자료구조의 기초

HW 1

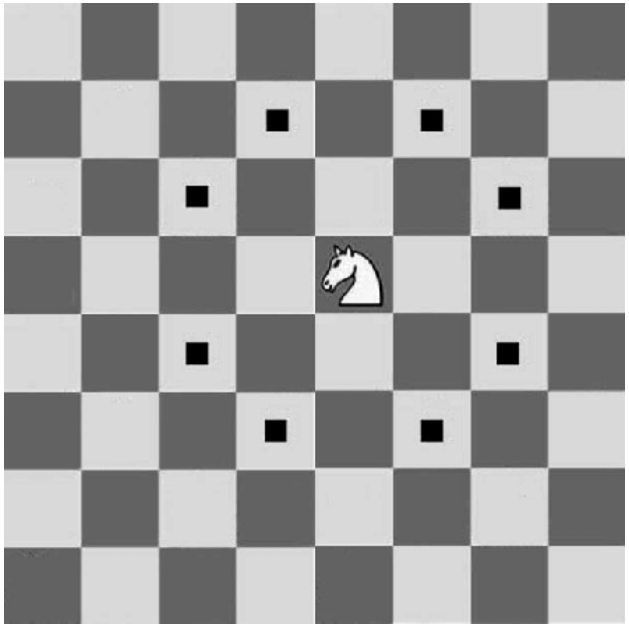
Knight’s Tour Problem

Programming Project

- Additional Exercise 9 [Programming Project] in the book “Fundamentals of Data Structures in C++” (by Horowitz et al.) p.125
    - You should implement in C++

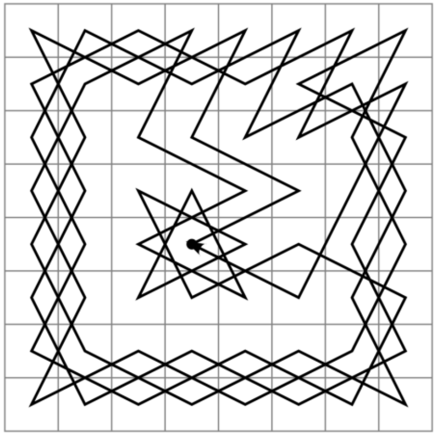
Knight’s Tour Problem

- In Chess, a knight moves in a L-shape on the chessboard
- Q: Is there a sequence of moves that a knight visits all the squares on the chessboard?



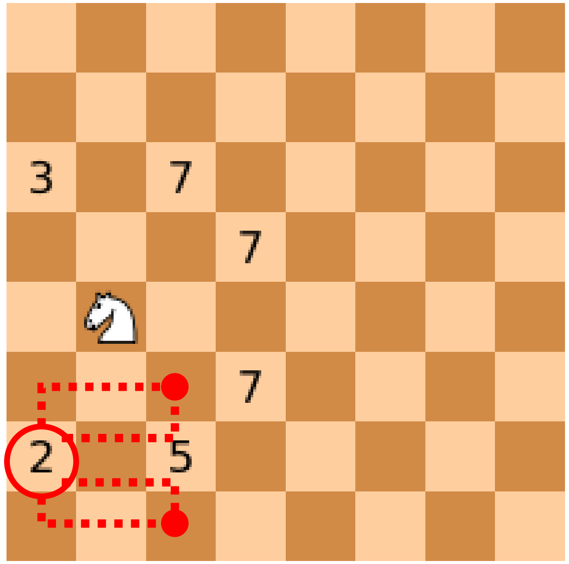
Problem Definition

- Given
  - A 8×8 chess board
  - A starting position (i, j) on the chessboard
    - $0 \leq i, j \leq 7$
    - The bottom-left square is $(0,0)$
- Find
  - A sequence that a knight traverses all squares on the chessboard



