Backtracking | Set 1 (The Knight's tour problem)

Backtracking is an algorithmic paradigm that tries different solutions until finds a solution that "works". Problems which are typically solved using backtracking technique have following property in common. These problems can only be solved by trying every possible configuration and each configuration is tried only once. A Naive solution for these problems is to try all configurations and output a configuration that follows given problem constraints. Backtracking works in incremental way and is an optimization over the Naive solution where all possible configurations are generated and tried.

For example, consider the following Knight's Tour problem.
The knight is placed on the first block of an empty board and, moving according to the rules of chess, must visit each square exactly once.

Let us first discuss the Naive algorithm for this problem and then the Backtracking algorithm.

Naive Algorithm for Knight's tour

The Naive Algorithm is to generate all tours one by one and check if the generated tour satisfies the constraints.

```
while there are untried tours
{
   generate the next tour
   if this tour covers all squares
   {
      print this path;
   }
}
```

Backtracking works in an incremental way to attack problems. Typically, we start from an empty solution vector and one by one add items (Meaning of item varies from problem to problem. In context of Knight's tour problem, an item is a Knight's move). When we add an item, we check if adding the current item violates the problem constraint, if it does then we remove the item and try other alternatives. If none of the alternatives work out then we go to previous stage and remove the item added in the previous stage. If we reach the initial stage back then we say that no solution exists. If adding an item doesn't violate constraints then we recursively add items one by one. If the solution vector becomes complete then we print the solution.

Backtracking Algorithm for Knight's tour

Following is the Backtracking algorithm for Knight's tour problem.

```
If all squares are visited
    print the solution
Else
   a) Add one of the next moves to solution vector and recursively
   check if this move leads to a solution. (A Knight can make maximum
   eight moves. We choose one of the 8 moves in this step).
   b) If the move chosen in the above step doesn't lead to a solution
   then remove this move from the solution vector and try other
   alternative moves.
   c) If none of the alternatives work then return false (Returning false
   will remove the previously added item in recursion and if false is
   returned by the initial call of recursion then "no solution exists" )
```

Following is C implementation for Knight's tour problem. It prints one of the possible solutions in 2D matrix form. Basically, the output is a 2D 8*8 matrix with numbers from 0 to 63 and these numbers show steps made by Knight.

```
#include<stdio.h>
#define N 8

int solveKTUtil(int x, int y, int movei, int sol[N][N], int xMove[],
                int yMove[]);

/* A utility function to check if i,j are valid indexes for N*N chessboard */
int isSafe(int x, int y, int sol[N][N])
{
    if ( x >= 0 && x < N && y >= 0 && y < N && sol[x][y] == -1)
        return 1;
    return 0;
}

/* A utility function to print solution matrix sol[N][N] */
```

```
void printSolution(int sol[N][N])
{
    for (int x = 0; x < N; x++)
    {
        for (int y = 0; y < N; y++)
            printf(" %2d ", sol[x][y]);
        printf("\n");
    }
}

/* This function solves the Knight Tour problem using Backtracking.  This
function mainly uses solveKTUtil() to solve the problem. It returns false if
no complete tour is possible, otherwise return true and prints the tour.
Please note that there may be more than one solutions, this function
prints one of the feasible solutions.  */
bool solveKT()
{
    int sol[N][N];

    /* Initialization of solution matrix */
    for (int x = 0; x < N; x++)
        for (int y = 0; y < N; y++)
            sol[x][y] = -1;

    /* xMove[] and yMove[] define next move of Knight.
       xMove[] is for next value of x coordinate
       yMove[] is for next value of y coordinate */
    int xMove[8] = {  2, 1, -1, -2, -2, -1,  1,  2 };
    int yMove[8] = {  1, 2,  2,  1, -1, -2, -2, -1 };

    // Since the Knight is initially at the first block
    sol[0][0]  = 0;

    /* Start from 0,0 and explore all tours using solveKTUtil() */
    if(solveKTUtil(0, 0, 1, sol, xMove, yMove) == false)
    {
        printf("Solution does not exist");
        return false;
    }
    else
        printSolution(sol);

    return true;
}

/* A recursive utility function to solve Knight Tour problem */
int solveKTUtil(int x, int y, int movei, int sol[N][N], int xMove[N],
                int yMove[N])
{
   int k, next_x, next_y;
   if (movei == N*N)
       return true;

   /* Try all next moves from the current coordinate x, y */
   for (k = 0; k < 8; k++)
   {
       next_x = x + xMove[k];
       next_y = y + yMove[k];
       if (isSafe(next_x, next_y, sol))
       {
         sol[next_x][next_y] = movei;
```

```
            if (solveKTUtil(next_x, next_y, movei+1, sol, xMove, yMove) == true)
                return true;
            else
                sol[next_x][next_y] = -1;// backtracking
        }
    }

    return false;
}

/* Driver program to test above functions */
int main()
{
    solveKT();
    getchar();
    return 0;
}
```

