INSTITUTO SUPERIOR TÉCNICO

Lógica para Programação

Projecto

18 de Abril de 2020

Solucionador de Palavras Cruzadas

Conteúdo

O objetivo deste projecto é escrever um programa em PROLOG para resolver puzzles de palavras cruzadas, de agora em diante designados apenas por "puzzles".

Um puzzle de palavras cruzadas é constituído por uma lista de palavras e uma grelha $n \times m$. Cada posição da grelha pode estar vazia, conter uma letra ou estar a negro. Na Figura 1 mostra-se um exemplo de um puzzle 5×8.

ato dao dia mae sede soar ameno drama mande

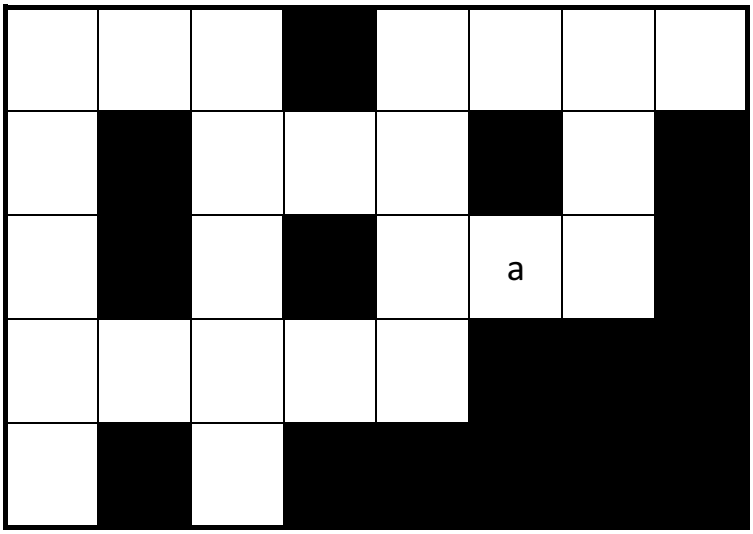

Figura 1: Puzzle de dimensão 5×8.

O objectivo é colocar todas as palavras na grelha. Por exemplo, o puzzle da Figura 1 tem uma única solução, que é apresentada na Figura 2.

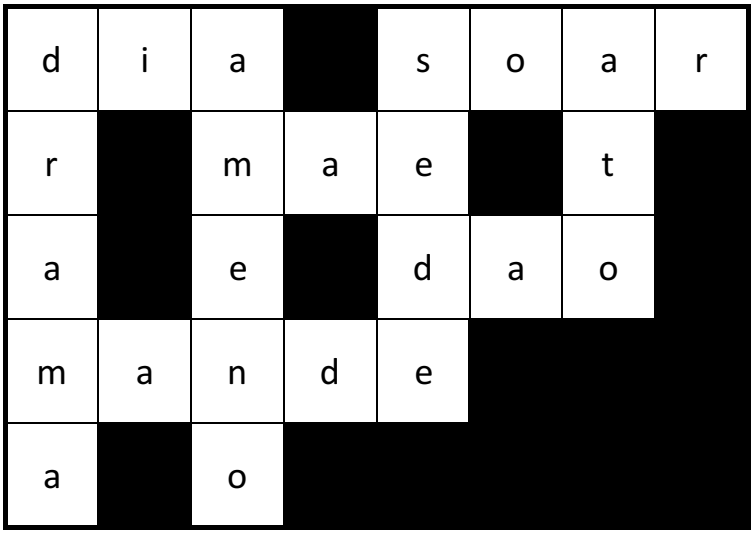

Figura 2: Solução do puzzle da Figura 1.

1 Representação de puzzles

Um puzzle é representado por uma lista de dois elementos:

- O primeiro elemento é uma lista de palavras,
- O segundo elemento é uma grelha.

Uma grelha de dimensão $n \times m$ é representada por uma lista de n listas de m elementos, em que cada uma das n listas representa uma linha do puzzle. Cada elemento é por sua vez:

- uma variável, se a posição correspondente do puzzle não estiver preenchida,
- o valor da posição correspondente do puzzle, se esta estiver preenchida,
- o símbolo "#", se a posição estiver a negro.

Por exemplo, o puzzle da Fig. 1 é representado por

```
[[ato,dao,dia,mae,sede,soar,ameno,drama,mande],
 [[P11, P12, P13,   #, P15, P16, P17, P18],
  [P21,   #, P23, P24, P25,   #, P27,   #],
  [P31,   #, P33,   #, P35,   a, P37,   #],
  [P41, P42, P43, P44, P45,   #,   #,   #],
  [P51,   #, P53,   #,   #,   #,   #,   #]]]
```

2 Abordagem

Nesta secção apresentamos o algoritmo que o seu programa deve usar na resolução de puzzles.

2.1 Inicialização

O primeiro passo na resolução de um puzzle consiste na sua inicialização. Em alguns casos, este passo é suficiente para resolver o puzzle.

Para inicializar um puzzle, devem ser seguidos os seguintes passos:

1. Obter uma lista ordenada cujos elementos são listas com as letras de cada palavra. Por exemplo, para o puzzle da Fig. 1 esta lista seria

```
[[a,m,e,n,o],[a,t,o],[d,a,o],[d,i,a],[d,r,a,m,a],
 [m,a,e],[m,a,n,d,e],[s,e,d,e],[s,o,a,r]]
```

2. Obter uma lista com os *espaços* disponíveis na grelha, aos quais podem ser atribuídas palavras. Considere que um espaço tem pelo menos três posições.[1] A lista deve estar ordenada de forma a apresentar os espaços da primeira para a última linha, e da esquerda para direita, seguidos dos espaços da primeira para a última coluna, e de cima para baixo. Por exemplo, para o puzzle da Fig. 1 esta lista seria

```
[[P11, P12, P13], [P15, P16, P17, P18],
 [P23, P24, P25],
 [P35, a, P37],
 [P41, P42, P43, P44, P45],
 [P11, P21, P31, P41, P51],
 [P13, P23, P33, P43, P53],
 [P15, P25, P35, P45],
 [P17, P27, P37]]
```

3. Obter a lista de palavras possíveis para cada espaço. Esta lista será daqui em diante designada por *lista de palavras possíveis*. Uma palavra *Pal* é possível para um espaço *Esp* se:

   - *Pal* e *Esp* tiverem o mesmo comprimento.
   - *Pal* respeitar as letras que já estejam atribuídas a elementos de *Esp*. Por exemplo, a palavra *ato* não é possível para o espaço *[P35, a, P37]*.
   - A colocação de *Pal* em *Esp* não impossibilitar o preenchimento de outros espaços com posições em comum com *Esp*. Por exemplo, a palavra *ato* não é possível para o espaço *[P11, P12, P13]*, porque a posição *P13* ficaria preenchida com *o*, esta posição é a primeira do espaço *[P13, P23, P33, P43, P53]*, e não existe nenhuma palavra de 5 letras começada por *o*.

