Para: pior caso, melhor caso e caso médio

Análise de Algoritmos

Considerações sobre o algoritmo

```
/* ordenação */

Bubble Sort (V, n)
{
  para i ← 1 até n faça
    para j ← 1 até n faça
        se V[ j] > V [ j + 1] então
           temp ← V [j]
           V[ j] ← V [ j + 1]
           V[ j + 1] ← temp
}
```

Existem várias possibilidades para os valores de entrada:

- Podemos ter um vetor de tamanho 10, completamente desordenado;
- Podemos ter um vetor de tamanho 10, mais ou menos ordenado;
- Podemos ter um vetor de tamanho 10, completamente ordenado;
- Podemos ter um vetor de um outro tamanho qualquer, mas que esteja nas mesmas combinações de valores internos, i.e., situações de ordenação.

A questão é: O algoritmo terá o mesmo comportamento em todos os casos?

- Para responder essa pergunta, devemos pensar em três situações possíveis: para o **pior caso**, para o **melhor caso** e então, para o **caso médio**

Bubble Sort para o *Pior caso*

/ ordenação */*

```
Bubble Sort (V, n)
{
(1) para i ← 1 até n faça
(2)    para j ← 1 até n faça
(3)          se V[ j] > V [ j + 1] então
(4)             temp ← V [j]
(5)             V[ j] ← V [ j + 1]
(6)             V[ j + 1] ← temp
}
```

Começando pelo laço mais interno:

O tempo para incrementar o índice e avaliar a condição de término do laço da linha 2 é O(1).

O tempo para avaliar a condição do comando de decisão é O(1).

Os comandos de atribuição das linhas (4), (5) e (6) levam tempo constante, i.e., O(1).

Aqui temos que pensar um pouco ... os comandos destas linhas somente serão executados se a condição da linha 3 for verdadeira.

Portanto, a pior situação (em termos de esforço) é termos que realizar os próximos comandos sempre!

Assim, o tempo para executar uma única vez o laço composto por (2), (3), (4), (5) e (6) é O(max(1,1,1,1,1)) = O(1).

Como o número de iterações do laço é n então o tempo gasto é:

$n.O(1) = O(n \,.\, 1) = O(n)$

O tempo para incrementar o índice e avaliar a condição de término do laço mais externo é O(1).

O tempo para executar uma única vez o laço composto pelo trecho de (1) à (6) é $O(\max(1,n)) = O(n)$.

Como o número de iterações do laço é n então:

$n.O(n) = O(n. \, n) = O(n^2)$

Bubble Sort para o *Melhor caso*

/ ordenação */*

```
Bubble Sort (V, n)
{
(1) para i ← 1 até n faça
(2)   para j ← 1 até n faça
(3)       se V[ j] > V [ j + 1] então
(4)           temp ← V [j]
(5)           V[ j] ← V [ j + 1]
(6)           V[ j + 1] ← temp
}
```

Começando pelo laço mais interno:

- O tempo para incrementar o índice e avaliar a condição de término do laço da linha 2 é O(1).
- O tempo para avaliar a condição do comando de decisão é O(1).
- Os comandos de atribuição das linhas (4), (5) e (6) levam tempo constante, i.e., O(1).

Novamente, aqui temos que pensar um pouco ... os comandos destas linhas somente serão executados se a condição da linha 3 for verdadeira.

Portanto, a melhor situação (em termos de esforço) seria nunca termos que realizar os próximos comandos

Assim, o tempo para executar uma única vez o laço composto por (2) e (3) será: O(max(1,1)) = O(1).

Como o número de iterações do laço é n então o tempo gasto é:

n.O(1) = O(n . 1) = O(n)

O tempo para incrementar o índice e avaliar a condição de término do laço mais externo é O(1).

O tempo para executar uma única vez o laço composto pelo trecho de (1) à (6) é O(max(1,n)) = O(n).

Como o número de iterações do laço é n então:

n.O(n) = O(n. n) = O(n^2)

Conclusões para o Bubble Sort

- Tempo para o *pior caso*: $O(n^2)$
- Tempo para o *melhor caso*: $O(n^2)$
- Intuitivamente, sabemos que o tempo para o caso médio também será $O(n^2)$, uma vez que não há alterações significativas na execução em função das variações na entrada.
- Portanto, podemos dizer que esse algoritmo tem um desempenho constante, e será sempre $O(n^2)$

Bubble Sort Modificado (*Pior Caso*)

/ ordenação */*

```
Bubble Sort Mod(V, n)
{
( 1) trocou ← verdadeiro
( 2) i ← 1
( 3) enquanto (i <= n) e (trocou) faça
( 4)    trocou ← falso
( 5)    para j ← 1 até n faça
( 6)    se ( V[ j ] > V[ j + 1] ) então
( 7)        temp ← V[ j ]
( 8)        V[ j ] ← V[ j + 1]
( 9)        V[ j + 1] ← temp
(10)        trocou ← verdadeiro
(11)    i ← i + 1
}
```

Começando pelo laço mais interno:

O tempo para incrementar o índice e avaliar a condição de término do laço da linha 5 é O(1).

O tempo para avaliar a condição do comando de decisão da linha (6) também é O(1).

Os comandos de atribuição das linhas (7), (8), (9) e (10) individualmente, levam tempo constante.

Os comandos destas linhas somente serão executados se a condição da linha (6) for verdadeira. Assim, a pior situação (em termos de esforço) é termos que realizar esses comandos sempre.

O tempo para executar uma única vez o trecho de (5) á (10) é:

$O(max(1,1,1,1,1,1)) = O(1)$.

Como o número de repetições do laço é n vezes, temos: $n.O(1) = O(n)$

Bubble Sort Modificado (*Pior Caso*)

```
/* ordenação */

Bubble Sort Mod(V, n)
{
( 1)  trocou ← verdadeiro
( 2)  i ← 1
( 3)  enquanto (i <= n) e (trocou) faça
( 4)     trocou ← falso
( 5)     para j ← 1 até n faça
( 6)     se ( V[ j ] > V[ j + 1] ) então
( 7)          temp ← V[ j ]
( 8)          V[ j ] ← V[ j + 1]
( 9)          V[ j + 1] ← temp
(10)          trocou ← verdadeiro
(11)     i ← i + 1
}
```

Passando agora para o laço mais externo:

O tempo para avaliar a condição do comando de repetição da linha (3) é O(1).

Os comandos de atribuição das linhas (4) e (11), individualmente, tem tempo constante.

O tempo para executar uma única vez o trecho de (3) à (11) é:

$O(max(1,1,n,1)) = O(n)$.

Como assumimos que a pior situação seria ter que realizar os comandos das linhas (7), (8), (9) e (10) sempre, o número de repetições do laço da linha (3) dependerá apenas da primeira condição, haja visto, que trocou terá valor igual a verdadeiro em todas as iterações.

Assim, o laço será executado *n* vezes e teremos:

$n.O(n) = O(n^2)$

Portanto a complexidade para o algoritmo será:

$O(máx(1,1, n^2)) = O(n^2)$

Bubble Sort Modificado (*Melhor Caso*)

/ ordenação */*

```
Bubble Sort Mod(V, n)
{
( 1)  trocou ← verdadeiro
( 2)  i ← 1
( 3)  enquanto (i <= n) e (trocou) faça
( 4)     trocou ← falso
( 5)     para j ← 1 até n faça
( 6)     se ( V[ j ] > V[ j + 1] ) então
( 7)          temp ← V[ j ]
( 8)          V[ j ] ← V[ j + 1]
( 9)          V[ j + 1] ← temp
(10)          trocou ← verdadeiro
(11)     i ← i + 1
}
```

Começando pelo laço mais interno:

O tempo para incrementar o índice e avaliar a condição de término do laço da linha 5 é O(1).

O tempo para avaliar a condição do comando de decisão da linha (6) também é O(1).

Os comandos de atribuição das linhas (7), (8), (9) e (10) individualmente, levam tempo constante.

Os comandos destas linhas somente serão executados se a condição da linha (6) for verdadeira. Assim, a melhor situação (em termos de esforço) seria nunca ter que realizar esses comandos.

O tempo para executar uma única vez o trecho de (5) á (10) é:

$O(\max(\underbrace{1}_{5},\underbrace{1}_{6})) = O(1)$.

Como o número de repetições do laço é *n* vezes, temos: $n.O(1) = O(n)$

Bubble Sort Modificado (*Melhor Caso*)

```
/* ordenação */

Bubble Sort Mod(V, n)
{
( 1) trocou ← verdadeiro
( 2) i ← 1
( 3) enquanto (i <= n) e (trocou) faça
( 4)    trocou ← falso
( 5)    para j ← 1 até n faça
( 6)    se ( V[ j ] > V[ j + 1] ) então
( 7)         temp ← V[ j ]
( 8)         V[ j ] ← V[ j + 1]
( 9)         V[ j + 1] ← temp
(10)         trocou ← verdadeiro
(11)    i ← i + 1
}
```

Passando agora para o laço mais externo:

O tempo para avaliar a condição do comando de repetição da linha (3) é O(1).

Os comandos de atribuição das linhas (4), (10) e (11), individualmente, tem tempo constante.

O tempo para executar uma única vez o trecho de (3) à (11) é:

$O(\max(1,1,n,1)) = O(n)$.

Como assumimos que a melhor situação seria não ter que realizar os comandos das linhas (7), (8), (9) e (10).

Desta maneira a variável trocou terá valor igual a falso sempre que completarmos a primeira execução do laço. Portanto, esse laço será executado uma única vez:

$1.O(n) = O(n)$

Assim a complexidade para o algoritmo será:

$O(\text{máx}(\underbrace{1}_{1},\underbrace{1}_{2}, \underbrace{n}_{3 \text{ à } 11})) = O(n)$

Conclusões para o Bubble Sort Mod.

- Tempo para o *pior caso*: $O(n^2)$
- Tempo para o *melhor caso*: $O(n)$
- O caso médio depende da distribuição dos casos particulares. Se cada caso ocorrer com igual freqüência, teremos: $\frac{1}{2}n^2 + \frac{1}{2}n$

  Assim, a complexidade média será: $O(n^2)$