Gestão de Contactos

2 de Junho de 2014

Grupo E:

Henrique Ferrolho – 120509079 – henriqueferrolho@gmail.com

Maria Marques – 120509104 – mariajoao.rmarques@gmail.com

Introdução

No âmbito da unidade curricular Conceção e Análise de Algoritmos do curso Mestrado Integrado em Engenharia Informática e Computação foi-nos proposto a implementação de funcionalidades eficientes de pesquisa de contactos.

A pesquisa deve ser dinâmica e a aplicação deve mostrar contatos que interessem ao utilizador, à medida que este escreve mais caracteres. Deve assim possibilitar que o utilizador se engane na escrita e o algoritmo de pesquisa deve sugerir correções que se aproximem da lista de contatos. Para isso recorremos à utilização do algoritmo de pesquisa aproximada de *strings EditDistance*.

Perguntas e respostas

Q: Que algoritmo será usado para a pesquisa de contatos?

A: O algoritmo *EditDistance*: um algoritmo de pesquisa aproximada de *strings*.

Q: Que contentor deverá ser usado para guardar os contatos durante o tempo de execução?

A: A escolha mais apropriada é a de usar o contentor *set*. Os *sets* não só armazenam elementos únicos segundo uma ordem específica, como também são implementados com uma árvore de pesquisa binária, o que será perfeito para operações de pesquisa, adição e remoção que teremos de implementar.

Q: Quanto tempo demorará uma inserção/remoção/pesquisa de contato?

A: Como o contentor dos contatos é um *set*, que novamente é implementado com uma árvore de pesquisa binária, estas operações terão uma complexidade temporal de O(log n).

Formalização do problema

Dados de entrada:

- contacts.txt:

```
nContacts
firstName    lastName    phoneNumber    email    address
firstName    lastName    phoneNumber    email    address
firstName    lastName    phoneNumber    email    address
...
```

- settings.txt: com o número máximo de resultados a apresentar numa pesquisa.

Restrições:

- Contatos diferentes não podem ter nomes iguais, número iguais ou e-mail iguais.
- O número de telemóvel tem de ter 9 dígitos, o e-mail tem de ter pelo menos um arroba (@) e um ponto (.).

Resultados esperados:

A aplicação deverá ser capaz de inserir/remover contatos fácil e intuitivamente, bem como pesquisar fluentemente um contato em específico.

Principais algoritmos implementados

O algoritmo implementado foi o *EditDistance*: um algoritmo de pesquisa aproximada de *strings* (o que foi apresentado nas aulas teóricas da unidade curricular).

O algoritmo consiste em construir uma matriz que é progressivamente preenchida com a distância entre o padrão, e o texto onde se procura o padrão.

Após a conclusão do algoritmo a célula inferior direita da matriz apresenta a distância entre as *strings*. A distância é um número inteiro que corresponde ao número de operações necessárias para transformar o padrão no texto. Podemos concluir então, que quanto menor a distância, maior é a similaridade entre as *strings*; e ainda que, quando a distância é zero, as *strings* são iguais. Este algoritmo apresenta uma complexidade temporal e espacial de: O(|P|.|T|).

Existe uma versão otimizada deste algoritmo que reduz a complexidade para: O(|T|). Isto é possível, uma vez que, em vez de utilizar uma matriz, usa apenas um vetor linear e duas variáveis auxiliares.

Lista de casos de utilização

É esperado que o utilizador use a aplicação como uma ferramenta simples e eficaz de gestão de contactos.

Em baixo encontra-se um diagrama das funcionalidades disponibilizadas pela aplicação desenvolvida.

