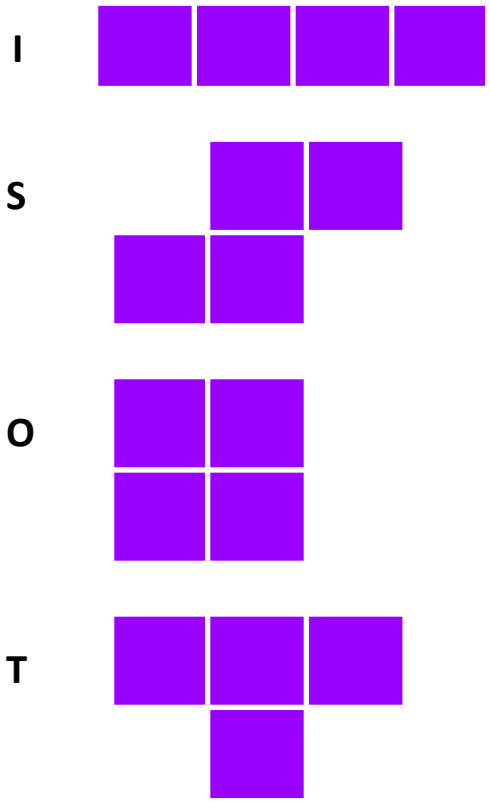
Relatório do trabalho de Programação III

Eunice Devesse - XXXX

Évora, Janeiro de 2024

1. Introdução

O trabalho consiste em criar um programa em Prolog ou OCalm (eu escolhi Prolog) para validar se uma sequência de jogadas de peças de tetris cabe no tabuleiro, considerando as regras de tetris. O tabuleiro do jogo é uma matriz de dimensão **NxM**, onde **N** é a altura do tabuleiro (isto é, o número de linhas) e **M** é a largura do tabuleiro (isto é, o número de colunas). Na implementação do programa consideram-se somente quatro (4) peças do jogo, onde cada uma é designada como apresentado a seguir:



Cada uma das peças acima apresentadas, é susceptível de ser reorientada somente por rotação, ou seja, as peças não podem ser reorientados por simetria. O programa deve admitir uma lista de de jogadas na forma de triplos (**PEÇA**, **NROT**, **NDIR**), em que:

- **PEÇA** - é o tipo de peça indicado em Prolog pelos átomos **i, s, o,** e **t**.
- **NROT** - é um número inteiro que indica o número de vezes que a peça é rodada 90^o no sentido horário (sentido dos ponteiros do relógio). Assumi-se que a rotação é sempre feita com o canto inferior esquerdo da peça mantido fixo.
- **NDIR** - é um número inteiro que indica o número de posições que a peça é empurrada para a direita antes de ser largada.
- A dimensão do Tabuleiro deve ser definida pelos predicados **n(N)**, **m(M)**.

O programa deve ser implementado por um predicado **paktris(L)**, em que L é uma lista de jogadas e que sucede ("yes") caso a sequência caiba no tabuleiro, e falhe ("no") caso contrário.

Nota: Em muitas partes deste documento, assim como nos comentários do código fonte, eu uso a designação matriz para me referir ao tabuleiro do jogo.

2. Metodologia

Nesta seção abordo sobre a metodologia usada para implementação do programa, ou seja, as decisões tomadas para representação do tabuleiro, das peças, dos movimentos e da jogabilidade.

2.1. Representação do tabuleiro

A primeira tarefa que tive foi pensar na representação do tabuleiro, o que serviu de base na tomada das decisões relacionadas às peças e todo jogo em si, afinal de contas, sem tabuleiro não há jogo.

Como o tabuleiro é uma matriz de dimensão **NxM**, então no programa este é representado como uma lista de listas, ou seja, uma lista cujos elementos são também listas (N listas), e os elementos válidos de cada lista (M elementos) são os átomos: **i, s, o, t** e **v**, onde os átomos **i, s, o** e **t** indicam que uma certa posição do tabuleiro é ocupada pela peça a que elas designam, e átomo **v** representa uma célula vazia no tabuleiro.

Exemplos de tabuleiros:

```
a) Tabulero de dimensão 4x4 com todas células vazias.

Matriz = [
    [v, v, v, v],
    [v, v, v, v],
    [v, v, v, v],
    [v, v, v, v]
]


b) Tabuleiro da alínea a) depois da jogada (i, 0, 0)

Matriz = [
    [v, v, v, v],
    [v, v, v, v],
    [v, v, v, v],
    [i, i, i, i]
]
```

```
c) Tabuleiro da alínea b) depois da jogada (s, 1, 1)

Matriz = [

    [v, s, v, v],

    [v, s, s, v],

    [v, v, s, v],

    [i, i, i, i]

]
```

No código prolog a matriz que representa o tabuleiro do jogo é gerada dinamicamente pelo predicado

```
gerar_matriz(N, M, Matriz)
```

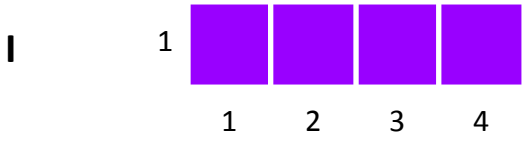
Onde as variáveis **N** e **M** indicam o número de linhas e de colunas da matriz e a variável Matriz conterá a matriz gerada no formato apresentado nos exemplos acima. ***Linhas de código: 98-104***.

