Trabalho Prático 3

Algoritmos I

Entrega: 22/11/2019

1 Introdução

O Sudoku é um *puzzle* de lógica baseado, na sua versão tradicional, em números. É composto por um "quadrante numérico" de linhas e colunas com algumas células pré-preenchidas. A sua resolução consiste em preencher cada célula de modo a não haver números repetidos em cada linha, coluna, ou subquadrante.

Um Sudoku de ordem $n$consiste de um arranjo de células dispostas em uma grade $n^2 \times n^2$ particionadas em regiões de células dispostas em uma subgrade $n \times n$, que possuem como objetivo o preenchimento de todas as células com elementos de um conjunto de símbolos — normalmente números em $\{1, 2, \ldots, n^2\}$ — de tal forma que em cada linha, coluna e região contenha somente um elemento do conjunto. A versão mais popular de Sudokus são os de ordem $n = 3$, onde dispõem-se números de 1 a 9 em uma grade 9×9.

Sudokus são considerados um caso especial de Quadrados Latinos, problema onde se deve alocar n^2 símbolos em uma grade de $n^2 \times n^2$ sem repetições em uma linha ou coluna. A diferença entre o Sudoku e Quadrado Latino é a restrição de não haver repetições de símbolos nas regiões $n \times n$. Uma instância de Sudoku é uma grade parcialmente completada com valores iniciais, comumente chamadas "pistas". As regiões podem também ser chamadas de "blocos". Células horizontalmente ajdacentes são chamadas de "linha" ou "banda" e verticalmente adjacentes são chamadas de "colunas" ou "pilha". Na Figura 1 temos um exemplo de uma instância de um Sudoku de ordem $n = 3$ e sua solução.

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

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

Figura 1: Uma instância de Sudoku e sua resolução.

Apesar da definição apresentada ser a de um Sudoku clássico, existem versões que não são um quadrado perfeito, como a de grade 8×8, que será considerada neste trabalho, no qual cada quadrante é da forma 4×2 (ver Figura 2). Um Sudoku é denominado próprio se ele possui somente uma única solução. Sudokus próprios são resolúveis logicamente, ou seja, a solução pode ser encontrada seguindo apenas uma cadeia lógica de passos, sem a necessidade de *backtracking*. Sudokus não-próprios requerem que escolhas arbitrárias

sejam feitas e armazenadas para que, caso encontre-se um ponto sem solução, as escolhas sejam desfeitas, reiniciando o processo de busca da solução; ou seja, realizar um *backtracking* nas escolhas.

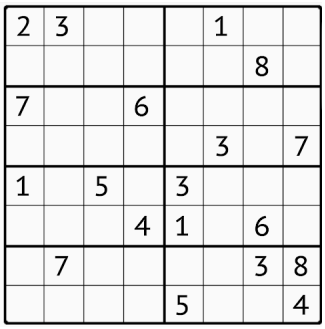

Figura 2: Exemplo sudoku de grade 8×8

A título de exemplo, pode-se deduzir a solução de um Sudoku considerando a instância dada pela Figura 1. É possível ver que a célula da sétima linha e sexta coluna — destacada na imagem — necessita conter o símbolo 6, uma vez que todos os símbolos de 1 a 5 e de 7 a 9 aparecem na linha, coluna e região adjacentes a célula. Se a instância Sudoku é própria (como de fato essa é), preencher a célula de única escolha leva outras a células a possuírem uma única escolha, permitindo eventualmente que se encontre a solução.

2 O que fazer?

Além do fato de Sudokus serem um caso especial de Quadrados Latinos, o Sudoku é um problema que pode ser transformado em um problema de Coloração de Grafos. Este problema é **NP-completo**. Para resolver uma instância Sudoku como uma Coloração de Grafos é preciso primeiramente representá-la como um grafo. Das diversas formas de realizar essa modelagem, a mais utilizada é considerar que cada célula do Sudoku é representada como um vértice no grafo conforme Figura 3. Então, para cada vértice de célula, adicionamos as arestas entre tal vértice todos os outros vértices que a célula possui uma relação de conflito: linha, coluna e quadrante.

Para solucionar uma instância de Sudoku de ordem n como uma coloração de vértices em S, não se busca o menor número k de cores possíveis, uma vez que já se conhece que $\chi(S) = n^2$. Tem-se então um problema de localização de uma coloração $\mathcal{C}$ que satisfaça S.

