Grupo 14

```
!pip install ortools
```

Problema 2: Sudoku

Pretende-se criar uma definição do jogo “Sudoku” generalizado para a dimensão N. Sendo que, o objetivo do Sudoku é preencher uma grelha de N^2*N^2,com inteiros positivos no intervalo 1 até N^2. Considerando, a variável inteira $m_{c,l}$ como a matriz de tamanho $c \times l$, iremos analisar os requisitos deste jogo:

$Limitaçõe s$

- Cada inteiro, no intervalo de 1 até N^2, ocorre só uma vez em cada coluna.

$$\forall_{l<N} \quad \sum_{c<N-1} m_{c,l} - \sum_{c<c1<N} m_{c1,l} \neq 0$$

- Cada inteiro, no intervalo de 1 até N^2, ocorre só uma vez em cada linha.

$$\forall_{c<N^2} \quad \sum_{l<N-1} m_{c,l} = 1$$

- Cada inteiro, no intervalo de 1 até N^2, ocorre só uma vez em cada secção N * N. (Considerando Ca, uma "casa" do sudoku)

$$\forall_{0<x<l} : l/x, \quad \forall_{0<y<c} : c/y, \quad Ca_{y,x} - (\forall_{x<x1<l} : l/x_1, \quad \forall_{y<y1<c} : c/y_1, \quad Ca_{y_1,x_1}) \neq 0$$

```
def trabalho_1_sudoku():
  n = [3,4,5,6]
  alfa = [0.0,0.2,0.4,0.6]

  print(" N \ Alfa|   0.0   |   0.2   |   0.4   |   0.6   |")
  print("-------------------------------------------------")

  n_tests = 3

  for n_value in n:
    print ("    "+ str(n_value) + "    |", end = "")
    for alfa_value in alfa:
      time = 0
      # Repetir cada teste, n_tests vezes
      for _ in range(n_tests):
        res = benchmark_sudoku_matrix(n_value, alfa_value, False)
        time = time + res
      print(" %.5f |" % (time/n_tests),  end = "")
      time = 0
    print("\n-------------------------------------------------")
```

```
trabalho_1_sudoku()
```