   O resultado deste passo consiste numa lista em que cada elemento é uma lista de 2 elementos: um espaço e a lista (ordenada) de palavras possíveis para esse espaço. Por exemplo, para o puzzle da Fig. 1 a lista de palavras possíveis seria

```
[[[P11, P12, P13], [[d, i, a]]],
 [[P15, P16, P17, P18], [[s, e, d, e], [s, o, a, r]]],
 [[P23, P24, P25], [[a, t, o], [m, a, e]]],
 [[P35, a, P37], [[d, a, o]]],
 [[P41, P42, P43, P44, P45], [[m, a, n, d, e]]],
 [[P11, P21, P31, P41, P51], [[d, r, a, m, a], [m, a, n, d, e]]],
 [[P13, P23, P33, P43, P53], [[a, m, e, n, o]]],
 [[P15, P25, P35, P45], [[s, e, d, e]]],
 [[P17, P27, P37], [[a, t, o], [d, a, o]]]]
```

4. Simplificar a lista de palavras possíveis. Para tal devem ser seguidos os seguintes passos:

   (a) Atribuir letras comuns a todas as palavras possíveis. Se todas as palavras possíveis para um espaço tiverem uma letra em comum numa dada posição, essa posição do espaço deverá ser preenchida com essa letra. Por exemplo, suponhamos que as palavras possíveis para o espaço `[X, Y, Z, W]` são `[[m, a,`

[1] Isto quer dizer que todas as palavras a colocar num puzzle têm pelo menos 3 letras.

`n, a], [m, o, n, o]]`; então o espaço deve ser actualizado para `[m, Y, n, W]`. Como resultado da aplicação deste passo, a lista de palavras possíveis anterior, seria actualizada para:

```
[[[d, i, a], [[d, i, a]]],
 [[s, P16, P17, P18], [[s, e, d, e], [s, o, a, r]]],
 [[m, P24, e], [[a, t, o], [m, a, e]]],
 [[d, a, o], [[d, a, o]]],
 [[m, a, n, d, e], [[m, a, n, d, e]]],
 [[d, P21, P31, m, P51],  [[d, r, a, m, a], [m, a, n, d, e]]],
 [[a, m, e, n, o], [[a, m, e, n, o]]],
 [[s, e, d, e], [[s, e, d, e]]],
 [[P17, P27, o], [[a, t, o], [d, a, o]]]]
```

(b) Retirar palavras impossíveis: depois do passo anterior, pode acontecer que algumas listas de palavras possíveis para um espaço contenham palavras que deixaram de ser possíveis para esse espaço. É o que acontece com o terceiro elemento da lista da alínea anterior: a palavra `[a, t, o]` deixou de ser possível, e consequentemente, deve ser retirada da lista. Como resultado da aplicação deste passo, a lista de palavras possíveis anterior, seria actualizada para:

```
[[[d, i, a], [[d, i, a]]],
 [[s, P16, P17, P18], [[s, e, d, e], [s, o, a, r]]],
 [[m, P24, e], [[m, a, e]]],
 [[d, a, o], [[d, a, o]]],
 [[m, a, n, d, e], [[m, a, n, d, e]]],
 [[d, P21, P31, m, P51], [[d, r, a, m, a]]],
 [[a, m, e, n, o], [[a, m, e, n, o]]],
 [[s, e, d, e], [[s, e, d, e]]],
 [[P17, P27, o], [[a, t, o], [d, a, o]]]]
```

(c) Retirar palavras únicas: se uma palavra é a única palavra possível para um espaço, então essa palavra deve ser retirada das restantes listas de palavras possíveis. Como resultado da aplicação deste passo, a lista de palavras possíveis anterior, seria actualizada para:

```
[[[d, i, a], [[d, i, a]]],
 [[s, P16, P17, P18], [[s, o, a, r]]],
 [[m, P24, e], [[m, a, e]]],
 [[d, a, o], [[d, a, o]]],
 [[m, a, n, d, e], [[m, a, n, d, e]]],
 [[d, P21, P31, m, P51], [[d, r, a, m, a]]],
 [[a, m, e, n, o], [[a, m, e, n, o]]],
 [[s, e, d, e], [[s, e, d, e]]],
 [[P17, P27, o], [[a, t, o]]]]
```

Estes 3 passos devem ser repetidos até não haver nenhuma modificação na lista de palavras possíveis.

2.2 Resolução de listas de palavras possíveis

Se após a inicialização de um puzzle restarem posições por preencher, isto é, se ainda existirem espaços com variáveis, devem seguir-se os seguintes passos:

1. Identificar os espaços que tiverem listas de palavras possíveis com mais de uma palavra. De entre estes espaços escolher o primeiro que tenha associado um número mínimo de palavras possíveis. Por exemplo, suponhamos que tínhamos a seguinte lista de palavras possíveis:

   ```
   [[espaço1, [palavra11, palavra12, palavra13]],
    [espaço2, [palavra21, palavra22]],
    [espaço3, [palavra31]],
    [espaço4, [palavra41, palavra42]]]
   ```

   Os espaços com listas de palavras possíveis com mais de uma palavra são `espaço1`, `espaço2`,`espaço4`. Destas, `espaço2`,`espaço4` têm o número mínimo de palavras possíveis: dois. Logo, será escolhido o espaço `espaço2`.

