Sudoku

October 24, 2021

1 TP1 - Grupo 4

Pedro Paulo Costa Pereira - A88062

Tiago André Oliveira Leite - A91693

2 Problema 2 - Sudoku

Da definição do jogo "Sudoku" generalizado para a dimensão N; o problema tradicional corresponde ao caso $N = 3$. O objetivo do Sudoku é preencher uma grelha de $N^2 \times N^2$ com inteiros positivos no intervalo 1 até N^2, satisfazendo as seguintes regras: - Cada inteiro no intervalo 1 até N^2 ocorre só uma vez em cada coluna, linha e secção $N \times N$. - No início do jogo uma fração $0 \leq < 1$ das N^4 casas da grelha são preenchidas de forma consistente com a regra anterior.

2.1 Implementação

Para a resolução deste problema, utilizaremos um solver disponivel na biblioteca de programação linear do OR-Tools, o *pywraplp*. Portanto, começaremos por instalar o OR-Tools.

[1]:
```
!pip install ortools
```

```
Requirement already satisfied: ortools in /usr/local/lib/python3.8/dist-packages
(9.1.9490)
Requirement already satisfied: protobuf>=3.18.0 in
/usr/local/lib/python3.8/dist-packages (from ortools) (3.18.1)
Requirement already satisfied: absl-py>=0.13 in /usr/local/lib/python3.8/dist-
packages (from ortools) (0.14.1)
Requirement already satisfied: six in /usr/lib/python3/dist-packages (from absl-
py>=0.13->ortools) (1.14.0)
```

[4]:
```
import math
import random
import timeit
import time
from ortools.linear_solver import pywraplp
import matplotlib.pyplot as plt
```

2.1.1 Variaveis do Problema

Para resolver o problema foram utilizadas $N^2 \times N^2 \times N^2$ variavies binarias, que nos premite verificar qual o número que vai ser colocado numa determinada linha e coluna da grelja do sudoku. Assim sendo foi definido o seguinte grupo de variavies: - $cube_{row,col,depth}$ - variavel que represena o número $depth + 1$ na linha row e na coluna col.

2.1.2 Restrições

- $\forall_{row,col,row}\ cube[(row, col, depth)] \in \{0, 1\}$
- $\forall_{row,col} \sum_{depth} cube[(row, col, depth)] == 1$
- $\forall_{row,depth} \sum_{col} cube[(row, col, depth)] == 1$
- $\forall_{depth,col} \sum_{row} cube[(row, col, depth)] == 1$

Seja n uma secção $N \times N$ do cubo: - $\forall_{n_{row},n_{col}} \sum_{depth} cube[(n_{row}, n_{col}, depth)] == 1$ - $\forall_{n_{row},depth} \sum_{n_{row}} cube[(n_{row}, n_{col}, depth)] == 1$ - $\forall_{depth,n_{col}} \sum_{n_{row}} cube[(n_{row}, n_{col}, depth)] == 1$

2.1.3 Função que imprime a grelha do sudoku

[5]:
```
def print_sudoku(sudoku):
    grid = 0
    l = len(sudoku)
    n = int(math.sqrt(l))
    pad = 1 + l // 10
    r = (l + n) * (pad + 1)  +1
    if n > 4 and not grid:
        for i in range(l):
            for j in range(l):
                print(f'{sudoku[i][j]}'.ljust(pad), end = " ")
            print('')
        return

    for i in range(l):
        if i % n == 0:
            print("-"*r)
        for j in range(l):
            if  j % n == 0:
                print("|".ljust(pad), end = " ")
            print(f'{sudoku[i][j]}'.ljust(pad), end = " ")
            if  j == l-1:
                print("|".ljust(pad), end = " ")
        print('')
    print("-" * r)
```

2.1.4 Função auxiliar que converte o resultado do solver numa matriz

[6]:
```
def converter(cube,dim):
    mat = [[0 for a in range(dim)] for b in range(dim) ]
    for row in range(dim):
        for col in range(dim):
            for depth in range(dim):
                if cube[(row,col,depth)].solution_value() == 1:
                    mat[row][col] = depth + 1
    return mat
```

2.1.5 Função que inicializa a grelha do sudoku

Recebe como parametros a dimensão N e a fracção $0 < 1$ de casas que sao preenchidas. Para tal coloca N números aletorios e depois tenta resolver o sudoku. De seguida apaga o número de números necessários para satisfazer

[23]:
```
def sudoku_generator(N, alpha):
    dim = N*N
    delete = int(dim * dim * (1-alpha))
    num_squares = int(math.sqrt(dim))
    solver = pywraplp.Solver.CreateSolver('SCIP')
    cube = {}

    if alpha == 0:
        mat = [[0 for c in range(dim)] for r in range(dim)]
        return mat


    for row in range(dim):
        for col in range(dim):
            for depth in range(dim):
                cube[(row,col,depth)] = solver.BoolVar('%i%i%i' %␣
↪(row,col,depth))

    for row in range(dim):
        for col in range(dim):
            val = []
            for depth in range(dim):
                val.append(cube[(row,col,depth)])
            solver.Add(sum(val) == 1)

    for row in range(dim):
        for depth in range(dim):
            val = []
            for col in range(dim):
                val.append(cube[(row,col,depth)])
```

```
            solver.Add(sum(val) == 1)

    for depth in range(dim):
        for col in range(dim):
            val = []
            for row in range(dim):
                val.append(cube[(row, col, depth)])
            solver.Add(sum(val) == 1)



    for depth in range(dim):
        for row in range(0, dim, N):
            for col in range (0, dim, N):
                val = []
                for i in range(N):
                    for j in range(N):
                        val.append(cube[(row + i , col + j , depth)] )
                solver.Add(sum(val) == 1)

    # gerar alguma aleatoriedade
    for i in range(N):
        randoms = []
        row = random.randint(0,dim-1)
        col = random.randint(0,dim-1)
        depth = i
        while (row,col) in randoms:
            row = random.randint(0,dim-1)
            col = random.randint(0,dim-1)
        randoms.append((row,col))
        solver.Add(cube[(row,col,depth)] == 1)

    status = solver.Solve()
    if status == pywraplp.Solver.OPTIMAL:
        mat = converter(cube, dim)

        # apagar celuas da matriz de forma a ficarem apenas alpha preenchidas
        candidates = [(a,b) for a in range(dim) for b in range(dim)]
        len_candidates = len(candidates)
        for d in range(delete):
            selected = random.randint(0,len_candidates-1)
            (row, col) = candidates.pop(selected)
            len_candidates -= 1
            mat[row][col] = 0

        return mat
    else:
```

```
        return False
```

Exemplo

[65]:
```
mat = sudoku_generator(3, 0.2)
print_sudoku(mat)
```

```
-------------------------
| 0 9 0 | 6 7 5 | 0 0 2 |
| 0 0 0 | 0 0 0 | 0 0 0 |
| 0 0 0 | 0 4 1 | 0 6 0 |
-------------------------
| 0 6 0 | 0 0 0 | 0 0 0 |
| 0 0 0 | 0 1 0 | 0 0 0 |
| 0 2 0 | 4 0 0 | 0 0 0 |
-------------------------
| 0 0 0 | 0 8 0 | 1 0 0 |
| 3 8 0 | 0 5 0 | 0 0 0 |
| 0 0 0 | 0 0 0 | 0 0 0 |
-------------------------
```

2.1.6 Outra forma mais simples de inicializar o sudoku mas que produz sudokus impossiveis

[66]:
```
def check(sudoku, row, column, number):
    if number in sudoku[row]:
        return False
    for r in sudoku:
        if r[column] == number:
            return False
    n = int(math.sqrt(len(sudoku)))
    r = (row // n) * n
    c = (column // n) * n
    for i in range(n):
        for j in range(n):
            if sudoku[r+i][c+j] == number:
                return False
    return True

def sudoku_generator_simple(N, alpha):
    dim = N*N
    filled = int(dim * dim * alpha)
    mat = [[0 for a in range(dim)] for a in range(dim)]
    while(filled > 0):
        row = random.randint(0,dim-1)
        col = random.randint(0,dim-1)
        number = random.randint(1, dim - 1)
```

```
        while(mat[row][col] != 0 or not check(mat,row,col,number)):
            row = random.randint(0, dim-1)
            col = random.randint(0, dim-1)
            number = random.randint(1,dim-1)
        mat[row][col] = number
        filled -=1

    return mat
```

