UNIVERSIDADE FEDERAL DE SANTA CATARINA

CAMPUS Trindade

DEPARTAMENTO INE

CURSO Ciências da Computação

Carlos Eduardo Vitorino Gomes (23150560)

Eduardo Cunha Cabral (23150561)

Enrico Caliolo (23150562)

Paradgimas de Programação: Trabalho 2

Introdução

O problema escolhido pelo grupo foi o Kojun. Esse puzzle é composto por um grid separado por regiões. A tarefa é encaixar números na região, respeitando três restrições:

- Cada região deve ter números de 1 até N (sendo N o tamanho da região)
- Números em célular ortogonalmente adjacentes não podem ser iguais.
- Se duas células são verticalmente adjacentes em uma mesma região, a de cima deve ser maior que a de baixo.

Modelagem do problema

Para armazenar os valores iniciais do grid e as regiões, usamos uma lista de listas, representadas por funções. O usuário deverá mudar essas funções para resolver um determinado problema. O programa imprime a solução ao terminar, caso haja.

```
(defun kojun-board ()
  '((5 0 2 0 2 0 3 1 3 1)
    (0 4 0 1 0 5 0 5 0 4)
    (7 5 1 7 0 0 3 1 3 0)
    (0 4 0 0 0 0 0 0 0 3)
    (2 0 3 4 0 2 0 0 4 0)
    (5 0 2 0 6 0 0 0 0 0)
    (0 1 3 0 1 0 0 4 0 3)
    (6 7 0 3 0 1 4 0 0 1)
    (4 0 3 0 4 0 0 0 0 3)
    (0 1 0 2 0 6 2 0 2 1)))

(defun kojun-regions ()
  '((01 02 02 02 03 03 03 03 04 04)
    (01 01 01 02 05 05 06 06 04 06)
    (07 07 01 05 05 08 09 06 06 06)
    (07 07 05 05 10 08 09 09 09 11)
    (07 07 07 05 10 10 12 11 11 11)
    (13 13 14 14 14 10 15 15 16 16)
    (13 13 13 14 14 18 19 20 16 16)
    (17 17 13 14 18 18 19 20 21 21)
    (17 17 22 22 22 22 19 20 20 23)
    (17 17 17 22 22 22 19 19 23 23)))
```

Porém, para usarmos as informações no código, transformamos as regiões em uma hash table, para facilitar o acesso das células de cada região em partes específicas do código. Os valores de chave da hashtable são os números das regiões, e os valores de cada uma são os índices das células de cada região.

O cerne do algoritmo está em uma função chamada `solve`, que irá usar a técnica ‘backtracking’ para resolver o Kojun.

```
(loop for num from 1 to (1- (length board)) do
    (if (is-valid board regions empty-cell-row empty-cell-col num)
        ; se for válido
        (progn
            (change-board-value board empty-cell-row empty-cell-col num)
            (if (solve board regions)
                (return-from solve t)
            )
            (change-board-value board empty-cell-row empty-cell-col 0))))
(return-from solve nil)
```

Entradas e Saídas de Dados

Os dados para o puzzle são colocados diretamente no código-fonte, como mostrado nas figuras abaixo. Os valores iniciais do puzzle e as regiões são representadas por uma lista de listas.

```
(defun kojun-board-10×10 ()
  "Valores iniciais do Kojun 10×10"
  '((5 0 2 0 2 0 3 1 3 1)
    (0 4 0 1 0 5 0 5 0 4)
    (7 5 1 7 0 0 3 1 3 0)
    (0 4 0 0 0 0 0 0 0 3)
    (2 0 3 4 0 2 0 0 4 0)
    (5 0 2 0 6 0 0 0 0 0)
    (0 1 3 0 1 0 0 4 0 3)
    (6 7 0 3 0 1 4 0 0 1)
    (4 0 3 0 4 0 0 0 0 3)
    (0 1 0 2 0 6 2 0 2 1))
)

(defun kojun-regions-10×10 ()
  "Regiões do Kojun 10×10"
  '((01 02 02 02 03 03 03 03 04 04)
    (01 01 01 02 05 05 06 06 04 06)
    (07 07 01 05 05 08 09 06 06 06)
    (07 07 05 05 10 08 09 09 09 11)
    (07 07 07 05 10 10 12 11 11 11)
    (13 13 14 14 14 10 15 15 16 16)
    (13 13 13 14 14 18 19 20 16 16)
    (17 17 13 14 18 18 19 20 21 21)
    (17 17 22 22 22 22 19 20 20 23)
    (17 17 17 22 22 22 19 19 23 23))
)
```

Como saída, nós recebemos o tabuleiro completo no terminal da seguinte forma:

```
Kojun 10x10
 5  3  2  4  2  4  3  1  3  1
 2  4  3  1  3  5  6  5  2  4
 7  5  1  7  1  2  3  1  3  2
 6  4  2  6  4  1  2  4  1  3
 2  1  3  4  3  2  1  2  4  1
 5  6  2  5  6  1  2  1  2  4
 4  1  3  4  1  3  5  4  1  3
 6  7  2  3  2  1  4  3  2  1
 4  2  3  5  4  7  3  2  1  3
 3  1  5  2  1  6  2  1  2  1
```

Dificuldades encontradas

Uma das dificuldades encontradas foi a questão de validar se um número poderia ser colocado em uma determinada célula, especificamente a questão de, numa mesma região, células verticais terem que ser em ordem ascendente. Demoramos um tempo para encontrar uma solução para essa questão. Inicialmente checávamos todas as linhas de uma coluna de uma mesma região, mas percebemos que poderíamos apenas checar as células de cima e de baixo e ver se o número seria válido.

Organização do grupo

Nos comunicamos pessoalmente, de modo assíncrono por meio de mensagens no Whatsapp e de maneira síncrona virtual por meio da ferramenta Discord, todos os membros do grupo colaboraram para a execução do trabalho. O código foi feito de maneira assíncrona, de modo que 1 dos membros iniciou o trabalho e os seguintes continuaram de maneira assíncrona. O relatório foi feito de maneira síncrona virtual.