Programação I
Relatório do Trabalho Prático

Jogo Drench

1ºsemestre 2015/2016

Docente: Prof. Vitor Beires Nogueira

Alunos: João Dias nº35476

Eduardo Romão nº35477

Índice

	Pág.
Introdução	3
1. Tabuleiro de Jogo	4
1.1 Base do Tabuleiro	4
1.2 Carregar Tabuleiro	4
1.3 Gerar Tabuleiro	4
2. Motor de jogo	5
3. Modo: Único Jogador	7
4. Modo: Jogador VS IA	8
5. Menu Jogo	9
Conclusão	10

Introdução

O objetivo deste trabalho prático consiste no desenvolvimento e implementação do Jogo Drench utilizando a linguagem de programação Python, o Jogo pode ser jogado como Único Jogador ou IA. Ambas os modos possuem a opção de carregar o tabuleiro a partir de um ficheiro .txt ou de gerar um tabuleiro aleatório possível.

O jogador começa o jogo na primeira casa do lado esquerdo. No modo Único Jogador o jogo consiste em concluir o tabuleiro apenas com um único número no menor número possível de jogadas, já o IA em ter a área maior do tabuleiro.

Para o desenvolvimento do Jogo foram aplicados os conhecimentos sobre a linguagem Python adquiridos durante as aulas e recorrendo à pesquisa e estudo da documentação disponibilizada na página oficial da linguagem em http://docs.python.org/.

1. Tabuleiro de jogo

Inicialmente, para criarmos o tabuleiro com números aleatórios de 1 a 6 usámos duas funções.

A primeira designada por *tabuleiro* não recebe nenhum argumento e cria uma lista bidimensional (Ex.: lista=[[],[],[]]), onde vão ser guardados os números de tabuleiro gerados aleatoriamente, depois de importado o módulo random.

A segunda função *ver_tab,* que recebe um argumento, que vai ser o tabuleiro de jogo, serve para ser visualizada na consola o tabuleiro gerado. De notar que esta função adiciona um espaço entre os números da mesma linha de modo a ser mais fácil a leitura para o utilizador.

1.1 Carregar Tabuleiro

Para o tabuleiro ser carregado a partir de um ficheiro .txt, no qual o tabuleiro já está pré-definido, lemos todas as linhas guardando-as numa lista em que cada linha corresponde a um argumento (Ex.: lista = [linha1, linha2, …]). Antes de associar o tabuleiro carregado com as coordenadas tivemos de chamar a função tabuleiro para criá-las. Recorrendo assim a dois ciclos for, conseguimos criar o tabuleiro. No primeiro ciclo, são percorridos todos os argumentos da lista, utilizando assim, o segundo ciclo para percorrer o interior de cada argumento. Logo, para cada linha e argumento o algoritmo associa-o às coordenadas.

1.2 Gerar Tabuleiro

Primeiramente, para definir o tamanho do tabuleiro criamos uma função em que verificamos se o numero introduzido pelo utilizador é um número superior a 3, de modo a não gerarmos um tabuleiro demasiado pequeno.

Para gerar o tabuleiro utilizamos a estratégia de mudar dois números vizinhos de um número com uma coordenada (linha, coluna) aleatória, gerando assim um grupo de três números iguais. Consoante o tamanho de o tabuleiro introduzido pelo o utilizador através de um input, repetimos o for que gera o grupo de três números iguais usando a seguinte equação:

Número_repetições = (linhas x colunas) // 3, tomando // como a divisão inteira.

2. Motor de jogo

Para o principal algoritmo do jogo, ou seja, a parte do código que consiste em fazer mudar a parte do tabuleiro que pertence ao utilizador para o número que ele inserir, criámos uma função que vai receber cinco argumentos.

1. Primeiro argumento vai receber o tabuleiro em que se pretende jogar;
2. O segundo e terceiro argumento recebe as coordenadas x,y onde se pretende começar a executar o algoritmo, ou seja onde se pretende começar a "inundar" o tabuleiro;
3. Terceiro argumento recebe o numero inteiro pertencente à coordenada do valor onde se vai começar a "inundar" o tabuleiro;
4. O último argumento recebe o novo número que se pretende atribuir à parte do tabuleiro pertencente ao jogador.