2. Experimentar atribuir uma palavra ao espaço escolhido. Suponhamos que um dos elementos da lista de palavras possíveis era `[[c,a,l,_160,m],[[c,a,l,a,m],[c,a,l,e,m]]]`. Após a aplicação deste passo, este elemento seria substituído por `[[c,a,l,a,m],[[c,a,l,a,m]]]`.

3. Simplificar a lista de palavras possíveis, como indicado no passo 4, da Secção 2.1.

Estes 3 passos devem ser repetidos enquanto existirem espaços com um número de palavras possíveis superior a um. Se a atribuição de uma palavra a um espaço levar a uma situação impossível, retrocede-se até à última escolha. Note que o mecanismo de retrocesso do PROLOG faz precisamente isto.

2.3 Resolução de puzzles

Finalmente, e usando os algoritmos anteriormente descritos, para resolver um puzzle basta em primeiro lugar proceder à inicialização, obtendo a lista de palavras possíveis simplificada (ver Secção 2.1), e em seguida resolver esta lista (ver Secção 2.2).

3 Trabalho a desenvolver

Nesta secção são descritos os predicados que deve implementar no seu projecto. Estes serão os predicados avaliados e, consequentemente, devem respeitar escrupulosamente as especificações apresentadas. Para além dos predicados descritos, poderá implementar todos os predicados que julgar necessários.

Na implementação dos seus predicados, poderá usar os predicados fornecidos no ficheiro `codigo_comum.pl`. Para tal, deve colocar o comando `:- [codigo_comum].` no início do ficheiro que contém o seu projecto.

3.1 Predicados para a inicialização de puzzles

Nesta secção são definidos os predicados necessários para a inicialização de um puzzle, tal como foi descrito na Secção 2.1.

3.1.1 Predicado `obtem_letras_palavras/2`

Implemente o predicado `obtem_letras_palavras/2`, tal que:

`obtem_letras_palavras(Lst_Pals, Letras)`, em que `Lst_Pals` é uma lista de palavras, significa que `Letras` é a lista ordenada cujos elementos são listas com as letras de cada palavra de `Lst_Pals`, tal como descrito na Secção 2.1, no passo 1.

Por exemplo,

```
?- obtem_letras_palavras([tia, filhas, mae, filha], Letras).
Letras = [[f, i, l, h, a], [f, i, l, h, a, s], [m, a, e], [t, i, a]].
```

Sugestão: utilize os predicados pré-definidos `atom_chars/2` e `sort/2`, tais que `atom_chars(Atom, Chars)` significa que `Chars` é a lista dos componentes de `Atom`, e `sort(L1, L2)` significa que `L2` é a lista `L1` ordenada. Por exemplo,

```
?- atom_chars(mae, Chars).
Chars = [m, a, e].
?- sort([ato,dao,dia,mae,sede,soar,ameno,drama,mande], L).
L = [ameno, ato, dao, dia, drama, mae, mande, sede, soar].
```

3.1.2 Predicado `espaco_fila/2`

Implemente o predicado `espaco_fila/2`, tal que:

`espaco_fila(Fila, Esp)`, em que `Fila` é uma fila (linha ou coluna) de uma grelha, significa que `Esp` é um espaço de `Fila`, tal como descrito na Secção 2.1, no passo 2.

Por exemplo,

```
?- Fila = [#, _, _, _, _, _, #, _, #, _, _, _],
   espaco_fila(Fila, Espaco).
Fila = [#,_310,_316,_322,_328,_334,#,_346,#,_358,_364,_370],
Espaco = [_310, _316, _322, _328, _334] ;

Fila = [#,_310,_316,_322,_328,_334,#,_346,#,_358,_364,_370],
Espaco = [_358, _364, _370] ;

false.
```

3.1.3 Predicado `espacos_fila/2`

Implemente o predicado `espacos_fila/2`, tal que:

`espacos_fila(Fila, Espacos)`, em que `Fila` é uma fila (linha ou coluna) de uma grelha, significa que `Espacos` é a lista de todos os espaços de `Fila`, da esquerda para a direita.

Por exemplo,

```
?- Fila = [#, _, _, _, _, _, #, _, #, _, _, _],
   espacos_fila(Fila, Espacos).
Fila = [#,_310,_316,_322,_328,_334,#,_346,#,_358,_364,_370],
Espacos = [[_310, _316, _322, _328, _334], [_358, _364, _370]].
```

3.1.4 Predicado `espacos_puzzle/2`

Implemente o predicado `espacos_puzzle/2`, tal que:

`espacos_puzzle(Grelha, Espacos)`, em que `Grelha` é uma grelha, significa que `Espacos` é a lista de espaços de `Grelha`, tal como descrito na Secção 2.1, no passo 2. Sugestão: use o predicado `mat_transposta`, definido no ficheiro `codigo_comum.pl`.

Por exemplo,

```
?- Grelha = [[P11,P12,P13,#,P15,P16,P17,P18],
             [P21,#,P23,P24,P25,#,P27,#],
             [P31,#,P33,#,P35,a,P37,#],
             [P41,P42,P43,P44,P45,#,#,#],
             [P51,#,P53,#,#,#,#,#]],
   espacos_puzzle(Grelha, Espacos),
   Espacos == [[P11, P12, P13], [P15, P16, P17, P18], [P23, P24, P25],
               [P35, a, P37], [P41, P42, P43, P44, P45],
               [P11, P21, P31, P41, P51], [P13, P23, P33, P43, P53],
               [P15, P25, P35, P45], [P17, P27, P37]].
Grelha = [[P11, P12, P13, #, P15, P16, P17, P18],  ...
Espacos = [[P11, P12, P13], [P15, P16, P17, P18], ...
```

3.1.5 Predicado `espacos_com_posicoes_comuns/3`

Implemente o predicado `espacos_com_posicoes_comuns/3`, tal que:

`espacos_com_posicoes_comuns(Espacos, Esp, Esps_com)`, em que `Espacos` é uma lista de espaços e `Esp` é um espaço, significa que `Esps_com` é a lista de espaços com variáveis em comum com `Esp`, exceptuando `Esp`. Os espaços em `Esps_com` devem aparecer pela mesma ordem que aparecem em `Esps`.

