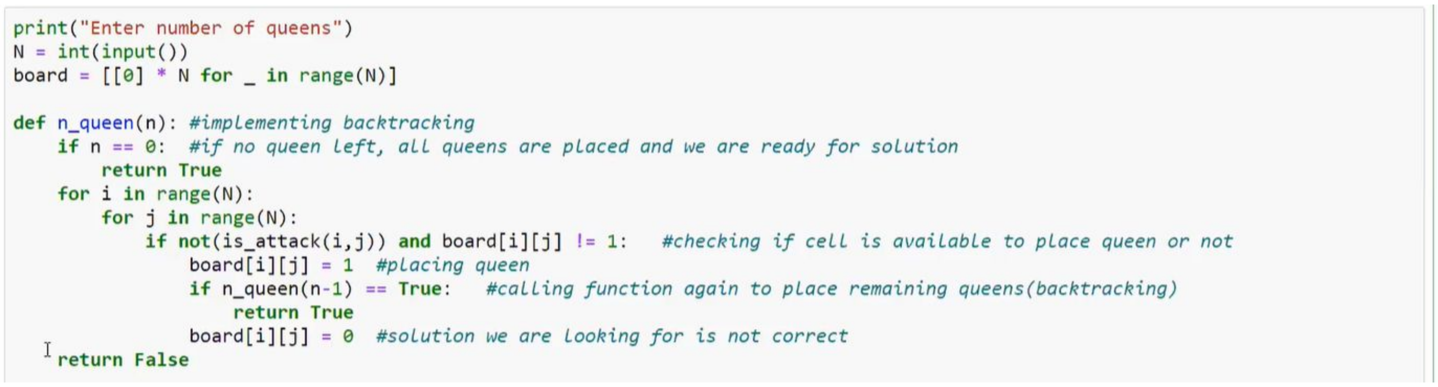
Date : 23rd - 09 - 2020
Morning Session : 9am – 11.00 PM
By ~ Rohan Kumar

Topics: N queen Problem & Sorting

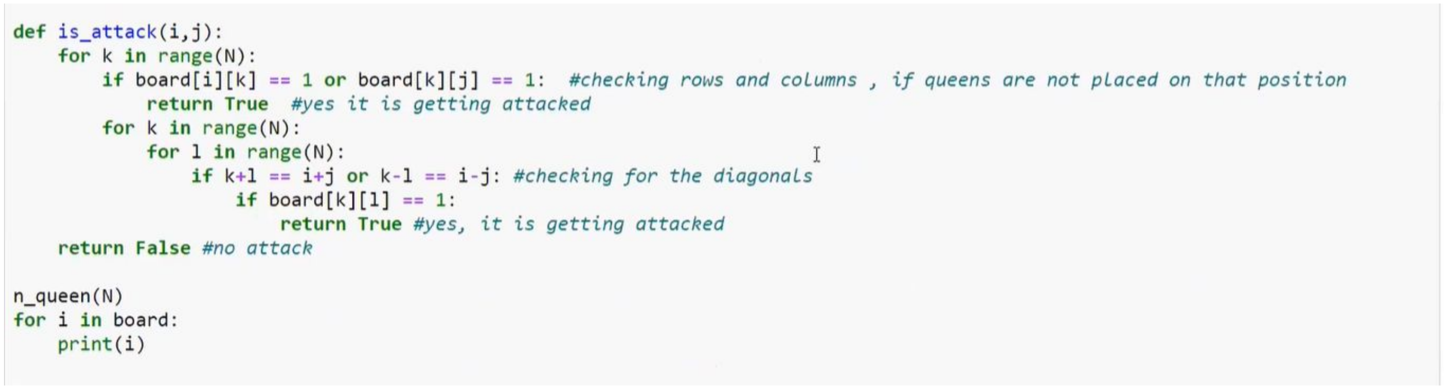
N Queen :

The N Queen is the problem of placing N chess queens on an N×N chessboard so that no two queens attack each other.

