DAA432C
Assignment-02

Presented By:-
Nischay Nagar (IIT2019198)
Abhishek Bithu (IIT2019199)
Raj Chandra (IIT2019200)

CONTENT

- Problem Statement
- Algorithm
- Pseudo Code
- Time Complexity and Space Complexity
- Result

PROBLEM STATEMENT

Given an NxN chessboard and a Knight at position (x,y). The Knight has to take exactly K steps, where at each step it chooses any of the 8 directions uniformly at random. What is the probability that the Knight remains in the chessboard after taking K steps, with the condition that it can’t enter the board again once it leaves it? Solve using Dynamic programming.

ALGORITHM

1. Define direction vectors for the knight.
   Ex:- dx[ ] = {1, 2, 2, 1, -1, -2, -2, -1}
   dy[ ] = {2, 1, -1, -1, -2, -1, 1, 2}
2. Take an array dp[N, N, steps + 1] which will store the probability of reaching (x,y) after (steps) number of moves ,
3. Base case: if the number of steps is 0, then the probability that the Knight will remain inside the board is 1
4. Take the position (x, y) after s number of steps.
5. Take prob = 0.0 then check for each positions reachable from (x, y) using the direction vectors and store it in new position (nx, ny).
6. Check if this new position (nx, ny) is inside of the chessboard, if yes then add dp1[nx][ny][s - 1] / 8.0 to prob.
7. Store the prob in dp[x][y][s].
8. Keep repeating for the given number of steps.
9. The required probability will be stored in dp[start_x][start_y][k], where (start_x, start_y) are the given initial position of the knight and k is the number of steps.

PSEUDO CODE

```
int dx[] = {1, 2, 2, 1, -1, -2, -2, -1};
int dy[] = {2, 1, -1, -2, -2, -1, 1, 2};

bool inside(int x, int y)
{
    return (x >= 0 and x < N and y >= 0
and y < N);
}
```

```
double findProb(int start_x, int start_y, int steps)
{
    double dp[N][N][steps + 1];

    for (int i = 0; i < N; ++i)
        for (int j = 0; j < N; ++j)
            dp[i][j][0] = 1;

    for (int s = 1; s <= steps; ++s)
    {

        for (int x = 0; x < N; ++x)
        {
            for (int y = 0; y < N; ++y)
            {
                double prob = 0.0;

                for (int i = 0; i < 8; ++i)
                {
                    int nx = x + dx[i];
                    int ny = y + dy[i];

                    if (inside(nx, ny))
                        prob+= dp[nx][ny][s - 1] / 8.0;
                }

                dp[x][y][s]= prob;
            }
        }
    }

    return dp[start_x][start_y][steps];
}
```

Time and Space Complexity

Time Complexity: O(NxNxK), where N is the size of the board and K is the number of steps.
As, there are 3 nested loops 2 of them runs for N number of times and one
For K number of times.

Space Complexity: O(NxNxK), where N is the size of the board and K is the number of steps.
As, we are using dp[N][N][K] array of size NxNxK.

Result

```
Enter the size of chessboard 8
Enter the number of steps 3
Enter the space-separated position of knight 0 0
0.125
```