Por exemplo,

```
?- Grelha = [[P11,P12,P13,#,P15,P16,P17,P18],
```

```
              [P21,#,P23,P24,P25,#,P27,#],
              [P31,#,P33,#,P35,a,P37,#],
              [P41,P42,P43,P44,P45,#,#,#],
              [P51,#,P53,#,#,#,#,#]],
   espacos_puzzle(Grelha, Espacos),
   nth1(1, Espacos, Esp1),
   espacos_com_posicoes_comuns(Espacos, Esp1, Esps_com) .

                                    ...
Esp1 = [P11, P12, P13],
Esps_com = [[P11, P21, P31, P41, P51], [P13, P23, P33, P43, P53]].
```

3.1.6 Predicado `palavra_possivel_esp/4`

Implemente o predicado `palavra_possivel_esp/4`, tal que:

`palavra_possivel_esp(Pal, Esp, Espacos, Letras)`, em que `Pal` é uma lista de letras de uma palavra, `Esp` é um espaço, `Espacos` é uma lista de espaços, e `Letras` é uma lista de listas de letras de palavras, significa que `Pal` é uma palavra possível para o espaço `Esp`, tal como descrito na Secção 2.1, no passo 3.

Por exemplo,

```
?-  Lst_Pals = [mae, ato, dao, dia, sede, soar, ameno, drama, mande],
    Grelha =
    [[P11,P12,P13,#,P15,P16,P17,P18],
     [P21,#,P23,P24,P25,#,P27,#],
     [P31,#,P33,#,P35,a,P37,#],
     [P41,P42,P43,P44,P45,#,#,#],
     [P51,#,P53,#,#,#,#,#]],
    obtem_letras_palavras(Lst_Pals, Letras),
    espacos_puzzle(Grelha, Espacos),
    palavra_possivel_esp([d, i, a], [P11,P12,P13], Espacos, Letras).
Lst_Pals = [mae, ato, dao, dia, sede, soar, ameno, drama, mande],
                        ...
?-  Lst_Pals = [mae, ato, dao, dia, sede, soar, ameno, drama, mande],
    Grelha =
    [[P11,P12,P13,#,P15,P16,P17,P18],
     [P21,#,P23,P24,P25,#,P27,#],
     [P31,#,P33,#,P35,a,P37,#],
     [P41,P42,P43,P44,P45,#,#,#],
     [P51,#,P53,#,#,#,#,#]],
    obtem_letras_palavras(Lst_Pals, Letras),
    espacos_puzzle(Grelha, Espacos),
    palavra_possivel_esp([a, t, o], [P11,P12,P13], Espacos, Letras).
false.
```

3.1.7 Predicado `palavras_possiveis_esp/4`

Implemente o predicado `palavras_possiveis_esp/4` , tal que:

`palavras_possiveis_esp(Letras, Espacos, Esp, Pals_Possiveis)`, em que `Letras` é uma lista de listas de letras de palavras, `Espacos` é uma lista de espaços, `Esp` é um espaço, significa que `Pals_Possiveis` é a lista ordenada de palavras possíveis para o espaço `Esp`, tal como descrito na Secção 2.1, no passo 3.

Por exemplo,

```
?-  Lst_Pals = [mae, ato, dao, dia, sede,
                soar, ameno, drama, mande],
    Grelha =
    [[P11,P12,P13,#,P15,P16,P17,P18],
     [P21,#,P23,P24,P25,#,P27,#],
     [P31,#,P33,#,P35,a,P37,#],
     [P41,P42,P43,P44,P45,#,#,#],
     [P51,#,P53,#,#,#,#,#]],
    obtem_letras_palavras(Lst_Pals, Letras),
    espacos_puzzle(Grelha, Espacos),
    palavras_possiveis_esp(Letras, Espacos,
                           [P11, P12, P13], Pals_Possiveis) .
                    ...
Pals_Possiveis = [[d, i, a]].

?-  Lst_Pals = [mae, ato, dao, dia, sede,
                soar, ameno, drama, mande],
    Grelha =
    [[P11,P12,P13,#,P15,P16,P17,P18],
     [P21,#,P23,P24,P25,#,P27,#],
     [P31,#,P33,#,P35,a,P37,#],
     [P41,P42,P43,P44,P45,#,#,#],
     [P51,#,P53,#,#,#,#,#]],
    obtem_letras_palavras(Lst_Pals, Letras),
    espacos_puzzle(Grelha, Espacos),
    palavras_possiveis_esp(Letras, Espacos,
                           [P17, P27, P37], Pals_Possiveis) .
                    ...
Pals_Possiveis = [[a, t, o], [d, a, o]].
```

3.1.8 Predicado `palavras_possiveis/3`

Implemente o predicado `palavras_possiveis/3`, tal que:

`palavras_possiveis(Letras, Espacos, Pals_Possiveis)`, em que `Letras` é uma lista de listas de letras de palavras e `Espacos` é uma lista de espaços, significa que `Pals_Possiveis` é a lista de palavras possíveis, tal como descrito na Secção 2.1, no passo 3.

