1, -1, -2, -2 };
     int[] pathCol = { -1, 1, -2, 2, -2, 2, -1, 1 };
     public static void main(String[] args) {
           KnightsTourAlgo knightTour = new KnightsTourAlgo();
           int[][] visited = new int[8][8];
           visited[0][0] = 1;
           if (!(knightTour.findKnightTour(visited, 0, 0, 1))) {
                System.out.println("Soultion Not Available :(");
           }
     }
     private boolean findKnightTour(int[][] visited, int row, int col,
int move) {
           if (move == 64) {
                for (int i = 0; i < 8; i++) {
                      for (int j = 0; j < 8; j++) {
                           System.out.print(visited[i][j]+" ");
                      System.out.println();
                return true;
           }else {
                for (int i = 0; i < 8; i++) {
                      int rowNew = row + pathRow[i];
                      int colNew = col + pathCol[i];
                      if (ifValidMove(visited, rowNew, colNew)) {
                           visited[rowNew][colNew] = move + 1;
```

```
if (findKnightTour(visited, rowNew, colNew,
move + 1)) {
                                  return true;
                            }
                                  // Backtrack the move
                                  visited[rowNew][colNew] = 0;
                      }
                 }
           }
           return false;
     }
     private boolean ifValidMove(int[][] visited, int rowNew, int
colNew) {
           if(rowNew >=0 && rowNew <8 && colNew >=0 && colNew<8 &&</pre>
visited[rowNew][colNew] ==0)
                 return true;
           return false;
     }
}
```

```
□ Day I ∋Java

                                                                         for (int j = 0; j < 8; j++) {

    ▶ Day20Task

                                                                             System.out.print(visited[i][j]+" ");

■ JRE System Library [JavaSE-17]
                                                   24
                                                                         System.out.println();
25
 v 🚜 com.wipro.algo
                                                   26
                                                                    return true;
   > 🎒 JobSeguence.iava
                                                   27
                                                               }else {
   28
                                                                    for (int i = 0; i < 8; i++) {
   int rowNew = row + pathRow[i];
   >  KnightsTourAlgo.java
                                                                         int colNew = col + pathCol[i];
   > I KnightTourAlgo.java
                                                                        if (ifValidMove(visited, rowNew, colNew)) {
   > 🗓 LCS.java
                                                   32
                                                                             visited[rowNew][colNew] = move + 1;
   > 🔊 LinearSearch.java
                                                                             if (findKnightTour(visited, rowNew, colNew, move + 1)) {
   > 🔊 LongestIncreasingSubsequence.java
                                                   34
                                                                                  return true;
   > I NQueensProblem.java
                                                   35
   OptimalBinarySearchTree.iava
                                                   36
                                                                                 // Backtrack the move
   > 

RatInMaze.iava
   38
                                                                                 visited[rowNew][colNew] = 0;
 > # com.wipro.dll
                                                   39
 > # com.wipro.ds
                                                   40
 > # com.wipro.ll
                                                  41
 > # com.wipro.q
 > # com.wipro.se
                                                  Problems @ Javadoc Declaration Console X
 > # com.wipro.sort
 > # com.wipro.st
                                                  <terminated> KnightsTourAlgo [Java Application] C:\Users\vaish\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0
                                                  1 36 47 50 57 52 61 40
 > # com.wipro.tree
                                                  46 49 58 37 60 39 56 53
 > I module-info.java
                                                  35 2 27 48 51 54 41 62
 Day6Dsa
                                                  26 45 34 59 38 43 32 55
₿ Day7Dsa
3 28 25 44 33 30 63 42
12 15 18 29 24 21 8 31
17 4 13 10 19 6 23 64
> March JRE System Library [JavaSE-17]
                                                  14 11 16 5 22 9 20 7
∨ Æ src
```

**Task 2: Rat in a Maze** mplement a function bool SolveMaze(int[,] maze) that uses backtracking to find a path from the top left corner to the bottom right corner of a maze. The maze is represented by a 2D array where 1s are paths and 0s are walls. Find a rat's path through the maze. The maze size is 6x6.

