Proposta de Trabalho

Project Proposal

Manipulação de vetores e estruturas dinâmicas usando a linguagem de programação C: caso de estudo aplicado a *instant messaging*

Manipulation of arrays and dynamic data structures using the C programming language: instant messaging case study

Algoritmos e Estruturas Dados I

Algorithms and Data Structures I

José Manuel Torres & André Ribeiro

{jtorres, arpinto}@ufp.edu.pt

Linguagens de Programação I

Programming Languages I

Rui Silva Moreira & Christophe Soares

{rmoreira, csoares}@ufp.edu.pt

Outubro de 2017

October 2017

Universidade Fernando Pessoa

Faculdade de Ciência e Tecnologia

Faculty of Science and Technology

1 Descrição do Problema

O *instant messaging* (IM) é uma forma popular de conversação online, em tempo real, usado na Internet. Entre os serviços mais populares de IM encontram-se o Facebook Messenger e o WhatsApp. A troca de mensagens pode ser feita na forma um-para-um ou em grupo.

No contexto deste trabalho não se pretende desenvolver um sistema de IM, apenas o módulo que permitirá armazenar e gerir conversas entre os utilizadores do serviço. Neste aspecto deve considerar-se o seguinte:

1. Num sistema de gestão IM considera-se que um conjunto de conversas ou documentos designa-se por *corpus* e pode ser associado a um utilizador;
2. As conversas/documentos são constituídas por sequência temporais (com *timestamps*) de mensagens trocadas entre os utilizadores;
3. Para este trabalho deve considerar-se que as mensagens são apenas de natureza textual, i.e., não suportam multimédia.
    a. Cada mensagem de um utilizador tem associada uma importância numérica (1..100) com o valor 50 por *default*;
    b. Cada mensagem de texto pode conter strings de comprimento variável, correspondendo a um ou vários parágrafos.

Pretende-se portanto implementar uma aplicação de software de suporte ao repositório de um serviço de IM, ou seja, que permita o armazenamento e a gestão de conversas e mensagens do serviço de IM usando arrays dinâmicos e estruturas dinâmicas ligadas. A aplicação deverá implementar funcionalidades de análise e pesquisa em texto, ordenação e comparação de strings, conforme será detalhado nos requisitos. Deste modo, deverão ser especificados e desenvolvidos vários modelos de dados e algoritmos associados para a sua manipulação.

Existirão **duas fases** de entrega do trabalho: uma **primeira fase** a entregar até meio do semestre (cf. assignment a definido no *elearning*) e uma **segunda fase** a entregar no final do semestre (cf. assignment definido no *elearning*).

2 Requisitos Funcionais

De seguida resumem-se os requisitos a cumprir neste projeto. Estes requisitos deverão seguir duas fases de implementação (com exceção dos requisitos identificados):

2.1 Fase I

Na primeira fase devem utilizar **arrays/vetores e matrizes dinâmicas** para organizar os dados referentes às mensagens, nomeadamente os parágrafos no seu formato original (i.e., frases enviadas nas mensagens) e em alternativa as palavras individualmente bem como os índices/posição das palavras nos parágrafos do formato original. Deverão ainda ser desenvolvidos algoritmos de processamento, gestão e pesquisa sobre as mensagens. As implementações deverão respeitar os seguintes requisitos funcionais:

R1. Permitir armazenar mensagens no seu formato original, ou seja, representar os parágrafos integrais em cada linha da matriz; Os parágrafos são terminados por '\n' e podem conter várias frases; cada frase é terminada por um caracter especial (cf. '.', '!', '?', etc.);

R2. Permitir armazenar individualmente as palavras das mensagens (sem repetições) e representar também (noutra estrutura) os índices/posições das respectivas ocorrências no texto das mensagens originais; por palavras entende-se qualquer conjunto de 1 ou mais letras ou números; também devem ser considerados e armazenados individualmente os caracteres de pontuação;

R3. Permitir transformar a representação da mensagem no formato original para o formato das palavras individuais e respectivas ocorrências, e vice-versa;

