DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Licenciatura em Engenharia Electrotécnica e de Computadores (LEEC)

# Algoritmos e Estruturas de Dados

## Enunciado do Projecto


# WordMutations

**Versão 3.0 (23/Setembro/2022)** – primeira versão para os alunos

2022/2023
1º Semestre

# Conteúdo

# 1 Introdução

O trabalho que se descreve neste enunciado possui duas componentes, que correspondem às duas fases de avaliação de projecto para a disciplina de Algoritmos e Estruturas de Dados. A descrição geral do projecto que se propõe diz respeito ao trabalho a desenvolver para a última fase de avaliação.

Para a primeira fase de avaliação o trabalho consiste apenas no desenvolvimento de algumas funcionalidades testáveis que podem ser usadas posteriormente para desenvolver a solução final.

A entrega do código fonte em ambas as fases é feita através de um sistema automático de submissão que verificará a sua correcção e testará a execução do programa em algumas instâncias do problema.

# 2 O problema – Caminhos entre palavras

Neste projecto pretende-se desenvolver um programa que seja capaz de produzir "caminhos" entre palavras do mesmo tamanho. Entende-se por um caminho entre duas palavras do mesmo tamanho, dadas como ponto de partida e de chegada, uma sequência de palavras de igual tamanho, em que cada palavra se obtém a partir da sua antecessora por substituição de um ou mais caracteres por outro(s). A resolução deste problema pressupõe a existência de um dicionário e todas as palavras do caminho têm de pertencer ao dicionário.

Por exemplo, seja a seguinte sequência de palavras, que estabelece um caminho de palavras entre a palavra **carro** e a palavra **pista**:

```
carro
corro
corto
porto
porta
posta
pista
```

Nesta sequência, todas as palavras são obtidas a partir da anterior por substituição de apenas um dos caracteres. Definindo por "mutação" a transformação de uma palavra noutra por substituição de algum(uns) dos seus caracteres, na sequência acima são realizadas seis mutações simples para transformar **carro** em **pista**.

Admitindo a possibilidade de se substituírem, no máximo, dois caracteres, seria possível produzir um outro caminho entre **carro** e **pista** com menos palavras: carro; corto; porta; pista. Ou seja, através de três mutações duplas.

Para que haja forma de distinguir entre os vários caminhos possíveis, quando exista mais do que um, é necessário introduzir um custo associado com as mutações. Neste projecto iremos assumir que o custo é quadrático no número de caracteres substituídos entre duas palavras consecutivas. Assim, o primeiro caminho possui um custo de 6 ($1^2$ por cada mutação simples) e o segundo possui um custo de 12 ($2^2$ por cada uma das mutações duplas). Por exemplo, é possível transformar a palavra **carro** na palavra **pista** por substituição directa de todos os caracteres simultaneamente, mas tal incorre num custo de 25 ($5^2$).

O que se pretende neste projecto é desenvolver uma aplicação em linguagem C que seja capaz de calcular o caminho de menor custo entre duas palavras, sujeito a um conjunto de restrições relativas ao número máximo de caracteres que pode mudar em cada um dos passos desse caminho.

Nas secções seguintes descreve-se o formato de utilização do programa a desenvolver, no que diz respeito aos ficheiros de entrada e saída; as regras de avaliação; e faz-se referência ao *Código de Conduta Académica*, que se espera ser zelosamente cumprido por todos os alunos que se submetam a esta avaliação.

Aconselha-se todos os alunos a lerem com a maior atenção todos os aspectos aqui descritos. Será obrigatória a adesão sem variações nem tonalidades a todas as especificações aqui descritas. A falha em cumprir alguma(s) especificação(ões) e/ou regra(s) constante(s) neste enunciado acarretará necessariamente alguma forma de desvalorização do trabalho apresentado.

Por esta razão, tão cedo quanto possível e para evitar contratempos tardios, deverão os alunos esclarecer com o corpo docente qualquer aspecto que esteja menos claro neste texto, ou qualquer dúvida que surja após uma leitura atenta e cuidada deste enunciado.

# 3 O programa "WordMutations"

O programa a desenvolver deverá ser capaz de ler pares de palavras e produzir soluções para cada par ou indicar não haver solução, quando esse seja o caso.

