19AIE203 “Data Structures & Algorithms 2”

Project Report

Minimum steps to reach target by a Knight & the Probability of Knight to remain in the chessboard using Dynamic programming

Bachelor of Technology

in

Artificial Intelligence & Engineering

Submitted by:

Maddala H S M Krishna Karthik

AM.EN.U4AIE20046

Subject Teacher

Mr. Sethuraman N Rao

School of Computer Science & Engineering

Kollam, Kerala, India – 690 525.

3rd Semester

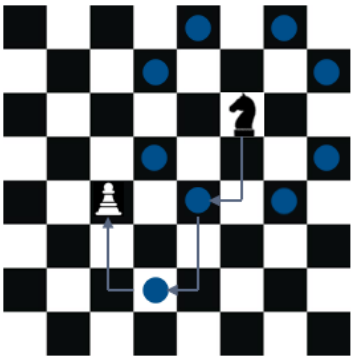
Introduction

Dynamic programming is a way for tackling a complex problem by breaking it down into a series of minor subproblems, solving each one just once, and then storing the results to prevent repeating calculations. It makes use of the notion that the best solution to the overall problem is determined by the best solution to each of its subproblems. The CPU's efficiency can be improved in this way.

Memorization is used by dynamic algorithms to recall the results of previously solved sub-problems. It is primarily an improvement over simple recursion. The temporal complexity of this simple optimization is reduced from exponential to polynomial.

Problem

Given a square chessboard of n x n size, the position of knight (kx, ky) and position of a target (tx, ty). The task is to find out the minimum steps a knight will take to reach the target position. And also, to calculate the probability that the knight remains in the chessboard after taking minimum steps, with the condition that it can't enter the board again once it leaves it. The problem can be solved using dynamic programming.



In a chessboard the knight has eight possible moves as shown in the figure. Each time the knight is to move, it chooses one of eight possible moves uniformly at random and moves there. Each move is two cells in a cardinal direction and then one cell in an orthogonal direction (forming the shape of L).

Solution

There are three cases to be considered for finding the minimum steps taken by a knight to reach the target position.

Case-1: If the position of the knight and the target cell are along different rows and columns. (Kx! =tx and Ky! =ty)

Example:

Let knight and target positions be (3,3) & (6,6)

There are 2 steps to move towards the target.

which are (3,3) to (4,5) and (3,3) to (5,4)

So, the minimum steps from (3,3) to (6,6) is equal to 1 + minimum of [minimum steps from (4,5) to (6,6), minimum steps from (5,4) to (6,6)].

Case-2: If the position of the knight and the target cell are along same rows or columns. (kx==tx or ky==ty)

Example:

Let knight and target positions be (2,4) & (7,4)

There are 4 steps to move towards the target.

which are (2,4) to (4,5) and (2,4) to (4,3) (both these steps are equivalent),

(2,4) -> (3,6) and (2,4) -> (3,2) (both these steps are equivalent)

So, the minimum steps from (2,4) to (7,4) is equal to 1 + minimum of [minimum steps from (2,4) to (4,5), minimum steps from (2,4) to (3,6)].
Case-3: If either the knight or the target is at the corner and [abs(kx-tx), abs(ky-ty)]=[1,1], then the minimum steps to reach the target will be 4.
The base cases for this problem are as follow.
when the knight & target are at adjacent squares (along the same row or column), minimum number of moves required to reach the target is 3.
when the knight & target are at adjacent squares but lie diagonally to each other, minimum number of moves required to reach the target is 2.
If the knight & target are at same position, minimum number of moves required is 0. And if number of moves is 0 then the probability that the knight will remain inside the board is 1.
Algorithm:

1. Firstly, add all the base cases which satisfies case-3.
2. Then, initialize a dp array to store the sub paths and add all the base cases as shown above. dp[1][0] =dp[0][1]=3 , dp[1][1]=dp[2][0]=dp[0][2]=2 , dp[2][1]=dp[1][2]=1
3. If the target is reached by the knight, then return 0.
4. If the cell is revisited then return the stored value in the dp array.
5. Else, select the coordinates which satisfy case-1 and case- 2.
6. Store 1 plus a minimum of paths obtained by two coordinates in the dp array and recur till the target is not reached.
7. Here, exchanging the coordinates x with y of both knight and target will result in same answer. So, update the dp array accordingly.
8. Finally, return dp[abs(kx-tx)][abs(ky-ty)] as the minimum steps a knight will take to reach the target position.