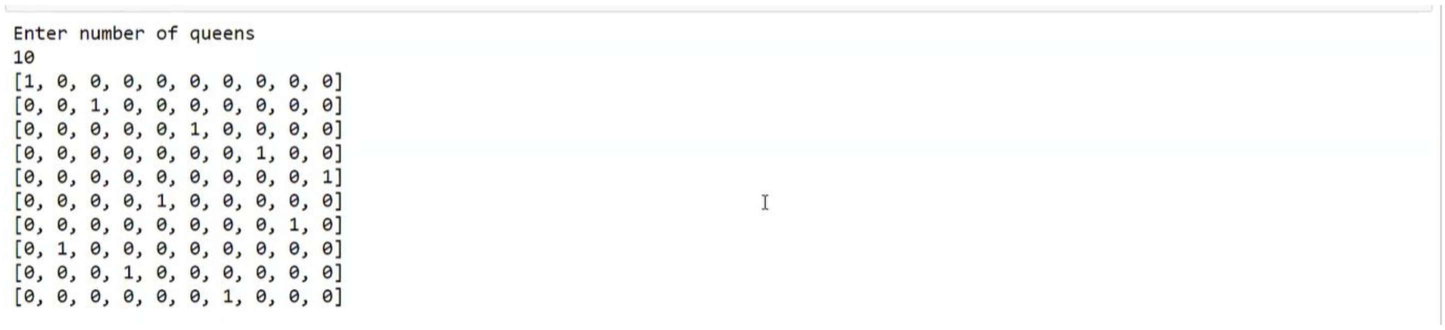
Problem: place N Queens in such a way that no Queen attack each other (attack means should not be in the same row , column, diagonal)



```
print("Enter number of queens")
N = int(input())
board = [[0] * N for _ in range(N)]

def n_queen(n): #implementing backtracking
    if n == 0:  #if no queen left, all queens are placed and we are ready for solution
        return True
    for i in range(N):
        for j in range(N):
            if not(is_attack(i,j)) and board[i][j] != 1:   #checking if cell is available to place queen or not
                board[i][j] = 1  #placing queen
                if n_queen(n-1) == True:   #calling function again to place remaining queens(backtracking)
                    return True
                board[i][j] = 0  #solution we are looking for is not correct
    return False
```



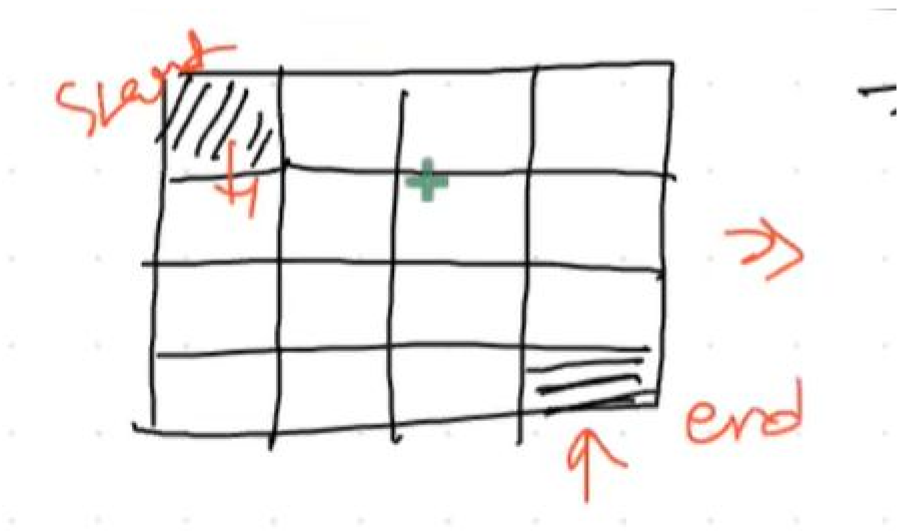
```
def is_attack(i,j):
    for k in range(N):
        if board[i][k] == 1 or board[k][j] == 1:  #checking rows and columns , if queens are not placed on that position
            return True  #yes it is getting attacked
        for k in range(N):
            for l in range(N):
                if k+l == i+j or k-l == i-j: #checking for the diagonals
                    if board[k][l] == 1:
                        return True #yes, it is getting attacked
    return False #no attack

n_queen(N)
for i in board:
    print(i)
```

