DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Mestrado Integrado em Engenharia Electrotécnica e de Computadores (MEEC)

Algoritmos e Estruturas de Dados

Enunciado do Projecto

Word Morph

Versão 4.0 (17/Outubro/2016) – primeira versão para os alunos

2016/2017
1º Semestre

Conteúdo

1 Introdução **2**

2 O problema – Caminhos entre palavras **2**

3 O programa "Word Morph" **3**
3.1 Execução do programa 3
3.2 Formato de entrada 4
3.3 Formato de saída 5

4 Primeira fase de submissões **6**
4.1 Formato de saída da primeira fase de submissões 6

5 Avaliação do Projecto **6**
5.1 Funcionamento 8
5.2 Código 8
5.3 Relatório 9
5.4 Critérios de Avaliação 10

6 Código de Honestidade Académica **10**

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 trabalho que se propõe diz respeito ao trabalho a desenvolver para a última fase de avaliação. O trabalho a realizar para a primeira fase de avaliação assenta no mesmo problema, mas consiste no desenvolvimento de algumas funcionalidades que, apesar de não determinarem em absoluto a solução final, podem ser usadas posteriormente para ajudar na sua resolução. Assim, os alunos deverão encarar a primeira fase de avaliação como uma primeira etapa no trabalho de concepção e desenvolvimento da sua 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. 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, entre duas palavras dadas, 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.

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 seja 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 "Word Morph"

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 WORD MORPH deverá ser invocado na linha de comandos da seguinte forma:

```
aed$ ./wordmorph <nome1>.dic <nome2>.pal
```

`wordmorph` designa o nome do ficheiro executável contendo o programa WORD MORPH;

`<nome1>.dic` 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>.pal` 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 `.dic` 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. Por exemplo,

```
aba
aia
bata colher sopa
asa
exito
bola aia
sequencia
...
```

De acordo com a ilustração acima, também não é obrigatório que o dicionário contenha apenas uma palavra por linha. A aplicação a desenvolver pelos alunos deverá ser suficientemente geral no seu formato de leitura dos dicionários.

O ficheiro de extensão `.pal`, poderá conter mais do que um problema. Para cada problema contém duas palavras e um 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 pertencem ao dicionário dado.

Por exemplo, um ficheiro de problemas poderá ser

```
carro pista 1
...
castigo virtude 3
```

Neste exemplo, independentemente de outros pares de palavras que aquele ficheiro contenha, 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 `.pal`. Os ficheiros contendo dicionários poderão também ter qualquer nome, mas têm obrigatoriamente a extensão `.dic`.

Assume-se que todos os ficheiros de extensão `.pal` 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 WORD MORPH 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 `.pal` 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 das várias soluções num mesmo ficheiro de saída, sugere-se que entre cada duas soluções exista uma 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** `.path`. Este ficheiro deve ser criado e aberto pelo programa. Por exemplo, se o ficheiro com problemas se chama `teste231.pal`, o ficheiro de saída deve-se chamar `teste231.path`. 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. A solução do primeiro problema é (poderá eventualmente existir uma solução de menor custo) a seguinte:

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

O segundo problema não possui solução, pelo que no ficheiro de saída se deverá escrever:

```
absolutamente -1
transpirantes
```

O ficheiro de saída seria a concatenação das duas soluções acima descritas.

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

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 `.dic`, contendo um dicionário e o nome de um ficheiro, de extensão `.pal`, 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.

- 1. Indicar quantas palavras existem de igual tamanho ao das duas dadas;
- 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 `.stat` e deverá incluir todos os resultados associados com cada um dos problemas presentes no ficheiro de entrada. Para pares de palavras seguidas do inteiro um, 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 dois, 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 se sugere a inclusão de uma linha em branco como separador das diferentes soluções.

5 Avaliação do Projecto

O projecto está dimensionado para ser feito por grupos de dois alunos, não se aceitando grupos de dimensão superior. 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 do projecto, 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, no que à primeira fase de submissões diz respeito. Naturalmente que um menor número de funcionalidades operacionais acarretará penalização na avaliação final. Relativamente à última fase de submissões, apenas programas que calculem correctamente os caminhos de menor custo entre palavras são admissíveis para aprovação.

Quando os grupos de projecto estiverem constituídos, os alunos devem obrigatoriamente inscrever-se no sistema Fenix, no grupo de Projecto correspondente, 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 do relatório em mãos aos docentes, entrega essa que ratifica e lacra a submissão electrónica anteriormente realizada. Na submissão final é possível submeter o projecto e entregar o relatório durante três dias consecutivos. No entanto, entregas depois da primeira data sofrerão uma penalização (veja a Tabela 1).

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. Caso os alunos aceitem a nota proposta pelo docente avaliador, 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.

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. Ao contrário de anos anteriores, neste semestre **não haverá qualquer extensão nos prazos de entrega**, pelo que os alunos devem organizar o seu tempo de forma a estarem em condições de entregar a versão final na primeira data indicada. As restantes datas devem ser encaradas como soluções de recurso, para a eventualidade de alguma coisa correr menos bem durante o processo de submissão. O relatório final deverá ser entregue em mão aos docentes no laboratório no dia indicado na Tabela 1.

