Segundo Exercício Programa (EP2) - Gemas

INFORMAÇÕES INICIAIS

Neste EP você desenvolverá uma cópia simplificada do popular jogo Bejeweled, que chamaremos de Gemas (pedras preciosas).

Para isso, você deve saber:

- criar e manipular listas,
- criar matrizes,
- percorrer linhas e colunas de matrizes, e
- escrever e utilizar funções que recebam e retornem matrizes e listas,

além dos conhecimentos exigidos no EP1.

Entrega até 4 de junho de 2017.

O JOGO

O jogo Gemas consiste em um tabuleiro com `m` colunas e `n` linhas contendo gemas de `c` cores distintas. A cada passo do jogo o jogador ou jogadora deve permutar de posição duas gemas adjacentes de tal forma que se crie uma cadeia de 3 ou mais gemas da mesma cor. Quando tal cadeia é criada, as gemas correspondentes são destruídas (eliminadas), gerando pontos para o jogador (igual ao número de gemas destruídas) e fazendo com que as gemas que se encontram acima “caiam”, tomando o lugar das gemas destruídas. Ao cair, é possível que novas cadeias se formem, causando uma reação em cadeia. Os espaços vazios criados pelas gemas que caíram são então preenchidos por gemas geradas aleatoriamente. Esse passo também pode criar novas cadeias que são automaticamente eliminadas, reiniciando o ciclo.

Duas gemas são consideradas adjacentes se elas se encontram na mesma linha e em colunas adjacentes, ou se elas se encontram na mesma coluna e em linhas adjacentes (diagonais não fazem parte da adjacência).

O jogo termina quando não existem permutações que gerem cadeias.

Tabuleiro e gemas

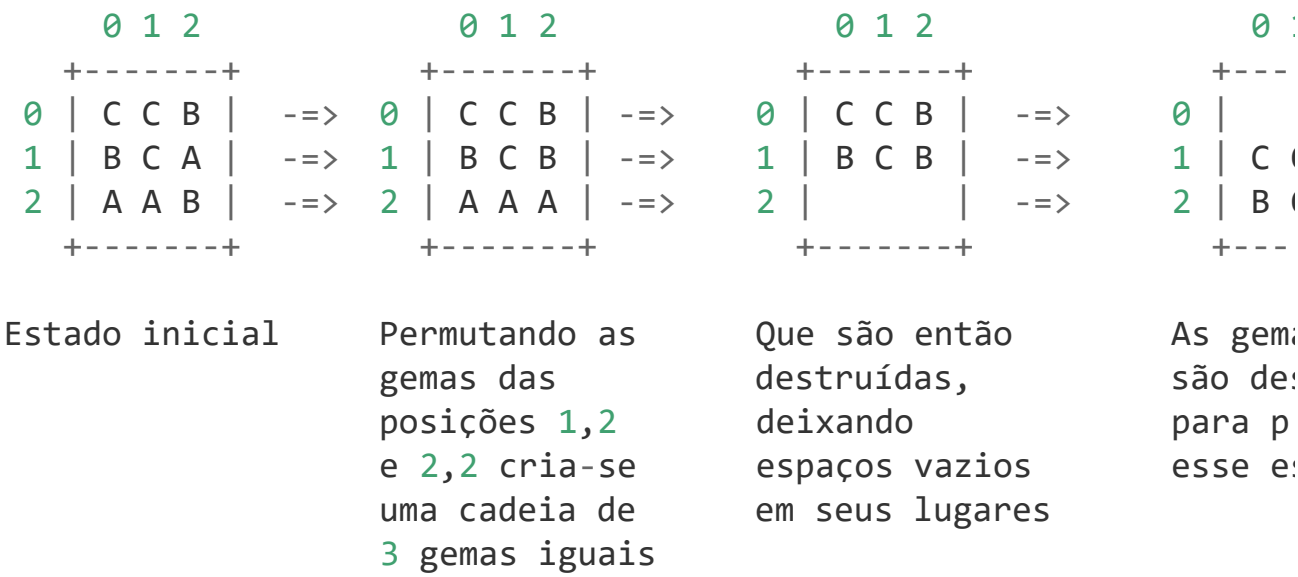
O tabuleiro deve ser representado por uma matriz `m`-por-`n` de strings, onde `m` e `n` são fornecidos pelo usuário. Cada tipo (cor) de gema é representado por uma letra maiúscula distinta, utilizando-se das `c` primeiras letras do alfabeto em caixa alta (`A`, `B`, `C`, …). Você pode assumir que o número de colunas e linhas é no máximo 10.

