

POPEYE

Prolog – Inteligência Artificial



Eduardo Nunes – 6021

Victor Paiva – 5973

Thiago Antônio – 5991

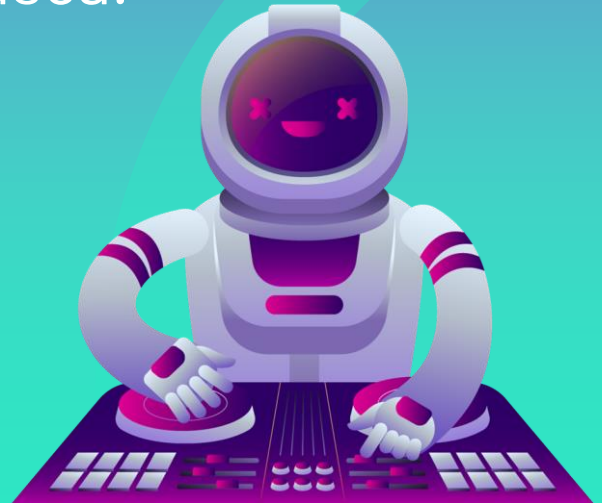
OBJETIVO

O objetivo do trabalho é fazer com que, a partir de um ambiente programável para o jogo, o Popeye consiga derrotar seu inimigo Brutus, passando por desafios e obstáculos durante o caminho.



OBJETIVO

Para isso, utilizando a linguagem Prolog, desenvolvemos um conjunto de regras consistentes para solucionar esse problema de busca.





AMBIENTE

O ambiente definido é uma matriz de 5 andares e 10 espaços para andar, totalizando 50 posições.

[illegible]



AGENTES

Os agentes podem ser definidos como os personagens do jogo.

AGENTE

POPEYE



- Popeye é o protagonista do jogo e sua posição é definida como a posição inicial do problema de busca.
- Pode ser setado em qualquer posição do ambiente.
- Para uma melhor experiência, recomendamos que ele inicie no primeiro andar da matriz de ambiente.

AGENTE

BRUTUS



- Brutus é o vilão do jogo, e sua posição é definida como a posição final do problema de busca.
- Pode ser setado em qualquer posição do ambiente.
- Para uma melhor experiência, recomendamos que ele inicie no último (5º) andar da matriz de ambiente.



OBJETOS

Os objetos são espalhados no ambiente para gerar obstáculos ou desafios durante o jogo.

OBJETO

CORAÇÃO



- Pode existir mais de um coração no ambiente.
- Pode ser inserido em qualquer posição vazia.
- Quando coletado, a pontuação é incrementada com o valor do respectivo andar que se encontra * 100.
- O coração é caracterizado como a primeira meta no caminho do Popeye em busca do Brutus.

OBJETO

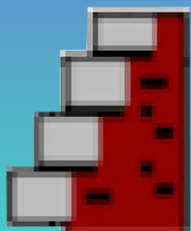
ESPINAFRE



- Pode existir apenas um espinafre no ambiente.
- Pode ser inserido em qualquer posição vazia.
- Só pode ser coletado após a completa coleta dos corações.
- O espinafre é caracterizado como a chave para a conclusão da busca de Brutus.

OBJETO

ESCADA



- Permite a movimentação do Popeye entre andares diferentes.
- Este objeto ocupa duas posições no ambiente (andar de baixo e andar de cima).
- É montada de forma diagonal.

OBJETO

GARRAFA



- A garrafa implica em um bloqueio na movimentação do Popeye.
- O ambiente pode conter mais de uma garrafa, em qualquer posição.
- O Popeye deve saltar a posição da garrafa.
- Caso exista duas garrafas em posições adjacentes, ou uma garrafa e o Brutus na posição seguinte, o Popeye não consegue saltá-la.

PROGRAMAÇÃO DO AMBIENTE

Você será o responsável por programar o ambiente do jogo, setando as coordenadas para cada agente e objeto explicados anteriormente, mas não se preocupe, iremos te instruir e mostrar como isso deve ser feito !



PROGRAMAÇÃO DO AMBIENTE

Os agentes e os objetos devem ser setados dessa maneira (como na imagem) no início do código principal, no caso, você precisará substituir o já existente, que está destacado em comentários. Observe que a maneira de inserir as escadas e os corações é diferente dos demais.

```
% DEFINIÇÃO DO AMBIENTE
```

```
garrafa([1,5]).  
garrafa([1,6]).  
garrafa([3,2]).  
garrafa([4,9]).  
garrafa([5,2]).  
garrafa([5,6]).
```

```
escada([[1,3],[2,4]]).  
escada([[2,2],[3,3]]).  
escada([[3,7],[4,6]]).  
escada([[4,3],[5,4]]).
```

```
coracao([[1,2],[1,3],[1,4],[2,6]]).
```

```
espinafre([3,8]).
```

```
brutus([5,10]).
```