Por exemplo,

```
?-  Lst_Pals = [mae, ato, dao, dia, sede, soar, ameno, drama, mande],
    Grelha =
    [[P11,P12,P13,#,P15,P16,P17,P18],
     [P21,#,P23,P24,P25,#,P27,#],
     [P31,#,P33,#,P35,a,P37,#],
     [P41,P42,P43,P44,P45,#,#,#],
     [P51,#,P53,#,#,#,#,#]],
     obtem_letras_palavras(Lst_Pals, Letras),
     espacos_puzzle(Grelha, Espacos),
     palavras_possiveis(Letras, Espacos, Pals_Possiveis),
     Pals_Possiveis ==
     [[[P11, P12, P13], [[d, i, a]]],
      [[P15, P16, P17, P18], [[s, e, d, e], [s, o, a, r]]],
      [[P23, P24, P25], [[a, t, o], [m, a, e]]],
      [[P35, a, P37], [[d, a, o]]],
      [[P41, P42, P43, P44, P45], [[m, a, n, d, e]]],
      [[P11, P21, P31, P41, P51], [[d, r, a, m, a], [m, a, n, d, e]]],
      [[P13, P23, P33, P43, P53], [[a, m, e, n, o]]],
      [[P15, P25, P35, P45], [[s, e, d, e]]],
      [[P17, P27, P37], [[a, t, o], [d, a, o]]]].
                    ...
```

3.1.9 Predicado `letras_comuns/2`

Implemente o predicado `letras_comuns/2`, tal que:

`letras_comuns(Lst_Pals, Letras_comuns)`, em que `Lst_Pals` é uma lista de listas de letras, significa que `Letras_comuns` é uma lista de pares `(pos, letra)`, significando que todas as listas de `Lst_Pals` contêm a letra `letra` na posição `pos`. Por exemplo,

```
?- Lst_Pals = [[a,t,o], [a, c, o], [a,n,o], [a,l,o]],
   letras_comuns(Lst_Pals, Letras_comuns).
Lst_Pals = [[a, t, o], [a, c, o], [a, n, o], [a, l, o]],
Letras_comuns = [(1, a),  (3, o)].
?- Lst_Pals = [[c, a, l], [d, i, a]],
   letras_comuns(Lst_Pals, Letras_comuns).
Lst_Pals = [[a, t, o], [a, c, o], [a, n, o], [a, l, o]],
```

```
Letras_comuns = [].
```

3.1.10 Predicado `atribui_comuns/1`

Implemente o predicado `atribui_comuns/1`, tal que:

`atribui_comuns(Pals_Possiveis)`, em que `Pals_Possiveis` é uma lista de palavras possíveis, actualiza esta lista atribuindo a cada espaço as letras comuns a todas as palavras possíveis para esse espaço, tal como descrito na Secção 2.1, no passo 4a.

Por exemplo, tendo `Pals_Possiveis` o valor do exemplo anterior,

```
?-  Pals_Possiveis = [[[P11, P12, P13], [[d, i, a]]],  ...,
    atribui_comuns(Pals_Possiveis),
    Pals_Possiveis ==
    [[[d, i, a], [[d, i, a]]],
     [[s, P16, P17, P18], [[s, e, d, e], [s, o, a, r]]],
     [[m, P24, e], [[a, t, o], [m, a, e]]],
     [[d, a, o], [[d, a, o]]],
     [[m, a, n, d, e], [[m, a, n, d, e]]],
     [[d, P21, P31, m, P51], [[d, r, a, m, a], [m, a, n, d, e]]],
     [[a, m, e, n, o], [[a, m, e, n, o]]],
     [[s, e, d, e], [[s, e, d, e]]],
     [[P17, P27, o], [[a, t, o], [d, a, o]]]].
Pals_Possiveis = [[[d, i, a], [[d, i, a]]],   ...
                  ...
```

3.1.11 Predicado `retira_impossiveis/2`

Implemente o predicado `retira_impossiveis/2`, tal que:

`retira_impossiveis(Pals_Possiveis, Novas_Pals_Possiveis)`,
em que `Pals_Possiveis` é uma lista de palavras possíveis, significa que `Novas_Pals_Possiveis` é o resultado de tirar palavras impossíveis de `Pals_Possiveis`, tal como descrito na Secção 2.1, no passo 4b.

Por exemplo, tendo `Pals_Possiveis` o valor do exemplo anterior após a aplicação de `atribui_comuns`

```
?-  Pals_Possiveis = [[[d, i, a], [[d, i, a]]], ...,
    retira_impossiveis(Pals_Possiveis, Novas_Pals_Possiveis),
    Novas_Pals_Possiveis ==
    [[[d, i, a], [[d, i, a]]],
     [[s, P16, P17, P18], [[s, e, d, e], [s, o, a, r]]],
     [[m, P24, e], [[m, a, e]]],
```

```
     [[d, a, o], [[d, a, o]]],
     [[m, a, n, d, e], [[m, a, n, d, e]]],
     [[d, P21, P31, m, P51], [[d, r, a, m, a]]],
     [[a, m, e, n, o], [[a, m, e, n, o]]],
     [[s, e, d, e], [[s, e, d, e]]],
     [[P17, P27, o], [[a, t, o], [d, a, o]]]].
Pals_Possiveis = [[[d, i, a], [[d, i, a]]],
                  ...
```

3.1.12 Predicado `obtem_unicas/2`

Implemente o predicado `obtem_unicas/2` , tal que:

`obtem_unicas(Pals_Possiveis, Unicas)`, em que `Pals_Possiveis` é uma lista de palavras possíveis, significa que `Unicas` é a lista de palavras únicas de `Pals_Possiveis`, tal como descrito na Secção 2.1, no passo 4c.

Por exemplo,

```
?-  Pals_Possiveis=
     [[[d, i, a], [[d, i, a]]],
      [[s, P16, P17, P18], [[s, e, d, e], [s, o, a, r]]],
      [[m, P24, e], [[m, a, e]]],
      [[d, a, o], [[d, a, o]]],
      [[m, a, n, d, e], [[m, a, n, d, e]]],
      [[d, P21, P31, m, P51], [[d, r, a, m, a]]],
      [[a, m, e, n, o], [[a, m, e, n, o]]],
      [[s, e, d, e], [[s, e, d, e]]],
      [[P17, P27, o], [[a, t, o], [d, a, o]]]],
     obtem_unicas(Pals_Possiveis, Unicas),
     writeln(Unicas).
[[d,i,a],[m,a,e],[d,a,o],[m,a,n,d,e],[d,r,a,m,a],[a,m,e,n,o],[s,e,d,e]]
Pals_Possiveis = [[[d, i, a], [[d, i, a]]],   ...
                    ...
```

3.1.13 Predicado `retira_unicas/2`

Implemente o predicado `retira_unicas/2` , tal que:

`retira_unicas(Pals_Possiveis, Novas_Pals_Possiveis)`, em que `Pals_Possiveis` é uma lista de palavras possíveis, significa que `Novas_Pals_Possiveis` é o resultado de retirar de `Pals_Possiveis` as palavras únicas, tal como descrito na Secção 2.1, no passo 4c.