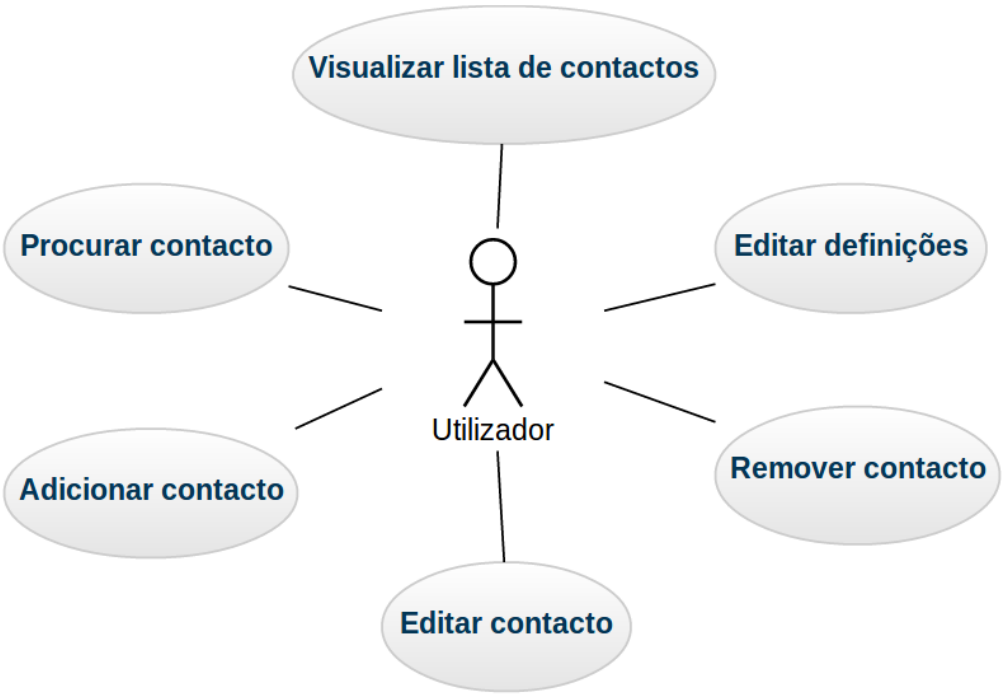


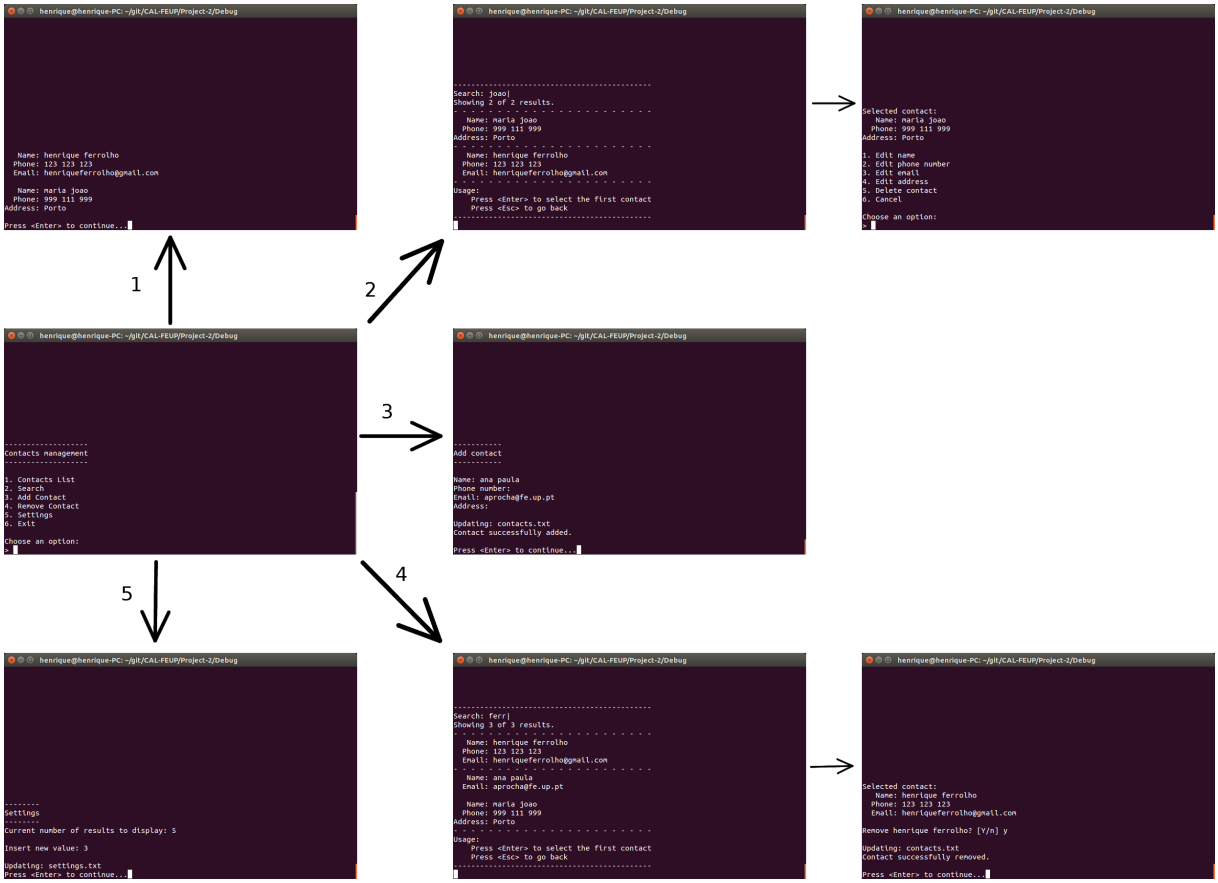
Fig. 1. *Diagrama de casos de utilização.*

A interface da aplicação foi desenvolvida para a linha de comandos. Para facilitar a navegação pela aplicação, todos os menus, submenus e respectivas solicitações de input para navegar entre estes estão implementados e encapsulados numa classe que gere toda a interface da aplicação: Interface.cpp.

Houve a preocupação de desenvolver uma aplicação versátil, que funcionasse tanto em Windows como em Unix. Para isso foi implementada uma função dependente do sistema em que a aplicação for compilada (ConsoleUtilities.cpp), necessária para a pesquisa dinâmica.

Relativamente à pesquisa dinâmica: é aconselhável não usar a consola do Eclipse, pois através desta a pesquisa dinâmica não funcionará; é aconselhável usar a linha de comandos se estiver a usar Windows, ou o terminal se estiver a usar um ambiente Unix.

De seguida é apresentado um diagrama relativamente aos diferentes menus e submenus da aplicação:



Dificuldades

A única dificuldade sentida foi na implementação dos algoritmos devido a um bug no pseudo-código dos slides das aulas teóricas, mas foi facilmente ultrapassado após testar a aplicação em tempo de execução.

Contribuição no projeto

Henrique Ferrolho	50%
Maria João Marques	40%
Rui Gonçalves	10%