

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

SCC-0218 – Algoritmos Avançados e Aplicações

Professor Gustavo Batista
gbatista@icmc.usp.br
Estagiários PAE Diego Silva e Denis Reis
diegofsilva@usp.br e denismr@usp.br

Projeto – Backtracking

Data de entrega: 16/09

	6		1		4		5		9	6	3	1	7	4	2	5	8
		8	3		5	6			1	7	8	3	2	5	6	4	9
2								1	2	5	4	6	8	9	7	3	1
8			4		7			6	8	2	1	4	3	7	5	9	6
		6				3			4	9	6	8	5	2	3	1	7
7			9		1			4	7	3	5	9	6	1	8	2	4
5								2	5	8	9	7	1	3	4	6	2
		7	2		6	9			3	1	7	2	4	6	9	8	5
	4		5		8		7		6	4	2	5	9	8	1	7	3

Este projeto deve ser realizado em grupos de dois alunos. A entrega deve ser realizada via TIDIA, fazendo upload dos arquivos na pasta de um dos integrantes do grupo na ferramenta escaninho.

O objetivo deste projeto é implementar um sistema que resolve quebra-cabeças Sudoku. 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 quebra-cabeça Sudoku utilizando algumas das heurísticas discutidas em aula. A implementação deve seguir as seguintes diretrizes:

1. A implementação deve ser capaz de trabalhar com qualquer tabuleiro de 3 dimensões. Um tabuleiro dessa dimensão possui 9 linhas e 9 colunas, bem como 9 quadrados de 3 x 3 internamente. As células de cada um desses 9 quadrados de 3 x 3 devem ser preenchidas com todos os números inteiros entre 1 e 9.

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 que consiste em uma linha com o número de casos de teste m . As próximas 9 linhas possuem cada uma 9 valores inteiros separados por um espaço em branco. Os valores zero indicam que a posição está sem valor associado. Segue um exemplo:

1
0 6 0 1 0 4 0 5 0
0 0 8 3 0 5 6 0 0
2 0 0 0 0 0 0 0 1
8 0 0 4 0 7 0 0 6
0 0 6 0 0 0 3 0 0
7 0 0 9 0 1 0 0 4
5 0 0 0 0 0 0 0 2
0 0 7 2 0 6 9 0 0
0 4 0 5 0 8 0 7 0

A saída do seu programa deve ter uma solução possível para o problema de entrada. Nessa solução, cada linha, coluna e quadrado interno deve ser preenchido por valores distintos entre 1 e 9. Imprima uma linha em branco após cada caso de teste. Por exemplo:

9	6	3	1	7	4	2	5	8
1	7	8	3	2	5	6	4	9
2	5	4	6	8	9	7	3	1
8	2	1	4	3	7	5	9	6
4	9	6	8	5	2	3	1	7
7	3	5	9	6	1	8	2	4
5	8	9	7	1	3	4	6	2
3	1	7	2	4	6	9	8	5
6	4	2	5	9	8	1	7	3

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.

Medir o desempenho em termos de tempo de execução utilizando um relógio pode levar a resultados que variam com a linguagem de programação, computador e outros fatores. Neste projeto o desempenho será avaliado pelo número de atribuições a variáveis realizadas durante toda a busca até se encontrar uma primeira solução.

Uma vez que número de atribuições pode ser muito grande para as podas mais fracas, o seu programa deve abortar a busca quando o número de atribuições exceder 10^6 , mas apenas se for necessário.

Faça um curto relatório (1 a 2 páginas) explicando a sua implementação e os resultados obtidos na Parte III.

Este projeto é inspirado no problema “Su Doku” disponível em:

<http://uva.onlinejudge.org/external/9/989.html>