```
int rowNew = row + pathRow[index];
                      int colNew = col + pathCol[index];
                      if(isValidMove(maze, visited, rowNew, colNew)) {
                            move++;
                            visited[rowNew][colNew] =move;
                            findPathInMaze(maze, visited, rowNew, colNew,
destRow,destCol, move);
                            move--;
                            visited[rowNew][colNew]=0;
                      }
                 }
           }
     }
     private boolean isValidMove(int[][] maze, int[][] visited, int
rowNew, int colNew) {
           return (rowNew >=0 && rowNew <6 && colNew>=0 && colNew<6 &&
maze[rowNew][colNew] ==1 && visited[rowNew][colNew] == 0);
     }
     public static void main(String[] args) {
           int[][] maze = {
                          \{1, 0, 1, 1, 1, 0\},\
                       {1, 1, 1, 1, 0, 0},
                       \{0, 0, 0, 1, 1, 1\},\
                       \{1, 1, 1, 1, 0, 1\},\
                       \{0, 0, 1, 1, 1, 0\},\
                       {1, 1, 0, 1, 1, 1}
                   };
           int[][] visited = new int[6][6];
           visited[0][0] = 1;
           RatInMaze ratInMaze = new RatInMaze();
            ratInMaze.findPathInMaze(maze, visited, 0, 0, 5, 5, 1);
     }
}
```

```
שע Day ו Dava

Day20Task

Day20Ta
                                                                                                                               41
42
                                                                                                                                                              return (rowNew >=0 && rowNew <6 && colNew>=0 && colNew<6 && maze[rowNew]
√ 📂 Day5DSA
    > A JRE System Library [JavaSE-17]
                                                                                                                                43
   44e
45
                                                                                                                                                   public static void main(String[] args) {

√ Æ com.wipro.algo

                                                                                                                                                              int[][] maze = {
              > 🕖 JobSequence.java
                                                                                                                              46
47
48
49
                                                                                                                                                                                               {1, 0, 1, 1, 1, 0},
              > 🗓 Knapsack.java
                                                                                                                                                                                               {1, 1, 1, 1, 0, 0},
              > II KnapsackProblem01.java
                                                                                                                                                                                               {0, 0, 0, 1, 1, 1},
              > 🗓 KnightsTourAlgo.java
                                                                                                                                                                                              {1, 1, 1, 1, 0, 1},
              > II KnightTourAlgo.java
                                                                                                                               50
                                                                                                                                                                                              {0, 0, 1, 1, 1, 0},
              > 🗓 LCS.java
                                                                                                                                                                                              {1, 1, 0, 1, 1, 1}
             > 🔃 LinearSearch.java
              > 🔝 LongestIncreasingSubsequence.java
              > 🗓 NQueensProblem.java
              > 

OptimalBinarySearchTree.java

    Problems @ Javadoc    Declaration    □ Console ×

              RatlnMaze.java
                                                                                                                             <terminated> RatinMaze [Java Application] C\Users\vaish\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.10.v2
              > # com.wipro.dll
                                                                                                                                              4 5 0
        > # com.wipro.ds
                                                                                                                                0
                                                                                                                                       0 0 6 0
                                                                                                                                                                       0
         > # com.wipro.ll
                                                                                                                                0
                                                                                                                                       0 8 7 0
                                                                                                                                                                       0
         > 🔠 com.wipro.q
                                                                                                                                0
                                                                                                                                       0 9 10 11
         > 🔠 com.wipro.se
                                                                                                                                               0 0 12 13
         > # com.wipro.sort
        > # com.wipro.st
                                                                                                                                1
                                                                                                                                       0 0 0
         > # com.wipro.tree
                                                                                                                                                        5
        > I module-info.iava
                                                                                                                                0
                                                                                                                                       0
                                                                                                                                                       6
        Day6Dsa
 → 🔐 Day7Dsa
                                                                                                                                              9 10
                                                                                                                                                               0
→ 😂 Day7n8AssignDsa
                                                                                                                                                0 11 12 13
→ B DSAAssianment

✓ I DSADav11Tasks

                                                                                                                               1
                                                                                                                                      0 0 0
    > M JRE System Library [JavaSE-17]
                                                                                                                               2
                                                                                                                                       3
                                                                                                                                               4
                                                                                                                                                        5
                                                                                                                                                                0
                                                                                                                                                                        0
   v 🕮 src
                                                                                                                                0
                                                                                                                                       0 0 6
0 0 7
                                                                                                                                                               0
                                                                                                                                                                       0

→ # com.dsa.assign11

             > 🛭 KMP.java
                                                                                                                                                                             {1, 1, 1, 1, 0, 0}, {0, 0, 0, 0, 1, 1, 1}, {1, 1, 1, 1, 0, 1}, {1, 1, 1, 1, 1, 0}, {1, 1, 0, 1, 1, 1, 0}, {1, 1, 0, 1, 1, 1}
              > 🗓 KnapsackProblem01.java
              > I KnightsTourAlgo.java
                                                                                                                    49
50
                KnightTourAlgo.java
              > II ICS.iava
                                                                                                                    51
              > 🕖 LinearSearch.java
              > 💹 LongestIncreasingSubsequence.java
              > 