Por exemplo,

```
?-  Pals_Possiveis=
     [[[d, i, a], [[d, i, a]]],
      [[s, P16, P17, P18], [[s, e, d, e], [s, o, a, r]]],
      [[m, P24, e], [[m, a, e]]],
      [[d, a, o], [[d, a, o]]],
      [[m, a, n, d, e], [[m, a, n, d, e]]],
      [[d, P21, P31, m, P51], [[d, r, a, m, a]]],
      [[a, m, e, n, o], [[a, m, e, n, o]]],
      [[s, e, d, e], [[s, e, d, e]]],
      [[P17, P27, o], [[a, t, o], [d, a, o]]]],
     retira_unicas(Pals_Possiveis, Novas_Pals_Possiveis),
     Novas_Pals_Possiveis ==
     [[[d, i, a], [[d, i, a]]],
      [[s, P16, P17, P18], [ [s, o, a, r]]],
      [[m, P24, e], [[m, a, e]]],
      [[d, a, o], [[d, a, o]]],
      [[m, a, n, d, e], [[m, a, n, d, e]]],
      [[d, P21, P31, m, P51], [[d, r, a, m, a]]],
      [[a, m, e, n, o], [[a, m, e, n, o]]],
      [[s, e, d, e], [[s, e, d, e]]],
      [[P17, P27, o], [[a, t, o]]]].
Pals_Possiveis = [[[d, i, a], [[d, i, a]]],   ...
                  ...
```

3.1.14 Predicado `simplifica/2`

Implemente o predicado `simplifica/2` , tal que:

`simplifica(Pals_Possiveis, Novas_Pals_Possiveis)`, em que `Pals_Possiveis` é uma lista de palavras possíveis, significa que `Novas_Pals_Possiveis` é o resultado de simplificar `Pals_Possiveis` , tal como descrito na Secção 2.1, no passo 4. Para simplificar uma lista de palavras possíveis, deve aplicar-lhe os predicados `atribui_comuns`, `retira_impossiveis` e `retira_unicas`, por esta ordem, até não haver mais alterações.

Por exemplo,

```
?-  Pals_Possiveis =
     [[[P11, P12, P13], [[d, i, a]]],
      [[P15, P16, P17, P18], [[s, e, d, e], [s, o, a, r]]],
      [[P23, P24, P25], [[a, t, o], [m, a, e]]],
      [[P35, a, P37], [[d, a, o]]],
      [[P41, P42, P43, P44, P45], [[m, a, n, d, e]]],
      [[P11, P21, P31, P41, P51], [[d, r, a, m, a], [m, a, n, d, e]]],
      [[P13, P23, P33, P43, P53], [[a, m, e, n, o]]],
      [[P15, P25, P35, P45], [[s, e, d, e]]],
      [[P17, P27, P37], [[a, t, o], [d, a, o]]]],
     simplifica(Pals_Possiveis, Novas_Pals_Possiveis),
```

```
     maplist(writeln, Novas_Pals_Possiveis).
[[d,i,a],[[d,i,a]]]
[[s,o,a,r],[[s,o,a,r]]]
[[m,a,e],[[m,a,e]]]
[[d,a,o],[[d,a,o]]]
[[m,a,n,d,e],[[m,a,n,d,e]]]
[[d,r,a,m,a],[[d,r,a,m,a]]]
[[a,m,e,n,o],[[a,m,e,n,o]]]
[[s,e,d,e],[[s,e,d,e]]]
[[a,t,o],[[a,t,o]]]
Pals_Possiveis = [[[d, i, a], [[d, i, a]]],   ...
                  ...
```

3.1.15 Predicado `inicializa/2`

Implemente o predicado `inicializa/2` , tal que:

`inicializa(Puz, Pals_Possiveis)`, em que `Puz` é um puzzle, significa que `Pals_Possiveis` é a lista de palavras possíveis *simplificada* para `Puz`.

Na implementação deste predicado deverá usar os predicados indicados anteriormente, de forma a seguir os passos descritos na Secção 2.1. Por exemplo,

```
?-  Puz =
     [[ato,dao,dia,mae,sede,soar,ameno,drama,mande],
      [[P11, P12, P13, #, P15, P16, P17, P18],
       [P21, #, P23, P24, P25, #, P27, #],
       [P31, #, P33, #, P35, a, P37, #],
       [P41, P42, P43, P44, P45, #, #, #],
       [P51, #, P53, #, #, #, #, #]]],
     inicializa(Puz, Pals_Possiveis),
     maplist(writeln, Pals_Possiveis).
[[d,i,a],[[d,i,a]]]
[[s,o,a,r],[[s,o,a,r]]]
[[m,a,e],[[m,a,e]]]
[[d,a,o],[[d,a,o]]]
[[m,a,n,d,e],[[m,a,n,d,e]]]
[[d,r,a,m,a],[[d,r,a,m,a]]]
[[a,m,e,n,o],[[a,m,e,n,o]]]
[[s,e,d,e],[[s,e,d,e]]]
[[a,t,o],[[a,t,o]]]
Pals_Possiveis = [[[d, i, a], [[d, i, a]]],   ...
                  ...
```

3.2 Predicados para a resolução de listas de palavras possíveis

Nesta secção são definidos os predicados necessários para a resolução de uma lista de palavras possíveis, tal como foi descrito na Secção 2.2.

3.2.1 Predicado `escolhe_menos_alternativas/2`

Implemente o predicado `escolhe_menos_alternativas/2` , tal que:

`escolhe_menos_alternativas(Pals_Possiveis, Escolha)`, em que `Pals_Possiveis` é uma lista de palavras possíveis, significa que `Escolha` é o elemento de `Pals_Possiveis` escolhido segundo o critério indicado na Secção 2.2, no passo 1. Se todos os espaços em `Pals_Possiveis` tiverem associadas listas de palavras unitárias, o predicado deve devolver "falso".

