JGI JAIN DEEMED-TO-BE UNIVERSITY | FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Artificial Intelligence and Data Engineering)

DATA STRUCTURE & ALGORITHM

GROUP MEMBERS:

- JEYAPATHY.M. – 22BTRAD016
- K. SHRISHANTH. -22BTRAD017
- KAPAROTU VENKATA SURYA THARANI. - 22BTRAD018
- KHUSHAL DONGA. - 22BTRAD019
- KRISH DODHIA. -22BTRAD020

QUESTION: -

Develop the chess game to store defence moves of player using suitable data structures?

To store the defence moves of a player in a chess game using data structures, we can use a hash table where each key represents a unique chess board position and its associated value is a list of defence moves. The hash table can be implemented using any hash table data structure such as dictionaries in Python or unordered map in C++.

Each time a player makes a move, the current board position can be hashed and used as a key to retrieve the list of defence moves. If the position is not present in the hash table, a new entry can be created with the position as the key and an empty list as the value. If the opponent makes a threatening move, the current player can add the defence move to the list associated with the current board position.

This way, the defence moves for each unique board position can be easily stored and retrieved, reducing the need for searching through all possible moves to find a suitable defence.

ALGORITHM: -

Here is an algorithm to develop a chess game that stores defence moves of a player using data structures:

Initialise an empty hash table defence moves

While the game is in progress:

1. Hash the current board position and use it as a key to retrieve the list of defence moves from the defence moves hash table
2. If the current position is not present in the hash table, add a new entry to the hash table with the current position as the key and an empty list as the value
3. If the opponent makes a threatening move, add the defence move to the list associated with the current board position in the defence moves hash table
4. The player can use the defence moves stored in the hash table to make a move
5. Repeat steps 1-4 for each player until the game is over

Note: The hash function used to hash the board positions should produce unique hashes for each unique position to avoid collisions in the hash table.

CODE :-

```
class ChessMove {
  int row;
  int column;

  public ChessMove(int row, int column) {
    this.row = row;
    this.column = column;
  }
}

class Player {
  String name;
  ChessMove[] defenceMoves;
  int moveCount;

  public Player(String name) {
    this.name = name;
    this.defenceMoves = new ChessMove[20];
    this.moveCount = 0;
```

```
  }

  public void addDefenceMove(int row, int column) {
    if (moveCount < 20) {
      defenceMoves[moveCount++] = new ChessMove(row,
column);
    }
  }

  public void displayDefenceMoves() {
    System.out.println(name + "'s defence moves:");
    for (int i = 0; i < moveCount; i++) {
      System.out.println((i + 1) + ". (" + defenceMoves[i].row +
", " + defenseMoves[i].column + ")");
    }
  }
}
```

OUTPUT :-

Shell

```
Enter your defence move: 2
2 : 1
Enter your defence move: 3
2 : 1
3 : 1
Enter your defence move: 7
2 : 1
3 : 1
7 : 1
Enter your defence move: 9
2 : 1
3 : 1
7 : 1
9 : 1
Enter your defence move: 6
2 : 1
3 : 1
7 : 1
9 : 1
6 : 1
Enter your defence move: 1
2 : 1
3 : 1
7 : 1
9 : 1
6 : 1
1 : 1
Enter your defence move: 10
2 : 1
3 : 1
7 : 1
9 : 1
6 : 1
1 : 1
10 : 1
Enter your defence move: q
{'2': 1, '3': 1, '7': 1, '9': 1, '6': 1, '1': 1, '10': 1, 'player1': [(0, 0)], 'player2': [(1, 1)]}
>
```