Universidade do Porto LEIC

Desenho de Algoritmos 2021/2022

Folha Prática 7

1. A distância mínima de edição *(minimum edit distance)* entre duas *strings* x e y, introduzida Levenshtein (1966) é habitualmente definida como o número mínimo de operações elementares (**inserções**, **remoções** e **substituições**) necessárias para transformar x em y. Pretendemos determinar as k palavras mais próximas de uma palavra dada por x num ficheiro, para tal métrica. Implemente em C++ as funções:

```
int editDistance(std::string x, std::string y);
void nearest(std::string filename,std::string x,int k,
             std::vector<std::pair<int,std::string> > &result);
```

Caso haja empates, deve dar preferência às *strings* que encontrou primeiro.

2. As frequências relativas das letras em Português (em percentagem), segundo Wikipedia, são:

a	14.63	b	1.04	c	3.88	d	4.99
e	12.57	f	1.02	g	1.30	h	1.28
i	6.18	j	0.40	k	0.02	l	2.78
m	4.74	n	5.05	o	10.73	p	2.52
q	1.20	r	6.53	s	7.81	t	4.34
u	4.63	v	1.67	w	0.01	x	0.21
y	0.01	z	0.47				

Admitindo que tem um ficheiro apenas com vogais (a, e, i, o, u) e que efetua uma compressão com representação de cada vogal por o mesmo número de bits, qual seria o comprimento mínimo por vogal? Se o ficheiro tiver n caracteres, qual seria o número de bits usado nessas condições e qual seria o valor esperado se se efetuasse uma compressão por aplicação do algoritmo de Huffmann?

3. **(extra-aula)** Implemente uma função para determinar a codificação de Huffmann dada uma tabela de caracteres e de frequências, como a indicada acima. A função deve retornar a árvore.

a) Comece por efetuar uma implementação com complexidade temporal $O(m^2)$, sendo m o número de caracteres e, posteriormente, altere-a para ser suportada por fila de prioridade com complexidade $O(m \log m)$.

b) Use a função definida na alínea anterior para construir a árvore e a seguir mostrar os códigos que atribui a cada letra para a instância indicada (efetue pesquisa em profundidade da esquerda para a direita na árvore).