

## ▼ PG47089 | Carlos Preto | MEI

### 1 Futoshiki Puzzle

Futoshiki é um puzzle lógico japonês jogado num tabuleiro  $N \times N$ , onde são assinaladas restrições de desigualdade entre algumas posições contíguas do tabuleiro.

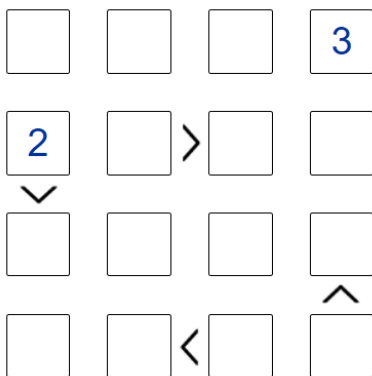
O objetivo é colocar os números  $1..N$  de forma a que cada número não apareça repetido em cada linha nem em cada coluna do tabuleiro, e que as relações de desigualdade assinaladas sejam respeitadas. Alguns números podem estar fixos no tabuleiro inicial.

Desenvolva um programa em Python para resolver este jogo como auxílio de um SMT solver.

- **Input:** a configuração do tabuleiro inicial deverá ser fornecida num ficheiro de texto, em formato que entendam adquado para o descrever. Como alternativa, pode receber o texto do tabuleiro diretamente numa string.
- **Output:** a solução do puzzle deverá ser impressa no ecrã.

### Resolução


Antes de desenvolver o programa em *Python*, é preciso de entender a composição de um tabuleiro do *Futoshiki* puzzle. Como se pode observar, podem existir restrições de desigualdade, quer entre elementos da mesma linha, quer entre elementos de cada coluna. Para além de tais restrições, verifica-se que já podem haver variáveis com valores predefinidos, como é o caso dos valores 2 e 3 na imagem.



Como foi pedido pelos docentes, o programa deverá aceitar quer ficheiros, quer *strings*. O programa desenvolvido dispõe de mecanismos que permitem ao utilizador indicar quais destes dois métodos pretende para ler o puzzle.

## Input de Ficheiro

O utilizador deverá introduzir o nome do ficheiro que pretende que seja lido. O formato do ficheiro deverá ser igual ao apresentado na figura, onde na primeira linha se coloca o tamanho do puzzle e nas restantes são colocadas as variáveis e as suas relações de desigualdade.

 puzzle - Bloco de notas

Ficheiro Editar Formatar Ver Ajuda

```
4
x00 - x01 - x02 - x03
> - - - - -
x10 - x11 - x12 - x13
- - - - -
4 - x21 < x22 - 2
- - - - - <
x30 - x31 < x32 - x33
```

## Input de String

O utilizador deverá introduzir a string manualmente. Esta deve estar devidamente construída e com a presença de `\n` sempre que se tratar de uma nova linha do tabuleiro. Mais uma vez, o primeiro valor da string deverá representar o tamanho do tabuleiro.

Exemplo: `4\n4 - x01 - x02 - x03\n- - - - -\nx10 - x11 - x12 - x13\n- - - - - < - -\nx20 - x21 - x22 - 3\n- - - - -\nx30 - x31 < x32 - 2`

## Restrições

Uma vez acedido ao tabuleiro, será necessário definir as diferentes restrições das nossas variáveis, de maneira a se obter uma solução final. Antes de partir para as restrições, é importante notar que se criou um dicionário denominado **x** onde se irão armazenar as variáveis que irão sendo lidas. Para melhor perceber as restrições implementadas, consideremos as variáveis *i*, *j* como as posições às quais estamos a aceder no puzzle.

- Em primeiro lugar será necessário declarar as variáveis do puzzle.  
`x[i][j] = Int('x'+str(i)+str(j))`
- O valor das variáveis terá de estar compreendido entre 1 e N, sendo N o tamanho do puzzle.  
`s.add(And(1<=x[i][j], x[i][j]<=N))`
- Caso a variável já tenha um valor, então é necessário fazer a respetiva atribuição.  
`s.add(x[i][j] == l[j])`, onde `l[j]` contém o valor dessa mesma variável.
- Posteriormente adicionam-se as restrições de desigualdade. Caso o sinal de ">" ou "<" esteja numa linha onde há variáveis, então será necessário adicionar a restrição entre a variável que está na coluna antes do símbolo e a que está na coluna depois do símbolo.  
`s.add(x[i][j-1] > x[i][j+1])` ou `s.add(x[i][j+1] > x[i][j-1])`
- No caso de estarmos numa linha onde não variáveis e apenas há símbolos, então temos de adicionar restrições de desigualdade entre as variáveis que estão na mesma coluna que o símbolo de ">" ou "<", mas que estão na linha imediatamente antes e depois.  
`s.add(x[i-1][j] > x[i+1][j])` ou `s.add(x[i+1][j] > x[i-1][j])`

- Por fim, é importante indicar que variáveis na mesma linha ou coluna terão de ter valores obrigatoriamente diferentes(**k** irá corresponder ao valor da próxima linha ou coluna).

```
s.add(Distinct(x[i][j], x[i][k]))
```

```
s.add(Distinct(x[j][i], x[k][i]))
```

