AED II - Notações de Complexidade

Luca Ribeiro Schettino Regne

1

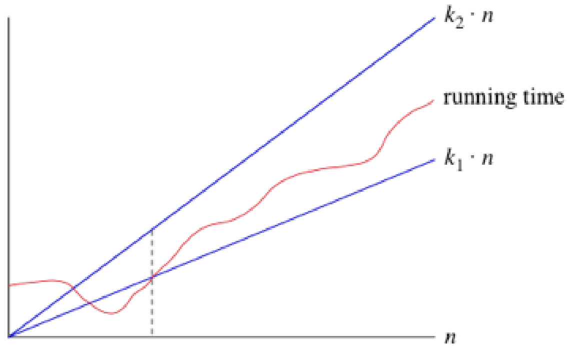
1. Notação assintótica

A notação assintótica é um método utilizado para quantificar e definir, através do comportamento de algoritmos, o seu nível de complexidade. Para simplificação, essa notação desconsidera constantes aditivas e mutiplicativas e leva em consideração apenas a **taxa de crescimento**, isto é, o quão rápido uma função cresce a medida que se aumenta o tamanho da entrada.

1.1. Big-Θ

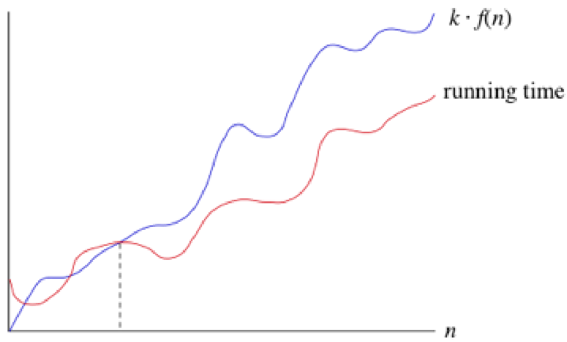
Na definição de $\Theta(n)$ são utilizados limites lineares para o tempo de execução, sendo que, quando n fica grande o bastante, o tempo de execução é de pelo menos $k_1 \cdot n$ e, no máximo, de $k_2 \cdot n$ para quaisquer constantes k_1 e k_2. Sendo assim temos o seguinte cenário $\Theta(n)$:

$$k_2 \cdot n > \Theta(n) > k_1 \cdot n$$



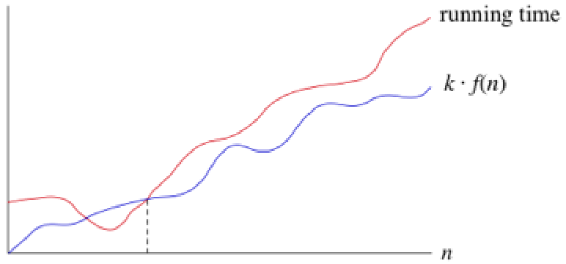
1.2. Big-O

Diferente da $\Theta(n)$ na notação chamado Big O, ou Grande-O, é utilizado quando se deseja encontrar limites assintóticos superiores, ou seja, é traçado apenas o pior caso.



1.3. Big-Ω

De maneira contrária ao Big-O a notação Big-Ω defini apenas um limite inferior. Então podemos concluir que, seja um tempo de execução $\Omega(f(n))$, para um n suficientemente grande, ele será ao menos $k \cdot f(n)$ para uma constante qualquer.