N \ Alfa	0.0	0.2	0.4	0.6
3	0.06180	0.02162	0.00423	0.00240
4	0.46834	0.12878	0.05535	0.00778
5	2.64160	0.54487	0.24516	0.00938
6	8.14690	1.93781	3.21002	0.02393

```
# Importar biblioteca
from ortools.sat.python import cp_model
import time

def benchmark_sudoku_matrix (n, alfa, print_matrix):

  # Criar a instância do solver
  solver = cp_model.CpSolver()
  model = cp_model.CpModel()

  region_size = n
  matrix_size = region_size * region_size

  # Inicializar a matriz
  inital_matrix = {}
  init_sudoku_matrix(model,inital_matrix,matrix_size, region_size, alfa)

  if print_matrix:
    print("Inital Matrix:")
    print_sudoku_matrix(inital_matrix, region_size)

  # Definir as variáveis
  matrix = {}
  for i in range(matrix_size):
      for j in range(matrix_size):
          if (inital_matrix[i,j]):
              matrix[i,j] = model.NewConstant(inital_matrix[i,j])
          else:
              matrix[i,j] = model.NewIntVar(1, matrix_size, 'x[{},{}]'.format(i,j)

  # Definir Constraints
  define_constraints(model,matrix, matrix_size, region_size)

  #Solve
  start = time.time()
  status = solver.Solve(model)
  end = time.time()

  if print_matrix:
    print("Solved Matrix:")
    print_sudoku_matrix_solver(solver, matrix, region_size)
```

```
  #if status == cp_model.OPTIMAL:
    #print("Solved Matrix:")
    #print_sudoku_matrix_solver(solver, matrix, region_size)

  return (end - start)

benchmark_sudoku_matrix (3, 0.0, True)
```

```
Inital Matrix:
- - - - - - - - - - - - - - - - - - - - - -
| 00 00 00 | 00 00 00 | 00 00 00 |
| 00 00 00 | 00 00 00 | 00 00 00 |
| 00 00 00 | 00 00 00 | 00 00 00 |
- - - - - - - - - - - - - - - - - - - - - -
| 00 00 00 | 00 00 00 | 00 00 00 |
| 00 00 00 | 00 00 00 | 00 00 00 |
| 00 00 00 | 00 00 00 | 00 00 00 |
- - - - - - - - - - - - - - - - - - - - - -
| 00 00 00 | 00 00 00 | 00 00 00 |
| 00 00 00 | 00 00 00 | 00 00 00 |
| 00 00 00 | 00 00 00 | 00 00 00 |
- - - - - - - - - - - - - - - - - - - - - -
Solved Matrix:
- - - - - - - - - - - - - - - - - - - - - -
| 01 03 05 | 04 07 06 | 02 09 08 |
| 02 06 08 | 05 01 09 | 07 04 03 |
| 04 07 09 | 02 03 08 | 05 01 06 |
- - - - - - - - - - - - - - - - - - - - - -
| 05 04 02 | 01 06 03 | 08 07 09 |
| 07 01 03 | 08 09 02 | 04 06 05 |
| 08 09 06 | 07 04 05 | 01 03 02 |
- - - - - - - - - - - - - - - - - - - - - -
| 06 05 04 | 03 02 01 | 09 08 07 |
| 03 02 01 | 09 08 07 | 06 05 04 |
| 09 08 07 | 06 05 04 | 03 02 01 |
- - - - - - - - - - - - - - - - - - - - - -
0.09252524375915527
```

```
def define_constraints (model, matrix, matrix_size, region_size):
  # Adicionar restrições
  # Declarar que em cada linha, os elementos têm que ser todos diferentes.
  for i in range(matrix_size):
      model.AddAllDifferent([matrix[i, j] for j in range(matrix_size)])

  # Declarar que em cada coluna, os elementos têm que ser todos diferentes.
  for j in range(matrix_size):
      model.AddAllDifferent([matrix[i, j] for i in range(matrix_size)])

  # Declarar que em cada região, os elementos têm que ser todos diferentes.
  for row_idx in range(0, matrix_size, region_size):
      for col_idx in range(0, matrix_size, region_size):
          model.AddAllDifferent([matrix[row_idx + i, j] for j in range(col_idx, (co

import random
```

```
def init_sudoku_matrix (model, matrix, matrix_size, region_size, alfa):

  solver1 = cp_model.CpSolver()
  #solver1.parameters.random_seed = 10
  model1 = cp_model.CpModel()

  matrix1 = {}

  for i in range(matrix_size):
      for j in range(matrix_size):
              matrix1[i, j] = model1.NewIntVar(1, matrix_size, 'x[{},{}]'.format(i,

  # Introduzir o random na matriz
  i_random = random.randint(0, matrix_size - 1)
  j_random = random.randint(0, matrix_size - 1)
  value_random = random.randint(1, matrix_size)
  matrix1[i_random,j_random] = model1.NewConstant(value_random)

  define_constraints(model1,matrix1, matrix_size, region_size)

  status1 = solver1.Solve(model1) # Passo 5

  #Escolher a casa para preencher

  fill_spots = int(alfa * matrix_size * matrix_size)

  fill_spots_array = []
  while fill_spots > 0:
    x, y = random.randint(0, matrix_size-1), random.randint(0, matrix_size-1)
    if (x,y) not in fill_spots_array:
      fill_spots_array.append((x,y))
      fill_spots = fill_spots - 1

  for i in range(matrix_size):
    for j in range(matrix_size):
      if (i,j) in fill_spots_array:
          matrix[i,j] = solver1.Value(matrix1[i, j])
      else:
          matrix[i,j] = 0


def print_sudoku_matrix_solver (result, matrix, region_size):
  matrix_size = region_size * region_size
  track_region = region_size
  track_region2 = region_size - 1
  for i in range(int(region_size + 1.5 * matrix_size)+1):
    print("- ", end="")
  print("")
  for i in range(matrix_size):
    for j in range(matrix_size):
      if track_region == region_size:
        print("| ",end='')
      track_region -= 1
      if track_region == 0:
        track_region = region_size
```

```
      value = result.Value(matrix[i,j])
      if value < 10:
        print("0",end="")
      print(value, end=' ')
    print("| \n", end='')
    if track_region2 == region_size:
      for i in range(int(region_size + 1.5 * matrix_size)+1):
        print("- ", end="")
      print("")
    track_region2 -= 1
    if track_region2 == 0:
      track_region2 = region_size


def print_sudoku_matrix(matrix, region_size):
  matrix_size = region_size * region_size
  track_region = region_size
  track_region2 = region_size - 1
  for i in range(int(region_size + 1.5 * matrix_size)+1):
    print("- ", end="")
  print("")
  for i in range(matrix_size):
    for j in range(matrix_size):
      if track_region == region_size:
        print("| ",end='')
      track_region -= 1
      if track_region == 0:
        track_region = region_size
      value = matrix[i,j]
      if value < 10:
        print("0",end="")
      print(value, end=' ')
    print("| \n", end='')
    if track_region2 == region_size:
      for i in range(int(region_size + 1.5 * matrix_size)+1):
        print("- ", end="")
      print("")
    track_region2 -= 1
    if track_region2 == 0:
      track_region2 = region_size
```

