Trabalho 1 - Sudoku

Outubro, 2021

Inês Pires Presa - A90355
Tiago dos Santos Silva Peixoto Carriço - A91695

Variáveis:

Inputs do Problema

- N - dimensão do *Sudoku*
- a - fração das casas do *Sudoku* que devem estar preeenchidas inicialmente

Auxiliares

- $matriz_{i,j,c}$ - representa a atribuição de um número c à casa (i,j) do *Sudoku* (usada para definir as variáveis do *Solver*)
- $mat_{i,j,c}$ - representa a atribuição de um número c à casa (i,j) do *Sudoku* (usada para guardar a solução)

Onde $i,j,c \in [0..N^2-1]$

Condições:

1. Não alterar números que já estejam atribuídos
2. Garantir que todas as casas têm número
3. Casas na mesma linha não podem ter números repetidos
4. Casas na mesma coluna não podem ter números repetidos
5. Casas no mesmo quadrado não podem ter números repetidos

```
!pip install ortools
!pip install matploitlib
```

```
    Requirement already satisfied: ortools in /usr/local/lib/python3.7/dist-packages (9.1.9490)
    Requirement already satisfied: protobuf>=3.18.0 in /usr/local/lib/python3.7/dist-packages (from ortools) (3.19.0)
    Requirement already satisfied: absl-py>=0.13 in /usr/local/lib/python3.7/dist-packages (from ortools) (0.15.0)
    Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from absl-py>=0.13->ortools) (1.15.0)
    ERROR: Could not find a version that satisfies the requirement matploitlib (from versions: none)
    ERROR: No matching distribution found for matploitlib
```

```
from ortools.linear_solver import pywraplp
import networkx as nx
import random
import timeit
import matplotlib.pyplot as plt
from tabulate import tabulate
```

Funções:

- `sudoku(mat, N)` - resolve um *Sudoku* de dimensão N^2

```
def sudoku(mat, N):
  solver = pywraplp.Solver('BOP', pywraplp.Solver.BOP_INTEGER_PROGRAMMING)

  matriz = {}
  for i in range(N**2):
    matriz[i] = {}
    for j in range(N**2):
      matriz[i][j] = {}
      for c in range(N**2):
        matriz[i][j][c] = solver.BoolVar("matriz[%i][%i][%i]" % (i,j,c))

  #1. nao alterar numeros que ja estao atribuidos
  for x in range(N**2):
    for y in range(N**2):
      for c in range(N**2):
        if mat[y][x][c] == 1:
          solver.Add(matriz[y][x][c] == 1)


  #2. garantir que todos tem numero
```

```
  #2. garantir que todos tem numero
  for x in range(N**2):
    for y in range(N**2):
      solver.Add(sum([matriz[y][x][c] for c in range(N**2)]) == 1)

  #3. ligar linhas
  for y in range(N**2):
    for c in range(N**2):
      solver.Add(sum([matriz[y][x][c] for x in range(N**2)]) <= 1)

  #4. ligar colunas
  for x in range(N**2):
    for c in range(N**2):
      solver.Add(sum([matriz[y][x][c] for y in range(N**2)]) <= 1)

  #5. ligar quadrados
  for limx in range(0,N**2,N):
    for limy in range(0,N**2,N):
      for c in range(N**2):
        solver.Add(sum([matriz[y][x][c] for x in range(limx, limx+N) for y in range(limy, limy+N)]) <= 1)

  r = solver.Solve()

  if r == pywraplp.Solver.OPTIMAL:
    for y in range(N**2):
      for x in range(N**2):
        for c in range(N**2):
          mat[y][x][c] = round(matriz[y][x][c].solution_value())
```

- `matriz_parcial(mat, N)` - preenche os quadrados que constituem a diagonal do *Sudoku*

```
def matriz_parcial(mat, N):
  for limx in range(0,N**2,N):
    for limy in range(0,N**2,N):
      if limx == limy:
        lista = [x for x in range(N**2)]
        for x in range(limx, limx+N):
          for y in range(limy, limy+N):
            p = random.randint(0,len(lista)-1)
            num = lista.pop(p)
            mat[y][x][num] = 1
```

