ECE 566

Assignment

Dr. M. Marefat

Constraint Satisfaction

1. Introduction and Goals of the assignment.

This is an assignment designed to practice the Constraint Satisfaction techniques discussed in class, especially Discrete Constraint Satisfaction problems.
This Assignment requires you to formulate and solve the 8-Queens Problem as a CSP (Constraint Satisfaction Problem). A computer implementation of two algorithms (specified below) is to be developed and a comparison of the performance of the two algorithms is to be carried out.

2. Problem Details.

The N X N Queen problem is a standard problem used in Computing for benchmarking algorithms. In this assignment we will use the 8x8 Queens problem. The object of the 8-Queens problem is to place 8 Queens on a Chess-board with 64 squares, so that no two queens attack each other.
In the figure below, "•" represents a Queen. Usually in the 8-Queens problem the position of a Queen is given in terms of the row and column number corresponding to that square. Rows are numbered from 1 to 8 and the columns are numbered from A to H. For example in the problem below, The Queen in the top left most corner is given by 1A.

	A	B	C	D	E	F	G	H
1	•	X	X	X	X	X	X	X
2	X	X						
3	X		X					
4	X			X				
5	X				X			
6	X					X		
7	X						X	
8	X							X

Attack: A queen A is said to attack another queen B, if the Queen B is in the same row as Queen A, or in the same column as Queen A or is in a square which is diagonal to the square occupied by Queen A.
Example: In the above figure, If we place a Queen in position 1A as shown it will attack all the squares marked with an X, Thus we cannot place any other queen in any of these squares.

Object or goal of the problem:

USING CSP, FIND POSITIONS FOR THE 8 QUEENS ON THE BOARD SUCH THAT NO QUEEN ATTACKS ANY OTHER QUEEN.

Example: Given below is one of the several possible solutions to the problem.

	A	B	C	D	E	F	G	H
1						●		
2				●				
3							●	
4	●							
5								●
6		●						
7					●			
8			●					

3. Project Tasks

3.1 Formulate this problem as a Constraint Satisfaction problem, State the variables, the domain of each variable and the constraints. Is this problem a Binary CSP? Can you draw a constraint graph for the smaller 4 Queens version of this problem?

3.2 Develop the implementations specified below to solve the problem. The program **should accept the Position of the first queen** in some row and generate the positions for the remaining rows. For both cases described below the program should feature a counter to ***count the number of backtracks made in each row*** (i.e. No. changes made to the assigned value for the Queen's position in a particular row before finding the possible solutions for the given problem?) till the final solution is found. (This is a simple minded measure of the efficiency for each method.)

(a) Implement the **Simple Chronological Backtracking** to solve the problem.
(b) Implement the **Directional Arc consistency Look-ahead** algorithm to solve the solution.

You may use either C/C++, or Java to implement your program.
You may also use two programs for the implementation of the two algorithms. Debug your program and make sure it runs correctly on your U. of A. ECE account.

The program should be able to **give the positions of all the queens for every row (using the convention specified) based on the position for the first queen specified/constrained by the user. Your program should be able to find all consistent solutions and report them or report that there are no possible solutions for the problem**. Please follow the convention shown below in designing the program to receive its input and generating its output.

Example input:

```
Please enter the position of a queen:
1A
```

(note that based on the first position, the number of backtracks to find a solution may vary).

Output format to be followed:

```
Solution 1:
   The positions of the Queens are:
   Row 1: 1A
   Row 2: 2E

   Etc….

   The numbers of backtracks before this solution found
   Chronologic Backtracking:
   Row 1:  0
   Row 2:  4

   Etc.

   Directional Look Ahead:
   Row 1:  0
   Row 2:  4

   Etc.


Solution 2:
   The positions of the Queens are:
   Row 1: 1A
   Row 2: 2F

   Etc….

   The numbers of backtracks before this solution found
   Chronologic Backtracking:
   Row 1:  0
   Row 2:  4

   Etc.

   Directional Look Ahead:
   Row 1:  0
Row 2:   4
```

Etc.

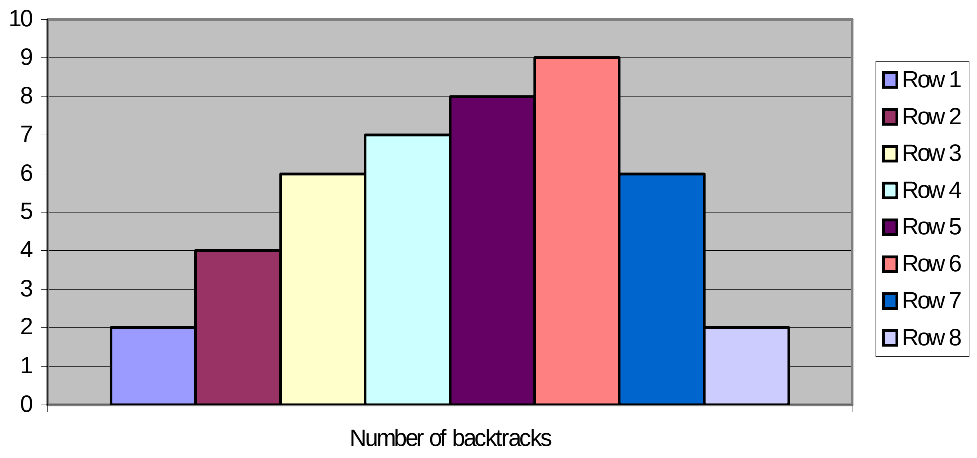
4. Outputs

Generate the following results:

(a) Using the position of the first Queen as 1A, generate **all possible** solutions to the problem and present the output in the format described in section 3 above.

(b) Using the row counter specified in section 3, generate a comparison of the number of backtracks made by each of the two algorithms in finding each solution in 4(a), and characterize the search space in the following way. Using any graphing program (Excel, X-graph etc.), Plot histograms *(for each distinct solution by each f the two algorithms)* to characterize/compare the number of backtracks made to generate the queen position for each row. For example, the graph below shows us that in the first row 2 changes were made, in the second row 4 changes were made, and so on till this particular solution was found. (Note that the shown graph is for fictitious data, your data/graphs may be completely different from the one shown here). You should have one pair of graphs for each consistent solution of 4(a) (or one graph depicting the result of the two algorithms side by side).

Notice that your program should be able to take from user any position for queen one, and it should find and report every solution consistent with that input. But for the purpose of the graphs, you only generate graphs for the case that position of the queen, is selected as 1A.



5. Summary

5.1 Turn in the following:

(1) Programs (Source code) to solve the problems in any of the languages specified above. Your program should allow the user to specify the desirable position of one queen, and find all solutions consistent with the user specification.
Note: The files must be named **program1.x** and **program2.x**, where **x** is the appropriate file extension based on the programming language used. For example: If C was used, the files will be **program1.c** and **program2.c**.

(2) Output file: showing Queen positions and number of backtracks for the case mentioned in Section 4(a) (that is all solutions consistent with having queen no. 1 in position 1A).
Note: The output files must be named **output1** and **output2**.

5.2 Turnin a a pdf file containing the following (assemble as a report) :

(1) Discussion of all the questions in Sec 3.0
(2) The output you obtained for the Question 4 (a) for the two algorithms.
(3) The graphs characterizing/comparing the search space for solution(s) by the two algorithms for the case mentioned in Section 4(a) (that is all solutions consistent with having queen no. 1 in position 1A).
.