Resultados e Conclusões Finais

Para os diferentes valores de N $\in$ {3,4,5,6} e alpha $\in$ {0.0,0.2,0.4,0.6}, obtivemos as seguintes soluções:

- N=3 e α =0.0:

```
Inital Matrix:
- - - - - - - - - - - - - - - - - -
| 00 00 00 | 00 00 00 | 00 00 00 |
| 00 00 00 | 00 00 00 | 00 00 00 |
| 00 00 00 | 00 00 00 | 00 00 00 |
- - - - - - - - - - - - - - - - - -
| 00 00 00 | 00 00 00 | 00 00 00 |
| 00 00 00 | 00 00 00 | 00 00 00 |
| 00 00 00 | 00 00 00 | 00 00 00 |
- - - - - - - - - - - - - - - - - -
| 00 00 00 | 00 00 00 | 00 00 00 |
| 00 00 00 | 00 00 00 | 00 00 00 |
| 00 00 00 | 00 00 00 | 00 00 00 |
- - - - - - - - - - - - - - - - - -

Solved Matrix:
- - - - - - - - - - - - - - - - - -
| 01 03 05 | 04 07 06 | 02 09 08 |
| 02 06 08 | 05 01 09 | 07 04 03 |
| 04 07 09 | 02 03 08 | 05 01 06 |
- - - - - - - - - - - - - - - - - -
| 05 04 02 | 01 06 03 | 08 07 09 |
| 07 01 03 | 08 09 02 | 04 06 05 |
| 08 09 06 | 07 04 05 | 01 03 02 |
- - - - - - - - - - - - - - - - - -
| 06 05 04 | 03 02 01 | 09 08 07 |
| 03 02 01 | 09 08 07 | 06 05 04 |
| 09 08 07 | 06 05 04 | 03 02 01 |
- - - - - - - - - - - - - - - - - -
0.09568047523498535
```

- N=4 e α =0.2:

```
Inital Matrix:
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 05 00 00 00 | 01 00 00 00 | 00 00 11 00 | 00 00 15 00 |
| 00 00 00 00 | 00 00 03 00 | 00 00 00 00 | 00 00 00 00 |
| 00 00 11 00 | 00 00 15 00 | 00 00 03 00 | 00 00 00 00 |
| 13 00 00 16 | 00 00 11 00 | 00 00 00 08 | 01 00 00 00 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 09 00 00 00 | 06 00 08 00 | 00 00 00 00 | 00 00 16 00 |
| 06 00 00 00 | 00 00 00 00 | 14 00 00 00 | 10 09 00 00 |
| 00 00 00 00 | 00 13 00 15 | 00 00 00 00 | 00 00 08 00 |
| 14 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 00 00 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 00 00 00 00 | 00 00 00 00 | 00 00 00 00 | 00 00 13 00 |
| 00 00 00 00 | 00 00 00 00 | 00 00 13 00 | 11 00 00 00 |
| 00 00 00 00 | 15 00 00 00 | 00 00 01 00 | 00 00 00 00 |
| 00 00 00 00 | 11 00 00 00 | 00 00 00 00 | 00 04 00 02 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 00 00 02 00 | 00 00 00 00 | 00 00 10 00 | 16 00 00 00 |
| 00 07 06 00 | 00 00 02 00 | 00 00 00 13 | 12 00 10 00 |
| 00 11 10 00 | 00 00 00 13 | 00 03 00 00 | 00 00 00 00 |
| 00 00 00 00 | 12 00 00 00 | 08 00 06 00 | 00 03 02 00 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
Solved Matrix:
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 05 14 04 03 | 01 02 09 16 | 10 13 11 12 | 06 08 15 07 |
| 10 09 08 12 | 13 06 03 04 | 07 01 05 15 | 02 14 11 16 |
| 02 01 11 07 | 05 08 15 10 | 06 14 03 16 | 09 13 04 12 |
| 13 06 15 16 | 14 12 11 07 | 02 04 09 08 | 01 10 05 03 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 09 02 03 11 | 06 14 08 12 | 01 10 15 07 | 13 05 16 04 |
| 06 12 13 08 | 02 01 04 03 | 14 05 16 11 | 10 09 07 15 |
| 01 05 07 04 | 10 13 16 15 | 03 09 12 06 | 14 02 08 11 |
| 14 10 16 15 | 07 09 05 11 | 13 08 04 02 | 03 01 12 06 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 03 04 01 02 | 09 05 07 06 | 11 12 08 10 | 15 16 13 14 |
| 07 08 05 06 | 03 04 01 02 | 15 16 13 14 | 11 12 09 10 |
| 11 13 09 10 | 15 16 12 14 | 05 02 01 04 | 07 06 03 08 |
| 15 16 12 14 | 11 10 13 08 | 09 06 07 03 | 05 04 01 02 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 04 03 02 01 | 08 07 06 05 | 12 11 10 09 | 16 15 14 13 |
| 08 07 06 05 | 04 03 02 01 | 16 15 14 13 | 12 11 10 09 |
| 12 11 10 09 | 16 15 14 13 | 04 03 02 01 | 08 07 06 05 |
| 16 15 14 13 | 12 11 10 09 | 08 07 06 05 | 04 03 02 01 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
0.25887393951416016
```

