Relatório 15 Puzzle

Rio de Janeiro - RJ

Universidade Federal do Rio de Janeiro

Xiao Yong Kong - 114176987
Breno Pontes da Costa - 114036496

Modelagem do problema

1 - Representação de estados do problema

Cada estado pode ser modelado como uma lista de tamanho 16 da forma:

[W1, W2, W3, W4, W5, W6, W7, W7, W8, W9, W10, W11, W12, W13, W14, W15, W16].

Onde, Wi, para 0 < i <=16, pode assumir os valores de 0 a 15 e o valor (*), que se refere a posição do tabuleiro vazia.

Sendo assim, temos como estado inicial : [15, 2, 1, 12, 8, 5, 6, 11, 4, 9, 10, 7, 3, 14, 13, *].

E como estado objetivo: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, *].

2 - Representação de em árvore do problema

Cada nó pode ser representado como um estado. As arestas do grafo serão representadas pela regra *operador*, como no exemplo:

operador([A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, *], esquerda, [A, B, C, D, E, F, G, H, I, J, K, L, M, N, *, O]).

3 - Algoritmo de busca A*

Semelhante ao algoritmo de busca genérico, porém com as seguintes alterações:

- O functor add_to_frontier será alterado de acordo com a heurística usada;

Sendo assim, o código da busca será:

```
search([Node | _]) :- is_goal(Node).
search([Node | F1]) :-
        neighbors(Node, NN),
        add_to_frontier(NN, F1, F2),
        search(h1, F2).
```

O functor add_to_frontier será baseado no número de peças fora do lugar, logo será:

```
add_to_frontier(NN,F1,F2):-
        append(F1, NN, F2),
        sort_by_h(F2, F3).
```

4 - Implementação da heurística

O código desenvolvido apenas utiliza a heurística h1 - número de peças fora do lugar.

A lógica de execução da heurística se encontra em sort_by_h e funciona da seguinte forma.

1 - Cria uma lista de estado com os seus respectivos custos da forma:
[[estado1, custo1], [estado2, custo2], ..., [estadoN, custoN]]

2 - Usa um algoritmo de insert sort para ordenar a lista anterior de forma crescente;

3 - Recupera apenas os estados (sem os custos), porém em uma lista ordenada por custo de forma decrescente;

4 - Inverte a lista de estados ordenados e sem custos para obter uma lista em ordem crescente.

5 - Observação

O código não chega a uma solução.

