

Trabalho da Unidade 01 - Code Crusher¹

Wagner Emanoel Costa

13 de março de 2023

- Prazo: 27 de Março
- Requisitos²:
 - Python 3
 - Tk para Python 3
- O programa deve estar funcional
- Entrega Atrasada Não será permitida
- Valor: 2 pontos para as partes 01 a 04 e diversão para a parte 05

Introdução

Uma semana se passou desde Asuna convenceu sua mãe a abandonar os planos do casamento arranjado. Agora, é hora de começar sua própria jornada para apoiar Kirito em seus sonhos. Há muito o que fazer, afinal ela não pode ser apenas sua companheira. É importante para ela entender a indústria de software, cibernetica e realidade virtual.

Ela convenceu Kirito-kun a ensiná-la a programar, ele não sabe das intenções dela e convencê-lo não foi difícil, bastou encará-lo com alguma reprovação quando ele hesitou por um momento e preparar o jantar de agradecimento após a primeira semana de aula é sempre um prazer. Agora ela tem que começar o dever de casa.

Kirito disse que a tarefa é simples, terminar o jogo CodeCrusher. O jogo está quase completo, faltando apenas algumas funções simples. No entanto, receber um código de jogo quase pronto é também intimidador, mas ela já sobreviveu Aincrad, lutou ao lado de Yuuki e os Sleeping Knights, aprendeu a lidar com seus medos e agora avança para concluir a tarefa diante dela.

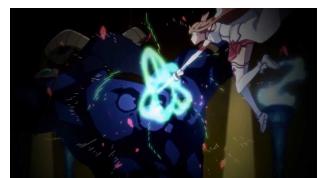
Descrição

Os jogos Bejeweled e Candy Crush são exemplos de jogos de 3-match. O objetivo desses jogos é fazer o swap das peças adjacentes para formar linhas verticais ou horizontais consistindo de 3 ou mais peças idênticas. Alguns jogos de 3-match também incluem mecanismos como power-ups que podem limpar várias peças do tabuleiro do

¹ UFPB - Campus IV
CCAE - DCX
Estrutura de Dados 1



² Necessários para rodar o exemplo e o trabalho.



jogo, ou objetivos específicos que devem ser atingidos para passar de nível, restrições de tempo, dentre outras.

Nesta atividade você vai adicionar funcionalidade ao CodeCrusher – um 3-match com temática de programação. Você receberá todo código de interface gráfica e de mouse para o jogo, bem como parte da lógica do jogo e da arte a ser utilizada. Sua tarefa é implementar as partes da lógica do jogo de forma a produzir um jogo totalmente funcional (bug free). As funções são descritas nas seções seguintes deste documento.

IMPORTANTE: Você deve seguir as instruções de implementação com precisão. Se suas funções tiverem nomes diferentes, receberem diferentes números de parâmetros, o retorne valores distintos daqueles esperados, o jogo não vai funcionar corretamente.



Parte 1 - Criar o Tabuleiro

O tabuleiro deve ser codificado como uma lista bidimensional. Cada elemento da lista será um inteiro que indica que tipo de peça reside atualmente naquela posição. Os inteiros de 0 até 5 (5 incluso) são usados para marcar as peças padrões: print, if, while, for, def e list. Espaços vazios são denotados por -1 enquanto que a peça especial que limpa todas as peças de um tipo particular é representada pelo número 6. A figura ao lado representa um tabuleiro e a lista que o representa.



Figura 1: Tabuleiro do CodeCrusher e sua representação

Crie uma função chamada `createBoard` (observe o uso de c minúsculo e B maiúsculo). A função recebe 3 parâmetros inteiros: o número de linhas no tabuleiro, o número de colunas do tabuleiro, e o número de peças únicas que podem aparecer no tabuleiro. Sua função deve retornar uma lista bidimensional com o número indicado de linhas e colunas. Cada valor na lista deve ser um inteiro aleatório r, onde r é maior ou igual a zero e menor que o número de símbolos

únicos fornecido como parâmetro da função. O código fornecido já tem a função `randrange` importada, a qual você pode usar para gerar inteiros aleatórios em uma dada faixa.