- N=5 e α =0.6:

```
Inital Matrix:
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 00 00 14 18 08 | 07 00 17 21 11 | 00 19 09 00 16 | 02 13 00 00 22 | 00 05 25 03 06 |
| 07 00 00 04 13 | 00 03 25 00 00 | 08 23 06 00 00 | 18 00 16 12 09 | 20 00 24 00 14 |
| 03 02 06 09 00 | 08 00 15 12 16 | 18 00 00 00 11 | 24 00 07 25 00 | 00 13 00 04 10 |
| 15 05 11 00 25 | 20 01 18 00 14 | 13 00 00 00 00 | 08 00 06 00 00 | 09 00 02 17 16 |
| 17 00 12 10 00 | 24 06 00 00 13 | 02 00 20 15 00 | 00 03 14 11 00 | 00 22 18 00 08 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 06 03 00 12 21 | 02 16 00 14 00 | 00 00 04 00 00 | 00 09 20 00 00 | 00 10 00 00 00 |
| 25 07 20 24 16 | 00 13 00 00 08 | 05 00 00 00 00 | 23 06 02 00 00 | 00 11 00 00 12 |
| 02 00 00 00 09 | 10 07 00 25 12 | 00 21 11 06 22 | 04 16 17 00 00 | 23 00 14 00 00 |
| 00 04 17 00 10 | 22 21 06 00 19 | 23 00 08 20 18 | 00 05 01 24 25 | 03 09 16 00 15 |
| 00 22 15 11 01 | 05 09 23 20 17 | 24 02 16 00 00 | 00 21 10 03 19 | 13 25 06 00 04 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 04 11 00 00 17 | 00 00 21 00 25 | 00 12 19 00 00 | 06 24 23 07 20 | 00 00 00 00 00 |
| 00 12 00 06 00 | 23 22 16 17 24 | 15 09 21 18 00 | 13 02 04 14 00 | 00 00 10 25 00 |
| 09 01 02 03 18 | 00 08 00 00 20 | 25 00 00 00 23 | 22 00 11 19 15 | 00 00 00 12 00 |
| 00 00 19 00 15 | 11 12 13 10 05 | 07 06 02 04 00 | 16 25 00 09 03 | 18 24 00 00 17 |
| 00 25 00 00 24 | 00 02 19 09 00 | 00 11 00 00 08 | 00 00 21 18 17 | 00 04 23 06 00 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 12 08 01 23 11 | 00 17 10 13 00 | 19 00 00 02 00 | 03 00 05 04 16 | 22 00 20 00 00 |
| 13 00 10 00 04 | 12 00 00 00 00 | 00 00 07 00 00 | 15 20 22 23 11 | 06 00 17 16 19 |
| 16 06 07 25 00 | 09 19 05 15 00 | 00 20 22 00 04 | 21 18 00 01 02 | 11 00 00 00 13 |
| 00 18 00 15 22 | 00 00 20 23 04 | 11 16 00 00 06 | 00 19 12 13 10 | 00 07 03 00 00 |
| 20 17 03 02 00 | 18 11 00 00 00 | 00 00 12 13 10 | 00 08 09 00 00 | 05 01 04 24 23 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 24 15 00 16 00 | 13 00 11 00 22 | 20 00 23 03 02 | 00 07 25 00 00 | 04 00 00 00 00 |
| 11 00 18 14 00 | 21 10 00 00 09 | 12 00 13 00 07 | 01 22 00 00 00 | 00 16 15 23 05 |
| 00 10 00 00 05 | 16 20 03 07 23 | 06 04 25 00 24 | 00 15 00 00 00 | 02 00 11 00 09 |
| 08 00 00 00 00 | 00 00 12 00 00 | 00 00 01 00 00 | 00 00 19 16 24 | 10 00 00 00 03 |
| 22 00 25 07 03 | 00 04 02 00 06 | 16 00 10 05 00 | 09 11 13 00 00 | 00 00 08 14 20 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