NQueensProblem.java
              DoptimalBinarySearchTree.java

    Problems @ Javadoc   □ Declaration □ Console ×

                                                                                                                   RatInMaze.iava
              > 🗓 SubsetSum.java
         > # com.wipro.dll
         > # com.wipro.ds
                                                                                                                            3
                                                                                                                                  4 5 0
         > 🔠 com.wipro.ll
         > # com.wipro.q
                                                                                                                    0
                                                                                                                           0 8 7 0 0
                                                                                                                                  9 10 0
         > # com.wipro.sort
         > # com.wipro.st
         > # com.wipro.tree
         > 1 module-info.java
                                                                                                                                                 0
        6 0 0
7 0 0
> 💕 Day7Dsa

<sup>™</sup> Day7n8AssignDsa

                                                                                                                                  0
⇒ BSAAssignment⇒ BSADay11Tasks
```

**Task 3: N Queen Problem** Write a function bool SolveNQueen(int[,] board, int col) in C# that places N queens on an N x N chessboard so that no two queens attack each other using backtracking. Place N queens on the board such that no two queens can attack each other. Use a standard 8x8 chessboard.

```
package com.wipro.algo;

public class NQueensProblem {
    public static void main(String[] args) {
        int size = 8;
        boolean[][] board = new boolean[size][size];
```

```
NQueensProblem nQueensProblem = new NQueensProblem();
           if (!nQueensProblem.nQueen(board, size, 0)) {
                 System.out.println("No solution found :( ");
           }
     }
     private boolean nQueen(boolean[][] board, int size, int row) {
           if (row == size) {
                 for (int i = 0; i < size; i++) {</pre>
                       for (int j = 0; j < size; j++) {</pre>
                            System.out.print(board[i][j] ? "0" : "-"+"
");
                       System.out.println();
                 }
                 System.out.println(" ");
                 return true;
           } else {
                 for (int col = 0; col < size; col++) {</pre>
                       if (isValidCell(board, size, row, col)) {
                       board[row][col]=true;
                       if(nQueen(board, size, row+1)) {
                       return true;
                       board[row][col]=false;
                       }
                 }
           }
           return false;
     }
     private boolean isValidCell(boolean[][] board, int size, int row,
int col) {
           // check column
           for (int i = 0; i < row; i++) {</pre>
                 if (board[i][col]) {
                       return false;
                 }
           }
```

```
// check upper left diagonal
for (int i = row, j = col; i >= 0 && j >= 0; i--, j--) {
      if (board[i][j]) {
          return false;
      }
}

// check upper right diagonal
for (int i = row, j = col; i >= 0 && j <size; i--, j++) {
      if (board[i][j]) {
          return false;
      }
}
return true;
}</pre>
```

}

```
🖹 🕃 🍃 🖁 🗖 🛮 🖸 Employeejava 🚨 Salary,java 🚨 customAnnoa... 🚨 KnightTourAl... 🚨 KnightsTourA... 🚨 RatínMazejava 💆 NQueensProbl... 🗴

■ Package Explorer × 🏗 Type Hierarchy
                                                                 private boolean nQueen(boolean[][] board, int size, int row) {
🖟 📂 algos
                                                                      f(row == size) {
   for (int i = 0; i < size; i++) {
     for (int j = 0; j < size; j++) {
        System.out.print(board[i][j] ? "Q" : "-"+" |");
}</pre>
→ 📴 Day18Assignment
                                                         17
, 😂 Dav19Java
                                                         18
→ 👺 Day20Task
√ 📂 Day5DSA
                                                         20
 ⇒ Mark JRE System Library [JavaSE-17]
                                                         21
                                                                               }
System.out.println();
 v 🎏 src
                                                                           System.out.println(" ");
      > An JobSequence.iava
      > 

Knapsack.java
                                                                           return true;
       KnapsackProblem01.java
                                                                      } else {
      > 🗓 KnightsTourAlgo.java
                                                         27
28
                                                                           for (int col = 0; col < size; col++) {</pre>
      > I KnightTourAlgo.java
      > LCS.java
                                                         29
                                                                                if (isValidCell(board, size, row, col)) {
      > 🔑 LinearSearch.java
                                                         30
                                                                                       board[row][col]=true;
                                                        31
                                                                                       if(nOueen(board.size.row+1)) {
      > A LongestIncreasingSubsequence.java
      >  NQueensProblem.java
      > 🗓 OptimalBinarySearchTree.java

    Problems @ Javadoc    Declaration    □ Console ×

      > 🗓 RatInMaze.java
                                                        > D SubsetSum.java
                                                       > # com.wipro.dll
   > # com.wipro.ds
   > # com.wipro.ll
   > 🔠 com.wipro.q
   > 🔠 com.wipro.se
   > # com.wipro.sort
                                                       - Q- - - -
   > # com.wipro.st
                                                       - - Q- - - -
   > I module-info.java
   → 🔂 Day7Dsa
i № Nav7n8ΔesinnDea
```