Knight's Tour in Cg

Suraj N. Kurapati

CMPE-220 final project

20 March 2007

Agenda

Knight's Tour

- Ancient chess puzzle
- Visit every square **once**
- $\mathcal{NP}$ complete

- Open tour
  - Hamiltonian path
- Closed tour
  - Hamiltonian cycle

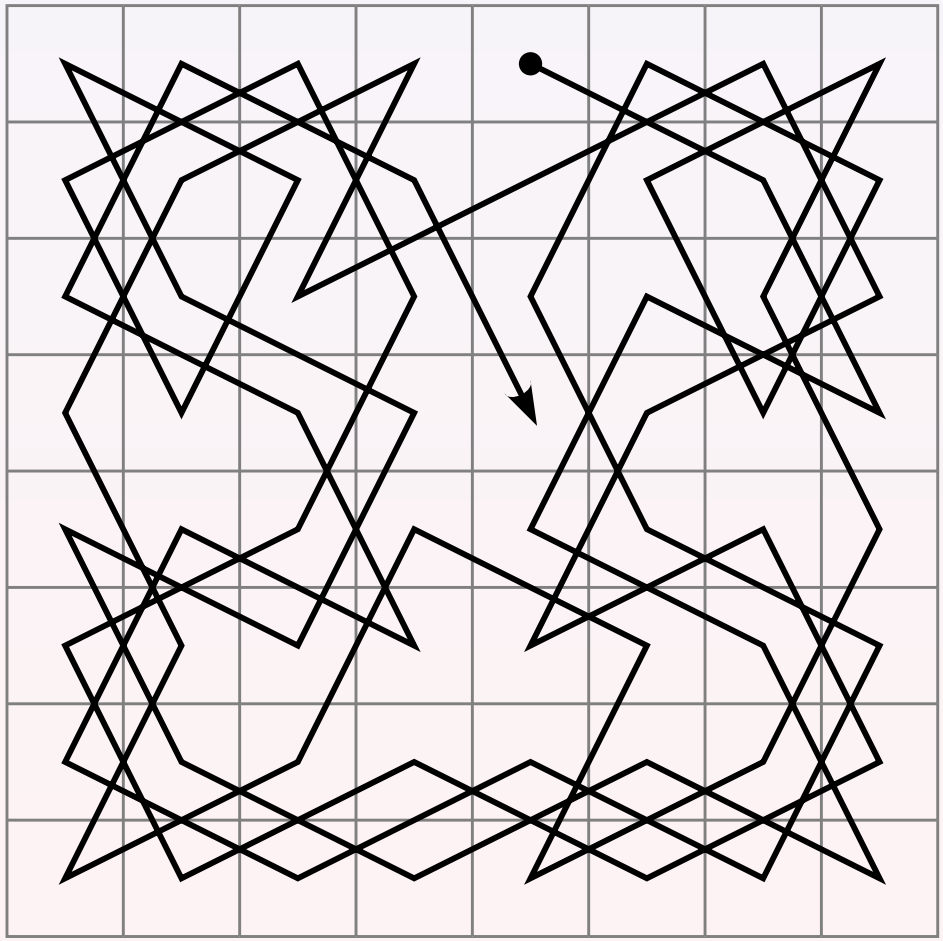


Figure: Courtesy of Ilmari Karonen [5]

Naïve approach

- Depth-first search
- Backtrack from dead ends
- $O(8^n)$ time complexity

Variations

- Visit neighbors **out of order**
- Try random permutations
- Look up precomputed solutions

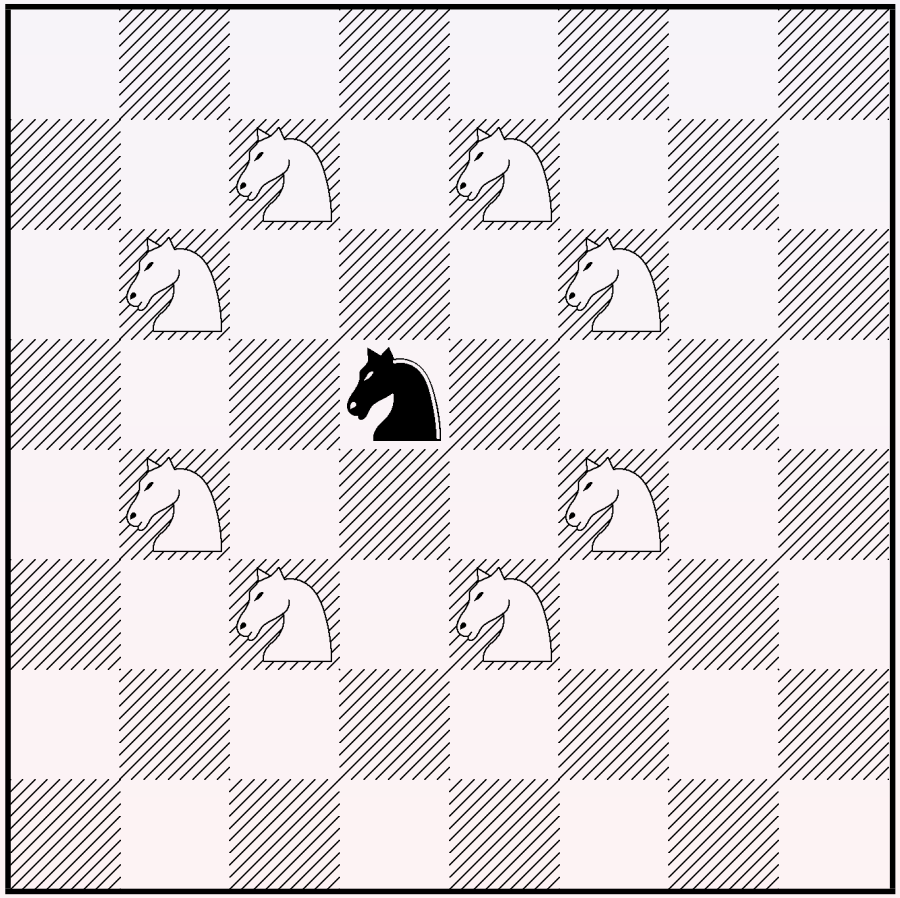


Figure: Possible moves for a knight.