Esta função tem base no principio da recursividade, ou seja, ao longo da sua execução vai se chamando a ela própria até se dar como terminada. Consiste em verificar se o número em contacto com o número inicial é igual. Se não for, a função não faz nada e através de um return vazio, acaba ali. Se pelo contrário estivermos na presença de dois números iguais, ambos vão ser alterados para o número desejado pelo utilizador e a função irá entrar no modo recursivo. Ou seja, mudando esses dois valores irá verificar-se em contacto com o segundo número, existe também um número igual, que possa ser alterado de acordo com as regras. Para termos a certeza que este processo iria ser repetido em todas as direções, e não só na mesma linha ou na mesma coluna, criamos quatro if's, sendo cada um responsável por chamar a função para cada um dos quatro sentidos do tabuleiro (direita, esquerda, cima, baixo). Este processo é finalizado sempre que o número correspondente à coordenada x,y for diferente do número da coordenada onde começou a "inundação".

Na parte do motor de jogo do IA desenvolvemos um algoritmo que faz o computador jogar com "inteligência" e não jogar aleatoriamente. Para isto utilizamos uma tática que consiste em simularmos todas as jogadas, num tabuleiro de teste, utilizando os números possíveis pelo computador, e aquele número que fizer o computador ficar com mais "casas" na sua parte do tabuleiro é o selecionado. Neste algoritmo usamos duas funções. A primeira chamada *melhor_area* recebe como parâmetros uma lista que irá ter os números que serão permitidos ao computador jogar. Em seguida, esta função irá criar um dicionário onde serão colocados os números possíveis como chaves e o número de vezes que aparecerão na área do tabuleiro do computador, como valores destas chaves respetivamente. Supondo que se o computador jogasse o número 2 e ficasse com uma área correspondente a 7 números 2, iríamos ficar com o seguinte dicionário: comb ={'2' : 7}, de notar que o algarismo 2 tem de ser convertido para string. Seguidamente, fazemos uma cópia do tabuleiro original para um tabuleiro de testes usando a função *deepcopy* do módulo *copy*, evitando assim que ao fazermos alterações no tabuleiro de teste o original seja alterado, visto que será no tabuleiro de teste que iremos experimentar todas as jogadas possíveis do computador. Para tal usamos um ciclo for que vai repetir o mesmo processo para todos os números

possíveis pelo computador. Este processo chama a função *jogada* para o tabuleiro de teste primeiro usando um número da lista, e em seguida usando o número zero também no tabuleiro de teste. Decidimos usar o número 0 pois só assim conseguimos contar a área inundada por esta jogada, porque o número 0 apenas existirá na parte que pertence ao computador. Para contarmos o número de zeros criamos a função *contar* que recebe um tabuleiro como parâmetro e o número que se pretende contar no respetivo tabuleiro. Depois, através de dois ciclos for, vai percorrer todas as linhas do tabuleiro e sempre que encontrar um 0 adiciona uma unidade a uma variável contador, e no final retorna essa variável. No final reinicia-se o tabuleiro de teste para ficar novamente igual ao original e volta a fazer-se o mesmo processo para os outros números possíveis. Após termos o dicionário concluído procuramos a chave que corresponde ao maior número de vezes que o algarismo está repetido, através de uma função *max*. O número que originar uma área maior será o jogado pelo computador.

Para concluir a parte do motor do jogo criámos uma função *fim* que tem o objetivo de verificar se o jogo terminou. Esta função recebe um argumento que no caso de ser uma string igual a “solo”, irá verificar se o jogo terminou no modo único jogador, e no caso de ser uma string igual a “ia”, vai verificar se o jogo terminou para o modo jogador VS IA. No modo único jogador esta função vai percorrer o tabuleiro através de dois ciclos for, e no caso de encontrar dois elementos do tabuleiro diferentes retorna False, o que significa que o jogo não terminou. No modo contra o computador, a função vai percorrer o tabuleiro novamente com dois ciclos for e adicionar a uma lista números presentes no tabuleiro caso não estejam presentes nessa mesma lista, de forma a não ficarem repetidos. Se essa lista contiver três ou mais elementos diferentes a função irá retornar False e o jogo não terá terminado. Neste último modo adicionamos duas variáveis que vão adicionar o número de vezes que cada número aparece no tabuleiro, e caso o jogo ter terminado teremos nessas variáveis a pontuação do jogador e do computador, que depois irão ser mostradas na consola.

