# Algoritmos e Estruturas de Dados I

**Prof. Jurair Rosa**

PROVA - VALOR: 10 PONTOS

Centro Federal de Educação Tecnológica Celso Suckow da Fonseca

Turma: $3^o$ período Data: 27/10/2021

**Instruções:**

- A prova deve ser feita integralmente sem consulta.
- Essa prova tem duração de 100 minutos.
- É permitido responder às questões a lápis, desde que fique legível.
- A compreensão do enunciado faz parte da avaliação. Portanto, dúvidas relativas às questões **NÃO** serão respondidas.
- Ao término anexe a(s) folha(s) com as respostas no MS Teams.

**Questão 1)** O algoritmo a seguir recebe um vetor `v` de número inteiros e rearranja esse vetor de tal forma que seus elementos, ao final, estejam ordenados de forma crescente: (2,0 ptos)

```
1  void ordena(int *v, int n)
2  {
3    int i, j, chave;
4    for(i=1; i<n; i++)
5    {
6      chave = v[i];
7      j = i-1;
8      while (j >= 0 && v[j] < chave)
9      {
10       v[j-1] = v[j];
11       j = j-1;
12     }
13     v[j+1] = chave;
14   }
15 }
```

Considerando que nesse algoritmo há erros de lógica que devem ser corrigidos para que os elementos sejam ordenados de forma crescente, assinale a opção correta no que se refere às correções adequadas.

**a)** A linha 4 deve ser corrigida da seguinte forma: `for(i=1; i<n-1; i++)` e a linha 13, do seguinte modo: `v[j-1] = chave;`.
**b)** A linha 4 deve ser corrigida da seguinte forma: `for(i=1; i<n-1; i++)` e a linha 7, do seguinte modo: `j = i+1;`.
**c)** A linha 7 deve ser corrigida da seguinte forma: `j = i+1;` e a linha 8, do seguinte modo: `while(j >= 0 && v[j] > chave)`.
**d)** A linha 8 deve ser corrigida da seguinte forma: `while(j >= 0 && v[j] > chave)` e a linha 10, do seguinte modo: `v[j+1] = v[j];`.
**e)** A linha 10 deve ser corrigida da seguinte forma: `v[j+1] = v[j];` e a linha 13, do seguinte modo: `v[j - 1] = chave;`.

**Questão 2)** A sequência de Fibonacci é uma sequência de números inteiros que começa em 1, a que se segue 1, e na qual cada elemento subsequente é a soma dos dois elementos anteriores. A função `fib` a seguir calcula o $n$-ésimo elemento da sequência de Fibonacci: (2,0 ptos)

```
1  unsigned int fib(unsigned int n)
2  {
3    if (n < 2)
4      return 1;
5    return fib(n-2) + fib(n-1);
6  }
```

Considerando a implementação acima, avalie as afirmações a seguir:

I. A complexidade de tempo da função `fib` é exponencial no valor de $n$.

II. A complexidade de espaço da função `fib` é exponencial no valor de $n$.

III. É possível implementar uma versão iterativa da função `fib` com complexidade de tempo linear no valor de $n$ e complexidade de espaço constante.

É correto o que se afirma em:

**a)** I, apenas.
**b)** II, apenas.
**c)** I e III, apenas.
**d)** II e III, apenas.
**e)** I, II e III.

**Questão 3)** No desenvolvimento de um *software* que analisa bases de DNA, representadas pelas letras `A`, `C`, `G`, `T`, utilizou-se as estruturas de dados: pilha e fila. Considere que, se uma sequência representa uma pilha, o topo é o elemento mais à esquerda; e se uma sequência representa uma fila, a sua frente é o elemento mais à esquerda.

Analise o seguinte cenário: "a sequência inicial ficou armazenada na primeira estrutura de dados na seguinte ordem: `(A,G,T,C,A,G,T,T)`. Cada elemento foi retirado da primeira estrutura de dados e inserido na segunda estrutura de dados, e a sequência ficou armazenada na seguinte ordem: `(T,T,G,A,C,T,G,A)`. Finalmente, cada elemento foi retirado da segunda estrutura de dados e inserido na terceira estrutura de dados e a sequência ficou armazenada na seguinte ordem: `(T,T,G,A,C,T,G,A)`".

Qual a única sequência de estruturas de dados apresentadas a seguir pode ter sido usada no cenário descrito acima? (2,0 ptos)

**a)** Fila - Pilha - Fila.
**b)** Fila - Fila - Pilha.
**c)** Fila - Pilha - Pilha.
**d)** Pilha - Fila - Pilha.
**e)** Pilha - Pilha - Pilha.

**Questão 4)** As filas de prioridades (*heaps*) são estruturas de dados importantes no projeto de algoritmos. Em especial, *heaps* podem ser utilizados na recuperação de informação em grandes bases de dados constituídos por textos. Basicamente, para se exibir o resultado de uma consulta, os documentos recuperados são ordenados de acordo com a relevância presumida para o usuário. Uma consulta pode recuperar milhões de documentos que certamente não serão todos examinados. Na verdade, o usuário examina os primeiros $m$ documentos dos $n$ recuperados, em que $m$ é da ordem de algumas dezenas.

Considerando as características dos *heaps* e sua aplicação no problema descrito acima, avalie as seguintes afirmações: (2,0 ptos)

I. Uma vez que o *heap* é implementado como uma árvore binária de pesquisa essencialmente completa, o custo computacional para sua construção é $O(n\ log\ n)$.

II. A implementação de *heaps* utilizando-se vetores é eficiente em tempo de execução e em espaço de armazenamento, pois o pai de um elemento armazenado na posição $i$ se encontra armazenado na posição $2i + 1$.

III. O custo computacional para se recuperar de forma ordenada os $m$ documentos mais relevantes armazenados em um *heap* de tamanho $n$ é $O(m\ log\ n)$.

IV. Determinar o documento com maior valor de relevância armazenado em um *heap* tem custo computacional $O(1)$.

É correto apenas o que se afirma em:

**a)** I e II.
**b)** II e III.
**c)** III e IV.
**d)** I, II e IV.
**e)** I, III e IV.

**Questão 5)** Suponha que se queira pesquisar a chave 287 em uma árvore binária de pesquisa com chaves entre 1 e 1000. Durante uma pesquisa como essa, uma sequência de chaves é examinada. Cada sequência abaixo é uma suposta sequência de chaves examinadas em uma busca da chave 287: (2,0 ptos)

I. 7, 342, 199, 201, 310, 258, 287.

II. 110, 132, 133, 156, 289, 288, 287.

III. 252, 266, 271, 294, 295, 289, 287.

IV. 715, 112, 530, 249, 406, 234, 287.

É válido apenas o que se apresenta em:

**a)** I.
**b)** III.
**c)** I e II.
**d)** II e IV.
**e)** III e IV.