Warnsdorff's heuristic

- Visit **lowest** ranking neighbor
  - Ties do not matter
- Eliminates backtracking
- Works most of the time

Definition

Rank — number of unvisited neighbors

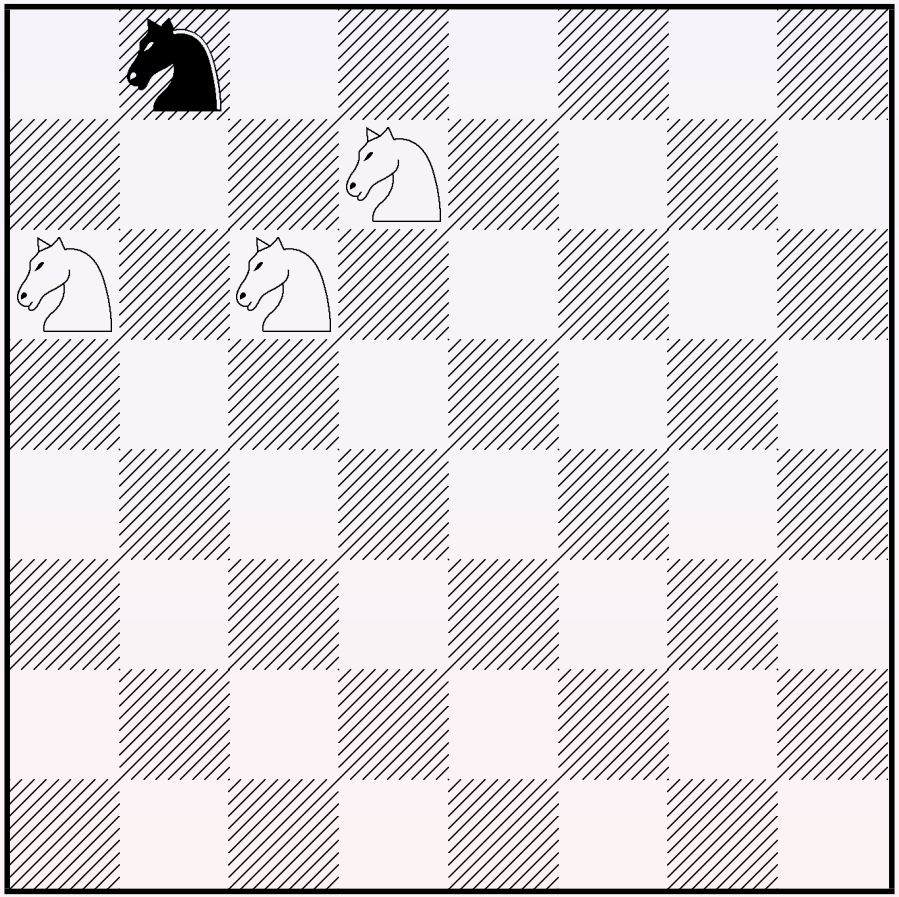


Figure: Which neighbor to visit first?

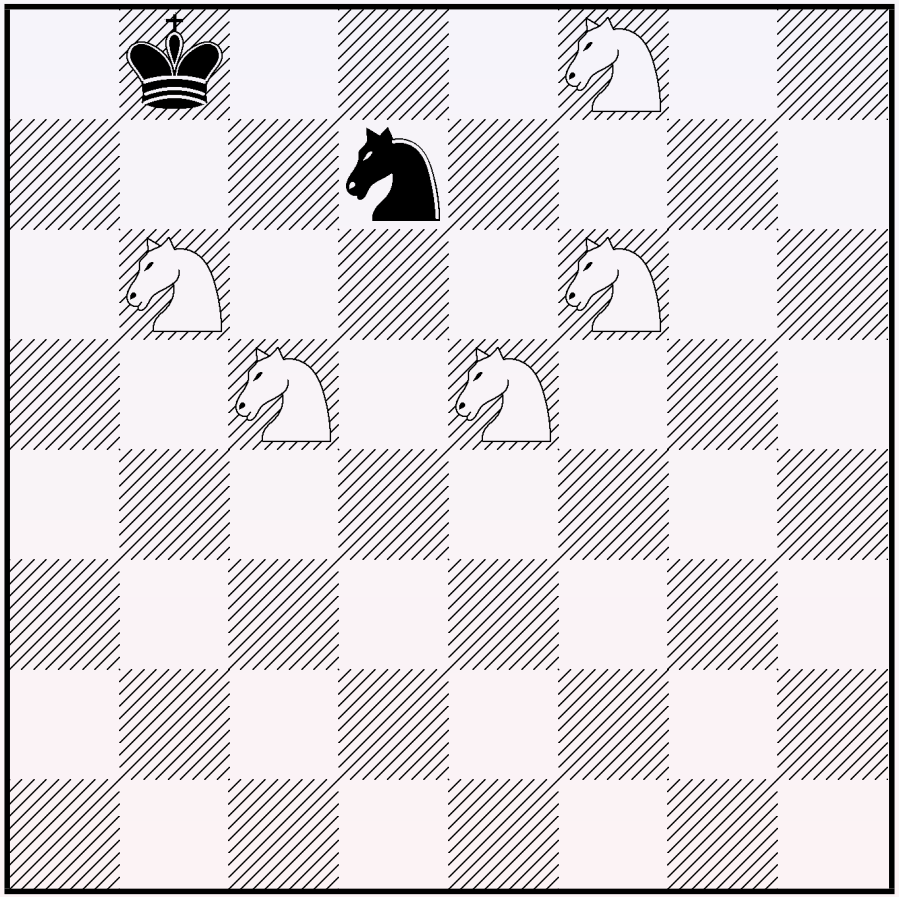


Figure: Right neighbor has rank 5.