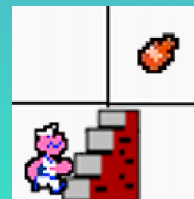
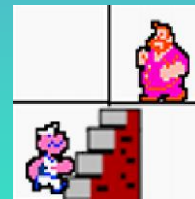
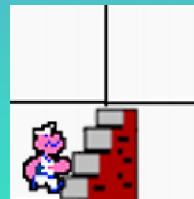
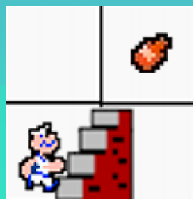
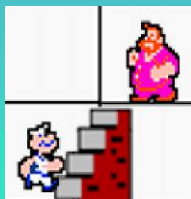
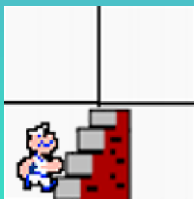
IMPLEMENTAÇÃO

- Regras de movimentação.
- Regras para manipular listas.
- Regras de busca em largura.

MOVIMENTAÇÃO – SUBIR

- Espinafre não coletado

```
/*movimentacao pra subir pela escada*/  
movimentacaoSemEspinafre([X,Y],[X2,Y2],Brutus):-  
    escada([[X,Y],[X2,Y2]]),  
    not(garrafa([X2,Y2])),  
    [X2,Y2] \= Brutus.
```



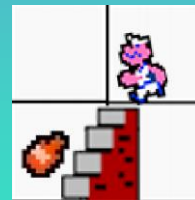
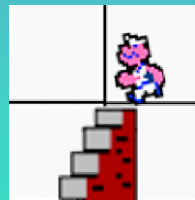
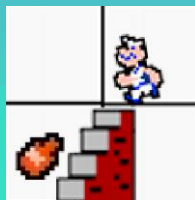
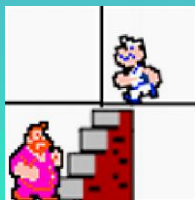
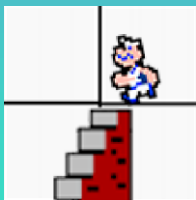
MOVIMENTAÇÃO - DESCER

- Espinafre não coletado

```
/*movimentacao pra descer pela escada*/  
movimentacaoSemEspinafre([X,Y],[X2,Y2],Brutus):-  
    escada([[X2,Y2],[X,Y]]),  
    not(garrafa([X2,Y2])),  
    [X2,Y2] \= Brutus.
```

- Espinafre já coletado

```
/*movimentacao pra descer pela escada*/  
movimentacaoComEspinafre([X,Y],[X2,Y2]):-  
    escada([[X2,Y2],[X,Y]]),  
    not(garrafa([X,Y2])).
```



MOVIMENTAÇÃO – DIREITA

- Espinafre não coletado

```
/*movimentacao para direita sem obstaculo*/  
movimentacaoSemEspinafre([X,Y],[X,Y2],Brutus):-  
    Y<10,  
    Y2 is Y+1,  
    not(garrafa([X,Y2])),  
    [X,Y2] \= Brutus.
```

```
/*movimentacao para direita com obstaculo*/  
movimentacaoSemEspinafre([X,Y],[X,Y2],Brutus):-  
    Y<9,  
    Y1 is Y+1,  
    Y2 is Y+2,  
    garrafa([X,Y1]),  
    not(garrafa([X,Y2])),  
    [X,Y1] \= Brutus,  
    [X,Y2] \= Brutus.
```

- Espinafre já coletado

```
/*movimentacao para direita sem obstaculo*/  
movimentacaoComEspinafre([X,Y],[X,Y2]):-  
    Y<10,  
    Y2 is Y+1,  
    not(garrafa([X,Y2])).
```

```
/*movimentacao para direita com obstaculo*/  
movimentacaoComEspinafre([X,Y],[X,Y2]):-  
    Y<9,  
    Y1 is Y+1,  
    Y2 is Y+2,  
    garrafa([X,Y1]),  
    not(garrafa([X,Y2])).
```

MOVIMENTAÇÃO – ESQUERDA

- Espinafre não coletado

```
/*movimentacao para esquerda sem obstaculo*/  
movimentacaoSemEspinafre([X,Y],[X,Y2],Brutus):-  
    Y>1,  
    Y2 is Y-1,  
    not(garrafa([X,Y2])),  
    [X,Y2] \= Brutus.  
  
/*movimentacao para esquerda com obstaculo*/  
movimentacaoSemEspinafre([X,Y],[X,Y2],Brutus):-  
    Y>1,  
    Y1 is Y-1,  
    Y2 is Y-2,  
    garrafa([X,Y1]),  
    not(garrafa([X,Y2])),  
    [X,Y1] \= Brutus,  
    [X,Y2] \= Brutus.
```

