5118020-03 Operating Systems

Homework 4. N-Queens Solver

Shin Hong

Overview

- Extend `nqueen.c`, a single-threaded N-Queen problem solver to a multi-threaded version

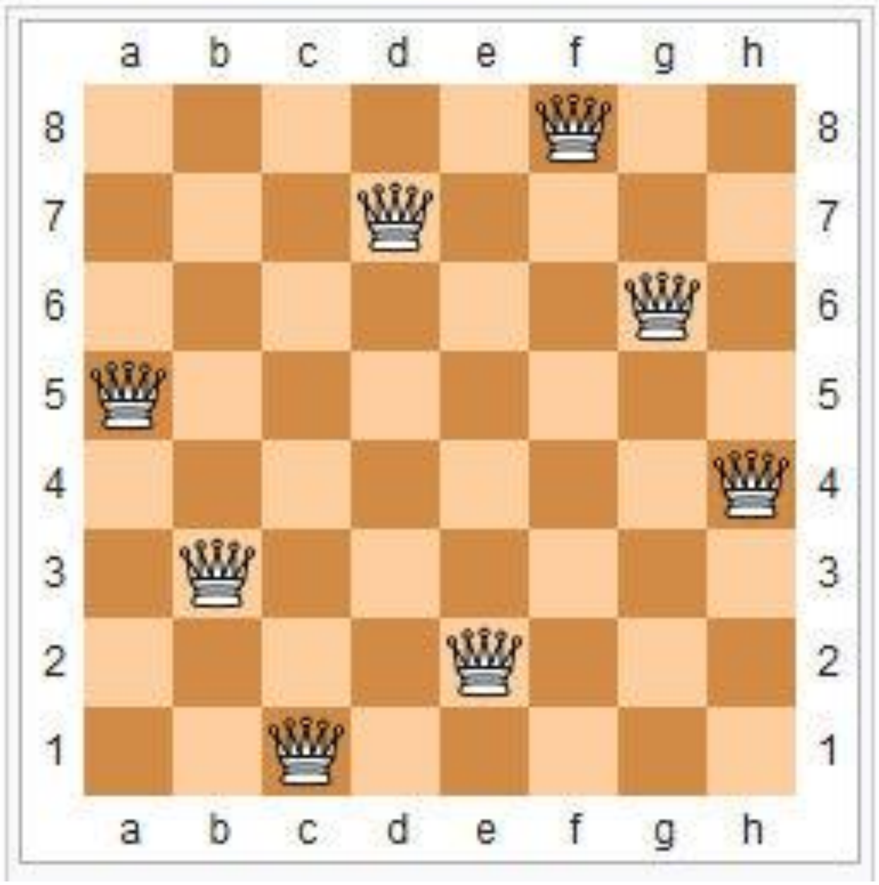
- Point of study
  - Bounded Buffer (or Producer-Consumer)

- Timelines
  - June 10 Mon: First announcement
  - June 17 Mon~: Help desks
  - June 26 Wed: Submission Deadline (Artifact & Presentation Video)

N-Queen Problem

- Find out all non-conflicting placements of N Queens on the N-by-N chessboard
  - Two queens are conflicting when they are placed on the same column, the same row or the same diagonal line in any direction

- Since it is NP-complete, we need to explore all combinations to find an exact solution



Single-threaded N-Queen Solver: `nqueen.c`

https://github.com/hongshin/OperatingSystem/tree/master/assignments/homework4

- Use a stack to explore all possible arrangements of N number of Queens by backtracking
- By default, `nqueen.c` is set to find all 15-Queen placements on the 15-by-15 chessboard
  - you can change the N value by defining `BOARD_SIZE`
- A solution is represented as a list of `N` position numbers
  - a position number `P` represents a cell at the row of index (`P % N`) and the column of index (`P / N`) in the chessboard

Important Functions

- `find_n_queens`
- `find_n_queens_with_preopositions`

Requirements

- Change `nqueen.c` to receive the number of concurrent threads as a command-line argument
  - use `getopt()`

- Use a bounded buffer to parallelize the N-queen solving algorithm

- Print out feasible N-Queen arrangements to the standard output
  - make sure that the printing is not intermixed due to race condition

- When the user presses `Ctrl+C`, print out the total number of found arrangements up to the point, and terminate the program

- You can make minor changes to the given code at `nqueen.c` while adding new functions

Video Presentation

- Take a 4-min video for reviewing the source code and testing the program
  - either in Korean or in English

- Your video must show explain the general structure of the program and how producer-consumer pattern is used

Submission

- All results must be submitted via LMS
  - Source code files
    - Submit all source code
    - You must provide a build script (e.g., bash script or Makefile) and its instruction document (e.g., README) if needed
  - Presentation
    - Submit the video record file; or you can submit the URL to the presentation video on web

- No late submissions will be accepted

Notes

- Welcome your questions anytime on the Slack channel

- Help desks will be offered online or offline, multiple times
  - prior appointment is mandatory

- It is strictly permitted to use auto-programming tools in any form