ISEP - LEI

# ALGORITMIA AVANÇADA

## PROPOSTA DE IMPLEMENTAÇÃO DE METRO DE PARIS EM LINGUAGEM PROLOG

TURMA 3DC
1121296 – RUI SILVA
1111XXX – FRANCISCO SANTOS

# ALGORITMIA AVANÇADA

## PROPOSTA DE IMPLEMENTAÇÃO DE METRO DE PARIS EM LINGUAGEM PROLOG

### INTRODUÇÃO

No âmbito da disciplina de Algoritmia Avançada (ALGAV), foi desenvolvido um projecto utilizando as técnicas e competências adquiridas nas aulas, para a criação de um sistema de assistência a utilizadores das linhas de metro de Paris.

O metro de Paris é conhecido por ser bastante complexo e abrangir todos os pontos importantes da cidade. É uma rede poderosa de transportes públicos que é capaz de satisfazer a maior parte dos utilizadores que a usam para se deslocar na cidade.

No entanto, dada a complexidade desta rede de transportes, uma parte significante de utilizadores podem encontrar dificuldades em se orientarem.

O utilizadores podem diferenciar-se nos seguintes grupos:

- Utilizador Ávido – anda em média 1-2 vezes de metro por dia
- Utilizador Frequente – anda em média 2-3 vezes de metro por semana
- Utilizador Casual – anda em média 1 vez de metro por mês
- Utilizador Pontual – anda em média 1-2 vezes de metro por ano

O sistema que pretendemos implementar tem como objectivo ajudar os utilizadores Casuais e Pontuais a se deslocarem pela cidade com mais facilidade e com grau diminuido de complexidade.

O sistema vai ser implementado numa numa lingua de programação chamada PROLOG.

> *Definição:* ***Prolog*** *é uma linguagem de programação que se enquadra no paradigma de* ***Programação em Lógica Matemática****. É uma linguagem de uso geral que é especialmente associada com a inteligência artificial e linguística computacional. Consiste numa linguagem puramente lógica, que pode ser chamada de Prolog puro, e numa linguagem concreta, a qual acrescenta o Prolog puro com componentes extra-lógicos.*
>
> *http://pt.wikipedia.org/wiki/Prolog*

Qualquer programa desenvolvido nesta linguagem é composto por dois segmentos mais marcantes. Uma base de conhecimentos - necessária para qualquer programa, onde se encontra a informação base para ponderações e cálculos - e predicados - para manipular a informação da base de conhecimentos de forma a corresponder com o pedido pelo utilizador.

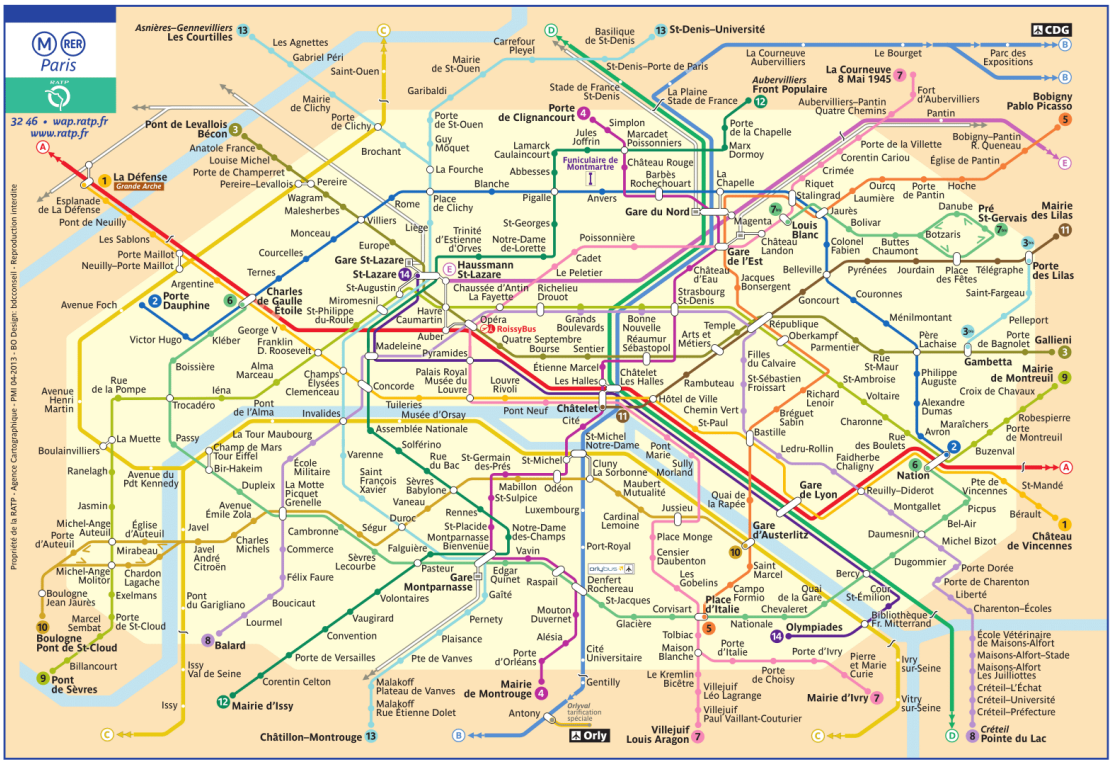
Alguns predicados são gerados em execução. O predicado "cruza" é gerado através das intersecções encontradas em cada lista, este processo é baseado no método "gera_cruzamentos" das aulas. Também o predicado "liga" torna possível a criação do grafo em memória, uma vez que liga as estações da mesma linha de forma bidirecional, para desenvolvimento dos restantes métodos de pesquisa. Pelo uso do, pre-existente, "assertz" é possível criar dinamicamente estes dados.