R4. Permitir representar e extrair a *bag-of-words*[1] de várias mensagens como descritores dessas mensagens de modo a poder ser utilizado em algoritmos de pesquisa, comparação e classificação;

R5. Com base no *bag-of-words*, pretende-se que seja possível calcular o *term frequency–inverse document frequency* (TF-IDF) dum determinado termo numa mensagem para, deste modo, calcular a sua relevância;

[1] https://en.wikipedia.org/wiki/Bag-of-words

R6. Pretende-se identificar métricas/estatísticas de mensagens que são compostas por parágrafos, como por exemplo, identificar o número de parágrafos, a frequência de reutilização de uma determinada palavra, distância textual de duas palavras entre duas posições na mensagem (i.e., número de palavras entre a primeira ocorrência das duas palavras), as palavras com possíveis raízes comuns (e.g., medicina, telemedicina, médico, etc.), os padrões de expressões repetidas (i.e. repetições de sub-strings no texto), as palavras que são compostas por um conjunto de predefinido caracteres[2] (e.g., diagramas, trigramas, etc.) , etc.;

R7. Pretende-se ainda calcular algumas estatísticas das mensagens, como por exemplo, a frequência relativa das letras do alfabeto; a frequência relativa de determinados digramas e trigramas; a frequência relativa das N primeiras e últimas letras das palavras; o comprimento médio das palavras; a frequência relativa das palavras com tamanho <=M.

2.2 Fase II

Na segunda fase devem utilizar **apontadores** e **estruturas dinâmicas** para agregar os dados anteriores e associá-los às conversas/documentos, corpus e respectivos utilizadores. Deverão ser desenvolvidos ainda algoritmos de processamento, gestão e pesquisa de toda a informação (cf. mensagens, conversas/documentos, corpus, utilizadores, etc.). As implementações deverão responder ou respeitar os seguintes requisitos funcionais:

R8. Permitir a representação de conversas/documentos; cada conversa deverá ser representada por uma estrutura (*struct*) com apontadores para referenciar as mensagens trocadas em cada conversa;

R9. Permitir a representação de várias conversas/documentos, num *corpus* de documentos; devem usar-se também estruturas (*struct*) e apontadores para organizar a informação necessária;

R10. Permitir a representação da associação dos *corpus* aos utilizadores do serviço de IM;

[2] https://www.mat.uc.pt/~pedro/lectivos/CodigosCriptografia1011/interTIC07pqap.pdf

R11. Permitir gerir e armazenar conversas de IM, i.e., inserir e/ou remover conversas, mensagens em conversas, associar timestamps com mensagens, associar utilizadores com conversas e identificar o utilizador que insere cada mensagem;

R12. Permitir extrair a *bag-of-words* de um conjunto de conversas (*corpus*), ou seja, do conjunto de mensagens de várias conversas. A *bag-of-words* poderá ser utilizada como descritor das conversas de modo a poder ser utilizado em algoritmos de pesquisa, comparação e classificação de documentos;

R13. Com base no *bag-of-words*, pretende-se que seja possível calcular o *term frequency–inverse document frequency* (TF-IDF) dum termo num *corpus*, para, deste modo, calcular a relevância de um termo que pertence ao *corpus* de documentos[3];

R14. Pretende-se fazer pesquisa nas conversas, por conteúdo textual usando uma *bag-of-words*, TF-IDF; como, por exemplo, identificar nessa *bag-of-words* quais são as palavras que têm partes comuns (i.e., uma sequência de caracteres idênticos). O tamanho desta sequência poderá ser parametrizado na pesquisa (e.g., a sequência "mar" de tamanho 3, poderá identificar as seguintes palavras "pomares", "marinha", "tomar"); neste requisito podem adaptar, por exemplo, o algoritmo Knuth-Morris-Pratt[4] (KMP);