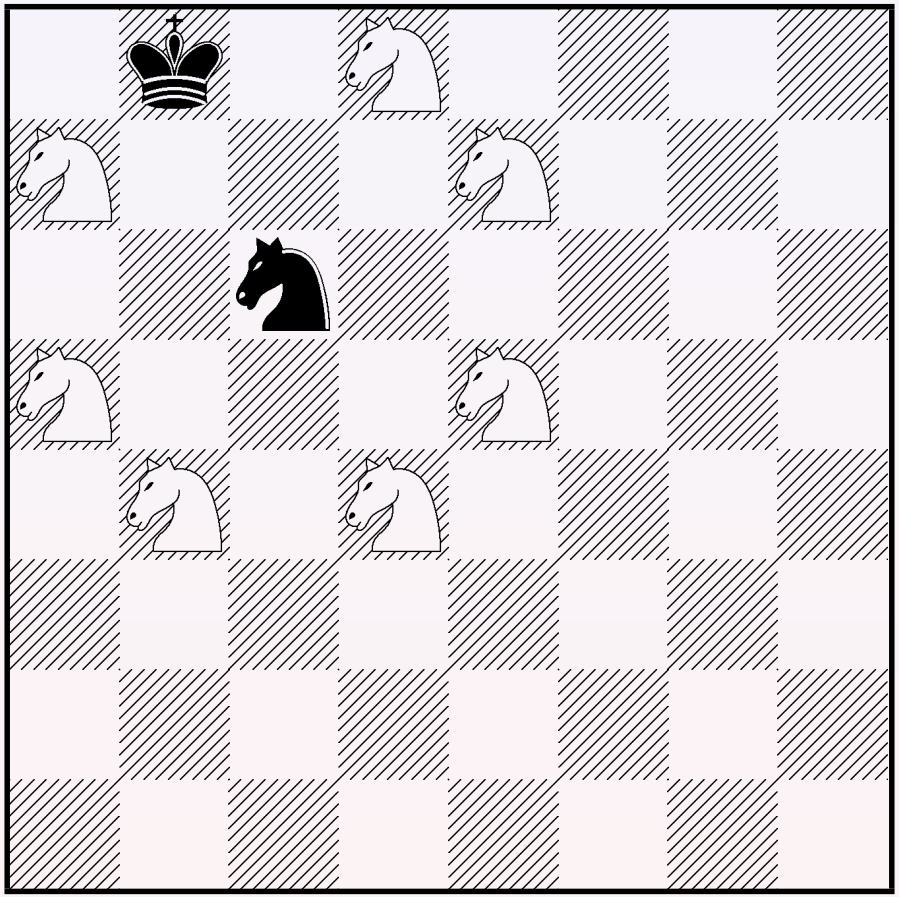


Figure: Middle neighbor has rank 7.

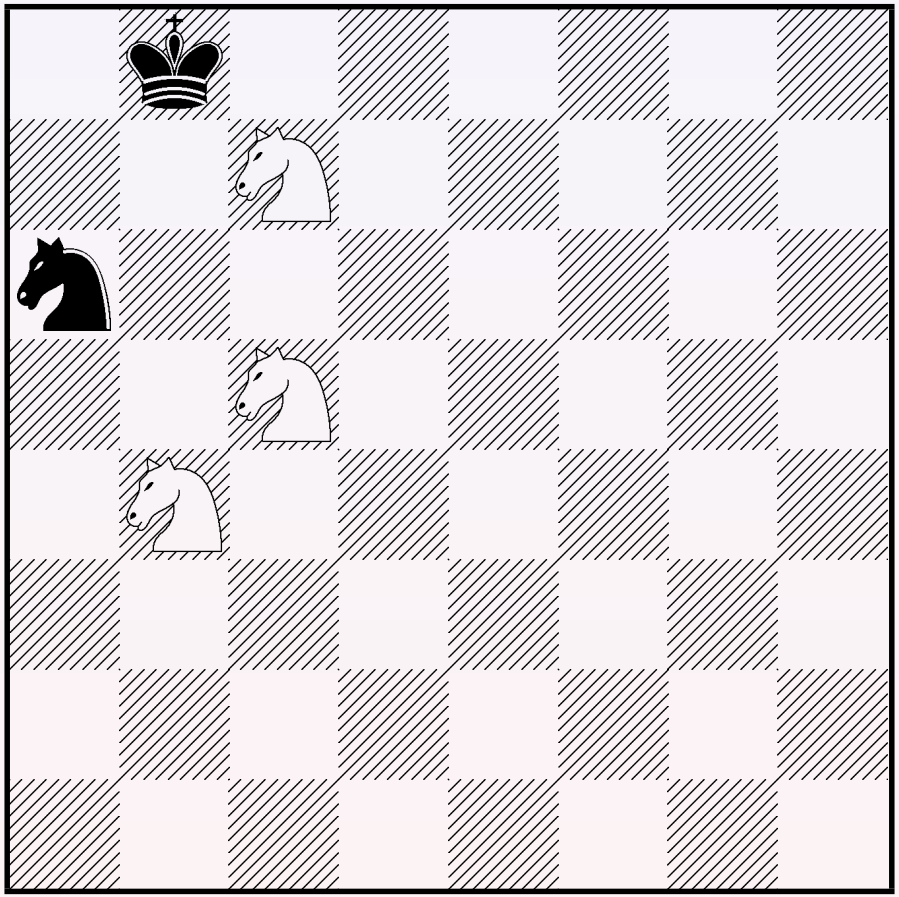


Figure: Left neighbor has rank 3.

Further optimizations

- $2, 3, \cdots n$ square look ahead for Warnsdorff's heuristic
  - Tie arbitration
  - Multiple hops
- Parallel divide and conquer algorithms [2]
  - Divide board into 4 quadrants
  - Solve quadrants in parallel
  - Merge results together
- Genetic algorithms for tour discovery [3]

Warnsdorff's heuristic in Cg

Parallel

- All squares calculate own rank
  - Knight simply reads neighboring ranks
- All squares calculate next move
  - Only knight's calculation is trusted

Sequential

- Only one knight is touring

Moving the knight

Problem

- Pixel shader's output position is fixed
- Cannot alter values of other pixels

Solution

- Knight **indicates** next square in tour
- Next square pulls knight onto itself

