Codigo do Algoritmo:

```
for j <- 2 ate comprimento[A]:
    chave <- A[j]
    i <- j - 1
    enquanto i > 0 e A[i] > chave:
        A[i + 1] <- A[i]
        i <- i - 1
    A[i + 1] <- chave
```

Avaliacao Linha a Linha:

Linha 1 - Laco For:
- Executado (n - 1) vezes.
- Cada ciclo realiza 1 comparacao e 1 incremento.
- Total: (n - 1)t

Linha 2 - Atribuicao da Chave:
- Leitura de A[j] + atribuicao -> 2 operacoes.
- Total: 2(n - 1)t

Linha 4 - Inicializacao de i:
- Calculo (j - 1) + atribuicao -> 2 operacoes.
- Total: 2(n - 1)t

Linha 5 - Condicao do While:
- Verificacao de i > 0 (1 op)
- Comparacao A[i] > chave -> 2 ops (acesso + comparacao)
- Total por repeticao do while: 3t
- Se repetir c_j vezes: 3t × c_j

Linha 6 - Deslocamento:
- Acesso, soma e atribuicao -> 3t por repeticao
- Total: 3t × c_j

Linha 7 - Decremento de i:
- Subtracao + atribuicao -> 2t por repeticao
- Total: 2t × c_j

Linha 8 - Insercao da Chave:
- Soma + atribuicao -> 2 operacoes
- Executada 1 vez por iteracao do laco -> 2(n - 1)t

Tempo Total T(n):

Somando todos os componentes:

```
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 a 7

T(n) = 7(n - 1)t + 8t × c_j

Casos Especificos:

- Melhor Caso (vetor ja ordenado):
- Nenhum deslocamento (c_j = 0)
- c_j = 0
- T(n) = 7(n - 1)t => O(n)

- Pior Caso (vetor decrescente):
- Maximo de deslocamentos: c_j = j - 1
- (j=2 ate n)(j - 1) = (n(n - 1)) / 2
- T(n) = 7(n - 1)t + 4t(n(n - 1))
- Complexidade: O(n2)

Conclusao:

- Melhor cenario: algoritmo linear -> O(n)
- Pior cenario: crescimento quadratico -> O(n2)
- A contagem considera todas as instrucoes basicas (atribuicoes, acessos, comparacoes etc.)