After finding the minimum steps to reach the target by knight, the probability that the knight remains in the chessboard is calculated. Top-down variant of dynamic programming is used for calculating the probability of the knight remains in the chessboard after taking minimum steps. This approach uses memorization.
Let us assume that the knight has to take k steps and after taking the kth step the knight reaches (x, y). There are 8 different positions from where the Knight can reach to (x, y) in one step, and they are (x+1, y+2), (x+2, y+1), (x+2, y-1), (x+1, y-2), (x-1, y-2), (x-2, y-1), (x-2, y+1), (x-1, y+2). If we calculate the probabilities of reaching these 8 positions after k-1 steps then the final probability after k steps will simply be equal to the (sum of probability of reaching each of these 8 positions after k-1 steps)/8. For the positions that lie outside the board, we will either take their probabilities as 0 or simply neglect it.
In this problem we have to keep track of the probabilities at each position for every number of steps. So, the probability of reaching (x, y) after number of moves (steps) is stored.
Algorithm:

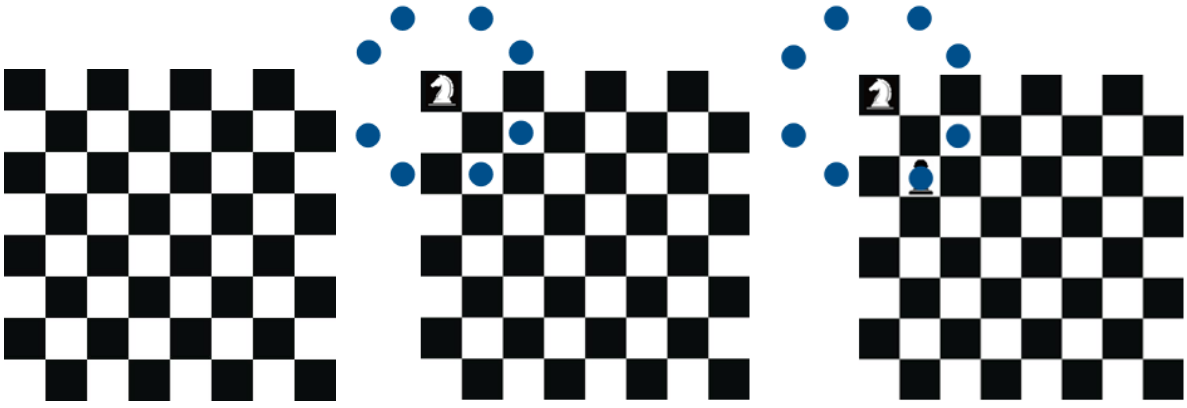
1. Initially, create a dp dictionary to store all probabilities.
2. If the knight is not on the board, then return 0.
3. If number of steps is 0 then return 1.0.
4. If the cell is revisited with same steps, then return stored value in the dp dictionary.
5. Recursively calculate the probabilities of the knight remain on board after making random moves steps-1 number of times. And store the probabilities in dp dictionary. P(Knight on board after s steps) = ΣP(Knight on board after s-1 steps)*1/8
6. Finally, return the on-board probability of the knight.

Sample input and output

Input format: The first line specifies the dimension of the N x N chessboard as an integer. The next two lines specifies the coordinates of knight and target in the chessboard.
Output format: The first line specifies the minimum steps required for a knight to reach the target position. And the second line specifies the probability that the knight remains in the chessboard after taking the minimum steps.
Sample Input:
8
0 0
2 1
Here, 8 implies dimension of the chessboard (8x8).
0 0 implies position of the knight in the chessboard (0,0).
2 1 implies position of the target in the chessboard (2,1).
Sample Output:
1
0.25
Here, 1 implies the minimum steps required for a knight to reach the target in the chessboard.
0.25 implies knight on board probability

```
>>>
=============== RESTART: C:\Users\Karthik Maddala\Desktop\dsa.py ===============
8
0 0
2 1
1
0.25
>>>
```

Pictorial Representation



Let the dimensions of chessboard be 8 x 8 as shown in fig-I. Knight's position be (0, 0) as shown in fig-II. Target position be (2, 1) as shown in fig-III.
We know that for each move the knight move two cells in a cardinal direction and then one cell in an orthogonal direction. When the knight starts from position (0,0), it takes only one step to reach the target position (2,1). So, the minimum steps a knight will take to reach the target position is 1.
At each step, the knight has 8 different positions to choose from. when knight starts from (1,1), the only possible positions for knight to remain inside the chessboard are (3, 2) and (2, 3). After

taking one step it will lie inside the board only at 2 out of 8 positions. Therefore, the probability is 2/8 = 0.25. So, the probability that the knight remains in the chessboard after taking minimum steps is 0.25.

Inference

The problem is solved with optimization using dynamic programming. Minimum steps a knight take to reach the target position is calculated. And its time and space complexity is O(abs(kx-tx)*abs(ky-ty)).Where, (kx, ky) are position coordinates of knight and (tx, ty) are position coordinates of target. Also, Probability of the knight remains in the chessboard after taking minimum steps is calculated. And its time and space complexity is O(NxNxK). Where, N is the dimension of the chess board (NxN) and K is number of steps.

References

Reference-1
Reference-2
Reference-3
Reference-4
Reference-5
Reference-6

---THE END---