Universidade Federal do Rio Grande do Norte

Instituto Metrópole Digital

Estruturas de Dados Básicas I • IMD0029

– 1ª Avaliação, 21 de setembro de 2017 –

Nome: _____ Matrícula: _____

LEIA ATENTAMENTE AS INSTRUÇÕES ABAIXO ANTES DE INICIAR A PROVA

- ⋆ *Esta é uma prova individual e sem consulta a qualquer material, analógico ou digital.*
- ⋆ *Você tem* **100 minutos** *para a realização da prova. As respostas devem ser fornecidas em caneta de cor azul ou preta. A nota máxima é obtida ao acertar 10 pontos. O valor de cada questão é fornecido junto com seu enunciado.*

Questão	1	2	3	4	5	6	7	8	Total
Pontuação Máxima	2.5	3.0	0.8	0.8	0.8	1.0	1.0	1.6	11.5
Pontos Obtidos									

∼ BOA PROVA ∼

1. [2.5 pts] Escreva uma função em C/C++ ou algoritmo em pseudocódigo que recebe um arranjo A de inteiros distintos em *ordem crescente* e retorna um índice i tal que $A[i] = i$ ou indique que tal índice não existe retornando -1. Por exemplo, se aplicado ao vetor abaixo o algoritmo deve retornar $i = 2$. Não serão consideradas soluções com complexidade temporal linear ou superior.

A:	-12	-8	2	17	28	34	35	46	57	61
	0	1	2	3	4	5	6	7	8	9

2. [3.0 pts] Escreva uma função em C/C++ ou algoritmo em pseudocódigo que recebe um vetor $A[0, \ldots, n-1]$ de inteiros e um índice $i \in [0, n-1]$, e rearranja os elementos de A de tal maneira que os elementos menores que $A[i]$ aparecem no início do vetor (em qualquer ordem), seguido dos elementos iguais a $A[i]$, seguido dos elementos maiores que $A[i]$ (em qualquer ordem). Por exemplo, se o vetor fornecido for $[-5, 7, 10, 7, 8, 9, 1, 7, -2, 3]$ com $i = 3$, uma possível saída seria $[-5, 3, -2, 1, 7, 7, 7, 9, 8, 10]$.

   Seu algoritmo deve ter complexidade temporal *linear* e complexidade espacial *constante*. **Prove a corretude** do seu algoritmo por invariante de laço. Solução correta mas com complexidade temporal superior a linear receberá, no máximo, 30% dos pontos da questão.

3. [0.8 pts] Sobre o algoritmo *intercala* ou *merge* do *mergesort*, indique a(s) opção(ões) verdadeira(s).

   (a) Precisa de memória extra para realizar a intercalação.
   (b) Combina dois vetores em um único vetor ordenado.
   (c) Tem complexidade $\Theta(log_2 n)$.
   (d) É responsável pela parte linear da complexidade do *mergesort*.
   (e) Intercala os menores elementos com os maiores.

4. [0.8 pts] Sobre o algoritmo *selection sort*, indique a(s) opção(ões) verdadeira(s):

   (a) É linear se o vetor está em ordem decrescente.
   (b) É linear se o vetor está em ordem crescente.
   (c) Realiza muitas trocas no pior caso.
   (d) Realiza poucas trocas no pior caso.

(e) Realiza a mesma quantidade de trocas, independente do caso.
(f) Insere elementos da parte não ordenada no parte ordenada do vetor.
(g) Insere sempre o menor elemento da parte não ordenada na parte ordenada do vetor.

5. [0.8 pts] Qual é a complexidade temporal de pior caso do algoritmo *insertion sort* quanto utilizamos a busca binária para determinar a posição de inserção do elemento na parte ordenada do vetor?

a) $O(n \log_2 n)$ b) $O(n \log_2 n^2)$ c) $O(n^2 \log_2 n)$ d) $O(n)$ e) $O(n^2)$

6. [1.0 pts] Em uma competição, quatro diferentes funções foram implementadas. Todas as funções usam um único laço e dentro do laço o mesmo conjunto de comandos são executados. Se $n > 0$ corresponde ao tamanho da entrada, assumindo que os comandos internos do laço não alteram o controle do laço, indique a complexidade temporal para cada item.

(a)
```
for( i=0 ; i < n ; ++i )
```
(b)
```
for( i=0 ; i < n ; i += 2 )
```
(c)
```
for( i=1 ; i < n ; i *= 2 )
```
(d)
```
for( i=n ; i > 1 ; i /= 2 )
```

7. [1.0 pts] Com relação a características de *instabilidade*, qual(is) afirmação(ões) é(são) correta(s)? Considere as versões apresentadas em sala de aula.

(a) Bubble sort é instável, porque quando a "bolha" maior sobe, elementos iguais podem ser trocados.
(b) Insertion sort é instável, porque não temos controle como elementos iguais serão inseridos na parte ordenada do vetor.
(c) Insertion sort é estável, visto que elementos iguais são inseridos na mesma ordem na parte ordenada.
(d) Selection sort é instável, porque a cada iteração o menor elemento da vez é trocado com o primeiro elemento da parte não ordenada.
(e) Selection sort é estável, porque sempre inserimos o menor elemento na parte ordenada, preservando a ordem relativa dos elementos iguais.
(f) Merge sort é instável, porque ele não é *in-place*, ou seja, ele ordena em um vetor externo.
(g) Quick sort é estável, visto que o partição não troca elementos iguais de lugar, mas apenas elementos menores ou maiores que o pivô.

8. [1.6 pt] Responda cada uma das questões com uma breve e convincente justificativa:

(a) Se provarmos que um algoritmo possui complexidade $O(n^2)$ no pior caso, é possível que o algoritmo seja $O(n)$ para alguma entrada?

(b) Se provarmos que um algoritmo possui complexidade $O(n^2)$ no pior caso, é possível que o algoritmo seja $O(n)$ para todas as entradas?

(c) Se provarmos que um algoritmo possui complexidade $\Theta(n^2)$, é possível que o algoritmo seja $O(n)$ para alguma entrada?

(d) Se provarmos que um problema possui complexidade $\Omega(n^2)$, é possível que um algoritmo que resolva tal problema seja $O(n \log_2 n)$ para alguma entrada?

(e) Se provarmos que um algoritmo possui complexidade $\Omega(n^2)$, é possível que o algoritmo seja $O(n)$ para alguma entrada?

(f) Podemos afirmar que $f(n) = \Theta(n^2)$, sendo que $f(n) = 100n^2$ quando n é par e $f(n) = 20n^2 - n \log_2 n$ quando n é impar?

(g) Se um problema possui complexidade $\Omega(n)$, é possível que um algoritmo que resolve o problema possua complexidade $\Theta(n \log_2 n)$?

(h) Se um algoritmo é considerado ótimo para resolver um problema de complexidade cúbica, este algoritmo pode ter complexidade no pior caso $O(n^2)$?

$\sim$ FIM $\sim$