

# Re: Proof of knight's tour in chess

---

*Source:* <http://sci.tech-archive.net/Archive/sci.math/2007-04/msg01503.html>

---

- *From:* "user923005" <[dcorbit@xxxxxxxx](mailto:dcorbit@xxxxxxxx)>
  - *Date:* 11 Apr 2007 12:30:14 -0700
- 

On Apr 11, 11:34 am, peace\_\_and\_\_prosper...@xxxxxxxx wrote:

On Apr 9, 5:57 pm, "user923005" <[dcor...@xxxxxxxx](mailto:dcor...@xxxxxxxx)> wrote:

On Apr 9, 8:34 am, peace\_\_and\_\_prosper...@xxxxxxxx wrote:

Any suggested resource which includes a proof of  
the knight's tour  
problem?

Proof of what?

To prove that it is possible, you just have to generate one, which is  
trivial with a computer program.

/\*Knight's Tour Problem – a kind of undirected Hamiltonian circuits  
Using an advanced algorithm – at least 10000 times faster than  
simple backtracking  
The key idea is to detect bi-degree nodes and also the art of  
programming  
Enjoy it!

Author : Fei Lu – student of Shanghai Jiao Tong University  
Email : [flyl...@xxxxxxxx](mailto:flyl...@xxxxxxxx)  
Homepage & Research Center : [flyland.bentium.net](http://flyland.bentium.net)  
\*/

Re: Proof of knight's tour in chess

```
#define N 8

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <limits.h>

clock_t start,
finish;
double duration;
int board[N * N];
int (*board2D)[N] = (int (*)[N]) &board;

int degree[N * N];
int (*degree2D)[N] = (int (*)[N]) &degree;

int branch[N * N][8],
Xbranch[N * N][8];
int (*branch2D)[N][8] = (int (*)[N][8]) &branch;

int nBranch[N * N],
XnBranch[N * N];
int (*nBranch2D)[N] = (int (*)[N]) &nBranch;

int blist[N * N][9][3];

int iDir[8] =
{-2, -1, 1, 2, 2, 1, -1, -2};
int jDir[8] =
{1, 2, 2, 1, -1, -2, -2, -1};
```

## Re: Proof of knight's tour in chess

```
int path[N * N + 1];
int dir[N * N + 1];
int n;

int cnt = 0;

unsigned int nSolution_High = 0,
nSolution_Low = 0;

void printBoard()
{
    int i,
    j;
    char buf[80];
    for (i = 0; i < 4 * N + 1; i++)
        buf[i] = '-';
    buf[i] = 0;
    printf("%s\n", buf);
    for (i = 0; i < N; i++) {
        for (j = 0; j < N; j++) {
            if (board2D[i][j] == 0)
                printf("|%3d", N * N - 1);
            else
                printf("|%3d", abs(board2D[i][j]));
        };
        printf("\n");
    };
    printf("%s\n", buf);
}

void record()
{
    nSolution_Low++;
    if (nSolution_Low == 0)
        nSolution_High++;

    if (nSolution_Low % 100000 == 0) {
        finish = clock();
    }
}
```

## Re: Proof of knight's tour in chess

```
duration = (double) (finish - start) / CLOCKS_PER_SEC;
printf("Solved %.2le (%u) %8.1f seconds\n",
((double) nSolution_High) * UINT_MAX + nSolution_Low,
nSolution_Low,
duration);
printBoard();
};}
```

```
int re_calc_degree(int t)
{
int d,
u,
m,
count = 0;
int degree2 = 0;
int u2;
m = nBranch[t];
for (d = 0; d < m; d++) {
u = branch[t][d];
if (board[u] > 0)
continue;
if (degree[u] == 2) {
if (degree2 > 0 || u == 2 * N + 1) {
for (d = 0; d < count; d++)
degree[Xbranch[t][d]]++;
return -1;
};
degree2++;
u2 = u;
};
degree[u]--;
Xbranch[t][count++] = u;
};
if (degree2 == 0) {
XnBranch[t] = count;
} else {
Xbranch[t][0] = u2;
XnBranch[t] = 1;
};
return 0;}
}
```

```
void restore_degree(int t)
{
int d,
m,
u;
m = nBranch[t];
```

## Re: Proof of knight's tour in chess

```
for (d = 0; d < m; d++) {
u = branch[t][d];
if (board[u] > 0)
continue;
degree[u]++;
};}

/*
// Precondition:Knightnow stays at the n(th) grid at (i,j)
// where i*N+j=path[n] and the travel goes on ...
// before travel further backup his direction in dir[n]
*/
void Travel()
{
int m,
d;
int here,
there;

while (n > 1) {
LOOP:
here = path[n];
m = XnBranch[here];
for (d = ++dir[n]; d < m; d++) {
there = Xbranch[here][d];
if (there == 2 * N + 1)
continue;
board[there] = n + 1;
if (re_calc_degree(there) == -1) {
board[there] = 0;
continue;
};
dir[n++] = d;
path[n] = there;
dir[n] = -1;

if (n < N * N - 2)
goto LOOP;
else {
record();
restore_degree(there);
board[there] = 0;
n--;
break;
};
};
```

## Re: Proof of knight's tour in chess

```
};  
restore_degree(here);  
board[here] = 0;  
n--;  
};}
```

```
void init()  
{  
int i,  
j,  
ip,  
jp,  
k;  
int count;  
memset(board, 0, sizeof(board));  
memset(dir, -1, sizeof(dir));  
for (i = 0; i < N; i++) {  
for (j = 0; j < N; j++) {  
count = 0;  
for (k = 0; k < 8; k++) {  
ip = i + iDir[k];  
jp = j + jDir[k];  
if (ip < 0 || jp < 0 || ip >= N || jp >= N)  
continue;  
branch[i * N + j][count] = ip * N + jp;  
count++;  
};  
degree[i * N + j] = nBranch[i * N + j] = count;  
};  
};
```

```
degree2D[2][1]++;  
board2D[2][1] = -N * N;
```

```
path[1] = 0;  
board2D[0][0] = 1;  
re_calc_degree(0);
```

```
path[2] = 1 * N + 2;  
board2D[1][2] = 2;  
re_calc_degree(1 * N + 2);  
n = 2;}
```

Re: Proof of knight's tour in chess

```
int main(void)
{
start = clock();
init();
Travel();
return 0;
```

}– Hide quoted text –

– Show quoted text –

Thanks for the program, but I also am interested in an existence proof that such a tour exists.

Maybe some of these links can help you:

[http://en.wikipedia.org/wiki/Knight's\\_tour](http://en.wikipedia.org/wiki/Knight's_tour)

<http://mathworld.wolfram.com/KnightsTour.html>

<http://www.mathpuzzle.com/leapers.htm>

<http://web.cs.ualberta.ca/~joe/Theses/vandegriend.html>

.