R15. Pretende-se identificar métricas/estatísticas de conversas que são compostas por mensagens, como por exemplo, identificar o número de parágrafos, a frequência de reutilização de uma determinada palavra, distância textual de duas palavras entre duas posições do texto (i.e., número de palavras entre as ocorrências de duas palavras), as palavras com possíveis raízes comuns (e.g., medicina, telemedicina, médico, etc.), os padrões de expressões repetidas (i.e. repetições de sub-strings no texto), as palavras que são compostas por um conjunto de predefinido caracteres[5] (e.g., diagramas, trigramas, etc.) , etc.;

R16. Dada uma conversa, composta por mensagens, pretende-se conseguir ordenar, utilizando os algoritmos de ordenação *mergesort* ou *quicksort*, a conversa pelo timestamp, pelo tamanho de cada mensagem e pela sua importância (i.e., valor entre 1..100);

[3] *NB:* um documento é representado por uma *bag-of-words* e um *corpus* por um conjunto de bag-of-words
[4] https://en.wikipedia.org/wiki/Knuth%E2%80%93Morris%E2%80%93Pratt_algorithm.
[5] https://www.mat.uc.pt/~pedro/lectivos/CodigosCriptografia1011/interTIC07pqap.pdf

R17. Implementar o cálculo da distância de edição entre strings de modo, por exemplo, a fazer correção ortográfica com base em dicionários de palavras[6];

R18. Calcular a medida de semelhança entre dois documentos usando a representação *bag-of-words* e produto interno entre duas descrições de documentos[7].

R19. Permitir a manipulação das estruturas de dados através da entrada e saída para ficheiros de texto; Pretende-se a definição de um formato textual adequado para representar/suportar a informação dos modelos de dados e a sua exportação e importação para ficheiro;

R20. Permitir a manipulação das estruturas de dados através da entrada e saída para ficheiros binários; Pretende-se a definição de um formato binário adequado para representar/suportar a informação dos modelos de dados e a sua exportação e importação para ficheiro (este requisito refere-se apenas a LP1).

3 Documentação

1.1 Anotações e comentários no código fonte

As estruturas de dados e os algoritmos definidos devem ser implementados em C com os comentários apropriados que facilitem a compreensão dos mesmos, inseridos no código fonte desenvolvido. Todas as funções devem estar anotadas em formato doxygen[8] incluindo uma explicação dos algoritmos implementados e uma menção ao desempenho dos mesmos assim como dos testes efetuados/implementados. As principais estruturas de dados e variáveis devem também estar anotadas neste formato. Deverão usar o software doxygen para gerar a documentação com base nos comentários [9].

[6] ver *approximate string matching* e https://en.wikipedia.org/wiki/Levenshtein_distance.
[7] https://nlp.stanford.edu/IR-book/html/htmledition/dot-products-1.html.
[8] A geração de anotações neste formato é normalmente suportada pelos IDEs ou plugins associados (http://www.doxygen.org).
[9] https://www.stack.nl/~dimitri/doxygen/manual/docblocks.html#cppblock

1.2 Relatório

Os alunos deverão entregar um ficheiro de texto no formato .dox[10] do doxygen, descrevendo as funcionalidades/requisitos implementados, parcialmente implementados e não implementados. Devem mencionar sempre o número do requisito de acordo com a numeração utilizada neste documento de especificação:

1. Funcionalidades implementadas: devem identificar todas as funções desenvolvidas para assegurar os requisitos funcionais solicitados.
2. Funcionalidades não implementadas: devem identificar as funcionalidades não implementadas e apontar as justificações e/ou dificuldades que impediram o seu desenvolvimento.

2 Submissão

A aplicação final (código) deve estar depurada de todos os erros de sintaxe e de acordo com os requisitos funcionais pedidos. Só serão considerados os programas que não contenham erros de sintaxe e implementam as funcionalidades pedidas. A documentação, em html ou pdf, juntamente com todo o código-fonte desenvolvido, deve ser submetida num ficheiro zip/rar (project.zip) na plataforma *elearning.ufp.pt*.

[10] Ver o ficheiro "mainpage.dox" fornecido como exemplo.