

## Universidade Federal de São Carlos Campus São Carlos

# Inteligência Artificial

Alunos:

Guilherme Vilar Balduino - 743546 Roberto Yamamoto- 743594

Turma A

**Professor:** 

Murilo Naldi

## Trabalho de Inteligência Artificial – Aspirador de pó

O objetivo do trabalho é programar um aspirador em pó com inteligência artificial que seja capaz de coletar todos os lixos de um ambiente, jogá-los na lixeira e chegar até o dock station, encerrando o trabalho. O aspirador pode carregar até duas sujeiras e só pode andar para cima ou para baixo se houver um elevador.

Para solucionar o problema, inicialmente é necessário guardar o estado atual do aspirador. Esse estado é guardado da seguinte maneira:

- X corresponde a posição horizontal do aspirador;
- Y corresponde a posição vertical do aspirador;
- S corresponde ao número de sujeiras que o aspirador carrega no momento;
- F corresponde ao número de sujeiras restantes no ambiente;
- L corresponde à lista de posições onde estão as sujeiras restantes no ambiente, sendo a lista composta por listas no formato [X,Y];
- P corresponde à lista de posições onde estão as paredes no ambiente, sendo a lista composta por listas no formato [X,Y];
- E corresponde à lista de posições onde estão os elevadores no ambiente, sendo a lista composta por listas no formato [X];
- G corresponde à lista de posições onde está(ão) a(s) lixeira(s) no ambiente, sendo a lista composta por listas no formato [X,Y];
- D corresponde à lista contendo a posição do dock station no ambiente, sendo a lista composta por listas no formato [X,Y];
- LimX corresponde a posição limite horizontal do ambiente;
- LimY corresponde a posição limite vertical do ambiente;
- no\_fim corresponde a posição horizontal do aspirador, muda para fim quando as condições para finalizar o programa são atingidas.

Para solucionar o problema, foi criada uma sequência de regras que devem ser seguidas pelo aspirador. Cada regra contém uma sucessão de estados que ocorre somente se o aspirador obedece a um conjunto de

condições no estado em que se encontra ao passar pela regra durante a recursão.

## Lista de regras:

```
%Fim do programa
s([X,Y,S,F,[],P,E,G,D,LimX,LimY,no_fim],[X,Y,S,F,[],P,E,G,D,LimX,LimY,fim]):- S == 0, F == 0, pertence([X,Y],D), !.
%pega lixo
s([X,Y,S,F,Ll,P,E,G,D,LimX,LimY,no_fim],[X,Y,M,K,L2,P,E,G,D,LimX,LimY,no_fim]):- M is S+1, K is F-1, S < 2, F > 0, pertence([X,Y],L1), retirar_elemento([X,Y],L1,L2), !.
%joga lixo
s([X,Y,S,F,L,P,E,G,D,LimX,LimY,no_fim],[X,Y,M,F,L,P,E,G,D,LimX,LimY,no_fim]):- S > 0, M is 0, pertence([X,Y],G), !.
%ddreita
s([X,Y,S,F,L,P,E,G,D,LimX,LimY,no_fim],[W,Y,S,F,L,P,E,G,D,LimX,LimY,no_fim]):- W is X+1, not(pertence([W],LimX)), not(pertence([W,Y],F)).
%esquerda
s([X,Y,S,F,L,P,E,G,D,LimX,LimY,no_fim],[W,Y,S,F,L,P,E,G,D,LimX,LimY,no_fim]):- W is X-1, W >= 0, not(pertence([W,Y],F)).
%sobe
s([X,Y,S,F,L,P,E,G,D,LimX,LimY,no_fim],[X,Z,S,F,L,P,E,G,D,LimX,LimY,no_fim]):- Z is Y+1, not(pertence([Z],LimY)), pertence([X],E).
%dessee
s([X,Y,S,F,L,P,E,G,D,LimX,LimY,no_fim],[X,Z,S,F,L,P,E,G,D,LimX,LimY,no_fim]):- Z is Y-1, Z >= 0, pertence([X],E).
```

Regra para finalizar o programa: o aspirador não pode estar carregando nenhuma sujeira (S = 0) e não podem haver sujeiras no ambiente (F = 0). Além disso, o aspirador deve estar no dock station (local representado por D). Quando o programa acaba, o termo no\_fim é modificado para fim.