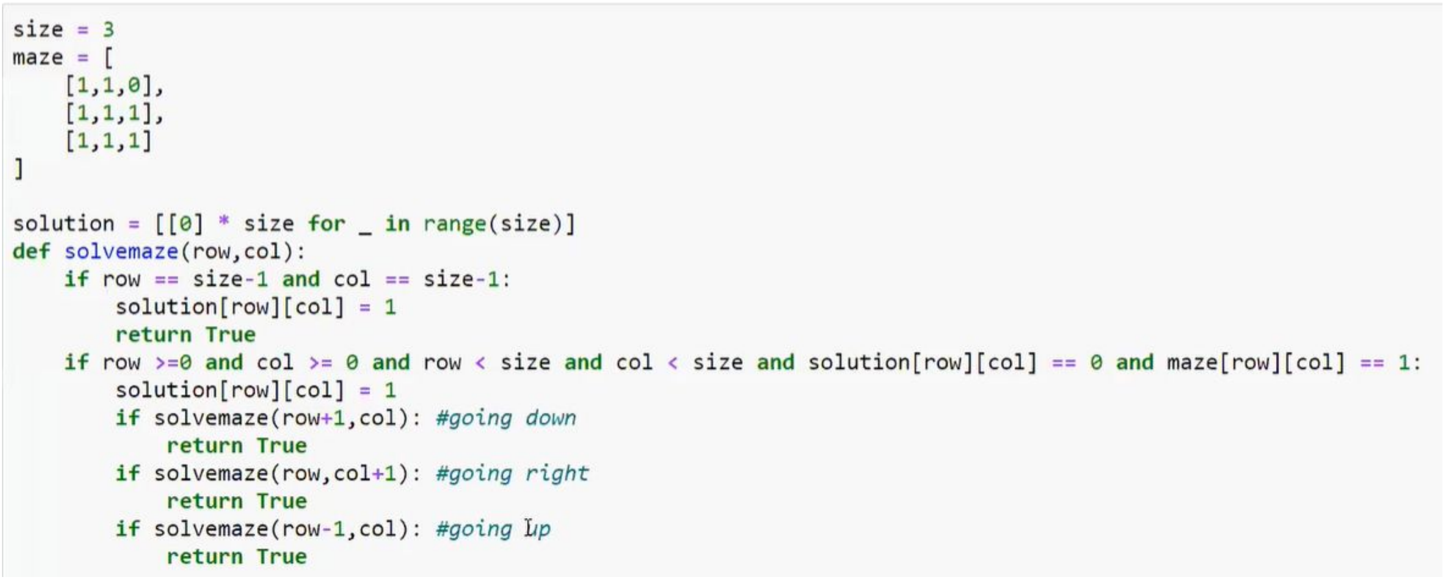


```
Enter number of queens
10
[1, 0, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 1, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 1, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 1, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 0, 1]
[0, 0, 0, 0, 1, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 0, 0, 1, 0]
[0, 1, 0, 0, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 1, 0, 0, 0, 0, 0, 0]
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0]
```