Exemplo

[67]:
```
mat = sudoku_generator_simple(3,0.2)
if mat:
    print_sudoku(mat)
```

```
-------------------------
| 0 0 0 | 0 0 4 | 0 5 2 |
| 0 0 6 | 0 0 0 | 0 0 0 |
| 7 2 0 | 0 0 0 | 0 0 0 |
-------------------------
| 1 0 0 | 0 0 0 | 0 3 0 |
| 0 0 0 | 0 2 3 | 0 0 0 |
| 0 0 0 | 4 0 1 | 0 0 8 |
-------------------------
| 0 0 0 | 0 8 0 | 0 0 0 |
| 0 0 0 | 0 0 0 | 0 0 0 |
| 0 0 0 | 3 0 0 | 0 6 0 |
-------------------------
```

2.1.7 Função que resolve o sudoku

[8]:
```
def sudoku_solver(sudoku):
    solver = pywraplp.Solver.CreateSolver('SCIP')
    dim = len(sudoku)
    N = int(math.sqrt(dim))
    cube = {}

    for row in range(dim):
        for col in range(dim):
            for depth in range(dim):
                cube[(row,col,depth)] = solver.BoolVar('%i%i%i' %␣
 ↪(row,col,depth))


    for row in range(dim):
        for col in range(dim):
            val = []
```

```
            for depth in range(dim):
                val.append(cube[(row,col,depth)])
            solver.Add(sum(val) == 1)

    for row in range(dim):
        for depth in range(dim):
            val = []
            for col in range(dim):
                val.append(cube[(row,col,depth)])
            solver.Add(sum(val) == 1)

    for depth in range(dim):
        for col in range(dim):
            val = []
            for row in range(dim):
                val.append(cube[(row, col, depth)])
            solver.Add(sum(val) == 1)

    for depth in range(dim):
        for row in range(0, dim, N):
            for col in range (0, dim, N):
                val = []
                for i in range(N):
                    for j in range(N):
                        val.append(cube[(row + i , col + j , depth)] )
                solver.Add(sum(val) == 1)

    for i in range(dim):
        for j in range(dim):
            if sudoku[i][j] != 0:
                solver.Add(cube[(i,j,sudoku[i][j]-1)] == 1)


    status = solver.Solve()
    mat = converter(cube, dim)
    if status == pywraplp.Solver.OPTIMAL:
        return mat
    else:
        return False
```

2.1.8 Testes

[9]:
```
def test_sudoku(N, alpha):
    mat = sudoku_generator(N, alpha)
    print("Sudoku")
    print_sudoku(mat)
    print("")
```

```
    mat = sudoku_solver(mat)
    if mat:
        print("Solução")
        print_sudoku(mat)
        print("")
```

[2]:
```
N = [3,4,5]
times = [[None for n in N] for n in range(4)]
```

2.1.9 $\alpha = 0.0$

$N = 3$

[72]:
```
times[0][0] = timeit.timeit(stmt='test_sudoku(3,0.0)', setup='from __main__␣
 ↪import test_sudoku',number=1 )
print("Tempo de execução em segundos:",times[0][0])
```

```
Sudoku
-------------------------
| 0 0 0 | 0 0 0 | 0 0 0 |
| 0 0 0 | 0 0 0 | 0 0 0 |
| 0 0 0 | 0 0 0 | 0 0 0 |
-------------------------
| 0 0 0 | 0 0 0 | 0 0 0 |
| 0 0 0 | 0 0 0 | 0 0 0 |
| 0 0 0 | 0 0 0 | 0 0 0 |
-------------------------
| 0 0 0 | 0 0 0 | 0 0 0 |
| 0 0 0 | 0 0 0 | 0 0 0 |
| 0 0 0 | 0 0 0 | 0 0 0 |
-------------------------

Solução
-------------------------
| 1 2 3 | 4 5 6 | 8 7 9 |
| 8 9 7 | 1 2 3 | 4 5 6 |
| 6 4 5 | 7 8 9 | 1 2 3 |
-------------------------
| 7 1 2 | 3 4 5 | 6 9 8 |
| 9 5 6 | 2 1 8 | 3 4 7 |
| 4 3 8 | 6 9 7 | 2 1 5 |
-------------------------
| 5 8 1 | 9 3 2 | 7 6 4 |
| 3 7 4 | 5 6 1 | 9 8 2 |
| 2 6 9 | 8 7 4 | 5 3 1 |
-------------------------
```

```
Tempo de execução em segundos: 0.04929005700068956
```

$N = 4$

```
[73]: times[0][1] = timeit.timeit(stmt='test_sudoku(4,0.0)', setup='from __main__␣
       ↪import test_sudoku',number=1)
      print("Tempo de execução em segundos:",times[0][1])
```

```
Sudoku
-------------------------------------------------------------
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
-------------------------------------------------------------
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
-------------------------------------------------------------
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
-------------------------------------------------------------
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
-------------------------------------------------------------

Solução
-------------------------------------------------------------
|  1  2  3  4  |  5  6  7  8  |  9  10 11 12 |  13 14 15 16 |
|  5  6  7  13 |  1  2  3  4  |  8  14 15 16 |  9  10 11 12 |
|  9  10 11 12 |  13 14 15 16 |  1  2  3  4  |  5  6  7  8  |
|  8  14 15 16 |  9  10 11 12 |  5  6  7  13 |  1  2  3  4  |
-------------------------------------------------------------
|  4  1  2  3  |  6  5  8  7  |  10 9  12 11 |  14 13 16 15 |
|  6  7  5  14 |  2  1  4  3  |  13 8  16 15 |  12 9  10 11 |
|  10 12 13 9  |  11 16 14 15 |  2  1  4  3  |  6  5  8  7  |
|  15 8  16 11 |  12 9  10 13 |  6  5  14 7  |  2  1  4  3  |
-------------------------------------------------------------
|  14 3  1  2  |  4  7  5  6  |  12 15 8  9  |  16 11 13 10 |
|  16 11 4  5  |  3  8  1  2  |  7  13 6  10 |  15 12 9  14 |
|  13 15 12 10 |  16 11 9  14 |  3  4  1  2  |  7  8  5  6  |
|  7  9  8  6  |  15 12 13 10 |  11 16 5  14 |  3  4  1  2  |
-------------------------------------------------------------
```

```
|  12 16 10 1   |  7  3  2  5   |  4  11 9  6   |  8  15 14 13 |
|  11 13 9  7   |  14 4  12 1   |  15 3  2  8   |  10 16 6  5  |
|  3  5  14 15 |  8  13 6  11 |  16 12 10 1   |  4  7  2  9  |
|  2  4  6  8   |  10 15 16 9   |  14 7  13 5   |  11 3  12 1  |
------------------------------------------------------------------------
```

```
Tempo de execução em segundos: 0.24585740499969688
```

$N = 5$

[79]:
```python
times[0][2] = timeit.timeit(stmt='test_sudoku(5,0.0)', setup='from __main__␣
↪import test_sudoku',number=1)
print("Tempo de execução em segundos:",times[0][2])
```

