O PROBLEMA DO ALGORITMO DAS DIFERENÇAS

ALAIR PEREIRA DO LAGO

ORIENTADOR: IMRE SIMON

Façamos algumas definições para melhor definir o problema:

Definições:

Chamamos de Alfabeto a um conjunto de letras e de palavra a toda seqüência finita de letras de um dado alfabeto.

À cardinalidade de uma palavra chamamos: com rimento da palavra.

Dizemos ainda que uma subseqüência de uma palavra constitui uma subpalavra.

Exemplos:

(A menos de uma menção explícita, o alfabeto a ser mencionado será o alfabeto comum.)

abcdec e abc são duas palavras de comprimento 6 e 3 sendo que a segunda é subpalavra da primeira. ba não é subpalavra de nenhuma, enquanto que dc o é da primeira.

Assim, o problema pode ser descrito nestes termos:

"Dado um alfabeto específico e duas palavras neste alfabeto, achar a subpalavra comum de maior comprimento".

Antes de prosseguir, faremos um breve relato de onde,

historicamente, surgiu este problema e qual a sua atualidade.

Em situações como esta recai-se na resolução do problema em questão:

. Dadas duas versões diferentes de um mesmo "software" ou de um mesmo arquivo, achar o que mais tem de comum em ambos de modo a guardar a versão original e apenas as diferenças da segunda versão para a primeira.

. Analisar e classificar Macromoléculas de uma mesma família (DNA por exemplo) identificando a máxima semelhança entre elas e o grupo de átomos que as distingüem.

. No estudo de reconhecimento de voz, identificar a máxima semelhança entre uma fala pré-programada e a de uma pessoa que fala.

Observamos ainda que para a primeira motivação, por exemplo, já foram implementados vários programas. O diff do UNIX é uma implementação (mais conhecida). O TURBOC traz um utilitário que implementa um algoritmo para o problema (é o FILECOMP).

Queremos observar ainda que existem programas comparadores tais como o COMP (do MS-DOS) que não implementam um algoritmo para o problema em questão, pois comparam seqüencialmente, letra a letra, de forma síncrona. Para um programa como este, basta inserir uma linha no começo de um arquivo para que se perca o sincronismo e não seja encontrada nenhuma semelhança.

É exatamente a exigência de não sincronia que complica este problema.

Voltando à análise, veremos agora uma sua solução comumente encontrada e redescoberta. É essa:

Sendo duas palavras A e B de comprimento m e n defino a matriz P_{ij}, $0 \leq i \leq m$, $0 \leq j \leq n$ tal que p_{ij} seja o comprimento da maior subpalavra comum entre as i primeiras letras da primeira palavra e as j primeiras letras da segunda.

faço $p_{i\emptyset} = 0$, $i=0,\ldots,m$ e $p_{\emptyset j} = \emptyset$, $j=\emptyset,\ldots,n$

$$\text{faço } p_{ij} = \begin{cases} 1 + p_{i-1,j-1} & \text{se } A_i = B_j \\ \max\{p_{i-1,j}, p_{i,j-1}\} & \text{se } A_i \neq B_j \end{cases}$$

$$\text{para } \begin{matrix} 1 \leq i \leq m \\ 1 \leq j \leq n \end{matrix}$$

Desta maneira $P_{m,n}$ conterá o comprimento da maior subpalavra.

A partir da matriz $P_{i,j}$ podemos obter a maior subpalavra comum.

d	0	1	1	2
c	0	1	1	2
a	0	1	1	1
	0	0	0	0
		a	b	c

Este algoritmo, no entanto, é estritamente quadrático. E isto é o mais inconveniente do mesmo, pois para um grande número de letras o tempo de processamento, o espaço de memória, tempo de resposta podem se tornar impraticáveis.

A grande pesquisa que se faz em cima do problema é esta: achar um algoritmo linear para resolver o problema. Até agora é uma questão em aberto se existe um tal algoritmo. Já se conseguiu um algoritmo de complexidade menor que a quadrática, mas não utilizável na prática e já se conseguiu algoritmos que na "maioria dos casos" é linear, mas mesmo estes últimos são sujeitos a "maus exemplos".