Regra para pegar lixo: o aspirador não pode estar carregando dois lixos e deve haver uma sujeira onde o aspirador está, ou seja, a posição [X,Y] da sujeira deve pertencer à lista L1. Quando essas condições são verdadeiras, o número de sujeiras que o aspirador está carregando aumenta (M is S+1), o número de lixos restantes diminui (K is F-1) e a lista [X,Y] é removida de L1 (retirar\_elemento([X,Y], L1, L2)). Dessa forma, os valores S, F e L1 são substituídas por M, K e L2 no novo estado, respectivamente.

Regra para jogar lixo: o aspirador deve estar na posição da lixeira e deve conter pelo menos um lixo armazenado (S > 0). No estado novo o aspirador não está armazenando nenhuma sujeira (M is 0), o estado novo substitui S por 0.

Regra para se mover: o aspirador se move para direita ou esquerda se não há paredes na posição almejada (not(pertence([W,Y], P))) ou se tal posição está entre 0 e a posição limite (LimX). No caso de movimentos para cima ou para baixo, deve haver um elevador onde o aspirador está (pertence([X], E)) e a posição almejada deve estar entre 0 e LimY. O novo estado substitui os

valores X e Y da posição para W e Z, respectivamente, porém o sucessor de um estado nunca mudará X e Y simultaneamente, ou seja, o aspirador não pode andar diagonalmente.

Para que o aspirador seja capaz de encontrar o caminho mais eficiente para concluir o seu objetivo, foi implementada uma função de busca em largura para buscar a solução. A busca em largura é uma busca cega que permite apenas a observação dos elementos vizinhos a um elemento já explorado. Essa busca é mais eficiente por apontar primeiro o elemento mais próximo em relação ao estado inicial, pois a busca faz a exploração nas posições que foram exploradas primeiro.

```
solucao_bl(Inicial,SolInv) :- bl([[Inicial]],Solucao), inversao(Solucao,SolInv).
bl([[Estado|Caminho]|_],[Estado|Caminho]) :- meta(Estado).
bl([[Primeiro|Outros], Solucao) :- estende(Primeiro,Sucessores),
concatena(Outros,Sucessores,NovaFronteira),bl(NovaFronteira,Solucao).
estende([Estado|Caminho],ListaSucessores):- bagof([Sucessor,Estado|Caminho],
(s(Estado,Sucessor),not(pertence(Sucessor,[Estado|Caminho]))), ListaSucessores),!.
estende(__,[]).
```

A função de busca em largura utilizada foi retirada dos slides do Profo Murilo Naldi.

As buscas e as verificações escritas no código dependem de algumas funções previamente definidas, por isso foram utilizadas as funções retirar\_elemento(), pertence() e concatena() retiradas dos slides do Prof<sup>o</sup> Murilo Naldi e a função inversao() foi retirada de um vídeo(https://www.youtube.com/watch?v=3THJwrC2Erw) do professor.

```
retirar_elemento(Elem, [Elem|Cauda], Cauda).
retirar_elemento(Elem, [Cabeca|Cauda], [Cabeca|Resultado]) :- retirar_elemento(Elem, Cauda, Resultado).

pertence(Elem, [Elem|__]).
pertence(Elem, [_| Cauda]) :- pertence(Elem, Cauda).

concatena([],L,L).
concatena([Cab|Cauda],L2,[Cab|Resultado]) :- concatena(Cauda,L2,Resultado).

inversao([],[]).
inversao([Cab|Cauda], Y):-inversao(Cauda,Inv), concatena(Inv,[Cab],Y).
```

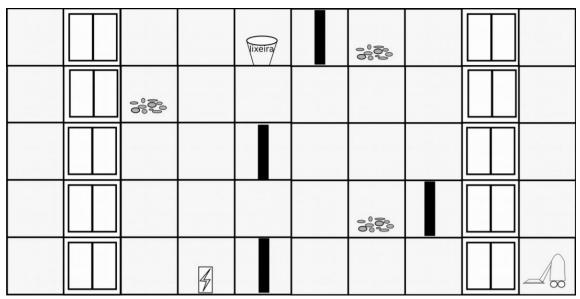
No código, há funções de write() e writeln() para permitir a visualização do estado em que o aspirador se encontra antes e depois das funções pegar lixo, jogar lixo e finalizar programa.