O tabuleiro da figura foi construído chamando `createBoard` com os parâmetros indicando 6 linhas, 7 colunas e 6 símbolos únicos. Como `createBoard` retorna um resultado aleatório, é provável que você obtenha diferentes inteiros no tabuleiro, mas a estrutura lista bidimensional que você criou deve coincidir com o tabuleiro mostrado.

Rode o CodeCrusher depois de implementar esta função. Testes automáticos fornecerão mensagens informando se sua função está funcionando ou não. Não avance para parte 2 até que essa função passe por todos os testes automáticos.

Parte 2 – Swap

Uma vez implementado a `createBoard` com sucesso, você pode tentar jogar o jogo, e o tabuleiro irá mostrar isso. Clicar sobre uma peça, depois clicar em uma peça vizinha (acima, abaixo, esquerda ou direita) tentará trocar as peças de posição. Embora uma animação mostrará as peças movendo-se para as novas posições, elas saltarão de volta para as posições iniciais. Este comportamento acontece porque o corpo da função `swap`, no momento, apenas contém `pass`, que é a palavra reservada de Python para nenhuma ação.

Sua próxima tarefa é uma correta implementação de `swap`. A função `swap` recebe 4 parâmetros: o tabuleiro, seguido da linha e a coluna para uma peça do jogo (`r1` e `c1`), e a linha e coluna da segunda peça do jogo (`r2` e `c2`). Sua função deve modificar o tabuleiro de forma que as peças em (`r1, c1`) e (`r2, c2`) sejam trocadas. Uma vez que sua função de `swap` esteja correta será possível jogar o jogo trocando as peças adjacentes. O jogo será fácil pois, quaisquer peças adjacentes podem ser trocadas, formem elas uma linha ou não. Na etapa 4, os swaps serão restringidos para aqueles que alinham pelo menos 3 símbolos idênticos.

Observe que a função de `swap` não retorna resultados. Ela modifica o tabuleiro passado como parâmetro.

Parte 3 – A Peça Especial

Quando 5 peças idênticas se alinham, seja em forma de linha, L ou T, as peças são removidas do tabuleiro e uma peça special é inserida no local onde o swap foi realizado. A peça especial é representada pelo símbolo da figura 3.

Esta peça pode ser trocada com qualquer peça adjacente, exceto por outra peça especial. Quando o swap é executado, a peça especial é removido do tabuleiro, junto com todas as ocorrências da outra peça





Figura 2: Peça Especial do CodeCrusher

que foi trocada com ela. Esta é um peça poderosa porque simplifica o tabuleiro e torna maiores as chances de 3-match ou mais ocorrerem. O jogador recebe 1000 (mil) pontos quando a peça é criada, algo muito útil para chegar a pontuação alvo.

Substitua a implementação atual de `clearAll` com uma nova implementação que substitua todas as ocorrências de um símbolo com espaços vazios. Sua função recebe uma lista bidimensional que representa o tabuleiro como primeiro parâmetro e receberá o símbolo que deve ser eliminado. Sua função não deve retornar resultado. Ao invés disso, ela modificará o tabuleiro passado como parâmetro. Lembre-se que espaços vazios são representados por `-1`, ou ainda melhor, use a constante `EMPTY` que já foi definida no código.

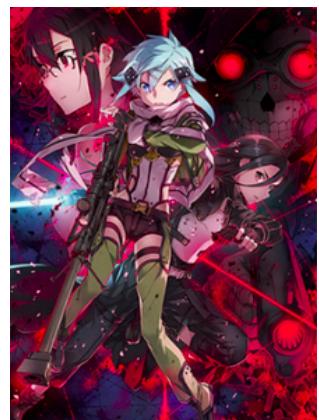
Parte 4 – Limitando os swaps

O jogo se torna mais interessante quando swaps são permitidos apenas quando 3 ou mais peças iguais alinharam. Para implementar essa restrição, crie duas funções auxiliares: `vLineAt` e `hLineAt`. Cada uma dessas funções recebe o tabuleiro como primeiro parâmetro, e uma linha e coluna representando uma posição no tabuleiro como segundo e terceiro parâmetros.