Warnsdoff’s Rule

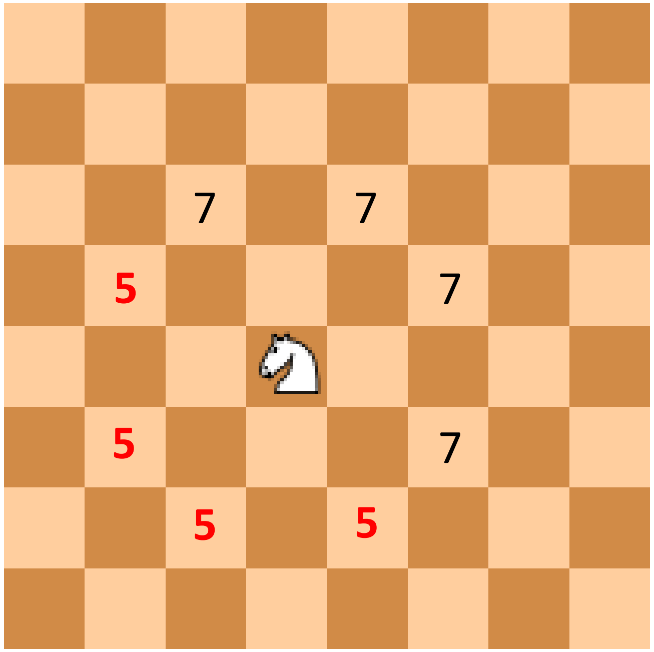
- A heuristic to find a knight tour
  - The knight is moved so that it always proceeds to the square from which the knight will have the *fewest next valid positions*.
  - In the example, an integer in a cell represents the number of legal moves available in the position. The knight moves to (0,1) where the number of next valid positions is the smallest (i.e., 2).



Breaking Ties

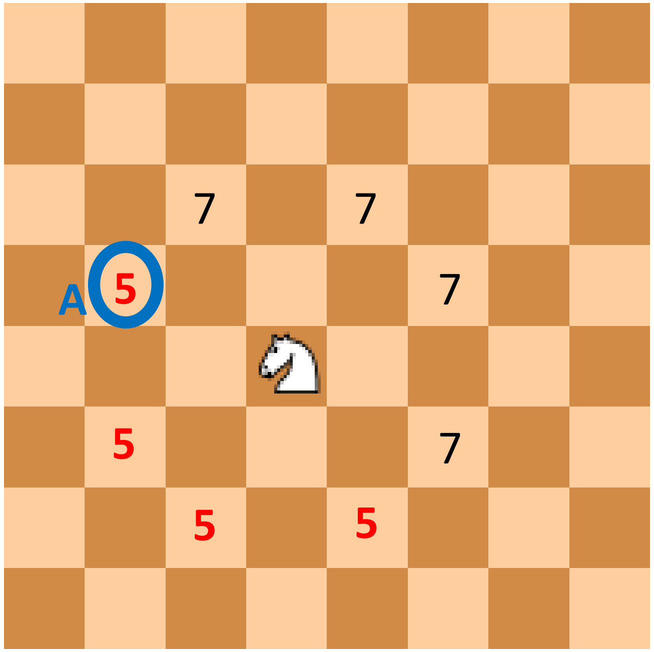
- During the iteration, there are many cases such that multiple squares (i.e., cells) have the same number of legal moves

- Which square should we traverse first?



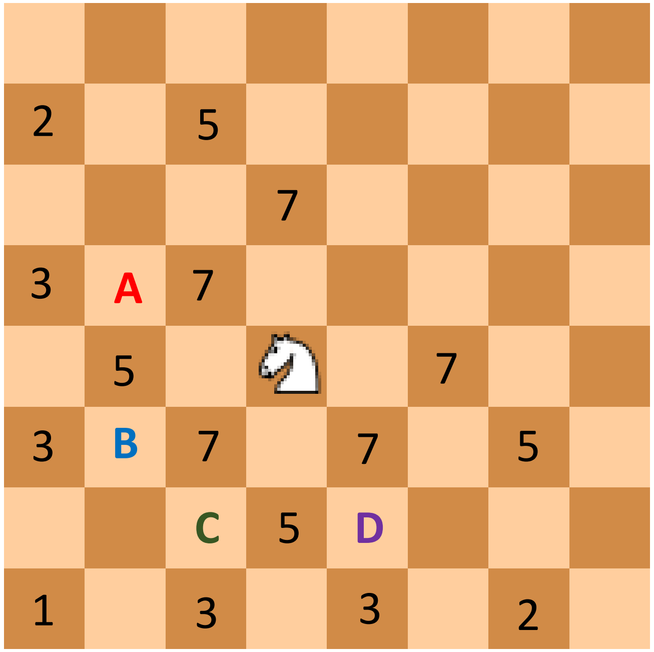
Breaking Ties

- Simple solution
  - Choose the first unvisited square where the minimum number of next positions occurs
  - In this project, starting from position A and iterate the squares in a clockwise direction
  - In this example, we select (1,4) as the next position