## 3.1 Execução do programa

Este semestre haverá duas fases de submissão do projecto. O que se descreve nas próximas secções diz respeito às especificações da versão final do projecto. Na secção 4 detalham-se as especificações relativas à primeira fase de submissões.

O programa de WordMutations deverá ser invocado na linha de comandos da seguinte forma:

```
aed$ ./wrdmttns <nome1>.dict <nome2>.pals
```

`wrdmttns` designa o nome do ficheiro executável contendo o programa WordMutations;

`<nome1>.dict` em que `<nome1>` é variável, identificando o ficheiro contendo o dicionário a utilizar. Restringe-se a aplicação a dicionários exclusivamente compostos com palavras usando caracteres latinos, não acentuadas e não hifenadas.

`<nome2>.pals` em que `<nome2>` é variável, identificando o ficheiro contendo um ou mais problemas que se pretende resolver.

Para cada problema o programa deverá fazer uma de duas coisas:

- produzir a solução de menor custo;
- indicar que não existe solução.

## 3.2 Formato de entrada

O ficheiro de extensão `.dict` apenas contém palavras, podendo ser um dicionário parcial de algum idioma ou uma mistura de vários idiomas, sem palavras acentuadas nem hifenadas, não necessariamente ordenado alfabeticamente nem por tamanho. Por exemplo,

```
aba
                                        aia

        bata
colher          sopa
asa
exito


bola                    ala
sequencia
...
```

De acordo com a ilustração acima, também não é obrigatório que o ficheiro contenha apenas uma palavra por linha, podendo haver espaços, tabs e/ou mudanças de linha em qualquer sítio do ficheiro. A aplicação a desenvolver pelos alunos deverá ser suficientemente geral e robusta no seu formato de leitura dos dicionários.

O ficheiro de extensão `.pals`, poderá conter mais do que um problema. Cada problema é descrito numa linha e cada linha contém duas palavras e um número inteiro. As palavras são o ponto de partida e chegada do caminho e o inteiro indica qual o número máximo de caracteres que se podem substituir numa mutação entre cada duas palavras consecutivas. Ambas as palavras dadas para cada problema têm que pertencer ao dicionário dado. Caso uma ou ambas não pertençam o problema está mal definido. A resposta para problemas mal definidos é a mesma que para problemas sem solução.

Por exemplo, um ficheiro de problemas poderá ser

```
carro pista 1
esforcados recompensa 1
pista carro 2
castigo virtude 3
```

Neste exemplo, independentemente de outros pares de palavras que aquele ficheiro contém, o primeiro problema consiste em determinar o caminho de menor custo entre a palavra **carro** e a palavra **pista** apenas usando mutações simples. O último problema do ficheiro consiste em determinar o caminho de menor custo entre a palavra **castigo** e a palavra **virtude** em que cada mutação pode ser de um, dois ou três caracteres, não sendo admissíveis mutações em que mais do que três caracteres sejam substituídos.

Os ficheiros com um ou mais problemas poderão ter qualquer nome, mas têm obrigatoriamente a extensão `.pals`. Os ficheiros contendo dicionários poderão também ter qualquer nome, mas têm obrigatoriamente a extensão `.dict`.

Assume-se que todos os ficheiros de extensão `.pals` estão correctos e no formato especificado anteriormente. Ou seja, serão sempre pedidos caminhos entre palavras do

mesmo tamanho e o número de carateres passíveis de substituição em cada mutação será sempre maior do que zero, podendo ser superior ao tamanho das palavras. Por essa razão, o programa não necessita fazer qualquer verificação da sua correcção. Apenas necessita de garantir que as extensões estão correctas e que esses ficheiros passados como argumento existem de facto.

## 3.3 Formato de saída

O resultado da execução do programa WordMutations consiste em listar as palavras que constituem o caminho de menor custo entre pares de palavras para cada um dos problemas e indicar o custo desse caminho.

Para qualquer problema, a primeira linha da solução deverá sempre possuir a palavra de partida seguida do custo. As linhas seguintes deverão conter cada uma das palavras do caminho, que deverá sempre terminar com a palavra destino. Se não houver caminho, a primeira linha contém a palavra de partida seguida de -1. Na segunda linha deverá estar a palavra destino.

