Questão **2**

Incorreto

Atingiu 0,00 de 1,00

Marcar questão

Considere a seguinte variação do algoritmo de ordenação *merge sort*:

```
void merge_sort(int * a, int ini, int fim){
  if(ini < fim){
    int q1 = ini + (fim - ini) / 4;

    merge_sort(a, ini, q1);
    merge_sort(a, q1 + 1, fim);
    merge(a, ini, q1, fim);
  }
}
```

Ao comparar essa variação com a versão clássica do algoritmo, é incorreto afirmar que (pode haver mais de uma afirmação incorreta):

- [ ] a. Essa variação difere da versão clássica do *merge sort* ao adotar uma nova estratégia de divisão do problema em subproblemas menores. Nesta variação do algoritmo, um vetor de tamanho **n** é dividido em subvetores de tamanho **n/4** e **3n/4**, que são ordenados recursivamente, e depois combinados com a função *merge*.
- [ ] b. A nova estratégia de divisão do problema em subproblemas não muda a complexidade assintótica da função *merge* (que continua linear), porém piora a complexidade do algoritmo *merge sort*, por apresentar uma árvore de recursão mais profunda que a observada na versão clássica.
- [x] c. A nova estratégia para divisão do problema em subproblemas não muda a complexidade assintótica nem da função *merge* (o fato de combinar dois subvetores de tamanhos distintos não é relevante, já que o que importa é a quantidade total de elementos combinados), e nem do algoritmo *merge sort* (apesar de a árvore de recursão ser mais profunda quando comparada à árvore da versão clássica, sua altura continua sendo logarítmica). ✖
- [x] d. A nova estratégia de divisão do problema em subproblemas piora tanto a complexidade assintótica da função *merge* (é necessário realizar mais comparações para combinar os dois subvetores ordenados), quanto do algoritmo *merge sort* por conta da árvore de recursão mais profunda em comparação com a árvore de recursão da versão clássica. ✔
- [x] e. Com a nova estratégia de divisão do problema em subproblemas, ao combinar um subvetor de tamanho **n/4** com outro de tamanho **3n/4**, a função *merge* passará a realizar, no cenário de **melhor caso**, apenas **n/4** comparações (contrastando com as **n/2** comparações necessárias, no cenário de melhor caso, para combinar dois subvetores de tamanho **n/2** na versão clássica do algoritmo). ✖

Sua resposta está incorreta.

As respostas corretas são:
A nova estratégia de divisão do problema em subproblemas não muda a complexidade assintótica da função *merge* (que continua linear), porém piora a complexidade do algoritmo *merge sort*, por apresentar uma árvore de recursão mais profunda que a observada na versão clássica.,

A nova estratégia de divisão do problema em subproblemas piora tanto a complexidade assintótica da função *merge* (é necessário realizar mais comparações para combinar os dois subvetores ordenados), quanto do algoritmo *merge sort* por conta da árvore de recursão mais profunda em comparação com a árvore de recursão da versão clássica.

Questão **16**

Incorreto

Atingiu 0,00 de 1,00

Marcar questão

Considerando, desta vez, a política de realocação descrita na **questão anteior**: qual o desperdício máximo de memória (isto é, a quantidade de memória alocada, mas não usada efetivamente para guardar informação útil) em uma lista sequencial cujo vetor (*array*) alocado tem, em um dado instante, **k posições** (com k > 100)? Assuma que desde a criação da lista, nenhuma remoção de elemento foi realizada, apenas inserções.

- a. k / 2 + 1
- b. k / 2 - 1
- c. k
- d. k - 1 ✖
- e. k / 2

A resposta correta é:
k / 2 - 1

Questão **25**

Incorreto

Atingiu 0,00 de 1,00

Marcar questão

Se os mesmos valores listados na **Questão 21** tivessem sido inseridos em uma árvore binária de busca convencional (isto é, uma estrutura não balanceada), na mesma ordem, qual seria a altura da árvore resultante?

- a. 6
- b. 5
- c. 7
- d. 3
- e. 4 ✖

A resposta correta é:
6