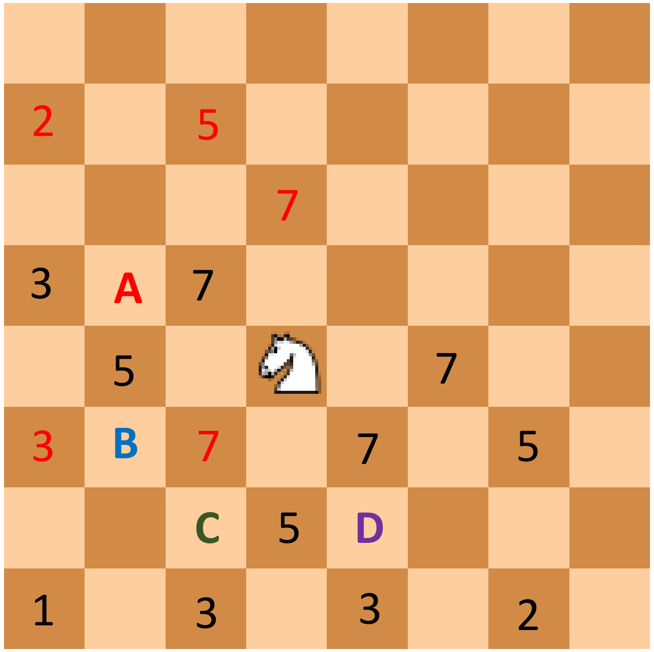
Breaking Ties

- A more complex solution:
  - We consider one more move and choose the first unvisited square where the number of legal moves is the minimum



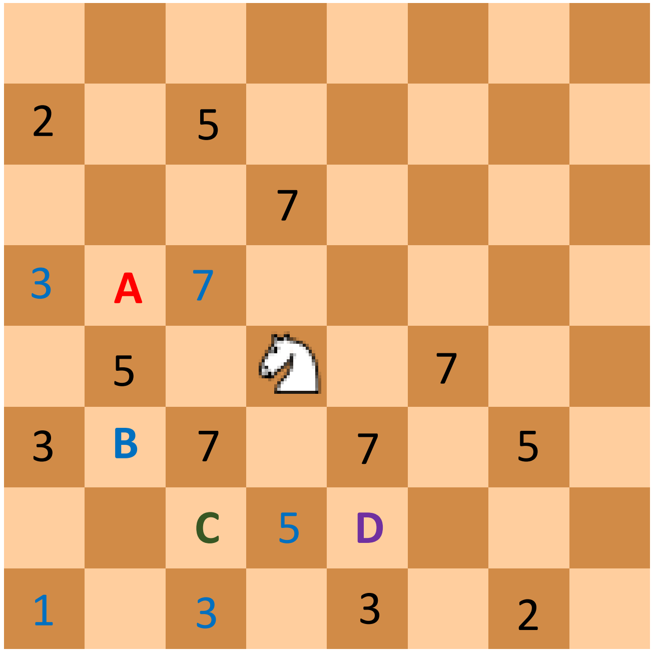
Breaking Ties

- A more complex solution:
  - We consider one more move and choose the first unvisited square where the number of legal moves is the minimum
    - A: 24 moves



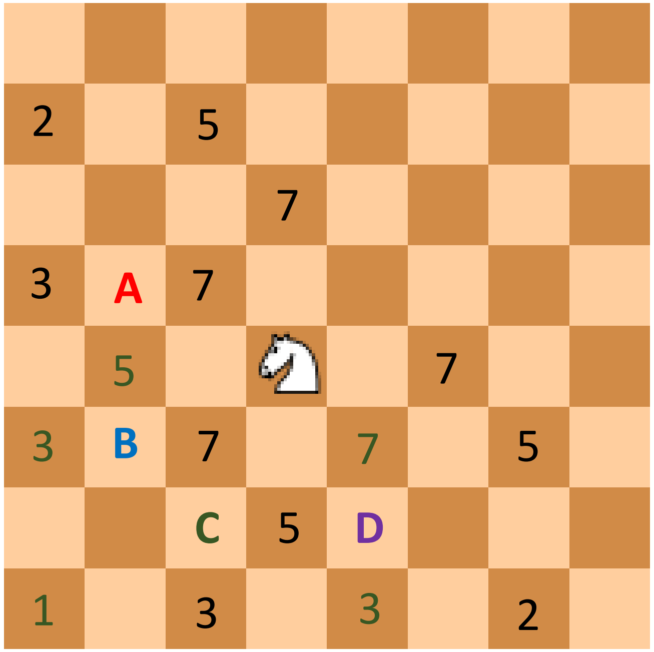
Breaking Ties

- A more complex solution:
  - We consider one more move and choose the first unvisited square where the number of legal moves is the minimum
    - A: 24 moves
    - B: 19 moves



