Sistema de Evacuação Parte 2

Tema 3

Bruno Carvalho
up201606517@fe.up.pt

24 de Maio de 2018

Conteúdo

1 Introdução

1.1 Definições preliminares

Começamos com as seguintes definições:

- P refere-se ao padrão a pesquisar, ou ao seu comprimento;
- T refere-se ao texto a ser pesquisado, ou ao seu comprimento.
- L é o alfabeto de T e de P, ou a sua cardinalidade;
- E é o subconjunto de L definido por P, ou a sua cardinalidade;
- A o texto a ser transformado num algoritmo de distância de edição;
- B o texto alvo num tal algoritmo.

1.2 Tema 3

1.2.1 Especificação do Tema

A especificação e descrição do Tema 3 – Sistema de Evacuação é a seguinte:

> Para a segunda parte deste trabalho, considere que as ruas têm nomes, por exemplo "Rua de Dr Roberto Frias" ou "A1", e que pertencem a um dado distrito, por exemplo, "Porto". Estenda o trabalho realizado com funcionalidades apropriadas que permitem a um automobilista ligar para uma linha de emergência (e.g. 112), dar a sua posição, e solicitar uma rota para evacuação; com o nome do da rua onde o automobilista está, o sistema retorna a rota de evacuação. Implemente esta funcionalidade, considerando tanto pesquisa exata, assim como pesquisa aproximada, das strings identificativas dos nomes das ruas fornecidas. Para pesquisa exata, caso o nome de rua não exista, deverá retornar mensagem de lugar desconhecido. Para a pesquisa aproximada, deverá retornar os nomes de ruas mais próximos, ordenados por similaridade.
>
> Estas novas funcionalidades deverão ser integradas no trabalho já realizado para a primeira parte. Avalie a complexidade (teórica e empiricamente) dos algoritmos implementados em função dos dados de input usados.

1.2.2 Descrição da Implementação

No seguimento das aulas de pesquisa aproximada e exata foram implementados algoritmos de pesquisa exata, algoritmos de distância de edição e os equivalentes algoritmos de pesquisa aproximada (*fuzzy search*).

O utilizador do programa seleciona o algoritmo a utilizar

2 Interface

No seguimento das aulas de pesquisa aproximada e exata foram implementados algoritmos de pesquisa exata, algoritmos de distância de edição e os equivalentes algoritmos de pesquisa aproximada (*fuzzy search*).

O utilizador do programa seleciona o algoritmo a utilizar

1. **Seleção do algoritmo.**

   O utilizador seleciona o algoritmo a utilizar da lista.

2. **Seleção da rua.**

   O utilizador indica o nome da *rua* a pesquisar com a sua posição. Dependendo do tipo de algoritmo, claro, o nome da rua encontrada deverá ser exatamente igual ou aproximadamente igual ao dado.

3. **Seleção de uma das ruas encontradas.**

   Se alguma rua for encontrada o utilizador deverá escolher aquela em que se encontra de entre as selecionadas. As ruas aparecem destacadas visualmente no grafo.

4. **Seleção da rua de destino.**

   Novamente o utilizador introduz o nome da rua para onde pretende evacuar, o sistema pesquisa o nome da rua com o mesmo algoritmo e pede ao utilizador novamente para escolher.

5. **Melhor caminho.**

   Com as duas ruas o sistema encontra o caminho mais rápido entre as duas ruas.

3 Algoritmos

Implementamos *Boyer-Moore* (com *shift rules* genéricas) e *Knuth-Morris-Pratt* para pesquisa exata.

Para distância de edição estão implementadas as distâncias de *Levenshtein* (inserção, remoção, substituição), *Damerau-Levenshtein* simplificado (*Optimal string alignment distance* segundo a *Wikipédia*) e *Damerau-Levenshtein* (inserção, remoção, substituição, transposição) – para pesquisa aproximada estão os implementados os equivalentes a estes.

Implementação: Boyer-Moore As regras definidas pelo algoritmo (*bad character rule* e *good suffix rule*) estão feitas de forma genérica, permitindo várias estruturas internas.

A *bad character rule* tem duas implementações disponibilizadas: uma com tabela de tamanho $L \times P$, outra com uma tabela mapeada de tamanho $E \times P$, requirindo menos memória.

A classe boyer_moore encapsula as duas regras para os algoritmos de pesquisa. As implementações do algoritmo e da regra *good suffix rule* seguem as orientações do artigo originalJ. Strother Moore e Robert S. Boyer. "A Fast String Searching Algorithm". Em: (1977). URL: http://www.cs.utexas.edu/~moore/publications/fstrpos.pdf (acedido em 18/05/2018).

Implementação: Knuth-Morris-Pratt A implementação segue as orientações dos slides das teóricas. O preprocessador é, à semelhança do algoritmo anterior, uma classe à parte.

Em ambos os algoritmos anteriores os preprocessamentos de P são reutilizáveis – é feito uma vez e aplicado a todas as ruas a pesquisar.

Implementação: Levenshtein A implementação segue mais uma vez o exemplo das teóricas. Há duas implementações: uma usando uma matriz $P \times T$ e outra usando dois vetores de tamanho T, poupando memória.

Implementação: Damerau-Levenshtein As implementações seguem as orientações bastante detalhadas em Wikipedia entry on the Damerau-Levenshtein distance. URL: https://en.wikipedia.org/wiki/Damerau%E2%80%93Levenshtein_distance (acedido em 22/05/2018).

O algoritmo simplificado tem uma implementação usando uma matriz $P \times T$ e outra implementação usando uma matriz $3 \times T$. O algoritmo *Damerau-Levenshtein* não pode ser simplificado de uma forma análoga aos anteriores e tem de ser implementado usando uma matriz $P \times T$.

Implementação: Pesquisa aproximada Os algoritmos de pesquisa aproximada são adaptados de forma direta dos algoritmos de distância de edição – nestes casos, estamos interessados em saber a menor das distâncias das *substrings* de T a P, em vez da distância entre T e P.

Referências

A Boyer-Moore example. URL: https://www.cs.utexas.edu/users/moore/best-ideas/string-searching/fstrpos-example.html (acedido em 22/05/2018).

Wikipedia entry on the Damerau-Levenshtein distance. URL: https://en.wikipedia.org/wiki/Damerau%E2%80%93Levenshtein_distance (acedido em 22/05/2018).

Apostilicao, Alberto e Raffaele Giancarlo. "The Boyer-Moore-Galil String Searching Strategies Revisited". Em: (1985).

Hyyro, Heikki. *On Boyer-Moore Preprocessing*. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.59.9557&rep=rep1&type=pdf (acedido em 21/05/2018).

Moore, J. Strother e Robert S. Boyer. "A Fast String Searching Algorithm". Em: (1977). URL: http://www.cs.utexas.edu/~moore/publications/fstrpos.pdf (acedido em 18/05/2018).