Se o ficheiro de extensão `.pals` possuir mais do que um problema, o ficheiro de saída deverá conter uma solução para cada um dos problemas indicados e pela mesma ordem em que surgem no ficheiro de entrada. Para facilitar a interpretação visual das várias soluções num mesmo ficheiro de saída, é obrigatório que entre cada duas soluções exista uma só linha vazia de separação.

A(s) solução(ões) deve(m) ser colocada(s) num único ficheiro de saída, cujo nome deve ser o mesmo do ficheiro de problemas mas **com extensão** `.paths`. Este ficheiro deve ser criado e aberto pelo programa. Por exemplo, se o ficheiro com problemas se chama `teste231.pals`, o ficheiro de saída deve-se chamar `teste231.paths`. Note-se que, em situações em que haja erro na passagem de argumentos ao programa, não faz qualquer sentido criar um ficheiro de saída.

Considere, por exemplo, o seguinte ficheiro de entrada:

```
carro pista 3
absolutamente transpirantes 1
```

Este ficheiro contém dois problemas. Assuma que a solução do primeiro problema é a seguinte, podendo haver outras de igual custo mas usando diferentes palavras no caminho:

```
carro 6
corro
corto
porto
porta
posta
pista
```

Assumindo que o segundo problema não possui solução, a explicitação de tal facto seria feita da seguinte forma:

```
absolutamente -1
transpirantes
```

O ficheiro de saída seria a concatenação das duas soluções acima descritas, com uma linha em branco de intervalo entre cada uma das soluções.

Se o programa for invocado com ficheiros inexistentes, que não possuam a extensão `.dict` ou a extensão `.pals`, sem qualquer argumento, com argumentos a mais ou ordem dos ficheiros trocada, deverá sair silenciosamente. Ou seja, sem escrever qualquer mensagem de erro, nem criar qualquer ficheiro de saída.

Em qualquer dos casos, seja o programa bem ou mal invocado a terminação do código deverá sempre retornar o valor zero. Qualquer outro valor de retorno será interpretado como "Erro de Execução"pelo software de avaliação das implementações.

# 4 Primeira fase de submissões

Nesta secção explicitam-se os objectivos, especificações e funcionalidades que deverão estar operacionais na data da primeira fase de submissões. Todas as funcionalidades desta fase de submissão dizem exclusivamente respeito ao processamento de dicionários e extração de informação a partir dos mesmos.

O formato de invocação do programa será o mesmo que o definido anteriormente. Ou seja, o executável tem o mesmo nome e deverão ser passados dois argumentos: o nome de um ficheiro, de extensão `.dict`, contendo um dicionário e o nome de um ficheiro, de extensão `.pals`, contendo um ou mais problemas. Este segundo ficheiro tem exactamente o mesmo formato que foi anteriormente definido: por cada linha tem duas palavras de igual tamanho seguidas de um inteiro.

O inteiro define a funcionalidade que deverá ser executada para o par de palavras dado nessa linha, que nesta primeira fase tem o seguinte significado.

1. Indicar quantas palavras existem de igual tamanho ao das duas dadas ,incluindo essas duas;

2. Para cada palavra indicar qual a sua posição numa tabela contendo todas as palavras do mesmo tamanho que esteja ascendentemente ordenada alfabeticamente (as posições nessa hipotética tabela deverão ser contabilizadas a partir de 0).

## 4.1 Formato de saída da primeira fase de submissões

O ficheiro de saída da primeira fase, tem o mesmo nome que o ficheiro de problemas, mas deverá ter extensão `.stats` e deverá incluir todos os resultados associados com cada um dos problemas presentes no ficheiro de entrada. Para pares de palavras seguidas do inteiro 1, o ficheiro de saída deverá conter a primeira das duas palavras seguida do inteiro que indica quantas palavras de igual tamanho estão presentes no dicionário. Para pares de palavras seguidas do inteiro 2, o ficheiro de saída deverá conter cada uma das palavras seguida do número pedido. Abaixo apresentam-se alguns exemplos possíveis (à esquerda a linha do ficheiro de entrada e à direita a solução correspondente).