Breaking Ties

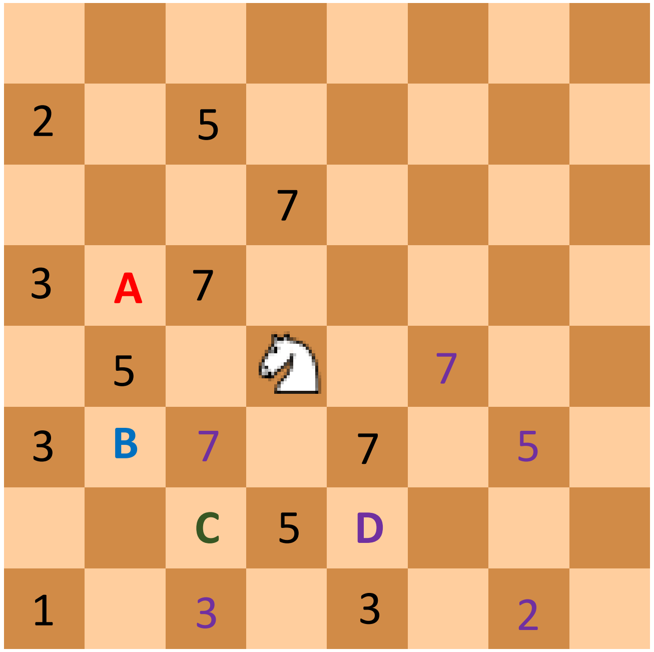
- A more complex solution:
  - We consider one more move and choose the first unvisited square where the number of legal moves is the minimum
    - A: 24 moves
    - B: 19 moves
    - C: 19 moves



2		5					
			7				
3	A	7					
	5		♘		7		
3	B	7		7		5	
		C	5	D			
1		3		3		2	

Breaking Ties

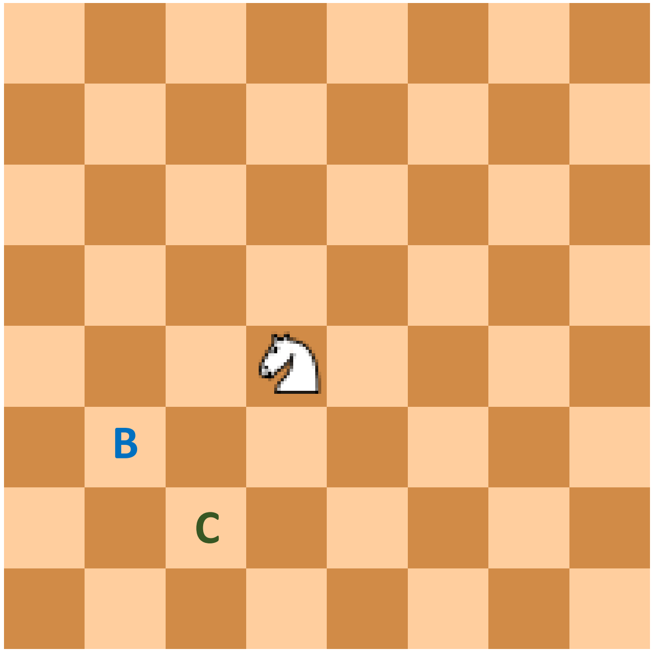
- A more complex solution:
  - We consider one more move and choose the first unvisited square where the number of legal moves is the minimum
    - A: 24 moves
    - B: 19 moves
    - C: 19 moves
    - D: 24 moves



Breaking Ties

- A more complex solution:
  - We consider one more move and choose the first unvisited square where the number of legal moves is the minimum
    - A: 24 moves
    - B: 19 moves
    - C: 19 moves
    - D: 24 moves

Based on the iteration order, we choose C



To Do

- Input
    - Take (i, j) as the starting point of the knight's tour
- Output
    - Success/Failure of the tour
    - 8×8 matrix whose entry is the order of the visit during the tour
        - Starting point is 0
        - If the tour is a failure, unvisited entries are -1
- Tie Break
    - Implement both simple and complex solutions

To Do

- Define Knight class
  - Define member functions required to implement the tour
  - Define member variables including 2-D array which represents the chessboard
  - You need to output the chessboard with the visited order twice (simple/complex tie break solutions)

- Comments in the class is required
  - Refer to Google Style Guide
  - https://google.github.io/styleguide/cppguide.html#Comment_Style

To Do

- Submission
  - The output of the starting position (4,4) and your code
  - Zip file name: (2020-XXXXX HW1.zip)

- Due date
  - 9월 18일(금) 23:59
    - 하루 delay 당 30% 감점
    - 9월 20일(일) 23:59 초과 시 점수 없음