Solved Matrix:
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 01 24 14 18 08 | 07 23 17 21 11 | 04 19 09 10 16 | 02 13 15 20 22 | 12 05 25 03 06 |
| 07 21 22 04 13 | 19 03 25 02 10 | 08 23 06 17 05 | 18 01 16 12 09 | 20 15 24 11 14 |
| 03 02 06 09 20 | 08 05 15 12 16 | 18 22 14 21 11 | 24 17 07 25 23 | 01 13 19 04 10 |
| 15 05 11 19 25 | 20 01 18 22 14 | 13 03 24 07 12 | 08 04 06 10 21 | 09 23 02 17 16 |
| 17 16 12 10 23 | 24 06 09 04 13 | 02 01 20 15 25 | 19 03 14 11 05 | 07 22 18 21 08 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 06 03 23 12 21 | 02 16 24 14 15 | 01 13 04 25 19 | 11 09 20 08 18 | 17 10 05 07 22 |
| 25 07 20 24 16 | 03 13 04 18 08 | 05 10 15 09 17 | 23 06 02 22 14 | 21 11 01 19 12 |
| 02 19 05 08 09 | 10 07 01 25 12 | 03 21 11 06 22 | 04 16 17 15 13 | 23 20 14 18 24 |
| 14 04 17 13 10 | 22 21 06 11 19 | 23 07 08 20 18 | 12 05 01 24 25 | 03 09 16 02 15 |
| 18 22 15 11 01 | 05 09 23 20 17 | 24 02 16 12 14 | 07 21 10 03 19 | 13 25 06 08 04 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 04 11 16 22 17 | 15 18 21 03 25 | 10 12 19 14 13 | 06 24 23 07 20 | 08 02 09 05 01 |
| 05 12 08 06 07 | 23 22 16 17 24 | 15 09 21 18 20 | 13 02 04 14 01 | 19 03 10 25 11 |
| 09 01 02 03 18 | 04 08 07 06 20 | 25 05 17 24 23 | 22 10 11 19 15 | 16 14 13 12 21 |
| 23 14 19 21 15 | 11 12 13 10 05 | 07 06 02 04 01 | 16 25 08 09 03 | 18 24 22 20 17 |
| 10 25 13 20 24 | 14 02 19 09 01 | 22 11 03 16 08 | 05 12 21 18 17 | 15 04 23 06 07 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 12 08 01 23 11 | 06 17 10 13 07 | 19 24 18 02 09 | 03 14 05 04 16 | 22 21 20 15 25 |
| 13 09 10 05 04 | 12 24 14 08 02 | 21 25 07 01 03 | 15 20 22 23 11 | 06 18 17 16 19 |
| 16 06 07 25 14 | 09 19 05 15 03 | 17 20 22 23 04 | 21 18 24 01 02 | 11 08 12 10 13 |
| 21 18 24 15 22 | 01 25 20 23 04 | 11 16 05 08 06 | 17 19 12 13 10 | 14 07 03 09 02 |
| 20 17 03 02 19 | 18 11 22 16 21 | 14 15 12 13 10 | 25 08 09 06 07 | 05 01 04 24 23 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
| 24 15 09 16 12 | 13 14 11 19 22 | 20 08 23 03 02 | 10 07 25 05 06 | 04 17 21 01 18 |
| 11 20 18 14 06 | 21 10 08 24 09 | 12 17 13 19 07 | 01 22 03 02 04 | 25 16 15 23 05 |
| 19 10 21 01 05 | 16 20 03 07 23 | 06 04 25 22 24 | 14 15 18 17 08 | 02 12 11 13 09 |
| 08 13 04 17 02 | 25 15 12 05 18 | 09 14 01 11 21 | 20 23 19 16 24 | 10 06 07 22 03 |
| 22 23 25 07 03 | 17 04 02 01 06 | 16 18 10 05 15 | 09 11 13 21 12 | 24 19 08 14 20 |
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
0.01798844337463379
```

