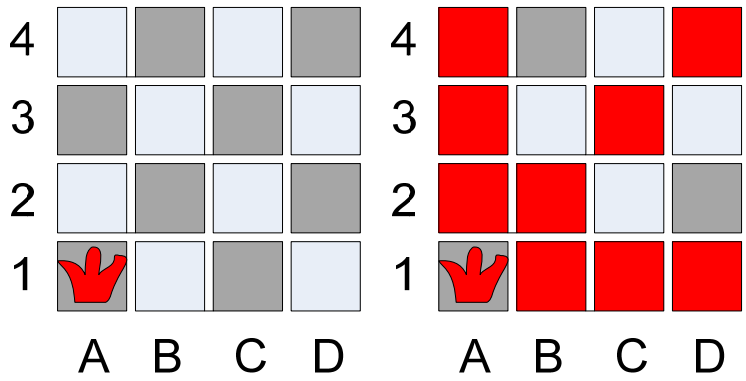
Solving 8 queen problem by backtracking

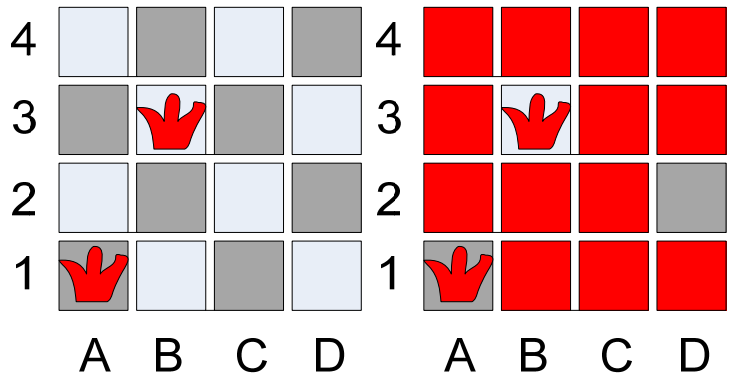
The 8 queen problem is a case of more general set of problems namely “n queen problem”. The basic idea: How to place n queen on n by n board, so that they don’t attack each other. As we can expect the complexity of solving the problem increases with n. We will briefly introduce solution by backtracking.
First let’s explain what is backtracking? The boar should be regarded as a set of constraints and the solution is simply satisfying all constraints. For example: Q1 attacks some positions, therefore Q2 has to comply with these constraints and take place, not directly attacked by Q1. Placing Q3 is harder, since we have to satisfy constraints of Q1 and Q2. Going the same way we may reach point, where the constraints make the placement of the next queen impossible. Therefore we need to relax the constraints and find new solution. To do this we are going backwards and finding new admissible solution. To keep everything in order we keep the simple rule: last placed, first displaced. In other words if we place successfully queen on the i^{th} column but cannot find solution for $(i+1)^{th}$ queen, then going backwards we will try to find other admissible solution for the i^{th} queen first. This process is called backtrack
Let’s discuss this with example. For the purpose of this handout we will find solution of 4 queen problem.

Algorithm:
- Start with one queen at the first column first row
- Continue with second queen from the second column first row
- Go up until find a permissible situation
- Continue with next queen

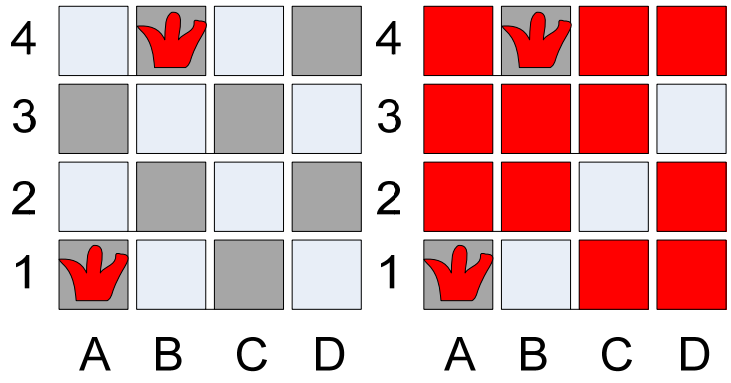
We place the first queen on A1:



