

Uma imagem com alimentação

Descrição gerada automaticamente

**Relatório de ALGAV**

**3DD \_ Grupo 100**

1191256 André Reis

1191362 Rúben Amorim

1170878 Milene Farias

**Data: 26/11/2023**

# 1 - Representação do conhecimento do domínio

# Predicados

# Base de conhecimento

|  |  |
| --- | --- |
| liga(a,b). | Serve para identificar que existe uma ligação entre 2 edifícios. Os parâmetros são os edifícios. |
| pisos(a,[a1,a2]). | Serve para identificar os pisos de um edifício. O primeiro parâmetro é o edifício específico e o segundo parâmetro uma lista com os pisos. |
| dimensoes(a1, 23, 11). | Serve para indicar as dimensões de cada piso. O primeiro parâmetro é o piso, os outros 2 parâmetros são as dimensões do andar. |
| elevador(a,[a1,a2]). | Serve para identificar os andares que o elevador pode percorrer. O primeiro parâmetro é o elevador em específico e o segundo parâmetro uma lista com os pisos. |
| passagem(a,b,a2,b2). | Serve para indicar a passagem entre 2 edifícios. Os 2 primeiros parâmetros servem para indicar os edifícios da ligação. Os 2 últimos parâmetros são as os pisos que irão ser ligados. |
| sala(apn, a1). | Serve para indicar uma sala. O primeiro parâmetro é o nome da sala e o 2 parâmetro o piso da mesma. |
| coordenadas(apn, a1, 10, 6). | Serve para indicar as coordenadas das portas de cada sala. O primeiro parâmetro indica o nome da sala, o segundo parâmetro o andar da sala e os 2 últimos parâmetros a localização da porta da sala no piso. |
| m(a1,0,0,3). | Serve para indicar o mapa do andar com todas as suas posições. O primeiro parâmetro é o andar, o segundo parâmetro a linha em que se encontra no array de posições, o terceiro parâmetro é a coluna e o quarto parâmetro o valor para dizer se é uma parede, espaço vazio, etc (0 ,1 ,2 ,3) |

# Alterações efetuadas nos algoritmos fornecidos

As alterações efetuadas nos algoritmos fornecidos foram as seguintes:

* Todos os algoritmos foram alterados para incluírem o piso;
* Na criação do graph foi alterado o código para incluir o peso das ligações das células;
* Durante a criação dos graphs, quando são efetuados movimentos na vertical e horizontal o peso é de 1 e nas diagonais é de

# 2 - Obtenção da solução ótima para o Planeamento de movimentação entre pisos de edifícios movendo-se por corredores de ligação entre edifícios e elevadores (deve indicar pontos de partida e de chegada que devem ser pontos de acesso a salas/gabinetes/elevadores/corredores externos).

# 3 - Movimentação do robot dentro de um piso de edifício com Primeiro em Profundidade;

dfs(Piso,Orig,Dest,Cam):-

dfs2(Piso,Orig,Dest,[Orig],Cam).

dfs2(\_,Dest,Dest,LA,Cam):-

reverse(LA,Cam).

dfs2(Piso,Act,Dest,LA,Cam):-

(ligacel(Piso,Act,X,\_); ligacel(Piso,X,Act,\_)),

\+ member(X,LA),

dfs2(Piso,X,Dest,[X|LA],Cam).

# 4 - Geração de todas as soluções do Primeiro em Profundidade e escolha da melhor;

better\_dfs(Piso,Orig,Dest,Cam):-

all\_dfs(Piso,Orig,Dest,LCam), shortlist(LCam,Cam,\_).

shortlist([L],L,N):-!,length(L,N).

shortlist([L|LL],Lm,Nm):-

shortlist(LL,Lm1,Nm1), length(L,NL),

((NL<Nm1,!,Lm=L,Nm is NL); (Lm=Lm1,Nm is Nm1)).

