Fall 2004

COSC 3P71 Artificial Intelligence: Assignment 1

Instructor: B. Ross

Due date: Friday October 8, 12:00 noon.
Lates: Monday October 11, 12:00 noon with 25% deduction; not accepted after this.
Goal: Using blind search and heuristic search to solve a problem.
Languages: Any programming language available in our labs!
Hand in: **(i)** A listing of your source code, and a listing of your program execution for a representative sample of cases, including those requested. Include department cover page. **(ii)** Electronic submission of all your code and data. Use "c3p71a" to submit your code. Run this at the top-level folder of your assignment directory. Be sure to name your top-level folder "Windows", "linux", or "irix", to indicate what platform your executable is compiled for! The marker will use this to try your program.

Super N-Queens: A queen is a chess piece that can move over all squares of a chessboard that are horizontal, vertical, or diagonal to it. For example:

		*			*		
*		*		*			
	*	*	*				
*	*	**Q**	*	*	*	*	*
	*	*	*				
*		*		*			
		*	x		*		
		*				*	

The queen (Q) can move to all the squares with *'s. The N-queens problem is a classical search problem. The problem is to place the N queens on an NxN chessboard, such that no queen takes any other queen. Only 1 queen can be placed on each board square. In the above, a second queen could be safely placed at the "x" location. However, that queen would then take more squares of the board. If you kept doing this, the board would soon be filled. Whether you could place 8 queens on the above board depends entirely on where you place them!

We are going to make the problem a little more interesting! Imagine that every square of the NxN chess board has a positive integer value S(i), i=1,...,NxN. Also imagine that you have K queens to place on the board, ($K \geq 1$). The new problem is as follows: place the queens on the board, so that the total score when summing all the square values S(i) for each queen, is maximized.
(An aside: imagine that these square values are dollars. Then the problem is to win the maximum amount of money, by placing your queens strategically on the board).

You are going to solve some variations of the problem using 2 different techniques:

1. Use a **blind search** scheme from chapter 4 (Winston text).
2. Use a **heuristic search** from chapter 5. Note that heuristic searches are reliant on pseudo-random number generators. If you change the seed of the RAND generator, you will get a different search. Hence you should run heuristic searches a number of times with new seeds.

You can use any blind or heuristic search from these chapters that you like. Note that the blind technique will run exhaustively, to find the best solution. The heuristic technique will use a heuristic score (what could it be?) to try to find a good solution.
(over)

Do the following problems. (Read all the questions before starting, however!)

1. Solve the classic N-queens problem. Use both the blind and heuristic search techniques.
2. Solve the Super N-queens problem using the blind search algorithm. Since it is exhaustive, you will only see a solution with a small board (4x4). Here is an example board to use:

1	8	2	3
1	10	5	4
2	2	6	5
4	9	10	1

Try using different K=1, 2, 3, 4, ... ALso use the following rule:

> **Easy Rule**: do not worry about queens taking each other. Therefore, this problem finds the highest total sum of values (rows, cols, diags) when placing the K queens on K squares of the board.

3. Repeat question 2, but use the following rule:

> **More difficult rule**: If a queen can take another queen(s) on a particular row, column, or diagonal, then the sum of that row (resp. column, diagonal) is taken to be negative. Hence it is probably worthwhile to not have queens interfere with each other, as it means a lower score. On the other hand, depending on the board values, it might be acceptable for some interference!

Note that if this rule uses a board with equal square values, then its solution also solves the classic N-queens problem in question 1 (hint!).

4. Repeat question 2, but use a heuristic search algorithm. Try the problem with different K (1, 2, ...). Also try the problem with this larger board, using K=4 and K=8.

5	100	10	5	8	2	1	1
5	5	5	50	5	25	25	5
2	5	5	5	50	5	1	1
20	100	50	200	25	25	50	25
1	5	100	1	1	1000	50	1
1	1	25	100	25	50	100	200
25	5	2	10	50	1	50	1
50	1	25	5	1	10	1	50

5. Repeat question 3, using a heuristic search algorithm. Try different K again, and also try the larger board in question 4, using K=4 and K=8.

Comments:

Each run should print the following out:

a) The random number seed used (for heuristic searches). This lets you reproduce your results, and is useful for debugging your program.
b) The board configuration of the solution, written as an ASCII table.
c) The final sum (prize money). You should be able to hand-calculate the sum to verify that it corresponds with the board configuration.
d) .The total # board configurations tested. This will give you an idea of how much work was done to find a solution.

Be sure to organize all your search experiments, since there are different variations of each (easy and difficult, blind and heuristic, different K, etc.) Clearly document the results of each experiment. The marker should be able to tell exactly what experiment is being run, by reading your documentation for the experimental results.

You can capture text output in a unix window using the "script" utility.

The next state generator you implement should return a configuration of queens on the NxN board. This could be a NxN binary table.

The programming language you use may influence your representation and implementation. Note that your program shouldn't literally denote the search tree with a tree data structure. The textbook uses queues, and recursive programs can use the recursive stack calls to implicitly represent the search tree.

Recursion can simplify your algorithm. But beware of infinite looping, especially with blind searches (ie. repeated states).

Your program will be marked on correctness and style. Marks will be deducted for inadequate documentation; please discuss your design decisions in your program comments.