Fundamentos de Projeto e Análise de Algoritmos

Técnicas para Análise de Algoritmos

Prof. João Caram

PUC Minas Bacharelado em Engenharia de Software

TÉCNICAS PARA ANÁLISE DE ALGORITMOS[1]

- Análise e determinação da função de complexidade de algoritmos *iterativos*.

- Análise e determinação da função de complexidade de algoritmos *recursivos*.

1- ZIVIANI, Nivio; BOTELHO, Fabiano Cupertino. **Projeto de algoritmos : com** implementações em Java e C++

Técnicas para análise de algoritmos[1]

- Não há técnica geral de análise de algoritmos: estudo caso-a-caso.

- Uso de princípios básicos de Aho, Hopcroft e Ullman(1983).

1- ZIVIANI, Nivio; BOTELHO, Fabiano Cupertino. **Projeto de algoritmos :** com implementações em Java e C++

Técnicas para análise de algoritmos

- Operações simples:
  - Leitura, escrita ou atribuição;
  - Operações aritméticas;
  - Operações lógicas;

- Cada uma delas tem custo 1 a cada execução.

Técnicas para análise de algoritmos

- Custo de uma sequência:
  - soma dos custos de cada operação da sequência.

- Custo em laços de repetição:
  - custo de execução dos comandos internos, multiplicado pelo número de repetições.
  - ⚠ atenção no cálculo do número de repetições!! ⚠

6

Algoritmos Iterativos

Mãos na massa

Algoritmos iterativos

- Analisar cada método separadamente, começando por métodos que não chamam outros métodos, até chegar ao programa principal:
  - Do laço interno até o externo;
  - Considerações sobre a operação mais relevante.

```java
public int alg1(int n){
	int res = 1;
	for(int i=n; i>1; i--){
		res = res*i;
	}
	return res;
}
```

```java
public int alg1(int n){
	int res = 1;
	for(int i=n; i>1; i--){
		res = res*i;
	}
	return res;
}
```

1 – Operação mais relevante?
2 – Marcar operações
3 – Avaliar sequências e laços
4 – Há variação de casos?

```java
public int alg1(int n){
      int res = 1;
      for(int i=n; i>1; i--){
            res = res*i;
      }
      return res;
}
```

```
public int alg1(int n){
      int res = 1;
      for(int i=n; i>1; i--){
            res = res*i; (2)
      }
      return res;
}
```

```
public int alg1(int n){
      int res = 1;
      for(int i=n; i>1; i--){
            res = res*i; (2)
      }
      return res;
}
```

```
public int alg1(int n){
      int res = 1;
      for(int i=n; i>1; i--){ (n-1)
            res = res*i; (2)
      }
      return res;
}
```

```
public int alg1(int n){
        int res = 1;
        for(int i=n; i>1; i--){ (n-1)
                res = res*i; (2)
        }
        return res;
}
```

$f(n) = (n-1)*2 = 2n-2$, para todos os casos

```java
public int alg2(int[] arr, int x){
        for(int i=0; i<arr.length; i++){
                if(arr[i] == x)
                                return x;
        }
        return -1;
}
```

1 – Operação mais relevante?
2 – Marcar operações
3 – Avaliar sequências e laços
4 – Há variação de casos?

```
public int alg2(int[] arr, int x){
      for(int i=0; i<arr.length; i++){
            if(arr[i] == x)
                  return x;
      }
      return -1;
}
```

```
public int alg2(int[] arr, int x){
        for(int i=0; i<arr.length; i++){
                if(arr[i] == x)                (1)
                        return x;
        }
        return -1;
}
```

```
public int alg2(int[] arr, int x){
	for(int i=0; i<arr.length; i++){
		if(arr[i] == x)
				return x;
	}
	return -1;
}
```

(1)

```
public int alg2(int[] arr, int x){
      for(int i=0; i<arr.length; i++){
            if(arr[i] == x)
                        return x;
      }
      return -1;
}
```

(variações...)

(1)

```
public int alg2(int[] arr, int x){
      for(int i=0; i<arr.length; i++){
            if(arr[i] == x)
                  return x;
      }
      return -1;
}
```

(variações...)

