MO 640 – Biologia Computacional

27/09/2004 - Ata de Aula

Prof. João Meidanis
Autor: Leandro Rodrigues Magalhães de Marco, RA 009089

1. Introdução

Serão cobertos nesta Ata os assuntos discutidos na aula do dia 27/09/2004. Mais especificamente serão tratados os seguintes assuntos:

- Funções Afins
- Implementação do modelo baseado em funções afins.
- Algoritmo de comparação de seqüências similares

2. Funções Afins

Foi definido na aula o conceito de função afim. Funções afins são funções com o comportamento definido pela equação

$$F(x) = ax + b$$

Sendo assim, uma função afim é uma função que e linear e necessariamente não passa pela origem. Toda função afim é linear, mas nem toda função linear é afim.

3. Implementação do modelo baseado em funções afins.

Foi descrito o modelo baseado em funções afins que trata o problema de penalizar espaços introduzidos em seqüência para efetuar alinhamento das mesmas.

As fórmulas que regem os modelos contém os seguintes termos que exigem explicação

s e *t* são duas seqüências

a[*i*, *j*] = máxima pontuação de um alinhamento entre *s*[*1..i*] e *t*[*1..j*] que termina com *s*[*i*] casado com *t*[*j*].

b[*i*, *j*] = máxima pontuação de um alinhamento entre *s*[*1..i*] e *t*[*1..j*] que termina com um espaço casado com *t*[*j*].

c[*i*, *j*] = máxima pontuação de um alinhamento entre *s*[*1..i*] e *t*[*1..j*] que termina com *s*[*i*] casado com um espaço.

p[*i*, *j*] = pontuação de um casamento entre *s*[*i*] e *t*[*j*]

$$a[i, j] = p[i, j] + \max \begin{cases} a[i-1, j-1] \\ b[i-1, j-1] \\ c[i-1, j-1] \end{cases}$$

$$b[i, j] = \max \begin{cases} -(h+g) + a[i, j-1] \\ -g + b[i, j-1] \\ -(h+g) + c[i, j-1] \end{cases}$$

$$c[i, j] = \max \begin{cases} -(h+g) + b[i-1, j] \\ -(h+g) + b[i-1, j] \\ -g + c[i-1, j] \end{cases}$$

Foi discutido também o fato do espaço requerido para o algoritmo triplicar.

Ainda com relação ao algoritmo discutiu-se o fato de alguns termos serem inicializados com valores infinitos. Esses infinitos na realidade são representados computacionalmente como constantes muito grandes.

Por fim, vimos que o tempo de execução do algoritmo é quadrático.

4. Algoritmo de comparação de seqüências similares

Para comparar seqüência similares usamos um algoritmo especial que tira vantagem da similaridade, para diminuir o tempo de execução.

Como as duas seqüências são similares, então o alinhamento ótimo estará na diagonal principal da matriz. Sendo assim não é necessário preencher a matriz toda. Utilizamos uma banda ao redor da diagonal principal.

A largura da banda começa com o valor 1 e vamos dobrando o seu valor a cada iteração.

Um ponto que gerou muita dúvida foi a condição de parada do algoritmo. Este algoritmo pára quando a pontuação do alinhamento é maior ou igual à máxima pontuação com k+1 espaços. Assim temos a seguinte condição de parada:

$$a_k[n, n] \geq M(n-k-1) + 2(k+1)g$$