MAC0338 - Entrega da lista 4

Exercício 8

Prove que o CountingSort *é estável.*

Vamos provar que o counting sort é estável construtivamente, usando como referência o código CountingSort(A,n) da aula 7, slide 7. Vamos focar nas linhas 7, 8 e 9:

Algorithm 1: CountingSort (linhas 7, 8 e 9)

```
para j ← n decrescendo até 1 faça
    B[C[A[j]]] ← A[j]
    C[A[j]] ← C[A[j]] − 1
```

Note que estamos percorrendo o vetor $A = [a_1, a_2, \ldots, a_n]$ de trás para frente e preencheremos um vetor $B = [b_1, b_2, \ldots, b_n]$. Também temos um vetor $C = [c_1, c_2, \ldots, c_k]$, que mostrará o índice que cada elemento do vetor A ocupará no vetor ordenado B.

Seja $a_x = a_y = m$ tal que $x < y$, ou seja, a_x está localizado antes de a_y no vetor A. Como estamos percorrendo o vetor A de trás para frente, então a_y será inserido no vetor B na posição c_m, e logo em seguida $c_m \leftarrow c_m - 1$. Note que os valores de C nunca crescem, apenas decrescem. Portanto, em seguida a_x será inserido no vetor B **pelo menos** na posição $c_m - 1$, que com certeza está localizado em uma posição anterior a a_y.

Logo, o algoritmo mantém a ordem relativa de elementos com chaves iguais durante o processo de ordenação. Isso prova que o counting sort é estável.

Exercício 14

Escreva um algoritmo que multiplica dois números complexos $a + bi$ e $c + di$ usando apenas três multiplicações reais. O seu algoritmo deve receber como entrada os números a, b, c, d e devolver os números $ac - bd$ (componente real do produto) e $ad + bc$ (componente imaginária do produto).

Algorithm 2: MultiplicaComplexos(a,b,c,d)

$x \leftarrow a \cdot c$
$y \leftarrow b \cdot d$
$z \leftarrow (a + b) \cdot (c + d)$
$real \leftarrow x - y$
$imaginario \leftarrow z - x - y$
return $real, imaginario$

Por definição, a parte real da multiplicação entre $a + bi$ e $c + di$ é $ac - bd$. Usamos uma multiplicação real para calcular ac e outra para bd.

A parte imaginária é $ad + bc$. Agora, note que

$$(a + b) \cdot (c + d) = ac + ad + bc + bd \implies ad + bc = (a + b) \cdot (c + d) - ac - bd,$$

e usamos mais uma multiplicação real para calcular $(a + b) \cdot (c + d)$.

Assim, calculamos a parte real e imaginária de uma multiplicação de complexos com 3 multiplicações reais.

Exercício 15

No SELECT-BFPRT*, os elementos do vetor são divididos em grupos de 5. O algoritmo continua linear se dividirmos os elementos em grupos de 7? E em grupos de 3? Prove sua resposta.*

Vamos analisar o consumo de tempo quando dividimos os elementos em grupos de 7.
Baseando-se no CLRS, o número de elementos maiores que x (mediana das medianas) nesse caso seria

$$4\Big(\Big\lceil\frac{1}{2}\Big\lceil\frac{n}{2}\Big\rceil\Big\rceil - 2\Big) \geq \frac{4n}{14} - 8.$$

Assim, a função de recorrência seria, para uma constante 63,

$$T(n) = \begin{cases} O(1) & \text{se } n < 63 \\ T(\lceil n/7\rceil) + T(6n/14 + 8) + O(n) & \text{se } n \geq 63 \end{cases}$$

Provaremos, assim como em CLRS, que $T(n) \leq cn$ para alguma constante c adequadamente grande e para todo $n > 0$. Também escolhemos uma constante a tal que $O(n) \geq an$ para todo $n > 0$. Logo, temos

$$\begin{aligned} T(n) &\leq c\lceil\ n/7\rceil + c(6n/14 + 8) + an \\ &\leq cn/7 + c + 6cn/14 + 8c + an \\ &= 8cn/14 + 9c + an \\ &= cn + (-6cn/14 + 9c + an) \end{aligned}$$

que é no máximo cn se $-6cn/14+9c+an \leq 0 \implies c \geq \frac{7a}{3}(n/(n-21))$ quando $n > 21$. Para $n = 63$, temos $n/(n-21) \leq 3$, sendo a constante $c \geq \frac{7a}{3} \cdot 3 = 7a$. Logo, o tempo de execução é linear quando dividimos os elementos em grupos de 7.

Agora, vamos analisar o consumo de tempo quando dividimos os elementos em grupos de 3.
Analogamente, o número de elementos maiores que x (mediana das medianas) nesse caso seria

$$2\Big(\Big\lceil\frac{1}{2}\Big\lceil\frac{n}{2}\Big\rceil\Big\rceil - 2\Big) \geq \frac{2n}{9} - 2.$$

Assim, a função de recorrência seria, para uma constante l,

$$T(n) = \begin{cases} O(1) & \text{se } n < l \\ T(\lceil n/3\rceil) + T(7n/9 + 2) + O(n) & \text{se } n \geq l \end{cases}$$

Provaremos, assim como em CLRS, que $T(n) \leq cn$ para alguma constante c adequadamente grande e para todo $n > 0$. Também escolhemos uma constante a tal que $O(n) \geq an$ para todo $n > 0$. Logo, temos

$$\begin{aligned} T(n) &\leq c\lceil\ n/3\rceil + c(7n/9 + 2) + an \\ &\leq cn/3 + c + 7cn/9 + 2c + an \\ &= 9cn/9 + 3c + an \\ &= cn + (3c + an) \end{aligned}$$

que é no máximo cn se $3c + an \leq 0 \implies c \leq -an/3$ para qualquer $n > 0$. Escolhendo $n \geq 3$, temos $-n/3 \leq 1$, e então $c \geq a$. Logo, o tempo de execução também é linear quando dividimos os elementos em grupos de 3.