Output:

```
 0  59  38  33  30  17   8  63
37  34  31  60   9  62  29  16
58   1  36  39  32  27  18   7
35  48  41  26  61  10  15  28
42  57   2  49  40  23   6  19
47  50  45  54  25  20  11  14
56  43  52   3  22  13  24   5
51  46  55  44  53   4  21  12
```

Note that Backtracking is not the best solution for the Knight's tour problem. See this for other better solutions. The purpose of this post is to explain Backtracking with an example.

References:
http://see.stanford.edu/materials/icspacs106b/H19-RecBacktrackExamples.pdf
http://www.cis.upenn.edu/~matuszek/cit594-2009/Lectures/35-backtracking.ppt
http://mathworld.wolfram.com/KnightsTour.html
http://en.wikipedia.org/wiki/Knight%27s_tour

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Related Topics:

- Linearity of Expectation
- Iterative Tower of Hanoi
- Count possible ways to construct buildings
- Build Lowest Number by Removing n digits from a given number
- Set Cover Problem | Set 1 (Greedy Approximate Algorithm)
- Find number of days between two given dates
- How to print maximum number of A's using given four keys
- Write an iterative O(Log y) function for pow(x, y)

Tags: Backtracking

Writing code in comment? Please use ideone.com and share the link here.

Vikram Ojha • a month ago

man wat r dese magic numbers
int xMove[8] = { 2, 1, -1, -2, -2, -1, 1, 2 };
int yMove[8] = { 1, 2, 2, 1, -1, -2, -2, -1 };

Renée ➚ Vikram Ojha • 18 days ago

Those are the 8 moves a Knight can make.
Consider negative xMove: move up; positive xMove: move down.
Also, consider negative yMove: move to the left; positive yMove: move to the right.
A chess board has numbers and letters on its sides, but a matrix has only numbers, so you calculate the coordinates to move up and down and sideways.
2 down, 1 right
1 down, 2 right
1 up, 2 right
...
Did you get it?

Vikram Ojha ➚ Renée • 16 days ago

Ya Reene on very dat day I had seen ur previous comment and dat was realy helpful...bt still thanx for replying

Tejwinder • 3 months ago

Do you know it makes a "Tessellation" while traversal

http://www.borderschess.org/KT...

Avishek Sachan • 5 months ago

// NON RECURSIVE APPROACH and Modified solveKTUtil function

#include <stack>

```
#include <stdio.h>
using namespace std;

#define N 8

typedef struct Point

{

int x;

int y;

} SChessXY;

int solveKTUtil(int x, int y, int movei, int sol[N][N], int xMove[],
```

see more

• Reply • Share ›

Coding Enthusiast Avishek Sachan • 3 months ago

The non-recursive version does not work. It gives wrong solution where 61 and 62 are far apart. I think the stack needs more information to be pushed in.

• Reply • Share ›

Aveek Biswas • 7 months ago

The program is failing if instead of
[int xMove[8] = { 2, 1, -1, -2, -2, -1, 1, 2 };
int yMove[8] = { 1, 2, 2, 1, -1, -2, -2, -1 };],

[int xMove[8] = { 1, 2, 1, 2, -2, -1, -1, -2 };
int yMove[8] = { 2, 1, -2, -1, 1, 2, -2, -1 };] is used.
The corresponding moves in x and y direction is maintained but the order is different int the 2nd matrix. But it fails. This means the solution also depends on these two matrices also. Could anybody plz explain the reason for this?

1 • Reply • Share ›

Avishek Sachan Aveek Biswas • 5 months ago

I pasted the non recursive solution, it works even you change the sequence of directions

• Reply • Share ›

dustbite Aveek Biswas • 6 months ago

Backtracking is trying every sequence of moves to find an answer. If we find an answer, we don't try any other sequence of moves.