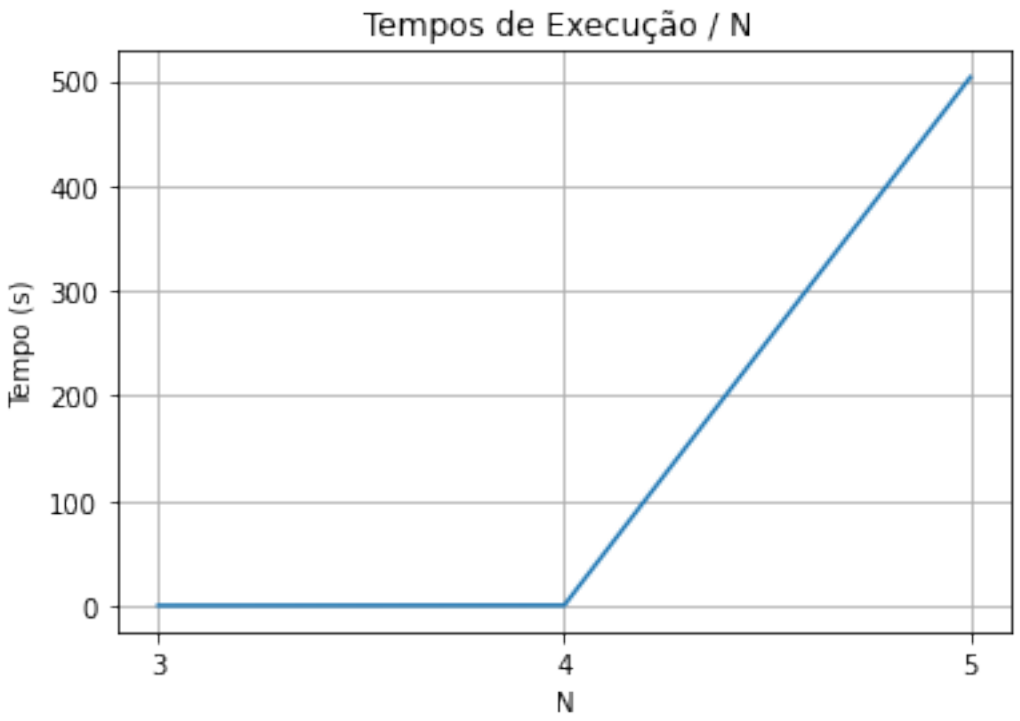
```
Sudoku
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
0    0    0    0    0
0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0    0
```

```
0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0
0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
0   0   0   0   0

Solução
3   19  25  11  18  16  5   21  24  2   22  7   1   15  10  23  20  13  9   8
4   12  17  14  6
15  8   5   4   22  13  11  1   6   18  3   21  23  17  25  19  12  14  2   7
24  16  9   20  10
20  23  14  21  17  10  12  25  19  4   9   2   16  13  6   22  24  5   18  15
8   11  7   1   3
24  16  7   10  13  9   3   14  17  15  20  19  11  12  8   25  6   21  1   4
2   23  5   18  22
12  2   6   1   9   20  8   7   22  23  4   18  14  5   24  11  10  16  17  3
21  13  25  19  15
23  1   2   8   24  3   21  19  15  20  17  25  7   10  18  9   4   22  6   5
12  14  11  16  13
22  13  10  20  19  12  23  6   5   11  15  8   4   24  21  17  25  7   14  16
3   1   18  9   2
11  3   9   5   25  7   18  22  13  16  6   14  12  1   23  15  21  10  8   2
20  4   19  17  24
7   18  17  6   15  14  25  4   9   24  16  3   19  2   13  12  1   20  23  11
5   22  21  10  8
21  4   12  16  14  2   17  8   1   10  11  5   9   22  20  13  3   24  19  18
15  7   6   23  25
13  22  3   9   5   6   16  20  10  17  1   23  18  7   2   8   15  12  25  19
11  21  24  4   14
16  25  11  23  12  19  4   2   14  3   8   13  15  6   5   10  18  1   24  21
7   9   20  22  17
2   24  8   18  4   25  7   13  23  21  12  9   17  20  3   14  22  11  5   6
1   10  16  15  19
10  6   1   17  7   8   9   15  11  5   14  24  22  21  19  4   23  2   16  20
13  25  3   12  18
19  14  21  15  20  24  1   18  12  22  25  4   10  11  16  3   7   9   13  17
```

```
6   2   23  8   5
4   9   13  22  8   11  15  23  16  14  18  20  3   19  12  7   17  6   21  10
25  24  2   5   1
17  11  24  2   21  4   10  5   3   7   23  1   25  16  22  20  19  18  12  13
14  15  8   6   9
18  12  15  14  16  22  13  9   25  6   10  11  5   8   7   1   2   23  3   24
17  19  4   21  20
1   5   20  3   23  21  2   17  8   19  24  6   13  4   14  16  9   25  15  22
10  18  12  7   11
25  10  19  7   6   18  24  12  20  1   21  15  2   9   17  5   8   4   11  14
16  3   22  13  23
9   21  22  12  2   15  6   3   18  8   13  10  20  23  4   24  14  17  7   25
19  5   1   11  16
14  17  4   13  11  5   19  10  7   25  2   22  21  3   9   18  16  8   20  1
23  6   15  24  12
6   7   16  24  1   17  14  11  21  9   19  12  8   18  15  2   5   3   10  23
22  20  13  25  4
8   15  23  25  3   1   20  24  4   12  5   16  6   14  11  21  13  19  22  9
18  17  10  2   7
5   20  18  19  10  23  22  16  2   13  7   17  24  25  1   6   11  15  4   12
9   8   14  3   21

Tempo de execução em segundos: 503.36963329100035
```

Gráfico Tempo de Execução / N

[80]:
```
plt.plot(N, times[0])
plt.xticks(N)
plt.title("Tempos de Execução / N ")
plt.ylabel("Tempo (s)")
plt.xlabel("N")
plt.grid(True)
```



2.1.10 $\alpha = 0.2$

$N = 3$

[74]:
```
times[1][0] = timeit.timeit(stmt='test_sudoku(3,0.2)', setup='from __main__␣
 ↪import test_sudoku',number=1 )
print("Tempo de execução em segundos:",times[1][0])
```

```
Sudoku
-------------------------
| 0 0 0 | 0 0 1 | 0 0 0 |
| 2 0 0 | 0 0 0 | 0 0 0 |
| 1 4 0 | 0 0 0 | 0 0 0 |
-------------------------
| 0 0 0 | 0 4 0 | 0 0 2 |
| 0 2 0 | 0 3 5 | 0 0 0 |
| 7 0 0 | 0 0 2 | 0 5 0 |
-------------------------
| 0 0 2 | 0 5 0 | 7 0 0 |
| 0 0 0 | 0 0 3 | 0 0 0 |
| 0 0 8 | 0 0 0 | 0 0 0 |
-------------------------

Solução
-------------------------
```

```
| 8 7 9 | 4 2 1 | 5 6 3 |
| 2 5 6 | 3 9 7 | 4 1 8 |
| 1 4 3 | 5 8 6 | 9 2 7 |
-------------------------
| 9 3 5 | 1 4 8 | 6 7 2 |
| 6 2 1 | 7 3 5 | 8 4 9 |
| 7 8 4 | 9 6 2 | 3 5 1 |
-------------------------
| 4 1 2 | 8 5 9 | 7 3 6 |
| 5 9 7 | 6 1 3 | 2 8 4 |
| 3 6 8 | 2 7 4 | 1 9 5 |
-------------------------

Tempo de execução em segundos: 0.09160416499980784
```