- Espinafre já coletado

```
/*movimentacao para esquerda sem obstáculo*/  
movimentacaoComEspinafre([X,Y],[X,Y2]):-  
    Y>1,  
    Y2 is Y-1,  
    not(garrafa([X,Y2])).  
  
/*movimentacao para esquerda com obstáculo*/  
movimentacaoComEspinafre([X,Y],[X,Y2]):-  
    Y>2,  
    Y1 is Y-1,  
    Y2 is Y-2,  
    garrafa([X,Y1]),  
    not(garrafa([X,Y2])).
```

MANIPULAR LISTAS — VERIFICAR SE EXISTE ELEMENTO

```
/*Verificar se o elemento pertence a lista*/  
pertence(Elem,[Elem|_]).  
pertence(Elem,[_|Cauda]):-  
    pertence(Elem,Cauda).
```

MANIPULAR LISTAS — CONCATENAR

```
/*Concatenar lista*/  
concatena([], L, L).  
concatena([Cab|L1], L2,[Cab|L3]):-  
    concatena(L1, L2, L3).
```

MANIPULAR LISTAS — INVERTER

```
/*Inverter lista*/  
inverter([],[]).  
inverter([Elem|Cauda], Lista_Invertida):-  
    inverter(Cauda,Cauda_Invertida),  
    concatena(Cauda_Invertida,[Elem], Lista_Invertida).
```

MANIPULAR LISTAS — RETIRAR ELEMENTO

```
/*Retirar um elemento da lista*/  
retirar_elemento(Elem,[Elem|Cauda],Cauda).  
retirar_elemento(Elem,[Elem1|Cauda],[Elem1|Cauda1]):-  
    retirar_elemento(Elem,Cauda,Cauda1).
```


MANIPULAR LISTAS — EXIBIR ÚLTIMO ELEMENTO

```
/*exibir ultimo elemento da lista*/  
retornar_ultimo(X,[X]).  
retornar_ultimo(X, [_|T]):-  
    retornar_ultimo(X,T).
```

MANIPULAR LISTAS — EXIBIR PRIMEIRO ELEMENTO

```
/*retorna o primeiro elemento da lista*/  
retornar_primeiro(X,[X|_]).
```

BUSCA EM LARGURA

```
/* -----Início de regras de busca para quando o espinafre ainda não foi encontrado -----*/

busca_em_largura_sem_espinafre([[Estado|Caminho]|_],Destino,[Estado|Caminho],Brutus):-
    Destino=Estado,Brutus=Brutus.

busca_em_largura_sem_espinafre([Primeiro|Outros],Destino,Solucao,Brutus):-
    estende_sem_espinafre(Primeiro,Sucessores,Brutus),
    concatena(Outros,Sucessores,NovaFronteira),
    busca_em_largura_sem_espinafre(NovaFronteira,Destino,Solucao,Brutus).

estende_sem_espinafre([Estado|Caminho],ListaSucessores,Brutus):-
    bagof([Sucessor,Estado|Caminho],(movimentacaoSemEspinafre(Estado,Sucessor,Brutus),not(pertence(Sucessor,[Estado|Caminho]))),ListaSucessores),
    !.

estende_sem_espinafre(_,[],_).
```

BUSCA EM LARGURA

```
/*----- Início de busca para quando o espinafre já foi encontrado ----- */

busca_em_largura_com_espinafre([[Estado|Caminho]|_],Destino,[Estado|Caminho]):-
    Destino=Estado.

busca_em_largura_com_espinafre([Primeiro|Outros],Destino,Solucao):-
    estende_com_espinafre(Primeiro,Sucessores),
    concatena(Outros,Sucessores,NovaFronteira),
    busca_em_largura_com_espinafre(NovaFronteira,Destino,Solucao).

estende_com_espinafre([Estado|Caminho],ListaSucessores):-
    bagof([Sucessor,Estado|Caminho],(movimentacaoComEspinafre(Estado,Sucessor),not(pertence(Sucessor,[Estado|Caminho]))),ListaSucessores),
    !.

estende_com_espinafre(_,[]).
```

