Beginner's guide to Solving the N-Queens problem using backtracking method

This post serves as a good introduction to the backtracking method which is used widely in various types of problems to find a possible solution. Here we'll be seeing how to solve the classical N-Queens problem using backtracking method. Before starting off, the prerequisites for understanding this post is:

- Java programming (Or at least be able to understand an algorithm)
- 2-Dimensional arrays
- Recursions.
- **N-Queens problem** (What it is. click on it to open Wikipedia page if you don't know.)

Backtracking:

- Backtracking is a form of recursion.
- A Recursion is a function/method calling itself.
- An example of recursion is the following function:

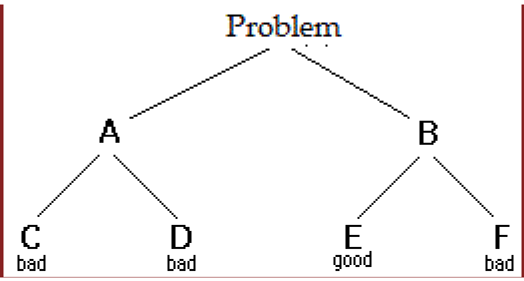
```
void printNumbers(int n) {
   if (n <= 0)
     return;
   printNumbers(n-1);
   printf("%d ", n);
 }
```

view raw **print numbers-recursion function** hosted with ❤ by **GitHub**

- The above function would print the integers from 1 to n when n is > 0
- If you study carefully the above function(or any recursive function), it has three main characteristics. They are:

  1. The problem has a base case(also called terminating condition) which in this case is return when n <= 0.
  2. The problem is broken down into smaller problems of the same type. This is done in this function with the recursive call printNumbers(n-1);
  3. Finally, it must be made sure that the base case is reached before a stack overflow occurs. Try removing the base case(the if block). You'd end up with a stack overflow as one function calls the other endlessly.

- Now coming back to Backtracking, here we divide the problem into many subproblems and solve every subproblem till a possible solution is obtained.
- For example, consider a problem which is divided into two subproblems A and B. A is again subdivided into C and D (and) B is subdivided into E and F.
- Let's say E is the required solution. The following steps are followed in a backtracking algorithm.



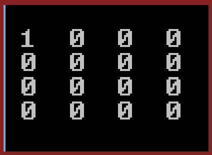
1. The problem is divided into A and B.
2. A is again subdivided into C and D.
3. The subproblem C is solved but the solution is not obtained,
4. so we backtrack and solve D.
5. Even then the solution is not obtained, so we backtrack and go back to B.
6. B is subdivided into E and F
7. E is solved and the required solution is obtained.
8. The search for a possible solution is terminated.

- This is a simple explanation of a backtracking algorithm. Now let us see how backtracking helps us solve N-Queens problem.

N-Queens problem:

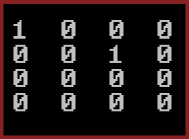
Before building an algorithm, let us solve the problem pictorially, Here's a picture of a solution to 4 queens problem using backtracking:

- First, we place queen at (1,1)



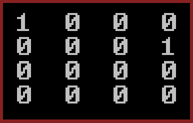
```
1   0   0   0
0   0   0   0
0   0   0   0
0   0   0   0
```

- Now the second queen cannot be placed in columns 1 and 2 as those positions can be attacked by the first queen.
- So we place queen two initially at (2,3)



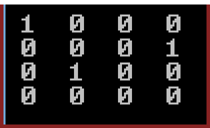
```
1   0   0   0
0   0   1   0
0   0   0   0
0   0   0   0
```

- Now when we try to place a queen in the third row, No possible location is present as all locations can be attacked by either first queen or second queen.
- So we backtrack and change the second queen's positon to (2,4)



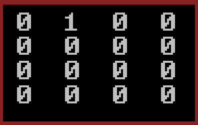
```
1   0   0   0
0   0   0   1
0   0   0   0
0   0   0   0
```

- Now while placing the third queen there is only one possible location which is (3,2) so we place the third queen at (3,2)



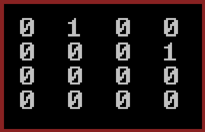
```
1   0   0   0
0   0   0   1
0   1   0   0
0   0   0   0
```

- Once again we end up with no possible locations for placing the next queen. So we backtrack, but there are no alternative positions even for the third and second queens. So we backtrack and change the position of the first queen as (1,2)



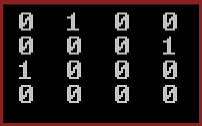
```
0   1   0   0
0   0   0   0
0   0   0   0
0   0   0   0
```

- Now for placing the second queen, we have only on choice which is (2,4)



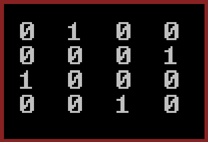
```
0   1   0   0
0   0   0   1
0   0   0   0
0   0   0   0
```

- Similarly, for placing the third queen, we have only one possible location (3,1)



```
0   1   0   0
0   0   0   1
1   0   0   0
0   0   0   0
```

- Finally, we have one possible location for placing the fourth queen which is (4,3)



```
0   1   0   0
0   0   0   1
1   0   0   0
0   0   1   0
```

- Thus a possible solution to the N-Queens problem is obtained and the algorithm terminates.