Exemplos

1. Tabuleiro 8-por-8 completo com 7 cores de gemas (o padrão em Bejeweled classic):

```
   0 1 2 3 4 5 6 7
  +-----------------+
0 | B B D F G D B C |
1 | A D C F A A E C |
2 | G F G D G A G A |
3 | C C G C B B A A |
4 | B D A D F D A A |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+
```

2. Sequências de configurações com tabuleiro com 3-por-2 posições e 3 cores (`A`,`B` e `C`):



Dicas

Durante o jogo, é possível obter dicas, que são fornecidas na forma da posição de uma gema cuja permutação com outra gema é válida. Cada dica obtida gera o desconto de 1 gema do total de pontos (dados pelo total de gemas destruídas).

O QUE SEU PROGRAMA DEVE FAZER

Você deverá basear a sua solução no esqueleto fornecido (se você experienciar problemas ao vizualizar o arquivo, tente mudar a codificação em seu navegador ou editor de texto para UTF8). Sua tarefa será implementar as funções necessárias para que o programa implemente uma versão funcional do jogo. **Todas as funções presentes no esqueleto devem estar na sua solução, com os mesmos argumentos e os mesmos valores de retorno**. Você pode criar funções adicionais se desejar, que devem estar devidamente documentadas.

Função `main ()`

Esta função contém o laço principal do jogo: ler comando, testar sua validade e executar comando. **Sua implementação será fornecida no esqueleto do EP2 e não deve ser modificada.**

Função `criar (num_linhas, num_colunas)`

Cria uma matriz de `num_linhas` linhas por `num_colunas` colunas e as preenche com espaços vazios representados pela string `" "`.

Exemplo

```
>>> tabuleiro = criar (8, 8)
>>> print (tabuleiro)
[[' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' '],
 [' ', ' ', ' ', ' ', ' ', ' ', ' ', ' ']]
```

Função `completar (tabuleiro, num_cores)`

Completa espaços vazios no tabuleiro gerando novas gemas cujas cores são escolhidas aleatoriamente entre as `num_cores` possíveis. **Sua implementação será fornecida no esqueleto do EP2 e não deve ser modificada.**

Exemplo

```
>>> exibir (tabuleiro)
   0 1 2 3 4 5 6 7
  +-----------------+
0 |       F G D B   |
1 | B B D F A A E   |
2 | A D C D G A G   |
3 | G F G G B B A C |
4 | B D A D F D A C |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+
>>> completar (tabuleiro, 7)
>>> exibir (tabuleiro)
   0 1 2 3 4 5 6 7
  +-----------------+
0 | A D F F G D B C |
1 | B B D F A A E A |
2 | A D C D G A G E |
```

```
3 | G F G G B B A C |
4 | B D A D F D A C |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+
>>> tabuleiro = criar (8, 8)
>>> exibir (tabuleiro)
    0 1 2 3 4 5 6 7
  +-----------------+
0 |                 |
1 |                 |
2 |                 |
3 |                 |
4 |                 |
5 |                 |
6 |                 |
7 |                 |
  +-----------------+
>>> completar (tabuleiro, 7)
>>> print (tabuleiro)
[ ['B', 'B', 'D', 'F', 'G', 'D', 'B', 'C'],
  ['A', 'D', 'C', 'F', 'A', 'A', 'E', 'C'],
  ['G', 'F', 'G', 'D', 'G', 'A', 'G', 'A'],
  ['C', 'C', 'G', 'C', 'B', 'B', 'A', 'A'],
  ['B', 'D', 'A', 'D', 'F', 'D', 'A', 'A'],
  ['G', 'D', 'E', 'C', 'D', 'B', 'G', 'G'],
  ['E', 'C', 'B', 'A', 'D', 'G', 'A', 'E'],
  ['A', 'C', 'A', 'E', 'E', 'A', 'F', 'F'] ]
```

Função `randrange` do pacote `random`

Para gerar novas gemas de maneira aleatório, usamos a função `randrange`, que retorna um inteiro (`int`) em um dado intervalo. Existem duas maneiras de chamar a função:

- `random.randrange(stop)`

retorna, com probabilidade uniforme, um inteiro entre `0` e `stop-1` (inclusive).