(1)

Pior caso: loop executa n vezes → f(n) = (n)*1 = n

```
public int alg2(int[] arr, int x){
      for(int i=0; i<arr.length; i++){ (variações...)
            if(arr[i] == x)                 (1)
                        return x;
      }
      return -1;
}
```

Pior caso: loop executa n vezes → f(n) = (n)*1 = n

Melhor caso: loop executa 1 vez → f(n) = 1*1 = 1

Caso médio: $\frac{(n+1)}{2}$

```java
void alg3(int []array){
   for(int pos = 0; pos<array.length-1; pos++){
     int menor = pos;
     for(int j =pos+1; j<array.length; j++){
        if (array[j] < array[menor]) menor = j;
     }
     int aux = array[menor];
     array[menor] = array[pos];
     array[pos] = aux;
   }
}
```

```
void alg3(int []array){
   for(int pos = 0; pos<array.length-1; pos++){
      int menor = pos;
      for(int j =pos+1; j<array.length; j++){
         if (array[j] < array[menor]) menor = j;
      }
      int aux = array[menor];
      array[menor] = array[pos];
      array[pos] = aux;
   }
}
```

1 – Operação mais relevante?
2 – Marcar operações
3 – Avaliar sequências e laços
4 – Há variação de casos?

```
void alg3(int []array){
   for(int pos = 0; pos<array.length-1; pos++){
     int menor = pos;
     for(int j =pos+1; j<array.length; j++){
        if (array[j] < array[menor])                    (1)
                       menor = j;
     }
     int aux = array[menor];
     array[menor] = array[pos];
     array[pos] = aux;
   }
}
```

```
void alg3(int []array){
   for(int pos = 0; pos<array.length-1; pos++){
      int menor = pos;
      for(int j =pos+1; j<array.length; j++){
         if (array[j] < array[menor])
                  menor = j;
      }
      int aux = array[menor];
      array[menor] = array[pos];
      array[pos] = aux;
   }
}
```

(1)

Repetido quantas vezes?

```
void alg3(int []array){
   for(int pos = 0; pos<array.length-1; pos++){
     int menor = pos;
     for(int j =pos+1; j<array.length; j++){
        if (array[j] < array[menor])                (1)
                  enor = j;


                  menor];
     array[menor] = array[pos];
     array[pos] = aux;
   }
}
```

Este laço não é independente

```
void alg3(int []array){
  for(int pos = 0; pos<array.length-1; pos++){ (n-1)
    int menor = pos;
    for(int j =pos+1; j<array.length; j++){
      if (array[j] < array[menor])              (1)
            menor = j;
    }
    int aux = array[menor];
    array[menor] = array[pos];
    array[pos] = aux;
  }
}
```

```
void alg3(int []array){
   for(int pos = 0; pos<array.length-1; pos++){ (n-1)
     int menor = pos;
     for(int j =pos+1; j<array.length; j++){
        if (array[j] < array[menor])            (1)
                    menor = j;
     }
     int aux = array[menor];
     array[menor] = array[pos];
     array[pos] = aux;
   }
}
```

```java
void alg3(int []array){
   for(int pos = 0; pos<array.length-1; pos++){
     int menor = pos;
     for(int j =pos+1; j<array.length; j+
        if (array[j] < array[menor])
                    menor = j;
     }
     int aux = array[menor];
     array[menor] = array[pos];
     array[pos] = aux;
   }
}
```

~~(n-1)~~

n-1, n-2, n-3...0

(1)

```
void alg3(int []array){
    for(int pos = 0; pos<array.length-1; pos++){
      int menor = pos;
      for(int j =pos+1; j<array.length; j++){
         if (array[j] < array[menor])
                     menor = j;
      }
      int aux = array[menor];
      array[menor] = array[pos];
      array[pos] = aux;
    }
}
```

~~(n-1)~~

n-1, n-2, n-3...0

(1)

