Processo Seletivo 2022/2

Camisa 101 • Programação • Etapa 1

Ambiente

Os problemas foram resolvidos em Python 3.10.7 no VScode

É possivel rodar o programa em qualquer ambiente com Python 3.10.7, apenas o problema "./dificeis_p2_extra.py" depende de um output com suporte a **cores** para mostrar os resultados corretamente (não roda corretamente no terminal).

Para executar no cmd:

```
python -u "{caminho_para_a_pasta}\dificeis_p2_extra.py"
```

Estrutura do código

.vscode

- launch.json: configurações do debug
- settings.json: configurações do VScode

problemas

Template: { catogoria }_p{ numero do problema }.py

Documentação

muito_simples_p1

`distancia ( P1, P2 )` : Recebe duas coordenadas P1 e P2 em forma de lista [x, y] e retorna a distância entre elas baseado na fórmula de Pitágoras

$${\sqrt{(x1 - x2)^2 - (y1 - y2)^2}}$$

muito_simples_p2

`fahrenheitParaCelsius ( fahrenheit )` : Recebe um valor em fahrenheit e retorna em Celcius baseado na função:

$${(F-32)*5 \over 9}$$

simples_p1

`perguntaLista ( )` : Retorna uma lista de inteiros do usuário

`perguntaPosicao ( )` : Retorna uma entrada de um numero inteiro entre 0 e 9

`main ( )` : Realiza a soma de duas posições na lista

simples_p2

A função `num_perfeito ( numero )` recebe um número e cria um loop de que gera uma sequencia de numeros perfeitos até que o mesmo sejam maior ou igual que o número recebido como parâmetro. Caso o numero perfeito seja igual ao número, ele é um numero perfeito.

simples_p3

A função `encontra_primo ( lista )` recebe uma lista de números. Para cada número da lista, verifica se existe resto de divisão para cada numero entre 2 e os item da lista. Se em algum número o resto da divisão for 0, o número não é primo pois ele é divisível por um número diferente de 1 e do próprio número da lista.

simples_p3

`encontra_letra( letra, frase )`

Verifica se cada caractére da frase é igual a letra em análise (recebida como parâmetro), e retorna o número de compatibilidade

intermediarios_p1

`getAngle(A, B, C)`

Recebe 3 coordenadas (x, y) e encontra o angulo entre ABC

> Explicação matemática

Utilizando a biblioteca **math**, podemos utilizar a função `math.atan2(y, x)` (arctg) que retorna o valor de um ângulo em radianos que forma a inclinação da reta formada por (y/x).

A partir das coordenadas A, B, C, é possíver formar os vetores BA e BC, em que BA = (Ax - Bx, Ay - By) = (x, y) e o mesmo vale para BC.

No código os valores já foram substituidos na função `math.atan2(y, x)` para ambos os vetores, resultando em 2 ângulos que formam BA e BC em relação a x.

Subtraindo os ângulos, é possivel obter o angulo entre os vetores. A função `math.degrees` converte radianos em graus.

Se o angulo for negativo, o sentido (horário, antihorário) em relação ao circulo trigonométrico é trocado, resultando em um ângulo positivo entre 0 e 360.

intermediarios_p2

A função `removeDuplicatas ( lista )` utiliza o methodo `set ( lista )` para remover as duplicatas, contudo, isso tambe modifica a ordem dos elementos.

A função `removeDuplicatasOrdenado ( lista )` cria um loop que apenas adciona os itens caso não existam na nova lista, removendo as duplicatas sem mudar a ordem.

difíceis_p1

`minerar ( str )` : Recebe uma string e retorna o número de diamantes <>, contando a quantidade de < e de > eretorna a menor contagem, pois o caracter em menor número que limita a quantidade de diamantes.

difíceis_p2

> ## checkXeque (l, c, tabuleiro, preto = False)

Parâmetros

- l: linha em que se encontra a peça (y)

- c: coluna em que se encontra a peça (x)
- tabuleiro: uma lista de strings que contem cada casa do tabuleiro:
    - ["..k.....", "ppp.pppp", "........", ...]
- branco: verdadeiro caso seja o rei branco (maiúsculo)

Expansão

A partir das coordenadas do rei, executa uma expanção em uma direção, e estabelece as peças que podem comer nessa respectiva direção.

Exemplo:

```
  0 1 2 3 4 5 6 7       c0   c1   c2   c3   c4   c5   c6   c7
0 . . . . ^ . . .    l0 0[0] 0[1] 0[2] 0[3] 0[4] 0[5] 0[6] 0[7]
1 . . . . ^ . . .    l1 1[0] 1[1] 1[2] 1[3] 1[4] 1[5] 1[6] 1[7]
2 . . . . ^ . . .    l2 2[0] 2[1] 2[2] 2[3] 2[4] 2[5] 2[6] 2[7]
3 . . . . ^ . . .    l3 3[0] 3[1] 3[2] 3[3] 3[4] 3[5] 3[6] 3[7]
4 . . . . ^ . . .    l4 4[0] 4[1] 4[2] 4[3] 4[4] 4[5] 4[6] 4[7]
5 . . . . ^ . . .    l5 5[0] 5[1] 5[2] 5[3] 5[4] 5[5] 5[6] 5[7]
6 . . . . K . . .    l6 6[0] 6[1] 6[2] 6[3] 6[4] 6[5] 6[6] 6[7]
7 . . . . . . . .    l7 7[0] 7[1] 7[2] 7[3] 7[4] 7[5] 7[6] 7[7]
```

Para a direção norte, apenas a rainha (Q) e a torre (R) podem atacar, então ataque = "RQ". Já a função posição ficaria (l - i, c) em que l é a linha, c é a coluna e i é a variável do loop, ou seja, expandindo para a direção norte, temos a coordenada (6 - 1, 4), (6 - 2, 4), (6 - 3, 4)...

especifico(cods, ataque)

cods: são coordenadas das peças com movimentos específicos como o cavale e o peão

Como essas peças não se movem de forma linear, é verificado cada coordenada possível separadamente

Check (y, x, ataque)

Pode retornar: - null: casa vazia - 1: bloqueio ou fora do trabuleiro - 2: xeque

Retorna 2 - bloqueio ou fora do trabuleiro

- `ataque.find(cs)` : a peça pode atacar o rei

Retorna 1 - bloqueio ou fora do trabuleiro

- `0 > x or x > 7 or 0 > y or y > 7` : as coordenadas esão fora do tabuleiro
- `cs != '.'` : a casa contem uma peça que não está na lista e peças que podem atacar nessa direção, dessa forma impedem peças nas coordenadas sequintes de atacar o rei
    - precisa estar abaixo de `ataque.find(cs)`

difíceis_p2_extra

O sistema é o mesmo que o difíceis_p2, porém, ele aplica a função chequeXeque para todas as peças do tabuleiro, e retorna uma informação mais abrangente.

Importante! Esse sistema depende de um outpu com suporte a cores, dessa forma os resultados poderam não sair como o esperado no cmd.

De forma geral, para cada peça do tabuleiro, o sistema imprime uma tabela com toda a área de ataque: - Verde: a própria peça - Roxo/azul: peças de bloqueio - Vermelho: casas vazias na área de ataque e peças que podem atacar - Branco: resto do tabuleiro

Além disso, a função retorna os possíveis ataque nas respectivas direções, e depois de compilar tudo, imprime quais peças estão sobre a área de ataque de cada peça.