SOLUÇÃO DOS CORAÇÕES

```
/*Regra recursiva para somar os caminhos de um coracao até o outro*/
recursivo(Elem,[Cauda|Cauda1],Solucao,Brutus,SomaSolucao,Cauda2>TotalPontos,SomaPontos):-
    busca_em_largura_sem_espinafre([[Elem]],Cauda,Solucao0,Brutus),
    concatena(Solucao0,SomaSolucao,Solucao2),
    (
        /*a primeira vez cai aqui*/
        Cauda2 = [] ,
        /*se so tiver 2 coracoes*/
        (Cauda1 = [],
         concatena(Solucao2,[],Solucao),
         retornar_primeiro(PontosTemp1,Elem),
         PontosTemp is PontosTemp1 *100,
         retornar_primeiro(PontosTemp2,Cauda),
         TotalPontos is PontosTemp2 *100 + PontosTemp
        );
        /*se tiver mais de 2 coracoes*/
        retornar_primeiro(PontosTemp1,Elem),
        PontosTemp is PontosTemp1 *100,
        recursivo(Cauda,Cauda1,Solucao,Brutus,Solucao2,Cauda>TotalPontos,PontosTemp))
    );
/*segunda vez em diante cai aqui*/
retirar_elemento(Cauda2,Solucao2,Solucao3),
/*se so restar 2 coracoes*/
(Cauda1 = [],
 concatena(Solucao3,[],Solucao),
 retornar_primeiro(PontosTemp1,Elem),
 PontosTemp is PontosTemp1 *100 + SomaPontos,
 retornar_primeiro(PontosTemp3,Cauda),
 TotalPontos is PontosTemp3 *100 + PontosTemp
);
/*se restar mais de 2 coracoes*/
retornar_primeiro(PontosTemp1,Elem),
PontosTemp is PontosTemp1 *100 + SomaPontos,
PontosTemp2 is PontosTemp,
recursivo(Cauda,Cauda1,Solucao,Brutus,Solucao3,Cauda>TotalPontos,PontosTemp2))
).
```

SOLUÇÃO

```
solucao_b1(Popeye,[Coracao|Cauda],Espinafre,Brutus,Solucao,Pontos):-  
    busca_em_largura_sem_espinafre([[Popeye]],Coracao,Solucao1,Brutus),  
    (  
        /*só tem 1 coração*/  
        Cauda = [] ,  
        retornar_primeiro(PontosTemp,Coracao),  
        Pontos is PontosTemp *100,  
        busca_em_largura_sem_espinafre([[Coracao]],Espinafre,Solucao5,Brutus),  
        concatena(Solucao5,Solucao1,Solucao6),  
        retirar_elemento(Coracao,Solucao6,Solucao7)  
    );  
    /*tem 2 ou mais corações*/  
    recursivo(Coracao,Cauda,Solucao2,Brutus,[],[],Pontos,Pontos),  
    concatena(Solucao2,Solucao1,Solucao3),  
    retirar_elemento(Coracao,Solucao3,Solucao4),  
    retornar_ultimo(X,Cauda),  
    busca_em_largura_sem_espinafre([[X]],Espinafre,Solucao6,Brutus),  
    retirar_elemento(X,Solucao4,Solucao5),  
    concatena(Solucao6,Solucao5,Solucao7)  
),  
    busca_em_largura_com_espinafre([[Espinafre]],Brutus,Solucao8),  
    concatena(Solucao8,Solucao7,Solucao9),  
    retirar_elemento(Espinafre,Solucao9,Solucao10),  
    inverter(Solucao10,Solucao).
```

EXEMPLO COM UM AMBIENTE COMPLETO



MATRIZ – AMBIENTE I COMPLETO



```
garrafa([1,5]).  
garrafa([1,6]).  
garrafa([3,2]).  
garrafa([4,9]).  
garrafa([5,2]).  
garrafa([5,6]).
```

```
escada([[1,3],[2,4]]).  
escada([[2,2],[3,3]]).  
escada([[3,7],[4,6]]).  
escada([[4,3],[5,4]]).
```

```
coracao([[2,10],[3,5],[4,1],[5,3]]).
```

```
espinafre([3,8]).
```

```
brutus([5,10]).
```

	1	2	3	4	5	6	7	8	9	10
5										
4										
3										
2										
1										

SAÍDA DO ARQUIVO

Caminho total percorrido por Popeye :

```
[[1,1],[1,2],[1,3],[2,4],[2,5],[2,6],[2,7],[2,8],[2,9],[2,10],[2,9],[2,8],[2,7],[2,6],[2,5],[2,4],[2,3],[2,2],[3,3],[3,4],[3,5],[3,6],[3,7],[4,6],[4,5],  
[4,4],[4,3],[4,2],[4,1],[4,2],[4,3],[5,4],[5,3],[5,4],[4,3],[4,4],[4,5],[4,6],[3,7],[3,8],[3,7],[4,6],[4,5],[4,4],[4,3],[5,4],[5,5],[5,7],[5,8],[5,9],[5,10]]
```

Localizacao do Coracao : [[2,10],[3,5],[4,1],[5,3]]

Localizacao do Espinafre : [3,8]

Localizacao do Brutus : [5,10]

Pontuacao : 1400

OBRIGADO !

Agradecimentos ao Prof. Matheus Haddad



eduardo.n.oliveira@ufv.br

victor.p.castro@ufv.br

thiago.a.ferreira@uvf.br