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)

Paradigmas da Programação: Trabalho 1

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 células 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.

Solução Escolhida

A entrada se encontra no arquivo Games.hs, onde há uma separação entre os valores de cada célula e as regiões que elas pertencem. Os valores do tabuleiro que devem ser descobertos pelo usuário, ou seja, as células vazias, são representados pelo número 0. Já as regiões, são representadas primeiramente por um índice utilizado para diferenciá-las e coordenadas de linha e coluna das células que pertencem a determinada região. Abaixo segue o código de um tabuleiro 6x6:

```haskell
board_6x6 :: Board
board_6x6 =
  [ [2, 0, 0, 0, 1, 0],
    [0, 0, 0, 3, 0, 0],
    [0, 3, 0, 0, 5, 3],
    [0, 0, 0, 0, 0, 0],
    [0, 0, 3, 0, 4, 2],
    [0, 0, 0, 0, 0, 0]
  ]

regions_6x6 :: Regions
regions_6x6 =
  [ (1, [(0, 0), (0, 1)]),
    (2, [(0, 2), (0, 3), (0, 4)]),
    (3, [(0, 5), (1, 5)]),
    (4, [(1, 0), (1, 1), (1, 2), (1, 3), (1, 4), (2, 4)]),
    (5, [(2, 0), (3, 0), (3, 1), (3, 2)]),
    (6, [(2, 1), (2, 2), (2, 3), (3, 3)]),
    (7, [(2, 5), (3, 5), (3, 4)]),
    (8, [(4, 0), (4, 1)]),
    (9, [(5, 0), (5, 1)]),
    (10, [(4, 2), (5, 2), (5, 3)]),
    (11, [(4, 3), (4, 4), (4, 5), (5, 4), (5, 5)])
  ]
```

A solução escolhida foi a de backtracking, onde procuramos um valor adequado para determinada célula vazia, repetindo até encontrarmos esse determinado valor, as funções abaixo representam essa recursividade, onde ela será chamada para cada valor possível:

```haskell
-- Função que resolve o tabuleiro de jogo
solve :: Board -> Regions -> Maybe Board
solve board regions = case findEmptyCell board of
  Nothing -> Just board -- Se não há células vazias, retorna o tabuleiro resolvido
  Just emptyCell -> solveCell board regions emptyCell [1 .. length board] -- Tenta resolver a célula vazia

-- Função auxiliar para resolver uma célula específica
solveCell :: Board -> Regions -> Cell -> [Int] -> Maybe Board
solveCell board regions emptyCell [] = Nothing -- Se não há mais números para tentar, retorna Nothing
solveCell board regions emptyCell (num : nums) =
  if isValid board regions emptyCell num
    then
      let newBoard = updateBoard board emptyCell num
       in case solve newBoard regions of
            Just solvedBoard -> Just solvedBoard -- Se resolver o tabuleiro com sucesso, retorna-o
            Nothing -> solveCell board regions emptyCell nums -- Continua tentando com os outros números
    else solveCell board regions emptyCell nums -- Continua tentando com os outros números
```

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

```
[2,1,3,2,1,2]
[1,4,2,3,6,1]
[4,3,4,2,5,3]
[3,1,2,1,2,1]
[1,2,3,5,4,2]
[2,1,2,1,3,1]
--------------------
```

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.

Dificuldades Encontradas

Nossa primeira dificuldade se deu em pensar em uma solução para o Kojun, de início nós já esperávamos que seria por meio de backtracking, porém, como nunca havíamos implementado algo semelhante, demoramos um pouco para conseguir fazer. Nossa segunda dificuldade se deu em nos adaptar com a linguagem, pois foi nosso primeiro contato com Haskell.