Temos um somatório

$$\sum_{i=0}^{n-1} i$$

```
void alg3(int []array){
   for(int pos = 0; pos<array.length-1; pos++){
     int menor = pos;
     for(int j =pos+1; j<array.length; j++){
        if (array[j] < array[menor])
                    menor = j;
     }
     int aux = array[menor];
     array[menor] = array[pos];
     array[pos] = aux;
   }
}
```

~~(n-1)~~

n-1, n-2, n-3...0

(1)

Temos um somatório

$$\sum_{i=0}^{n-1} i = \frac{(0+n-1)*n}{2}$$

```
void alg3(int []array){
   for(int pos = 0; pos<array.length-1; pos++){
      int menor = pos;
      for(int j =pos+1; j<array.length; j++){
         if (array[j] < array[menor])
                  menor = j;
      }
      int aux = array[menor];
      array[menor] = array[pos];
      array[pos] = aux;
   }
}
```

~~(n-1)~~

n-1, n-2, n-3...0

(1)

Temos um somatório

$$\sum_{i=0}^{n-1} i = \frac{n^2 - n}{2}$$

```
void alg3(int []array){
   for(int pos = 0; pos<array.length-1; pos++){
     int menor = pos;
     for(int j =pos+1; j<array.length; j++){
        if (array[j] < array[menor])
                    menor = j;
     }
     int aux = array[menor];
     array[menor] = array[pos];
     array[pos] = aux;
   }
}
```

~~(n-1)~~

n-1, n-2, n-3...0

(1)

Temos um somatório

$$f(n) = \frac{n^2 - n}{2}$$

Algoritmos recursivos

Algoritmos recursivos

- Quantas vezes uma chamada recursiva executará?
  - Representação pela função matemática do método.

- Teremos uma *equação de recorrência* contendo *base* e *recursividade.*
  - *Resolver a recorrência.*

```
int rec1(int n){
	if(n==0)
		return 1;
	else
		return n * rec1(n-1);
}
```

0 – Base e recursividade

1 – Operação mais relevante?

2 – Marcar operações

3 – Avaliar sequências

```
int rec1(int n){
	if(n==0)
		return 1;
	else
		return n * rec1(n-1);
}
```

$S(0) = 1$

```
int rec1(int n){
	if(n==0)
		return 1;
	else
		return n * rec1(n-1);
}
```

$S(0) = 1$

$S(n) = S(n-1) + 1$

```
int rec1(int n){
        if(n==0)
                return 1;
        else
                return n * rec1(n-1);
}
```

$S(0) = 1$

$S(n) = S(n-1) + 1$

<u>Atenção</u>: uma <u>*operação*</u> (neste caso, a multiplicação)

```
S(0) = 1
S(n) = S(n-1) + 1
```

$S(0) = 1$

$S(n) = S(n-1) + 1$

$S(n-1) = S(n-2) + 1$

$S(0) = 1$

$S(n) = S(n-1) + 1$

$\quad\;\; = S(n-2) + 1 + 1$

$S(n-1) = S(n-2) + 1$

$$S(0) = 1$$

$$S(n) = S(n-1) + 1$$

$$= S(n-2) + 2$$

$$S(n-1) = S(n-2) + 1$$

$$S(0) = 1$$

$$S(n) = S(n-1) + 1$$

$$= S(n-2) + 2$$

$$S(n-2) = S(n-3) + 1$$

$$S(0) = 1$$

$$S(n) = S(n-1) + 1$$

$$= S(n-2) + 2 \qquad\qquad S(n-2) = S(n-3) + 1$$

$$= S(n-3) + 1 + 2$$

$$S(0) = 1$$

$$\begin{aligned} S(n) &= S(n-1) + 1 \\ &= S(n-2) + 2 \\ &= S(n-3) + 3 \end{aligned}$$

$$S(n-2) = S(n-3) + 1$$

```
S(0) = 1
S(n) = S(n-1) + 1
     = S(n-2) + 2
     = S(n-3) + 3

F.G. = S(n-i) + i
```

```
S(0) = 1
S(n) = S(n-1) + 1
     = S(n-2) + 2
     = S(n-3) + 3

F.G. = S(n-i) + i
```

Até onde este valor vai?