O objetivo deste trabalho é, dada uma instância do Sudoku de tamanho $n \times n$, desenvolva um algoritmo que informe se o Sudoku possui ou não solução e apresente a solução. Algumas observações importantes:

1. Você pode utilizar qualquer heurística para resolver o problema;
2. É importante ressaltar também que, sendo o problema de coloração de grafos um problema NP-Completo, os algoritmos heurísticos não garantem uma solução exata para a resolução do sudoku, ou seja, dependendo do caminho de solução tomada pela heurística do algoritmo, pode ser que a coloração não resolva o *puzzle*;
3. O tamanho de instância máxima será 9×9.

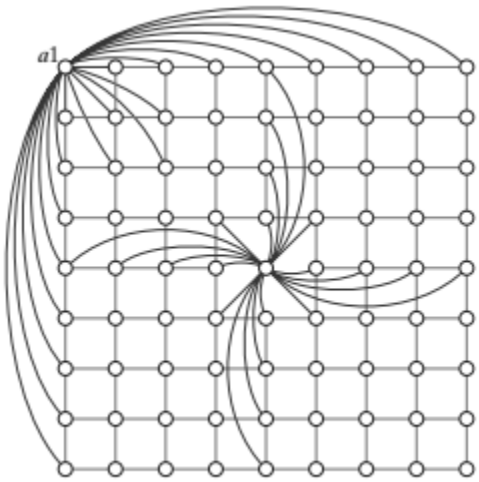

Figura 3: Representação do Sudoku como um grafo.

3 Os arquivos de entrada

Na primeira linha são dados três inteiros, em que N representa o tamanho (N x N) da tabela do *Sudoku*, I representa a quantidade de colunas de cada quadrante e J representa a quantidade de linhas de cada quadrante. Nas próximas NxN linhas, é dada a instância do *Sudoku*. Conforme mencionado na seção anterior, o tamanho máximo de instância será 9x9.

Arquivo de entrada

```
4 2 2                                   // <tamanho N x N > <I> <J>
0 4 0 1                                 // <instância>
3 0 0 0
0 0 0 4
0 0 0 0


8 4 2                                   // <tamanho N x N > <I> <J>
0 0 0 0 0 3 2 0                         // <instância>
7 0 0 3 0 4 0 5
0 2 0 0 0 8 4 1
4 0 8 0 0 0 0 0
0 0 6 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 5 4 1 0 0 0
2 0 0 8 0 0 3 0
```

4 O arquivo de saída

Caso o seu algoritmo deu uma solução válida, na primeira deve ser impressa "solução" e as seguintes NxN linhas devem conter a solução encontrada pela heurística adotada. Caso seu algoritmo não resultou em uma solução válida imprima "sem solução" e as seguintes NxN linhas devem conter a solução que o seu algoritmo resultou (mesmo que errada e/ou incompleta). O terceiro exemplo é uma ilustração de como deve ser a resposta quando a solução encontrada não for válida, ela não representa nenhuma entrada apresentada anteriormente.

Saída Esperada

```
solução                                 // <resultado>
2 4 3 1                                 // <solução>
3 1 4 2
1 3 2 4
4 2 1 3

solução                                 // <resultado>
5 8 4 1 7 3 2 6                         // <solução>
7 1 2 3 6 4 8 5
3 2 7 6 5 8 4 1
4 6 8 5 2 1 7 3
1 4 6 7 3 2 5 8
6 7 3 2 8 5 1 4
8 3 5 4 1 7 6 2
2 5 1 8 4 6 3 7

sem solução                             // <resultado>
2 4 3 1                                 // <solução>
3 3 4 2
0 0 0 0
1 0 0 3
```

5 Avaliação Experimental

Considerando que será implementada uma heurística para solucionar o Sudoku. Para a avaliação experimental o aluno deverá realizar as seguintes atividades:

1. Análise de pior caso do tempo de execução e da quantidade de memória utilizada pelo algoritmo.
2. Avaliação experimental da complexidade do algoritmo, guardando os tempos de execução e colocando esses valores em gráficos para melhor visualização. Para isso, execute o algoritmo proposto para solucionar o problema mais de uma vez para entradas com o mesmo tamanho do sudoku (no mínimo 10 vezes), guardando o tempo de execução e reportando a **média** e o **desvio padrão** deste tempo. Faça isso para vários tamanhos de sudoku (por exemplo: 4x4, 8x8, 9x9).
3. Considerando os resultados obtidos pela heurística adotada para solucionar a instância do *puzzle* e os respectivos tempos para os mesmos, avalie as saídas da heurística adotada para responder a

seguinte pergunta: Quais os formatos de tabela do Sudoku (4x4, 9x9, etc..) a heurística adotada obteve melhores soluções.