BASE DE CONHECIMENTO:

Para criar a Base de Conhecimento foi fornecido um conjunto de ligações (**links**) para obtermos a informação que precisamos sobre o Metro de Paris:

1. http://www.plan-metro-paris.fr/
2. http://www.ratp.fr/horaires/fr/ratp/metro
3. http://www.evous.fr/Metro-Paris.html
4. http://www.ratp.fr/en/ratp/c_21879/visiting-paris/

Do conjunto de ligações acima conseguimos extrair o mapa com as estações que constituem a rede de metro de Paris.




Também conseguimos ter acesso aos horários de cada linha e seus repectivos sentidos.

Para uma melhor organização de código, este foi separado em ficheiros diferentes, o principal "main.pl" é responsável por incluir o código dos restantes, pelo uso do "consult" e gerar a informação dinâmica pela evocação dos métodos geradores "gera_cruzamentos" e "gera_ligações". Assim a base de conhecimento apresenta a seguinte estrutura:

| nome | estrutura de dados |
|---|---|
| linha | número, lista de estações ordenadas |
| estacao | nome, |
| liga | linha, estação inicial, estação seguinte (predicado gerado em execução) |
| cruza | linha, linha 2, lista de estações em comum |
| horário | linha, destino, horas de abertura, horas de fecho, dia[1] |
| ponto de interesse | estação mais próxima, nome do ponto de interesse, horário de abertura, horário de fecho |

## PONTOS DE INTERESSE

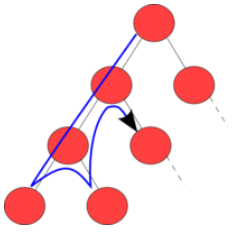
Para a implementação dos métodos para visita de meio dia, dia completo e circular, torna-se necessário o uso adicional de pontos de interesse. Foram assumidos alguns pontos de interesse, assim como os seus horários, apenas para teste do algoritmo. É possível que a mesma estação sirva mais do que um ponto de interesse e vice versa.

## ALGORITMOS DE PESQUISA

Para processamento da base de conhecimento existente, recorreu-se ao uso dos algoritmos de pesquisa lecionados nas aulas. A informação da base de conhecimento permite a criação de um grafo, onde a informação é disposta numa arvore de sequência, sendo cada nó uma estação e cada ramo uma linha. Os diversos algoritmos de pesquisa permitem, com mais ou menos eficácia, uma análise total ou parcial da arvore, determinando a melhor solução, ou seja o melhor caminho entre dois pontos.

### *Primeiro em Profundidade*

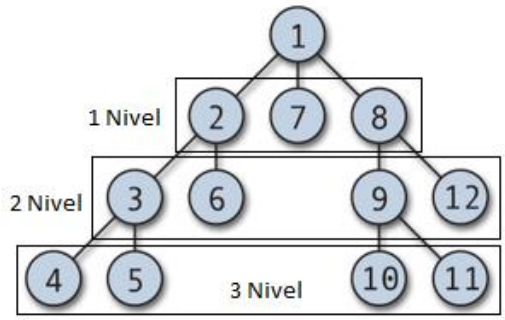
De implementação simples, este algoritmo avaliada cada nó da arvore em sequência analisando "o próximo" ao final de cada iteração. Pode ser eficaz dependendo da arvore de pesquisa usada.



[1] dia menor que 0 para domingos, segundas, terças, quartas e quintas ou 1 para sextas e sábados

### *Primeiro em Largura*

Este algoritmo permite uma análise "por níveis", sendo a sua implementação um pouco mais complexa do que o algoritmo anterior, apresenta um nível de eficácia semelhante, variando com a arvore de decisão apresentada.