Por exemplo,

```
?- Pals_Possiveis =
     [[[P11, P12, P13], [[d, i, a], [a, t, o], [m, a, e]]],
      [[P15, P16, P17, P18], [[s, e, d, e], [s, o, a, r]]],
      [[P23, P24, P25], [[a, t, o], [m, a, e]]],
      [[P35, a, P37], [[d, a, o]]],
      [[P41, P42, P43, P44, P45], [[m, a, n, d, e]]],
      [[P11, P21, P31, P41, P51], [[d, r, a, m, a], [m, a, n, d, e]]],
      [[P13, P23, P33, P43, P53], [[a, m, e, n, o]]],
      [[P15, P25, P35, P45], [[s, e, d, e]]],
      [[P17, P27, P37], [[a, t, o], [d, a, o]]]],
    escolhe_menos_alternativas(Pals_Possiveis, Escolha).
Pals_Possiveis = [[[P11, P12, P13], ...
                   ...
Escolha = [[P15, P16, P17, P18], [[s, e, d, e], [s, o, a, r]]].

?- Pals_Possiveis =
     [[[d, i, a], [[d, i, a]]],
      [[s, o, a, r], [ [s, o, a, r]]],
      [[[m, a, e]], [[m, a, e]]],
      [[d, a, o], [[d, a, o]]],
      [[m, a, n, d, e], [[m, a, n, d, e]]],
      [[d, r, a, m, a], [[d, r, a, m, a]]],
      [[a, m, e, n, o], [[a, m, e, n, o]]],
      [[s, e, d, e], [[s, e, d, e]]],
      [[[a, t, o]], [[a, t, o]]]],
    escolhe_menos_alternativas(Pals_Possiveis, Escolha).
false.
```

3.2.2 Predicado `experimenta_pal/3`

Implemente o predicado `experimenta_pal/3` , tal que:

A chamada `experimenta_pal(Escolha, Pals_Possiveis, Novas_Pals_Possiveis)`, em que `Pals_Possiveis` é uma lista de palavras possíveis, e `Escolha` é um dos seus elementos (escolhido pelo predicado anterior), segue os seguintes passos:

1. Sendo `Esp` e `Lst_Pals` o espaço e a lista de palavras de `Escolha`, respectivamente, escolhe uma palavra de `Lst_Pals`, `Pal`. Utilize o predicado `member` para escolher esta palavra.
2. Unifica `Esp` com `Pal`.
3. `Novas_Pals_Possiveis` é o resultado de substituir, em `Pals_Possiveis`, o elemento `Escolha` pelo elemento `[Esp, [Pal]]`.

Por exemplo,

```
?- Pals_Possiveis =
     [[[P11, P12, P13], [[d, i, a], [a, t, o], [m, a, e]]],
      [[P15, P16, P17, P18], [[s, e, d, e], [s, o, a, r]]],
      [[P23, P24, P25], [[a, t, o], [m, a, e]]],
      [[P35, a, P37], [[d, a, o]]],
      [[P41, P42, P43, P44, P45], [[m, a, n, d, e]]],
      [[P11, P21, P31, P41, P51], [[d, r, a, m, a], [m, a, n, d, e]]],
      [[P13, P23, P33, P43, P53], [[a, m, e, n, o]]],
      [[P15, P25, P35, P45], [[s, e, d, e]]],
      [[P17, P27, P37], [[a, t, o], [d, a, o]]]],
    Escolha = [[P15, P16, P17, P18], [[s, e, d, e], [s, o, a, r]]],
    experimenta_pal(Escolha, Pals_Possiveis, Novas_Pals_Possiveis).
                                  ...
P15 = s,
P16 = P18, P18 = e,
P17 = d,
....
Novas_Pals_Possiveis =
[[[P11, P12, P13], [[d, i, a], [a, t, o], [m, a, e]]],
 [[s, e, d, e], [[s, e, d, e]]],
 [[P23, P24, P25], [[a, t, o], [m, a|...]]], .....
                    ;
P15 = s,
P16 = o,
P17 = a,
P18 = r,
....
Novas_Pals_Possiveis =
[[[P11, P12, P13], [[d, i, a], [a, t, o], [m, a, e]]],
 [[s, o, a, r], [[s, o, a, r]]],
```

```
 [[P23, P24, P25], [[a, t, o], [m, a|...]]], .....
```

3.2.3 Predicado `resolve_aux/2`

Implemente o predicado `resolve_aux/2` , tal que:

`resolve_aux(Pals_Possiveis, Novas_Pals_Possiveis)`, em que `Pals_Possiveis` é uma lista de palavras possíveis, significa que `Novas_Pals_Possiveis` é o resultado de aplicar o algoritmo descrito na Secção 2.2 a `Pals_Possiveis`.

Por exemplo,

```
?- Puz =
    [[aia,dai,dao,das,dei,dia,diz,doa,doi,ida,ira],
    [[#,#,#,P14,P15,P16,#,P18],
     [#,P22,P23,P24,#,P26,P27,P28],
     [#,P32,#,P34,P35,P36,#,P38],
     [P41,P42,P43,#,P45,#,#,#],
     [#,#,#,P54,P55,P56,#,#]]],
    inicializa(Puz, Pals_Possiveis),
    resolve_aux(Pals_Possiveis, Novas_Pals_Possiveis),
    maplist(writeln, Novas_Pals_Possiveis).
[[d,a,i],[[d,a,i]]]
[[d,e,i],[[d,e,i]]]
[[d,o,a],[[d,o,a]]]
[[a,i,a],[[a,i,a]]]
[[d,i,z],[[d,i,z]]]
[[d,a,o],[[d,a,o]]]
[[d,o,i],[[d,o,i]]]
[[d,i,a],[[d,i,a]]]
[[i,r,a],[[i,r,a]]]
[[i,d,a],[[i,d,a]]]
[[d,a,s],[[d,a,s]]]
Puz = [[aia, dai, ...
```

3.3 Predicados para a resolução de puzzles

3.3.1 Predicado `resolve/1`

Implemente o predicado `resolve/1` , tal que:

`resolve(Puz)`, em que `Puz` é um puzzle, resolve esse puzzle, isto é, após a invocação deste predicado a grelha de `Puz` tem todas as variáveis substituídas por letras que constituem as palavras da lista de palavras de `Puz`.

