Universidade de São Paulo – USP
Instituto de Ciências Matemáticas e de Computação – ICMC
Departamento de Ciências de Computação – SCC

SCC5900 – Projeto de Algoritmos

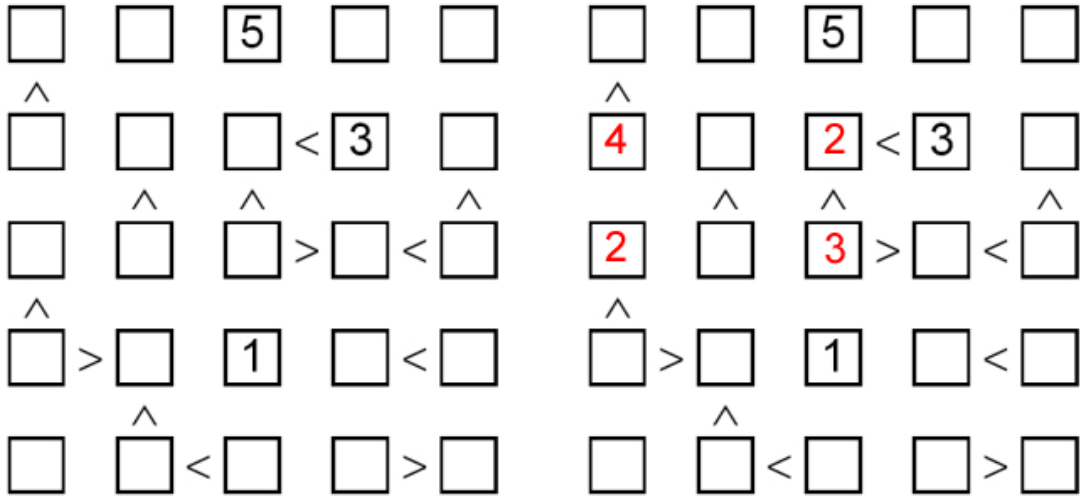
Denis Reis (PAE)
denismr@gmail.com

Professor Gustavo Batista
gbatista@icmc.usp.br

Projeto – Backtracking

Data de entrega: 20/04

Este projeto é individual. A entrega deve ser realizada via TIDIA, fazendo *upload* dos arquivos na pasta "Projeto 1" na ferramenta escaninho.



O **Futoshiki** é um jogo de tabuleiro quadrado cujo objetivo é preencher o tabuleiro com números, tal que um dígito não se repete em uma mesma linha ou em uma mesma coluna. Alguns dígitos podem ser fornecidos no tabuleiro inicial, assim como símbolos de maior (>) e menor (<). Esses símbolos indicam se um quadrado deve ter um valor maior ou menor que seu vizinho. O objetivo deste projeto é implementar um sistema que preenche todas as posições do tabuleiro. Este projeto possui três partes descritas a seguir:

Parte I – Implementação (25% da nota final)

Esta parte do projeto consiste em implementar um algoritmo força bruta para encontrar uma solução para o tabuleiro do Futoshiki, utilizando algumas das heurísticas discutidas em aula. A implementação deve seguir as seguintes diretrizes:

1. A implementação deve ser capaz de trabalhar com diferentes dimensões de tabuleiro, os quais possuem **D** linhas e **D** colunas, sendo $4 \leq \mathbf{D} \leq 9$. As células do tabuleiro devem ser preenchidas com números inteiros entre 1 e **D**.

2. A implementação deve ser capaz de utilizar *backtracking* simples ou com duas composições de heurísticas de poda:

    a. *Backtracking* simples, sem poda;

    b. *Backtracking* com verificação adiante. Backtracking é realizado quando uma variável fica sem nenhum valor disponível;

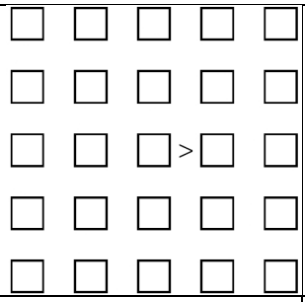
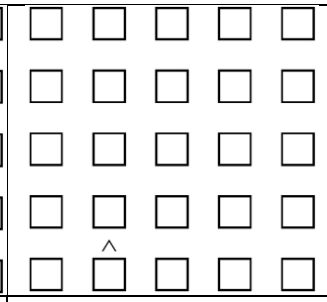
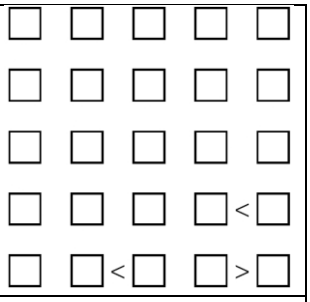
    c. *Backtracking* com verificação adiante e mínimos valores remanescentes. Idem ao anterior com a adição da heurística MVR para decidir a próxima variável.

