**Week 4: Paper and pencil analysis of solving 8 queen problem using Backtracking**

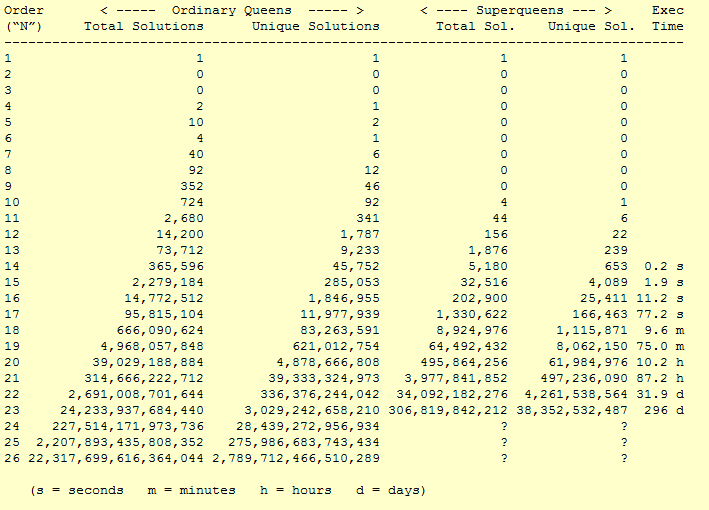
*Submitted by: Sandip Sahani, Iksha Gurung, Milan Thapa*

**Backtracking review:**

Backtracking method represents one of the most general techniques. It is used for most of the problems seeking for a set of solution or an optimal solution. Let us consider a set of solution say Si then the best possible outcome or the optimal solution can be obtained using backtracking, the set being (x1,x2,x3,…,xn ), where xi represents an optimal solution.

The problem in hand i.e. the 8 Queen problem be solved using the backtracking method. The problem also can be solved using the brute force method but would lead high computation. The backtracking method enables us to eliminate the computations that are not necessary to be computed minimizing the number of steps to be computed. Here, while using the brute force method we need to compute a combination of (64, 8) which equals about 4.4 billion combinations to be computed. But using the backtracking method we avoid the combinations that is not possible by checking whether the queen can be placed at the certain cell or not and reduce the computation to 8! This equals only 40,320 combinations.

**Time complexity reviews:**



The above table shows the computation time taken by the N-Queen.

The following is the plot of time complexity of the given data.

Figure 1: time complexity of N-Queen problem

\*Note:- the time consumption is measured in seconds.

**Algorithm:**

At first we need to consider the placement of the queens in the board. When a queen is placed in a cell, no other queen can be placed in the same row and column as of the placed queen in addition with the diagonal from the cell where the queen is placed. With close observation we see that the diagonal going from left-top corner to right-bottom corner has a fixed row-column value I.e. 2 and the diagonal from right-top corner to left-bottom corner has a fixed value of row + column value that is 6. Including these values in the placement algorithm we get the pseudo code for placement to be:-

|  |
| --- |
| **Algorithm Place(k,i)**  /\*Returns true if a queen can be placed in the kth row and ith column. Otherwise it returns false. Abs(r) returns the absolute value of r. x[] is a global variable whose k-1 values are set.\*/  {  for j:=1 to k-1 do  if((x[j]=i)or (Abs(x[j]-1)= Abs(j-k)))  then return false;  return true;  } |

The above algorithm returns true if a queen can be placed in the kth row and ith column else returns false.

The following pseudo-code is of backtracking method for the 8 Queen problems.

|  |
| --- |
| **Algorithm N\_QUEEN\_BACKTRACKING(n)**  {  j:=1;  for(k=1;k<=n;k++)  {  while(j!=0) do  {  if(j<n and place(k,j))  {  if(k==n)then write the current co-ordinate  j:=j+1;  }  else  j:=j-1;  }  }  } |

In the above pseudo-code we first take the kth row to place a queen. Then the column of the row where the queen can be placed is found.

The efficiency of the above algorithm purely relies on the following four factors:

1. Time taken to place next queen in the board.
2. Time to check the placement of the queen.
3. The number of cells satisfying the given constraints like where the queens are placed.
4. Number of cells satisfying the placement of the queens in the board.

The solutions from the above algorithms are placed in the leaf nodes of a tree. While traversing a tree (depth first) we must first know the number of nodes created while looking for solutions. So depending upon the situations the number of nodes created will be either 2^n or n! which gives a time complexity of O(p(n)2^n) or O(q(n)n!) respectively.(here p(n) and q(n) are the polynomials in n.)

**References:**

* Horowitz, E., Sahni S., Rajasekaran S., “Fundamentals of Computer Algorithms”, University Press, Second Edition
* Cormen et. Al. “Introduction to algorithms”, MIT Press, Third Edition
* Time complexity of N Queen problem, <http://www.durangobill.com/N_Queens.html>, [Accessed on June 12, 2012]