

Programação Orientada a Objectos – 2006/07

Ficha Prática 4

Numa linguagem de programação orientada a objectos à sua escolha¹, construa um *framework* para resolver problemas de *procura em espaços de estados*.

1. Defina as entidades necessárias para representar *espaços de estados*. Como exemplos de espaços de estados, sugere-se a representação do problema dos *jarros de água* e do problema dos *missionários e canibais*.

2. Defina as entidades necessárias para representar *métodos de procura*. Como exemplos de métodos de procura, sugere-se a implementação dos métodos de *pesquisa em profundidade* e de *pesquisa em largura*.

3. Em função da linguagem escolhida e da experiência recolhida no desenvolvimento do *framework*, procure as respostas às seguintes questões:

a) Quais as vantagens em definirmos e implementarmos espaços de estados e métodos de procura como conceitos ortogonais?

b) O objectivo de um *framework* é permitir resolver não um problema particular mas uma classe de problemas. De que forma utilizamos um *framework* para resolver um novo problema ou adicionar um novo método de procura?

c) Um dos *frameworks* mais utilizados na desenvolvimento de aplicações dotadas de uma interface gráfica dá pelo nome de *model-view-controller (MVC)*. Investigue de que forma este *framework* está disponível para a linguagem e sistema operativo com que habitualmente trabalha.

Poderá apresentar e discutir as suas respostas durante as aulas teórico-práticas ou no horário de atendimento dos docentes da disciplina.

Notas, recursos e bibliografia:

Os métodos de pesquisa em profundidade e de pesquisa em largura podem ser implementados com recurso, respectivamente, a uma *pilha* e a uma *fila de espera*. Para evitar ciclos durante a procura, pode utilizar uma *lista* para guardar o caminho já percorrido.

Defina os métodos de procura de modo a aceitarem como parâmetro a profundidade máxima a explorar do grafo correspondente ao espaço de estados.

¹ Recomenda-se a utilização da linguagem Java na resolução dos exercícios das fichas práticas. Recorde-se que esta linguagem é utilizada no contexto de várias disciplinas. Recomenda-se também a escolha de uma segunda linguagem que lhe permita a comparação de resultados e a generalização dos conceitos estudados.

Um exemplo de uma implementação completa deste *framework* está disponível com a linguagem de programação orientada a objectos Logtalk (<http://logtalk.org/>).

Tratando-se de um problema clássico, existe abundante bibliografia sobre procura em espaços de estados. Veja-se, por exemplo, a bibliografia disponível na área de Inteligência Artificial. Também na Internet se encontra facilmente descrições pormenorizadas de problemas de espaços de estados e métodos de procura. Consulte, por exemplo, os seguintes recursos:

http://en.wikipedia.org/wiki/State_space

http://en.wikipedia.org/wiki/State_space_search

http://en.wikipedia.org/wiki/Depth-first_search

http://en.wikipedia.org/wiki/Breadth-first_search

http://en.wikipedia.org/wiki/Missionaries_and_Cannibals_Problem

http://en.wikipedia.org/wiki/Fox%2C_goose_and_bag_of_beans_puzzle

<http://www.mathsisfun.com/puzzles/measuring-puzzles-index.html>

Sugestões para trabalho complementar:

Implemente espaços de estados heurísticos e métodos de procura heurísticos. Implemente, por exemplo, o problema do *puzzle de oito* e o método de procura *trepa-colinas* (*hill-climbing*).

Defina uma interface *desempenho* que especifique um conjunto de métodos para medir o desempenho de um método de procura na resolução de um problema. Pode inspirar-se na funcionalidade análoga presente no exemplo *searching* distribuído com o Logtalk.

Defina uma interface gráfica que permita a um utilizador resolver um problema seleccionando um espaço de estados e um método de procura (permitindo também especificar a profundidade máxima).