Note-se que, na versão final, o projecto só é considerado entregue aquando da entrega do relatório em papel. As submissões electrónicas do código não são suficientes para concretizar a entrega. Um grupo que faça a submissão electrónica do código e a entrega do relatório em papel, por exemplo, na 1ª data de entrega pode fazer submissões nas datas seguintes, mas se fizer a entrega de um novo relatório em papel, será este, e as respectivas submissões, o considerado para avaliação, com a penalização indicada. De

Tabela 1: Datas importantes do Projecto

Data	Documentos a Entregar
17 de Outubro de 2016, 2ª feira	Enunciado do projecto disponibilizado na página da disciplina.
até 11 de Novembro de 2015 (24h)	Inscrição dos grupos no sistema Fenix.
18 de Novembro de 2016, 6ª feira	Primeira submissão.
14 de Dezembro de 2016, 4ª feira 12h 15h	**1ª Data de entrega do projecto:** Submissão electrónica do projecto. Entrega do relatório do projecto em papel.
15 de Dezembro de 2016, 5ª feira 12h 15h	**2ª Data de entrega do projecto: penalização de um (1) valor** Submissão electrónica do projecto. Entrega do relatório do projecto em papel.
16 de Dezembro de 2016, 6ª feira 12h 15h	**3ª Data de entrega do projecto: penalização de dois (2) valores** Submissão electrónica do projecto. Entrega do relatório do projecto em papel.
	Submissões posteriores a 16 de Dezembro têm penalização de 20 valores.
a partir de 16 de Dezembro de 2016	Eventual discussão do trabalho (data combinada com cada grupo).

modo semelhante, aos grupos que façam a sua última submissão electrónica na primeira data, mas entreguem o relatório numa das outras duas datas posteriores, será contada como data de submissão aquela em que o relatório for apresentado.

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 Relatório

Os relatórios devem ser entregues na altura indicada na Tabela 1. O relatório do projecto deverá contemplar os aspectos seguidamente indicados. A apresentação do mesmo deverá iniciar-se por aspectos de ordem geral, seguindo-se descrições mais detalhadas dos módulos e estruturas de dados utilizados. Sugere-se a seguinte estrutura para o relatório:

- Uma capa com os dados dos membros do grupo, incluindo nome, número e e-mail. Esta capa deverá seguir o formato indicado na página da disciplina (oportunamente será disponibilizado);
- Uma página com o índice das secções em que o relatório se divide;
- Uma descrição do problema que foi resolvido, com indicação clara das especificações do mesmo tal como foram entendidas;
- Um texto simples que indique como os alunos abordaram e resolveram o problema;
- Uma descrição completa da arquitectura do programa, incluindo fluxogramas detalhados e um texto claro, mas sucinto, indicando a divisão lógica e funcional dos módulos desenvolvidos para a resolução do problema, explicitando os respectivos objectivos, as funções utilizadas e as estruturas de dados de suporte;
- Uma descrição detalhada dos tipos de dados utilizadas e justificação dos mesmos (tabelas, listas, filas, pilhas, árvores, grafos, acervos, etc.);
- Descrição dos algoritmos usados (por exemplo, na manipulação dos tipos de dados);
- Uma descrição dos subsistemas funcionais que existam e, para cada um
  - a descrição dos objectivos do subsistema (até 5 linhas);
  - o nome do módulo onde estão definidos os tipos de dados abstractos (ficheiros `.h`) a utilizar no subsistema;
  - o nome do módulo `C` onde estão as funções do respectivo subsistema;
  - listagem das funções implementadas no subsistema, indicando para cada uma a respectiva assinatura e os objectivos da função (descrição sumária, sem código);
- Uma análise, formal e/ou empírica, dos requisitos computacionais do programa desenvolvido, tanto em termos da memória que utiliza como da complexidade computacional, com particular ênfase no custo das operações de processamento sobre os tipos de dados usados e/ou criados;
- Uma análise crítica do funcionamento do programa e a avaliação do desempenho do projecto implementado;
- Pelo menos, um pequeno exemplo completo e detalhado de aplicação, com descrição da utilização das estruturas de dados em cada passo e de como são tomadas as decisões.

5.4 Critérios de Avaliação

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

- Testes passados na primeira submissão electrónica – 10% a 15%
- Testes passados na última submissão electrónica – 60% a 65%
- Estruturação do código e comentários – 5%
- Gestão de memória e tipos abstractos – 5%
- Relatório escrito – 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 60 segundos. Para o limite de memória, cada um dos testes não poderá exceder 100MB como pico de memória usada. Cada teste resolvido dentro dos orçamentos temporal e de memória que produza soluções correctas recebe um ponto.

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.

No que à avaliação do relatório diz respeito, os elementos de avaliação incluem: apreciação da abordagem geral ao problema e respectiva implementação; análise de complexidade temporal e de memória; exemplo de aplicação; clareza e suficiência do texto, na sua capacidade de descrever e justificar com precisão o que está feito; e qualidade do texto escrito e estruturação do relatório.

Pela análise da grelha de avaliação aqui descrita, deverá ficar claro que a ênfase da avaliação se coloca na capacidade de um programa resolver correctamente os problemas a que for submetido. 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 dificilmente terá uma nota substancial.

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/` .