Construção e Análise de Algoritmos

lista de exercícios 22

1. **Subsequência comum mais longa**

   Considere uma sequência de números

   $$A \;=\; (a_1,\; a_2,\; a_3,\; \ldots,\; a_n)$$

   Nós dizemos que $C = (c_1, \ldots, c_k)$ é uma *subsequência* de A se os números $c_1, \ldots, c_k$ podem ser encontrados na sequência A nessa ordem, mas não necessariamente um ao lado do outro.

   Por exemplo, se

   $$A \;=\; (8,\; 3,\; 4,\; 6,\; ,11,\; 5,\; 9,\; 12) \qquad \text{e} \qquad C \;=\; (8,\; 11,\; 5,\; 12)$$

   então é fácil ver que C é uma subsequência de A

   $$A \;=\; (\underline{8},\; 3,\; 4,\; 6,\; ,\underline{11},\; \underline{5},\; 9,\; \underline{12})$$

   Nós dizemos que C é uma *subsequência comum* de A e B se C é uma subsequência de ambas as sequências A e B.

   O nosso problema de otimização consiste em encontrar uma subsequência comum de A e B com o maior comprimento possível.

   a) Apresente um algoritmo de programação dinâmica para esse problema.

   b) Estime a complexidade do seu algoritmo em termos dos comprimentos n, m das sequências A e B.

      (É possível resolver esse problema em tempo $O(nm)$.)

2. **Subsequência crescente comum mais longa**

   Uma variante natural do problema anterior consiste em encontrar subsequências *crescentes* comuns a duas sequências A e B.

   Por exemplo, se

   $$A \;=\; (8,\; 3,\; 4,\; 6,\; ,11,\; 5,\; 9,\; 12) \qquad \text{e} \qquad B \;=\; (5,\; 3,\; 6,\; 11,\; 9,\; 15,\; 12,\; 14)$$

   então $C = (3, 6, 9, 12)$ é uma subsequência crescente comum de A e B.

   Apresente um algoritmo que encontra a subsequência crescente comum mais longa de A e B, e estime a sua complexidade.

3. **O problema do roupeiro**

O futebol é democrático: todos podem jogar, tanto os baixinhos como os grandões.

Mas, isso acaba gerando um problema para o roupeiro.

Quer dizer, como hoje em dia os times trocam de jogadores o tempo todo, não dá para ficar comprando uniformes novos a toda hora (ao menos na 2a divisão).

Por outro lado, também não pega bem jogar com a roupa muito apertada, nem com os calções lá embaixo nos joelhos.

E é aqui que aparece o problema do roupeiro.

Mais especificamente, dado um conjunto de jogadores que vestem os tamanhos

$$t_1,\ t_2,\ t_3,\ \ldots,\ t_n$$

e uma coleção de uniformes com os tamanhos

$$u_1,\ u_2,\ u_3,\ \ldots,\ u_m$$

com $m \geq n$, o problema consiste em distribuir uniformes para o jogadores de modo a minimizar a seguinte função objetivo

$$\frac{1}{n} \cdot \sum_i \left(t_i - u_{(i)}\right)^2$$

onde $u_{(i)}$ é o uniforme dado ao i-ésimo jogador. (A ideia dessa função objetivo é que ninguém fique com um uniforme de tamanho muito diferente do seu.)

a) Apresente um algoritmo de programação dinâmica para esse problema.

b) Estime a complexidade do seu algoritmo em termos de n e m.
(É possível resolver esse problema em tempo $O(nm)$.)

4. **Palíndromes**

Como todos sabem, uma palíndrome é uma sequência de caracteres que é a mesma coisa quando lida da esquerda para a direita ou da direita para a esquerda.

Por exemplo,

```
Acuda cadela da Leda caduca.
```

Nesse caso, nós estamos interessados em encontrar subsequências dentro de uma sequência maior, que formam uma palíndrome — os caracteres da palíndrome não precisam aparecer juntos na sequência maior.

Apresente um algoritmo que encontra a maior subsequência de `A[1..n]` que forma uma palíndrome. (Seu algoritmo deve executar em tempo $O(n^2)$).

5. **Teste de DNA**

O problema da edição, que vimos na Seção 3 da aula 22, tem uma aplicação direta na comparação de sequências de DNA.

Mas, para aplicar as ideias nesse contexto, é preciso levar em conta como as coisas acontecem na biologia.

Quando ocorrem erros no processo de replicação genética, é comum que ocorra a inserção ou a remoção de toda uma sequência de nucleotídeos

Por exemplo,

```
A T T G C T C A     ==>     A T T C A
G C T C G           ==>     G C T A C A C G
```

Mas, quando o algoritmo realiza o alinhamento entre as duas sequências

```
A T T G C T C A          G C T - - - C G
A T - - - T C A          G C T A C A C G
```

aquilo que era para ser considerado como um único erro, será tratado como toda uma sequência de erros.

a) Modifique o algoritmo apresentado na Seção 3 da aula 22, de modo que as sequências de brancos (-) inseridas durante o processo de alinhamento sejam contadas como apenas 1 erro.

b) Para tornar o algoritmo mais realista, nós podemos levar em conta o tamanho da sequência de brancos da seguiente maneira

- o primeiro branco da sequência conta como 1 erro
- cada branco adicional na sequência conta como $1/k$ erro

onde k é um parâmetro do algoritmo, ajustado antes da execução.

Modifique o algoritmo do ítem (a) para levar em conta essa função de erro.