Departamento de Informática - UEM
Disciplina: Modelagem e Otimização Algorítmica
2ª Avaliação de 2020
Nome: _____

Descrição.
Implementação do algoritmo A* para a resolução do Jogo do Tabuleiro de 15 peças. A implementação deverá permitir entrar com qualquer estado inicial e o objetivo é encontrar a menor sequência de movimentos que leve ao estado final do jogo. Estado final desejado é apresentado a seguir:

Estado final desejado:

1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	

Estado final desejado:

Heurísticas a serem implementadas e analisadas:

1. $h'_1(n)$ = número de peças foras de seu lugar na configuração final.
2. $h'_2(n)$ = número de peças fora de ordem na sequência numérica das 15 peças, seguindo a ordem das posições no tabuleiro.
3. $h'_3(n)$ = para cada peça fora de seu lugar somar a distância Manhattam (quantidade de deslocamentos) para colocar em seu devido lugar. Neste caso considera-se que o caminho esteja livre para fazer o menor número de movimentos.
4. $h'_4(n) = p_1*h'1(n) + p_2*h'2(n) + p_3*h'3(n)$, sendo $p_1 + p_2 + p_3$ são pesos (número real) tais que $p_1 + p_2 + p_3 = 1$. A escolha desses pesos deverão ser escolhidos conforme os resultado dos experimentos.
5. $h'_5(n) = \max(h'_1(n), h'_2(n), h'_3(n))$.

A avaliação se dará em duas partes, sendo que cada parte corresponde a 50% da nota.
Parte 1 (individual**)**: Esta parte da avaliação deve ser realizada individualmente. Esta parte consiste na submissão e correção dos trabalhos pelo sistema Run.Code (https://run.codes/). O sistema só aceita código nas seguintes linguagens de programação: **Java 8, C/C++, Python 3, Pascal, e Fortran90**. Detalhes de como utilizar o sistema pode ser obtidos em support@run.codes. Cadastre-se no sistema com o código: 31PQ. Fique atento ao sistema, pois esta parte será dividida em algumas fases com submissão em diferentes datas. O sistema Run.Codes realiza tanto testes de execução de seu código como, também, avaliação de similaridade do seu código fonte com uma base de outros códigos. Portanto, se for constatado plágio de código, seja de outro colega ou de qualquer outra fonte, a nota do trabalho será penalizada. Nesta parte do trabalho, você terá que escolher a heurística que desejar visando solucionar os casos em menor tempo e memória possível. Pode fazer tantas submissões quanto desejarem ao Run.Codes até a data final de submissão. Os casos deixados como testes, como o nome diz, são para testes apenas e não servem como avaliação final. A avaliação será posterior ao fechamento da submissão com outras instâncias (casos) diferentes. Então faça a submissão com versão (heurística) que julgar mais eficiente e eficaz. Será avaliada a capacidade de resolução e uso dos recursos tempo e memória.

Parte 2 (equipe). Esta parte pode ser realizada por equipes de até dois membros. Esta parte consiste de um relatório impresso, incluindo a análise do comportamento das 5 propostas de heurísticas de todos os casos disponibilizados pelo professor no Moodle. Nesta análise deve levar em consideração o consumo de memória e o tempo de processamento. Considerar todos os casos disponibilizados no *Moodle*. Incluir no relatório a informação computador, SO e linguagem utilizada, além de informar a estrutura de dados escolhida para representar os conjuntos A e F. Avalie, também, se as linhas 9 e 10 do algoritmo A* (versão II) teve alguma influência nos resultados. Nesta avaliação, também, será considerado o relatório de atividades de todas as aulas práticas.

Prazo para entrega: 12/04/2020 .

Exemplo de como calcular a heurística h'$_2$.

5	0	9	13
1	2	10	14
3	6	7	11
4	8	12	15

Primeiro passo considerar a sequências de "posições" no tabuleiro conforme a sequência "numérica" esperada para o estado final, ou seja, iniciando na posição [1,1] (linha 1 e coluna 1) e terminando na posição [4,4] (linha 4, coluna 4).

A partir da tabela acima, teremos a seguinte sequência:

5	1	3	4	0	2	6	8	9	10	7	12	13	14	11	15

Agora é verificada a sequência numérica. As posições em amarelo são posições contabilizadas para esta heurística. Conforme se observa, as ocorrências contabilizadas são estas:

1. Peça 1 após a peça 5, enquanto esperava a peça 6;
2. Peça 3 após a peça 1;
3. Espaço vazio após a peça 4;
4. Peça 6 após a peça 2;
5. Peça 8 após a peça 6;
6. Peça 7 após a peça 10;
7. Peça 12 após a peça 7;
8. Peça 11 após a peça 15;
9. Peça 15 após a peça 11.

Note que após o espaço vazio não é feita a contabilização.

Portanto, para esta caso h'2=9.

Sugestões de ordem de implementação:

1. Escolher a como representar a informação de cada nó (vértice), incluindo o tabuleiro (estado), o custo G, o custo h' e o ponteiro para o nó pai. Lembrando que cada nó é criado e armazenado de forma dinâmica em memória.
2. Representação dos conjuntos A e F (dois conjuntos principais utilizados pelo algoritmo A*) Lista? Árvores? Árvore *heap*? *Hash*? Escolha a estrutura que julgar mais eficiente do ponto de vista computacional para fazer as operações de busca, inserção e remoção.
3. Função *GeraSucessor*(*nó_pai*). Função responsável por receber o nó (estado) escolhido pelo algoritmo A* e gerar todos os estados (filhos) sucessores a partir dele.
4. Implemente a heurística 1 primeiro e teste seu algoritmo. Esta é a heurística mais simples de ser implementada. Uma vez constatado que a sua implementação está funcionando, então faça a implementação das demais heurísticas.
5. Algoritmo A*.
6. Testar o algoritmo com as primeiras instâncias fornecidas.
7. Implementar as heurísticas 2, 3, 4 e 5. Testar as heurísticas.
   Como podem notar, o algoritmo A* propriamente é uma das últimas tarefas de implementação, embora seja importante entendê-lo antes de todas as implementações.