Alunos: Eduardo de Moraes e Pedro Henrique Dadalt de Queiroz;

Trabalho II - Lisp

Definição Kojun

Kojun é um jogo que consiste no preenchimento de uma matriz de células, delimitado por grupos de tamanho e formato variado e que respeita certas condições. Cada grupo possui células valoradas de 1 até a quantidade de células presentes nesse grupo. Cada célula deve obedecer a regra de não conter o mesmo valor no mesmo grupo e nem em células verticalmente e horizontalmente ortogonais, ou seja, não pode ser igual aos vizinhos imediatos da esquerda, direita, cima e baixo. Além disso, se duas células são verticalmente adjacentes na mesma área, o valor da célula de cima deve ser maior que o valor da célula de baixo, exemplo presente na imagem a seguir.

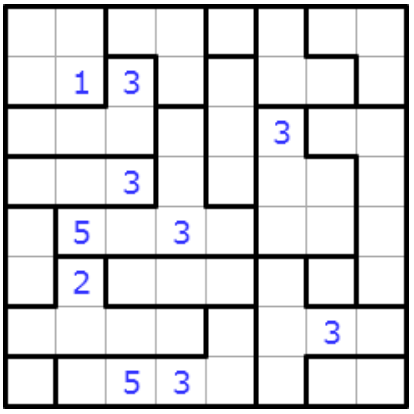

Imagem 1

Para dar início às análises que permitem o preenchimento completo da matriz, o jogo geralmente é inicializado contendo algumas células valoradas.

Modelagem do trabalho

O passo-a-passo para instalação de dependências e execução do projeto estão no *README.md*. Dessa vez foi utilizado como base a resolução por *backtracking*, que consiste em encontrar uma solução após cobrir todas as possibilidades, e portanto, encontrar a resposta por força bruta.

Como para encontrar todas as possibilidades é necessário obter muita informação por simples acesso a certos índices de vetores e matrizes, foi escolhido uma biblioteca externa chamada "*iterate*" com o intuito de facilitar justamente esse exercício trabalhoso de iterar várias estruturas e ainda ter que comparar valores, índices, etc.

Dessa vez o código foi todo pensado em adicionar o tamanho da matriz em alguma estrutura e depois apenas acessar e utilizar esse valor estático. Ainda sobre valores estáticos, o input do problema se dá em colocar *hardcoded* a tabela com os valores pré fixados e uma tabela com o identificador das áreas.

Organização do trabalho

A organização do trabalho consistiu na participação coletiva de todos os membros nas etapas que formam a tarefa: código fonte e relatório. Os integrantes reuniram seus conhecimentos acerca da linguagem Lisp e gerenciamento de pacotes da mesma linguagem, visando encontrar uma ideia em comum acordo que tivesse a capacidade de resolver o problema proposto pelo trabalho. Cada integrante buscou apoio em soluções já implementadas que ajudassem a resolver o Kojun.

Dificuldades

Na verdade o código não era para resolver o Kojun por *backtracking*, era para ser usado um "algoritmo guloso". Inicialmente, foi pensado para com que a resolução fosse feita etapa-por-etapa como um ser humano faria, ou seja, iria ser observado qual célula possuía a menor quantidade de possibilidades, desejavelmente com uma possibilidade apenas, e assim iria encontrando de forma certeira uma-por-uma. Observa-se que essa resolução só daria certo se o tabuleiro tivesse apenas uma resolução. Apesar de boa ideia, a implementação não se deu tão trivial.

Houve uma grande dificuldade de *debugar* o problema, não tinha como saber se as iterações estavam corretas e se estavam retornando os valores corretos. Comparado com o Trabalho I, apesar de ter sido difícil *debugar* também, ele foi mais fácil de definir as listas, as iterações facilitadas no contexto de *list comprehension* do Haskell foram um fator decisor para um trabalho funcionar completamente e outra nem tanto.