O ficheiro de entrada poderá conter mais do que um problema para resolver e cada um desses problemas poderá ter qualquer uma das duas opções para o inteiro.

```
pista porta 1          pista 3425


pista porta 2          pista 2789
                       porta 2896
```

Se, por hipótese, o ficheiro de entrada fosse a concatenação dos dois exemplos acima, o ficheiro de saída seria a concatenação das duas soluções apresentadas. Aqui também é obrtigatória a inclusão de uma linha em branco como separador das diferentes soluções.

Na hipótese de uma das duas palavras dadas não pertencer ao dicionário e/ou o inteiro que se lhes segue for diferente de 1 ou de 2, a solução passa simplesmente por repetir no ficheiro de saída a linha associada com esse problema mal definido.

# 5 Avaliação do Projecto

O projecto está dimensionado para ser feito por grupos de dois alunos, não se aceitando grupos de dimensão superior nem inferior. Para os alunos que frequentam o laboratório, o grupo de projecto não tem de ser o mesmo do laboratório, mas é aconselhável que assim seja.

Do ponto de vista do planeamento, os alunos deverão ter em consideração que o tempo de execução e a memória usada serão tidos em conta na avaliação do projecto submetido. Por essas razões, a representação dos dados necessários à resolução dos problemas deverá ser criteriosamente escolhida tendo em conta o espaço necessário, mas também a sua adequação às operações necessárias sobre eles.

Serão admissíveis para avaliação versões do programa que não possuam todas as funcionalidades, seja no que à primeira fase de submissões diz respeito, como para a fase final. Naturalmente que um menor número de funcionalidades operacionais acarretará penalização na avaliação final.

Quando os grupos de projecto estiverem constituídos, os alunos devem obrigatoriamente inscrever-se no sistema Fénix, no agrupamento Projecto, que será criado oportunamente.

A avaliação do projecto decorre em três ou quatro instantes distintos. O primeiro instante coincide com a primeira submissão electrónica, onde os projectos serão avaliados automaticamente com base na sua capacidade de cumprir as especificações e funcionalidades definidas na Secção 4. Para esta fase existe apenas uma data limite de submissão (veja a Tabela 1) e não há qualquer entrega de relatório. O segundo instante corresponde à submissão electrónica do código na sua versão final e à entrega de uma ficha de auto-avaliação em moldes a definir. Esta entrega ratifica e lacra a submissão electrónica anteriormente realizada.

Num terceiro instante há uma proposta enviada pelo corpo docente que pode conter a indicação de convocatória para a discussão e defesa do trabalho ou uma proposta de nota para a componente de projecto. No caso de ser proposta uma nota, se os alunos a aceitarem a discussão não é necessária e a nota torna-se final. Se, pelo contrário, os alunos decidirem recorrer da nota proposta, será marcada uma discussão de recurso em data posterior. O quarto instante acontece apenas caso haja marcação de uma discussão, seja por convocatória do docente, seja por solicitação dos alunos. Nestas circunstâncias, a discussão é obrigatoriamente feita a todo o grupo, sem prejuízo de as classificações dos elementos do grupo poderem vir a ser distintas. A falta de um ou ambos os alunos à oral

Tabela 1: Datas importantes do Projecto

| Data | Documentos a Entregar |
|---|---|
| 23 de Setembro de 2022 | Enunciado do projecto disponibilizado na página da disciplina. |
| até 30 de Setembro de 2022 (18h00) | Inscrição dos grupos no sistema Fénix. |
| 12 de Outubro de 2022, (18h00) | Conclusão da primeira fase de submissões. |
| 04 de Novembro de 2022 (18h00) | Fim da submissão electrónica da fase final. |
| 05 de Novembro de 2022 (18h00) | Entrega da ficha de auto-avaliaçãoe (via Fénix). |
| até algures de Fevereiro de 2022 | Eventual discussão do trabalho (data combinada com cada grupo). |

convocada pelos docentes acarreta reprovação automática de quem faltar e o grupo terá de justificar as razões dessa(s) falta(s).

As datas de entrega referentes aos vários passos da avaliação do projecto estão indicadas na Tabela 1.

As submissões electrónicas estarão disponíveis em datas e condições a indicar posteriormente na página da disciplina e serão aceites trabalhos entregues até aos instantes finais indicados.