- N=6 e α =0.8:

Inital Matrix:

00 30 17 26 24 19	00 04 00 22 29 00	00 12 20 14 21 35	15 27 08 00 13 07	03 34 33 00 05 00	25 06 02 09 18 11
15 06 05 00 16 00	00 14 00 19 13 12	08 24 30 07 32 28	17 00 22 26 00 29	02 23 18 11 09 31	01 33 35 00 34 36
35 11 07 25 12 08	05 26 18 00 24 23	29 04 31 34 06 27	19 00 02 36 00 28	15 14 01 20 21 17	10 03 16 22 32 13
34 31 18 00 22 29	00 36 30 03 15 17	02 26 11 05 01 13	23 00 21 00 25 04	32 24 28 00 06 35	00 07 27 19 14 08
00 23 00 36 00 00	08 02 07 27 06 16	00 00 09 22 18 03	35 00 31 11 34 00	29 12 13 04 25 00	00 17 24 00 15 26
04 02 00 20 00 13	34 11 01 32 35 09	15 16 17 36 25 33	06 05 03 18 00 24	26 08 07 22 19 27	30 23 21 29 31 28
17 20 01 31 13 33	07 25 32 15 10 21	35 00 04 19 00 34	22 03 11 24 00 05	14 27 26 16 00 29	02 00 00 06 08 23
22 10 35 16 07 00	27 05 33 36 19 02	32 00 00 12 28 26	34 08 23 15 29 06	21 30 24 13 00 18	04 09 17 14 00 25
09 00 04 05 06 00	12 30 00 28 08 34	13 18 23 31 00 00	07 00 20 35 19 27	00 36 00 00 10 15	03 24 26 11 29 21
30 26 23 00 03 27	00 00 04 29 31 00	21 14 24 10 08 00	09 02 17 12 16 01	07 35 25 32 34 20	18 22 00 05 19 00
24 14 00 00 19 34	09 23 35 00 03 18	00 27 07 00 15 00	36 13 26 30 33 00	22 00 08 17 04 12	31 10 28 16 20 32
18 21 12 00 15 36	24 20 14 16 26 22	17 01 29 02 33 00	28 10 32 25 00 31	06 03 11 09 00 19	27 00 00 00 35 34
02 00 33 15 25 10	13 34 27 00 36 00	12 00 06 00 05 00	31 32 18 00 20 00	00 00 04 00 29 11	23 14 19 03 00 30
00 22 20 13 01 23	10 32 09 31 11 05	14 00 02 29 26 18	00 17 24 00 03 19	25 16 00 30 12 21	00 27 04 08 06 35
05 19 34 30 32 12	17 18 00 04 00 06	03 00 33 00 36 21	27 00 29 23 00 35	31 02 00 01 15 14	26 00 10 28 16 00
03 07 31 24 00 18	00 12 00 23 02 33	30 22 08 11 00 19	26 00 00 04 00 10	27 20 00 00 32 34	13 01 29 36 00 09
36 08 29 00 00 14	26 15 22 07 30 28	04 17 00 01 23 20	12 16 06 13 05 00	18 00 03 00 33 00	11 31 32 24 21 02
26 04 16 11 00 06	29 01 19 35 14 00	34 07 27 32 10 31	30 22 00 02 21 00	24 13 17 28 00 23	15 18 12 33 05 20
06 00 10 33 20 01	00 31 15 30 12 00	16 19 00 21 24 17	18 34 09 28 11 02	08 00 29 00 27 26	00 00 00 23 00 00
16 27 08 22 18 02	00 00 28 14 01 00	07 00 15 06 11 12	24 25 30 21 26 33	04 09 23 31 00 32	17 34 20 13 10 19
19 03 28 17 23 32	21 08 00 26 00 11	09 25 00 20 27 02	04 29 12 00 01 00	13 06 00 07 18 36	33 00 30 15 24 31
31 34 15 12 29 24	04 19 00 20 09 25	33 23 26 28 00 10	03 36 27 17 06 32	16 11 35 14 00 02	08 21 18 01 07 00
14 36 09 21 26 04	33 00 06 00 00 27	01 30 18 08 00 00	00 07 35 31 00 13	19 28 15 12 20 05	00 11 25 02 00 29
25 00 11 00 30 05	02 00 00 18 16 32	31 29 36 03 00 04	08 19 14 20 00 15	17 01 00 21 24 33	28 26 06 27 09 12
21 00 00 10 11 35	22 07 20 00 28 36	00 02 00 18 00 14	32 12 04 34 17 03	05 33 31 00 00 08	06 00 23 30 27 16
08 16 22 18 00 09	23 27 02 12 33 14	00 36 00 35 00 00	13 00 05 29 31 25	10 17 19 34 11 24	21 20 00 26 28 01
13 29 32 23 05 07	00 16 03 10 18 00	27 31 34 15 17 25	20 26 19 33 28 11	30 21 09 35 02 00	00 08 22 12 36 24
20 24 19 03 31 30	15 17 26 00 34 08	28 21 01 33 04 23	14 06 36 22 02 09	12 18 32 27 16 07	29 13 05 35 25 10
27 00 06 02 00 15	00 21 29 09 05 31	10 11 00 26 20 07	16 24 01 00 35 18	00 25 14 03 00 28	32 19 34 00 33 04
33 00 26 34 14 17	19 00 24 25 00 01	05 08 16 13 12 00	21 23 00 27 07 30	36 00 20 06 22 04	09 02 31 18 11 03
00 09 03 32 00 16	18 33 12 05 04 29	22 00 10 30 00 24	00 31 13 00 36 20	11 26 21 15 17 00	07 28 01 34 00 27
07 00 00 00 28 22	32 09 10 02 27 26	25 34 00 23 31 00	00 04 33 14 08 12	20 19 16 18 35 00	24 00 00 21 30 17
11 18 27 01 02 20	31 00 00 00 21 30	00 03 28 17 00 15	05 00 34 00 32 23	33 29 00 00 36 25	19 04 08 00 13 14
12 00 13 19 00 26	14 28 36 24 25 20	18 00 21 27 02 09	29 30 00 00 15 22	34 04 00 00 00 00	00 16 11 31 23 06
00 15 30 14 35 31	01 03 11 34 17 19	20 00 05 00 29 00	02 21 16 06 24 26	28 32 00 00 07 13	22 00 09 25 12 18
00 25 00 04 34 21	35 13 23 08 07 15	36 06 12 00 19 00	11 18 28 00 00 17	09 00 30 00 14 00	20 05 03 32 26 33