- `random.randrange(start, stop[, step])`

retorna, com probabilidade uniforme, um inteiro entre `start` e `stop-1`. Se `step` for fornecido, então apenas números no interval `[start,start + step, start + 2step, ..., stop-1]` são retornados.

Note que essas funções propositadamente se assemelham a função `range`, que cria uma lista de inteiros.

Função `exibir (tabuleiro)`

Exibe o tabuleiro atual identificado linhas e colunas correspondentes.

Exemplo

```
>>> tabuleiro = [ ['B', 'B', 'D', 'F', 'G', 'D', 'B', 'C'],
                  ['A', 'D', 'C', 'F', 'A', 'A', 'E', 'C'],
                  ['G', 'F', 'G', 'D', 'G', 'A', 'G', 'A'],
                  ['C', 'C', 'G', ' ', 'C', 'B', 'A', 'A'],
                  ['B', 'D', 'A', 'D', 'F', 'D', 'A', 'A'],
                  ['G', 'D', 'E', 'C', 'D', 'B', 'G', 'G'],
                  ['E', 'C', 'B', 'A', 'D', 'G', 'A', 'E'],
                  ['A', 'C', 'A', 'E', 'E', 'A', 'F', 'F'] ]
>>> exibir (tabuleiro)
    0 1 2 3 4 5 6 7
  +-----------------+
0 | B B D F G D B C |
1 | A D C F A A E C |
2 | G F G D G A G A |
3 | C C G   C B A A |
4 | B D A D F D A A |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+
```

Função `trocar (linha1, coluna1, linha2, coluna2, tabuleiro)`

Tenta permutar as gemas das posições `linha1, coluna1` e `linha2, coluna2`. Se o movimento é inválido (não gera cadeias de 3 ou mais ou se as gemas não são adjacentes), retorna `False`, caso contrário retorna `True`.

Exemplo

```
>>> tabuleiro = [ ['B', 'B', 'D', 'F', 'G', 'D', 'B', 'C'],
                  ['A', 'D', 'C', 'F', 'A', 'A', 'E', 'C'],
                  ['G', 'F', 'G', 'D', 'G', 'A', 'G', 'A'],
                  ['C', 'C', 'G', 'C', 'E', 'B', 'A', 'F'],
```

```
                    ['B', 'D', 'A', 'D', 'F', 'D', 'A', 'A'],
                    ['G', 'D', 'E', 'C', 'D', 'B', 'G', 'G'],
                    ['E', 'C', 'B', 'A', 'D', 'G', 'A', 'E'],
                    ['A', 'C', 'A', 'E', 'E', 'A', 'F', 'F'] ]
>>> exibir (tabuleiro)
    0 1 2 3 4 5 6 7
  +-----------------+
0 | B B D F G D B C |
1 | A D C F A A E C |
2 | G F G D G A G A |
3 | C C G C E B A F |
4 | B D A D F D A A |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+
>>> trocar (0, 2, 1, 2, tabuleiro) # não altera o tabuleiro
False
>>> trocar (3, 2, 3, 2, tabuleiro)
False
>>> trocar (3, 2, 3, 3, tabuleiro)
True
>>> exibir (tabuleiro)
    0 1 2 3 4 5 6 7
  +-----------------+
0 | B B D F G D B C |
1 | A D C F A A E C |
2 | G F G D G A G A |
3 | C C C G E B A F |
4 | B D A D F D A A |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+
```

Função `eliminar (tabuleiro)`

Elimina cadeias substituindo gemas destruídas por `" "` e retorna número de gemas destruídas. **Esta função deverá necessariamente fazer uso das funções:**

- `identificar_cadeias_horizontais`
- `identificar_cadeias_verticais`
- `eliminar_cadeia`

