

Instruções Gerais

- Esta atividade deve ser resolvida **individualmente**.
- Os itens teóricos devem resolvidos de forma organizada, clara e formal.
- A solução encontrada deve ser submetida, em um único arquivo PDF, no moodle. Certifique-se de que todas as resoluções digitalizadas estão legíveis antes de submetê-las.
- Entregas após o prazo estabelecido no moodle serão desconsideradas.
- É permitida a consulta a livros e outros materiais, mas a atividade apenas pode ser discutida com a equipe de ensino.
- Os algoritmos desenvolvidos nos itens práticos devem ser organizados e comentados. Todos os códigos utilizados devem ser submetidos como anexos no moodle.
- Qualquer tentativa de fraude, se detectada, implicará na reprovação (com nota final 0.0) de todos os envolvidos, além das penalidades disciplinares previstas no Regimento Geral da Unicamp (Arts. 226 – 237).

Apresentação

Diversos jogos e *puzzles* podem ser modelados como problemas de otimização inteira ou combinatória. Isso se dá porque o conjunto de decisões disponíveis em cada passo do jogo é finito. Mesmo os jogos mais simples, no entanto, estão entre os problemas mais desafiadores do ponto de vista computacional.

Questões

- **Questão 1:** Os alunos de uma universidade adoram um jogo que consiste em um tabuleiro com peças brancas e pretas. O objetivo do jogo é tornar todas as peças pretas. A única operação possível é a de selecionar uma peça, que terá a sua cor invertida. Esta operação, no entanto, inverte também as cores das peças adjacentes (acima, abaixo e aos lados – **sem modificar as peças vizinhas nas diagonais**). Veja, por exemplo, a figura abaixo.

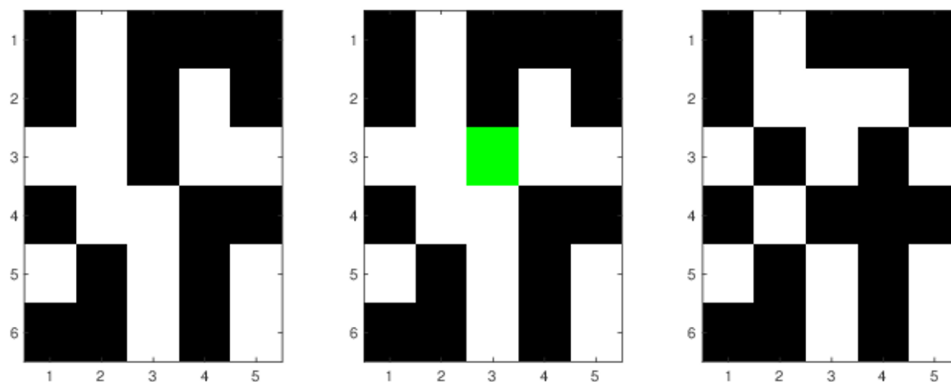


Figura 1: Exemplo do Jogo.

Você decide implementar um **jogador (artificial) ótimo** para resolver este passatempo; por ótimo, entende-se que o seu jogador deve realizar o menor número de inversões possível. O seu programa recebe, como entrada, uma matriz M , $m \times n$, que representa o tabuleiro ($m_{ij} = 1$ se a peça for preta e $m_{ij} = 0$ se a peça for branca). Por exemplo, para

o primeiro tabuleiro da Figura acima, a matriz M é dada por

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

A saída do seu programa deve ser uma matriz X , também de dimensão $m \times n$, tal que $x_{ij} = 1$ se esta peça for selecionada para inversão de cores ou $x_{ij} = 0$ caso contrário. Para ajudá-lo a checar o resultado, você pode usar a função MATLAB/Octave/*python* `tabuleiro(M,X)`, que recebe a matriz original e a matriz de seleções e simula um jogo em tempo real com a estratégia fornecida. Observe que a ordem de inversão de cores não afeta o resultado final.

- **Questão 2: (SUDOKU)** Sudoku é um jogo americano (muito famoso no Japão) que consiste em um tabuleiro 9×9 , onde cada célula deve conter um número de 1 a 9. O jogo normalmente se inicia com alguns destes valores já postos no tabuleiro, e o objetivo é completar as outras células de forma que cada uma das 9 linhas, cada uma das 9 colunas, e cada um dos 9 blocos principais 3×3 possua todos os dígitos de 1 a 9. Um exemplo de tabuleiro inicial e resolvido se encontra na Figura abaixo:

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

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

Figura 2: Exemplo de Sudoku - (E) início; (D) - resolvido.

Crie um programa baseado em otimização inteira que resolva (caso possível) qualquer instância de Sudoku. Seu programa deve receber uma matriz 9×9 de entrada, contendo as dicas iniciais do tabuleiro (e valor 0 para as posições vazias), e fornecer como saída esta matriz preenchida conforme as regras do Sudoku caso seja possível.

Dica: Apesar de ser um problema apenas de factibilidade, definir alguma função objetivo (mesmo que aleatória) pode ajudar os métodos de *branch-and-bound* e assim resolver mais facilmente este problema.

- **Questão 3: (SENHA)** Duas crianças brincam com o seguinte jogo: uma delas deve abrir um cadeado com uma senha de **três dígitos distintos** e a outra dá cinco dicas de senha suficientes para que o código seja descoberto, conforme ilustrado na Figura 3. Tomando este exemplo como base, construa um modelo e um programa baseados em otimização inteira que resolva este problema. O que aconteceria se as dicas não forem suficientes? O que aconteceria se as dicas forem conflitantes? Crie um exemplo para cada caso e verifique como o seu programa se comporta.

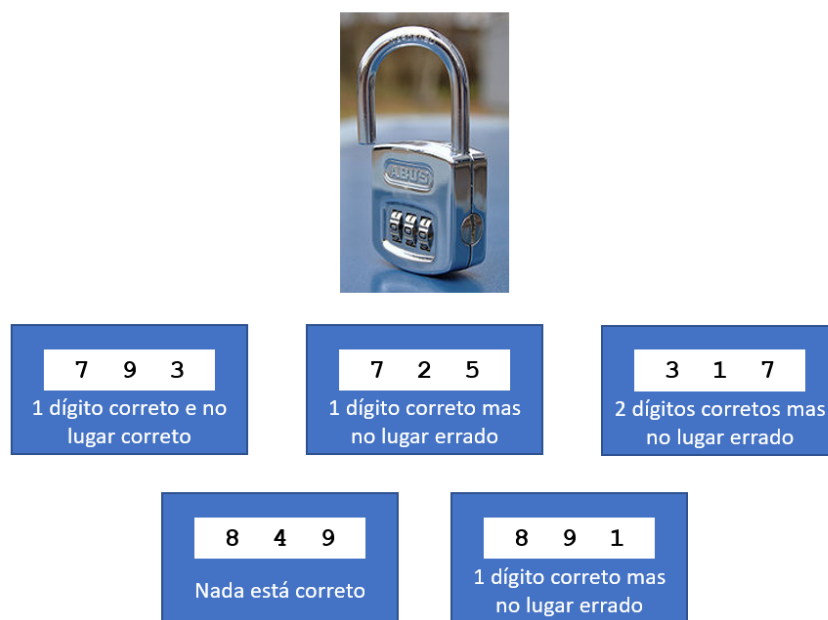


Figura 3: Exemplo do jogo SENHA.