Solved Matrix:

01 30 17 26 24 19	28 04 31 22 29 10	23 12 20 14 21 35	15 27 08 32 13 07	03 34 33 36 05 16	25 06 02 09 18 11
15 06 05 27 16 03	25 14 21 19 13 12	08 24 30 07 32 28	17 20 22 26 10 29	02 23 18 11 09 31	01 33 35 04 34 36
35 11 07 25 12 08	05 26 18 33 24 23	29 04 31 34 06 27	19 09 02 36 30 28	15 14 01 20 21 17	10 03 16 22 32 13
34 31 18 09 22 29	20 36 30 03 15 17	02 26 11 05 01 13	23 33 21 16 25 04	32 24 28 10 06 35	12 07 27 19 14 08
32 23 21 36 33 28	08 02 07 27 06 16	19 10 09 22 18 03	35 01 31 11 34 14	29 12 13 04 25 30	05 17 24 20 15 26
04 02 14 20 10 13	34 11 01 32 35 09	15 16 17 36 25 33	06 05 03 18 12 24	26 08 07 22 19 27	30 23 21 29 31 28
17 20 01 31 13 33	07 25 32 15 10 21	35 09 04 19 30 34	22 03 11 24 18 05	14 27 26 16 28 29	02 12 36 06 08 23
22 10 35 16 07 11	27 05 33 36 19 02	32 20 03 12 28 26	34 08 23 15 29 06	21 30 24 13 31 18	04 09 17 14 01 25
09 32 04 05 06 25	12 30 17 28 08 34	13 18 23 31 16 22	07 14 20 35 19 27	01 36 02 33 10 15	03 24 26 11 29 21
30 26 23 28 03 27	11 06 04 29 31 13	21 14 24 10 08 36	09 02 17 12 16 01	07 35 25 32 34 20	18 22 33 05 19 15
24 14 02 29 19 34	09 23 35 01 03 18	11 27 07 25 15 06	36 13 26 30 33 21	22 05 08 17 04 12	31 10 28 16 20 32
18 21 12 08 15 36	24 20 14 16 26 22	17 01 29 02 33 05	28 10 32 25 04 31	06 03 11 09 23 19	27 30 13 07 35 34
02 17 33 15 25 10	13 34 27 21 36 24	12 28 06 09 05 16	31 32 18 01 20 08	35 07 04 26 29 11	23 14 19 03 22 30
28 22 20 13 01 23	10 32 09 31 11 05	14 15 02 29 26 18	33 17 24 07 03 19	25 16 36 30 12 21	34 27 04 08 06 35
05 19 34 30 32 12	17 18 08 04 20 06	03 13 33 24 36 21	27 11 29 23 09 35	31 02 22 01 15 14	26 25 10 28 16 07
03 07 31 24 21 18	16 12 25 23 02 33	30 22 08 11 35 19	26 28 15 04 14 10	27 20 06 05 32 34	13 01 29 36 17 09
36 08 29 35 27 14	26 15 22 07 30 28	04 17 25 01 23 20	12 16 06 13 05 34	18 10 03 19 33 09	11 31 32 24 21 02
26 04 16 11 09 06	29 01 19 35 14 03	34 07 27 32 10 31	30 22 25 02 21 36	24 13 17 28 08 23	15 18 12 33 05 20
06 13 10 33 20 01	03 31 15 30 12 07	16 19 35 21 24 17	18 34 09 28 11 02	08 22 29 25 27 26	36 32 14 23 04 05
16 27 08 22 18 02	36 29 28 14 01 35	07 05 15 06 11 12	24 25 30 21 26 33	04 09 23 31 03 32	17 34 20 13 10 19
19 03 28 17 23 32	21 08 34 26 22 11	09 25 14 20 27 02	04 29 12 05 01 16	13 06 10 07 18 36	33 35 30 15 24 31
