Analysis & Design of Algorithms

Semester III 2018-19

Lab - 5

Topics: To solve the problem of N-Queen using Backtracking

INTRODUCTION

In this lab we would be evaluating the performance of sorting algorithm. Performance of insertion sort, selection sort bubble sort and heap sort would be carried out.

Backtracking is a systematic method to iterate through all the possible configurations of a search space. It is a general algorithm/technique which must be customized for each individual application.

In the general case, we will model our solution as a vector $a = (a_1, a_2, a_n)$, where each element a_i is selected from a finite ordered set S_i. Such a vector might represent an arrangement where a_i contains the i^{th} element of the permutation. Or the vector might represent a given subset S, where a_i is true if and only if the i^{th} element of the universe is in S.

At each step in the backtracking algorithm, we start from a given partial solution, say, $a = (a_1, a_2, a_k)$, and try to extend it by adding another element at the end. After extending it, we must test whether what we have so far is a solution.

If not, we must then check whether the partial solution is still potentially extendible to some complete solution. If so, recur and continue. If not, we delete the last element from a and try another possibility for that position, if one exists.

```
Backtrack(a, k)
if a is a solution, print(a)
else {
  k = k + 1
  compute S_k
  while S_k \neq 0 do
    a_k = an element in S_k
    S_k = S_k - a_k
Backtrack(a, k)
```

EXERCISE : PROBLEM

1. Implement N queen program using backtracking algorithm. And compute the time taken for N=2, 3, 4 .....N. Put the analysis in tabular format.

PSEUDO CODE

N-Queen Problem

```
is_attacked( x, y, board[][], N)
  //checking for row and column
  if any cell in xth row is 1
    return true
  if any cell in yth column is 1
    return true

//checking for diagonals
  if any cell (p, q) having p+q = x+y is 1
    return true
  if any cell (p, q) having p-q = x-y is 1
    return true
return false

N-Queens( board[][], N )
  if N is 0                    //All queens have been placed
    return true
  for i = 1 to N {
    for j = 1 to N {
      if is_attacked(i, j, board, N) is true
        skip it and move to next cell
      board[i][j] = 1          //Place current queen at cell (i,j)
      if N-Queens( board, N-1) is true    // Solve subproblem
        return true                  // if solution is found return true
      board[i][j] = 0          /* if solution is not found undo whatever changes
                  were made i.e., remove  current queen from (i,j)*/
          }
      }
  return false
```

EXAMPLE:

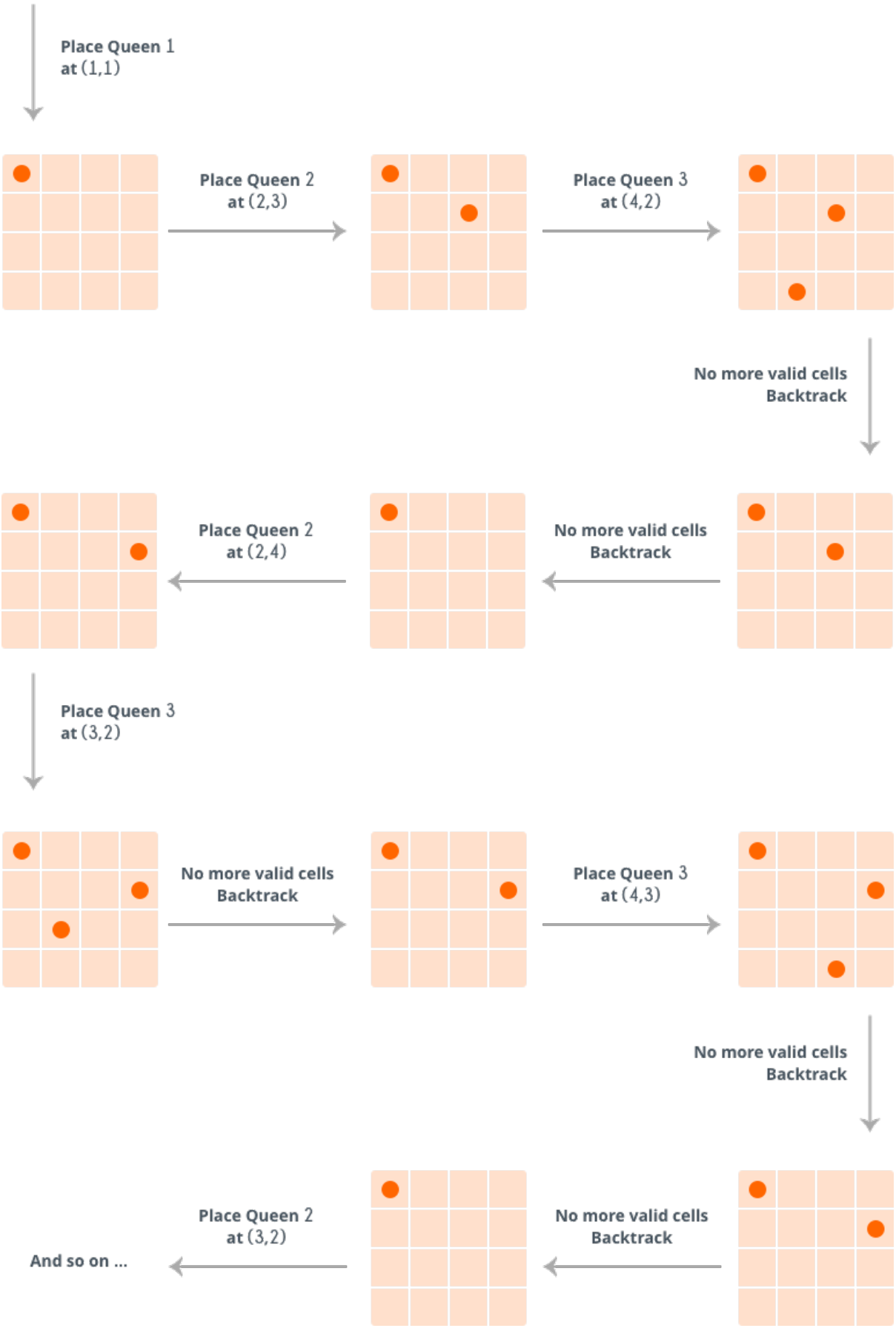

Figure 1: N-Queen Problem