## Universidade Federal do Rio Grande do Norte Instituto Metrópole Digital

Estruturas de Dados Básicas I ● IMD0029 - 1ª Avaliação, 6 de abril de 2017 -

Nome:		Matrícula:	

## A PROVA TEM A DURAÇÃO DE **100 MINUTOS**. 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 terá 100 minutos para a realização da prova. As respostas devem ser fornecidas em caneta de cor azul ou preta. Repostas escritas com grafite **não** serão consideradas. A nota máxima é obtida ao acertar 10 pontos. O valor de cada questão é fornecido junto com seu enunciado.
- \* Identifique com seu nome cada folha fornecida com esta prova. Ao final do período de realização de prova, devolva todas as folhas que lhe foram entregues.
- \* Perguntas sobre a avaliação não serão respondidas—a interpretação do enunciado faz parte da questão.

Questão	1	2	3	Multipla escolha	Total
Pontuação Máxima	2.5	3.0	2.5	4.5	12.5
Pontos Obtidos					

## $\sim$ BOA PROVA $\sim$

1. [2.5 pts] Um algoritmo óbvio para calcular  $x^n$  usa n-1 multiplicações. Um algoritmo recursivo pode ser melhor. Quanto  $n \leq 1$  temos o caso base. Caso contrário, se n for par, temos que  $x^n = x^{n/2} \cdot x^{n/2}$ , e se n for par, pa

Por exemplo, para calcular  $x^{62}$ , são feitas os seguintes cálculos (9 multiplicações):

$$x^{62} = (x^{31})^2; \ x^{31} = (x^{15})^2 \cdot x; \ x^{15} = (x^7)^2 \cdot x; \ x^7 = (x^3)^2 \cdot x; \ x^3 = (x^2)^2 \cdot x$$

já o algoritmo óbvio precisaria de 61 multiplicações.

Escreva uma função em C/C++ ou algoritmo em pseudocódigo **recursivo** que implementa esta ideia, ou seja, você precisa implementar uma função pow(a,b) que retorna  $a^b$ .

2. [3.0 pts] Um vetor A de comprimento n é considerado **ordenado ciclicamente** se o menor elemento do vetor está no índice i, e a sequência  $\langle A[i], A[i+1], \ldots, A[n-1], A[0], A[1], \ldots, A[i-1] \rangle$  está ordenada em *ordem crescente*, como ilustrado abaixo (menor elemento é 37, índice i=4).

Escreva uma função em C/C++ ou algoritmo em pseudocódigo com complexidade  $O(\log_2 n)$  que encontra e retorna a posição do menor elemento em um vetor ciclicamente ordenado. Assuma que todos os elementos são distintos. Para o exemplo acima, sua função deveria retornar a posição 4.

a) insertion

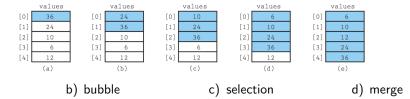
a) insertion

a) insertion

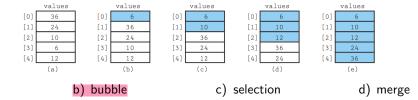
3. [2.5 pts] Escreva uma função em C/C++ ou algoritmo em pseudocódigo que **não use mais do que**  $\lceil 3n/2 \rceil - 2$  **comparações** para encontrar e retornar os elementos *máximo* e *mínimo* em um vetor com n>0 inteiros. Descreva como você contou as comparações da sua solução.

Nas questões 4 a 21 marque a(s) opção(ões) que você considera correta. Cada questão vale  $0.25~\rm pt.$ 

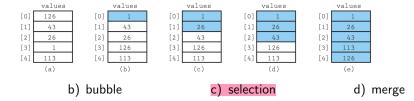
4. Qual é o algoritmo de ordenação representado na imagem abaixo



5. Qual é o algoritmo de ordenação representado na imagem abaixo



6. Qual é o algoritmo de ordenação representado na imagem abaixo



- 7. De que maneira a busca binária (BB) poderia ser útil para o selection sort?
  - (a) Para ordenar o arranjo antes de realizar a busca.
  - (b) De maneira alguma, são algoritmos com propósitos diferentes.
  - (c) Podemos usar a BB para achar o menor elemento para troca.
  - (d) Usamos a mesma estratégia da BB no selecion sort, mas ao invés de buscar fazemos a ordenação.
- 8. De que maneira a busca binária poderia ser útil para o insertion sort?
  - (a) De maneira alguma, são algoritmos com propósitos diferentes.
  - (b) Para localizar o menor elemento na parte ainda não processada.
  - (c) Para achar o local certo de inserção do item sendo processado.
  - (d) Para localizar o menor elemento na parte ordenada.
- 9. A complexidade temporal da busca linear quando um elemento não está na lista é:
  - (a) Proporcional ao tamanho da lista.
  - (b) Constante, independente da organização dos dados.
  - (c) Proporcional ao tamanho da metade da lista.
  - (d) Independente da organização dos dados.
  - (e) Constante, se a lista está ordenada.