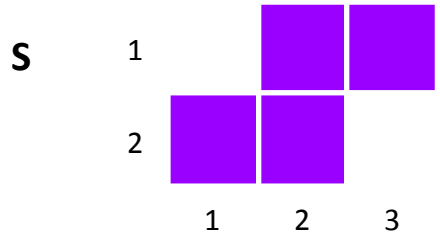
2.2. Representação das jogadas

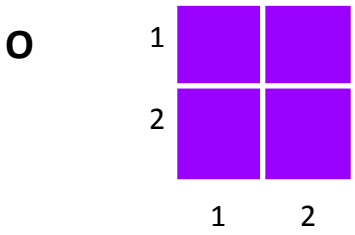
No enunciado descreve-se que cada jogada é representada como um triplo na forma **(PEÇA, NROT, NDIR)**, onde é apresentada também a explicação do que cada parâmetro indica. Com base nessa representação de jogada e olhando para a estrutura de cada uma das peças, cheguei a as seguintes conclusões:

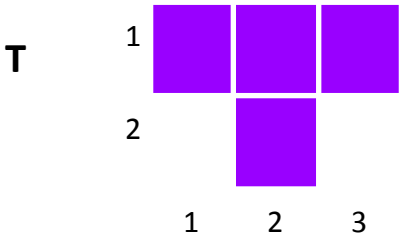
1. As peças são representadas pela sua designação (**i**, **s**, **o** e **t**).
2. Cada uma das 4 peças ocupa exatamente quatro (4) células dentro do tabuleiro.
3. Cada peça tem associada uma posição inicial dentro do tabuleiro (ou matriz).
    a. Esta posição fica situada no canto superior esquerdo do tabuleiro.
4. A posição inicial de uma peça (as células que um peça ocupa inicialmente) podem variar de acordo com as operações de rotação e deslocação para a direita aplicadas sobre ela.

A posição inicial de cada peça para o caso onde nenhuma operação de rotação ou deslocação para a direita é aplicada é a apresentada a seguir:









Onde a posição de cada peça é definida pela lista de coordenadas das células do tabuleiro que ela ocupa, ou seja:

- Para **i**: PosicaoInicial = [(1, 1), (1, 2), (1, 3), (1, 4)];
- Para **s:** PosicaoInicial = [(1, 2), (1, 3), (2, 1), (2, 3)];
- Para **o**: PosicaoInicial = [(1, 1), (1, 2), (2, 1), (1, 2)];
- Para **t**: PosicaoInicial = [(1, 1), (1, 2), (1, 3), (1, 2)];

Aplicando operação de rotação uma (1) vez em cada uma das peças, o que é linha passa a ser coluna, e o que é coluna passa a ser linha. Portanto a posição de cada peça muda para:

- Para **i**: PosicaoInicial = [(1, 1), (2, 1), (3, 1), (4, 1)];
- Para **s:** PosicaoInicial = [(1, 1), (2, 1), (2, 2), (3, 2)];
- Para **o**: PosicaoInicial = [(1, 1), (1, 2), (2, 1), (1, 2)]; (Mantêm o mesmo)
- Para **t**: PosicaoInicial = [(1, 2), (2, 1), (2, 2), (3, 2)];

Observações:

- ❖ Aplicando outras rotações (2x, 3x, 4x, …, nx), a posição final da peça pode ser facilmente achada a partir das imagens de demonstração acima, basta rodar o ecrã na mesma proporção para ver o resultado.

- ❖ A aplicação do deslocamento de **NDIR** unidades para a direita consiste no acréscimo de **NDIR** unidades na componente coluna das coordenadas de cada uma das células que a peça ocupa. Por exemplo se uma peça ocupa inicialmente a posição [(1, 1), (2, 1), (3, 1), (4, 1)], após a aplicação do deslocamento de 2 unidades, a nova posição fica [(1, 3), (2, 3), (3, 3), (4, 3)].
- ❖ A ordem de aplicação das operações é: Rotação primeiro, depois translação (ou deslocamento) para a direita.

No código prolog, a posição inicial de uma peça é representada pelo predicado:

```
posicao_peca(DesignacaoDaPeca, NROT, Posicao).
```

Onde:

- DesignacaoDaPeca - átomo que indica a designação da peça (i, s, o, t).
- NROT - número inteiro que indica o número de vezes que se aplica rotação de 90 graus sobre a peça.
- Posicao - posição da peça.

