ALGAV-Algoritmia Avançada



2014/2015

Modelação da rede de metro de Paris

Trabalho realizado por:

António Campos nº 1111305

Miguel Fontes nº 1110738

Turma 3DE

**Índice**

Introdução------------------------------------------------------------------------------------3

Funcionalidades Implementadas----------------------------------------------------------4

Base de Conhecimento---------------------------------------------------------------4

Funções Auxiliares--------------------------------------------------------------------6

Métodos de Pesquisa------------------------------------------------------------------7

Conclusão----------------------------------------------------------------------------------------9

Bibliografia-------------------------------------------------------------------------------------10

**Introdução**

O presente relatório foi realizado no âmbito da unidade curricular de Algoritmia Avançada.

O mesmo visa a realização de um trabalho prático relacionado com a modelação da rede de metro de Paris.

O objectivo deste trabalho foi desenvolver uma aplicação que permitisse sugerir trajectos aos turistas em função dos seus interesses e nos locais que os mesmos pretendiam visitar.

Nos próximos capítulos serão apresentadas as metodologias usadas para a realização dos problemas propostos no enunciado, bem como as conclusões a que nós chegamos ao termos realizado este trabalho.

**Funcionalidades implementadas**

Para realizar o trabalho recorremos a um exercício realizado nas aulas prático-laboratoriais em que era pedido para gerir uma rede de metro. Reaproveitamos a base de conhecimento usada e alguns predicados que consideramos como sendo úteis para a implementação do trabalho.

**Base de conhecimento:**

Para elaborar a base de conhecimento do trabalho, foi-nos pedido para criar informação sobre as linhas, estações, cruzamentos, pontos turísticos, horário e frequências.

* **Linhas:**

Para criar as linhas do metro elaborámos factos com o número da linha e a lista de estações correspondentes como argumentos. Exemplos de linhas:

linha(1, ['Château de Vincennes', 'Bérault', 'Saint-Mandlé-Tourelle', 'Porte de Vincennes', 'Nation', 'Reuilly-Diderot', 'Gare de Lyon', 'Bastille', 'Saint-Paul', 'Hotel de Ville', 'Châtelet', 'Louvre-Rivoli', 'Palais Royal-Museé du Louvre', 'Tuileries', 'Concorde', 'Champs-Élysées-Clemenceau', 'Franklin D.Roosevelt', 'George V', 'Charles de Gaulle - Étoile', 'Argentine', 'Porte Maillot', 'Les Sablons', 'Pont de Neuilly', 'Esplanade de La Défense', 'La Défense']).

linha(2, ['Porte Dauphine', 'Victor Hugo', 'Charles de Gaulle Étoile', 'Ternes', 'Courcelles', 'Monceau', 'Villiers', 'Rome', 'Place de Clichy', 'Blanche', 'Pigalle', 'Anvers', 'Barbès-Rochechouart', 'La Chapelle', 'Stalingrad', 'Jaurès', 'Colonel Fabien', 'Belleville', 'Couronnes', 'Ménilmontant', 'Père Lachaise', 'Philippe Auguste', 'Alexandre Dumas', 'Avron', 'Nation']).

* **Pontos turísticos:**

Para os pontos turísticos, usamos como argumentos o nome do ponto turístico, a estação que passa pelo mesmo, o tempo de visita, em minutos, horário de abertura e horário de encerramento.

ponto\_turistico('Torre Eiffel','Champ de Mars Tour Eiffel',30,'9:30','23:30').

ponto\_turistico('Arco do Triunfo','Charles de Gaulle Êtoile',25,'8:00','23:00').

ponto\_turistico('Sacré Cour','Jules Joffrin',40,'6:00','10:30').

* **Horários:**

Em relação aos horários, passamos como argumentos o número da linha, a hora de abertura e hora de fecho.

horario(1,'6:05','1:15').

horario(2,'5:30','0:40').

horario(3,'5:30','0:45').