2. Exercícios Resolvidos

2.1. Exercício Resolvido 1

```
...
a--;
a -= 3;
a = 1 - 2;
```

$O(1)$
$\Theta(1)$
$\Omega(1)$

2.2. Exercício Resolvido 2

```
...
if(a + 5 < b + 3)
  i++;
  ++b;
  a += 3;
} else {
  j++;
}
```

$O(1)$
$\Theta(1)$
$\Omega(1)$

2.3. Exercício Resolvido 3

```
...
if(a + 5 < b + 3 || c + 1 < d + 3)
  i++;
  ++b;
  a += 3;
} else {
  j++;
}
```

$O(1)$
$\Theta(1)$
$\Omega(1)$

2.4. Exercício Resolvido 4

```
...
for(int i = 0; i < 4; i++){
  a--;
}
```

$O(1)$
$\Theta(1)$
$\Omega(1)$

2.5. Exercício Resolvido 5

```
...
for(int i = 0; i < n; i++){
  a--;
  b--;
}
```

$O(n)$
$\Theta(n^2)$
$\Omega(1)$

2.6. Exercício Resolvido 6

```
...
int i =0, b = 10;
while(i < 3){
  i++;
  b--;
}
```

$O(1)$
$\Theta(1)$
$\Omega(1)$

2.7. Exercício Resolvido 7

```
...
for(int i = 3; i < n; i++){
  a--;
}
```

$O(n)$
$\Theta(n)$
$\Omega(n)$

2.8. Exercício Resolvido 8

```
int a = 10;
...
for(int i = 0; i < 3; i++){
  for(int j = 0; j < 2;j++){
    a--;
  }
}
```

$O(1)$
$\Theta(1)$
$\Omega(1)$

2.9. Exercício Resolvido 9

Calcule o número de multiplicações que o código abaixo realiza:

```
...
for(int i = n; i < 0; i /= 2){
  a *= 2;
}
```

$O(\log_2 n)$
$\Theta(\log_2 n)$
$\Omega(1)$

2.10. Exercício Resolvido 10

```
i = 0;

while (i < n){
  i++;a--;
  b--;
  c--;
}

for (i = 0;  i < n; i++){
  for (j = 0;  j < n; j++){
    a--;
    b--;
  }
}
```

$O(n^3)$
$\Theta(n^3)$
$\Omega(n)$

2.11. Exercício Resolvido 11

Encontre o menor valor em um array de inteiros

```
int min = array[0];

for (int i = 1; i < n; i++){
  if (min > array[i]){
    min = array[i];
  }
}
```

$O(n)$
$\Theta(n)$
$\Omega(1)$

3. Exercícios

3.1. Exercício 1

a) $2^0 = 1$
b) $2^1 = 2$
c) $2^2 = 4$
d) $2^3 = 8$
e) $2^4 = 16$
f) $2^5 = 32$
g) $2^6 = 64$
h) $2^7 = 128$
i) $2^8 = 256$
j) $2^9 = 512$
k) $2^{10} = 1024$
l) $2^{11} = 2048$

$O(1)$ - $\Theta(1)$ - $\Omega(1)$

3.2. Exercício 2

a) $\log 2048 = 11$
b) $\log 1024 = 10$
c) $\log 512 = 9$
d) $\log 256 = 8$
e) $\log 128 = 7$
f) $\log 64 = 6$
j) $\log 32 = 5$
h) $\log 16 = 4$
i) $\log 8 = 3$
j) $\log 4 = 2$
k) $\log 2 = 1$
l) $\log 1 = 0$

$O(1)$ - $\Theta(1)$ - $\Omega(1)$

3.3. Exercício 3

a) $\lceil 4,01 \rceil = 5$
b) $\lfloor 4,01 \rfloor = 4$
c) $\lceil 4,99 \rceil = 5$
d) $\lfloor 4,99 \rfloor = 4$
e) $\lceil \log 16 \rceil = 4$
f) $\lfloor \log 16 \rfloor = 4$
g) $\log 17 = 4.08$
h) $\lceil \log 17 \rceil = 4$
i) $\lfloor \log 17 \rfloor = 5$
j) $\log 15 = 3.90$
k) $\lceil \log 15 \rceil = 3$
l) $\lfloor \log 15 \rfloor = 4$