6 - Código

```
%Nó objetivo
is_goal([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, *]).

%Para definir os operadores possíveis para cada estado,
%tenha como índice no tabuleiro a forma: [Linha,Coluna].

% [1,4]
operador([A, B, C, *, D, E, F, G, H, I, J, K, L, M, N, O], esquerda, [A, B, *, C, D, E, F, G, H, I, J, K, L, M, N, O]).
operador([A, B, C, *, D, E, F, G, H, I, J, K, L, M, N, O], baixo, [A, B, C, G, D, E, F, *, H, I, J, K, L, M, N, O]).

% [1,3]
operador([A, B, *, C, D, E, F, G, H, I, J, K, L, M, N, O], direita, [A, B, C, *, D, E, F, G, H, I, J, K, L, M, N, O]).
operador([A, B, *, C, D, E, F, G, H, I, J, K, L, M, N, O], esquerda, [A, *, B, C, D, E, F, G, H, I, J, K, L, M, N, O]).
operador([A, B, *, C, D, E, F, G, H, I, J, K, L, M, N, O], baixo, [A, B, F, C, D, E, *, G, H, I, J, K, L, M, N, O]).

% [1,2]
operador([A, *, B, C, D, E, F, G, H, I, J, K, L, M, N, O], direita, [A, B, *, C, D, E, F, G, H, I, J, K, L, M, N, O]).
operador([A, *, B, C, D, E, F, G, H, I, J, K, L, M, N, O], esquerda, [*, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O]).
operador([A, *, B, C, D, E, F, G, H, I, J, K, L, M, N, O], baixo, [A, E, B, C, D, *, F, G, H, I, J, K, L, M, N, O]).

% [1,1]
% [*, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O]
operador([*, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O], direita, [A, *, B, C, D, E, F, G, H, I, J, K, L, M, N, O]).
operador([*, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O], baixo, [D, A, B, C, *, E, F, G, H, I, J, K, L, M, N, O]).


% [2,4]
operador([A, B, C, D, E, F, G, *, H, I, J, K, L, M, N, O], esquerda, [A, B, C, D, E, F, *, G, H, I, J, K, L, M, N, O]).
operador([A, B, C, D, E, F, G, *, H, I, J, K, L, M, N, O], baixo, [A, B, C, D, E, F, G, K, H, I, J, *, L, M, N, O]).
operador([A, B, C, D, E, F, G, *, H, I, J, K, L, M, N, O], cima, [A, B, C, *, E, F, G, D, H, I, J, K, L, M, N, O]).

% [2,3]
operador([A, B, C, D, E, F, *, G, H, I, J, K, L, M, N, O], direita, [A, B, C, D, E, F, G, *, H, I, J, K, L, M, N, O]).
operador([A, B, C, D, E, F, *, G, H, I, J, K, L, M, N, O], esquerda, [A, B, C, D, E, *, F, G, H, I, J, K, L, M, N, O]).
```

```
% [2,2]
operador([A, B, C, D, E, *, F, G, H, I, J, K, L, M, N, O], direita, [A, B, C, D, E, F, *, G, H, I, J, K, L, M, N, O]).
operador([A, B, C, D, E, *, F, G, H, I, J, K, L, M, N, O], esquerda, [A, B, C, D, *, E, F, G, H, I, J, K, L, M, N, O]).
operador([A, B, C, D, E, *, F, G, H, I, J, K, L, M, N, O], baixo, [A, B, C, D, E, I, F, G, H, *, J, K, L, M, N, O]).
operador([A, B, C, D, E, *, F, G, H, I, J, K, L, M, N, O], cima, [A, *, C, D, E, B, F, G, H, I, J, K, L, M, N, O]).

% [2,1]
operador([A, B, C, D, *, E, F, G, H, I, J, K, L, M, N, O], direita, [A, B, C, D, E, *, F, G, H, I, J, K, L, M, N, O]).
operador([A, B, C, D, *, E, F, G, H, I, J, K, L, M, N, O], baixo, [A, B, C, D, H, E, F, G, *, I, J, K, L, M, N, O]).
operador([A, B, C, D, *, E, F, G, H, I, J, K, L, M, N, O], cima, [*, B, C, D, A, E, F, G, H, I, J, K, L, M, N, O]).

%[3,4]
operador([A, B, C, D, E, F, G, H, I, J, K, *, L, M, N, O], esquerda, [A, B, C, D, E, F, G, H, I, J, K, *, K, L, M, N, O]).
operador([A, B, C, D, E, F, G, H, I, J, K, *, L, M, N, O], baixo, [A, B, C, D, E, F, G, H, I, J, K, O, L, M, N, *]).
operador([A, B, C, D, E, F, G, H, I, J, K, *, L, M, N, O], cima, [A, B, C, D, E, F, G, *, I, J, K, H, L, M, N, O]).

%[3,3]
operador([A, B, C, D, E, F, G, H, I, J, *, K, L, M, N, O], direita, [A, B, C, D, E, F, G, H, I, J, K, *, L, M, N, O]).
operador([A, B, C, D, E, F, G, H, I, J, *, K, L, M, N, O], esquerda, [A, B, C, D, E, F, G, H, I, *, J,  K, L, M, N, O]).
operador([A, B, C, D, E, F, G, H, I, J, *, K, L, M, N, O], baixo, [A, B, C, D, E, F, G, H, I, J, N, K, L, M, *, O]).
operador([A, B, C, D, E, F, G, H, I, J, *, K, L, M, N, O], cima, [A, B, C, D, E, F, *,H, I, J, G, K, L, M, N, O]).

%[3,2]
operador([A, B, C, D, E, F, G, H, I, *, J, K, L, M, N, O], direita, [A, B, C, D, E, F, G, H, I, J, *, K, L, M, N, O]).
operador([A, B, C, D, E, F, G, H, I, *, J, K, L, M, N, O], esquerda, [A, B, C, D, E, F, G, H, *, I, J, K, L, M, N, O]).
operador([A, B, C, D, E, F, G, H, I, *, J, K, L, M, N, O], baixo, [A, B, C, D, E, F, G, H, I, M, J, K, L, *, N, O]).
operador([A, B, C, D, E, F, G, H, I, *, J, K, L, M, N, O], cima, [A, B, C, D, E, *, G, H, I, F, J, K, L, M, N, O]).

%[3,1]
operador([A, B, C, D, E, F, G, H, *, I, J, K, L, M, N, O], direita, [A, B, C, D, E, F, G, H, I, *, J, K, L, M, N, O]).
operador([A, B, C, D, E, F, G, H, *, I, J, K, L, M, N, O], baixo, [A, B, C, D, E, F, G, H, L, I, J, K, *, M, N, O]).
