Sistemas de Representação de Conhecimento e Racíocinio (3º ano de Curso)

Trabalho de Recurso

Relatório de Desenvolvimento

Gonçalo Nogueira (a86617)

14 de julho de 2020

Resumo

Neste relatório será apresentado todo o processo de desenvolvimento do projeto de recurso no âmbito desta unidade curricular, acompanhado de uma explicação intuitiva e fundamentada, derivada das aulas teóricas assim nas aulas práticas.

Este trabalho em específico engloba:

- Parsing em Java do ficheiro cidades.xlsx e escrita para um ficheiro .pl.

- Algoritmos de procura para a resposta a algumas querys propostas, quer algoritmos sem ter em conta o peso bem como o oposto.

Conteúdo

2.1Estratégia de pesquisa não informada(cega) 3

2.2 Estratégia de pesquisa informada 3

3.1 Base do Conhecimento 4

3.2 Resolução dos pontos do trabalho 5

3.2.1 Calcular um trajeto possível entre duas cidades 5
3.2.2 Selecionar apenas cidades com uma determinada característica 6
3.2.3 Excluir uma ou mais características de cidades para um percurso 6
3.2.4 Identificar num determinado percurso qual a cidade com o maior número de ligações 7
3.2.5 Escolher o menor percurso (usando o critério do menor número de cidades percorridas) 7
3.2.6 Escolher o percurso mais rápido (usando o critério da distância) 7
3.2.7 Escolher um percurso que passe apenas por cidades "minor" 7
3.2.8 Escolher uma ou mais cidades intermédias por onde o percurso deverá obrigatoriamente passar. 7

Capítulo 1

Introdução

No âmbito da unidade curricular de Sistemas de Representação de Conhecimento e Raciocínio foi proposta a realização de um trabalho prático cujo objetivo é a motivação dos alunos face à programação em lógica que tem sido ensinada através da linguagem de programação *PROLOG*.

Este projeto denota entre outros os algoritmos de procura mais relevantes para descobrir o melhor caminho entre dois locais.

É importante mencionar o panorama do conhecimento a introduzir na nossa base de dados:

- local: #Id, 'City',latitude,longitude,'capital','admin','caracteristica' ~ {V,F,D}
- ligacao: local,local~ {V,F,D}

Todo o conteúdo que será abordado nos capítulos seguintes, este tipo de conhecimento, é importante mencionar que será sempre consistente quanto a esta definição prévia de local e ligação, demonstrando a sua utilização prática no cenário dado pelos docentes da unidade curricular.

O contexto do problema é, como definido pelo enunciado do mesmo, a realização de um sistema de representação de conhecimento e raciocínio capaz de atuar num universo de vários locais com ligações entre si. Esta contextualização será explicada com maior exatidão nos capítulos que se seguem.

Capítulo 2

Preliminares

É necessário que, para qualquer sistema, este tenha a capacidade de armazenar e manipular informação.
Para lidar com os problemas de pesquisa de rotas/caminhos existem duas formas distintas, pesquisas informadas e não informadas.
Neste trabalho utilizo sobretudo o método de pesquisa primeiro em largura para a pesquisa não informada e o método A* para a pequisa informada.

2.1 Estratégia de pesquisa não informada(cega)

Usam apenas as informações disponíveis na definição do problema.
-Primeiro em largura.
-Primeiro em profundidade.
-Custo uniforme.
-Pesquisa iterativa.
-Pesquisa bidirecional.

2.2 Estratégia de pesquisa informada

Dá-se ao algoritmo “dicas” sobre a adequação de diferentes estados.
-Pesquisa gulosa.
-Algoritmo A*.

Capítulo 3

Descrição do Trabalho

Este trabalho tem como objetivo criar uma base de conhecimento e estendê-lo de forma a caracterizar um universo de diferentes locais e as respetivas ligações entre si.

3.1 Base do Conhecimento

Inicialmente foi desenvolvido um programa em Java que faz o parsing do ficheiro fornecido pelos docentes com as cidades e as suas respetivas informações.
Este programa guarda todas as informações contidas no ficheiro .xlsx e escreve-os num ficheiro .pl para ser utilizado no trabalho e ser lido no prolog.
No ficheiro “trabalho.pl” após ser efetuado a conversão do ficheiro fornecido para a utilização em prolog foram também criadas ligações aleatórias para o contexto do problema fazer sentido.

3.2 Resolução dos pontos do trabalho

3.2.1 Calcular um trajeto possível entre duas cidades

Para calcular um trajeto possível entre 2 cidades utilizei um método de pesquisa não informada, pesquisa primeiro em largura.

```
resolvebf(Start, Goal, Final) :- bfs([Start], Goal, [], Path),naoPath(Goal,Path,[],Final).

bfs([Goal|_], Goal, Visited, [Goal|Visited]).                                   %quando chegamos ao destino
bfs([Start|Rest], Goal, Visited, Path) :- findall(Prox,
                                          (connected(Start, Prox),
                                           \+ membro(Prox, Visited),            %verifica se o proximo local ainda nao foi visitado
                                           \+ membro(Prox, Rest)),
                                           Seguintes),                          %coloca em seguintes os locais possiveis
                                          sort(Seguintes, Ordered),             %ordenar
                                          append(Rest, Ordered, Next),          %meter no fim
                                          bfs(Next, Goal, [Start|Visited], Path).   %volta a correr o algoritmo mas o que antes era start agora é o proximo local que chegamos,
                                                                                    % a queue esta updated


naoPath(Start, [], Adj, [Start|Adj]).
naoPath(Start, [First|Path], Bounded, Final) :- connected(Start, First) -> naoPath(First, Path, [Start|Bounded], Final);
                                                                           naoPath(Start, Path, Bounded, Final).
```

Na imagem acima, está o algoritmo utilizado e em comentário algumas explicações críticas sobre o dito algoritmo e a forma como foi implementado.

