CSCI B 551: Assignment #0

Due on Tuesday, September 6, 2016

Prof Crandall

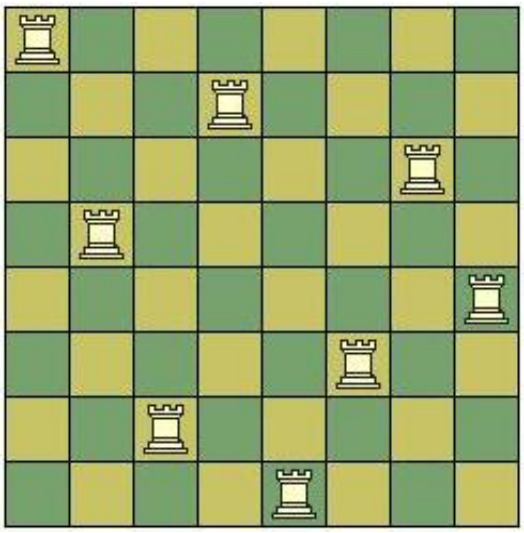
Vivek Patani (vpatani)

Contents

Problem 1

Spend some time familiarizing yourself with the code. Write down the precise abstraction that the programme is using. In other words, what is the set of valid states, the successor function, the cost function, the goal state definition, and the initial state?

- **N Rooks:**
  - **Initial States:** An empty board with no Rooks on the board.
  - **Set of Valid States:** Any of the arrangement which follows the following conditions:
    * There may be 0 to N*N Rooks placed on the board.
    * Only one rook occupies one block on the board. No two rooks can be placed in one block.
  - **Successor Function:**
    * Function that returns set of another states reachable from the current state under consideration.
  - **Cost Function:**
    * There can be about two ways to see it, either there is **no cost** associated to the problem or we can apply a cost of **c (constant)** for each time a Rook is placed on the board or removed from the board.
  - **Goal State Definition:** A Goal state is to have reached when N rooks are placed in such a way that they all abide the valid state conditions i.e.
    * N rooks are on board and none of them attack each other.
    * One of the many solutions for 8 rooks [1]



*

Problem 2

Is the algorithm using BFS or DFS? Switch to the opposite, and explain exactly how to modify the code to do this. What happens now for N=4 and N=8?

- **Algorithm:** DFS (Depth First Search)
- **Reason:** We see the use of a Stack, as we go through the execution of code, we see the last element inserted is popped and being used.
- **Modifications:** Instead of using a Stack Data Structure, we would use a Queue Data Structure for Breadth First Search. Instead of popping the last element of the list, we should pop the very first element. That would just make it work like a Queue. One line of code change is required:

  ```
  for s in successors( fringe.pop(0) ):
  ```

- **For N=4 or N=8:** The programme runs within a decent amount of time for N=4, but as the N increases, for N=8 it runs extremely slowly.

Problem 3

The successor function in the code is defined in a very simplistic way, including generating states that have N+1 rooks on them, and allowing moves that involve not adding a rook at all. Create a new successors() function called successors2() that fixes these two problems. Now does the choice of BFS or DFS matter? Why or why not?

- **successor2()**: The functions needs to check two things:
  - When the count of rooks tries to exceed N, disallow addition in the successor function itself.
  - Check for the current state that is being/going to be pushed to the fringe whether if it's the same as the last one. If it is the same, we do not need to push that otherwise we will be popping the same element over and over again causing an infinite loop.

    ```
    # Get list of successors of given board state
    def successors2(board):
        return [ add_piece(board, r, c) for r in range(0, N) for c in range(0,N)
        if count_pieces(board) <= N and add_piece(board,r,c) != board ]
    ```

  - Here the *if* is of the essence, since it makes sure we do not exceed the number of Rooks and also we do not actually add the same state over and over again.
- **Choice of DFS/BFS?** If we are thinking in terms of reaching the goal then either of them will do the job after modifications made. If time is a factor, then DFS.
- **Why?:** DFS will be able to reach quicker to the solution as compared to BFS.

Problem 4

Even with the modifications so far, N=8 is still very slow. Recall from class that we could define our state space and successor functions in a smarter way, to reduce the search space to be more tractable. Describe your new abstraction, and then modify the code to implement it with a new successors function called successors3(). Feel free to make other code improvements as well. What is the largest value of N your new version can run on within about 1 minute?
Tip: In Linux, you can use the timeout command to kill a program after a specified time period: timeout 1m python nrooks.py

- **N Rooks:**
  - **Initial States:** An empty board with no Rooks on the board.
  - **Set of Valid States:** Any of the arrangement which follows the following conditions:
    - No two rooks are horizontally in line with each other.
    - No two rooks are vertically in line with each other.
    - Basically no two rooks are attacking each other.
    - The number of Rooks on board are $\leq$N.
    - One rook can exactly only occupy one block on the board.
  - **Successor Function:**
    - Function that returns set of another states reachable from the current state under consideration. The difference between the previous function (in Q1) and this function is that, each time we check whether if the placement is a legal one or not.
    - By **Legal** we mean that, No rooks are attacking each other and there are 0 to N rooks on the board each occupying exactly one block.
  - **Cost Function:**
    - There can be about two ways to see it, either there is **no cost** associated to the problem or we can apply a cost of **c (constant)** for each time a Rook is placed on the board or removed.
  - **Goal State Definition:** A Goal state is to have reached when N rooks are placed in such a way that they all abide the valid state conditions i.e.
    - N rooks are on board and none of them attack each other.
    - One rook is allowed to occupy exactly one block at one time.
  - **Largest N value in 1 minute for N Rooks:** 523
- **Side note:** There is a readme file attached on how to execute code for both N Queens and N Rooks.

Problem 5

Now, modify your code so that it solves and displays solutions for both the N-rooks and N-queens problem, using the enhancements above. What is the largest value of N that your new version can solve within 1 minute?

- **For N Rooks**: 523
- **For N Queens**: 15-16 (It has sometimes worked for 16 but last before submission it did not hence 15-16).
- Both results are tested and verified on Burrow Server.

References

1. N Rooks Image
2. Guide #1 For Understanding N Queens
3. Guide #2 For Understanding N Queens
4. Stuart Russell and Peter Norvig - Artificial Intelligence - A Modern Approach (3rd ed.)
5. LaTeX Templates
6. LaTeX Commands
7. Definitions