Foram feitos três exemplos de aplicação do programa criado, abaixo seguem os resultados de cada exemplo correspondente à imagem que aparece antes da sequência de estados. Para solucionar cada problema, o usuário deve inserir o comando contendo o estado inicial do aspirador no formato abaixo:

```
?- solucao bl([X, Y, S, F, L, P, E, G, D, no fim], X).
```

## Exemplos de aplicação:

#### Ambiente 1:

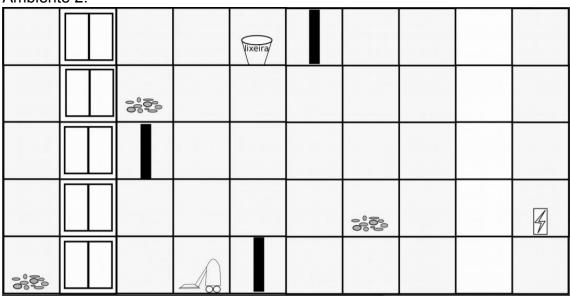


solucao\_bl([9,0,0,3,[[6,1],[6,4],[2,3]],[[5,4],[4,2],[7,1],[4,0]],[[1],[8]],[[4,4]], [[3,0]],[[10]],[[5]],no\_fim], X).

```
[[9, 0, 0, 3, [[6, 1], [6, 4], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[8, 0, 0, 3, [[6, 1], [6, 4], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[8, 1, 0, 3, [[6, 1], [6, 4], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[8, 2, 0, 3, [[6, 1], [6, 4], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[8, 3, 0, 3, [[6, 1], [6, 4], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[8, 4, 0, 3, [[6, 1], [6, 4], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[7, 4, 0, 3, [[6, 1], [6, 4], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[6, 4, 0, 3, [[6, 1], [6, 4], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[6, 4, 1, 2, [[6, 1], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[7, 4, 1, 2, [[6, 1], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[8, 4, 1, 2, [[6, 1], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[8, 3, 1, 2, [[6, 1], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[7, 3, 1, 2, [[6, 1], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[6, 3, 1, 2, [[6, 1], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[5, 3, 1, 2, [[6, 1], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[4, 3, 1, 2, [[6, 1], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[3, 3, 1, 2, [[6, 1], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[2, 3, 1, 2, [[6, 1], [2, 3]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[2, 3, 2, 1, [[6, 1]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[1, 3, 2, 1, [[6, 1]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[1, 4, 2, 1, [6, 1]], [5, 4], [4, 2], [7, 1], [4, 0]], [11], [8]], [4, 4]], [[3, 0]], [[10]], [[5]], no fim],
[2, 4, 2, 1, [[6, 1]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[3, 4, 2, 1, [[6, 1]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[4, 4, 2, 1, [[6, 1]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[4, 4, 0, 1, [[6, 1]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[3, 4, 0, 1, [[6, 1]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[2, 4, 0, 1, [[6, 1]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[1, 4, 0, 1, [[6, 1]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[1, 3, 0, 1, [[6, 1]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[1, 2, 0, 1, [[6, 1]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[1, 1, 0, 1, [[6, 1]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[2, 1, 0, 1, [[6, 1]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
```

```
[3, 1, 0, 1, [[6, 1]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[4, 1, 0, 1, [[6, 1]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[5, 1, 0, 1, [[6, 1]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[6, 1, 0, 1, [[6, 1]], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[6, 1, 1, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[5, 1, 1, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[4, 1, 1, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[3, 1, 1, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[2, 1, 1, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[1, 1, 1, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[1, 2, 1, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[1, 3, 1, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[1, 4, 1, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[2, 4, 1, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[3, 4, 1, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[4, 4, 1, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[4, 4, 0, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[3, 4, 0, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[2, 4, 0, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], \\ no\_fim], [[1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4], [1, 4]
[1, 4, 0, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[1, 3, 0, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no fim],
[1, 2, 0, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[1, 1, 0, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[1, 0, 0, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[2, 0, 0, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[3, 0, 0, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], no_fim],
[3, 0, 0, 0, [], [[5, 4], [4, 2], [7, 1], [4, 0]], [[1], [8]], [[4, 4]], [[3, 0]], [[10]], [[5]], fim]]
```

