
CÓDIGO:

```
1  for j ← 2 to length[A]
2      key ← A[j]
3      ▷ Inserir A[j] na sequência ordenada A[1..j-1]
4      i ← j - 1
5      while i > 0 and A[i] > key
6          A[i + 1] ← A[i]
7          i ← i - 1
8      A[i + 1] ← key
```

ANÁLISE POR LINHA:

Linha 1: `for j ← 2 to length[A]`

- Executa (n - 1) vezes.
- Cada iteração envolve: incremento + verificação de condição → **1 operação por iteração**.
- **Total:** `(n - 1)t`

Linha 2: `key ← A[j]`

- Acesso (`A[j]`) = 1 operação, atribuição = 1 operação.
- **Total por iteração:** `2t`
- **Total geral:** `2(n - 1)t`

Linha 4: `i ← j - 1`

- Subtração (1 operação) + atribuição (1 operação).
- **Total por iteração:** `2t`
- **Total geral:** `2(n - 1)t`

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

- Verificação da condição:
  - `i > 0` = 1 operação
  - `A[i] > key` = acesso + comparação = 2 operações
- Total por iteração do `while`: `3t`
- **Número de iterações depende do número de elementos maiores que `key`, chamaremos isso de `c_j`**
- **Total para todos j:** `3t * ∑(j=2 até n) c_j`

Linha 6: `A[i + 1] ← A[i]`

- Acesso `A[i]` (1) + soma `i + 1` (1) + atribuição (1) → **3t**
- **Total para todos j:** `3t * ∑(j=2 até n) c_j`

Linha 7: `i ← i - 1`

- Subtração + atribuição → `2t`
- **Total para todos j:** `2t * ∑(j=2 até n) c_j`

Linha 8: `A[i + 1] ← key`

- Soma + atribuição → `2t`
- **Executado 1 vez por iteração do `for`** → `2(n - 1)t`

FUNÇÃO DE TEMPO TOTAL T(n)

Somando tudo:

```
T(n) = (n - 1)t                ← linha 1
     + 2(n - 1)t               ← linha 2
     + 2(n - 1)t               ← linha 4
     + 2(n - 1)t               ← linha 8
     + (3 + 3 + 2)t * ∑ c_j    ← linhas 5-7
```

Simplificando:

```
T(n) = 7(n - 1)t + 8t * ∑(j=2 até n) c_j
```

CASOS ESPECIAIS

Melhor caso (vetor já ordenado):

- `c_j = 0` (nenhum elemento maior que `key`)
- `∑ c_j = 0`
- **T(n) = 7(n - 1)t → Complexidade: O(n)**

Pior caso (vetor em ordem decrescente):

- `c_j = j - 1` → `∑(j=2 até n)(j - 1) = (n - 1)n / 2`
- **T(n) = 7(n - 1)t + 8t * (n(n - 1)/2)**
- **T(n) = 7(n - 1)t + 4t(n(n - 1)) → Complexidade: O(n²)**

RESUMO FINAL:

- **Melhor caso:** vetor ordenado → $O(n)$
- **Pior caso:** vetor reverso → $O(n^2)$
- A contagem leva em conta todas as operações elementares (atribuição, comparação, acesso etc).