3. Modo: Único jogador

Para o utilizador jogar no modo único jogador criámos duas funções: uma para carregar o tabuleiro e outra para o gerar.

3.1. Carregar tabuleiro

Ao escolher esta opção chama-se a função *solo_carregar*. A função começa por recorrer à função *carregar_tabuleiro* para que seja criado o tabuleiro de jogo.

Definimos que o número máximo de jogadas possíveis do utilizador fosse o tamanho do tabuleiro vezes 2. A partir daí criamos um ciclo while em que pede ao utilizador um número. Depois de verificar se o número inserido pelo utilizador é válido, recorrendo à função *numero_valido*, faz-se então a jogada utilizando a função *jogada*. Sempre que é feito uma jogada retira-se uma unidade às vaiável jogadas.

O ciclo termina quando a função *fim* se torna verdade ou quando não existem jogadas restantes. Se o número de jogadas restantes chegar ao fim o utilizador perde o jogo e aparece a mensagem que perdeu, se conseguir acabar o jogo ainda com jogadas restantes o utilizador ganha.

3.2. Gerar tabuleiro

Em gerar tabuleiro, o algoritmo consiste no mesmo que em carregar tabuleiro. No entanto, em vez de recorrer à função *carregar_tabuleiro* para criar o tabuleiro, este algoritmo recorre à função *gerar_tabuleiro*.

4. Modo: Jogador VS IA

À semelhança do modo único jogador, para jogar neste modo criámos também duas funções: uma que carrega o tabuleiro e outra que o gera.

4.1. Carregar tabuleiro

Para utilizarmos esta opção criamos a função *ia_carregar()*. Esta função, tal como no modo único jogador, vai começar por recorrer à função *carregar tabuleiro()* para ser criado um tabuleiro de jogo.

Neste modo em que o objetivo é ficarmos com apenas dois números no tabuleiro, ganhando o jogador que tiver mais "casas" , recorremos a um ciclo while que correrá enquanto a função *fim* com parâmetros 'ia' for falsa. Dentro deste while vamos ter a função base deste modo de jogo, denominada *ia()*. Nesta última função temos a mesma base que o utilizador joga no modo único jogador, exceto o facto de não poder ser selecionado o número jogado pelo computador. Depois de chamada a função *jogada* para a jogada do utilizador, criamos uma lista com todos os números que o computador vai poder jogar, retirando o número jogado pelo utilizador e o número presente na área do computador. Tendo a lista de números possíveis, chama-se novamente a função *jogada* mas desta vez para o computador, e no argumento do número, a jogar pelo computador, chama-se a função *melhor_area* que vai escolher a melhor solução para o computador.

4.2. Gerar o tabuleiro

No modo gerar o tabuleiro o princípio é o mesmo, mas criámos a função *ia_gerar* que em vez de carregar um ficheiro .txt chama a função *gerar_tabuleiro* que gera um tabuleiro.

5. Menu Jogo

Para criarmos o menu importámos o módulo Tkinter. Chamámos ao evento root e a partir desse evento criámos os botões. Ao iniciar o jogo aparecem três botões: "Único Jogador", "IA" e "Sair". Ao clicar num botão, este chama uma determinada função. Quando se clica em "Único Jogador" e "IA" aparecem mais três opções onde se dá ao utilizador a hipótese de escolher o tipo de tabuleiro em que quer jogar ou de voltar para o menu inicial.

Conclusão

Finalizado o trabalho podemos concluir que atingimos todos os objetivos, pondo em prática grande parte da matéria dada durante o semestre e pesquisando novas práticas que enriqueceram o nosso conhecimento da linguagem de programação Python. Tendo também verificado, exaustivamente, a existência de erros no nosso programa, pensamos ter alcançado um código sem ou quase sem erros.

De notar também que na entrega deste trabalho não enviaremos uma pasta de inputs, que permitiriam testar o programa mais facilmente, por não termos o sistema operativo Linux à disposição.