A knight's visit

Cycle 1

- Knight arrives

Cycle 2

- Neighboring ranks computed

Cycle 3

- Next square computed
- Knight leaves

Cycle 0

- Knight indicates next square
- Knight prepares for departure

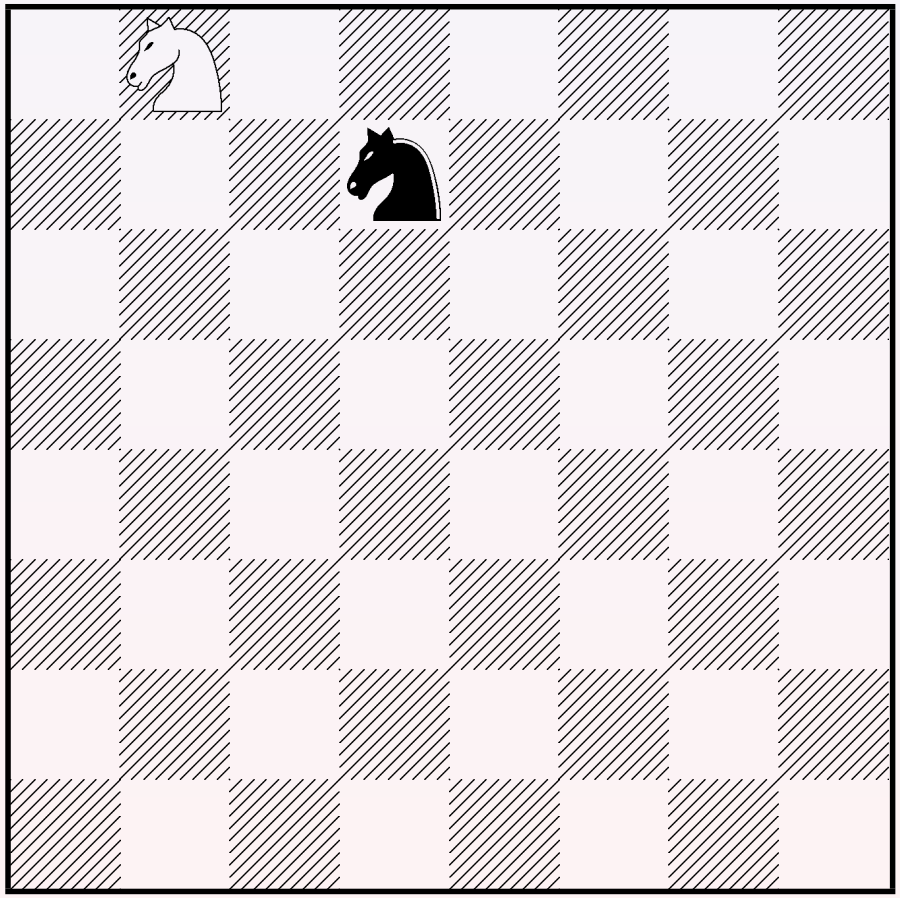
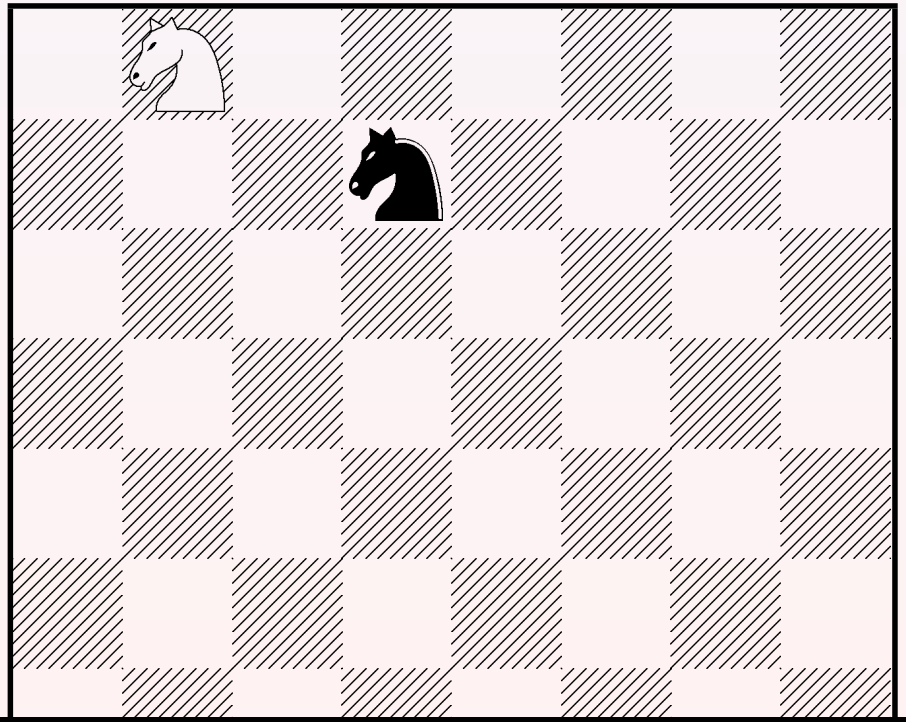


Figure: A knight's travel plans.

Cycle 1

- Knight arrives
- Neighbors compute **wrong** rank

Figure: Previously...



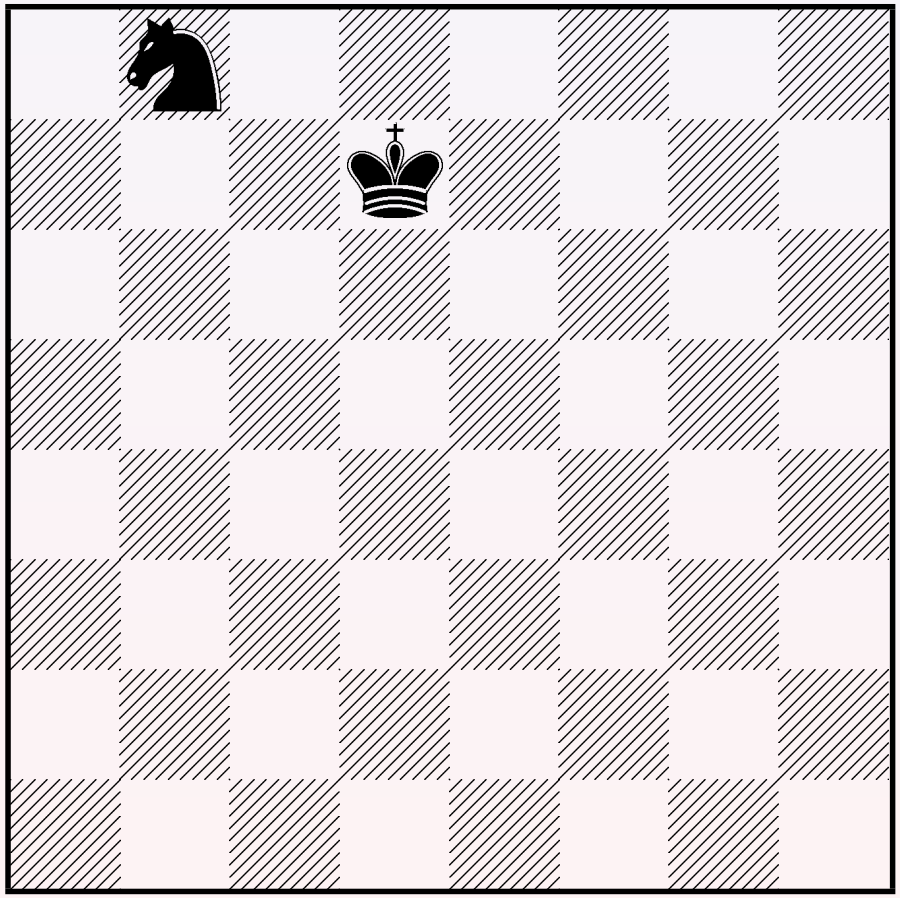
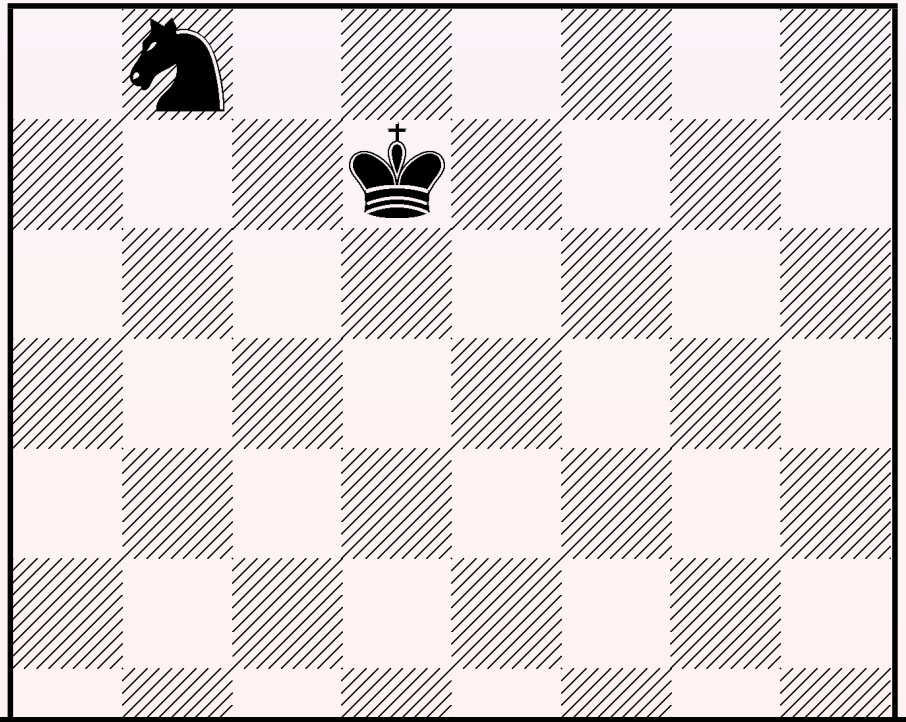