* **Frequências:**

Para criar as frequências, usamos como parâmetros o nome da estação e o tempo de espera entre o último metro que partiu e o próximo.

frequencia('Abbesses',4).

frequencia('Alexandre Dumas',3).

frequencia('Alma-Marceau',3).

frequencia('Alésia',2).

**Funções Auxiliares**

* **Gera\_estacoes**

gera\_estacoes:- findall(L, linha(\_,L), LE), gera\_estacoes(LE, Estacoes),

assertz(estacoes(Estacoes)).

gera\_estacoes([],[]).

gera\_estacoes([H|T], LE):- gera\_estacoes(T,LR), append(LR, H, LE).

Esta função permite a criação dinâmica de estações carregadas na base de conhecimento através dos factos linha e transforma-as em factos do tipo estacao(Estacoes), em que Estacoes é uma lista com todas as estações.

* **Gera\_cruzamentos**

gera\_cruzamentos:- findall(\_,cruzamento,\_).

cruzamento:- linha(N1,LE1), linha(N2,LE2), N1 \== N2, intersection(LE1,LE2,LI),

LI \==[], assertz(cruzamento(N1,N2,LI)).

Esta função cria factos do tipo cruzamento(N1,N2,LI), em que os dois primeiros parâmteros são os números das linhas que se cruzam e o terceiro é uma lista com as estações pertencentes ao cruzamento.

* **Gera\_ligacoes**

gera\_ligacoes:- findall(\_, (linha(N,L), gera\_ligacoes(N, L, L)), \_).

gera\_ligacoes(\_, \_, [\_|[]]).

gera\_ligacoes(N, [H|T], [H, H1|\_]):- N == 10, !, gera\_ligacoes(N, T, T), \+liga(H,H1,5),

assertz(liga(H, H1, 5)).

gera\_ligacoes(N, [H|T], [H, H1|\_]):- N == 10.2, !, gera\_ligacoes(N, T, T),

\+liga(H,H1,5), assertz(liga(H1, H, 5)).

gera\_ligacoes(N, [H|T], [H, H1|\_]):- gera\_ligacoes(N, T, T), \+liga(H,H1,5),

assertz(liga(H, H1, 5)), \+liga(H1,H,5),

assertz(liga(H1, H, 5)).

Esta função cria dinamicamente factos do tipo liga(EstacaoOrigem, EstacaoDestino, Tempo).

**Métodos de pesquisa**

* **Caminho mais rápido**

O enunciado pedia para elaborar um método de pesquisa em que dadas as estações de origem e destino, determinar qual o percurso mais rápido entre ambas. Para elaborar a pesquisa, utilizamos o algoritmo A\* pois é o único que usa estimativas em termos de distância e custo acumulado.

:- use\_module(library(statistics)). % time/1

mais\_rapido(Orig,Dest,Perc,Total):-

estimativa(Orig,Dest,H), F is H + 0, % G = 0

time((hbf1([c(F/0,[Orig])],Dest,P,Total),

reverse(P,Perc))).

hbf1(Percursos,Dest,Percurso,Total):-

menor\_percurso(Percursos,Menor,Restantes),

percursos\_seguintes(Menor,Dest,Restantes,Percurso,Total).

percursos\_seguintes(c(\_/Dist,Percurso),Dest,\_,Percurso,Dist):- Percurso=[Dest|\_].

percursos\_seguintes(c(\_,[Dest|\_]),Dest,Restantes,Percurso,Total):-!,

hbf1(Restantes,Dest,Percurso,Total).

percursos\_seguintes(c(\_/Dist,[Ult|T]),Dest,Percursos,Percurso,Total):-

findall(c(F1/D1,[Z,Ult|T]),proximo\_no(Ult,T,Z,Dist,Dest,F1/D1),Lista),

append(Lista,Percursos,NovosPercursos),

hbf1(NovosPercursos,Dest,Percurso,Total).

proximo\_no(X,T,Y,Dist,Dest,F/Dist1):-

liga(X,Y,Z),

\+ member(Y,T),

Dist1 is Dist + Z,

estimativa(Y,Dest,H), F is H + Dist1.

menor\_percurso([H|Percurso], Menor, [H| Percurso1]):-

menor\_percurso(Percurso, Menor, Percurso1),

menor(H,Menor),!.

menor\_percurso([H|T],H,T).

menor(c(A1/B1,\_), c(A2/B2,\_)):- As1 is A1 + B1, As2 is A2 + B2, As2 < As1.

