MAC 5726 – Biologia Computacional

Primeira lista de exercícios – Entrega **13 de setembro de 1999**

Exercício 1 O *Algoritmo Alinhamento* determina um alinhamento ótimo entre dois strings s e t. Modifique o algoritmo para imprimir *todos* os alinhamentos ótimos.

Seu algoritmo deve gastar tempo $O(|s|+|t|)$ para imprimir cada alinhamento. Explique porque isso é verdade.

Exercício 2 Com um sistema de pontuação com $g = 0$, $p(i,j) = 1$ se $s[i] = t[j]$ e $p(i,j) = 0$ se $s[i] \neq t[j]$, os símbolos alinhados num alinhamento ótimo formam uma *subseqüência* de comprimento máximo. Prove que isso é verdade. Dê detalhes em como *encontrar* a subseqüência.

Exercício 3 (Tandem Repeats)
Seja P um padrão de comprimento n e T um texto de comprimento m. Seja P^m a concatenação de P com ele próprio m vezes (P^m tem comprimento mn). Queremos computar um alinhamento local entre P^m e T. Com isso, encontraremos um intervalo em T que tem o melhor alinhamento global com alguma repetição do padrão P.

Este problema aparece no estudo de estruturas secundárias de proteínas que formam a chamada *coiled-coil*. Naqule contexto, P representa um *motif* ou *domínio* que pode repetir na proteína um número desconhecido de vezes, e T representa a proteína.

Um alinhamento local entre P^m e T escolhe um intervalo em T que consiste "otimamente" de repetições seguidas de P. Se P^m é criado explicitamente, então o algoritmo de alinhamento local resolve o problema em tempo $O(nm^2)$. Mas como P^m consiste de m cópias idênticas de P, um algoritmo $O(nm)$ é possível. O método consiste essencialmente em simular o que o algoritmo de programação dinâmica faria se fosse executado com P^m e T explicitamente.

O algoritmo de programação dinâmica irá preencher uma matriz a com $m+1$ linhas e $n+1$ colunas. Linha 0 e coluna 0 são inicializadas com 0's. Então, para cada linha i, o algoritmo faz o seguinte: executa o algoritmo padrão para na linha i; atribui a $a[i,0]$ o valor de $a[i,n]$ e executa novamente o algoritmo padrão para a linha i. Depois de preencher cada linha, o algoritmo escolhe como resposta a maior entrada de a.

Determine a complexidade desse algoritmo e prove que ele encontra corretamente o valor do alinhamento ótimo entre P^m e T. Dê detalhes de como encontrar o alinhamento ótimo local. Discuta porque P foi expandido (conceitualemente) para P^m e não para um string maior ou menor.

Exercício 4 (The tRNA folding problem)
Esta é uma versão bastante primitiva do problema da predição da estrutura secundária (planar) do tRNA. Seja s um string de n caracteres representando um tRNA. Definimos um *pareamento* como um conjunto de pares disjuntos de s. Um pareamento é chamado *próprio* se ele contém somente pares $\{A, U\}$ ou $\{C, G\}$, ou seja, somente pares de nucleotídeos complementares. Se desenharmos s como um string curcular, definimos como pareamento *aninhado* um pareamento próprio onde cada par no pareamento é ligado por um segmento de linha dentro do círculo, e onde os segmentos não se cruzam (ver Figura 1). O problema é encontrar um pareamanto aninhado de maior cardinalidade. Mostre como resolver este problema em tempo $O(n^3)$.

Agora, modifique o problema adicionando pesos aos pares: o peso de um par $\{A, U\}$ é diferente do peso de um par $\{C, G\}$. O objetivo agora é encontrar um pareamento aninhado de peso total máximo. Dê um algoritmo eficiente para este problema.

Exercício 5
Implemente um dos seguintes algoritmos.

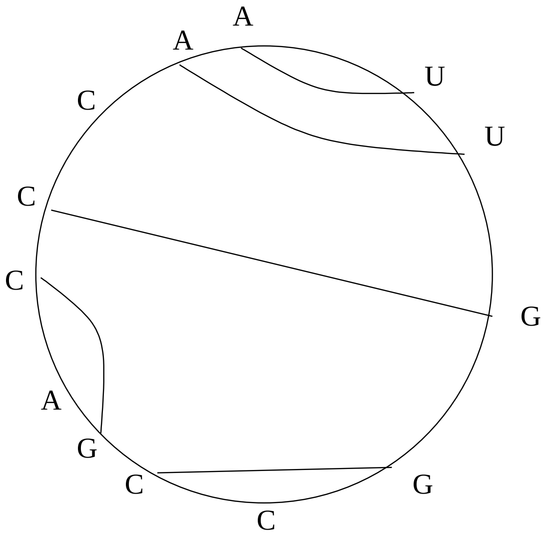

Figura 1: Representação de um pareamento aninhado

1. Dados strings s e t, calcular o alinhamento entre eles usando espaço $O(|s| + |t|)$.
2. Dados strings $s_1, s_2, \ldots, s_k$, calcular o valor do alinhamento múltiplo ótimo entre eles.

Mande um e-mail para mim dizendo como faço para acessar ou copiar ou compilar seu programa.