UFLA - CPA - BCC - 2021-1 - Prova 1 - 8/9 até 14/9/2021 às 16h40.

A avaliação é individual. Cada estudante deve enviar suas respostas. A correta organização das folhas de respostas é responsabilidade do aluno. A interpretação faz parte da prova. O nome do(a) estudante é item obrigatório no arquivo com as respostas. O arquivo com as respostas deve ser enviado exclusivamente na atividade criada para essa finalidade na página da disciplina no Campus Virtual. Não serão aceitas respostas enviadas com atraso. Há tempo bastante suficiente para fazer e enviar a resposta. Não serão aceitas respostas após o prazo por qualquer motivo, como atraso no envio das respostas motivado por eventuais falhas de conexões com a internet, falta de energia elétrica, congestionamento das linhas de comunicação, bem como outros fatores de ordem técnica que impossibilitem a conexão ou a transferência de dados. Essas eventualidades não serão aceitas como argumento para envio das respostas após o prazo. Recomenda-se que as respostas sejam encaminhadas com prudente antecedência.

Nome: Pedro Antônio de Souza

**Questão 1: 46 pontos. São 4,6 pontos para cada entrada a ser preenchida na Tabela 1.**

```
   Entrada: vetor a[ ] com n elementos.
   Saída: vetor a[ ] ordenado de forma crescente;
1  início
2  |  i ← 1;
3  |  repita
4  |  |  para j ← i + 1 até n faça
5  |  |  |  se a[i] < a[j] então
6  |  |  |  |  t ← a[i]; a[i] ← a[j]; a[j] ← t;
7  |  |  |  fim
8  |  |  fim
9  |  |  i ← i + 1;
10 |  até i ≥ n;
11 fim
```

**Algoritmo 1:** Ordenação 1.

Tabela 1: Preencha, em *notação assintótica*, a complexidade final de cada linha indicada abaixo do Algoritmo 1. Preencha com "-" se a linha não tiver execução. Também preencha as complexidades do algoritmo no último item.

| Linha | Pior caso | Melhor caso |
|---|---|---|
| 4 | $O(n^2)$ | $\Omega(n^2)$ |
| 5 | $O(n^2)$ | $\Omega(n^2)$ |
| 6 | $O(n^2)$ | - |
| 10 | $O(n)$ | $\Omega(n)$ |
| Complexidade do algoritmo | $O(n^2)$ | $\Omega(n^2)$ |

O escopo interno do **repita** executará $n$–1 vezes, assim como a verificação da linha 10.

Analisando a linhas 4, temos que:

- Para $i$=1, $j$ irá variar de 2 até $n$ e, portanto, a verificação do **para** será executada $n$ vezes.
- Para $i$=2, $j$ irá variar de 3 até $n$ e, portanto, a verificação do **para** será executada $n$–1 vezes.
- Para $i$=3, $j$ irá variar de 4 até $n$ e, portanto, a verificação do **para** será executada $n$–2 vezes.
- Para $i$=$n$–1, $j$ irá variar de $n$ até $n$ e, portanto, a verificação do **para** será executada 2 vezes.

Podemos escrever a contribuição total para o custo da execução da linha 4 através da expressão $\sum_{j=2}^{n} j = \frac{1}{2}(n^2+n-2)$.

A linha 5 será executada uma vez a menos que a linha 4.

Por fim, a linha 6 nunca será executada no melhor caso, já que a condição do **se** nunca é satisfeita. Por outro lado, no pior caso a condição é sempre satisfeita e o quantidade de execuções da linha 6 será $n^2$.

**Questão 2: 54 pontos. São três pontos para cada entrada a ser preenchida na Tabela 2.**

```
   Entrada: vetor a[ ] com n elementos.
   Saída: vetor a[ ] ordenado de forma crescente;
1  início
2  |  p ← 1;
3  |  enquanto p ≤ n faça
4  |  |  se p = 1 ∨ a[p] ≥ a[p − 1] então
5  |  |  |  p ← p + 1;
6  |  |  senão
7  |  |  |  t ← a[p];
8  |  |  |  a[p] ← a[p − 1];
9  |  |  |  a[p − 1] ← t;
10 |  |  |  p ← p − 1;
11 |  |  fim
12 |  fim
13 fim
```

**Algoritmo 2:** Ordenação 2.

Tabela 2: Preencha, em *notação assintótica*, a complexidade final de cada linha indicada abaixo do Algoritmo 2. Preencha com "-" se a linha não tiver execução. Também preencha as complexidades do algoritmo no último item.

| Linha | Pior caso | Melhor caso |
|---|---|---|
| 2 | O(1) | Ω(1) |
| 3 | O($n^2$) | Ω(n) |
| 4 | O($n^2$) | Ω(n) |
| 5 | O($n^2$) | Ω(n) |
| 7 | O($n^2$) | - |
| 8 | O($n^2$) | - |
| 9 | O($n^2$) | - |
| 10 | O($n^2$) | - |
| Complexidade do algoritmo | O($n^2$) | Ω(n) |

**Análise do pior caso**

A linha 2 será executada uma única vez.

Pela sequência de incrementos e decrementos na variável $p$ dentro do **enquanto**, para cada valor de $p$, todos os valores de índice menor são percorridos e depois o índice retorna ao $p$. Dessa forma, cada incremento de $p$ contribui com $2p$–1 execuções. Portanto, para a execução completa do algoritmo temos que a linha 3 será executada $\sum_{p=1}^{n}(2p-1)+1=n^2+1$ vezes.

A linha 4 será executada uma vez a menos que a verificação do enquanto, isto é $n^2$ vezes.

Para cada incremento de $p$, todas as posições de índice menor são percorridos e, posteriormente, o índice retorna ao $p$. Assim, o código entrará $p$ vezes no **se** para cada $p$. Portanto, a contribuição da linha 5 para o custo total do algoritmo pode ser descrita como $\sum_{p=1}^{n} p = \frac{(n^2+n)}{2}$.

Para cada incremento de $p$, todas as posições de índice menor são percorridos, isto é, o código entrará $p$–1 vezes no **senão** para cada $p$. Assim, a contribuição das linhas 7, 8, 9 e 10 para o custo total do algoritmo pode ser descrita como $\sum_{p=1}^{n} p-1 = \frac{(n^2-n)}{2}$.

**Análise do melhor caso**

A linha 2 será executada uma única vez.

A linha 3 será executada $n$+1 vezes, já que a variável $p$ será sempre incrementada como veremos a seguir.

A linha 4 será executada uma vez a menos que a verificação do **enquanto**, isto é $n$ vezes.

Como a condição da linha 4 sempre é satisfeita, a linha 5 será executada o mesmo número de vezes que sua linha antecessora, ou seja, $n$ vezes.

Já que a condição da linha 4 sempre é satisfeita, o algoritmo nunca executará as linhas 7, 8, 9 e 10.