Biologia Computacional - 2004/2	12/07/04
Aula 5: Computando Alinhamentos Múltiplos	
Instrutor: Berilhes Borges Garcia	Escriba: Luiz Celso Gomes Jr.

DRAFT

1 Introdução

Considere o problema de computar um alinhamento múltiplo de strings ótimo. A primeira questão a ser respondida é: Qual é a medida de adequação que vai ser otimizada?

Não existe nenhuma forma universalmente aceita de avaliar alinhamentos múltiplos.

Na prátrica muitos métodos computam alinhamentos múltiplos sem qualquer garantia de sua qualidade!

A soma de pares de escores é uma medida simples que é muitas vezes utilizada.

2 A soma de pares de escores

A soma de pares de escores é definida da seguinte forma:

O alinhamento dois a dois de duas strings S_i e S_j participando em um alinhamento múltiplo M é obtido removendo-se todas as linhas de M exceto as linhas de S_i e S_j.

O escore deste alinhamento induzido é o escore do alinhamento das duas strings. (O escore de um alinhamento espaço/espaço é considerado 0).

O escore das somas de pares (sum of pairs - SP) de um alinhamento múltiplo M é a soma dos escores de todos os alinhamentos dois a dois induzidos por M.

Example: Exemplo de escore SP: Considere um alinhamento múltiplo M das strings $S_1 = AAGAAA$, $S_2 = ATAATG$ e $S_3 = CTGGG$:

```
S1':  A A G A A _ A
S2':  A T _ A A T G
S3':  C T G _ G _ G
```

⊠

Com a distância de edição padrão o escore SP de M é: $D(S_1', S_2')+D(S_1', S_3')+D(S_2', S_3') = 4+5+5$

Os resultados a serem apresentados se aplicam tanto à distância de edição (ponderada) quanto ao escore de alinhamento de duas strings.

3 Resolvendo o problema do Alinhamento SP

Definição: Dado um conjunto de strings $S_1, ..., S_k$, o problema do alinhamento SP é computar um alinhamento global que tenha um escore soma-de-pares minimal.

O problema do alinhamento SP pode ser resolvido de forma exata por programação dinâmica em tempo $O(n^k)$, se $n = |S_1| = ... = |S_k|$. Este método é impraticável para mais que 4 strings, quando o tamanho destas é algumas centenas de caracteres.

Tempo exponencial parece ser inevitável, uma vez que o problema foi mostrado ser NP-completo.

Vamos esquematizar as idéias principais de um algoritmo de programação dinâmica para computar o alinhamento SP otimal de strings $A[1...n_1]$, $B[1...n_2]$, e $C[1...n_3]$.

De modo a simplificar a notação, vamos assumir que o escore de qualquer alinhamento caractere/espaço é d.

O método preenche um vetor $D(i, j, k)$ de tamanho $(n_1 + 1) \times (n_2 + 1) \times (n_3 + 1)$, onde $D(i, j, k)$ é o escore do alinhamento SP otimal dos prefixos $A[1...i]$,$B[1...j]$ e $C[1...k]$.

3.1 Recorrências para o alinhamento SP

Casos Bases:

$D(0, 0, 0) = 0$, o que é óbvio

O que nós já sabemos acerca das fronteiras iniciais de $D(i, j, k)$ com $i = 0$, $j = 0$, ou $k = 0$?

Considere $D(i, j, k)$: Qual é o escore SP para um alinhamento otimal de $A[1...i]$,$B[1...j]$ e $C[1...0] = e$?

A: $D_{A,B}(i, j) + (i + j) \times d$, onde $D_{A,B}(i, j)$ é a distância de edição de $A[1...i]$ e $B[1...j]$.

Fórmulas para $D(i, 0, k)$ e $D(0, j, k)$ são similares.

3.2 Computando as células internas de $D(i, j, k)$

Os valores das células internas $D(i, j, k)$ com $i, j, k > 0$ depende de sete células adjacentes $D(i', j', k')$ onde $i' = i - 1$, $j' = j - 1$, $k' = k - 1$.

O ótimo para $D(i, j, k)$ é o mínimo dos seguintes sete casos:

Primeiro, um alinhamento ótimo de $A[1...i]$, $B[1...j]$, e $C[1...k]$ pode alinhar os últimos caracteres de cada uma destas strings dando o seguinte escore:

$D(i-1, j-1, k-1) + s(A[i], B[j]) + s(A[i], C[k]) + s(B[j], C[k])$

Segundo, qualquer uma das três strings pode terminar com um espaço alinhado com os últimos caracteres das outras duas. Isto dá um escore de:

$D(i-1, j-1, k) + s(A[i], B[j]) + 2d$

(se "_" é inserido no final de C; o mesmo para A e B)

Finalmente, qualquer uma das duas strings pode terminar com um espaço alinhado contra o último caractere da terceira string, por exemplo C, logo o escore é:

$D(i, j, k-1) + 2d$

Cada célula pode ser preenchida em tempo constante de acordo com as recorrências mostradas anteriormente. O tempo total para preencher a matriz da programação dinâmica é $O(n_1 n_2 n_3)$.