$N = 4$

[75]:
```
times[1][1] = timeit.timeit(stmt='test_sudoku(4,0.2)', setup='from __main__␣
 ↪import test_sudoku',number=1)
print("Tempo de execução em segundos:",times[1][1])
```

```
Sudoku
-----------------------------------------------------------
|  0  0  0  0  |  0  0  0  8  |  0  0  0  11 |  0  0  0  0  |
|  0  0  0  0  |  0  0  0  0  |  0  0  0  15 |  9  0  0  0  |
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
|  0  0  0  0  |  0  10 0  0  |  0  0  0  8  |  0  2  3  0  |
-----------------------------------------------------------
|  2  1  0  0  |  0  5  0  7  |  0  0  11 0  |  0  0  0  0  |
|  0  0  8  0  |  0  1  0  0  |  0  0  0  0  |  0  0  12 0  |
|  0  9  0  0  |  0  0  0  0  |  0  0  4  0  |  6  0  0  7  |
|  0  0  0  0  |  0  0  0  0  |  0  0  8  0  |  0  0  4  0  |
-----------------------------------------------------------
|  0  0  0  0  |  0  0  0  0  |  11 0  0  0  |  0  16 15 10 |
|  7  0  0  6  |  0  4  0  5  |  10 0  16 0  |  11 0  0  13 |
|  0  12 0  0  |  0  0  0  0  |  0  4  0  0  |  7  0  5  0  |
|  0  16 0  0  |  0  12 0  0  |  0  0  0  0  |  0  0  0  0  |
-----------------------------------------------------------
|  0  0  0  0  |  0  7  0  0  |  14 0  0  10 |  0  11 0  0  |
|  0  0  0  0  |  0  11 0  1  |  0  0  0  0  |  0  0  0  0  |
|  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |  0  0  0  0  |
|  0  15 0  0  |  0  3  0  9  |  4  11 0  5  |  0  0  0  0  |
-----------------------------------------------------------

Solução
-----------------------------------------------------------
|  1  6  5  2  |  4  9  3  8  |  12 16 14 11 |  13 7  10 15 |
|  8  7  11 16 |  1  6  12 13 |  3  2  10 15 |  9  4  14 5  |
```

```
|  15 13 10 3  |  5  16 14 2  |  9  1  7  4  |  8  12 11 6  |
|  4  14 12 9  |  11 10 7  15 |  13 5  6  8  |  16 2  3  1  |
-----------------------------------------------------------
|  2  1  6  4  |  9  5  10 7  |  15 12 11 3  |  14 13 8  16 |
|  11 3  8  14 |  2  1  15 4  |  7  6  13 16 |  10 5  12 9  |
|  12 9  15 5  |  13 8  16 11 |  1  10 4  14 |  6  3  2  7  |
|  13 10 16 7  |  12 14 6  3  |  5  9  8  2  |  1  15 4  11 |
-----------------------------------------------------------
|  5  8  4  13 |  7  2  1  6  |  11 14 3  9  |  12 16 15 10 |
|  7  2  3  6  |  8  4  9  5  |  10 15 16 12 |  11 14 1  13 |
|  10 12 9  11 |  14 15 13 16 |  6  4  2  1  |  7  8  5  3  |
|  14 16 1  15 |  3  12 11 10 |  8  7  5  13 |  4  9  6  2  |
-----------------------------------------------------------
|  3  5  13 1  |  6  7  2  12 |  14 8  9  10 |  15 11 16 4  |
|  9  4  14 8  |  16 11 5  1  |  2  13 15 6  |  3  10 7  12 |
|  6  11 2  10 |  15 13 4  14 |  16 3  12 7  |  5  1  9  8  |
|  16 15 7  12 |  10 3  8  9  |  4  11 1  5  |  2  6  13 14 |
-----------------------------------------------------------

Tempo de execução em segundos: 0.8251932869989105
```

$N = 5$

[76]:
```
times[1][2] = timeit.timeit(stmt='test_sudoku(5,0.2)', setup='from __main__␣
 ↪import test_sudoku',number=1)
print("Tempo de execução em segundos:",times[1][2])
```

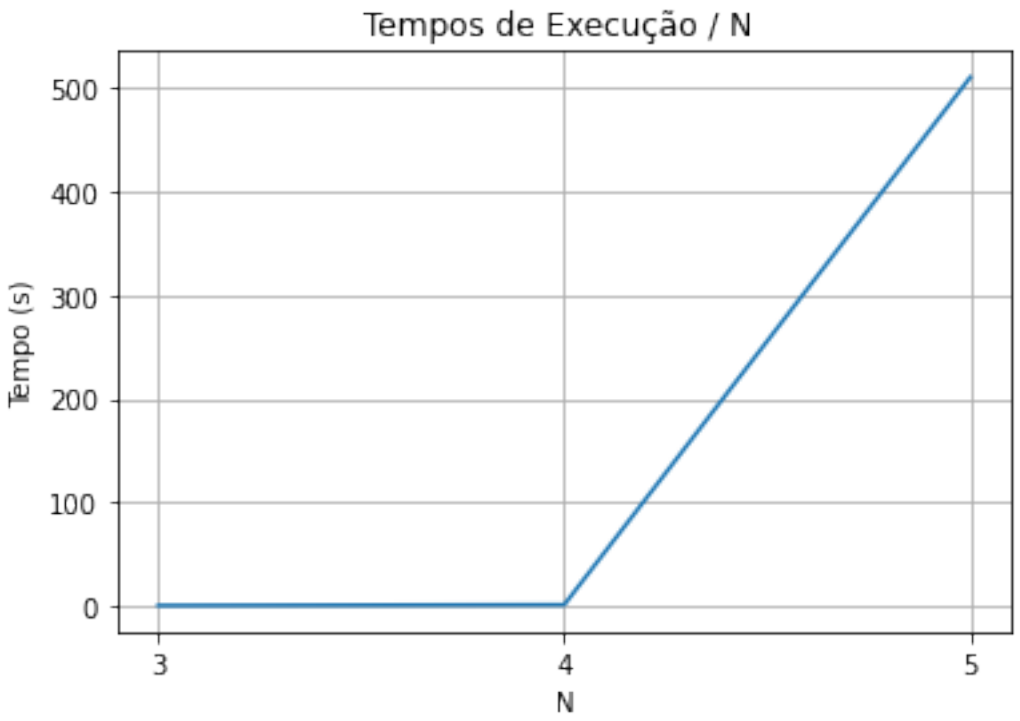
```
Sudoku
0   0   0   0   0   14  0   0   19  0   0   0   10  0   0   11  2   0   0   0
8   23  0   1   0
0   0   1   0   0   0   0   0   0   15  0   9   18  0   14  0   7   0   0   0
0   19  0   0   16
0   0   0   14  0   0   0   0   0   0   0   21  5   16  0   9   18  0   0   0
0   0   0   0   0
2   16  0   12  0   0   0   0   0   20  24  0   0   0   0   0   0   0   0   0
0   0   0   0   0
0   0   11  0   0   0   0   0   0   0   0   0   0   0   0   0   10  0   0   24
0   0   0   0   0
0   0   0   0   7   0   0   0   0   0   0   0   0   0   0   0   4   9   0   0   0
5   0   0   13  0
0   0   0   0   0   11  0   0   0   0   0   0   0   0   0   0   5   0   0   0
0   24  0   6   0
0   0   0   0   24  0   0   0   0   0   0   0   15  0   0   0   16  6   0   2
23  0   0   0   0
0   0   0   21  0   18  0   12  0   4   0   0   0   0   0   3   0   0   0   0
10  0   0   0   0
0   0   0   0   0   24  14  15  0   0   18  0   0   0   0   0   0   0   11  0
0   0   0   0   4
```

```
16 0 0 0 2 8 18 0 0 0 0 0 0 0 0 0 0 1 0 0
25 0 17 0 0
0 0 24 0 0 7 0 0 15 0 0 0 0 0 0 2 0 0 0 0
12 0 0 0 0
5 4 0 0 0 0 9 0 0 0 0 14 0 0 0 0 11 0 0 6
0 0 0 0 23
0 0 0 0 0 0 0 0 0 16 0 0 0 0 0 0 0 0 0 0
0 0 0 0 5
0 0 0 0 0 6 0 0 0 2 0 19 0 0 0 0 0 0 0 0
7 0 0 0 0
0 0 0 0 0 0 0 0 6 19 0 0 24 0 0 0 0 15 0 0
0 0 0 0 22
0 0 3 0 0 20 13 0 0 0 0 0 0 0 0 0 1 0 0 25
14 0 0 0 0
8 0 0 0 0 0 0 1 0 0 0 0 0 10 0 0 0 0 0 0
0 0 0 0 0
22 0 0 0 0 15 7 0 0 0 0 0 0 0 21 0 0 0 0 0
20 0 0 0 0
19 0 25 18 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 7 0 16 11
0 0 0 0 0 0 0 7 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0
21 0 0 0 0 0 0 0 0 0 0 0 0 19 0 10 15 0 0 0 18
0 0 0 22 0
0 18 0 22 0 0 0 0 0 12 0 0 0 0 0 0 0 0 0 0
3 0 0 0 0
0 12 0 0 10 0 0 16 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0
7 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 9
0 0 0 0 1

Solução
13 5 21 24 22 14 4 6 19 18 3 15 10 12 7 11 2 25 17 16
8 23 9 1 20
20 25 1 23 8 3 17 10 24 15 4 9 18 13 14 12 7 5 6 22
21 19 11 2 16
6 19 10 14 3 25 11 13 7 22 23 21 5 16 2 9 18 8 1 20
4 12 15 17 24
2 16 18 12 4 9 1 21 23 20 24 8 11 17 6 14 15 19 13 3
22 10 5 25 7
9 17 11 7 15 16 8 2 12 5 1 22 20 19 25 21 10 4 23 24
6 18 13 14 3
12 11 23 10 7 1 19 20 22 6 2 24 3 8 17 4 9 18 14 15
5 16 25 13 21
4 20 14 17 16 11 2 23 3 7 21 10 1 9 19 25 5 12 22 13
18 24 8 6 15
18 3 9 5 24 21 10 17 13 25 7 4 15 22 11 19 16 6 8 2
23 14 1 20 12
```

```
15  8   2   21  19  18  16  12  5   4   6   25  14  20  13  3   23  7   24  1
10  11  22  9   17
1   13  22  6   25  24  14  15  8   9   18  16  23  5   12  20  21  17  11  10
2   3   19  7   4
16  22  12  19  2   8   18  11  21  14  15  5   13  6   23  24  20  1   7   4
25  9   17  3   10
11  10  24  25  13  7   3   4   15  23  20  17  9   18  16  2   19  22  21  5
12  1   6   8   14
5   4   8   15  18  22  9   19  17  13  10  14  7   25  1   16  11  3   12  6
24  2   20  21  23
14  6   7   1   20  12  24  25  10  16  11  3   21  2   8   18  13  23  9   17
19  22  4   15  5
3   23  17  9   21  6   20  5   1   2   22  19  12  4   24  8   25  10  15  14
7   13  16  11  18
23  1   5   20  9   10  25  18  6   19  16  2   24  14  3   7   17  15  4   11
13  8   21  12  22
10  21  3   16  17  20  13  22  4   11  5   12  8   7   18  6   1   2   19  25
14  15  24  23  9
8   7   15  2   11  23  12  1   9   24  19  20  6   10  22  13  14  16  3   21
17  4   18  5   25
22  14  4   13  12  15  7   3   16  8   25  1   17  11  21  5   24  9   18  23
20  6   10  19  2
19  24  25  18  6   5   21  14  2   17  9   13  4   23  15  22  12  20  10  8
1   7   3   16  11
24  9   16  3   1   2   6   7   25  21  14  18  22  15  4   17  8   13  5   12
11  20  23  10  19
21  2   20  4   23  13  5   24  11  1   17  7   19  3   10  15  6   14  16  18
9   25  12  22  8
25  18  6   22  5   17  15  9   14  12  8   23  16  1   20  10  4   11  2   19
3   21  7   24  13
17  12  19  8   10  4   23  16  20  3   13  11  2   24  9   1   22  21  25  7
15  5   14  18  6
7   15  13  11  14  19  22  8   18  10  12  6   25  21  5   23  3   24  20  9
16  17  2   4   1

Tempo de execução em segundos: 510.60874791599963
```