Os alunos não devem esperar qualquer **extensão nos prazos de entrega**, pelo que devem organizar o seu tempo de forma a estarem em condições de entregar a versão final dentro dos prazos indicados.

Note-se que, na versão final, o projecto só é considerado entregue aquando da entrega da ficha de auto-avaliação. As submissões electrónicas do código não são suficientes para concretizar a entrega. A ausência de ficha de auto-avaliação é a forma que os alunos têm de informar os docentes que não pretendem que a sua implementação seja avaliada.

## 5.1 Funcionamento

A verificação do funcionamento do código a desenvolver no âmbito do projecto será exclusivamente efectuada nas máquinas do laboratório da disciplina, embora o desenvolvimento possa ser efectuado em qualquer plataforma ou sistema que os alunos escolham. Esta regra será estritamente seguida, não se aceitando quaisquer excepções. Por esta razão, é essencial que os alunos, independentemente do local e ambiente em que desenvolvam os seus trabalhos, os verifiquem no laboratório antes de os submeterem, de forma a evitar problemas de última hora. Uma vez que os laboratórios estão abertos e disponíveis para os alunos em largos períodos fora do horário das aulas, este facto não deverá causar qualquer tipo de problemas.

## 5.2 Código

**Não deve ser entregue código em papel**. Os alunos devem entregar por via electrónica o código do programa (ficheiros `.h` e `.c`) e uma `Makefile` para gerar o executável. Todos

os ficheiros (`*.c`, `*.h` e `Makefile`) devem estar localizados na directoria raiz.

O código deve ser estruturado de forma lógica em vários ficheiros (`*.c` e `*.h`). As funções devem ter um cabeçalho curto mas explicativo e o código deve estar correctamente indentado e com comentários que facilitem a sua legibilidade.

## 5.3 Critérios de Avaliação

Os projectos submetidos serão avaliados de acordo com a seguinte grelha:

- Testes passados na primeira submissão electrónica – 10% a 20%
- Número de submissões na primeira fase – ver descrição abaixo
- Testes passados na última submissão electrónica – 65% a 55%
- Número de submissões na fase final – ver descrição abaixo
- Estruturação do código e comentários – 5%
- Gestão de memória e tipos abstractos – 5%
- Ficha de auto-avaliação – 15%

Tanto na primeira como na submissão electrónica final, cada projeto será testado com vários ficheiros de problemas de diferentes graus de complexidade, onde se avaliará a capacidade de produzir soluções correctas dentro de limites de tempo e memória. Para o limite de tempo, cada um dos testes terá de ser resolvido em menos de $x$ segundos[1]. Para o limite de memória, cada um dos testes não poderá exceder 100MB como pico de memória usada[2]. Cada teste resolvido dentro dos orçamentos temporal e de memória que produza soluções correctas recebe uma pontuação de acordo com o grau de dificuldade desse mesmo teste. Na primeira fase de submissão, embora se mantenha o mesmo limite de memória, o tempo limite para resolver qualquer teste será muito mais baixo do que os 60 segundos (por exemplo, na faixa dos 10 a 15 segundos), que ficam reservados apenas para os testes da fase final.

Um teste considera-se errado se, pelo menos, um dos problemas do ficheiro de entrada correspondente for incorrectamente resolvido.

Se o corpo docente entender necessário, face à complexidade dos problemas a resolver, poderão os limites de tempo e/ou memória ser alterados.

Caso o desempenho de alguma submissão electrónica não seja suficientemente conclusivo, poderá ser sujeita a testes adicionais fora do contexto da submissão electrónica. O desempenho nesses testes adicionais poderá contribuir para subir ou baixar a pontuação obtida na submissão electrónica.

Com o objectivo de se contribuir para que os alunos façam melhor uso do seu tempo no desenvolvimento das suas implementações, introduz-se um factor correctivo associado ao número de submissões de cada grupo. O site de submissões não pode nem deve ser encarado como um mecanismo de apoio ao desenvolvimento do projecto, na medida em que o feedback que fornece é muito limitado. Por outro lado, é entendimento da equipa

[1] O limite de tempo para os testes da primeira fase será inferior, $y < x$ a definir. Os valores típicos na primeira fase são 10 a 15 segundos e os típicos na fase final são 60 segundos, mas podem ser ajustados face à complexidade estrutural dos problemas a resolver.