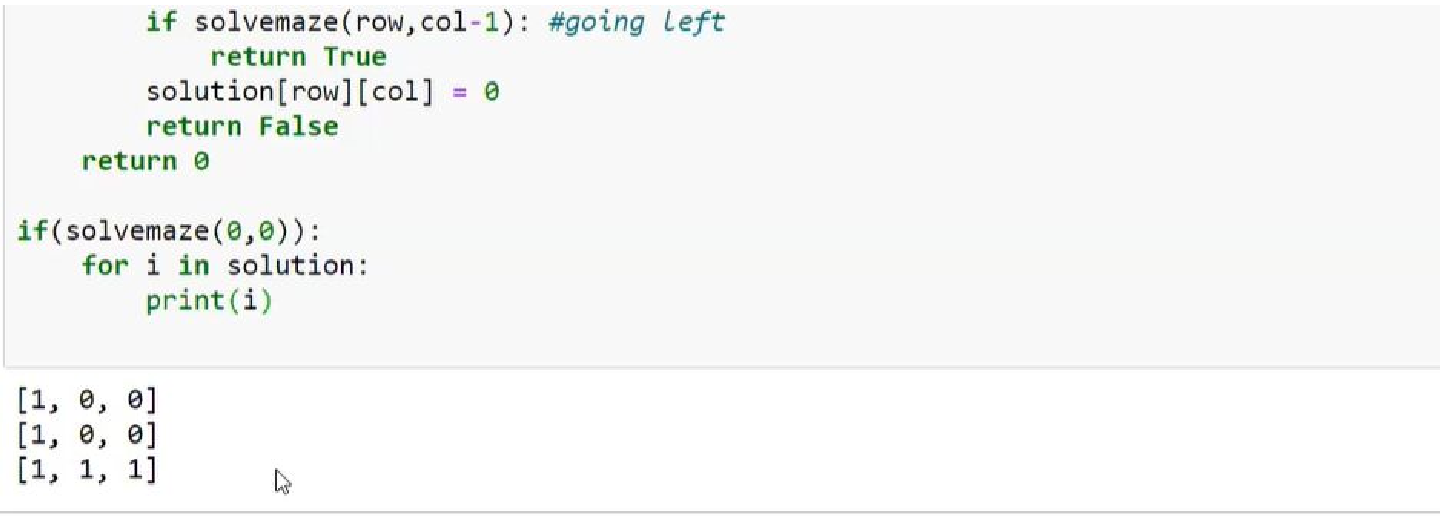
RAT IN A MAZE PROBLEM:

A Maze is given as N*N binary matrix of blocks where source block is the upper left most block i.e., maze[0][0] and destination block is lower rightmost block i.e., maze[N-1][N-1]. A rat starts from source and has to reach the destination. The rat can move only in two directions: forward and down.
In the maze matrix, 0 means the block is a dead end and 1 means the block can be used in the path from source to destination.





```
size = 3
maze = [
    [1,1,0],
    [1,1,1],
    [1,1,1]
]

solution = [[0] * size for _ in range(size)]
def solvemaze(row,col):
    if row == size-1 and col == size-1:
        solution[row][col] = 1
        return True
    if row >=0 and col >= 0 and row < size and col < size and solution[row][col] == 0 and maze[row][col] == 1:
        solution[row][col] = 1
        if solvemaze(row+1,col): #going down
            return True
        if solvemaze(row,col+1): #going right
            return True
        if solvemaze(row-1,col): #going up
            return True
```



```
        if solvemaze(row,col-1): #going left
            return True
        solution[row][col] = 0
        return False
    return 0

if(solvemaze(0,0)):
    for i in solution:
        print(i)
```

```
[1, 0, 0]
[1, 0, 0]
[1, 1, 1]
```

