MAC–IME–USP

Imre Simon
Sala 290A Tel.: 3818 6142
email is@ime.usp.br
Monitora: Elidia Y. Itikawa

Carlos Eduardo Ferreira
Sala 297A Tel.: 3818 6140
e-mail cef@ime.usp.br
Monitor: Marcio C. Cabral

MAC 122 – Princípios de Desenvolvimento de Algoritmos

Segundo semestre de 2000

Exercício-Programa 3 – Entrega: **21 de novembro de 2000**

Ordenação
(pode ser feito em duplas)

> Este exercício-programa será avaliado pelo código e também pelos testes que vocês fizerem para avaliar a corretude e eficiência do algoritmo, assim como o relatório com os resultados que vocês obtiverem para o problema proposto.

Um dos algoritmos que estudamos nas aulas para ordenação de vetores foi o **bubblesort**. A idéia deste algoritmo é aplicar sucessivamente inversão de pares de elementos que estão na ordem incorreta, até que ao final de certo número de passos o vetor estará ordenado. A idéia deste exercício-programa é oferecer uma operação de inversão mais "poderosa", e buscar algoritmos de ordenação mais eficientes utilizando tal operação.

Escreva uma função `inverte` com o seguinte protótipo

```
int inverte (int v[], int i, int k, int n)
```

que, quando aplicada inverte a ordem dos elementos `v[i], v[i+1], ..., v[i+k-1]`, caso isso seja possível, e retorna 1. Se $i+k-1 \geq n$, a função não altera a ordem dos elementos no vetor e retorna 0.

Exemplo: Suponha que o vetor contém os elementos 1, 3, 4, 2, 6, 7, 8, 9, 5. Ao aplicarmos `inverte(v, 3, 3, 9)` obtemos no vetor a seguinte ordem: 1, 3, 4, 7, 6, 2, 8, 9, 5. Ao aplicarmos ao mesmo vetor inicial `inverte(v, 0, 4, 9)` obtemos: 2, 4, 3, 1, 6, 7, 8, 9, 5.

Sua tarefa neste exercício-programa é escrever algoritmos que recebam um vetor com n elementos e ordenam este vetor, utilizando a operação `inverte` descrita acima. Obviamente um jeito fácil de fazer isso é implementar o bubblesort (as trocas são chamadas do `inverte` com $k = 2$). Este algoritmo faz no máximo $O(n^2)$ chamadas da função inverte.

Uma aplicação deste exercício-programa ocorre em Biologia Computacional. Sabemos que o DNA é uma cadeia dupla de unidades básicas (bases nitrogenadas) e uma forma de tentar estabelecer graus de "parentesco" (ou proximidade) entre espécies é calcular o número mínimo de operações básicas (mutações, reversões, translocações, etc) que levam a seqüência genética de um indivíduo de uma espécie em um da outra. Uma dessas operações é a *reversão*. Uma reversão ocorre quando um "pedaço" do DNA se desprende e se "cola" novamente na seqüência, porém na ordem invertida. Esse fenômeno parece ser relativamente freqüente, e seu estudo é de interesse aos pesquisadores de filogenias. Podemos pensar no problema de transformar uma seqüência de DNA em outra através de "reversões" como o problema de ordenar uma seqüência através de aplicações da função `inverte`.

Este exercício-programa está aberto a diversas perguntas, que vocês poderão e deverão explorar em suas soluções:

- Escreva vários algoritmos diferentes de ordenação e compare o número de chamadas que cada um faz para ordenar uma seqüência qualquer. É possível determinar qual de seus algoritmos é mais eficiente?

- É possível escrever um algoritmo que ordena a seqüência utilizando **sempre** um número mínimo de chamadas à função `inverte`?

- Pense agora no problema em que o k é fixo (por exemplo, $k = 3$ ou $k = 4$). É possível ordenar um vetor com n elementos aplicando a função `inverte` com k fixo? Se é, em que casos? Para quais valores de k e n? Se não, por que? Coloque todas as suas conclusões em seu relatório.

- Suponha agora que a operação `inverte` funciona "circularmente" no fim do vetor.

  Exemplo: Aplicando ao vetor do exemplo acima `inverte(v, 7, 5, 9)` obtemos: 1, 5, 9, 2, 6, 7, 8, 4, 3.

  As mesmas perguntas acima se aplicam a esta nova versão do problema.

  Um joguinho baseado nesta idéia pode ser visto na sala 297A, em que $n = 20$ e $k = 4$. Quem quiser brincar... ;-)