Universidade do Minho

Mestrado Integrado em Engenharia Biomédica

Introdução aos Algoritmos e à Programação

Enunciado do Trabalho de Grupo

Ano letivo 2018/2019

Enunciado do Trabalho de Grupo

Objetivos

O presente trabalho tem como objetivo avaliar os conhecimentos adquiridos nas aulas da unidade curricular de "Introdução aos Algoritmos e à Programação", nomeadamente no que diz respeito à análise de problemas, elaboração de algoritmos e programação de computadores.

Em termos práticos, o trabalho envolverá a utilização das seguintes ferramentas:

- Ambiente de desenvolvimento integrado (e.g. Spyder) e linguagem de programação Python para a codificação;
- Editor de texto para escrita do relatório;

A resolução dos problemas deve desenrolar-se segundo as seguintes fases:

- Especificação do problema;
- Planeamento das soluções;
- Codificação;
- Verificação;
- Escrita de um breve relatório que não deverá exceder 5 páginas de tamanho A4 com a descrição do trabalho efetuado

---

Problema

Pretende-se o desenvolvimento de uma aplicação, em Python, para resolver um tipo de puzzle chamado GandaGalo. O GandaGalo é um jogo semelhante ao jogo do galo e ao sudoku: é jogado numa grelha com 'X's e 'O's, sendo que cada puzzle tem uma solução única. Tal como

nos jogos referidos anteriormente, neste também não podem existir mais do que dois símbolos iguais consecutivos em qualquer direção – horizontal, vertical ou diagonal.

Este puzzle, modelado sob a forma de um ficheiro de texto neste trabalho, é jogado numa área quadriculada, sendo que cada casa desta área pode ser de diferentes tipos:

- Casa Vazia, representada por um '.' no ficheiro de texto;
- Casa Bloqueada, representada por um '#' no ficheiro de texto;
- Casa Preenchida, representada por um 'X' ou um 'O' consoante a peça.

O objetivo do jogo é preencher o tabuleiro com 'X's e 'O's, sem deixar nenhuma casa livre.

Siga-se o exemplo ilustrado na Fig. 1:

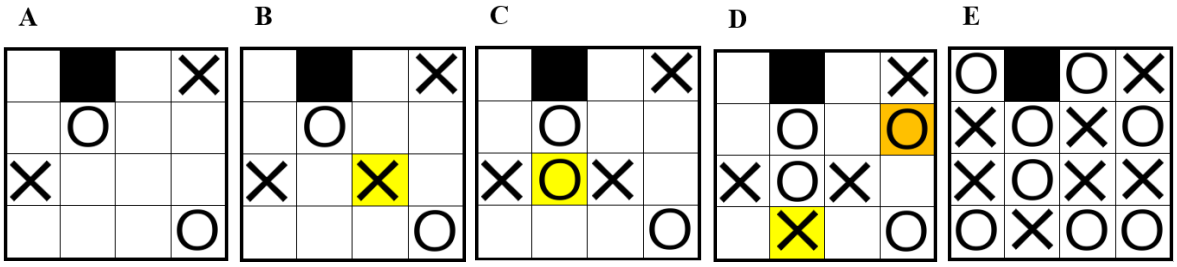

Fig. 1 – A. Tabuleiro inicial do jogo. B. Segundo movimento. C. Terceiro movimento. D. Quarto e quinto movimentos. E. Tabuleiro concluído.

- em A temos o tabuleiro inicial do jogo com uma casa bloqueada (preenchida a negro), dois 'X's e dois 'O's;
- em B temos o passo seguinte do jogo – sendo que é possível haver mais que duas peças iguais consecutivas, o movimento óbvio é a colocação de um 'X' na casa assinalada a amarelo;
- em C temos o terceiro movimento – pela mesma lógica já explicada, foi colocado um 'O' na casa assinalada a amarelo;
- em D temos o quarto e o quinto movimentos – primeiro um 'X' na casa assinalada a amarelo, seguido de um 'O' na casa assinalada a laranja;
- em E temos o tabuleiro concluído após vários movimentos – trata-se de um puzzle válido porque não ficaram casas por preencher nem há mais que duas peças ('X's e 'O's) iguais consecutivas.

Para a interface de utilizador devem criar um interpretador de comandos (Shell) com comandos que implementem as funcionalidades pretendidas.

Para facilitar a visualização do estado de um puzzle é fornecida, juntamente com este enunciado, uma classe baseada numa biblioteca gráfica desenvolvida especificamente para o

efeito. Embora o seu uso não seja obrigatório, este deverá facilitar de forma significativa o processo de desenvolvimento, nomeadamente na fase de validação. O código encontra-se devidamente comentado. A janela em que o puzzle é desenhado fica bloqueada até que ocorra um clique.

Tarefas

Pretende-se com este trabalho criar um programa em Python que permita a um utilizador preencher o tabuleiro casa a casa até ficar completo ou não existirem mais jogadas válidas possíveis. É necessário verificar que as regras de jogo são cumpridas ao longo dos movimentos executados pelo utilizador. Adicionalmente, deve ser possível pedir ao sistema para validar movimentos e/ou dar sugestões do movimento seguinte.