3. A sua implementação deve ser única com *flags* que permitem ligar/desligar cada uma das heurísticas citadas.

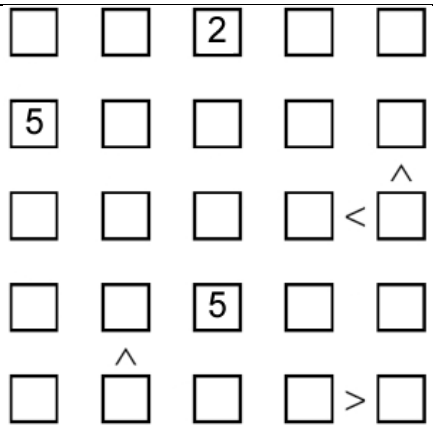
Parte II – Avaliação de Funcionamento (45% da nota final)

Esta avaliação consiste em avaliar o correto funcionamento do seu programa. Para isso o programa deve ler uma entrada e imprimir uma saída em formatos rígidos descritos a seguir. Cada uma das estratégias de *backtracking* e poda será avaliada individualmente. Assim, a corretude de cada uma delas vale um terço do total de pontos dessa parte da avaliação, ou seja, 15% da nota final.

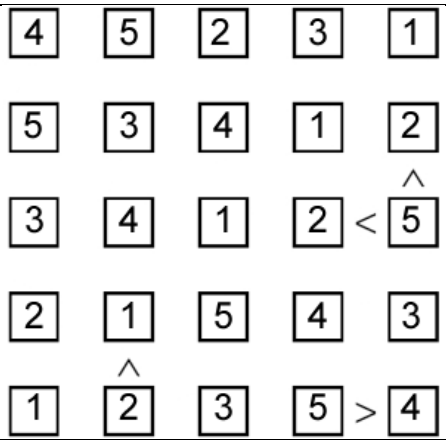
O programa deve ler uma entrada na qual a primeira linha contem o número de casos de teste **N**. Cada caso de teste consiste de uma sequência de linhas. A primeira linha de um caso de teste possui a dimensão **D** do tabuleiro e o número de restrições **R**, separados por um espaço. Assim sendo, as próximas **D** linhas contém os valores iniciais do tabuleiro, as quais indicam a configuração inicial do jogo. Valores 0 indicam posições em banco e valores diferentes de zero indicam valores pré-definidos no tabuleiro inicial. Após a leitura das **D** linhas que descrevem a configuração inicial do tabuleiro, o programa deverá ler as próximas **R** linhas, sendo que cada uma das **R** linhas indica uma restrição no tabuleiro. Cada restrição é composta por quatro valores inteiros que indicam onde a restrição ocorre **L1**, **C1**, **L2** e **C2**. O par **L1** e **C1** indica que o valor da célula na posição (L1, C1) deverá ser menor que valor contido na célula da posição (L2, C2). Sendo $1 \leq \mathbf{L}, \mathbf{C} \leq \mathbf{D}$ e a posição (1, 1) é localizada no canto superior esquerdo. Na figura abaixo são mostrados alguns exemplos de restrições.

Tabuleiro			
Mapeamento da Restrição	3 4 3 3	4 2 5 2	4 4 4 5 5 2 5 3 5 5 5 4

Desse modo, abaixo é apresentado um tabuleiro com sua respectiva entrada.

Tabuleiro Inicial	Entrada
	1 5 4 2 5 3 5 3 4 3 5 4 2 5 2 5 5 5 4 0 0 2 0 0 5 0 0 0 0 0 0 0 0 0 0 0 5 0 0 0 0 0 0 0

A **saída** do seu programa deve imprimir uma solução para cada caso de teste, na mesma ordem fornecida pela entrada. Deve ser apresentada em uma matriz de valores inteiros de 1 a **D**, separados por um espaço. Por exemplo, uma solução possível para o tabuleiro acima é:

Tabuleiro Final	Saída
	1 4 5 2 3 1 5 3 4 1 2 3 4 1 2 5 2 1 5 4 3 1 2 3 5 4

Parte III – Avaliação de Desempenho (30% da nota final)

A avaliação de desempenho consiste em comparar os tempos de execução do algoritmo de força bruta simples e com o uso de heurísticas de poda para um conjunto de casos de teste.

Neste projeto o desempenho será avaliado por dois critérios: o número de atribuições a variáveis realizadas durante toda a busca até se encontrar uma primeira solução e o tempo de execução do algoritmo. Utilize o número de atribuições para avaliar a efetividade das podas, e o tempo de execução para avaliar o esforço adicional causado por elas.

Uma vez que número de atribuições pode ser muito grande para as podas mais fracas, o seu programa pode abortar a busca quando o número de atribuições exceder 10^6, mas apenas se for necessário. Nesse caso, imprima a mensagem "**Numero de atribuicoes excede limite maximo**".

Faça um curto relatório (2 páginas) explicando a sua implementação e os resultados obtidos na Parte III.