5.1 Média e Desvio Padrão

Conforme requerido no item acima, para a avaliação experimental é necessário que o aluno faça o cálculo da médio e o cálculo do desvio padrão do tempo de execução.

Para o cálculo da média (μ), utilize a seguinte fórmula:

$$\mu = \frac{\sum_{n=1}^{k} t_n x^n}{K} \tag{1}$$

Onde n é o número da execução, t_n representa o tempo de cada execução e K o número total de execuções.

Para o cálculo do desvio padrão (σ), utilize a seguinte fórmula:

$$\sigma = \sqrt{\frac{\sum_{n=1}^{k} |t_n - \mu|^2}{K-1}} \tag{2}$$

Onde n é o número da execução, t_n representa o tempo de cada execução, μ representa a média do tempo de execuções e K o número total de execuções.

6 O que deve ser entregue

Você deve submeter um arquivo compacto (**zip**) no formato **seu_nome_sua_matrícula** via Moodle contendo:

- todos os arquivos do código *.c*, *.cpp* e *.h* que foram implementados,
- um arquivo *makefile*[1] **que crie um executável tp3**,
  - **ATENÇÃO:** O makefile é para garantir que o código está sendo compilado corretamente, de acordo com o modo que vocês modelaram o programa. É **essencial** que ao digitar "make" na linha de comando dentro da pasta onde reside o arquivo makefile, o mesmo compile o programa e gere um executável chamado **tp3**.
- Sua documentação.

Sua documentação deve ter até 10 páginas contendo:

- Uma breve descrição do problema,
- Explicações sobre as heurísticas adotadas. Explique porquê optou-se por determinada heurística. Além disso, forneça uma breve explicação da heurística. Para tal, artifícios como pseudo-códigos, exemplos ou diagramas podem ser úteis. Note que documentar uma solução não é o mesmo que documentar seu código. Não inclua textos de códigos em sua documentação.
- análise de complexidade da solução proposta (espaço e tempo). Cada complexidade apresentada deverá ser devidamente justificada para que seja aceita.

[1] https://pt.wikibooks.org/wiki/Programar_em_C/Makefiles

- avaliação experimental.
- Conclusão do trabalho.

O seu TP deverá ser entregue de acordo com a data especificada no moodle. A penalidade em porcentagem para os TPs atrasados é dada pela fórmula $2^{d-1}/0.16$.

7 Implementação

7.1 Linguagem, Ambiente e Parâmetros

O seu programa deverá ser implementado nas linguagens **C** ou **C++** e poderá fazer uso de funções da biblioteca padrão da linguagem. Trabalhos que utilizem qualquer outra linguagem de programação e/ou que façam uso de bibliotecas que não a padrão não serão aceitos. Além disso, certifique-se que seu código compile e execute corretamente nas máquinas **Linux** dos laboratórios do **DCC**.

ATENÇÃO: O arquivo da entrada deve ser passado como parâmetros para o programa através da linha de comando (e.g., *$ ./tp3 sudoku.txt*) e imprimir a saída no **stdout** (com printf), não em um arquivo.

ATENÇÃO: certifique-se que o programa execute com os três arquivos passados como exemplos junto com a documentação no Moodle (dataset_tp3.zip), sem alterar os arquivos passados. Isso vai garantir que seu programa vai conseguir ler corretamente os arquivos do corretor automático.

7.2 Testes

A sua implementação passará por um processo de correção automática, o formato da saída de seu programa deverá ser idêntico aquele descrito nas seções anteriores. Saídas diferentes serão consideradas erro para o programa. Para auxiliar na depuração do seu trabalho, será fornecido um pequeno conjunto de entradas e suas saídas. É seu dever certificar que seu programa atenda corretamente para qualquer entrada válida.

7.3 Qualidade do código

É importante prestar atenção para a qualidade do código, mantendo-o organizado e comentado para não surgir dúvidas na hora da correção. Qualquer decisão que não estiver clara dada a documentação e a organização do código será descontada na nota final.

8 Consideração Final

Assim como em todos os trabalhos dessa disciplina é estritamente proibido a cópia parcial ou integra de códigos, seja da internet ou de colegas. Seja honesto! Você não aprende nada copiando código de terceiros. Se a cópia for detectada, sua nota será zerada e o professor será informado para que as devidas providências sejam tomadas.

- Caso haja qualquer dúvida, favor enviar mensagens no fórum criados para o trabalho.
- Caso o(a) aluno(a), não tenha acesso aos laboratórios favor entrar em contato com as monitoras via moodle.