Exercício Programa 1 - Reversões em DNA

Tópicos de Programação

11 de janeiro de 2010

1 Descrição do problema

Esse EP foi utilizado pela disciplina MAC 122 do curso de Bacharelado em Ciência da Computação em 2008 (`http://www.ime.usp.br/~am/122/eps/EP2.html`). *Algumas adaptações foram feitas para viabilizar a sua aplicação no curso de verão Tópicos de Programação.*

Este exercício-programa está baseado num problema que ocorre em Biologia Computacional. Todos sabem sobre DNA. Para muitos aspectos, um DNA pode ser resumido como um string nas letras ATCG; um baita stringão! O DNA pode sofrer uma série de mutações, alguma das quais podem ser descritas como transformações algorítmicas no string. Uma delas é a **reversão**, e este problema consiste em simular uma série de reversões.

Dada uma sequência de DNA inicial (descrita por uma sequência de N caracteres, com valor A, T, C ou G), uma reversão consiste em destacar um trecho contínuo desta sequência e recolocá-lo na ordem invertida, invertendo a sequência de caracteres deste trecho. A operação é descrita pelos índices da posição e final do trecho (indexamento começando em 1).
Exemplo de reversão:

$$AT\underbrace{AATGC}_{\text{Inicial}}TATG$$

$$AT\quad\underbrace{CGTAA}_{\text{Reversão em [3, 7]}}\quad TATG$$

2 A Tarefa

Seu programa deve ler, da entrada padrão (stdin), uma sequência de DNA inicial e uma série de operações de inversão a serem realizadas em sequência

e imprimir a sequência resultante.

2.1 Entrada

A entrada conterá 3 linhas. A primeira linha conterá um número, M, com a quantidade de operações a serem realizadas. A segunda linha conterá uma sequência de N caracteres representando o DNA inicial, cada caracter com valor A, T, C ou G. A terceira linha conterá M pares de números (a_i e b_i, respeitando $1 \leq a_i < b_i \leq N$) representando o índice inicial e final de cada operação.

2.2 Saída

A saída será a sequência de Dna após a aplicação da sequência de operações de reversão.

2.3 Restrições

Intervalos: $1 < N \leq 10.000.000$; $1 \leq M \leq 10.000$
Seu programa deverá ser eficiente.

2.4 Exemplo de entrada e saida

```
3
ATATGCGA
3 6 4 8 3 4
-----------
ATACGATG
```

Veja o que aconteceu:

Sequência inicial:

$$ATATGCGA$$

Sequência 2:

$$AT \quad \underbrace{CGTA} \quad GA$$

Operação 1: [3, 6]

Sequência 3:

$$ATC \quad \underbrace{AGATG}$$

Operação 2: [4, 8]

Sequência final:

$$AT \quad \underbrace{AC} \quad GATG$$

Operação 3: [3, 4]

3 Considerações finais

O objetivo desse EP é desenvolver a capacidade de propor estrutura de dados para a aplicações específicas com o objetivo de melhorar a eficiência da aplicação.

Uma solução intuitiva, pelo menos inicialmente, usa um vetor para armazenar e processar a string. Após cada reversão, o algoritmo muda a ordem das letras no vetor. Entretanto, para strings grandes com muitas reversões, isso pode deixar o tempo de processamento muito grande.

Explore uma idéia alternativa usando listas ligadas na qual você não tenha que inverter a string caractere por caractere.

A nota recebida por esse EP depende fortemente não somente da correção mas também da eficiência de seu programa. Portanto, para que você possa estar seguro de que seu programa é rápido, deixamos alguns testes para você testar o seu programa em `http://www.linux.ime.usp.br/~shakavp/Testes.zip`.

Compare o tempo de seu processamento com o tempo gasto por nós. O processador que utilizamos é um processador*AMD Semprom 2800+*.

Teste	Tempo gasto
exemplo1.in	0.027s
exemplo2.in	0.039s
exemplo3.in	0.063s
exemplo4.in	0.053s
exemplo5.in	0.044s
exemplo6.in	0.411s
exemplo7.in	0.218s
exemplo8.in	1.384s
exemplo9.in	1.546s
exemplo10.in	2.440s