$$S(0) = 1$$

$$\begin{aligned} S(n) &= S(n-1) + 1 \\ &= S(n-2) + 2 \\ &= S(n-3) + 3 \end{aligned}$$

$$\text{F.G.} = S(n-i) + i$$

Na base, $S(n-i) = S(0) \Rightarrow n-i = 0 \Rightarrow n = i$

$$S(0) = 1$$

$$\begin{aligned} S(n) &= S(n-1) + 1 \\ &= S(n-2) + 2 \\ &= S(n-3) + 3 \end{aligned}$$

F.G. $= S(n-i) + i$

Na base, $S(n-i) = S(0) \Rightarrow n-i = 0 \Rightarrow n = i$

Substituindo na F.G, $S(n) = S(0) + n$

$$S(0) = 1$$

$$\begin{aligned} S(n) &= S(n-1) + 1 \\ &= S(n-2) + 2 \\ &= S(n-3) + 3 \end{aligned}$$

$$F.G. = S(n-i) + i$$

Na base, $S(n-i) = S(0) \Rightarrow n-i = 0 \Rightarrow n = i$

Substituindo na F.G, $S(n) = S(0) + n = 1 + n$

$S(0) = 1$

$S(n) = S(n-1) + 1$

$= S(n-2) + 2$

$= S(n-3) + 3$

$F.G. = S(n-i) + i$

Na base, $S(n-i) = S(0) \Rightarrow n-i = 0 \Rightarrow n = i$

Substituindo na F.G, $S(n) = S(0) + n = 1 + n$

$$S(n) = n + 1$$

```
int rec2(int[] vet, int a, int b){
   if(a==b)
        return vet[a];
    else{
        int c = (a+b)/2;
        int d = rec2(vet, a, c);
        int e = rec2(vet, c, b);
        return d+e;
    }
}
```

0 – Base e recursividade
1 – Operação mais relevante?
2 – Marcar operações
3 – Avaliar sequências

```
int rec2(int[] vet, int a, int b){
    if(a==b)
         return vet[a];
     else{
         int c = (a+b)/2;
         int d = rec2(vet, a, c);
         int e = rec2(vet, c, b);
         return d+e;
     }
}
```

```
int rec3(int[] vet, int i){
   if(i==0)
      return vet[0];
   else{
      for(int j=0; j<vet.length; j++){
         v[j] += v[i];
      }
      return rec3(vet, i-1);
   }
}
```

0 – Base e recursividade
1 – Operação mais relevante?
2 – Marcar operações
3 – Avaliar sequências

```java
int rec3(int[] vet, int i){
   if(i==0)
      return vet[0];
   else{
      for(int j=0; j<vet.length; j++){
         v[j] += v[i];
      }
      return rec3(vet, i-1);
   }
}
```

Obrigado.

Dúvidas?

Quicksort

$$S(1) = 0$$
$$S(n) = 2S(n/2) + n$$

```
quicksort(int inicio, int fim, int[] dados){
	if(inicio<fim){
		int part = partição(ini, fim, dados);
		quicksort(ini, part-1, dados);
		quicksort(part+1, fim, dados);
	}
}
```

$T(n) = \Theta(n^{\log_b a})$, se $f(n)=O(n^{\log_b a - \varepsilon})$

$T(n) = 9T(n/3) + n$

$T(n) = \Theta(n^{\log_b a}), \text{ se } f(n)=O(n^{\log_b a - \varepsilon})$

$T(n) = 9T(n/3) + n$

a =
b =
f(n) =

$T(n) = \Theta\left(n^{\log_b a}\right)$, se $f(n)=O\left(n^{\log_b a - \varepsilon}\right)$