$O(1)$ - $\Theta(1)$ - $\Omega(1)$

3.4. Exercício 4

a) $f(n) = n$
$O(n) - \Theta(n) - \Omega(1)$

b) $f(n) = n^2$
$O(n^2) - \Theta(n^2) - \Omega(n)$

c) $f(n) = n^3$
$O(n^3) - \Theta(n^3) - \Omega(n^2)$

d) $f(n) = sqrt(n)$
$O(n^{1/2}) - \Theta(n^{1/2}) - \Omega(n^{1/4})$

e) $f(n) = \lg n = \log_2 n$
$O(\log_2 n) - \Theta(\log_2 n) - \Omega(1)$

f) $f(n) = 3n^2 + 5n - 3$
$O(n^2) - \Theta(n^2) - \Omega(n)$

g) $f(n) = -3n^2 + 5n - 3$
$O(n^2) - \Theta(n^2) - \Omega(n)$

h) $f(n) = |-n^2|$
$O(n^2) - \Theta(n^2) - \Omega(n)$

i) $f(n) = 5n^4 + 2n^2$
$O(n^4) - \Theta(n^4) - \Omega(n^3)$

j) $f(n) = n * \lg(n)$
$O(n \cdot lg(n)) - \Theta(n \cdot lg(n)) - \Omega(lg(n))$

3.5. Exercício 5

```
...
int i = 10;
while(i >= 7){
  i--;
}
```

$O(1)$
$\Theta(1)$
$\Omega(1)$

3.6. Exercício 6

```
...
for(int i = 5; i >- 2; i--){
  a--;
}
```

$O(1)$
$\Theta(1)$
$\Omega(1)$

3.7. Exercício 7

```
...
for (int i = 0; i < 5; i++){
  if (i % 2 == 0){
    a--;
    b--;
  } else{
    c--;
  }
}
```

$O(1)$
$\Theta(1)$
$\Omega(1)$

3.8. Exercício 8

```
...
for (int i = 0; i < n; i++){
  for (int j = 0; j < n; j++){
    a--;
  }
}
```

$O(n^2)$
$\Theta(n^2)$
$\Omega(n)$

3.9. Exercício 9

```
...
int i = 1, b = 10;
while (i > 0){
  b--;
  i = i >> 1;
}
i = 0;
while (i < 15){
  b--;
  i += 2;
}
```

$O(1)$
$\Theta(1)$
$\Omega(1)$

3.10. Exercício 10

Calcule o número de multiplicações que o código abaixo realiza:

```
...
for (int i = 0; i < n; i++)
  for (int j = 0; j < n - 3; j++)
    a *= 2;
```

$O(n^2)$
$\Theta(n^2)$
$\Omega(n^2)$

3.11. Exercício 11

Calcule o número de multiplicações que o código abaixo realiza:

```
...
for (int i = n - 7; i >= 1; i--)
  for (int j = 0; j < n; j++)
    a *= 2;
```

$O(n)$
$\Theta(n)$
$\Omega(1)$

3.12. Exercício 12

Calcule o número de multiplicações que o código abaixo realiza:

```
...
for (int i = n; i > 0; i /= 2)
    a *= 2;
```

$O(n)$
$\Theta(n)$
$\Omega(1)$

3.13. Exercício 13

Calcule o número de multiplicações que o código abaixo realiza:

```
...
for (int i = n+4; i > 0; i >>= 1)
  a *= 2;
```

$O(n)$
$\Theta(n)$
$\Omega(1)$

3.14. Exercício 14

Calcule o número de multiplicações que o código abaixo realiza:

```
...
for (int i = n - 7; i >= 1; i--)
  for (int j = n - 7; j >= 1; j--)
    a *= 2;
```

$O(n^2)$
$\Theta(n^2)$
$\Omega(n)$

3.15. Exercício 15

Calcule o número de multiplicações que o código abaixo realiza:

```
...
for (int i = n + 1; i > 0; i /= 2)
  a *= 2;
```

$O(lg_n)$
$\Theta(lg_n)$
$\Omega(1)$

3.16. Exercício 16

Calcule o número de multiplicações que o código abaixo realiza:

```
...
for (int i = n; i > 1; i /= 2)
  a *= 2
```

$O(lg_n)$
$\Theta(lg_n)$
$\Omega(1)$

3.17. Exercício 17

Calcule o número de multiplicações que o código abaixo realiza:

```
...
for (int i = 1; i < n; i *= 2)
  a *= 2;
```

$O(2^n)$
$\Theta(2^n)$
$\Omega(n^2)$

3.18. Exercício 18

Calcule o número de multiplicações que o código abaixo realiza:

```
...
for (int i = 1; i <= n; i*= 2)
  a *= 2;
```

$O(2^n)$
$\Theta(2^n)$
$\Omega(n^2)$