Exemplo

```
>>> exibir (tabuleiro)
    0 1 2 3 4 5 6 7
  +-----------------+
0 | B B D F G D B C |
1 | A D C F A A E C |
2 | G F G D G A G A |
3 | C C C G E B A A |
4 | B D A D F D A A |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+
>>> eliminar (tabuleiro)
6
>>> exibir (tabuleiro)
    0 1 2 3 4 5 6 7
  +-----------------+
0 | B B D F G D B C |
1 | A D C F A A E C |
2 | G F G D G A G   |
3 |       G E B A   |
4 | B D A D F D A   |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+

>>> exibir (tabuleiro2)
    0 1 2 3 4 5 6 7
  +-----------------+
0 | B B D F G D B G |
1 | A D C F A A E D |
2 | G F C D G A G G |
3 | C G C C C B A C |
4 | B D A D F D A C |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+
>>> eliminar (tabuleiro2)
5
>>> exibir (tabuleiro2)
    0 1 2 3 4 5 6 7
  +-----------------+
0 | B B D F G D B G |
1 | A D   F A A E D |
2 | G F   D G A G G |
```

```
3 | C G       B A C |
4 | B D A D F D A C |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+
```

Função `identificar_cadeias_horizontais (tabuleiro)`

Retorna uma lista contendo cadeias horizontais de 3 ou mais gemas. Cada cadeia é representada por uma lista `[linha, coluna_i, linha, coluna_f]`, onde:

- `linha`: o número da linha das gemas cadeia
- `coluna_i`: o número da coluna da gema mais à esquerda (menor) da cadeia
- `coluna_f`: o número da coluna da gema mais à direita (maior) da cadeia

Não modifica o tabuleiro.

```
>>> tabuleiro = [ ['B', 'B', 'B', 'F', 'G', 'D', 'B', 'C'],
                  ['A', 'D', 'C', 'F', 'A', 'A', 'E', 'C'],
                  ['G', 'F', 'G', 'D', 'G', 'A', 'G', 'A'],
                  ['C', 'C', 'G', 'C', '*', 'B', 'A', 'F'],
                  ['B', 'D', 'A', 'D', 'F', 'D', 'A', 'A'],
                  ['G', 'D', 'E', 'C', 'D', 'B', 'G', 'G'],
                  ['E', 'C', 'B', 'A', 'D', 'G', 'A', 'E'],
                  ['A', 'C', 'A', 'E', 'E', 'F', 'F', 'F'] ]
>>> identificar_cadeias_horizontais (tabuleiro)
[[0, 0, 0, 2], [7, 5, 7, 7]]
```

Função `identificar_cadeias_verticais (tabuleiro)`

Retorna uma lista contendo cadeias verticais de 3 ou mais gemas. Cada cadeia é representada por uma lista `[linha_i, coluna, linha_f, coluna]`, onde:

- `linha_i`: o número da linha da gema mais superior (menor) da cadeia
- `coluna`: o número da coluna das gemas da cadeia
- `linha_f`: o número da linha mais inferior (maior) da cadeia

Não modifica o tabuleiro.

Função `eliminar_cadeia (tabuleiro, cadeia)`

Elimina (substitui pela string espaço " ") as gemas compreendidas numa cadeia, representada por uma lista `[linha_inicio, coluna_inicio, linha_fim, coluna_fim]`, tal que:

- `linha_i`: o número da linha da gema mais superior (menor) da cadeia
- `coluna_i`: o número da coluna da gema mais à esquerda (menor) da cadeia
- `linha_f`: o número da linha mais inferior (maior) da cadeia
- `coluna_f`: o número da coluna da gema mais à direita (maior) da cadeia

Retorna o número de gemas eliminadas (note que se uma gema estiver em mais de uma cadeia, ela só deve é eliminada uma única vez).

Note que as funções `identifica_cadeias_horizontais` e `identifica_cadeias_verticais` retornam uma lista de cadeias (listas).

```
>>> tabuleiro = [ ['B', 'B', 'B', 'F', 'G', 'D', 'B', 'C'],
                  ['A', 'D', 'C', 'F', 'A', 'A', 'E', 'C'],
                  ['G', 'F', 'G', 'D', 'G', 'A', 'G', 'A'],
                  ['C', 'C', 'G', 'C', 'E', 'B', 'A', 'F'],
                  ['B', 'D', 'A', 'D', 'F', 'D', 'A', 'A'],
                  ['G', 'D', 'E', 'C', 'D', 'B', 'G', 'G'],
                  ['E', 'C', 'B', 'A', 'D', 'G', 'A', 'E'],
                  ['A', 'C', 'A', 'E', 'E', 'A', 'F', 'F']]
>>> eliminar_cadeia (tabuleiro, [0, 0, 0, 2])
>>> exibir (tabuleiro)
    0 1 2 3 4 5 6 7
  +-----------------+
0 |       F G D B C |
1 | A D C F A A E C |
2 | G F G D G A G A |
3 | C C C G E B A A |
4 | B D A D F D A A |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+
```