Figure: Unbeknown, a knight arrives.

Cycle 2

- Neighbors acknowledge knight
- Neighbors compute correct rank

Figure: Previously...



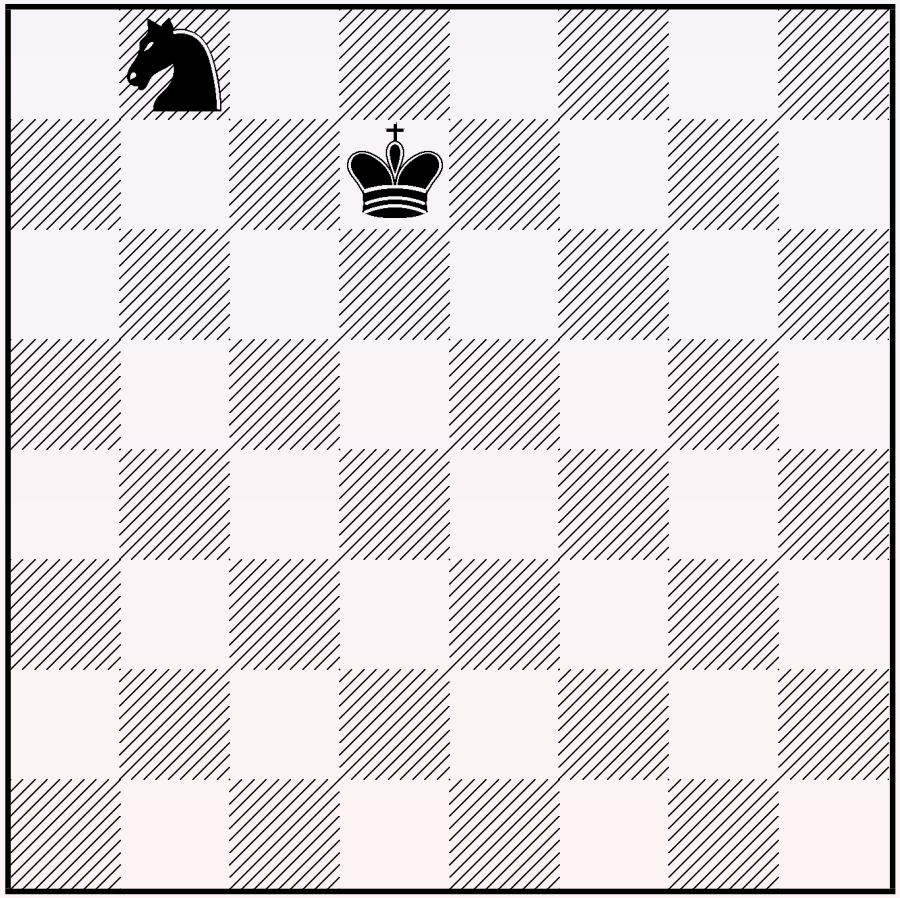
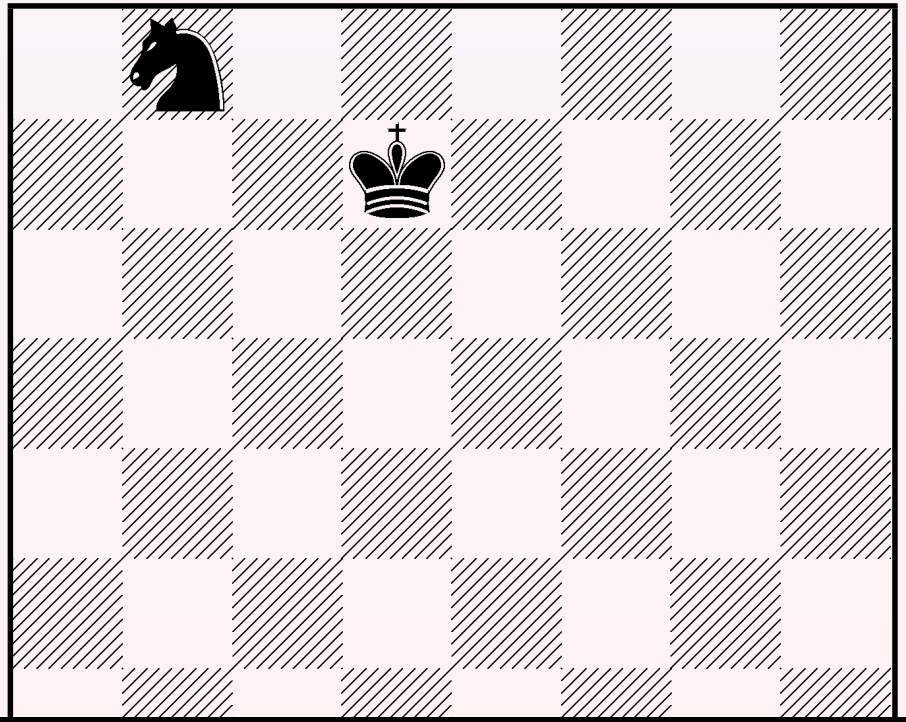


Figure: Hark! A knight be among us!

Cycle 3

- Knight indicates next square
- Knight prepares for departure

Figure: Previously...