O programa deve implementar, no mínimo, as seguintes funcionalidades:

- Leitura de um puzzle por resolver, em ficheiro;
- Permitir jogar o puzzle;
- Gravar o estado atual de um dado puzzle em ficheiro.

Para o efeito deverá ser implementada uma Shell (ver código fornecido) em que se implementam os seguintes comandos:

mostrar comando mostrar que leva como parâmetro o nome de um ficheiro;
abrir comando carregar que leva como parâmetro o nome de um ficheiro;
gravar comando gravar que leva como parâmetro o nome de um ficheiro;
jogar comando jogar que leva como parâmetro o caractere referente à peça a ser jogada (‘X’ ou ‘O) e dois inteiros que indicam o número da linha e o número da coluna, respetivamente, onde jogar;
validar comando validar que testa a consistência do puzzle e verifica se o tabuleiro está válido;
ajuda comando ajuda que indica a próxima casa lógica a ser jogada (sem indicar a peça a ser colocada);
undo comando para anular movimentos (retroceder no jogo);
resolver comando para resolver o puzzle.

ancora comando âncora que deve guardar o ponto em que está o jogo para permitir mais tarde voltar a este ponto através do comando **undoancora**. Marcar um ponto em que se está no jogo usa-se quando se sabe que essa seria a próxima casa lógica a ser jogada, mas não se tem certeza da peça a ser colocada e, mesmo assim, pretende-se continuar a jogar a partir desse estado. Funciona como um estado temporário.
undoancora comando undo para voltar à última ancora registada;
gerar comando gerar que gera puzzles com solução única e leva três números inteiros como parâmetros: o nível de dificuldade (1 para 'fácil' e 2 para 'difícil'), o número de linhas e o número de colunas do puzzle;

ver comando para visualizar o puzzle em ambiente gráfico.
sair comando para sair do jogo.

Após cada comando, deve ser apresentado o tabuleiro e uma mensagem de confirmação de ação ou de erro.

Formato do ficheiro

Os tabuleiros podem ser retangulares ou quadrados. O formato do ficheiro é o seguinte:

- A primeira linha contém o número de linhas e o número de colunas, respetivamente;
- Cada uma das linhas subsequentes representa uma linha do tabuleiro;
- O estado de cada casa é representado por um caractere;
- Os estados estão separados por um espaço;
- Os caracteres usados são os seguintes:
    - # casa bloqueada;
    - **.** casa não bloqueada e vazia;
    - **X** casa bloqueada com um 'X';
    - **O** casa bloqueada com um 'O';

Exemplo de utilização de alguns comandos

> abrir puzzle1

```
3 4
X . X .
. . . O
X . . .
```

> jogar O 2 1

```
3 4
X . X .
O . . O
X . . .
```

> gerar 1 3 4

```
3 4
X . X .
. . . O
X . . .
```

> gerar 1 4 5

```
4 5
. O . # X
. . O . O
. X . X O
X . . . X
```

> gerar 2 4 5

```
4 5
. O . # .
. . O . .
O . . . .
. . . . .
```

Repare que o comando **jogar** não faz nenhuma verificação ao efetuar a jogada para ver se é uma jogada válida de acordo com as regras ou não. A única verificação que o comando faz é se tentar colocar uma peça numa casa bloqueada ou fora do puzzle. Nesse caso, o comando não deverá fazer nada devolvendo uma mensagem de erro de casa bloqueada ou fora do puzzle.

Observações

- Deverá programar utilizando o paradigma da programação por objetos, desenvolvendo classes próprias para a resolução do problema proposto.
- Pretende-se que na resolução deste problema sejam utilizados as "estruturas de dados" dinâmicas e os tipos abstratos de dados (caso se justifiquem) que foram dados nas aulas;
- O código deve ser bem documentado.
- Na fase de "especificação do problema" pode fazer as considerações relativas ao problema que entender mais adequadas.

Metodologia

O trabalho prático será realizado em grupo, devendo este ser constituído no máximo por 3 elementos. Cada grupo deverá enviar um email para valves@di.uminho.pt com a constituição do grupo, colocando no tema da mensagem "IAP-grupo de trabalho".

A entrega do trabalho compreende a entrega de um relatório (formato .pdf) e os módulos Python desenvolvidos num só ficheiro comprimido (.zip) com a solução do problema proposto. A entrega será efetuada por upload no portal de e-learning por um dos elementos do grupo.

A avaliação do trabalho será realizada, fundamentalmente, com base nos seguintes critérios:

- Programa desenvolvido;
- Soluções algorítmicas;
- Relatório escrito;
- Cumprimento das regras estabelecidas.

Os elementos dos grupos farão a apresentação individual do trabalho desenvolvido de acordo com o calendário que, entretanto, será publicado no portal de e-learning.

Prazo de Entrega do Trabalho: 10 de janeiro de 2019.