Repare que a função elimina qualquer cadeia especificada, mesmo que a cadeia não seja válida:

```
>>> eliminar_cadeia (tabuleiro, [7, 0, 7, 7])
>>> exibir (tabuleiro)
    0 1 2 3 4 5 6 7
  +-----------------+
0 |       F G D B C |
```

```
1 | A D C F A A E C |
2 | G F G D G A G A |
3 | C C C G E B A A |
4 | B D A D F D A A |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 |                 |
  +-----------------+
```

Função `deslocar (tabuleiro)`

Desloca gemas para baixo, de forma que nenhum espaço vazio apareça abaixo de uma gema. **Esta função deve necessariamente fazer uso da função `deslocar_coluna`.**

Exemplo

```
>>> exibir (tabuleiro)
    0 1 2 3 4 5 6 7
  +-----------------+
0 | B B D F G D B C |
1 | A D C F A A E C |
2 | G F G D G A G   |
3 |       G B B A   |
4 | B D A D F D A   |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+
>>> deslocar (tabuleiro)
>>> exibir (tabuleiro)
    0 1 2 3 4 5 6 7
  +-----------------+
0 |       F G D B   |
1 | B B D F A A E   |
2 | A D C D G A G   |
3 | G F G G B B A C |
4 | B D A D F D A C |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+
```

Função `deslocar_coluna (tabuleiro, i)`

Desloca as gemas na coluna i para baixo, ocupando espaços vazios abaixo.

Exemplo

```
>>> exibir (tabuleiro)
    0 1 2 3 4 5 6 7
  +-----------------+
0 | B B D F G D B C |
1 | A D C F A A E C |
2 | G F G D G A G   |
3 |       G B B A   |
4 | B D A D F D A   |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+
>>> deslocar_coluna (tabuleiro,0)
>>> exibir (tabuleiro)
    0 1 2 3 4 5 6 7
  +-----------------+
0 |   B D F G D B C |
1 | B D C F A A E C |
2 | A F G D G A G   |
3 | G     G B B A   |
4 | B D A D F D A   |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+
>>> deslocar_coluna (tabuleiro,7)
>>> exibir (tabuleiro)
    0 1 2 3 4 5 6 7
  +-----------------+
0 |   B D F G D B   |
1 | B D C F A A E   |
2 | A F G D G A G   |
3 | G     G B B A C |
4 | B D A D F D A C |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+
>>> deslocar_coluna (tabuleiro,0)
>>> exibir (tabuleiro)
    0 1 2 3 4 5 6 7
```

```
  +-----------------+
0 |   B D F G D B   |
1 | B D C F A A E   |
2 | A F G D G A G   |
3 | G     G B B A C |
4 | B D A D F D A C |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+
```

Função `existem_movimentos_validos (tabuleiro)`

Retorna `True` se existem permutações válidas e `False` caso contrário. **Esta função deverá necessariamente utilizar a função `obter_dica`.**

Função `obter_dica (tabuleiro)`

Retorna uma dica: a posição `linha, coluna` de uma gema que faz parte de uma permutação válida.

Se não houver permutação válida, retorna `-1, -1`.

```
>>> exibir (tabuleiro)
   0 1 2 3 4 5 6 7
  +-----------------+
0 | A D F F G D B C |
1 | B B D F A A E A |
2 | A D C D G A G E |
3 | G F G G B B A B |
4 | B D A D F D A C |
5 | G D E C D B G G |
6 | E C B A D G A E |
7 | A C A E E A F F |
  +-----------------+
>>> obter_dica (tabuleiro)
3, 6
>>> tabuleiro = [ ['A', 'B'],
                  ['C', 'D'],
                  ['B', 'A']]
>>> obter_dica (tabuleiro)
-1, -1
```

INFORMAÇÕES SOBRE A ENTREGA DO EP

A entrega deve ser feita até 4 de junho de 2017.

Sua solução deve se basear no esqueleto fornecido.

Todo exercício-programa deve seguir as observações contidas aqui, onde estão descritas as diretrizes para forma de entrega do exercício, aspectos importantes na avaliação etc.