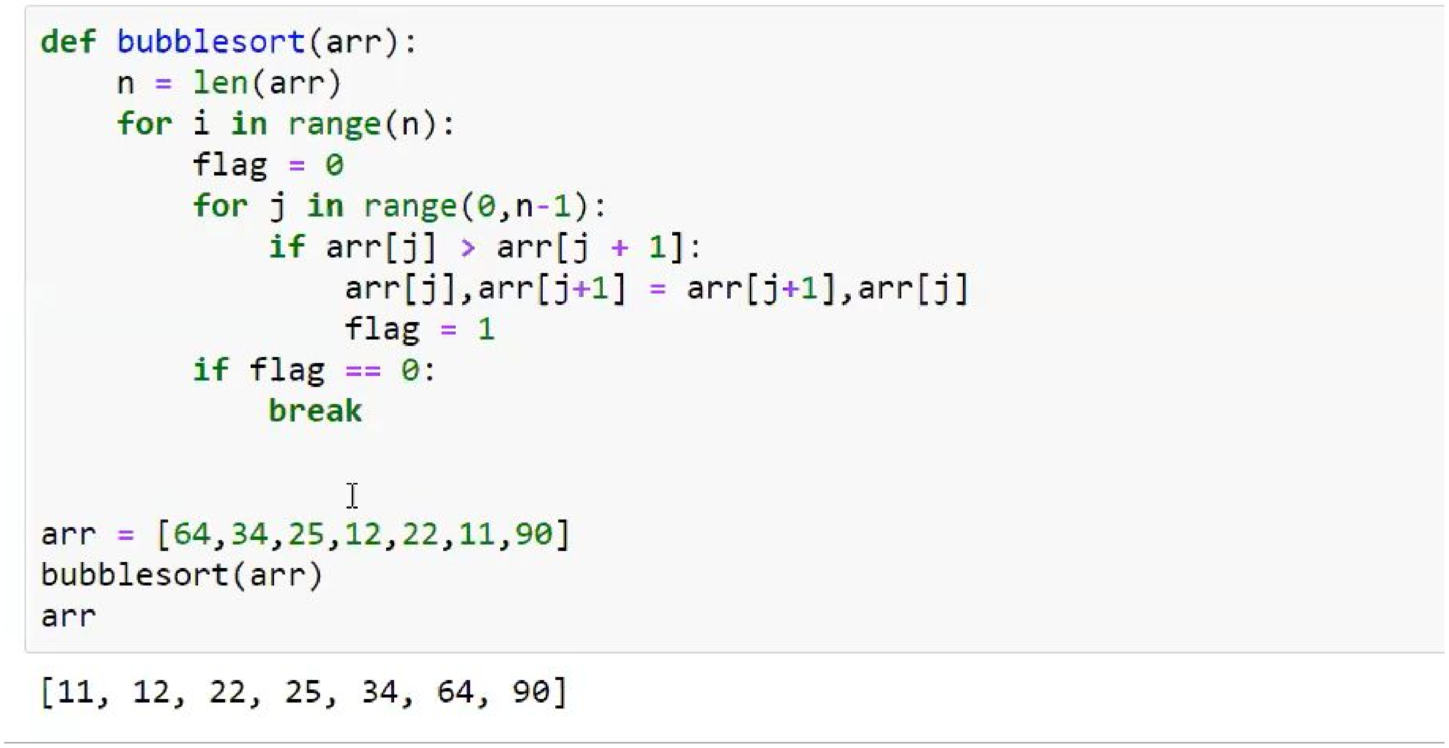
Sorting: **Sorting** refers to arranging data in a particular format. Sorting algorithm specifies the way to arrange data in a particular order. Most common orders are in numerical or lexicographical order.

The importance of sorting lies in the fact that data searching can be optimized to a very high level, if data is stored in a sorted manner. Sorting is also used to represent data in more readable formats. Below we see five such implementations of sorting in python.

- Quick Sort
- Bubble sort
- Merge Sort
- Selection sort
- insertion sort

Bubble Sort:

It is a comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order.