- 10. Quantas comparações são executadas se aplicarmos o selection sort em um vetor de 100 elementos já ordenados? (100\*99)/2=4950
  - a) 9900
- b) 10000
- c) 4950
- d) 99
- e) 9801
- 11. Em linhas gerais, a busca binária por uma chave K segue a estratégia de:
  - (a) Dividir a lista em 2 metades e procurar K nas metades.
  - (b) Dividir o vetor em elementos ordenados e não ordenados.
  - (c) Ordenar a lista, para aplicar a busca linear.
  - (d) Particionar a lista em elementos  $\leq K$  e > K.
  - (e) Se elemento central não for K, buscar em uma das metades.
- 12. Merge sort (MS) é usado para ordenar 1000 elementos em um vetor. Qual afirmação relativa a complexidade temporal é verdadeira?
  - (a) MS é mais eficiente se o vetor está em ordem não decrescente.
  - (b) MS é menos eficiente se o vetor está em ordem aleatória.
  - (c) MS é mais eficiente se o vetor está em ordem não crescente.
  - (d) Em termos de eficiência, tanto faz a organização do vetor.
- 13. Suponha um vetor em ordem não decrescente. Qual algoritmo levará mais tempo para executar?
- O(n^2) a) Quick sort, com o 1º elemento como pivô.
- b) Bubble sort. O(n)

O(n) c) Insertion sort.

- d) Merge sort. Theta(nlogn)
- 14. Com relação a características de **instabilidade**, quais afirmações são corretas? 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.
- 15. O quick sort é um algoritmo **ótimo** pois sua complexidade é  $O(n \log_2 n)$ .
  - (a) Falso, sua complexidade é  $\Theta(n \log_2 n)$ .
  - (b) Falso, sua complexidade de pior caso é  $O(n^2)$ .
  - (c) Verdadeiro, é eficiente em todos os casos.
  - (d) Falso, pois sua complexidade no melhor caso é O(n).
- 16. O selection sort é recomendado se os elementos ordenados têm grande footprint de memória.
  - (a) Verdadeiro, pois ele realiza poucas trocas.
  - (b) Falso, pois o elemento não vai logo para a posição final.
  - (c) Verdadeiro, pois ele realiza poucas comparações.
  - (d) Falso, visto que ele é  $\Theta(n^2)$  independente da organização.

17. Seja $k$ um índice válido em um vetor qualquer de $n$ elementos. Qual a complexidade para acessar $k$ ?									
a) $O(k)$	<b>b)</b> $O(1)$	c) $O(n)$	d) $O(\log_2 n)$	e) $O(n^2)$	f) $O(\log_{10} n)$				
18. O selection sort no pior caso é melhor do que o insertion sort no pior caso.									
a) Sim.	b) Não.		c) São iguais.	d) De	epende dos dados.				
19. Durante a execução do quick sort o vetor é recursivamente dividido em partes iguais.									
(a) Não. A divisão igual só ocorre em vetores de tamanho par.									

- (b) Sim. Ele e o merge sort adotam a estratégia divisão-e-conquista.
- (c) Não. O tamanho das partes depende do pivô.
- (d) Sim. Por isso sua complexidade é  $O(n \log_2 n)$ .
- 20. O algoritmo intercala ou merge do merge sort...
  - (a) Tem complexidade  $O(log_2n)$ .
  - (b) Passa apenas uma vez em cada vetor para gerar um vetor ordenado.
  - (c) Recebe dois vetores e os retorna ordenados.
  - (d) Intercala os menores elementos com os maiores.
- 21. Sobre o algoritmo partição do quick sort...
  - (a) É responsável pela parte  $O(\log_2 n)$  da complexidade do quick sort.
  - (b) Compacta os elementos menores que o pivô no início.
  - (c) Posiciona o elemento central em seu local definitivo.
  - (d) Faz o quick sort ser quadrático se uma das metades é sempre vazia.

 $\sim$  FIM  $\sim$