# 5 - Primeiro em Largura

bfs(Piso,Orig,Dest,Cam):-bfs2(Piso,Dest,[[Orig]],Cam).

bfs2(\_,Dest,[[Dest|T]|\_],Cam):-

reverse([Dest|T],Cam).

bfs2(Piso,Dest,[LA|Outros],Cam):-

LA=[Act|\_],

findall([X|LA],

(Dest\==Act,(ligacel(Piso,Act,X,\_);ligacel(Piso,X,Act,\_)),

\+ member(X,LA)),

Novos),

append(Outros,Novos,Todos),

bfs2(Dest,Todos,Cam).

# 6 - A\*

% calcular a distância euclidiana entre duas células

estimativa(cel(X1, Y1), cel(X2, Y2), Distancia) :-

Distancia is sqrt((X1 - X2)^2 + (Y1 - Y2)^2).

% predicado principal do A\*

aStar(Orig, Dest, Cam, Custo, Piso) :-

aStar2(Piso, Dest, [(\_, 0, [Orig])], Cam, Custo).

% predicado auxiliar para o A\*

aStar2(\_, Dest, [(\_, Custo, [Dest|T])|\_], Cam, Custo) :-

reverse([Dest|T], Cam).

aStar2(Piso, Dest, [(\_, Ca, LA)|Outros], Cam, Custo) :-

LA = [Act|\_],

findall((CEX, CaX, [X|LA]),

(Dest \== Act,

(ligacel(Piso, Act, X,CustoX);ligacel(Piso, X,

Act,CustoX)), \+ member(X, LA),

CaX is CustoX + Ca, estimativa(X, Dest, EstX),

CEX is CaX + EstX),

Novos),

append(Outros, Novos, Todos),

% write('Novos='),write(Novos),nl,

sort(Todos, TodosOrd),

% write('TodosOrd='),write(TodosOrd),nl,

aStar2(Piso, Dest, TodosOrd, Cam, Custo).

# 7 - Consideração de movimentos nas diagonais

cria\_grafo\_lin(Piso,Col,Lin):-

m(Piso,Lin,Col,0),!,

ColS is Col+1, ColA is Col-1,

LinS is Lin+1,LinA is Lin-1,

((m(Piso,Lin,ColS,0),assertz(ligacel(Piso,cel(Col,Lin),cel(ColS,Lin),**1**));true)),

((m(Piso,Lin,ColA,0),assertz(ligacel(Piso,cel(Col,Lin),cel(ColA,Lin),**1**));true)),

((m(Piso,LinS,Col,0),assertz(ligacel(Piso,cel(Col,Lin),cel(Col,LinS),**1**));true)),

((m(Piso,LinA,Col,0),assertz(ligacel(Piso,cel(Col,Lin),cel(Col,LinA),**1**));true)),

((m(Piso,LinS,ColS,0), **m(Piso,LinS,Col,0)**, **m(Piso,Lin,ColS,0)**, assertz(ligacel(Piso,cel(Col,Lin),cel(ColS,LinS),**sqrt(2)**));true)),

((m(Piso,LinA,ColA,0), **m(Piso,LinA,Col,0)**, **m(Piso,Lin,ColA,0)**, assertz(ligacel(Piso,cel(Col,Lin),cel(ColA,LinA),**sqrt(2)**));true)),

((m(Piso,LinA,ColS,0), **m(Piso,LinA,Col,0)**, **m(Piso,Lin,ColS,0)**, assertz(ligacel(Piso,cel(Col,Lin),cel(ColS,LinA),**sqrt(2)**));true)),

((m(Piso,LinS,ColA,0), **m(Piso,Lin,ColA,0)**, **m(Piso,LinS,Col,0)**, assertz(ligacel(Piso,cel(Col,Lin),cel(ColA,LinS),**sqrt(2)**));true)),

Col1 is Col-1,

cria\_grafo\_lin(Piso,Col1,Lin).

# 8 - Integração do ponto 2 com o ponto 3

# Conclusões