So, initially from the starting point (0, 0), we have to make a choice which is which place should the knight be landed on such that it will derive me a knight tour. We can land on 2 possible locations (1, 2) and (2, 1). Assume that (1, 2) doesn't give us a knight tour but (2, 1) gives. So if you try (1, 2) first, you would reach the solution later than you would if you had chosen (2, 1). It's because from (1, 2) there would be more choices to explore, but any of them would not lead us to the solution(first all possible paths from (1, 2) need to be explored before we back track and try (2, 1) which is obviously too slow.). So, if you could prune the search path earlier, then could find solution faster. There are some heuristics that would help us derive solution for the knight tour efficiently(and for any N x N board if knight tour is possible).

1 ▲ | ▼ • Reply • Share ›

ankit Aggarwal • 10 months ago

what will be the time complexity of this algorithm?

1 ▲ | ▼ • Reply • Share ›

Mayank ➦ ankit Aggarwal • 6 months ago

http://stackoverflow.com/quest...

▲ | ▼ • Reply • Share ›

Deepesh ➦ ankit Aggarwal • 6 months ago

@Admin Plz reply

▲ | ▼ • Reply • Share ›

Stéphane • a year ago

Has anyone tried to start from others positions than [0][0] ?

▲ | ▼ • Reply • Share ›

Guest ➦ Stéphane • 10 months ago

Yes, it is taking very long for other positions

▲ | ▼ • Reply • Share ›

arjomanD • a year ago

Hello Every1 . I had a question . i have my code BUT it works very very Slow for N=8 (in fact i havn't still got the answer !

Would you Plz help me make my code and algorithm better ?

http://paste.ubuntu.com/742019...

Thanks !

▲ | ▼ • Reply • Share ›

KC • a year ago

Knight moves 2 squares in the direction of one of the axes of the board, and then 1 square in an

Knight moves 2 squares in the direction of one of the axes of the board, and then 1 square in an orthogonal direction. This definition can then be generalized to obtain an (a, b) knight. If the numbers a+b and a-b are coprime, tour is also possible in non 8x8 board using this similar algorithm.

  • Reply • Share ›

Probe • 2 years ago

Here is also my implementation of the above algorithm for java:

[code]public class HorseTraversing{

public static void printBoard(int[][] board){
for (int i=0; i<board.length; i++){="" for="" (int="" y="0;" y<board.length;="" y++){="" if="" (board[i][y]="">= 9){
System.out.print(board[i][y] + " ");
}else {
System.out.print(" " + board[i][y] + " ");
}
}
System.out.println();
}
}

public static boolean isSafeMove(int moveX, int moveY, int[][] board){
int maxPosition = (board.length - 1);

---

see more

  • Reply • Share ›

dark_night • 2 years ago

I am still not getting the difference between backtracking and recursion.. especially for this problem

Here also we are using every possible move and in recursion too we do the same , so how does backtracking is more efficient than recursion.

2 • Reply • Share ›

theCuriosityEnthusiast ➚ dark_night • 10 months ago

You are using recursion here as well only thing is if we find a path which doesn't lead to solution, we immediately return back (backtracking) to the nearest node from which other unexplored paths are available. In recursion we only stop at the base-cases, hence lots of useless computations are done..

  • Reply • Share ›

Let's Make a New India • 2 years ago

Muhammad Barrima thanks s a lot 4 this kind of help 4 geek like me

Reply • Share ›

Let's Make a New India • 2 years ago

Muhammad Barrima thanks s a lot 4 this kind of help 4 geek like me

Reply • Share ›

shashank • 2 years ago

How to decide xMove[8] and yMove[8] arrays contents?

Reply • Share ›

Dev Khanna ➦ shashank • 2 years ago

They are the contents of the possible moves from any given square. For instance, one possible move is to go left one and up two. The corresponding "vector" would be (1, 2)

Reply • Share ›

Mahendra Mundru • 2 years ago

It seems to be O(N power (N**2)).

Reply • Share ›

coder • 2 years ago

can anyone tell me why this code works so slow

```
 #include<iostream>
#define max 8
using namespace std;
void printsol(int arr[][max])
{
    int i,j;
    for(i=0;i<=max-1;i++)
    {
        for(j=0;j<=max-1;j++)
        cout<<arr[i][j]<<' ';
        cout<<'\n';
    }
    return ;
}
bool issafe(int arr[][max],int i,int j)
{
```

see more

Reply • Share ›

Guest ➦ coder • a year ago

There is a reason why geekforgeeks folks chose
xMove[8] = { 2, 1, -1, -2, -2, -1, 1, 2 }
yMove[8] = { 1, 2, 2, 1, -1, -2, -2, -1 }

Only for this configuration of steps by knight, the recursion runs faster because it backtracks quite quickly in later stages.