A função `vLineAt` retorna `True` se há uma linha vertical com 3 símbolos idênticos que incluem a linha e a coluna indicada. Há três maneiras que isso pode ocorrer: a posição indicada é a o topo da linha, a posição indicada está no meio da lina, ou a posição é a base da linha. Tome cuidado para garantir que sua função examine apenas posições que estão no tabuleiro. Sua função deve retornar falso quando não há linha vertical que inclua a linha e a coluna indicadas.

A função `hLineAt` comporta-se de forma similar a `vLineAt`, exceto que ela verifica linhas horizontais ao invés de verticais. Assim, a linha e a coluna indicadas podem ser a peça a esquerda do alinhamento, a do meio ou a da direita. A função deve retornar `False` quando não há alinhamento de peças idênticas que incluem a linha e a coluna indicados.

A implementação original de `canSwap` sempre retorna `True`, permitindo que quaisquer par de peças adjacentes sejam trocadas. Uma vez que você tenha `vLineAt` e `hLineAt`, você deve atualizar `canSwap` de forma que permita apenas swaps que alinhem 3 ou mais peças. O primeiro parâmetro de `canSwap` é o tabuleiro, enquanto o segundo e o terceiros parâmetros são a linha e a coluna de uma peça do tabu-



leiro, e o quarto e o quinto parâmetros são a linha e a coluna de uma segunda peça no tabuleiro. A função canSwap deve retornar True quando trocando as peças indicadas resultar em um alinhamento. Caso contrário, ela deve retornar False. Observem que o alinhamento pode incluir (r_1, c_1) , ou pode incluir (r_2, c_2) - ambas peças envolvidas no swap devem ser verificadas

Atenção também para o fato de que canSwap deve determinar se o swap pode ou não ser realizado. Ela não deve realizar o swap das peças. Como resultado, se ao final do exame, se o tabuleiro for alterado, deve-se alterá-lo de volta ao estado original antes de encerrar a função.

Embora seja livre para criar sua própria canSwap, há abaixo um sugestão de algoritmo, sinta-se a vontade para usá-la ou não.

`canSwap(r_1, c_1, r_2, c_2)`

- 1: Swap (r_1, c_1) com (r_2, c_2)
 - 2: **se** Há um alinhamento horizontal de 3 peças envolvendo a primeira posição ou
há um alinhamento horizontal de 3 peças envolvendo a segunda posição ou
Há um alinhamento vertical de 3 peças envolvendo a primeira posição ou
há um alinhamento vertical de 3 peças envolvendo a segunda posição
então
 - 3: Swap (r_1, c_1) com (r_2, c_2)
 - 4: **retorne** True
 - 5: **senão**
 - 6: Swap (r_1, c_1) com (r_2, c_2)
 - 7: **retorne** False
 - 8: **fim se**
-



Figura 3: Sugestão de canSwap

Parte 5

Só comece essa etapa, após ter cumprido as anteriores e de fazer um backup de uma versão totalmente funcional do jogo.

A próxima etapa é a personalização do jogo. O código fornecido contém um arquivo .gif de onde o jogo obtém as imagens para cada símbolo, bem como o em torno do tabuleiro. O objetivo da personalização é trocar as figuras dos símbolos para seguir um novo tema.

A personalização do jogo pode seguir quaisquer temas não ofensivo. Abaixo segue uma lista de temas sugeridos em turmas anteriores.

- Fairy Tail



- Sword Art Online
- Naruto
- DC Comics
- Neo Genesis Evangelion
- League of Legends
- Bleach
- Super Mario
- Star Wars
- Senhor dos Anéis
- Game of Thrones