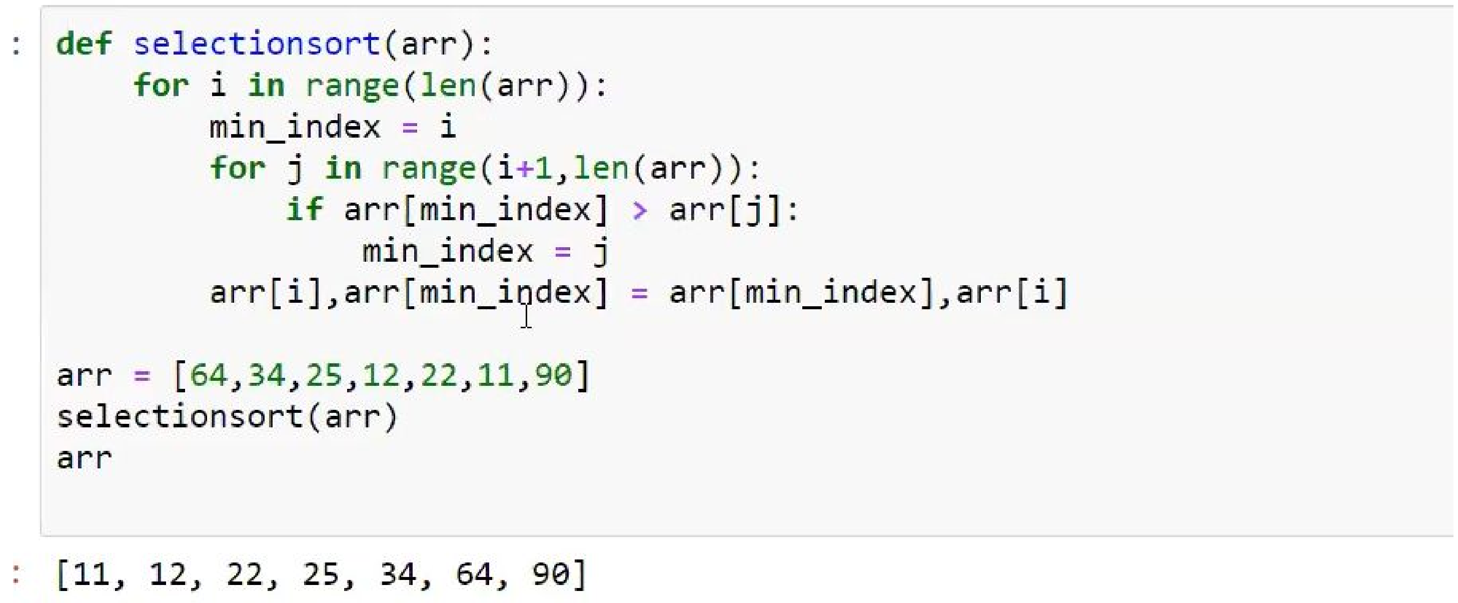
```
def bubblesort(arr):
    n = len(arr)
    for i in range(n):
        flag = 0
        for j in range(0,n-1):
            if arr[j] > arr[j + 1]:
                arr[j],arr[j+1] = arr[j+1],arr[j]
                flag = 1
        if flag == 0:
            break


arr = [64,34,25,12,22,11,90]
bubblesort(arr)
arr
```

```
[11, 12, 22, 25, 34, 64, 90]
```

Time complexity of Bubble Sort: O(n^2)

Selection Sort :

In selection sort we start by finding the minimum value in a given list and move it to a sorted list. Then we repeat the process for each of the remaining elements in the unsorted list. The next element entering the sorted list is compared with the existing elements and placed at its correct position. So at the end all the elements from the unsorted list are sorted.

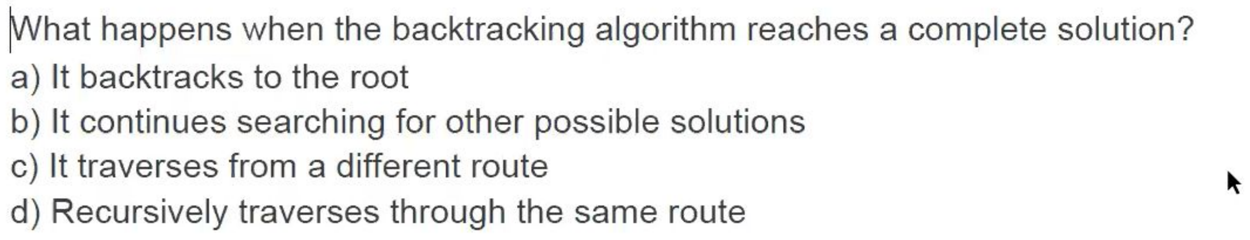


```python
def selectionsort(arr):
    for i in range(len(arr)):
        min_index = i
        for j in range(i+1,len(arr)):
            if arr[min_index] > arr[j]:
                min_index = j
        arr[i],arr[min_index] = arr[min_index],arr[i]

arr = [64,34,25,12,22,11,90]
selectionsort(arr)
arr
```

```
[11, 12, 22, 25, 34, 64, 90]
```

