Divisão e Conquista

Norton T. Roman

Apostila baseada nos trabalhos de Cid de Souza,
Cândida da Silva e Delano M. Beder

Divisão e Conquista

- Construção incremental
  - Consiste em, inicialmente, resolver o problema para um sub-conjunto dos elementos da entrada e, então adicionar os demais elementos um a um. Em muitos casos, se os elementos forem adicionados em uma ordem ruim, o algoritmo não será eficiente.
- Ex:
  - Calcule n!, recursivamente

Divisão e Conquista

- Divisão e Conquista
  - Para resolver um problema eles chamam a si mesmos para resolver subproblemas menores e combinam as subrespostas.
  - É mais um paradigma de projeto de algoritmos baseado no princípio da indução.
  - Informalmente, podemos dizer que o paradigma incremental representa o projeto de algoritmos por indução fraca, enquanto o paradigma de divisão e conquista representa o projeto por indução forte.

Divisão e Conquista

- **Dividir** o problema em determinado número de subproblemas.
- **Conquistar** os subproblemas, resolvendo-os recursivamente.
  - Se o tamanho do subproblema for pequeno o bastante, então a solução é direta.
- **Combinar** as soluções fornecidas pelos subproblemas, a fim de produzir a solução para o problema original.

Divisão e Conquista

- A busca binária recursiva utiliza essa técnica?

Divisão e Conquista

- A busca binária recursiva utiliza essa técnica?
  - Dividir:
    - Divide o problema em sub-problemas?
  - Conquistar:
    - Resolve os sub-problemas recursivamente?
  - Combinar:
    - Forma a solução final a partir da combinação das soluções dos sub-problemas?

Divisão e Conquista

- A busca binária recursiva utiliza essa técnica?
  - Dividir:
    - Divide o problema em sub-problemas?
  - Conquistar:
    - Resolve os sub-problemas recursivamente?
  - Combinar:
    - Forma a solução final a partir da combinação das soluções dos sub-problemas?
    - Nesse caso, a etapa de combinar tem custo zero, pois o resultado do subproblema já é o resultado do problema maior.

Exemplo: Exponenciação

- Calcule a^n, para todo real a e inteiro $n \geq 0$
  - Caso base:

Exemplo: Exponenciação

- Calcule a^n, para todo real a e inteiro $n \geq 0$
  - **Caso base**:
    - $n = 0$; $a^0 = 1$
  - **Hipótese de Indução**:

Exemplo: Exponenciação

- Calcule a^n, para todo real a e inteiro $n \geq 0$
  - **Caso base**:
    - $n = 0$; $a^0 = 1$
  - **Hipótese de indução**:
    - Suponha que, para qualquer inteiro $n > 0$ e real a, sei calcular a^{n-1}
  - **Passo da indução**:

Exemplo: Exponenciação

- Calcule a^n, para todo real a e inteiro $n \geq 0$
  - **Caso base**:
    - $n = 0$; $a^0 = 1$
  - **Hipótese de indução**:
    - Suponha que, para qualquer inteiro $n > 0$ e real a, sei calcular a^{n-1}
  - **Passo da indução**:
    - Queremos provar que conseguimos calcular a^n , para $n > 0$. Por hipótese de indução, sei calcular a^{n-1} . Então, calculo a^n multiplicando a^{n-1} por a.

Exemplo: Exponenciação

```
/* Entrada: A base a e o expoente n.
   Saída: O valor de a elevado a n. */
double exponenciacao(double a, int n) {
  double an;
  if (n == 0) {
     return 1; /* caso base */
  } else {
     return a * exponenciacao(a, n - 1);
  }
}
```

Solução 1: Complexidade

- Qual a complexidade dessa solução?

```
/* Entrada: A base a e o expoente n.
   Saída: O valor de a elevado a n. */
double exponenciacao(double a, int n) {
  double an;
  if (n == 0) {
      return 1; /* caso base */
  } else {
      return a * exponenciacao(a, n - 1);
  }
}
```

