Demonstração do algoritmo BEST FIRST

---Algoritmo completo

Esta é a primeira parte do algoritmo, semelhante ao que nos foi fornecido nas aulas, aqui a única diferença é que o adaptamos para retornar os paths pelos quais passou durante o seu caminho.

Depois de correr o bestfs2 vai correr o calx que vai calcular o momento em que vai chegar ao destino.

```
bestfs(Orig,Dest,Cam,Lreverse,PassingTime,MomentoChegada):-

bestfs2(Dest,(0,[Orig]),Cam,[],Lreverse),!,
calx(Cam,Lreverse,PassingTime,MomentoChegada).


bestfs2(Dest,(_,[Dest|T]),Cam,Listapathspercorridos,Lreverse):-

reverse([Dest|T],Cam),

reverse(Listapathspercorridos, Lreverse),

!.



bestfs2(Dest,(Ca,LA),Cam,Listapathspercorridos,Lreverse):-

LA=[Act|_],

findall((EstX,CaX,[X|LA],[Hp|Listapathspercorridos]),
(

liga(Act,X,Hp),

estimativabf(Act,X,Cx),

\+member(X,LA),
estimativabf(X,Dest,EstX),



CaX is Ca+Cx)
 ,Novos),


sort(Novos,NovosOrd),
%NovosOrd = [(_,Cm,Melhor,Listpath)|_], comentado porque fiz a alteraÃ§Ã£o sugerida pelo regente
proximo(NovosOrd,Cm,Melhor,Listpath),

bestfs2(Dest,(Cm,Melhor),Cam,Listpath,Lreverse).



estimativabf(Nodo1,Nodo2,Estimativa):-

nodes(_,Nodo1,_,_,X1,Y1),
nodes(_,Nodo2,_,_,X2,Y2),
Estimativa is sqrt((X1-X2)^2+(Y1-Y2)^2).


proximo([(_,Cm,Melhor,Listpath)|_],Cm,Melhor,Listpath).
proximo([_|L],Cm,Melhor,Listpath):- proximo(L,Cm,Melhor,Listpath).
```

Algoritmo “calx” calcula o Momento de chegada ao destino.

```
calx([H1|[H2|T2]],[H3|T3],PassingTime,MomentoChegada):-
calx2([H1|[H2|T2]],[H3|T3],PassingTime,MomentoChegada),!.


calx2([_],_,Lastpassingtime,Lastpassingtime):- !.


calx2([H1|[H2|T2]],[H3|T3],Lastpassingtime,MomentoChegada):-

linhas(_,H3,Perc,_,_),


nth0(PosAct,Perc,H1),


horario(H3,ListaHorario),

nth0(PosAct,ListaHorario,Tempoespera),

Lastpassingtime=<Tempoespera,




nth0(Posd,Perc,H2),

Posd>PosAct,

nth0(Posd,ListaHorario,LastpassingTime),



calx2([H2|T2],T3,LastpassingTime,MomentoChegada).
```

Para testar o algoritmo usei a seguinte base de conhecimento:

```
nodes('Aguiar de Sousa', 'AGUIA', f, f, -8.4464785432391, 41.1293363229325).
nodes('Baltar', 'BALTR', f, f, -8.38716802227697, 41.1937898023744).
nodes('Cete', 'CETE', f, f, -8.35164059584564, 41.183243425797).
nodes('Cristelo', 'CRIST', t, f, -8.34639896125324, 41.2207801252676).
nodes('Duas Igrejas', 'DIGRJ', f, f, -8.35481024956726, 41.2278665802794).
nodes('Lordelo', 'LORDL', t, f, -8.42293614720057, 41.2452627470645).
nodes('Parada de Todeia', 'PARAD', f, f, -8.37023578802149, 41.1765780321068).
nodes('Paredes', 'PARED', t, f, -8.33566951069481, 41.2062947118362).
nodes('Recarei', 'RECAR', t, f, -8.42215867462191, 41.1599363478137).
nodes('Vila Cova de Carros', 'VCCAR', f, f, -8.35109395257277, 41.2090666564063).

liga('AGUIA','PARED',1).
liga('AGUIA','RECAR',1).
liga('RECAR','PARED',1).

liga('LORDL','PARAD',2).

liga('PARED','LORDL',3).
liga('PARED','RECAR',3).
liga('PARED','CETE',3).

liga('RECAR','LORDL',3).

linhas('Paredes_Aguiar', 1, ['AGUIA','RECAR','PARED'], 31, 15700).
linhas('Lordelo_Parada',2,['LORDL','DIGRJ','CRIST','VCCAR','BALTR','PARAD'],22,11000).
linhas('Paredes_Aguiar', 3, ['PARED', 'CETE','RECAR', 'LORDL'], 31, 15700).

horario(1,[200,500,1000]).
horario(1,[2000,5000,10000]).
horario(2,[2200,2300,2400,2500,2600,2700]).
horario(2,[4200,4300,4400,4500,4600,4700]).
horario(3,[1300,1500,1700,2000]).
horario(3,[1300,1500,5100,6100]).
```