[2] Não existe diferença nos limites de memória nas duas fases.

docente que se deve valorizar a capacidade dos grupos em serem capazes de validar as suas implementações fora do contexto do site de submissões. Por isso, dois projectos que resolvam todos os testes no site de submissões não valem o mesmo se um deles atingir essa marca em, por exemplo, 5 submissões e o outro atingir a mesma marca em 18.

Na Figura 1 apresenta-se um gráfico com o multiplicador associado ao número de submissões. Cada grupo dispõe de 10 submissões livres em cada uma das duas fases. As submissões livres servem para pequenos percalços com erros de compilação ou erros na composição dos ficheiros a submeter. Se um grupo fizer mais que 10 submissões em alguma das fases, a pontuação que recebe nessa fase é penalizada em 2% por cada submissão adicional até à vigésima submissão. A partir da vigésima submissão, cada submissão adicional é penalizada em 3%. Por exemplo, suponha-se um grupo que atinge a pontuação ilíquida de 180 pontos na primeira fase: recebe esses 180 pontos por inteiro se fizer 10 ou menos submissões; recebe 162 pontos se tiver feito 15 submissões; e recebe 117 pontos se tiver feito 25 submissões.

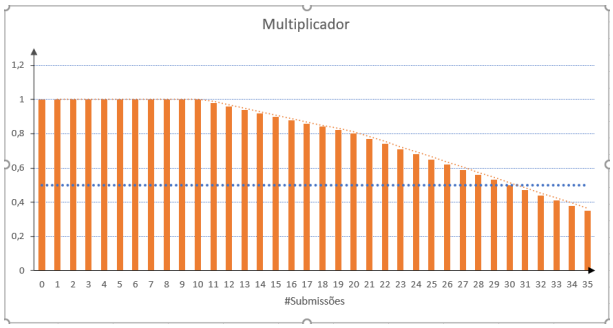


Figura 1: Multiplicador para cada uma das duas fases de submissão.

Qualquer que venha a ser o número de submissões feitas, avaliar-se-á apenas a que tiver obtido melhor pontuação, independentemente se ser a útima, primeira ou intermédia. Na eventualidade de existir empate, compete aos alunos indicar na ficha de auto-avaliação quais as suas duas submissões (uma por fase) que pretendem ver contabilizadas para produção da pontuação, sendo que a submissão da fase final que indicarem será a única a ser avaliada pelos docentes.

No que à avaliação da ficha de auto-avaliação diz respeito, os elementos de avaliação incluem: rigor do seu preenchimento; apreciação da qualidade da abordagem geral ao problema e respectiva implementação, face às alternativas disponíveis; e clareza e suficiência do texto, na sua capacidade de descrever e justificar com precisão algumas das opções.

Pela análise da grelha de avaliação aqui descrita, deverá ficar claro que a ênfase da avaliação se coloca na capacidade de um programa resolver correcta e eficazmente os problemas a que for submetido e que os alunos sejam também eficientes no seu desenvolvimento. Ou seja, o código de uma submissão até pode ser muito bonito e bem estruturado e o grupo até pode ter dispendido muitas horas no seu desenvolvimento. No entanto, se esse código não resolver um número substancial de testes na submissão electrónica ou se necessitar de um número elevado de submissões para demonstrar a sua correcção, dificilmente terá uma nota positiva.

# 6 Código de Honestidade Académica

Espera-se que os alunos conheçam e respeitem o Código de Honestidade Académica que rege esta disciplina e que pode ser consultado na página da cadeira. O projecto é para ser planeado e executado por grupos de dois alunos e é nessa base que será avaliado. Quaisquer associações de grupos ou outras, que eventualmente venham a ocorrer, serão obviamente interpretadas como violação do Código de Honestidade Académica e terão como consequência a anulação do projecto aos elementos envolvidos.

Lembramos igualmente que a verificação de potenciais violações a este código é feita de forma automática com recurso a sofisticados métodos de comparação de código, que envolvem não apenas a comparação directa do código mas também da estrutura do mesmo. Esta verificação é feita com recurso ao software disponibilizado em

`http://moss.stanford.edu/` .