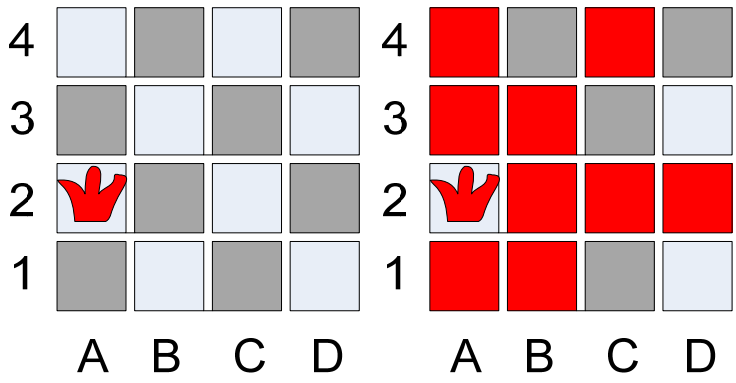
Note the positions which Q1 is attacking. So the next queen Q2 has to options: B3 or B4. We choose the firs one B3



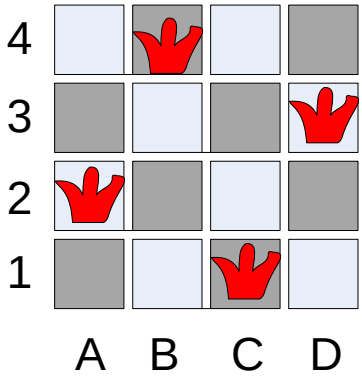
Again with red we show the prohibited positions. It turned out that we cannot place the third queen on the third column (we have to have a queen for each column!). In other words we imposed a set of constraints in a way that we no longer can satisfy them in order to find a solution. Hence we need to revise the constraints or rearrange the board up to the state which we were stuck. Now we may ask a question what we have to change. Since the problem happened after placing Q2 we are trying first with this queen.
OK we know that there were to possible places for Q2. B3 gives problem for the third queen, so there is only one position left – B4:



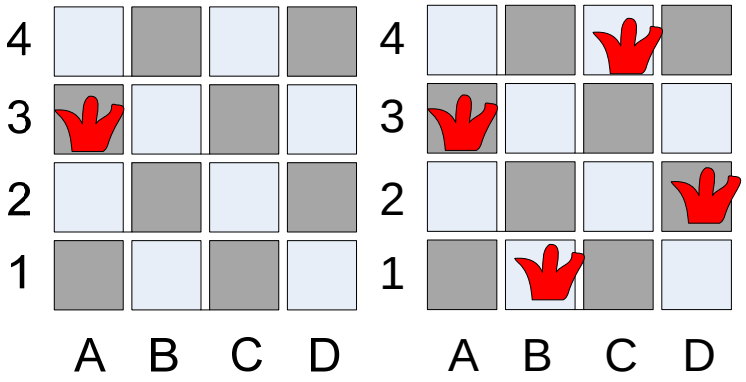
As you can see from the new set of constraints (the red positions) now we have admissible position for Q3, but it will make impossible to place Q4 since the only place is D3. Hence placing Q2 on the only one left position B4 didn't help. Therefore the one step backtrack was not enough. We need to go for second backtrack. Why? The reason is that there is no position for Q2, which will satisfy any position for Q4 or Q3. Hence we need to deal with the position of Q1.
We have started from Q1 so we will continue upward and placing the queen at A2



Now it is easy to see that Q2 goes to B4, Q3 goes to C1 and Q4 takes D3:



To find this solution we had to perform two backtracks. So what now? In order to find all solutions we use as you can guess – backtrack!
Start again in reverse order we try to place Q4 somewhere up, which is not possible. We backtrack to Q3 and try to find admissible place different from C1. Again we need to backtrack. Q2 has no other choice and finally we reach Q1. We place Q1 on A3:



Continuing further we will reach the solution on the right. Is this distinct solution? No it is rotated first solution. In fact for 4x4 board there is only one unique solution. Placing Q1 on A4 has the same effect as placing it on A1. Hence we explored all solutions.

How implement backtrack in code. Remember that we used backtrack when we cannot find admissible position for a queen in a column. Otherwise we go further with the next column until we place a queen on the last column. Therefore your code must have fragment:

```
int PlaceQueen(int board[8], int row)

If (Can place queen on ith column)
        PlaceQueen(newboard, 0)
Else
        PlaceQueen(oldboard,oldplace+1)
End
```

If you can place queen on ith column try to place a queen on the next one, or backtrack and try to place a queen on position above the solution found for i-1 column.