Para além disso a função do naoPath é.

3.2.2 Selecionar apenas cidades com uma determinada característica

Para a resolução deste ponto foi apenas criado a hasDesc que verifica se esse local tem a descrição pretendida pelo utilizador.
Essa função é inserida no findall do algoritmo, ou seja, quando o algoritmo descobre as ligações que o local onde esta tem apenas seleciona e coloca em seguintes aqueles que têm essa característica

```
resolvebf3(Start, Goal, Zat, Final) :- bfs3([Start], Goal, [], Zat, Path),naoPath(Goal,Path,[],Final).

bfs3([Goal|_], Goal, Visited, Zat, [Goal|Visited]).
bfs3([Start|Rest], Goal, Visited, Zat, Path) :- findall(Prox,
                                                (connected(Start, Prox),
                                                 \+ membro(Prox, Visited),
                                                 \+ membro(Prox, Rest),
                                                  hasDesc(Zat,Prox)),
                                                  Seguintes),
                                                sort(Seguintes, Ordered),
                                                append(Rest, Ordered, Next),
                                                bfs3(Next, Goal, [Start|Visited], Zat ,Path).



hasDesc(Zat,Prox):- local(_,Prox,_,_,_,_,Zat).
```

3.2.3 Excluir uma ou mais características de cidades para um percurso

A resolução deste ponto é apenas diferente na parte do findall onde negamos apenas a "hasDesc" que criei para ponto anterior, ou seja, quando o algoritmo está a descobrir todas as ligações que o local onde se encontra possui apenas adiciona aos seguintes aqueles que não têm a característica que se pretende excluir do processo.

```
resolvebf2(Start, Goal, Zat, Final) :- bfs2([Start], Goal, [], Zat, Path),naoPath(Goal,Path,[],Final).

bfs2([Goal|_], Goal, Visited, Zat, [Goal|Visited]).
bfs2([Start|Rest], Goal, Visited, Zat, Path) :- findall(Prox,
                                                (connected(Start, Prox),
                                                 \+ membro(Prox, Visited),
                                                 \+ membro(Prox, Rest),
                                                 \+ hasDesc(Zat,Prox)),
                                                  Seguintes),
                                                sort(Seguintes, Ordered),
                                                append(Rest, Ordered, Next),
                                                bfs2(Next, Goal, [Start|Visited], Zat ,Path).



hasDesc(Zat,Prox):- local(_,Prox,_,_,_,_,Zat).
```

3.2.4 Identificar num determinado percurso qual a cidade com o maior número de ligações

3.2.5 Escolher o menor percurso (usando o critério do menor número de cidades percorridas)

3.2.6 Escolher o percurso mais rápido (usando o critério da distância)

3.2.7 Escolher um percurso que passe apenas por cidades "minor"

A resolução deste ponto é bastante semelhante ao ponto 3.2.2 e apenas diferencia na hasZat em vez de hasDesc que em suma verifica a condição de cada local, se é admin ou minor.

```
resolvebf1(Start, Goal, Zat, Final) :- bfs1([Start], Goal, [], Zat, Path),naoPath(Goal,Path,[],Final).

bfs1([Goal|_], Goal, Visited, Zat, [Goal|Visited]).
bfs1([Start|Rest], Goal, Visited, Zat, Path) :- findall(Prox,
                                                (connected(Start, Prox),
                                                 \+ membro(Prox, Visited),
                                                 \+ membro(Prox, Rest),
                                                     hasZat(Zat,Prox)),
                                                 Seguintes),
                                                sort(Seguintes, Ordered),
                                                append(Rest, Ordered, Next),
                                                bfs1(Next, Goal, [Start|Visited], Zat ,Path).




hasZat(Zat,Prox):- local(_,Prox,_,_,_,Zat,_).
```

3.2.8 Escolher uma ou mais cidades intermédias por onde o percurso deverá obrigatoriamente passar.

Capítulo 4

Conclusões e Sugestões

Através das aulas lecionados ao longo de toda a unidade curricular, nomeadamente as práticas que forneceram um apoio muitíssimo crucial ao desenvolvimento de todo este projeto

Surgiram ao longo do projeto erros que o tive mais ou menos dificuldade a superar, mas creio que acabo com um trabalho final que considero satisfatório e essencial para a aprendizagem da unidade curricular.

Quanto a sugestões face o trabalho apresentado, não tenho nenhuma a colocar, visto que o acompanhamento das aulas permitiu ao grupo facilmente desenvolver uma base de conhecimento com os requerimentos colocados pelos docentes.

Em suma, consolidei os conhecimentos adquiridos ao longo do plano curricular de Sistemas de Representação de Conhecimento e Raciocínio relativos a métodos de procura sobretudo.