operador([A, B, C, D, E, F, G, H, *, I, J, K, L, M, N, O], cima, [A, B, C, D, *, F, G, H, E, I, J, K, L, M, N, O]).


%[4,4]
operador([A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, *], esquerda, [A, B, C, D, E, F, G, H, I, J, K, L, M, N, *, O]).
operador([A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, *], cima, [A, B, C, D, E, F, G, H, I, J, K, *, M, N, O, L]).

%[4,3]
operador([A, B, C, D, E, F, G, H, I, J, K, L, M, N, *, O], direita, [A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, *]).
operador([A, B, C, D, E, F, G, H, I, J, K, L, M, N, *, O], esquerda, [A, B, C, D, E, F, G, H, I, J, K, L, M, *, N, O]).
operador([A, B, C, D, E, F, G, H, I, J, K, L, M, N, *, O], cima, [A, B, C, D, E, F, G, H, I, J, *, L, M, N, K, O]).

%[4,2]
operador([A, B, C, D, E, F, G, H, I, J, K, L, M, *, N, O], direita, [A, B, C, D, E, F, G, H, I, J, K, L, M, N, *, O]).
operador([A, B, C, D, E, F, G, H, I, J, K, L, M, *, N, O], esquerda, [A, B, C, D, E, F, G, H, I, J, K, L, *, M, N, O]).
operador([A, B, C, D, E, F, G, H, I, J, K, L, M, *, N, O], cima, [A, B, C, D, E, F, G, H, I, *, K, L, M, J, N, O]).

%[4,1]
operador([A, B, C, D, E, F, G, H, I, J, K, L, *, M, N, O], direita, [A, B, C, D, E, F, G, H, I, J, K, L, M, *, N, O]).
operador([A, B, C, D, E, F, G, H, I, J, K, L, *, M, N, O], cima, [A, B, C, D, E, F, G, H, *, J, K, L, I, M, N, O]).

% neighbors usa findall para buscar todos os nós filhos de um dado estado.
```

```
% neighbors usa findall para buscar todos os nós filhos de um dado estado.
neighbors([A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P], NN) :-
    findall([A_r, B_r, C_r, D_r, E_r, F_r, G_r, H_r, I_r, J_r, K_r, L_r, M_r, N_r, O_r, P_r], operador([A, B, C, D, E, F, G, H, I, J,
 K, L, M, N, O, P], _, [A_r, B_r, C_r, D_r, E_r, F_r, G_r, H_r, I_r, J_r, K_r, L_r, M_r, N_r, O_r, P_r]), NN).

% O código de busca A* foi baseado no código apresentado na aula em vídeo.
% A implementação da regra da heurística será feita dentro de sort_by_h.

search([Node | _]) :- is_goal(Node).
    search([Node | F1]) :-
    neighbors(Node, NN),
    add_to_frontier(NN, F1, F2),
    search(h1, F2).

add_to_frontier(NN,F1,F2):-
    sort_by_h(NN, F3),
    append(F1, F3, F2).



% ----- Lógica para execução de heurística ---------
% Este código implementa apenas a heurística de contagem de peças fora do lugar.
% Sendo assim, sort_by_h terá a seguinte forma.

% Implementação da heurística:
% 1 - Cria uma lista de estado com os seus respectivos custos da forma:
%       [[estado1, custo1], [estado2, custo2], ..., [estadoN, custoN]]
% 2 - Usa um algorítimo de insert sort para ordenar a lista anterior de forma crescente
% 3 - Recupera apenas os estados (sem os custos), porém em uma lista ordenada por custo
%       de forma decrescente.
% 4 - Inverte a lista de estados ordenados e sem custos para obter uma lista em ordem crescente
sort_By_h(L,SortedByH):-
    criaListaDeEstadosComH(L,ListaDeEstadosComCusto),
    insert_sort(ListaDeEstadosComCusto, sorted),
    recuperaListaDeEstados(sorted,ListaDeEstadosPorCustoDecrescente),
    reverse(ListaDeEstadosPorCustoDecrescente, SortedByH).

% Recebe uma lista de estados com seus custos e retorna apenas os estados.
% Devido o fluxo de execução do algoritimo, a lista retornada tem ordem invertida.
recuperaListaDeEstados([],[]).
recuperaListaDeEstados([[H,_]|T1],L):-
    recuperaListaDeEstados(T1,L1),
    append(L1,[H],L).



% Modificação do algoritmo de insert sort. Neste caso,
% é testado os custos dos estados.
insert_sort(List,Sorted):-i_sort(List,[],Sorted).
i_sort([],Acc,Acc).
i_sort([H|T],Acc,Sorted):-insert(H,Acc,NAcc),i_sort(T,NAcc,Sorted).

insert([X,TX],[[Y,TY]|T],[[Y,TY]|NT]):-TX>TY,insert([X,TX],T,NT).
insert([X,TX],[[Y,TY]|T],[[X,TX],[Y,TY]|T]):-TX=<TY.
insert(X,[],[X]).

% Recebe uma lista com estados e retorna uma lista da forma:
% [[estado1, custo1], [estado2, custo2], ..., [estadoN, custoN]]
criaListaDeEstadosComH([],[]).
criaListaDeEstadosComH([H|T],L):-
    calculaCustoAoObjetivo(H,[1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,*],Custo),
    criaListaDeEstadosComH(T,L1),
    append(L1,[[H,Custo]],L).

% Compara duas listas, elemento por elemento, e retorna a quantidade de diferenças
% encontradas.
calculaCustoAoObjetivo([],[], Custo):- Custo is 0.
calculaCustoAoObjetivo([H|T],[H1|T1], Custo):-
    H = H1,
    calculaCustoAoObjetivo(T, T1, Custo).
calculaCustoAoObjetivo([H|T],[H1|T1], Custo):-
    H \= H1,
    calculaCustoAoObjetivo(T, T1, Custo1),
```