Demonstração do algoritmo:

Segundo a base de conhecimento de testes incluída no ficheiro testes.pl (onde está também o algoritmo):

-1- dando o nó inicial como ‘AGUIA’ e nó final como ‘RECAR’, este será o exemplo mais simples visto que ambos os nós fazem parte do mesmo percurso (tem ligação direta) e dando um tempo inicial de 1000 (1000 segundos depois do meio dia) obtenho o resultado:

PS: Lreverse é a lista de percursos pelos quais passei, embora não fosse pedido, foi me útil para elaborar o algoritmo e é sempre mais uma informação que posso dar a quem for executar o best first.

```
?- bestfs('AGUIA','RECAR',Cam,Custo,Lreverse,1000,M).

Cam = ['AGUIA', 'RECAR'],
Lreverse = [1],
M = 5000
```

O que está coreto visto que pelos horários, a próxima viagem a partir de 1000 segundos apos o meio dia, começa 1000 segundos depois dessa hora e a chegada ao destino acontece aos 5000 segundos depois do meio dia.

-2-Dando o tempo inicial como 500 segundos depois do meio dia obtenho exatamente o mesmo resultado

```
?- bestfs('AGUIA','RECAR',Cam,Custo,Lreverse,500,M).

Cam = ['AGUIA', 'RECAR'],
Lreverse = [1],
M = 5000.
```

Pois a única viagem depois desse tempo inicial será a mesma viagem do que na alínea 1, logo o tempo de chegada ao destino será o mesmo.

Se puser um tempo inicial de 50 obtenho um resultado diferente:

```
?- bestfs('AGUIA','RECAR',Cam,Custo,Lreverse,50,M).

Cam = ['AGUIA', 'RECAR'],
Lreverse = [1],
M = 500.
```

Visto que existe uma viagem que começa aos 100 segundos depois do meio dia e, portanto, entro nessa viagem, que chega ao destino aos 500 segundos depois do meio dia ao invés da viagem que começa aos 1000 segundos.

Se puser um tempo inicial de 10000 não encontrei uma solução visto que a partir dessa hora não há mais viagens disponíveis

```
?- bestfs('AGUIA','RECAR',Cam,Custo,Lreverse,10000,M).
```

false.

-3-Outro exemplo mais complexo: Agora quero ir de Aguiar até Lordelo, agora os eles não estão na mesma linha, mas como existe um caminho possível com viagens (horários) possíveis então vou obter uma solução.
Dando um tempo inicial de 500:

PS: Lreverse neste caso tem 2 percursos, o que significa que entre os nós AGUIA e RECAR fui pelo percurso 1 e entre os nós RECAR e LORDL fui pelo percurso 3.

```
?- bestfs('AGUIA','LORDL',Cam,Custo,Lreverse,500,M).
```

```
Cam = ['AGUIA', 'RECAR', 'LORDL'],
Lreverse = [1, 3],
M = 16000.
```

O que está correto visto que AGUIA está ligado a RECAR que esta ligado a LORDL, os percursos que percorro são o 1 de AGUIA para RECAR e o percurso 3 de RECAR para LORDL e a chegada em LORD será aos 16000 segundos depois do meio dia.

De AGUIA para Recar vai entrar em AGUIA na viagem que começa aos 2000 segundos, vai chegar em RECAR aos 5000 segundos e de RECAR para LORDL a próxima viagem que esta disponível começara aos 5100 segundos, entrando aí e saindo em LORDL aos 16000 segundos.

-4- Se na mesma situação da alínea anterior der um tempo inicial de 100, obtenho um resultado diferente:

```
?- bestfs('AGUIA','LORDL',Cam,Custo,Lreverse,100,M).
```

```
Cam = ['AGUIA', 'RECAR', 'LORDL'],
Lreverse = [1, 3],
M = 2000.
```

Pois na primeira parte da viagem, entre AGUIA e RECAR apanho uma viagem que começa mais cedo (200 segundos), saio em RECAR aos 500 segundos e apanho a viagem do percurso que faz Recar ate LORDL aos 1700 segundos, saindo em LORDL aos 2000 segundos.

-5- Ainda mais um exemplo, dando o nó inicial AGUIA e final como CETE e um tempo inicial de 50, obtenho:

```
?- bestfs('AGUIA','CETE',Cam,Custo,Lreverse,100,M).
```

```
Cam = ['AGUIA', 'PARED', 'CETE'],
Lreverse = [1, 3],
M = 1500.
```

O que é o esperado visto que apanho a viagem que vai de AGUIA para PARED (percurso 1), que começa aos 200 segundos, chego em PARED aos 1000 segundos, depois apanho a viagem que vai de PARED para CETE que começa aos 1300 segundos e chega ao destino (CETE) aos 1500 segundos.