## Programa Python

```
pip install z3-solver
```

```
from z3 import *
```

```
#caso seja ficheiro
```

```
def read_file():
```

```
    file = input("Insira o nome do ficheiro: ")
```

```
    f = open(file, "r")
```

```
    levels = int(f.readline())
```

```
    s = Solver()
```

```
    x = {}
```

```
    line = 0
```

```
#decarar variáveis e seus limites
```

```
for i in range(2*levels-1):
```

```
    column = 0
```

```
    l = f.readline().rstrip().split(" ")#remover "\n" e separar variáveis
```

```
    x[line] = {}
```

```
    for j in range(len(l)):
```

```
        if (i%2 == 0):
```

```
            if (l[j] != "-"):
```

```
                if (l[j] != "<" and l[j] != ">"):
```

```
                    if not l[j].isnumeric():
```

```
                        x[line][column] = Int('x'+str(line)+str(column))
```

```
                        s.add(And(1<= x[line][column], x[line][column]<=levels)) #decl
```

```
                        column+=1
```

```
                    else:
```

```
                        x[line][column] = Int('x'+str(line)+str(column))
```

```
                        s.add(x[line][column] == l[j]) #tipo x00 == 2
```

```
                        column+=1
```

```
            else:
```

```
                column+=1
```

```
        if (i%2 == 0):
```

```
            line+=1
```

```
#voltar ao início do ficheiro
```

```
f.seek(0)
```

```
f.readline()
```

```
#condições de desigualdade
```

```
for i in range(2*levels-1):
```

```
    l = f.readline().rstrip().split(" ")#remover "\n" e separar variáveis
```

```

for j in range(2*levels-1):
    if (i%2 == 0):
        if (l[j] == ">"):
            i1 = int(i/2)
            j1 = int((j-1)/2)
            j2 = int((j+1)/2)
            s.add(x[i1][j1] > x[i1][j2])
        else:
            if (l[j] == "<"):
                i1 = int(i/2)
                j1 = int((j-1)/2)
                j2 = int((j+1)/2)
                s.add(x[i1][j2] > x[i1][j1])
    else:
        if (l[j] == ">"):
            j1 = int(j/2)
            i1 = int((i-1)/2)
            i2 = int((i+1)/2)
            s.add(x[i1][j1] > x[i2][j1])
        else:
            if (l[j] == "<"):
                j1 = int(j/2)
                i1 = int((i-1)/2)
                i2 = int((i+1)/2)
                s.add(x[i2][j1] > x[i1][j1])

#restrições de linha
for i in range(len(x)):
    for j in range(len(x[i])-1):
        for k in range(len(x)):
            if (j != k):
                s.add(Distinct(x[i][j], x[i][k]))#restrições de linha
                s.add(Distinct(x[j][i], x[k][i]))#restrições de coluna

if s.check() == sat:
    m = s.model()
    print(m)
else:
    print("\nUNSAT")

#caso seja string
def read_string():

    tabuleiro = input('Insira o tabuleiro em forma de String:')
    li = tabuleiro.replace(r'\n', '\n').split("\n")
    levels = int(li[0])

    lines = tabuleiro.replace(r'\n', '\n').split("\n")[1:]#retirar o level

    s = Solver()
    x = {}
    line = 0

    #decarar variáveis e seus limites

```

```

for i in range(2*levels-1):
    column = 0
    l = lines[i].rstrip().split(" ")#remover "\n" e separar variáveis
    x[line] = {}
    for j in range(len(l)):
        if (i%2 == 0):
            if (l[j] != "-"):
                if (l[j] != "<" and l[j] != ">"):
                    if not l[j].isnumeric():
                        x[line][column] = Int('x'+str(line)+str(column))
                        s.add(And(1<= x[line][column], x[line][column]<=levels)) #decl
                        column+=1
                    else:
                        x[line][column] = Int('x'+str(line)+str(column))
                        s.add(x[line][column] == l[j]) #tipo x00 == 2
                        column+=1
            else:
                column+=1
        if (i%2 == 0):
            line+=1

#condições de desigualdade
for i in range(2*levels-1):
    l = lines[i].split(" ")#remover "\n" e separar variáveis
    for j in range(2*levels-1):
        if (i%2 == 0):
            if (l[j] == ">"):
                i1 = int(i/2)
                j1 = int((j-1)/2)
                j2 = int((j+1)/2)
                s.add(x[i1][j1] > x[i1][j2])
            else:
                if (l[j] == "<"):
                    i1 = int(i/2)
                    j1 = int((j-1)/2)
                    j2 = int((j+1)/2)
                    s.add(x[i1][j2] > x[i1][j1])
        else:
            if (l[j] == ">"):
                j1 = int(j/2)
                i1 = int((i-1)/2)
                i2 = int((i+1)/2)
                s.add(x[i1][j1] > x[i2][j1])
            else:
                if (l[j] == "<"):
                    j1 = int(j/2)
                    i1 = int((i-1)/2)
                    i2 = int((i+1)/2)
                    s.add(x[i2][j1] > x[i1][j1])

#rstrições de linha
for i in range(len(x)):
    for j in range(len(x[i])-1):
        for k in range(len(x)):
            if (j != k):

```

```
s.add(Distinct(x[i][j], x[i][k]))#restrições de linha  
s.add(Distinct(x[j][i], x[k][i]))#restrições de coluna
```

```
if s.check() == sat:  
    m = s.model()  
    print(m)  
else:  
    print("\nUNSAT")
```

```
while True:  
    option = input("1 - Ler Ficheiro | 2 - Ler String: ")  
    if (option == "1"):  
        read_file()  
        break  
    else:  
        if(option == "2"):  
            read_string()  
            break  
        else:  
            print("Opção inválida!")
```