Gráfico Tempo de Execução / N

[77]:
```
plt.plot(N, times[1])
plt.xticks(N)
plt.title("Tempos de Execução / N ")
plt.ylabel("Tempo (s)")
plt.xlabel("N")
plt.grid(True)
```



2.1.11 $\alpha = 0.4$

$N = 3$

[10]:
```
times[2][0] = timeit.timeit(stmt='test_sudoku(3,0.4)', setup='from __main__␣
 ↪import test_sudoku',number=1 )
print("Tempo de execução em segundos:",times[2][0])
```

```
Sudoku
-------------------------
| 1 0 0 | 0 0 2 | 0 0 0 |
| 7 0 0 | 9 0 6 | 0 0 0 |
| 0 0 2 | 5 4 0 | 9 8 0 |
-------------------------
| 3 7 0 | 8 2 5 | 0 0 0 |
| 0 4 5 | 7 0 0 | 2 0 0 |
| 8 0 0 | 0 0 4 | 7 3 0 |
-------------------------
| 5 8 0 | 0 6 0 | 0 0 1 |
| 0 0 0 | 0 0 0 | 8 2 3 |
| 2 1 0 | 0 0 0 | 0 0 6 |
-------------------------

Solução
-------------------------
```

```
| 1 9 8 | 3 7 2 | 6 5 4 |
| 7 5 4 | 9 8 6 | 3 1 2 |
| 6 3 2 | 5 4 1 | 9 8 7 |
-------------------------
| 3 7 6 | 8 2 5 | 1 4 9 |
| 9 4 5 | 7 1 3 | 2 6 8 |
| 8 2 1 | 6 9 4 | 7 3 5 |
-------------------------
| 5 8 3 | 2 6 9 | 4 7 1 |
| 4 6 9 | 1 5 7 | 8 2 3 |
| 2 1 7 | 4 3 8 | 5 9 6 |
-------------------------

Tempo de execução em segundos: 0.11473362099968654
```

$N = 4$

[11]:
```
times[2][1] = timeit.timeit(stmt='test_sudoku(4,0.4)', setup='from __main__␣
 ↪import test_sudoku',number=1)
print("Tempo de execução em segundos:",times[2][1])
```

```
Sudoku
-----------------------------------------------------------
|  0  0  3  4  |  0  6  0  0  |  0  0  0  12 |  0  0  0  16 |
|  0  0  0  8  |  1  0  0  4  |  13 0  0  0  |  9  0  0  0  |
|  0  0  0  12 |  0  14 0  16 |  1  2  3  0  |  5  0  7  0  |
|  13 14 15 0  |  0  10 11 0  |  0  6  0  0  |  0  2  3  4  |
-----------------------------------------------------------
|  0  0  5  3  |  4  0  6  9  |  8  0  10 0  |  0  0  0  0  |
|  4  7  0  9  |  0  0  0  0  |  0  0  16 0  |  8  11 10 0  |
|  0  0  0  0  |  12 0  0  0  |  0  1  4  0  |  0  0  0  0  |
|  0  0  16 0  |  0  0  0  0  |  0  5  9  0  |  0  0  4  0  |
-----------------------------------------------------------
|  0  4  0  0  |  0  0  0  7  |  0  0  12 0  |  0  0  13 15 |
|  6  5  8  0  |  10 4  0  0  |  16 3  14 0  |  11 0  12 2  |
|  0  0  0  0  |  0  0  14 0  |  0  0  0  6  |  0  1  8  0  |
|  0  0  13 0  |  0  9  0  0  |  0  8  1  0  |  10 0  5  0  |
-----------------------------------------------------------
|  7  0  0  0  |  11 0  0  5  |  0  0  0  0  |  15 0  0  10 |
|  11 8  0  6  |  7  0  0  0  |  3  4  0  10 |  0  13 14 5  |
|  15 0  0  10 |  0  0  2  0  |  0  12 0  0  |  4  7  0  0  |
|  16 12 4  0  |  14 13 0  0  |  0  7  0  1  |  0  0  0  11 |
-----------------------------------------------------------

Solução
-----------------------------------------------------------
|  5  9  3  4  |  2  6  7  15 |  11 14 8  12 |  13 10 1  16 |
|  2  16 7  8  |  1  12 3  4  |  13 10 15 5  |  9  6  11 14 |
```

```
|  10 6  11 12 |  13 14 9  16 |  1  2  3  4  |  5  15 7  8  |
|  13 14 15 1  |  5  10 11 8  |  9  6  7  16 |  12 2  3  4  |
-----------------------------------------------------------
|  12 1  5  3  |  4  15 6  9  |  8  11 10 7  |  14 16 2  13 |
|  4  7  6  9  |  3  2  5  13 |  12 15 16 14 |  8  11 10 1  |
|  8  2  10 13 |  12 11 16 14 |  6  1  4  3  |  7  5  15 9  |
|  14 11 16 15 |  8  7  1  10 |  2  5  9  13 |  6  3  4  12 |
-----------------------------------------------------------
|  1  4  2  16 |  6  5  8  7  |  10 9  12 11 |  3  14 13 15 |
|  6  5  8  7  |  10 4  13 1  |  16 3  14 15 |  11 9  12 2  |
|  9  10 12 11 |  15 3  14 2  |  4  13 5  6  |  16 1  8  7  |
|  3  15 13 14 |  16 9  12 11 |  7  8  1  2  |  10 4  5  6  |
-----------------------------------------------------------
|  7  3  1  2  |  11 8  4  5  |  14 16 13 9  |  15 12 6  10 |
|  11 8  9  6  |  7  16 15 12 |  3  4  2  10 |  1  13 14 5  |
|  15 13 14 10 |  9  1  2  6  |  5  12 11 8  |  4  7  16 3  |
|  16 12 4  5  |  14 13 10 3  |  15 7  6  1  |  2  8  9  11 |
-----------------------------------------------------------
```