$T(n) = 9T(n/3) + n$

$a = 9$
$b = 3$
$f(n) = n$

$T(n) = \Theta(n^{\log_b a}),$ se $f(n) = O(n^{\log_b a - \varepsilon})$

$a = 9$
$b = 3$
$f(n) = n$

$T(n) = 9T(n/3) + n$

$n^{\log_3 9} = n^2$

$T(n) = \Theta(n^{\log_b a})$, se $f(n) = O(n^{\log_b a - \varepsilon})$

$T(n) = 9T(n/3) + n$

$n^{\log_3 9} = n^2$

$a = 9$
$b = 3$
$f(n) = n$
$\varepsilon = 1$

$T(n) = \Theta\ (n^{\log_b a}),$ se $f(n)=O(n^{\log_b a - \varepsilon})$

$T(n) = 9T(n/3) + n$

$n^{\log_3 9} = n^2 \rightarrow n^{2-1} \rightarrow n^1$

$a = 9$
$b = 3$
$f(n) = n$
$\varepsilon = 1$

$T(n) = \Theta(n^{\log_b a})$, se $f(n) = O(n^{\log_b a - \varepsilon})$

$a = 9$
$b = 3$
$f(n) = n$
$\varepsilon = 1$

$T(n) = 9T(n/3) + n$

$n^{\log_3 9} = n^2 \rightarrow n^{2-1} \rightarrow n^1$

$f(n) = n$

$f(n) = O(n^1)$?

$T(n) = \Theta(n^{\log_b a}),$ se $f(n)=O(n^{\log_b a - \varepsilon})$

$T(n) = 9T(n/3) + n$

$n^{\log_3 9} = n^2 \rightarrow n^{2-1} \rightarrow n^1$

$f(n) = n$

$f(n) = O(n^1)$?

$T(n) = \Theta(n^{\log_b a})$

$T(n) = \Theta(n^2)$

$a = 9$
$b = 3$
$f(n) = n$
$\varepsilon = 1$

$T(n) = \Theta(n^{\log_b a} * \log n)$, se $f(n) = \Theta(n^{\log_b a})$

$T(n) = 2T(n/2) + n+1$

$T(n) = \Theta(n^{\log_b a} * \log n)$, se $f(n) = \Theta(n^{\log_b a})$

$T(n) = 2T(n/2) + n+1$

a =
b =
f(n) =

$T(n) = \Theta\ (n^{\log_b a} * \log n)$, se $f(n) = \Theta(n^{\log_b a})$

$T(n) = 2T(n/2) + n+1$

a = 2
b = 2
f(n) = (n+1)

$T(n) = \Theta\ (n^{\log_b a} * \log n)$, se $f(n) = \Theta(n^{\log_b a})$

$T(n) = 2T(n/2) + n+1$

$n^{\log_b a} = n^{\log_2 2} = n^1$

$a = 2$
$b = 2$
$f(n) = (n+1)$

$T(n) = \Theta\ (n^{\log_b a} * \log n)$, se $f(n) = \Theta(n^{\log_b a})$

$a = 2$
$b = 2$
$f(n) = (n+1)$

$T(n) = 2T(n/2) + n+1$

$n^{\log_b a} = n^{\log_2 2} = n^1$

$f(n) = n+1$

$T(n) = \Theta(n^{\log_b a} * \log n)$, se $f(n) = \Theta(n^{\log_b a})$

$a = 2$
$b = 2$
$f(n) = (n+1)$

$T(n) = 2T(n/2) + n+1$

$n^{\log_b a} = n^{\log_2 2} = n^1$

$f(n) = n+1$

$f(n) = \Theta(n^{\log_b a}) = \Theta(n)$?

$T(n) = \Theta\ (n^{\log_b a} * \log n)$, se $f(n) = \Theta(n^{\log_b a})$

$T(n) = 2T(n/2) + n+1$

$n^{\log_b a} = n^{\log_2 2} = n^1$

$f(n) = n+1$

$f(n) = \Theta(n^{\log_b a}) = \Theta(n)$?

$(n+1) = \Theta(n)$?

$a = 2$
$b = 2$
$f(n) = (n+1)$

$T(n) = \Theta\ (n^{\log_b a} * \log n)$, se $f(n) = \Theta(n^{\log_b a})$

a = 2
b = 2
f(n) = (n+1)

$T(n) = 2T(n/2) + n+1$

$n^{\log_b a} = n^{\log_2 2} = n^1$

$f(n) = n+1$

$f(n) = \Theta(n^{\log_b a}) = \Theta(n)$?