- `gerar_sudoku(mat, N, a)` - gera um *Sudoku* de dimensão N^2 com uma fração a das casas preenchidas

```
def gerar_sudoku(mat, N, a):

  for x in range(N**2):
    mat[x] = {}
    for y in range(N**2):
      mat[x][y] = {}
      for c in range(N**2):
        mat[x][y][c] = 0

  matriz_parcial(mat, N)
  sudoku(mat, N)

  remover = int((1-a)*(N**4))

  lista = [(x,y) for x in range(N**2) for y in range(N**2)]

  while remover != 0:
    num = random.randint(0,len(lista)-1)
    x,y = lista.pop(num)
    for c in range(N**2):
      mat[y][x][c] = 0
    remover -= 1

  return mat
```

- `print_sudoku(mat, N)` - imprime o *Sudoku*

```
def print_sudoku(mat, N):

  for y in range(N**2):
    for x in range(N**2):
```

```
      colorido = False
      for c in range(N**2):
        if mat[y][x][c] == 1:
          colorido = True
          print(" ", end ="")
          print("%02d" % (c+1), end ="")
          if (x+1) % N == 0 and x != N**2-1:
            print(" |", end="")
          break
      if not colorido:
        print(" ", end ="")
        print("..", end ="")
        if (x+1) % N == 0 and x != N**2-1:
          print(" |", end="")

    print()
    if (y+1) % N == 0 and y != N**2-1:
      for k in range(N):
        for p in range((3*N)+1):
          print("-", end ="")
        if k != N-1:
          print("+", end ="")
        else:
          print("\n", end ="")
```

Inicializar *Sudoku* a partir dos parâmetros N e a:

```
N = 6

a = 0.6

mat = {}

gerar_sudoku(mat, N,a)

print_sudoku(mat, N)
```

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

Resolver *Sudoku*:

```
sudoku(mat, N)
print_sudoku(mat, N)
```

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

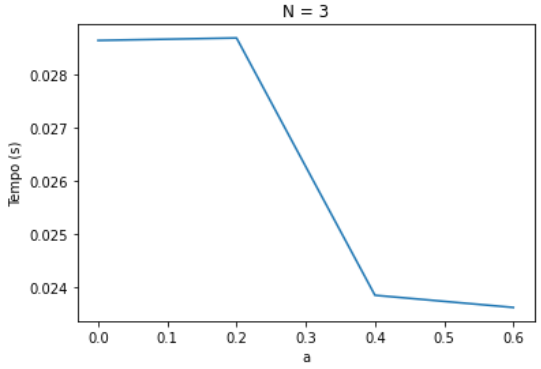
Analisar complexidade:

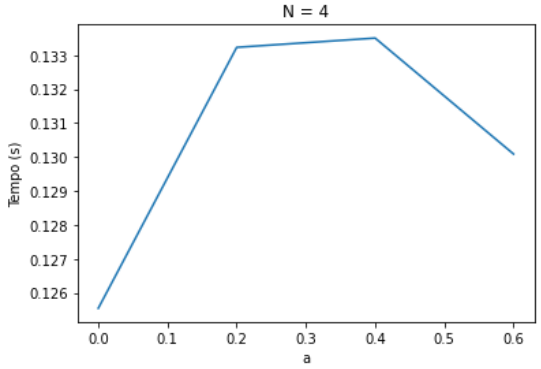
```
aLista = [0, 0.2, 0.4, 0.6]
tempos = {}

head = [str(a) for a in aLista]
head.insert(0, "N")

h = [[], [], []]

N = 3

tempos[N] = {}
y1 = []

mat = {}
h[0].insert(0, "3")
for j in range(len(aLista)):
  a = aLista[j]
  tempos[N][a] = []
  for i in range(100):
    gerar_sudoku(mat, N, a)
    t = timeit.timeit(lambda: sudoku(mat, N), number=1)
    tempos[N][a].append(t)
  h[0].insert(j+1, "Med: " + str( sum(tempos[N][a])/100 ) + "\nMax: "
              + str(max(tempos[N][a])) + "\nMin: " + str(min(tempos[N][a])) + "\n " )
  y1.append(sum(tempos[N][a])/100)



N = 4

tempos[N] = {}
y2 = []
```

