Cost analysis of the algorithm (c_4 → problem_1.py)

```
class NQueensSolver:
    def __init__(self, n):
        self.n = n                                           #c1=1
        self.board = [-1]*n                                  #c2=n
        self.columns = [False]*n                             #c3=n
        self.majorDiagonals = [False]*(2*n - 1)              #c4=2(n-1)
        self.minorDiagonals = [False]*(2*n - 1)              #c5=2(n-1)
        self.solutions = []                                  #c6=1

    def insertQueens(self, row):
        if row == self.n:                                   #c7=n!
            self.solutions.append(self.board[:])            #c8=S.n
            return                                          #c9=S

        for col in range(self.n):                           #c10=C.n
            majorIndex = row - col + (self.n - 1)           #c11=C.n
            minorIndex = row + col                          #c12=C.n

            if not self.columns[col] and not
self.majorDiagonals[majorIndex] and not
self.minorDiagonals[minorIndex]:                            #c13=3.C.n
                self.board[row] = col                       #c14=C.n
                self.columns[col] = True                    #c15=C.n
                self.majorDiagonals[majorIndex] = True      #c16=C.n
                self.minorDiagonals[minorIndex] = True      #c17=C.n

                self.insertQueens(row + 1)                  #c18=C

                self.board[row] = -1                        #c19=C.n
                self.columns[col] = False                   #c20=C.n
                self.majorDiagonals[majorIndex] = False #c21=C.n
                self.minorDiagonals[minorIndex] = False  #c22=C.n

    def buildBoard(self,board):
        solutionBoard = []                                  #c23=1
        for i in range(self.n):                             #c24=n
            rowList = ['x'] * self.n                        #c25=n.n
            qCol = board[i]                                 #c26=n
            rowList[qCol] = 'Q'                             #c27=n
            solutionBoard.append("".join(rowList))          #c28=n.n
    totalQueens = sum(row.count('Q') for row in solutionBoard) #c29=n.n
        return solutionBoard, totalQueens                   #c30=1
```

- Basic operation:
  $C_{13} = 3.(C \approx n!).n$

- Time complexity calculation:
  $T(n) = 3.(C \approx n!).n$
  $T(n) = 3n.n!$
  $T(n) = n.n!$
  $T(n) \in O(n.n!)$

- Solving the recurrence:
  $T(n) = n.T(n-1) + O(n),\ n >= 0,\ T(0) = 1$
  $T(1) = 1.T(1-1) + O(1) = 1.T(0) + 1 = 1.1 + 1 = 2$
  $T(2) = 2.T(2-1) + O(2) = 2.T(1) + 2 = 2.2 + 2 = 6$
  $T(3) = 3.T(3-1) + O(3) = 3.T(2) + 3 = 3.6 + 3 = 21$
  $T(4) = 4.T(4-1) + O(4) = 4.T(3) + 4 = 4.21 + 4 = 88$

$$T(n) = \sum_{i=1}^{n=1} (i^2 - i) + n$$