* Caminho com menos trocas:

menos\_trocas(Orig,Dest,Perc,Trocas):-

time((menos\_trocas1([(0,[Orig])],Dest,P,Trocas),

reverse(P,Perc))).

menos\_trocas1(Percursos,Dest,Percurso,Trocas):-

menor\_percurso\_trocas(Percursos,Menor,Restantes),

percursos\_seguintes\_trocas(Menor,Dest,Restantes,Percurso,Trocas).

menor\_percurso\_trocas([H|Percurso], Menor, [H| Percurso1]):-

menor\_percurso\_trocas(Percurso, Menor, Percurso1),

menor\_trocas(H,Menor),!.

menor\_percurso\_trocas([H|T],H,T).

menor\_trocas((A1,\_), (A2,\_)):- A2 < A1.

percursos\_seguintes\_trocas((Trocas,Percurso),Dest,\_,Percurso,Trocas):- Percurso=[Dest|\_].

percursos\_seguintes\_trocas((\_,[Dest|\_]),Dest,Restantes,Percurso,Trocas):-!,

menos\_trocas1(Restantes,Dest,Percurso,Trocas).

percursos\_seguintes\_trocas((Tro,[Ult|T]),Dest,Percursos,Percurso,Trocas):-

findall((T1,[Z,Ult|T]),proximo\_no\_trocas(Ult,T,Z,Tro,Dest,T1),Lista),

append(Lista,Percursos,NovosPercursos),

menos\_trocas1(NovosPercursos,Dest,Percurso,Trocas).

proximo\_no\_trocas(X,T,Y,Tro,\_,T1, N1, N):-

liga(X,Y,\_),

\+ member(Y,T),

(linha(N, LE), member(X,LE), member(Y,LE),

T1 is Tro, N1 is N ; T1 is Tro + 1, linha(N1, LE), member(Y, LE)).

**Análise de Complexidade**

* **Caminho mais rápido**

|  |  |  |  |
| --- | --- | --- | --- |
| **Origem** | **Destino** | **Tempo de Geração A\*** | **Solução A\*** |
| Château de Vincennes | Avron | 0,011 | ['Château de Vincennes', 'Bérault', 'Saint-Mandlé-Tourelle', 'Porte de Vincennes', 'Nation', 'Avron'] |
| Ternes | Rome | 0 | ['Ternes', 'Courcelles', 'Monceau', 'Villiers', 'Rome'] |
| Rennes | Convention | 0,001 | ['Rennes', 'Notre-Dame-des-Champs', 'Falguière', 'Pasteur', 'Volontaires', 'Vaugirard', 'Convention'] |
| Château de Vincennes | Porte Dauphine | 14803 | ['Château de Vincennes', 'Bérault', 'Saint-Mandlé-Tourelle', 'Porte de Vincennes', 'Nation', 'Gare dAusterlitz', 'Châtelet', 'Pyramides', 'Madeleine'|...] |

**Conclusão**

Este trabalho foi útil para aplicar os conhecimentos adquiridos nas aulas de Algoritmia Avançada e mostrar que é possível utilizar a linguagem Prolog em situações reais .Uma das maiores dificuldades foi a criação da base de conhecimento, uma vez que o metro é constituído por centenas de estações e contém inúmeros cruzamentos, nos quais gastamos bastante tempo para efectuar a sua modelação da maneira mais correcta e apropriada.

Esperamos que o relatório consiga exprimir a abordagem que considerámos ao realizar este trabalho.

**Bibliografia**

Site metro de paris:

<http://www.plan-metro-paris.fr/>

Horários:  
<http://www.ratp.fr/horaires/fr/ratp/metro>

Pontos turísticos:

<http://www.avidafrancesa.com/como-chegar-facilmente-nos-principais-pontos-turisticos-de-paris/>