Tempo de execução em segundos: 0.449589902000298

$N = 5$

[12]:
```
times[2][2] = timeit.timeit(stmt='test_sudoku(5,0.4)', setup='from __main__␣
 ↪import test_sudoku',number=1)
print("Tempo de execução em segundos:",times[2][2])
```

```
Sudoku
0   23  0   7   0   0   0   24  20  0   0   0   25  0   0   0   0   0   0   4
13  0   0   0   0
0   8   19  0   0   0   23  5   0   22  0   13  24  0   0   0   0   0   25  0
9   0   0   0   6
11  0   13  24  0   17  0   18  15  0   23  12  0   0   10  1   0   8   3   7
16  0   0   0   22
0   12  0   0   20  0   0   11  0   4   1   0   0   17  0   14  0   2   0   0
0   0   0   10  0
3   16  0   0   0   6   10  12  1   0   0   7   11  0   22  0   17  24  20  0
0   8   5   0   0
0   20  0   0   4   0   1   0   0   25  11  0   3   0   0   12  23  14  0   9
0   0   19  0   16
21  0   23  2   0   5   0   4   12  24  0   15  22  20  0   0   0   0   0   6
8   13  14  11  9
0   0   14  0   7   18  16  0   0   0   0   23  0   0   6   24  0   20  0   0
0   22  0   17  0
25  0   6   0   9   0   17  23  0   20  13  8   12  0   0   0   22  4   5   21
18  0   0   24  1
0   24  5   0   0   15  0   0   0   6   17  0   4   0   0   0   19  0   18  0
0   0   0   0   0
```

```
18 9 0 19 0 0 14 0 0 0 15 0 2 0 20 0 0 21 0 24
11 0 0 0 0
0 0 1 0 0 12 21 0 0 0 0 0 0 0 0 0 0 3 0 0
0 5 0 0 0
0 0 4 20 0 0 5 0 0 0 10 0 17 0 0 0 16 0 0 0
0 3 2 0 23
0 0 10 21 0 0 18 0 4 0 0 19 0 6 0 0 0 0 0 0
0 17 0 20 7
0 0 17 0 0 11 20 22 0 0 0 21 0 0 4 0 0 9 0 0
0 25 0 18 24
0 1 0 0 0 0 13 0 0 0 0 22 21 16 0 0 8 0 0 0
0 0 0 0 0
0 0 0 18 24 0 0 0 0 0 0 0 0 12 0 0 15 10 2 0
22 11 0 4 0
0 21 0 0 0 0 0 0 0 0 0 0 0 0 15 0 14 0 4 0
25 0 0 8 5
0 22 15 23 0 0 0 16 0 0 0 0 0 0 0 18 0 0 7 5
0 0 0 0 21
0 6 0 0 5 22 4 0 0 0 0 0 14 19 0 0 0 13 0 1
0 0 0 0 0
0 0 0 10 0 0 22 0 0 17 24 0 16 0 0 3 0 23 0 12
6 0 15 0 0
0 0 16 12 21 4 0 0 8 0 20 0 15 14 7 6 0 0 1 0
24 0 0 0 3
0 0 0 0 0 0 0 20 0 13 0 0 0 21 0 4 0 0 22 0
1 0 0 19 11
2 0 0 9 0 0 12 0 0 0 6 0 19 23 0 0 0 15 0 0
0 0 0 0 8
7 17 0 8 0 0 15 3 10 23 0 25 0 0 5 20 0 0 21 0
0 0 0 12 13
```

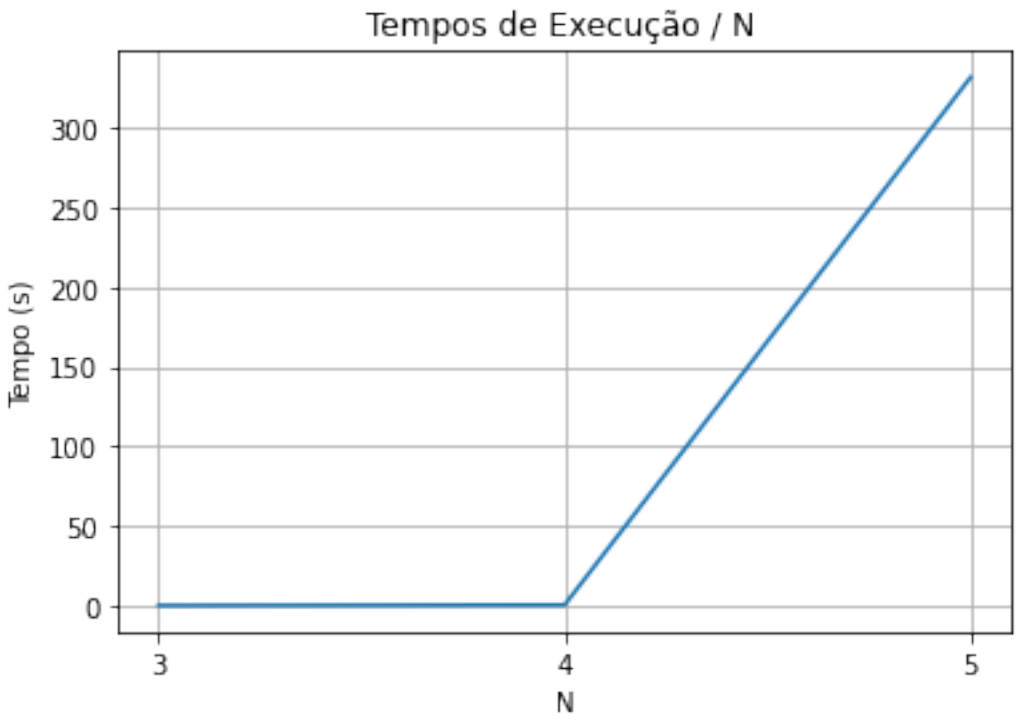
Solução

```
6 23 21 7 10 14 8 24 20 3 16 2 25 15 19 5 18 22 9 4
13 12 11 1 17
17 8 19 1 15 7 23 5 21 22 18 13 24 4 14 11 12 16 25 10
9 2 20 3 6
11 14 13 24 2 17 19 18 15 9 23 12 20 5 10 1 6 8 3 7
16 4 25 21 22
22 12 9 5 20 13 25 11 16 4 1 3 6 17 8 14 21 2 19 15
23 7 24 10 18
3 16 18 4 25 6 10 12 1 2 21 7 11 9 22 23 17 24 20 13
19 8 5 14 15
13 20 8 17 4 2 1 10 22 25 11 18 3 24 21 12 23 14 15 9
5 6 19 7 16
21 18 23 2 19 5 3 4 12 24 7 15 22 20 1 25 10 17 16 6
8 13 14 11 9
1 10 14 15 7 18 16 13 9 21 19 23 5 25 6 24 2 20 11 8
12 22 3 17 4
```

```
25  3   6   11  9   19  17  23  14  20  13  8   12  2   16  7   22  4   5   21
18  15  10  24  1
12  24  5   22  16  15  7   8   11  6   17  14  4   10  9   13  19  1   18  3
2   21  23  25  20
18  9   25  19  23  3   14  6   17  8   15  5   2   22  20  10  7   21  13  24
11  1   4   16  12
24  11  1   13  22  12  21  25  7   10  14  16  8   18  23  2   4   3   17  20
15  5   9   6   19
14  7   4   20  12  9   5   19  13  15  10  24  17  1   25  8   16  18  6   11
21  3   2   22  23
16  15  10  21  3   24  18  2   4   1   9   19  13  6   11  22  25  5   12  23
14  17  8   20  7
5   2   17  6   8   11  20  22  23  16  12  21  7   3   4   15  1   9   14  19
10  25  13  18  24
4   1   2   14  11  20  13  15  3   12  5   22  21  16  18  19  8   6   24  25
17  9   7   23  10
19  25  20  18  24  8   9   17  5   7   3   6   23  12  13  21  15  10  2   16
22  11  1   4   14
10  21  12  3   13  23  24  1   6   19  2   20  9   7   15  17  14  11  4   22
25  16  18  8   5
9   22  15  23  17  10  2   16  25  14  4   11  1   8   24  18  3   12  7   5
20  19  6   13  21
8   6   7   16  5   22  4   21  18  11  25  10  14  19  17  9   20  13  23  1
3   24  12  15  2
20  4   11  10  14  21  22  7   19  17  24  1   16  13  2   3   5   23  8   12
6   18  15  9   25
23  19  16  12  21  4   11  9   8   5   20  17  15  14  7   6   13  25  1   18
24  10  22  2   3
15  5   3   25  18  16  6   20  2   13  8   9   10  21  12  4   24  7   22  14
1   23  17  19  11
2   13  22  9   1   25  12  14  24  18  6   4   19  23  3   16  11  15  10  17
7   20  21  5   8
7   17  24  8   6   1   15  3   10  23  22  25  18  11  5   20  9   19  21  2
4   14  16  12  13

Tempo de execução em segundos: 331.8154337640008
```

