Trabalho 2
Problemas em Prolog

Disciplina: 5200 – Paradigma de Programação Lógica e Funcional

Professor: Lucas Pupulin Nanni

Introdução

O objetivo deste trabalho é a implementação de dois problemas em Prolog. Arquivos com os códigos iniciais do trabalho serão disponibilizados no Moodle. O professor realizará uma explanação detalhada do trabalho em pelo menos uma das aulas da disciplina. Todos os predicados construídos devem ser devidamente documentados com a assinatura e a descrição dos mesmos. A descrição do predicado deve conter o significado da resposta quando satisfeito e o funcionamento lógico do mesmo. Os predicados auxiliares que forem criados devem possuir um conjunto de testes junto de sua implementação.

O trabalho é em equipe de até duas pessoas. O compartilhamento de informações entre as equipes é permitido (e aconselhado), mas o compartilhamento de código não é permitido. Trabalhos que tenham porções significativas de código iguais, ou copiadas da internet, serão anulados. Veja a resolução Nº 008/2007-COU para as possíveis sanções disciplinares.

> Este trabalho vale 2/3 (dois terços) da nota do segundo bimestre.

Data de entrega: até o dia 04/08/2017 às 23:55.

Problema #1 – N-Rainhas

O problema das N-rainhas é um problema clássico da computação. O objetivo é posicionar N rainhas em um tabuleiro de xadrez NxN de forma que as rainhas não se ataquem, isto é, duas rainhas não podem estar posicionadas na mesma linha, coluna ou diagonal.

Sua tarefa é escrever dois predicados em Prolog:

- rainhas_p(?Q, +N) que é verdadeiro se Q é uma solução para o problema das N-Rainhas utilizando o processo de gerar e testar a solução. O processo de gerar e testar consiste em construir todas as possíveis configurações do problema (uma de cada vez) e então testar uma a uma se é solução.
- rainhas_r(?Q, +N) que é verdadeiro se Q é uma solução para o problema das N-Rainhas utilizando retrocesso (*backtracking*). O Processo de *backtracking* consiste em resolver um problema de tamanho N-1 recursivamente e, para cada possibilidade de posicionar uma nova rainha no tabuleiro (tornando o tamanho do problema igual a N), testar se o posicionamento é uma solução.

Dica: Utilize uma representação da solução que seja fácil de manipular e construir. Uma estratégia é representar a solução como uma lista de inteiros que indicam a linha de cada rainha, com a coluna implícita na posição do elemento na lista.

Exemplo:

```
?- rainhas_p(Q, 8).
Q = [1, 5, 8, 6, 3, 7, 2, 4] ;
Q = [1, 6, 8, 3, 7, 4, 2, 5] ;
...
```

Observe que na primeira resposta, interpretamos o posicionamento das rainhas como: uma rainha na (coluna 1 / linha 1), outra rainha na (coluna 2 / linha 5), outra rainha na (coluna 4 / linha 8), e assim sucessivamente.

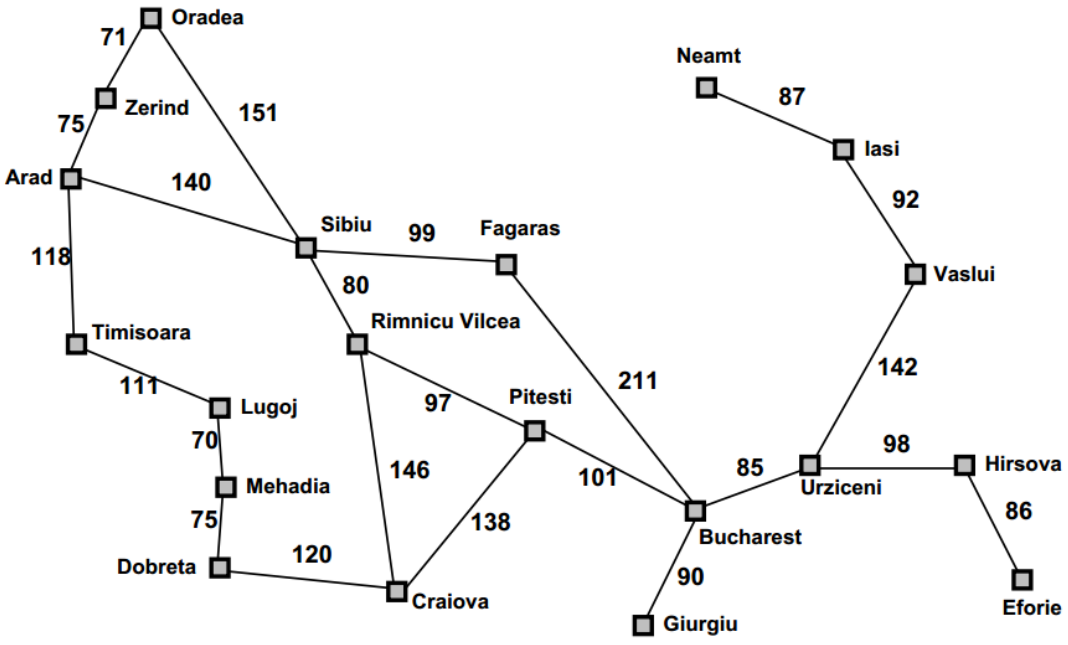
Problema #2 – Viagem à Bucareste

Você foi contratado por uma agência de viagens da Romênia que deseja descobrir os caminhos mais curtos partindo das principais cidades até a capital Bucareste. Você recebeu o mapa das principais estradas que ligam as cidades da Romênia, juntamente de uma tabela que relaciona as distâncias em linha reta entre Bucareste e as demais cidades.

Sua tarefa é implementar um predicado melhor_caminho(?O, ?C) que é verdadeiro se C é o melhor caminho partindo da cidade de origem O até Bucareste. O seu predicado deve descobrir o melhor caminho realizando uma busca gulosa melhorada com a heurística da distância em linha reta até Bucareste.

Para descobrir qual é o melhor vizinho a partir de uma cidade origem qualquer, basta escolher o vizinho com menor custo, onde o custo é a soma da distância entre a origem e a heurística do vizinho. Exemplo: o melhor vizinho de Sibiu é Rimnicu Vilcea com custo 80 + 193 = 273.

As distâncias do mapa já foram pré-calculadas com o intuito de não gerar laços no caminho. Dessa forma, não é necessário realizar esse tratamento durante a busca.



Distância em linha reta até Bucareste	
Arad	366
Bucharest	0
Craiova	160
Dobreta	238
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

Envio do trabalho

Cada problema deve ser implementado em seu arquivo particular. Nomeie os arquivos com da seguinte forma:

Problema #1 → “rainhas.pl”

Problema #2 → “bucareste.pl”

Para enviar o trabalho, crie uma versão compactada no formato zip com os arquivos fonte. Renomeie o arquivo de forma a conter o RA dos integrantes da equipe, como no exemplo a seguir.

Exemplos:

- ra12345_ra54321.zip (dois integrantes)
- ra12345.zip (um integrante)

O arquivo compactado deve ser enviado até as 23:55 do dia 04/08/2017 pelo Moodle. Arquivos compactados que não seguirem esse procedimento serão desconsiderados.

Avaliação

Este trabalho vale 2/3 (dois terços) da nota do segundo bimestre. O trabalho será avaliado de acordo com os critérios:

- Clareza lógica: a lógica nos programas deve ser clara.
- Corretude e completude: os programas têm que passar nos testes que você escrever.
- Boas práticas de programação: o código deve estar bem escrito e organizado; os recursos da linguagem devem ser usados corretamente.