### Ambiente 2:

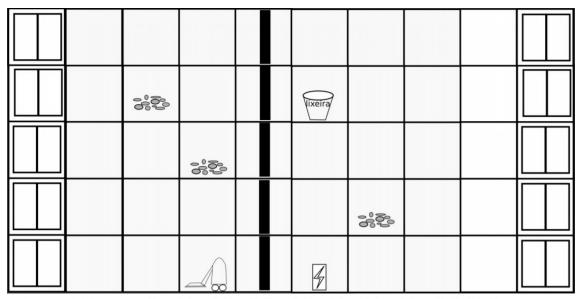


solucao\_bl([3,0,0,3,[[0,0],[2,3],[6,1]],[[5,4],[2,2],[4,0]],[[1]],[[4,4]],[[9,1]], [[10]],[[5]],no\_fim], X).

```
 [3, 0, 0, 3, [[0, 0], [2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim], [2, 0, 0, 3, [[0, 0], [2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[11]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim], [1, 0, 0, 3, [[0, 0], [2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim], [0, 0, 0, 3, [[0, 0], [2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim], [0, 0, 1, 2, [[2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim], [1, 0, 1, 2, [[2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim], [1, 0, 1, 2, [[2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim], [1, 0, 1, 2, [[2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim], [1, 0, 1, 2, [[2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim], [1, 0, 1, 2, [[2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim], [1, 0, 1, 2, [[2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim], [1, 0, 1, 2, [[2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim], [1, 0, 1, 2, [[2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim], [1, 0, 1, 2, [[2, 3], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim], [1, 0, 1, 2, [[2, 3], [4, 0]], [[2, 3], [4, 0]], [[2, 4]], [2, 2], [4, 0]], [[2, 4]], [2, 2], [4, 0]], [2, 2], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]], [4, 2]],
```

```
[1, 1, 1, 2, [[2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[2, 1, 1, 2, [[2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[3, 1, 1, 2, [[2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[4,\,1,\,1,\,2,\,[[2,\,3],\,[6,\,1]],\,[[5,\,4],\,[2,\,2],\,[4,\,0]],\,[[1]],\,[[4,\,4]],\,[[9,\,1]],\,[[10]],\,[[5]],\,no\_fim],
[5, 1, 1, 2, [[2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[6, 1, 1, 2, [[2, 3], [6, 1]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[6, 1, 2, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[5, 1, 2, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[4, 1, 2, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[3, 1, 2, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[2, 1, 2, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[1, 1, 2, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[1, 2, 2, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[1, 3, 2, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[1, 4, 2, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[2, 4, 2, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[3, 4, 2, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[4, 4, 2, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[4, 4, 0, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[3, 4, 0, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[2, 4, 0, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no fim],
[1, 4, 0, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[1, 3, 0, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[2, 3, 0, 1, [[2, 3]], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[2, 3, 1, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[1, 3, 1, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[1, 4, 1, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[2, 4, 1, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[3, 4, 1, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[4, 4, 1, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[4,\,4,\,0,\,0,\,[],\,[[5,\,4],\,[2,\,2],\,[4,\,0]],\,[[1]],\,[[4,\,4]],\,[[9,\,1]],\,[[10]],\,[[5]],\,no\_fim],
[3, 4, 0, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[2, 4, 0, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[1, 4, 0, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[1, 3, 0, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[1, 2, 0, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no fim],
[1, 1, 0, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[2, 1, 0, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[3, 1, 0, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[4, 1, 0, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[5, 1, 0, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[6, 1, 0, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[7, 1, 0, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[8, 1, 0, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[9, 1, 0, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], no_fim],
[9, 1, 0, 0, [], [[5, 4], [2, 2], [4, 0]], [[1]], [[4, 4]], [[9, 1]], [[10]], [[5]], fim]
```

#### Ambiente 3:



 $solucao\_bl([3,0,0,3,[[3,2],[2,3],[6,1]],[[4,4],[4,3],[4,2],[4,1],[4,0]],[[0],[9]],\\ [[5,3]],[[5,0]],[[10]],[[5]],no\_fim], X).$ 

```
?- solucao_bl([3,0,0,3,[[3,2],[2,3],[6,1]],[[4,4],[4,3],[4,2],[4,1],[4,0]],[[0],[9]], [[5,3]],[[5,0]],[[10]],[[5]],no_fim],X).
=> Pegou o lixo
Antes: [2,3,0,3,[[3,2],[2,3],[6,1]]]
=> Pegou o lixo
Antes: [2,3,1,2,[[3,2],[6,1]]]
=> Pegou o lixo
Antes: [2,3,1,2,[[2,3],[6,1]]]
=> Pegou o lixo
Antes: [2,3,1,2,[[2,3],[6,1]]]
=> Pegou o lixo
Antes: [2,3,1,2,[[3,2],[6,1]]]
=> Pegou o lixo
Antes: [3,2,1,2,[[6,1]]]
```

Obs: todos os ambientes e imagens apresentados foram retirados das notas de aula do Prof<sup>o</sup> Murilo Naldi.

Com base nos conhecimentos de inteligência artificial do grupo e nos resultados obtidos com os exemplos, foi possível programar um aspirador de pó que é capaz de coletar todas as sujeiras do ambiente, jogá-las no lixo e ir até o dock station seguindo o caminho mais eficiente encontrado usando busca em largura.