$(n+1) = \Theta(n)$?

$T(n) = \Theta\ (n^{\log_b a} * \log n)$, se $f(n) = \Theta(n^{\log_b a})$

a = 2
b = 2
f(n) = (n+1)

$T(n) = 2T(n/2) + n+1$

$n^{\log_b a} = n^{\log_2 2} = n^1$

$f(n) = n+1$

$f(n) = \Theta(n^{\log_b a}) = \Theta(n)$?

$(n+1) = \Theta(n)$?

$T(n) = \Theta\ (n^{\log_b a} * \log n) = \Theta(n * \log n)$

$T(n) = \Theta(f(n))$, se $f(n) = \Omega(n^{\log_b a + \varepsilon})$, para alguma constante $\varepsilon > 0$, e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$

$T(n) = T(n/2) + 3n$

$T(n) = \Theta(f(n))$, se $f(n) = \Omega(n^{\log_b a + \varepsilon})$, para alguma constante $\varepsilon > 0$, e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$

$T(n) = T(n/2) + 3n$

a =
b =
f(n) =

T(n)= $\Theta(f(n))$, se $f(n)=\Omega(n^{\log_b a + \varepsilon})$, para alguma constante $\varepsilon>0$,
e se $af(n/b) \le cf(n)$ para alguma constante $c<1$

$T(n) = T(n/2) + 3n$

$a = 1$
$b = 2$
$f(n) = 3n$

$T(n) = \Theta(f(n))$, se $f(n) = \Omega(n^{\log_b a + \varepsilon})$, para alguma constante $\varepsilon > 0$,
e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$

$T(n) = T(n/2) + 3n$

$a = 1$
$b = 2$
$f(n) = 3n$

$T(n) = \Theta(f(n))$, se $f(n) = \Omega(n^{\log_b a + \varepsilon})$, para alguma constante $\varepsilon > 0$,
e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$

$T(n) = T(n/2) + 3n$

$a = 1$
$b = 2$
$f(n) = 3n$

$n^{\log_b a} = n^{\log_2 1} = n^0$

$T(n) = \Theta(f(n))$, se $f(n) = \Omega(n^{\log_b a + \varepsilon})$, para alguma constante $\varepsilon > 0$,
e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$

$T(n) = T(n/2) + 3n$

$a = 1$
$b = 2$
$f(n) = 3n$
$\varepsilon = 1$

$n^{\log_b a} = n^{\log_2 1} = n^0$

$T(n) = \Theta(f(n))$, se $f(n) = \Omega(n^{\log_b a + \varepsilon})$, para alguma constante $\varepsilon > 0$,
e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$

$T(n) = T(n/2) + 3n$

$a = 1$
$b = 2$
$f(n) = 3n$
$\varepsilon = 1$

$n^{\log_b a} = n^{\log_2 1} = n^0 \rightarrow n^{0+\varepsilon} = n^{0+1} = n^1$

$T(n) = \Theta(f(n))$, se $f(n) = \Omega(n^{\log_b a + \varepsilon})$, para alguma constante $\varepsilon > 0$,
e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$

$T(n) = T(n/2) + 3n$

$a = 1$
$b = 2$
$f(n) = 3n$
$\varepsilon = 1$

$n^{\log_b a} = n^{\log_2 1} = n^0 \rightarrow n^{0+\varepsilon} = n^{0+1} = n^1$

$f(n) = 3n$

$f(n) = \Omega(n^1)$?

$T(n) = \Theta(f(n))$, se $f(n) = \Omega(n^{\log_b a + \varepsilon})$, para alguma constante $\varepsilon > 0$,
e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$

$T(n) = T(n/2) + 3n$

$a = 1$
$b = 2$
$f(n) = 3n$
$\varepsilon = 1$

$n^{\log_b a} = n^{\log_2 1} = n^0 \rightarrow n^{0+\varepsilon} = n^{0+1} = n^1$

$f(n) = 3n$

$f(n) = \Omega(n^1)$? SIM!