For any other configuration like
xMove[8] = { 2, 2, 1, 1, -2, -2, -1, -1 }
yMove[8] = { 1, -1, 2, -2, 1, -1, 2, -2 }

it's much much slower.

But the 2nd configuration is correct too. It's just that it takes more time

1 • Reply • Share ›

theCuriosityEnthusiast Guest • 10 months ago

I agree with Guest, the 1st configuration is clearly circular and hence faster, the 2nd is quite abrupt and so should take more time. By circular I mean if you plot the 1st vector, the 2nd vector will be in clockwise or anticlockwise direction and so will be the 3rd, 4th, 5th so on..

• Reply • Share ›

zaraki • 2 years ago

Hi,
Maybe I am missing something. I did code this up to get the exact same output.
I believe 0 represents the leftmost square and 63 the rightmost.

The order that is printed does not seem to be how a knight would move:
row-wise: 0(0,0) 59(7,3) 38(4,6) 33 30 17 8 63

if I take it columnwise still the coordinates seem too far. Please let me know if this is unordered or there is some other order of the tour that I am missing.
Thanks!

• Reply • Share ›

Priyanka zaraki • 2 years ago

Hi,

the values being printed here in the solution tells the order of Knight moves.
Square with value 0 means knight will start from this square.
Square with value 1 means knight will move to this square next.
Square with value 2 means knight will move to this square next and so on..

Since we are moving 64 times (0 to 63) it signifies the solution covers all the square,

nence the correct solution.
Hope this helps.

• Reply • Share ›

yashraj • 2 years ago

```
#include
#define N 8
typedef enum{false,true} bool;

int solveKTutil(int sol[N][N],int nx[N],int ny[N],int x,int y,int nmove);

int is_safe(int sol[N][N],int x,int y)
{
if(x>=0 && x=0 && y<N && sol[x][y]==-1)
return 1;
return 0;

}
void print_sol(int sol[N][N])
{
int i,j;
for(i=0;i<N;i++)
{
for(j=0;j<N;j++)
```

see more

• Reply • Share ›

Ajeet Singh Yadav • 2 years ago

thanks a lot :)

• Reply • Share ›

Muhammad Barrima • 2 years ago

the knight on a chess board moves in this way
2 moves up and then 1 right
2 moves up and then 1 left
2 moves down and then 1 right
2 moves down and then 1 right
2 moves right and then 1 up
2 moves right and then 1 down
2 moves left and then 1 up
2 moves left and then 1 down

Then he made those arrays to check for all possible next moves :)

• Reply • Share ›

Vikram Ojha ➚ Muhammad Barrima • a month ago

excellent man thanx!! u helped me lot

• Reply • Share ›

Ajeet Singh Yadav • 2 years ago

can somebody please explain the.

int xMove[8] = { 2, 1, -1, -2, -2, -1, 1, 2 };.

int yMove[8] = { 1, 2, 2, 1, -1, -2, -2, -1 };.

please do explain how to write these?

• Reply • Share ›

Abhinav Priyadarshi • 3 years ago

one very interesting thing to notice here is that changing the order of moves increases the execution time by a huge amount.

here (http://ideone.com/gzVrt) is the orignal code executing in 0.48 seconds.
while this(http://ideone.com/qfsf3) one is taking huge amount of time(time limit exceeded), in which only the order of moves has been changed.

• Reply • Share ›

nitin gupta • 3 years ago

```
 /* can some body tell me why?
int xMove[8] = {  2, 1, -1, -2, -2, -1,  1,  2 };
    int yMove[8] = {  1, 2,  2,  1, -1, -2, -2, -1 };
are like that ...wt it means?*/
```

1 • Reply • Share ›

nitin gupta • 3 years ago

can some body tell me ...XMove and YMove array ke content aise hi kyo hai ?

```
/* Paste your code here (You may delete these lines if not writing code) */
```

• Reply • Share ›

anant • 3 years ago

@geeksforgeeks:
please tell(or if possible explain), the time complexity for knight tour using backtracking

1 • Reply • Share ›

kartik ➚ anant • 3 years ago

@anant: Getting a tight upper bound for such problems is never easy. We can get a loose upper bound though.

The recursion tree will be a tree of depth 64. Every internal node will have at most 8 children. So we can say the upper bound on time complexity is 1 + 8 + 8*8 + 8*8*8 + ... (64 times).