31 34 15 12 29 24	04 19 05 20 09 25	33 23 26 28 13 10	03 36 27 17 06 32	16 11 35 14 30 02	08 21 18 01 07 22
14 36 09 21 26 04	33 24 06 17 23 27	01 30 18 08 34 32	10 07 35 31 22 13	19 28 15 12 20 05	16 11 25 02 03 29
25 35 11 07 30 05	02 10 13 18 16 32	31 29 36 03 22 04	08 19 14 20 23 15	17 01 34 21 24 33	28 26 06 27 09 12
21 01 25 10 11 35	22 07 20 13 28 36	24 02 19 18 09 14	32 12 04 34 17 03	05 33 31 29 26 08	06 15 23 30 27 16
08 16 22 18 04 09	23 27 02 12 33 14	06 36 32 35 03 30	13 15 05 29 31 25	10 17 19 34 11 24	21 20 07 26 28 01
13 29 32 23 05 07	06 16 03 10 18 04	27 31 34 15 17 25	20 26 19 33 28 11	30 21 09 35 02 01	14 08 22 12 36 24
20 24 19 03 31 30	15 17 26 11 34 08	28 21 01 33 04 23	14 06 36 22 02 09	12 18 32 27 16 07	29 13 05 35 25 10
27 12 06 02 36 15	30 21 29 09 05 31	10 11 22 26 20 07	16 24 01 08 35 18	23 25 14 03 13 28	32 19 34 17 33 04
33 28 26 34 14 17	19 35 24 25 32 01	05 08 16 13 12 29	21 23 10 27 07 30	36 15 20 06 22 04	09 02 31 18 11 03
23 09 03 32 08 16	18 33 12 05 04 29	22 35 10 30 14 24	25 31 13 19 36 20	11 26 21 15 17 06	07 28 01 34 02 27
07 05 36 06 28 22	32 09 10 02 27 26	25 34 13 23 31 11	01 04 33 14 08 12	20 19 16 18 35 03	24 29 15 21 30 17
11 18 27 01 02 20	31 22 16 06 21 30	26 03 28 17 07 15	05 35 34 09 32 23	33 29 12 24 36 25	19 04 08 10 13 14
12 33 13 19 17 26	14 28 36 24 25 20	18 32 21 27 02 09	29 30 07 03 15 22	34 04 05 08 01 10	35 16 11 31 23 06
10 15 30 14 35 31	01 03 11 34 17 19	20 33 05 04 29 08	02 21 16 06 24 26	28 32 27 23 07 13	22 36 09 25 12 18
29 25 24 04 34 21	35 13 23 08 07 15	36 06 12 16 19 01	11 18 28 10 27 17	09 31 30 02 14 22	20 05 03 32 26 33

0.014595573745727539

Observando a tabela abaixo, temos que à medida que o alpha aumenta, ou seja, à medida que diminui as casas do sudoku que o utilizador tem de preencher, diminui o tempo necessário para gerar o mesmo. Já à medida que o N aumenta, se observarmos para o valor do alpha mais baixo (0.0), ou seja, para o valor que significa que só existem casas brancas no sudoku que vamos gerar, deparamos que aumenta também o tempo de gerar o sudoku, pelo que a função que o gera é exponencial.

N \ Alfa	0.0	0.2	0.4	0.6
3	0.06076	0.02192	0.00843	0.00295
4	0.47593	0.13101	0.05242	0.00608
5	2.68906	0.55139	0.22028	0.01081
6	8.21655	2.00852	1.47335	0.02182