Gráfico Tempo de Execução / N

[20]:
```
plt.plot(N, times[2])
plt.xticks(N)
plt.title("Tempos de Execução / N ")
plt.ylabel("Tempo (s)")
plt.xlabel("N")
plt.grid(True)
```



2.1.12 $\alpha = 0.6$

$N = 3$

[16]:
```
times[3][0] = timeit.timeit(stmt='test_sudoku(3,0.6)', setup='from __main__␣
 ↪import test_sudoku',number=1 )
print("Tempo de execução em segundos:",times[3][0])
```

```
Sudoku
-------------------------
| 0 9 0 | 0 6 5 | 4 0 3 |
| 0 0 0 | 0 0 0 | 8 0 1 |
| 6 3 0 | 0 4 1 | 5 9 0 |
-------------------------
| 9 0 0 | 0 8 0 | 0 5 2 |
| 8 0 6 | 5 1 9 | 7 3 0 |
| 5 1 0 | 4 0 7 | 0 8 0 |
-------------------------
| 0 8 0 | 0 7 6 | 3 1 0 |
| 3 7 0 | 0 9 8 | 2 4 6 |
| 2 6 1 | 3 5 4 | 0 7 8 |
-------------------------

Solução
-------------------------
```

```
| 1 9 8 | 7 6 5 | 4 2 3 |
| 7 5 4 | 9 3 2 | 8 6 1 |
| 6 3 2 | 8 4 1 | 5 9 7 |
-------------------------
| 9 4 7 | 6 8 3 | 1 5 2 |
| 8 2 6 | 5 1 9 | 7 3 4 |
| 5 1 3 | 4 2 7 | 6 8 9 |
-------------------------
| 4 8 9 | 2 7 6 | 3 1 5 |
| 3 7 5 | 1 9 8 | 2 4 6 |
| 2 6 1 | 3 5 4 | 9 7 8 |
-------------------------

Tempo de execução em segundos: 0.07458067299921822
```

$N = 4$

[18]:
```
times[3][1] = timeit.timeit(stmt='test_sudoku(4,0.6)', setup='from __main__␣
 ↪import test_sudoku',number=1)
print("Tempo de execução em segundos:",times[3][1])
```

```
Sudoku
-----------------------------------------------------------
|  11 6  0  15 |  0  1  8  0  |  0  9  0  10 |  4  14 0  13 |
|  0  0  0  7  |  13 12 11 0  |  4  0  5  2  |  0  6  0  0  |
|  3  1  2  0  |  4  0  6  7  |  0  13 16 15 |  11 0  5  0  |
|  8  13 5  4  |  9  0  15 0  |  0  6  7  11 |  2  3  0  10 |
-----------------------------------------------------------
|  7  0  8  10 |  1  3  14 0  |  16 4  13 12 |  6  0  15 0  |
|  6  12 0  0  |  0  0  0  15 |  5  0  8  14 |  13 2  0  4  |
|  0  0  0  0  |  11 0  4  8  |  9  7  0  0  |  12 0  10 0  |
|  16 4  0  11 |  0  0  13 6  |  2  3  0  0  |  0  0  8  0  |
-----------------------------------------------------------
|  0  3  0  13 |  8  15 12 14 |  11 16 0  7  |  0  0  1  0  |
|  0  0  6  0  |  7  11 0  1  |  10 0  4  13 |  3  8  14 2  |
|  15 8  0  0  |  3  16 10 4  |  0  2  14 5  |  9  0  13 0  |
|  0  5  7  14 |  6  0  2  9  |  3  12 0  0  |  16 15 4  11 |
-----------------------------------------------------------
|  1  0  0  6  |  14 4  7  11 |  0  5  2  16 |  8  0  0  0  |
|  0  0  12 2  |  16 0  0  10 |  8  1  6  0  |  0  0  11 0  |
|  4  0  14 5  |  0  8  0  13 |  0  10 12 3  |  7  0  0  0  |
|  0  16 10 8  |  0  6  1  12 |  0  0  0  0  |  0  0  0  0  |
-----------------------------------------------------------

Solução
-----------------------------------------------------------
|  11 6  16 15 |  5  1  8  2  |  12 9  3  10 |  4  14 7  13 |
|  14 10 9  7  |  13 12 11 3  |  4  8  5  2  |  1  6  16 15 |
```

```
|  3  1  2  12 |  4  10 6  7  |  14 13 16 15 |  11 9  5  8  |
|  8  13 5  4  |  9  14 15 16 |  1  6  7  11 |  2  3  12 10 |
-----------------------------------------------------------------
|  7  2  8  10 |  1  3  14 5  |  16 4  13 12 |  6  11 15 9  |
|  6  12 1  9  |  10 7  16 15 |  5  11 8  14 |  13 2  3  4  |
|  5  14 13 3  |  11 2  4  8  |  9  7  15 6  |  12 1  10 16 |
|  16 4  15 11 |  12 9  13 6  |  2  3  10 1  |  14 7  8  5  |
-----------------------------------------------------------------
|  2  3  4  13 |  8  15 12 14 |  11 16 9  7  |  10 5  1  6  |
|  12 9  6  16 |  7  11 5  1  |  10 15 4  13 |  3  8  14 2  |
|  15 8  11 1  |  3  16 10 4  |  6  2  14 5  |  9  12 13 7  |
|  10 5  7  14 |  6  13 2  9  |  3  12 1  8  |  16 15 4  11 |
-----------------------------------------------------------------
|  1  15 3  6  |  14 4  7  11 |  13 5  2  16 |  8  10 9  12 |
|  13 7  12 2  |  16 5  3  10 |  8  1  6  9  |  15 4  11 14 |
|  4  11 14 5  |  2  8  9  13 |  15 10 12 3  |  7  16 6  1  |
|  9  16 10 8  |  15 6  1  12 |  7  14 11 4  |  5  13 2  3  |
-----------------------------------------------------------------

Tempo de execução em segundos: 3.4145393689996126
```