Por exemplo,

```
?- Puz =
     [[aia,dai,dao,das,dei,dia,diz,doa,doi,ida,ira],
     [[#,#,#,P14,P15,P16,#,P18],
      [#,P22,P23,P24,#,P26,P27,P28],
      [#,P32,#,P34,P35,P36,#,P38],
      [P41,P42,P43,#,P45,#,#,#],
      [#,#,#,P54,P55,P56,#,#]]],
     Puz = [Lst_Pals, Grelha],
     writeln("Puzzle:"), escreve_Grelha(Grelha),
     resolve(Puz),
     writeln("Solucao:"), escreve_Grelha(Grelha).
Puzzle:
# # # - - - # -
# - - - # - - -
# - # - - - # -
- - - # - # # #
# # # - - - # #
Solucao:
# # # d a i # d
# d e i # d o a
# o # a i a # s
d i z # r # # #
# # # d a o # #
Puz = [[aia, dai,  ....
```

O predicado `escreve_Grelha/1` está definido no ficheiro `codigo_comum.pl`.

4 Avaliação

A nota do projecto será baseada nos seguintes aspectos:

- Execução correcta (80% - 16 val.). Estes 16 valores serão distribuídos da seguinte forma:

obtem_letras_palavras	0.50
espaco_fila	1.00
espacos_fila	0.75
espacos_puzzle	0.75
espacos_com_posicoes_comuns	0.75
palavra_possivel_esp	0.75
palavras_possiveis_esp	0.75
palavras_possiveis	0.75
letras_comuns	0.75
atribui_comuns	0.75
retira_impossiveis	1.00
obtem_unicas	0.75
retira_unicas	1.00
simplifica	0.75
inicializa	0.50
escolhe_menos_alternativas	0.75
experimenta_pal	1.25
resolve_aux	1.50
resolve	1.00

- Qualidade do código, a qual inclui abstracção relativa aos predicados implementados, nomes escolhidos, paragrafação e qualidade dos comentários (20% - 4 val.).

5 Penalizações

- Caracteres acentuados, cedilhas e outros semelhantes: 3 val.
- Presença de *warnings*: 2 val.

6 Condições de realização e prazos

O projecto deve ser realizado individualmente.

O código do projecto deve ser entregue obrigatoriamente por via electrónica até às **23:59 do dia 6 de Maio** de 2020, através do sistema Mooshak. Depois desta hora, não serão aceites projectos sob pretexto algum.[2]

Deverá ser submetido um ficheiro .pl contendo o código do seu projecto. O ficheiro de código deve conter em comentário, na primeira linha, o número e o nome do aluno.

No seu ficheiro de código não devem ser utilizados caracteres acentuados ou qualquer caractere que não pertença à tabela ASCII, sob pena de falhar todos os testes automáticos. Isto inclui comentários e cadeias de caracteres.

[2]Note que o limite de 10 submissões simultâneas no sistema Mooshak implica que, caso haja um número elevado de tentativas de submissão sobre o prazo de entrega, alguns alunos poderão não conseguir submeter nessa altura e verem-se, por isso, impossibilitados de submeter o código dentro do prazo.

É prática comum a escrita de mensagens para o ecrã, quando se está a implementar e a testar o código. No entanto, **não se esqueçam de remover/comentar as mensagens escritas no ecrã na versão final** do código entregue. Se não o fizerem, correm o risco dos testes automáticos falharem, e irão ter uma má nota na execução.

A avaliação da execução do código do projecto será feita automaticamente através do sistema Mooshak, usando vários testes configurados no sistema. O tempo de execução de cada teste está limitado, bem como a memória utilizada. Só poderá efectuar uma nova submissão pelo menos 15 minutos depois da submissão anterior.[3] Só são permitidas 10 submissões em simultâneo no sistema, pelo que uma submissão poderá ser recusada se este limite for excedido. Nesse caso tente mais tarde.

Os testes considerados para efeitos de avaliação podem incluir ou não os exemplos disponibilizados, além de um conjunto de testes adicionais. O facto de um projecto completar com sucesso os exemplos fornecidos não implica, pois, que esse projecto esteja totalmente correcto, pois o conjunto de exemplos fornecido não é exaustivo. É da responsabilidade de cada aluno garantir que o código produzido está correcto.

Duas semanas antes do prazo da entrega, serão publicadas na página da cadeira as instruções necessárias para a submissão do código no Mooshak. Apenas a partir dessa altura será possível a submissão por via electrónica. Até ao prazo de entrega poderá efectuar o número de entregas que desejar, sendo utilizada para efeitos de avaliação a última entrega efectuada. Deverão portanto verificar cuidadosamente que a última entrega realizada corresponde à versão do projecto que pretendem que seja avaliada. Não serão abertas excepções.

Pode ou não haver uma discussão oral do projecto e/ou uma demonstração do funcionamento do programa (será decidido caso a caso).

7 Cópias

Projectos iguais, ou muito semelhantes, originarão a reprovação na disciplina e, eventualmente, o levantamento de um processo disciplinar. Os programas entregues serão testados em relação a soluções existentes na web. As analogias encontradas com os programas da web serão tratadas como cópias. O corpo docente da disciplina será o único juiz do que se considera ou não copiar num projecto.

8 Recomendações

- Recomenda-se o uso do SWI PROLOG, dado que este vai ser usado para a avaliação do projecto.
- Durante o desenvolvimento do programa é importante não se esquecer da Lei de Murphy:
  - Todos os problemas são mais difíceis do que parecem;

[3] Note que, se efectuar uma submissão no Mooshak a menos de 15 minutos do prazo de entrega, fica impossibilitado de efectuar qualquer outra submissão posterior.

- Tudo demora mais tempo do que nós pensamos;
- Se alguma coisa puder correr mal, ela vai correr mal, na pior das alturas possíveis.