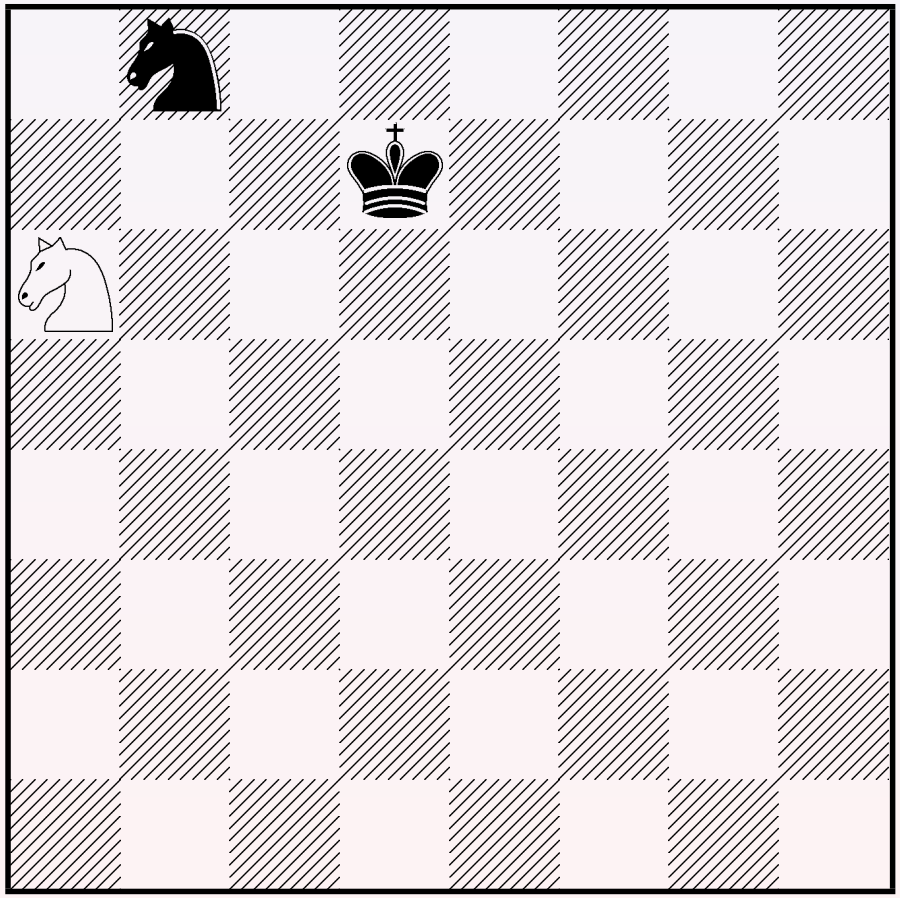
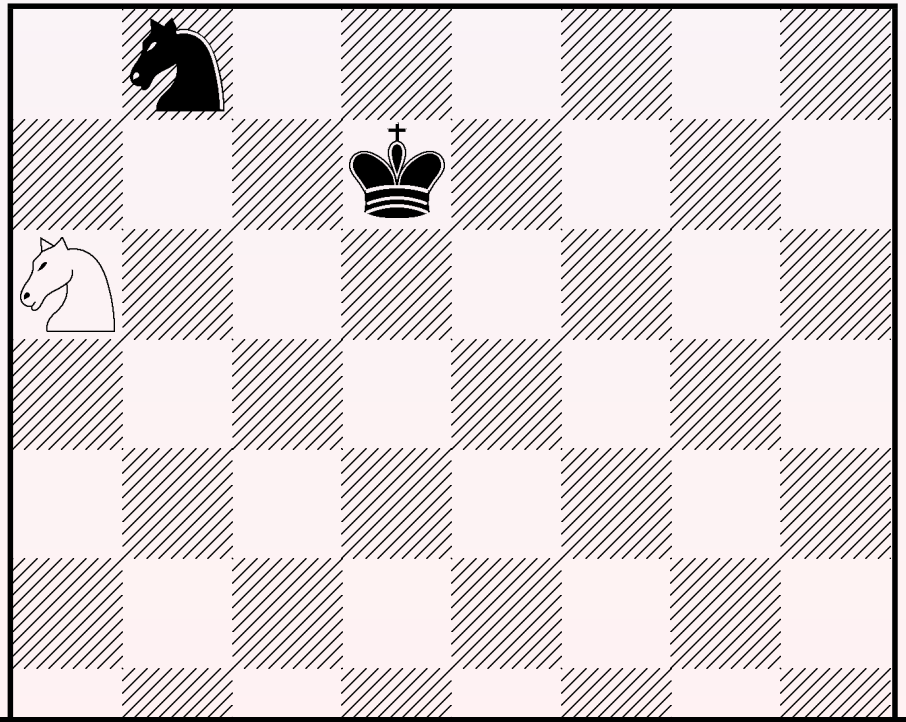


Figure: A knight's travel plans.

Cycle 4

- The process continues

Figure: Previously...



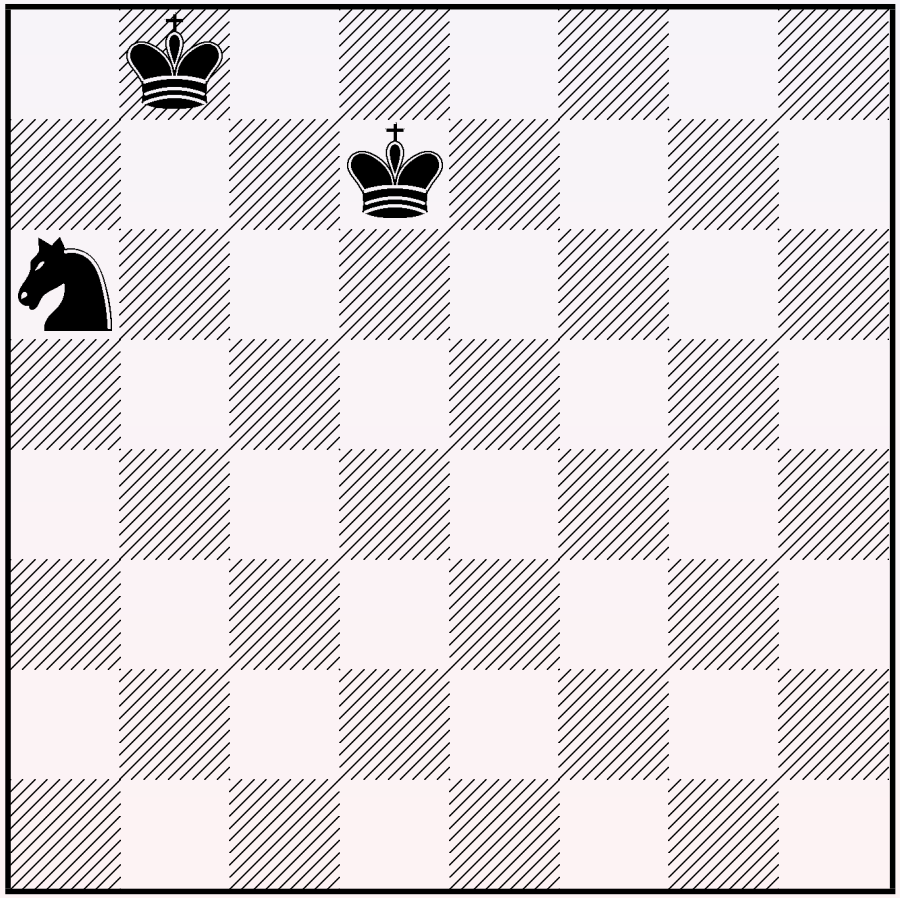


Figure: The tour continues, as before.