$N = 5$

[19]:
```
times[3][2] = timeit.timeit(stmt='test_sudoku(5,0.6)', setup='from __main__␣
 ↪import test_sudoku',number=1)
print("Tempo de execução em segundos:",times[3][2])
```

```
Sudoku
0   2   11  0   12  13  0   0   4   0   0   22  25  23  0   20  0   21  0   17
14  7   1   8   3
0   8   0   25  24  18  11  0   0   17  3   14  0   21  20  23  0   0   6   1
13  0   0   19  4
0   0   14  0   17  20  0   0   1   19  0   0   11  0   16  0   0   0   7   12
23  2   18  9   15
0   9   19  20  1   0   0   0   0   12  7   2   0   0   17  0   3   5   0   4
0   11  24  25  6
18  15  0   0   3   0   5   0   8   2   0   4   13  0   1   24  0   11  10  0
17  0   12  0   16
0   18  13  0   0   0   0   12  5   22  2   0   4   6   15  17  23  8   0   21
9   25  16  0   20
8   0   0   0   4   0   21  19  9   6   23  0   22  7   12  0   20  3   16  0
10  24  0   2   0
12  0   15  0   20  14  17  0   13  3   21  11  0   8   0   0   0   6   24  0
0   23  7   0   5
0   0   0   16  0   0   4   2   24  25  0   20  0   3   13  11  15  0   0   19
8   22  0   6   14
24  0   3   0   7   23  18  11  20  0   5   0   0   10  14  0   0   13  22  0
0   19  21  0   1
```

```
3   19  16  0   22  25  12  15  11  23  0   0   8   0   0   7   0   0   21  20
4   13  9   17  0
0   20  12  15  11  0   24  14  18  16  17  23  1   25  2   0   0   4   13  0
19  21  5   7   0
0   1   0   0   0   19  22  21  2   0   13  3   0   0   0   0   0   0   9   24
0   14  0   18  0
0   0   0   0   13  4   7   8   3   5   16  19  0   20  0   14  2   0   0   0
24  0   6   22  0
5   7   2   24  8   0   0   0   0   13  0   18  21  0   22  0   11  0   15  10
3   0   20  12  23
0   24  0   18  0   11  20  1   0   0   0   0   3   2   6   0   5   0   0   16
7   8   23  0   25
19  23  20  8   2   12  0   3   14  15  1   5   9   17  0   21  0   25  0   7
16  6   22  10  0
0   0   0   6   5   0   0   0   23  0   11  0   0   13  10  4   12  0   0   0
15  0   3   0   0
0   13  4   12  0   16  8   5   10  21  14  7   20  0   18  3   0   24  0   15
11  0   2   1   17
1   0   7   11  16  0   0   18  0   9   8   15  23  0   0   0   17  10  0   0
5   12  4   0   21
20  12  0   23  0   0   6   0   0   0   19  0   14  0   7   0   21  15  0   0
2   0   8   0   9
4   25  5   13  14  0   0   0   0   0   20  12  24  0   8   16  7   0   0   0
6   0   19  11  0
0   0   17  0   9   0   19  10  0   24  0   0   6   0   21  8   0   0   0   18
1   0   25  23  0
0   0   0   0   15  8   0   0   25  11  18  9   0   4   0   19  0   17  2   0
0   0   0   24  0
21  11  8   0   0   9   15  0   0   14  25  10  2   1   0   0   0   23  5   22
20  17  13  16  0
```

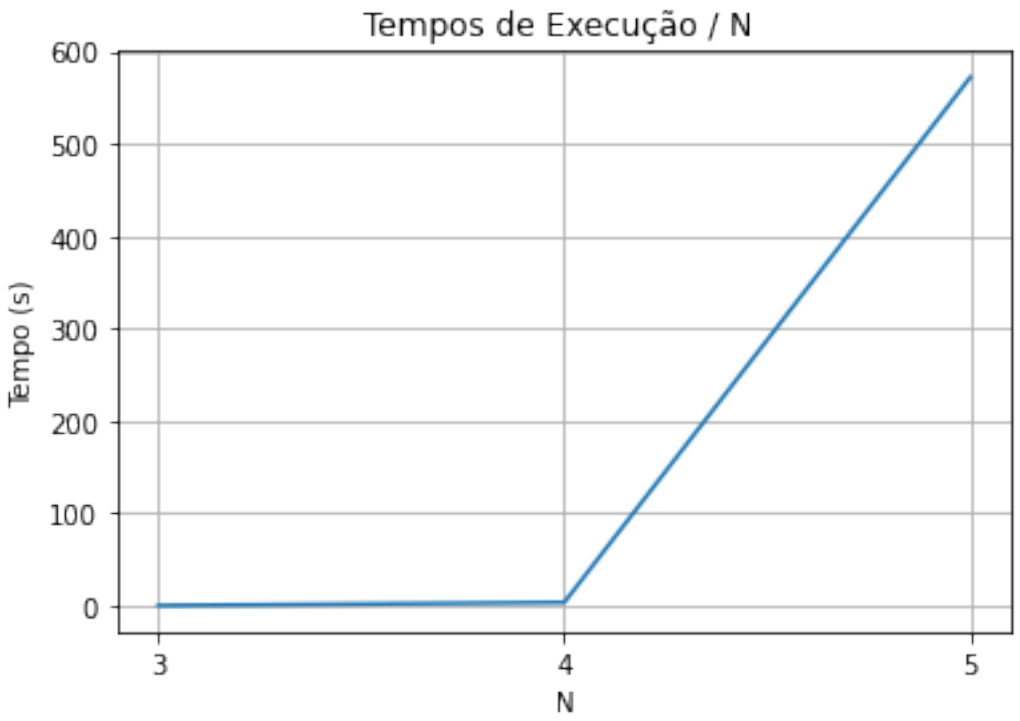
Solução

```
6   2   11  5   12  13  9   24  4   10  15  22  25  23  19  20  16  21  18  17
14  7   1   8   3
16  8   22  25  24  18  11  7   15  17  3   14  12  21  20  23  9   2   6   1
13  5   10  19  4
10  4   14  21  17  20  3   6   1   19  24  8   11  5   16  13  25  22  7   12
23  2   18  9   15
13  9   19  20  1   21  14  23  16  12  7   2   10  18  17  15  3   5   8   4
22  11  24  25  6
18  15  23  7   3   22  5   25  8   2   6   4   13  9   1   24  19  11  10  14
17  20  12  21  16
11  18  13  1   19  7   10  12  5   22  2   24  4   6   15  17  23  8   14  21
9   25  16  3   20
8   14  25  17  4   15  21  19  9   6   23  1   22  7   12  18  20  3   16  5
10  24  11  2   13
12  22  15  9   20  14  17  16  13  3   21  11  19  8   25  1   10  6   24  2
18  23  7   4   5
```

```
23  5   10  16  21  1   4   2   24  25  9   20  18  3   13  11  15  7   12  19
8   22  17  6   14
24  6   3   2   7   23  18  11  20  8   5   16  17  10  14  9   4   13  22  25
12  19  21  15  1
3   19  16  14  22  25  12  15  11  23  10  6   8   24  5   7   1   18  21  20
4   13  9   17  2
9   20  12  15  11  10  24  14  18  16  17  23  1   25  2   6   22  4   13  3
19  21  5   7   8
17  1   6   4   23  19  22  21  2   20  13  3   7   12  11  5   8   16  9   24
25  14  15  18  10
25  21  18  10  13  4   7   8   3   5   16  19  15  20  9   14  2   12  17  23
24  1   6   22  11
5   7   2   24  8   17  1   9   6   13  4   18  21  14  22  25  11  19  15  10
3   16  20  12  23
15  24  9   18  10  11  20  1   17  4   12  21  3   2   6   22  5   14  19  16
7   8   23  13  25
19  23  20  8   2   12  13  3   14  15  1   5   9   17  4   21  18  25  11  7
16  6   22  10  24
14  17  21  6   5   24  2   22  23  7   11  25  16  13  10  4   12  9   1   8
15  18  3   20  19
22  13  4   12  25  16  8   5   10  21  14  7   20  19  18  3   6   24  23  15
11  9   2   1   17
1   3   7   11  16  6   25  18  19  9   8   15  23  22  24  2   17  10  20  13
5   12  4   14  21
20  12  24  23  18  3   6   13  22  1   19  17  14  16  7   10  21  15  25  11
2   4   8   5   9
4   25  5   13  14  2   23  17  21  18  20  12  24  15  8   16  7   1   3   9
6   10  19  11  22
2   16  17  3   9   5   19  10  12  24  22  13  6   11  21  8   14  20  4   18
1   15  25  23  7
7   10  1   22  15  8   16  20  25  11  18  9   5   4   23  19  13  17  2   6
21  3   14  24  12
21  11  8   19  6   9   15  4   7   14  25  10  2   1   3   12  24  23  5   22
20  17  13  16  18

Tempo de execução em segundos: 571.9829137320012
```

Gráfico Tempo de Execução / N

[21]:
```
plt.plot(N, times[3])
plt.xticks(N)
plt.title("Tempos de Execução / N ")
plt.ylabel("Tempo (s)")
plt.xlabel("N")
plt.grid(True)
```



2.1.13 Nota

Para $N = 5$ a função demora varios minutos a executar, pelo que para $N \geq 5$ ainda vai ser mais lento, o que deve ser provocado pelo ao facto de as soluções terem complexidade exponencial em N. Se fizermos uma analise ao número de variaveis em função de N verificamos que: - Para inicializar o sudoku temos $N^6 - N$ variaveis. - Para resolver o sudoku temos $N^6 \times (1 - \alpha)$ variaveis.