Portanto, a posição da peça retornada depende do número de rotações aplicadas. ***Linhas de código: 19-36***.

O predicado

```
translacao_coluna(Posicao, NDIR, NovaPosicao)
```

Permite a aplicação da translação de **NDIR** unidades para a direita da posição de uma peça. A variável `NovaPosicao` conterá o valor da nova posição após o deslocamento. ***Linhas de código: 49-54***.

2.3. Condições para validade das jogadas

Para validade das jogadas, uma série de condições devem ser verificadas de modo a aprová-las, caso todas passem pelas condições, e reprová-las caso alguma das jogadas não passe em pelo menos uma das condições.

Nota: Antes da verificação das condições de validade de uma jogada, primeiro são aplicadas as operações de rotação e translação para a direita de forma a encontrar a **posição de partida / posição inicial** para queda.

Para validade de uma jogada são aplicadas as seguintes operações:

1. Verificar se a posição atual da peça é válida, ou seja, se não excede os limites do tabuleiro.

2. Verificar se a posição atual da peça não tem sequer uma das células ocupadas, ou seja, se todas as posições que a peça ocupa inicialmente estão vazias.
3. Baixar a peça (deslocar em linha para baixo) uma unidade a uma, até bater com outra peça ou chegar no fundo do tabuleiro.
    a. Em cada queda de uma (1) unidade as verificações 1 e 2 são efetuadas.
    b. A queda só termina se uma das verificações 1 ou 2 falhar.
    c. A falha da verificação 1 mostra que a peça chegou no fundo do tabuleiro ou não tem espaço vazio no tabuleiro.
    d. A falha da verificação 2 mostra que a peça colidiu com outra peça.

No código prolog, o predicado

```
posicao_valida_na_matriz(Posicao)
```

verifica se uma posição de uma peça é válida na matriz (***Linhas de código: 191-195***), e o predicado

```
determinar_posicao_de_pouso_da_peca(Matriz, PosicaoInicial, PosicaoFinal)
```

Determina a posição de pouso de uma peça que é a posição da matriz onde a peça termina de cair (o fundo do tabuleiro ou o ponto onde bate com outra peça). No processo de determinar a posição, o predicado realiza todas as operações anteriormente apresentadas, retornando como posição final uma posição inválida caso as verificações 1 e 2 falhem. ***Linhas de código: 209-257***.

O predicado prolog

```
inserir_peca_na_matriz(MatrizAtual, PosicaoPeca, Peca, NovaMatriz)
```

Insere a peça Peca que se encontra na posição PosicaoPeca na Matriz. A variável NovaMatriz contém o estado da matriz depois de inserida a peça. O que este predicado faz é atualizar o estado da matriz inserindo a peça Peca. ***Linhas de código: 262-281***.

O predicado prolog

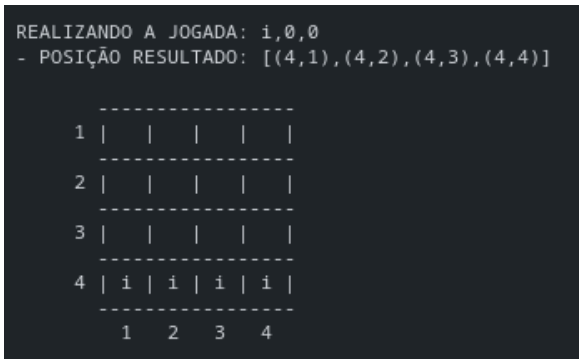
```
imprimir_matriz(Matriz)
```

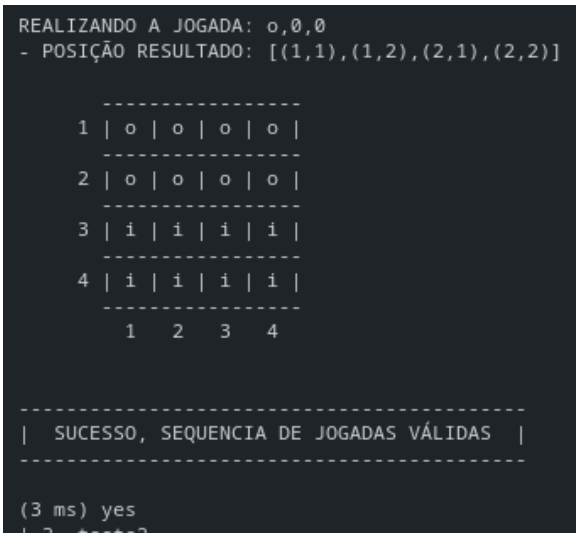
Imprime a matriz do tabuleiro na forma de uma tabela para facilitar a visualização dos resultados. ***Linhas de código: 107-150***.

Testes

Os predicados `teste1/0` e `teste2/0` podem ser usados para testar o programa usando os exemplos apresentados no enunciado. Eis os resultados dos testes:

Exemplo 1:





Exemplo 2: