GBC034 - Análise Matemática de Algoritmos

Marcelo K. Albertini

6 de Junho de 2013

Aula de hoje

Aula passada vimos:

- Análise empírica de algoritmos

Nesta aula veremos:

- Análise matemática de algoritmos

Definição de um problema

Exemplo: problema de ordenação crescente

- Entrada: uma sequência de n números $a_1, a_2, a_3, \ldots, a_n$
- Saída: uma reordenação $b_1, b_2, \ldots, b_n$, tal que $b_i < b_j$ se $i < j$

A solução: um algoritmo

- correto: se para cada instância de entrada, o algoritmo conclui com a saída correta
- se o é algoritmo é correto então ele resolve o problema

Análise empírica de algoritmos

Custo = memória + tempo

- Quanto espaço de memória o algoritmo vai consumir?
- Quanto tempo levar o algoritmo?

Como atribuir custo/estimar tempo de operações

- uniforme - todas as operações tem o mesmo custo fixo T
  - atribuições de valores, cópias de variáveis, chamadas de funções
  - operações lógico-aritméticas
- proporcional - operações tem custo proporcional ao número de bits sendo operados

Análise empírica de algoritmos

Custo = memória + tempo

- Quanto espaço de memória o algoritmo vai consumir?
- Quanto tempo levar o algoritmo?

Como atribuir custo/estimar tempo de operações

- uniforme - todas as operações tem o mesmo custo fixo T
  - atribuições de valores, cópias de variáveis, chamadas de funções
  - operações lógico-aritméticas
- proporcional - operações tem custo proporcional ao número de bits sendo operados

Avaliação empírica

A avaliação empírica permite

- avaliação do desempenho em uma determinada configuração de computador/linguagem
- considera custos não aparentes: custo de alocação de memória, sobrecarga de chamada de funções
- comparar computadores
- comparar linguagens
- reconhecer modelo de desempenho para casos difíceis de avaliação matemática de algoritmos

Avaliação empírica

Dificuldades

- necessário implementar o algoritmo
- resultados dependem do computador utilizado e de eventos ocorridos no momento de avaliação
- difícil de generalizar

Avaliação matemática

A avaliação matemática

- uso de algoritmo em um “computador” idealizado
- simplificação para considerar somente os custos dominantes

Avaliação de custos em um algoritmo

```java
// Entrada: inteiro positivo "n"
// Saída: imprimir multiplicações de 0 a "n"
if (n > 10) {
    System.out.println("Isto pode demorar...");
}
for (int i = 0; i < n; i++) {
    for (int j = 0; j < i; j++) {
        System.out.println(i * j);
    }
}
```

Avaliação de custos em um algoritmo

```
// Entrada: inteiro positivo "n"
// Saída: imprimir multiplicações de 0 a "n"
if (n > 10) {  // 1 vez, custo c1
 System.out.println("Isto pode demorar..."); c2
}
for (int i = 0; i < n; i++) { // n vezes, c3 + nc1 + nc4
 for (int j = 0; j < i; j++) { // ∑_{j=1}^{n} j vezes,
     c3 + nc1 + nc4
  System.out.println(i * j); // c2 + c5
 }
}
```

Custos

comparação c_1, impressão c_2, atribuição c_3, incremento c_4, multiplicação c_5

Avaliação de custos em um algoritmo

```
// Entrada: inteiro positivo "n"
// Saída: imprimir multiplicações de 0 a "n"
if (n > 10) {  // 1 vez, custo c1
 System.out.println("Isto pode demorar..."); c2
}
for (int i = 0; i < n; i++) { // n vezes, c3 + nc1 + nc4
 for (int j = 0; j < i; j++) { // ∑_{j=1}^{n} j vezes,
     c3 + nc1 + nc4
  System.out.println(i * j); // c2 + c5
 }
}
```

Custos

comparação c_1, impressão c_2, atribuição c_3, incremento c_4, multiplicação c_5

Avaliação de custos em um algoritmo

```
// Entrada: inteiro positivo "n"
// Saída: imprimir multiplicações de 0 a "n"
if (n > 10) {   // 1 vez, custo c1
 System.out.println("Isto pode demorar..."); c2
}
for (int i = 0; i < n; i++) { // n vezes, c3 + nc1 + nc4
 for (int j = 0; j < i; j++) { // ∑_{j=1}^{n} j vezes,
    c3 + nc1 + nc4
  System.out.println(i * j); // c2 + c5
 }
}
```

Custos

comparação c_1, impressão c_2, atribuição c_3, incremento c_4, multiplicação c_5

Avaliação de custos em um algoritmo

```
// Entrada: inteiro positivo "n"
// Saída: imprimir multiplicações de 0 a "n"
if (n > 10) {   // 1 vez, custo c1
 System.out.println("Isto pode demorar..."); c2
}
for (int i = 0; i < n; i++) { // n vezes, c3 + nc1 + nc4
 for (int j = 0; j < i; j++) { // ∑_{j=1}^{n} j vezes, c3 + nc1 + nc4
  System.out.println(i * j); // c2 + c5
 }
}
```

Custos

comparação c_1, impressão c_2, atribuição c_3, incremento c_4, multiplicação c_5

Avaliação de custos em um algoritmo

```
// Entrada: inteiro positivo "n"
// Saída: imprimir multiplicações de 0 a "n"
if (n > 10) {   // 1 vez, custo $c_1$
 System.out.println("Isto pode demorar..."); $c_2$
}
for (int i = 0; i < n; i++) { // n vezes, $c_3 + nc_1 + nc_4$
  for (int j = 0; j < i; j++) { // $\sum_{j=1}^{n} j$ vezes,
      $c_3 + nc_1 + nc_4$
    System.out.println(i * j); // $c_2 + c_5$
  }
}
```

Custos

comparação c_1, impressão c_2, atribuição c_3, incremento c_4, multiplicação c_5

Avaliação de custos em um algoritmo

```
// Entrada: inteiro positivo "n"
// Saída: imprimir multiplicações de 0 a "n"
if (n > 10) {   // 1 vez, custo c1
 System.out.println("Isto pode demorar..."); c2
}
for (int i = 0; i < n; i++) { // n vezes, c3 + nc1 + nc4
 for (int j = 0; j < i; j++) { // ∑_{j=1}^{n} j vezes,
     c3 + nc1 + nc4
  System.out.println(i * j); // c2 + c5
 }
}
```

Custos

comparação c_1, impressão c_2, atribuição c_3, incremento c_4, multiplicação c_5

Avaliação de custos em um algoritmo

```
// Entrada: inteiro positivo "n"
// Saída: imprimir multiplicações de 0 a "n"
if (n > 10) {  // 1 vez, custo c1
 System.out.println("Isto pode demorar..."); c2
}
for (int i = 0; i < n; i++) { // n vezes, c3 + nc1 + nc4
 for (int j = 0; j < i; j++) { // ∑_{j=1}^{n} j vezes, c3 + nc1 + nc4
  System.out.println(i * j); // c2 + c5
 }
}
```

Custos

comparação c_1, impressão c_2, atribuição c_3, incremento c_4, multiplicação c_5

Avaliação de custos em um algoritmo

```
// Entrada: inteiro positivo "n"
// Saída: imprimir multiplicações de 0 a "n"
if (n > 10) {  // 1 vez, custo $c_1$
 System.out.println("Isto pode demorar..."); $c_2$
}
for (int i = 0; i < n; i++) { // n vezes, $c_3 + nc_1 + nc_4$
 for (int j = 0; j < i; j++) { // $\sum_{j=1}^{n} j$ vezes,
    $c_3 + nc_1 + nc_4$
  System.out.println(i * j); // $c_2 + c_5$
 }
}
```

Custos

comparação c_1, impressão c_2, atribuição c_3, incremento c_4, multiplicação c_5

Avaliação de custos em um algoritmo

```
// Entrada: inteiro positivo "n"
// Saída: imprimir multiplicações de 0 a "n"
if (n > 10) {  // 1 vez, custo $c_1$
 System.out.println("Isto pode demorar..."); $c_2$
}
for (int i = 0; i < n; i++) { // n vezes, $c_3 + nc_1 + nc_4$
 for (int j = 0; j < i; j++) { // $\sum_{j=1}^{n} j$ vezes,
    $c_3 + nc_1 + nc_4$
  System.out.println(i * j); // $c_2 + c_5$
 }
}
```

Custos

comparação c_1, impressão c_2, atribuição c_3, incremento c_4, multiplicação c_5

Avaliação de custos em um algoritmo

```
// Entrada: inteiro positivo "n"
// Saída: imprimir multiplicações de 0 a "n"
if (n > 10) { // 1 vez, custo c1
 System.out.println("Isto pode demorar..."); c2
}
for (int i = 0; i < n; i++) { // n vezes, c3 + nc1 + nc4
 for (int j = 0; j < i; j++) { // m = ∑_{j=1}^{n} j vezes,
        c3 + mc1 + mc4
  System.out.println(i * j); // c2 + c5
 }
}
```

Custos: pior caso – quando o algoritmo executar tudo

$$T(n) = c_1 + c_2 + (c_3 + nc_1 + nc_4) + (c_3 + mc_1 + mc_4) + m * (c_2 + c_5)$$

Avaliação de custos em um algoritmo

```
// Entrada: inteiro positivo "n"
// Saída: imprimir multiplicações de 0 a "n"
if (n > 10) { // 1 vez, custo c1
 System.out.println("Isto pode demorar..."); c2
}
for (int i = 0; i < n; i++) { // n vezes, c3 + nc1 + nc4
 for (int j = 0; j < i; j++) { // m = ∑_{j=1}^{n} j vezes,
     c3 + mc1 + mc4
  System.out.println(i * j); // c2 + c5
 }
}
```

Custos: pior caso – quando o algoritmo executar tudo

$$T(n) = c_1 + c_2 + (c_3 + nc_1 + nc_4) + (c_3 + mc_1 + mc_4) + m * (c_2 + c_5)$$

Avaliação de custos em um algoritmo

```
// Entrada: inteiro positivo "n"
// Saída: imprimir multiplicações de 0 a "n"
if (n > 10) { // 1 vez, custo c_1
 System.out.println("Isto pode demorar..."); c_2
}
for (int i = 0; i < n; i++) { // n vezes, c_3 + nc_1 + nc_4
 for (int j = 0; j < i; j++) { // m = sum_{j=1}^{n} j vezes,
      c_3 + mc_1 + mc_4
  System.out.println(i * j); // c_2 + c_5
 }
}
```

Custos: pior caso – quando o algoritmo executar tudo

$$T(n) = c_1 + c_2 + (c_3 + nc_1 + nc_4) + (c_3 + mc_1 + mc_4) + m * (c_2 + c_5)$$

Avaliação de custos em um algoritmo

```
// Entrada: inteiro positivo "n"
// Saída: imprimir multiplicações de 0 a "n"
if (n > 10) { // 1 vez, custo c_1
 System.out.println("Isto pode demorar..."); c_2
}
for (int i = 0; i < n; i++) { // n vezes, c_3 + nc_1 + nc_4
 for (int j = 0; j < i; j++) { // m = \sum_{j=1}^{n} j vezes, c_3 + mc_1 + mc_4
  System.out.println(i * j); // c_2 + c_5
 }
}
```

Custos: pior caso – quando o algoritmo executar tudo

$$T(n) = c_1 + c_2 + (c_3 + nc_1 + nc_4) + (c_3 + mc_1 + mc_4) + m * (c_2 + c_5)$$

Avaliação de custos em um algoritmo

```
// Entrada: inteiro positivo "n"
// Saída: imprimir multiplicações de 0 a "n"
if (n > 10) { // 1 vez, custo c1
 System.out.println("Isto pode demorar..."); c2
}
for (int i = 0; i < n; i++) { // n vezes, c3 + nc1 + nc4
 for (int j = 0; j < i; j++) { // m = ∑_{j=1}^{n} j vezes,
     c3 + mc1 + mc4
  System.out.println(i * j); // c2 + c5
 }
}
```

Custos: pior caso – quando o algoritmo executar tudo

$$T(n) = c_1 + c_2 + (c_3 + nc_1 + nc_4) + (c_3 + mc_1 + mc_4) + m * (c_2 + c_5)$$

Avaliação de custos em um algoritmo

```
// Entrada: inteiro positivo "n"
// Saída: imprimir multiplicações de 0 a "n"
if (n > 10) { // 1 vez, custo c_1
 System.out.println("Isto pode demorar..."); c_2
}
for (int i = 0; i < n; i++) { // n vezes,  c_3 + nc_1 + nc_4
 for (int j = 0; j < i; j++) { //  m = \sum_{j=1}^{n} j vezes,
      c_3 + mc_1 + mc_4
  System.out.println(i * j); // c_2 + c_5
 }
}
```

Custos: pior caso – quando o algoritmo executar tudo

$$T(n) = c_1 + c_2 + (c_3 + nc_1 + nc_4) + (c_3 + mc_1 + mc_4) + m * (c_2 + c_5)$$

Avaliação de custos em um algoritmo

Custos: pior caso – quando o algoritmo executar tudo

- $T(n) = c_1 + c_2 + (c_3 + nc_1 + nc_4) + (c_3 + mc_1 + mc_4) + m * (c_2 + c_5))$
- se todos custos forem considerados iguais a "c":
- $T(n) = c + c + (c + nc + nc) + (c + mc + mc) + m * (c + c))$
- $T(n) = 2c + (c + 2nc) + (c + 2mc) + 2mc$
- $T(n) = 4c + 2nc + 4mc$

Avaliação de custos em um algoritmo

Custos: pior caso – quando o algoritmo executar tudo

- $T(n) = c_1 + c_2 + (c_3 + nc_1 + nc_4) + (c_3 + mc_1 + mc_4) + m * (c_2 + c_5))$
- se todos custos forem considerados iguais a "c":
- $T(n) = c + c + (c + nc + nc) + (c + mc + mc) + m * (c + c))$
- $T(n) = 2c + (c + 2nc) + (c + 2mc) + 2mc$
- $T(n) = 4c + 2nc + 4mc$

Avaliação de custos em um algoritmo

Custos: pior caso – quando o algoritmo executar tudo

- $T(n) = c_1 + c_2 + (c_3 + nc_1 + nc_4) + (c_3 + mc_1 + mc_4) + m * (c_2 + c_5))$
- se todos custos forem considerados iguais a "c":
- $T(n) = c + c + (c + nc + nc) + (c + mc + mc) + m * (c + c))$
- $T(n) = 2c + (c + 2nc) + (c + 2mc) + 2mc$
- $T(n) = 4c + 2nc + 4mc$

Avaliação de custos em um algoritmo

Custos: pior caso – quando o algoritmo executar tudo

- $T(n) = c_1 + c_2 + (c_3 + nc_1 + nc_4) + (c_3 + mc_1 + mc_4) + m * (c_2 + c_5))$
- se todos custos forem considerados iguais a "c":
- $T(n) = c + c + (c + nc + nc) + (c + mc + mc) + m * (c + c))$
- $T(n) = 2c + (c + 2nc) + (c + 2mc) + 2mc$
- $T(n) = 4c + 2nc + 4mc$

Avaliação de custos em um algoritmo

Custos: pior caso – quando o algoritmo executar tudo

- $T(n) = c_1 + c_2 + (c_3 + nc_1 + nc_4) + (c_3 + mc_1 + mc_4) + m * (c_2 + c_5))$
- se todos custos forem considerados iguais a "c":
- $T(n) = c + c + (c + nc + nc) + (c + mc + mc) + m * (c + c))$
- $T(n) = 2c + (c + 2nc) + (c + 2mc) + 2mc$
- $T(n) = 4c + 2nc + 4mc$

Avaliação de custos em um algoritmo

Custo do algoritmo é

$T(n) = 4c + 2nc + 4mc$

Quanto é m ?

$m = \sum_{j=1}^{n} j =$

- $1 + 2 + 3 + 4 + 5 + \ldots + (n-3) + (n-2) + (n-1) + n =$
- $(n+1) + (2 + (n-1)) + (3 + (n-2)) + \ldots + (n+1)$
- Se n é um número par, são $n/2$ somas de $(n+1)$.
  - $m = \frac{n}{2}(n+1)$
- Se n é ímpar, são $(n-1)/2$ somas de $(n+1)$ mais metade de $(n+1)$.
  - $m = \frac{n-1}{2}(n+1) + (n+1)/2 = \frac{n}{2}(n+1)$

Custo do algoritmo é

$T(n) = 4c + 2nc + 4(\frac{n}{2}(n+1))c$

Avaliação de custos em um algoritmo

Custo do algoritmo é

$T(n) = 4c + 2nc + 4mc$

Quanto é m ?

$m = \sum_{j=1}^{n} j =$

- $1 + 2 + 3 + 4 + 5 + \ldots + (n-3) + (n-2) + (n-1) + n =$
- $(n+1) + (2 + (n-1)) + (3 + (n-2)) + \ldots + (n+1)$
- Se n é um número par, são $n/2$ somas de $(n+1)$.
  - $m = \frac{n}{2}(n+1)$
- Se n é ímpar, são $(n-1)/2$ somas de $(n+1)$ mais metade de $(n+1)$.
  - $m = \frac{n-1}{2}(n+1) + (n+1)/2 = \frac{n}{2}(n+1)$

Custo do algoritmo é

$T(n) = 4c + 2nc + 4(\frac{n}{2}(n+1))c$

Avaliação de custos em um algoritmo

Custo do algoritmo é

$T(n) = 4c + 2nc + 4mc$

Quanto é m ?

$m = \sum_{j=1}^{n} j =$

- $1 + 2 + 3 + 4 + 5 + \ldots + (n-3) + (n-2) + (n-1) + n =$
- $(n+1) + (2 + (n-1)) + (3 + (n-2)) + \ldots + (n+1)$
- Se n é um número par, são $n/2$ somas de $(n+1)$.
  - $m = \frac{n}{2}(n+1)$
- Se n é ímpar, são $(n-1)/2$ somas de $(n+1)$ mais metade de $(n+1)$.
  - $m = \frac{n-1}{2}(n+1) + (n+1)/2 = \frac{n}{2}(n+1)$

Custo do algoritmo é

$T(n) = 4c + 2nc + 4(\frac{n}{2}(n+1))c$

Avaliação de custos em um algoritmo

Custo do algoritmo é

$T(n) = 4c + 2nc + 4mc$

Quanto é m ?

$m = \sum_{j=1}^{n} j =$

- $1 + 2 + 3 + 4 + 5 + \ldots + (n-3) + (n-2) + (n-1) + n =$
- $(n+1) + (2 + (n-1)) + (3 + (n-2)) + \ldots + (n+1)$
- Se n é um número par, são $n/2$ somas de $(n+1)$.
    - $m = \frac{n}{2}(n+1)$
- Se n é ímpar, são $(n-1)/2$ somas de $(n+1)$ mais metade de $(n+1)$.
    - $m = \frac{n-1}{2}(n+1) + (n+1)/2 = \frac{n}{2}(n+1)$

Custo do algoritmo é

$T(n) = 4c + 2nc + 4(\frac{n}{2}(n+1))c$

Avaliação de custos em um algoritmo

Custo do algoritmo é

$T(n) = 4c + 2nc + 4mc$

Quanto é m ?

$m = \sum_{j=1}^{n} j =$

- $1 + 2 + 3 + 4 + 5 + \ldots + (n-3) + (n-2) + (n-1) + n =$
- $(n+1) + (2 + (n-1)) + (3 + (n-2)) + \ldots + (n+1)$
- Se n é um número par, são $n/2$ somas de $(n+1)$.
  - $m = \frac{n}{2}(n+1)$
- Se n é ímpar, são $(n-1)/2$ somas de $(n+1)$ mais metade de $(n+1)$.
  - $m = \frac{n-1}{2}(n+1) + (n+1)/2 = \frac{n}{2}(n+1)$

Custo do algoritmo é

$T(n) = 4c + 2nc + 4(\frac{n}{2}(n+1))c$

Avaliação de custos em um algoritmo

Custo do algoritmo é

$T(n) = 4c + 2nc + 4mc$

Quanto é m ?

$m = \sum_{j=1}^{n} j =$

- $1 + 2 + 3 + 4 + 5 + \ldots + (n-3) + (n-2) + (n-1) + n =$
- $(n+1) + (2 + (n-1)) + (3 + (n-2)) + \ldots + (n+1)$
- Se n é um número par, são $n/2$ somas de $(n+1)$.
  - $m = \frac{n}{2}(n+1)$
- Se n é ímpar, são $(n-1)/2$ somas de $(n+1)$ mais metade de $(n+1)$.
  - $m = \frac{n-1}{2}(n+1) + (n+1)/2 = \frac{n}{2}(n+1)$

Custo do algoritmo é

$T(n) = 4c + 2nc + 4(\frac{n}{2}(n+1))c$

Avaliação de custos em um algoritmo

Custo do algoritmo é

$T(n) = 4c + 2nc + 4mc$

Quanto é m ?

$m = \sum_{j=1}^{n} j =$

- $1 + 2 + 3 + 4 + 5 + \ldots + (n-3) + (n-2) + (n-1) + n =$
- $(n+1) + (2 + (n-1)) + (3 + (n-2)) + \ldots + (n+1)$
- Se n é um número par, são $n/2$ somas de $(n+1)$.
  - $m = \frac{n}{2}(n+1)$
- Se n é ímpar, são $(n-1)/2$ somas de $(n+1)$ mais metade de $(n+1)$.
  - $m = \frac{n-1}{2}(n+1) + (n+1)/2 = \frac{n}{2}(n+1)$

Custo do algoritmo é

$T(n) = 4c + 2nc + 4(\frac{n}{2}(n+1))c$

Avaliação de custos em um algoritmo

Custo do algoritmo é

$T(n) = 4c + 2nc + 4mc$

Quanto é m ?

$m = \sum_{j=1}^{n} j =$

- $1 + 2 + 3 + 4 + 5 + \ldots + (n-3) + (n-2) + (n-1) + n =$
- $(n+1) + (2 + (n-1)) + (3 + (n-2)) + \ldots + (n+1)$
- Se n é um número par, são $n/2$ somas de $(n+1)$.
  - $m = \frac{n}{2}(n+1)$
- Se n é ímpar, são $(n-1)/2$ somas de $(n+1)$ mais metade de $(n+1)$.
  - $m = \frac{n-1}{2}(n+1) + (n+1)/2 = \frac{n}{2}(n+1)$

Custo do algoritmo é

$T(n) = 4c + 2nc + 4(\frac{n}{2}(n+1))c$

Avaliação de custos em um algoritmo

Custo do algoritmo é

$T(n) = 4c + 2nc + 4mc$

Quanto é m ?

$m = \sum_{j=1}^{n} j =$

- $1 + 2 + 3 + 4 + 5 + \ldots + (n-3) + (n-2) + (n-1) + n =$
- $(n+1) + (2 + (n-1)) + (3 + (n-2)) + \ldots + (n+1)$
- Se n é um número par, são $n/2$ somas de $(n+1)$.
  - $m = \frac{n}{2}(n+1)$
- Se n é ímpar, são $(n-1)/2$ somas de $(n+1)$ mais metade de $(n+1)$.
  - $m = \frac{n-1}{2}(n+1) + (n+1)/2 = \frac{n}{2}(n+1)$

Custo do algoritmo é

$T(n) = 4c + 2nc + 4(\frac{n}{2}(n+1))c$

Avaliação de custos em um algoritmo

Custo do algoritmo é

- $T(n) = 4c + 2nc + 4(\frac{n}{2}(n+1))c$
- $T(n) = 4c + 2nc + 2n(n+1)c$
- $T(n) = 4c + 2nc + 2n^2c + 2nc$
- $T(n) = 4c + 4cn + 2cn^2$

Qual é o custo dominante do algoritmo?

Avaliação de custos em um algoritmo

Custo do algoritmo é

- $T(n) = 4c + 2nc + 4(\frac{n}{2}(n+1))c$
- $T(n) = 4c + 2nc + 2n(n+1)c$
- $T(n) = 4c + 2nc + 2n^2c + 2nc$
- $T(n) = 4c + 4cn + 2cn^2$

Qual é o custo dominante do algoritmo?

Avaliação de custos em um algoritmo

Custo do algoritmo é

- $T(n) = 4c + 2nc + 4(\frac{n}{2}(n+1))c$
- $T(n) = 4c + 2nc + 2n(n+1)c$
- $T(n) = 4c + 2nc + 2n^2c + 2nc$
- $T(n) = 4c + 4cn + 2cn^2$

Qual é o custo dominante do algoritmo?

Avaliação de custos em um algoritmo

Custo do algoritmo é

- $T(n) = 4c + 2nc + 4(\frac{n}{2}(n+1))c$
- $T(n) = 4c + 2nc + 2n(n+1)c$
- $T(n) = 4c + 2nc + 2n^2c + 2nc$
- $T(n) = 4c + 4cn + 2cn^2$

Qual é o custo dominante do algoritmo?

Complexidade de tempo

Custo do algoritmo de multiplicação de números

A função que descreve o custo do nosso algoritmo é $T(n) = 4c + 4cn + 2cn^2$.

Complexidade de tempo

Para descrever a complexidade usamos somente o custo dominante: $2cn^2$.

Porquê?

Existe um número $n = a$ a partir do qual o custo $2ca^2$ é sempre superior a $4ca$, tornando-o pouco importante.

Ordem de crescimento

Dizemos que a complexidade de *pior caso* do nosso algoritmo é $O(n^2)$.

Complexidade de tempo

Custo do algoritmo de multiplicação de números

A função que descreve o custo do nosso algoritmo é $T(n) = 4c + 4cn + 2cn^2$.

Complexidade de tempo

Para descrever a complexidade usamos somente o custo dominante: $2cn^2$.

Porquê?

Existe um número $n = a$ a partir do qual o custo $2ca^2$ é sempre superior a $4ca$, tornando-o pouco importante.

Ordem de crescimento

Dizemos que a complexidade de *pior caso* do nosso algoritmo é $O(n^2)$.

Complexidade de tempo

Custo do algoritmo de multiplicação de números

A função que descreve o custo do nosso algoritmo é $T(n) = 4c + 4cn + 2cn^2$.

Complexidade de tempo

Para descrever a complexidade usamos somente o custo dominante: $2cn^2$.

Porquê?

Existe um número $n = a$ a partir do qual o custo $2ca^2$ é sempre superior a $4ca$, tornando-o pouco importante.

Ordem de crescimento

Dizemos que a complexidade de *pior caso* do nosso algoritmo é $O(n^2)$.

Complexidade de tempo

Custo do algoritmo de multiplicação de números

A função que descreve o custo do nosso algoritmo é $T(n) = 4c + 4cn + 2cn^2$.

Complexidade de tempo

Para descrever a complexidade usamos somente o custo dominante: $2cn^2$.

Porquê?

Existe um número $n = a$ a partir do qual o custo $2ca^2$ é sempre superior a $4ca$, tornando-o pouco importante.

Ordem de crescimento

Dizemos que a complexidade de *pior caso* do nosso algoritmo é $O(n^2)$.

Exercício: maior valor de um vetor

Qual é a função que descreve o custo e a ordem de crescimento de pior caso do seguinte algoritmo?

```
// Entrada: vetor com números inteiros
// Saída: maior valor do vetor
int max (int a[]){
    int i, maior;
    maior = a[0];

    for (i = 1; i < a.length; i++){
        if (maior < a[i]) {
            maior = a[i];
        }
    }
    return(maior);
}
```

Exercício: busca número em vetor

Qual é a função que descreve o custo e a ordem de crescimento de pior caso do seguinte algoritmo?

```
// Entrada: a[]- vetor ordenado de números inteiros, n
   - número sendo buscado, inf - limite inferior de
   busca no vetor a[], sup - limite superior de busca
   no vetor a[]
// Saída: false se não encontrou, true caso contrário
public static boolean busca(int a[], int n, int inf,
   int sup){
 if (inf > sup) {
  return(false);
 } else {
   int m = (int) (inf+sup)/2;
   if (a[m] == n)
    return(true);

   if (n> a[m]) {
    return(busca(a, n, m+1, sup));
   } else {
    return(busca(a, n, inf, m-1));
   }
  }
 }
}
```