Data structure

RGBA array of 32-bit floating point numbers

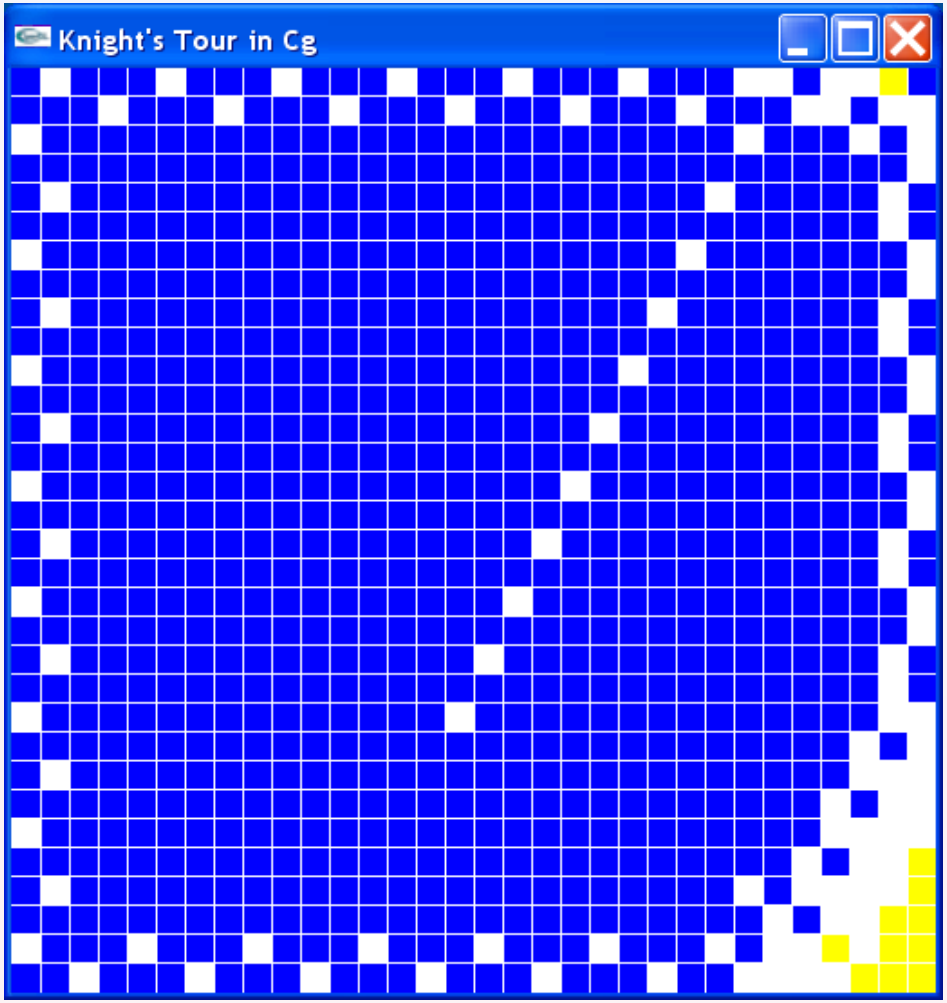
Red	Counter for knight's visits $1, 2, \cdots n$
Green	Counter for knight's decision cycle
Blue	Warnsdorff's rank at this square
Alpha	Knight's next move from this square