$T(n) = \Theta(f(n))$, se $f(n) = \Omega(n^{\log_b a + \varepsilon})$, para alguma constante $\varepsilon > 0$,
e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$

$T(n) = T(n/2) + 3n$

$a = 1$
$b = 2$
$f(n) = 3n$
$\varepsilon = 1$
$c = ½$

$n^{\log_b a} = n^{\log_2 1} = n^0 \rightarrow n^{0+\varepsilon} = n^{0+1} = n^1$

$f(n) = 3n$

$f(n) = \Omega(n^1)$? SIM!

$T(n) = \Theta(f(n))$, se $f(n) = \Omega(n^{\log_b a + \varepsilon})$, para alguma constante $\varepsilon > 0$,
e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$

$T(n) = T(n/2) + 3n$

$n^{\log_b a} = n^{\log_2 1} = n^0 \rightarrow n^{0+\varepsilon} = n^{0+1} = n^1$

$f(n) = 3n$

$f(n) = \Omega(n^1)$? SIM!

$af(n/b) \leq cf(n)$

$a = 1$
$b = 2$
$f(n) = 3n$
$\varepsilon = 1$
$c = ½$

$T(n) = \Theta(f(n))$, se $f(n) = \Omega(n^{\log_b a + \varepsilon})$, para alguma constante $\varepsilon > 0$, e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$

$T(n) = T(n/2) + 3n$

$a = 1$
$b = 2$
$f(n) = 3n$
$\varepsilon = 1$
$c = ½$

$n^{\log_b a} = n^{\log_2 1} = n^0 \rightarrow n^{0+\varepsilon} = n^{0+1} = n^1$

$f(n) = 3n$

$f(n) = \Omega(n^1)$? SIM!

$af(n/b) \leq cf(n) \rightarrow 1 * 3n/2 \leq ½ * 3n$

$T(n) = \Theta(f(n))$, se $f(n) = \Omega(n^{\log_b a + \varepsilon})$, para alguma constante $\varepsilon > 0$, e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$

$T(n) = T(n/2) + 3n$

$a = 1$
$b = 2$
$f(n) = 3n$
$\varepsilon = 1$
$c = ½$

$n^{\log_b a} = n^{\log_2 1} = n^0 \rightarrow n^{0+\varepsilon} = n^{0+1} = n^1$

$f(n) = 3n$

$f(n) = \Omega(n^1)$? SIM!

$af(n/b) \leq cf(n) \rightarrow 1 * 3n/2 \leq ½ * 3n$

$3n/2 \leq 3n/2$?

$T(n) = \Theta(f(n))$, se $f(n) = \Omega(n^{\log_b a + \varepsilon})$, para alguma constante $\varepsilon > 0$, e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$

$T(n) = T(n/2) + 3n$

$a = 1$
$b = 2$
$f(n) = 3n$
$\varepsilon = 1$
$c = ½$

$n^{\log_b a} = n^{\log_2 1} = n^0 \rightarrow n^{0+\varepsilon} = n^{0+1} = n^1$

$f(n) = 3n$

$f(n) = \Omega(n^1)$? SIM!

$af(n/b) \leq cf(n) \rightarrow 1 * 3n/2 \leq ½ * 3n$

$3n/2 \leq 3n/2$? SIM!

$T(n) = \Theta(f(n))$, se $f(n) = \Omega(n^{\log_b a + \varepsilon})$, para alguma constante $\varepsilon > 0$, e se $af(n/b) \leq cf(n)$ para alguma constante $c < 1$

$T(n) = T(n/2) + 3n$

$a = 1$
$b = 2$
$f(n) = 3n$
$\varepsilon = 1$
$c = ½$

$n^{\log_b a} = n^{\log_2 1} = n^0 \rightarrow n^{0+\varepsilon} = n^{0+1} = n^1$

$f(n) = 3n$

$f(n) = \Omega(n^1)$? SIM!

$af(n/b) \leq cf(n) \rightarrow 1 * 3n/2 \leq ½ * 3n$

$3n/2 \leq 3n/2$? SIM!

$T(n) = \Theta(f(n)) \rightarrow T(n) = \Theta(3n) \rightarrow T(n) = \Theta(n)$

Obrigado.

Dúvidas?