CALCULANDO TEMPO DE EXECUÇÃO

Vamos calcular o tempo de execução do seguinte algoritmo:

```
INSERTION-SORT(A)
1  for j ← 2 to length[A]
2      do key ← A[j]
3         ▷ Insert A[j] into the sorted
                sequence A[1 . . j − 1].
4         i ← j − 1
5         while i > 0 and A[i] > key
6             do A[i + 1] ← A[i]
7                i ← i − 1
8         A[i + 1] ← key
```

Vamos calcular o tempo de execução do algoritmo considerando cada instrução como uma unidade de tempo. Vamos examinar linha por linha:

LINHA 1: `for j <- 2 to length[A]`

- Operações:

- Início do loop `for` (uma vez): 1 unidade de tempo.
- Comparação para continuar o loop (`j <= length[A]`): 1 unidade de tempo por iteração.
- Incremento de `j`: 1 unidade de tempo por iteração.
- Acesso a `length[A]`: 1 unidade de tempo por iteração.

<u>Total por iteração</u>: (1 + 1 + 1 + 1 = 4) unidades de tempo por iteração.

<u>Iterações totais</u>: Supondo que `length[A]` seja (n), temos (n-1) iterações.

<u>Total para a LINHA 1</u>: (4(n - 1)).

LINHA 2: `key <- A[j]`

- Operações:

- Acesso ao array `A[j]`: 1 unidade de tempo.
- Atribuição para `key`: 1 unidade de tempo.

Total por iteração: (1 + 1 = 2) unidades de tempo.

LINHA 3: `> Insert A[j] into the sorted`

- Essa linha parece ser um comentário ou uma descrição, sem uma operação explícita, então não há tempo associado.

LINHA 4: `i <- j -1`

- Operações:

- Subtração (`j - 1`): 1 unidade de tempo.
- Atribuição para `i`: 1 unidade de tempo.

Total por iteração: (1 + 1 = 2) unidades de tempo.

LINHA 5: `while i > 0 and A[i] > key`

- Operações por iteração do `while`:

- Comparação `i > 0`: 1 unidade de tempo.
- Comparação `A[i] > key`: 1 unidade de tempo.
- Acesso ao array `A[i]`: 1 unidade de tempo.

Total por iteração do `while`: (1 + 1 + 1 = 3) unidades de tempo.

O número de iterações do `while` depende de `j` e da posição correta de `A[j]`. No pior caso, o `while` é executado \(j-1\) vezes.

LINHA 6: `do A[i + 1] <- A[i]`

- Operações por iteração do `while`:

- Acesso ao array `A[i]`: 1 unidade de tempo.
- Cálculo `i + 1`: 1 unidade de tempo.
- Acesso ao array `A[i + 1]`: 1 unidade de tempo.
- Atribuição `A[i + 1] <- A[i]`: 1 unidade de tempo.

<u>Total por iteração do `while`</u>: (1 + 1 + 1 + 1 = 4) unidades de tempo.

LINHA 7: `i <- i – 1`

- Operações por iteração do `while`:

- Subtração `i - 1`: 1 unidade de tempo.
- Atribuição para `i`: 1 unidade de tempo.

<u>Total por iteração do `while`:</u> (1 + 1 = 2) unidades de tempo.

LINHA 8: `A[i + 1] <- key`

- Operações:

- Cálculo `i + 1`: 1 unidade de tempo.
- Acesso ao array `A[i + 1]`: 1 unidade de tempo.
- Atribuição `A[i + 1] <- key`: 1 unidade de tempo.

<u>Total por iteração:</u> (1 + 1 + 1 = 3) unidades de tempo.

Tempo Total de Execução

O tempo total de execução depende do número de iterações do `for` e do `while`. Para cada iteração do `for`, temos:

- LINHA 2: 2 unidades de tempo.
- LINHA 4: 2 unidades de tempo.
- LINHA 8: 3 unidades de tempo.
- LINHA 5-7: As linhas 5, 6 e 7 são executadas dentro do `while`, e o número de iterações depende de `j`.

No pior caso, quando o `while` executa (j-1) vezes, o tempo total do `while` para uma única iteração do `for` seria:

- LINHA 5: (3(j-1)) unidades de tempo.
- LINHA 6: (4(j-1)) unidades de tempo.
- LINHA 7: (2(j-1)) unidades de tempo.

Somando tudo, o tempo total no pior caso seria:

$$\mathrm{T(n)} = \sum_{j=2}^{n} (2 + 2 + 3 + 3(j-1) + 4(j-1) + 2(j-1))$$

Simplificando:

$$\mathrm{T(n)} = \sum_{j=2}^{n} (7 - 9(j-1))$$

$$\mathrm{T(n)} = \sum_{j=2}^{n} (9j - 2)$$

$$\mathrm{T}(\mathrm{n}) = 9\sum_{j=2}^{n} j - 2(n-1)$$

$$\mathrm{T}(\mathrm{n}) = 9\sum_{j=2}^{n} j - 2(n-1)$$

$$T(n) = 9\left(\frac{(n(n+1)}{2} - 1\right) - 2(\mathrm{n}-1)$$

$$T(n) = \frac{9n(n+1)}{2} - 9 - 2n + 2$$

$$T(n) = \frac{9n^2 + 9n}{2} - 2n - 7$$