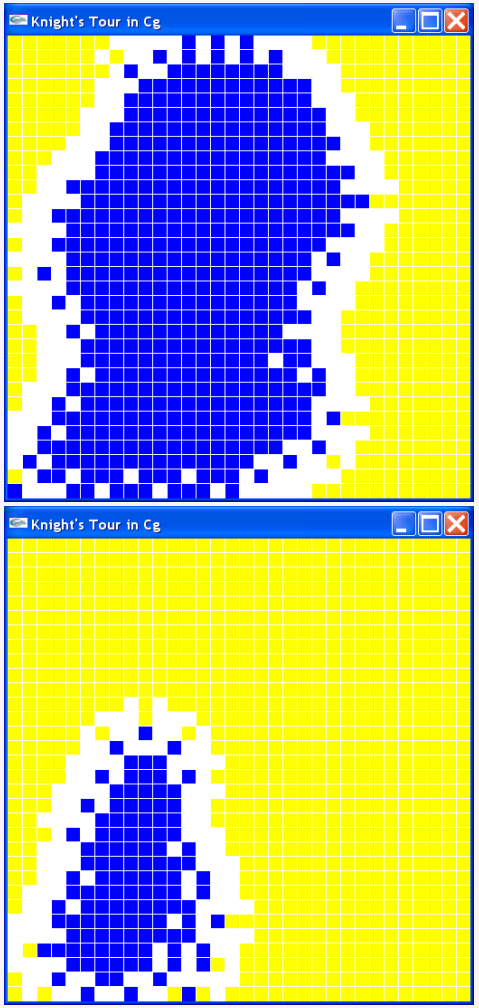
Sample output

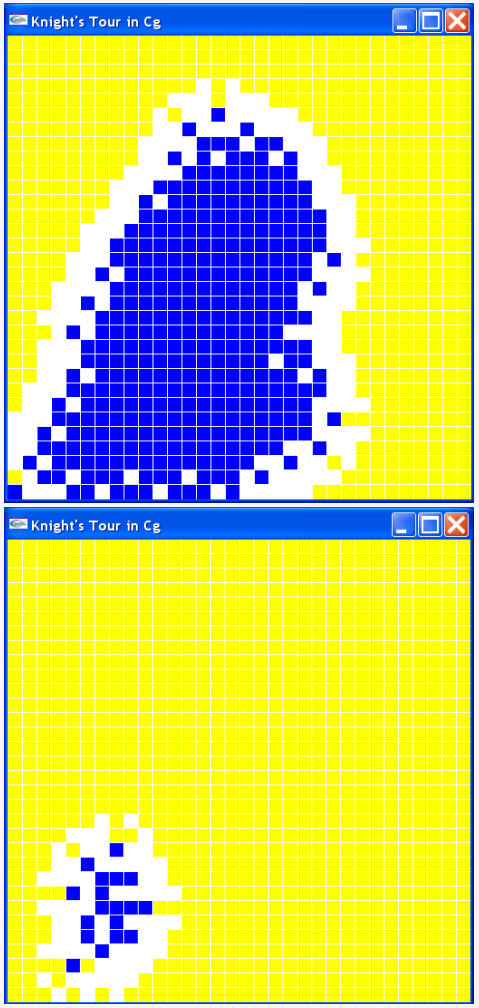
```
chess board's width?       8
knight's starting row?     2
knight's starting column? 4

The Knight toured in this order:

  27  12  37  16  25   2  47  18
  36  15  26  13  50  17  24   3
  11  28  63  38   1  48  19  46
  56  35  14  49  62  51   4  23
  29  10  57  64  39  22  45  20
  34  55  32  61  52  59  42   5
   9  30  53  58   7  40  21  44
  54  33   8  31  60  43   6  41
```








```
chess board's width?       32
knight's starting row?     9
knight's starting column? 15
```

Knight's decision with Warnsdorff's heuristic

- minRank $\leftarrow \infty$
- successor $\leftarrow \emptyset$
- for each neighbor do
  - rank $\leftarrow$ neighbor's rank
  - if rank $<$ minRank
    - minRank $\leftarrow$ rank
    - successor $\leftarrow$ neighbor
  - end
- end

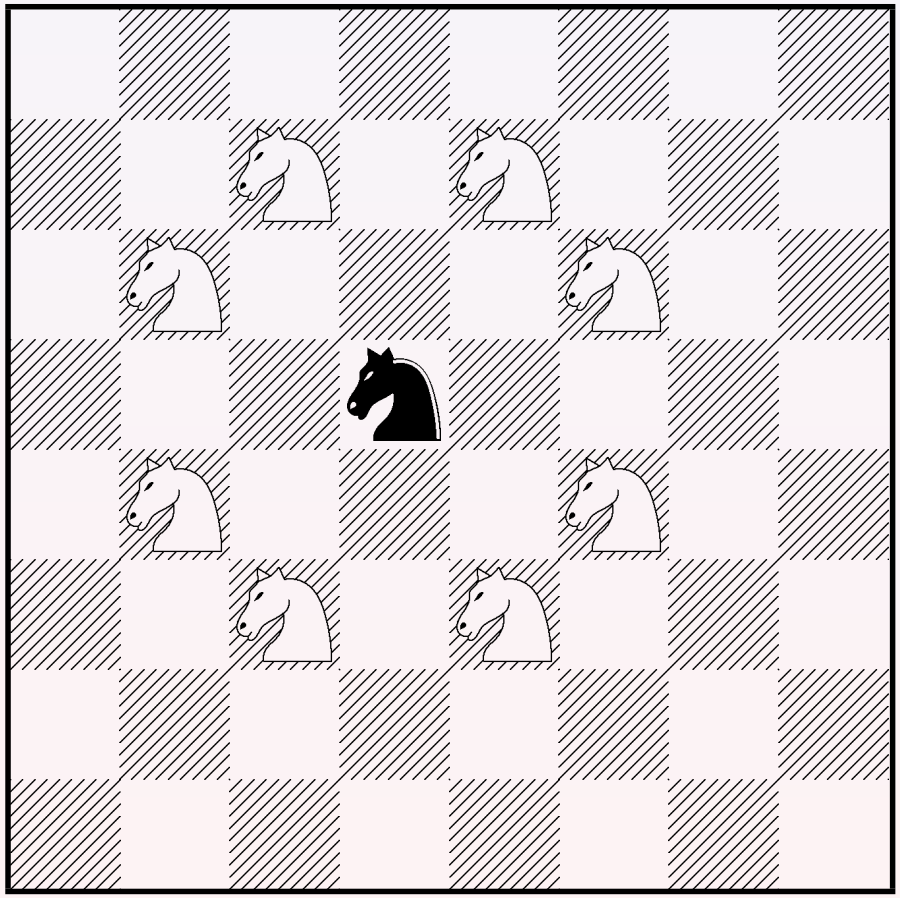


Figure: A knight and his neighbors.

Tour duration with Warnsdorff's heuristic

$$
\begin{aligned}
CPU_{NextMove} &= 8_{neighbors} \times (1_{rank} \times 8_{neighbors} + 1_{minRank}) \\
CPU_{duration} &= N_{knights} \times CPU_{NextMove} \times 1_{cycle} \\
&= N_{knights} \times 72_{cycles}
\end{aligned}
$$

$$
\begin{aligned}
GPU_{NextMove} &= 8_{neighbors} \times (1_{rank} \times 0_{neighbors} + 1_{minRank}) \\
GPU_{duration} &= N_{knights} \times GPU_{NextMove} \times 3_{cycles} \\
&= N_{knights} \times 24_{cycles}
\end{aligned}
$$

Speedup from parallelization of rank calculation

$$
\begin{aligned}
speedup &= \frac{CPU_{duration}}{GPU_{duration}} \\
&= \frac{N_{knights} \times 72_{cycles}}{N_{knights} \times 24_{cycles}} \\
&= 3
\end{aligned}
$$

Lessons learned

- OpenGL back buffer is RGBA8
  - Use FBO extension to preserve floats
- `GL_TEXTURE_RECTANGLE_ARB` does not fit to screen
  - Visualize data using `GL_TEXTURE_2D`
- `texRECT()` clamps out of bound indices
  - Check bounds beforehand
- Cg dislikes > 4 nested loops
  - Unroll loops with code generation
  - eRuby (embedded Ruby) template

K. McGown, A. Leininger, and P. Cull, "Knight's Tour", 15 August 2002, [Article]. Available: http://www.math.oregonstate.edu/~math_reu/Papers/Cull2-2002.pdf. [Accessed: 17 March 2007].

I. Parberry, "An efficient algorithm for the Knight's tour problem", in *Discrete applied mathematics*, 1997, vol. 73, no. 3, pp. 251–260.

V. S. Gordon and T. J. Slocum, "The knight's tour — evolutionary vs. depth-first search", in Proc. *Congress on Evolutionary Computation*, 19–23 June 2004, vol. 2, pp. 1435–1440.

G. Törnberg, "Warndorff's rule", 1999, [Online document]. Available: http://web.telia.com/~u85905224/knight/eWarnsd.htm. [Accessed: 16 March 2007].

I. Karonen, "Knight's Tour", [Image]. Available: http://en.wikipedia.org/wiki/Image:Knight%27s_tour.svg. [Accessed: 20 March 2007].

Questions?

Thanks for your attention.