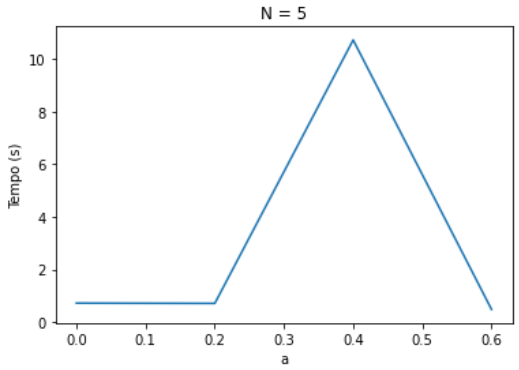
```
mat = {}
h[1].insert(0, "4")
for j in range(len(aLista)):
  a = aLista[j]
  tempos[N][a] = []
  for i in range(100):
    gerar_sudoku(mat, N, a)
    t = timeit.timeit(lambda: sudoku(mat, N), number=1)
    tempos[N][a].append(t)
  h[1].insert(j+1, "Med: " + str( sum(tempos[N][a])/100 ) + "\nMax: "
              + str(max(tempos[N][a])) + "\nMin: " + str(min(tempos[N][a])) + "\n " )
  y2.append(sum(tempos[N][a])/100)



N = 5

tempos[N] = {}
y3 = []

mat = {}
h[2].insert(0, "5")
for j in range(len(aLista)):
  a = aLista[j]
  tempos[N][a] = []
  for i in range(40):
    gerar_sudoku(mat, N, a)
    t = timeit.timeit(lambda: sudoku(mat, N), number=1)
    tempos[N][a].append(t)
  h[2].insert(j+1, "Med: " + str( sum(tempos[N][a])/40 ) + "\nMax: "
              + str(max(tempos[N][a])) + "\nMin: " + str(min(tempos[N][a])) + "\n " )
  y3.append(sum(tempos[N][a])/40)

print(tabulate(h, headers=head))
```

N	0	0.2	0.4	0.6
3	Med: 0.028638941879871708 Max: 0.04475399299917626 Min: 0.023206410001876066	Med: 0.028685506409965457 Max: 0.0465018909999344 Min: 0.02242190600009053	Med: 0.02384762308003701 Max: 0.031209400000079768 Min: 0.022193397999217268	Med: 0.023618704870023065 Max: 0.0380085169999802 Min: 0.022189391998836072
4	Med: 0.12554394073002187 Max: 0.1406551740001305 Min: 0.11659611700088135	Med: 0.13322966569001438 Max: 0.15590187600173522 Min: 0.11920810599985998	Med: 0.13350263527016068 Max: 0.16989972200099146 Min: 0.12095265199968708	Med: 0.13008753920021263 Max: 0.1533567980004591 Min: 0.1204048719991988
5	Med: 0.7214898833747612 Max: 0.7793578409982729 Min: 0.6729779849993065	Med: 0.7114401382747928 Max: 1.26244105799924 Min: 0.5013536529986595	Med: 10.732293781374755 Max: 102.33071397399908 Min: 0.711016440000094	Med: 0.4806195766997007 Max: 0.5125443640026788 Min: 0.4502373159994022

```
plt.plot(aLista, y1, label="N=3")

plt.xlabel('a')
plt.ylabel('Tempo (s)')

plt.title('N = 3')

plt.show()

plt.plot(aLista, y2, label="N=4")

plt.xlabel('a')
plt.ylabel('Tempo (s)')

plt.title('N = 4')

plt.show()

plt.plot(aLista, y3, label="N=5")

plt.xlabel('a')
plt.ylabel('Tempo (s)')

plt.title('N = 5')

plt.show()
```







No caso de N = 3 reparamos que, uma vez que a solução é obtida de forma relativamente rápida, não se verifica uma grande discrepância entre o tempo de execução para cada $a \in \{0, 0.2, 0.4, 0.6\}$.

Considerando o N = 4 começamos a verificar que a fração de casas preechidas influencia a complexidade do problema, tornando-se evidente que até $a = 0.4$ o tempo de execução aumenta e a partir de $a = 0.6$ este diminui.

Para o caso do N = 5, tal como para N = 4, o tempo de execução aumenta substancialmente com $a = 0.4$ e diminui para $a = 0.6$, mas para $a = 0.2$ mantem-se perto de valor obtido para $a = 0$.

Para N = 6, uma vez que o número de variáveis é significativamente maior, o tempo de execução é gravemente afetado, pelo que se tornou inconcebível fazer a comparação entre os diferentes $a's$.