More expert comments from other users are welcome!

Reply • Share ›

Dedicated Programmer • 3 years ago

I have used this code, but unable to figure out why it is printing same solution multiple times..pls help me out

Reply • Share ›

Dedicated Programmer • 3 years ago

```
 /* Paste your code here (You may delete these lines if not writing code) */#include <stdio
#define N 5

int isValidPosition(int (*Table)[N],int row,int col)
{
        if(row>=0 && row<N && col>=0 && col<N && !Table[row][col])
                return 1;
        return 0;
}

void printSolution(int (*Table)[N],int top)
{
        int i,j;

        for(i=0;i<N;i++)
        {
                for(j=0;j<N;j++)
                        printf("%d ",Table[i][j]);
```

see more

Reply • Share ›

Ajinkya • 3 years ago

What is the time complexity of this??
and also please comment on the time complexity of a backtracking solution in general... I know it is dependent on the problem, but if possible discuss complexities of 8 queens problem, rat maze problem...
Thanks a ton...
Geeksforgeeks is savior and a great educator! :)

```
/* Paste your code here (You may delete these lines if not writing code) */
```


vivek • 4 years ago

Additional information about topic:

Knight's tour is a variation of Hamiltonian path problem in graph theory.
Knight's tour is a NP complete problem.

anantha89 • 4 years ago

Hi All,

When I tried with,
int a[8] = {-2, +2, +1, +1, -2, +2, -1, -1};
int b[8] = {+1, +1, -2, +2, -1, -1, -2, +2};

It did not solve,

when I used
int a[8] = {2, 1, -1, -2, -2, -1, 1, 2};
int b[8] = {1, 2, 2, 1, -1, -2, -2, -1};

It solved in less time.

How?

yashraj ➚ anantha89 • 2 years ago

i think the number of waste moves with the second one are far less compared to the first one.. that is number of backtracks are less in second case.. so it is running fast towards the solution.

```
/* Paste your code here (You may delete these lines if not writing code) */
```


theCuriosityEnthusiast ➚ yashraj • 10 months ago

Because you're points aren't in a circular fashion.. They are abrupt, so you're checking the incorrect points before the correct ones... In such algorithms which have high time complexities these things can make a big difference.. The first sequence will also solve it cause we are going through all possible options and the solution does exist...

alan • 4 years ago

@geeksforgeeks
In your back-traking code, how are you making sure that 'knight ends on a square attacking the square from which it began'.

1 • Reply • Share ›

Sandeep alan • 4 years ago

@alan: There was an error in post. The code and algorithm actually print a Knight's tour. The printed tour can be open or close. I have updated the post.

• Reply • Share ›

asd • 4 years ago

Is this the best possible solution. ? Is there any better solution ?

• Reply • Share ›

Sandeep asd • 4 years ago

@asd: The above solution is not the best solution for Knight's tour problem. See this for other better solutions. The purpose of this post is to explain Backtracking with an example.

I have added same note to the original post.

• Reply • Share ›

Load more comments

- Interview Experiences
- Advanced Data Structures
- Dynamic Programming
- Greedy Algorithms
- Backtracking
- Pattern Searching
- Divide & Conquer
- Mathematical Algorithms
- Recursion
- Geometric Algorithms

Popular Posts

- All permutations of a given string
- Memory Layout of C Programs
- Understanding “extern” keyword in C
- Median of two sorted arrays
- Tree traversal without recursion and without stack!
- Structure Member Alignment, Padding and Data Packing
- Intersection point of two Linked Lists
- Lowest Common Ancestor in a BST.
- Check if a binary tree is BST or not
- Sorted Linked List to Balanced BST

Recent Comments

- Ashish Aggarwal

  Try Data Structures and Algorithms Made Easy -...

  Algorithms · 17 minutes ago

- Vlad

Thanks. Very interesting lectures.

Expected Number of Trials until Success · 1 hour ago

- cfh

  My implementation which prints the index of the...

  Longest Even Length Substring such that Sum of First and Second Half is same · 1 hour ago

- Gaurav pruthi

  forgot to see that part ;)

  Bloomberg Interview | Set 1 (Phone Interview) · 1 hour ago

- saeid aslami

  thanks

  Greedy Algorithms | Set 7 (Dijkstra's shortest path algorithm) · 1 hour ago

- Cracker

  Implementation:...

  Implement Stack using Queues · 2 hours ago