### *Primeiro o Melhor*

Um algoritmo que tem por base o "Primeiro em profundidade", permite uma decisão local relativamente ao ramo a ser percorrido na iteração seguinte com base num critério de decisão, sendo esta a sua principal diferença dos restantes algoritmos, uma vez que obriga a criação de um valor numérico de "peso" para cada nó, como critério de decisão. Neste projecto o horário e os pontos de interesse servem como forma de determinar esse valor.

### *Branch and Bound*

O *Branch and Bound* é semelhante ao "primeiro o melhor", variando no pormenor de não expandir obrigatoriamente num descendente do ultimo nó. A decisão do ramo a explorar em seguida é baseado num peso acumulado entre nós, não sendo sensível à distancia do nó inicial.

### *A* (A star)*

O A* é uma função heurística, que tem em consideração pesos e medições estatísticas na determinação do melhor caminho. Mostra semelhanças com o "Primeiro o Melhor" e "*Branch and Bound*".

## ALGORITMOS DESENVOLVIDOS

### *Algoritmos geradores*

Executados no arranque do programa, o objectivo destes métodos é gerar a informação relacional consoante a base de conhecimento base. Utilizam o predicado "findall" que se trata de um algoritmo recursivo, do qual resulta uma lista. Do resultado é ainda iterado para a determinação dos elementos a serem convertidos em átomos (através do "assertz"). Os predicados gerados são: "gera_cruzamentos", "gera_ligacoes", "gera_estacoes", "gera_estacoes_linhas".

Portanto, o uso do "findall" apresenta uma complexidade $n^2$ e como o seu resultado é iterado novamente, apresenta um complexidade de $n^2 \times n^2 = n^4$.

### *Percurso com menos trocas*

Pretendemos determinar o percurso com menos trocas entre uma origem e um destino. Tal implica um caminho com o menor número de trocas de linhas. Tentamos uma implementação que é bastante parecida com a da Pesquisa em Largura com a diferença de que a cada iteração dessa pesquisa verificamos se a linha da estação anteriormente visitada corresponde à linha da estação que estamos prestes a visitar. Se esta for diferente aumentamos ao contador de trocas de linha.

Porém, tivemos algumas dificuldade as chamadas recursivas e temos essa parte onde é possivel melhorar. No presente apenas é possivel comprovar que uma dada estação Origem e outra de Destino que estejam na mesma linha têm de facto 0 trocas de linha.

Exemplo: menosTrocas('La Defense Grande Arche','Nation',C,X).

A análise da complexidade deste algoritmo corresponde a $n^4+n^2$.

### *Percurso mais rápido*

Pretendemos determinar o percurso mais rápido entre uma origem e um destino. Tal implica obter um caminho com o menor número de estações.

Procuramos para tal utilizar o algoritmo de pesquisa em largura, uma vez que as soluções surgem em ordem crescente do seu tamanho, ou seja com mais estações, ou seja, a nossa melhor solução será sempre a primeira. Através do predicado "findall" é possível percorrer, linha a linha, todas as estações. Uma vez encontrado o destino, a solução é encaminhada para um ficheiro de texto.

Este algoritmo apresenta uma complexidade acrescida relativamente aos predicados geradores, uma vez que percorre cada lista (de linha de metro) recursivamente ($n^2$ x $n^2$ = $n^4$) e no final faz uso do "reverse" para inverter a lista apresentando a complexidade:

$n^2$ x $n^2$ + $n^2$ = $n^4+n^2$.

exemplo: maisRapido('La Defense Grande Arche','Bourse',Caminho).

### *Visita dia completo*

A visita de dia completo recebe uma lista de pontos de interesse a visitar. Pela base de conhecimento é possível determinar a estação mais próxima de cada ponto, tendo em atenção o facto de que dois pontos a visitar podem partilhar a mesma estação. A partir da lista de estações a atingir, é iterado o menor caminho entre cada par.

Dado utilizar o algoritmo "Percurso mais rápido" na iteração recursiva de uma lista, o algoritmo apresenta a complexidade: n x ($n^4+n^{2)}$

exemplo: visitaDiaInteiro(['Pyramides','Bastille','Grande Arche'],Caminho).

## CONCLUSÃO

Com este trabalho compreendemos o uso que podemos dar à linguagem de programação Prolog e as suas potencialidades. Demonstrou ser uma linguagem muito diferente do que desenvolvemos até então. A forte componente lógica da linguagem torna-a de implementação complexa.

Compreendemos que existe onde melhorar neste trabalho, e que deve ser feito um melhor estudo sobre os metodos de pesquisa.

A principal dificuldade que sentimos foi o pensamento de acontecimentos de forma recursiva e a gestão de tempo para desenvolver este projecto.

## BIBLIOGRAFIA


- http://pt.wikipedia.org/wiki/Prolog
- http://www.plan-metro-paris.fr/
- http://www.ratp.fr/horaires/fr/ratp/metro
- http://www.evous.fr/Metro-Paris.html
- http://www.ratp.fr/en/ratp/c_21879/visiting-paris/