Time complexity of Bubble Sort: O(n^2)

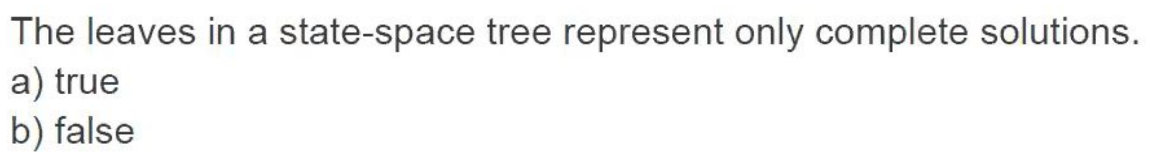
MCQ 1:



What happens when the backtracking algorithm reaches a complete solution?
a) It backtracks to the root
b) It continues searching for other possible solutions
c) It traverses from a different route
d) Recursively traverses through the same route

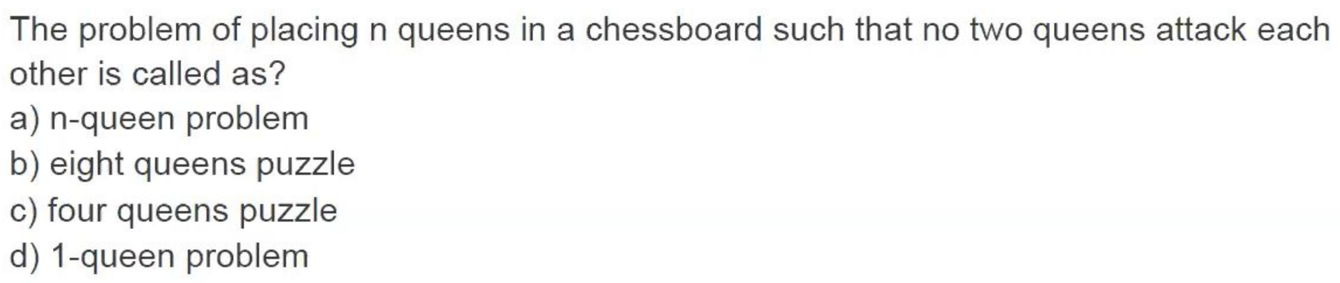
Answer: B, It continues searching for other possible solutions.

MCQ 2:


The leaves in a state-space tree represent only complete solutions.
a) true
b) false

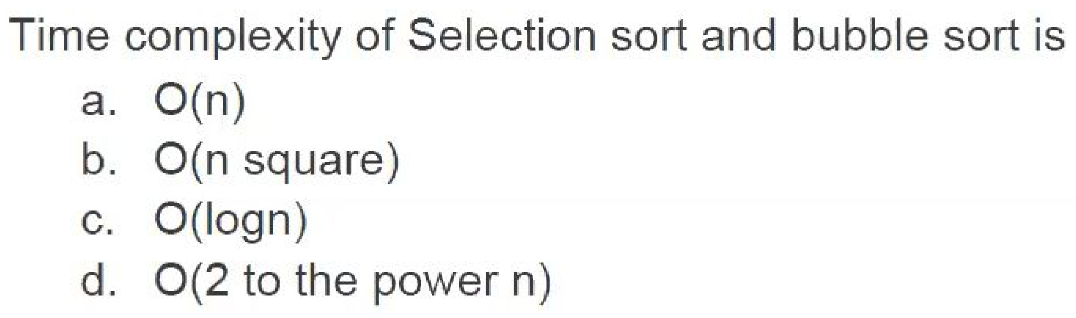
Answer: B, False

MCQ 3:


The problem of placing n queens in a chessboard such that no two queens attack each other is called as?
a) n-queen problem
b) eight queens puzzle
c) four queens puzzle
d) 1-queen problem

Answer: A, N-Queen Problem

MCQ 4:


Time complexity of Selection sort and bubble sort is

a. O(n)
b. O(n square)
c. O(logn)
d. O(2 to the power n)

Answer: B, O(n^2)