Solução 1: Complexidade

- Qual a complexidade dessa solução?
  - n = 0:

```
/* Entrada: A base a e o expoente n.
    Saída: O valor de a elevado a n. */
double exponenciacao(double a, int n) {
  double an;
  if (n == 0) {
      return 1; /* caso base */
  } else {
      return a * exponenciacao(a, n - 1);
  }
}
```

Solução 1: Complexidade

- Qual a complexidade dessa solução?
  - N = 0: 1 comparação. Chamemos c_1
  - $n > 0$:

```
/* Entrada: A base a e o expoente n.
   Saída: O valor de a elevado a n. */
double exponenciacao(double a, int n) {
  double an;
  if (n == 0) {
     return 1; /* caso base */
  } else {
     return a * exponenciacao(a, n - 1);
  }
}
```

Solução 1: Complexidade

- Qual a complexidade dessa solução?
  - n = 0: 1 comparação. Chamemos c_1
  - n > 0:
    - 1 comparação + 1 multiplicação
    - Chamemos c_2

```
/* Entrada: A base a e o expoente n.
   Saída: O valor de a elevado a n. */
double exponenciacao(double a, int n) {
  double an;
  if (n == 0) {
      return 1; /* caso base */
  } else {
      return a * exponenciacao(a, n - 1);
  }
}
```

Solução 1: Complexidade

- Qual a complexidade dessa solução?
  - n = 0: 1 comparação. Chamemos c_1
  - n > 0:
    - 1 comparação + 1 multiplicação
    - Chamemos c_2

$$T(n) = \begin{cases} c1 & \text{se} \quad n = 0 \\ T(n-1) + c2 & \text{se} \quad n > 0 \end{cases}$$

```
/* Entrada: A base a e o expoente n.
   Saída: O valor de a elevado a n. */
double exponenciacao(double a, int n) {
  double an;
  if (n == 0) {
      return 1; /* caso base */
  } else {
      return a * exponenciacao(a, n - 1);
  }
}
```

Solução 1: Complexidade

- Qual a complexidade dessa solução?
  - n = 0: 1 comparação. Chamemos c_1
  - n > 0:
    - $$T(n) = \begin{cases} c1 & \text{se} \quad n = 0 \\ T(n-1) + c2 & \text{se} \quad n > 0 \end{cases}$$
    - Expandindo a relação de recorrência, teremos:

$$T(n) = c1 + \sum_{i=1}^{n} c2 = c1 + nc2 = \Theta(n)$$

Solução 2

- Vamos agora projetar um algoritmo para o problema usando indução forte de forma a obter um algoritmo de divisão e conquista.
- Indução forte?
  - Caso base: $n = 1$
  - Provar que para $\forall n \geq 2$, se a propriedade é válida para $\forall 1 \geq m \geq n$, ela é valida para $n + 1$.
  - Ou seja, em vez de supor que sabemos a reposta para $n-1$, supomos que sabemos para qualquer $k < n$

Solução 2 – Indução Forte

- **Caso base:**

Solução 2 – Indução Forte

- **Caso base**:
  - $n = 0$; $a^0 = 1$
- **Hipótese de indução**:
  -

Solução 2 – Indução Forte

- **Caso base**:
  - $n = 0$; $a^0 = 1$
- **Hipótese de indução**:
  - Suponha que, para qualquer inteiro $n > 0$ e real a, sei calcular a^k, para todo $k < n$.
- **Passo**:

Solução 2 – Indução Forte

- **Caso base**:
  - $n = 0$; $a^0 = 1$
- **Hipótese de indução**:
  - Suponha que, para qualquer inteiro $n > 0$ e real a, sei calcular a^k , para todo $k < n$.
- **Passo**:
  - Como ficaria o cálculo de $a^{\lfloor n \rfloor}$ se soubéssemos $a^{\lfloor n/2 \rfloor}$?

Solução 2 – Indução Forte

- **Passo**:
  - Como ficaria o cálculo de a^n se soubéssemos $a^{\lfloor n/2 \rfloor}$?
  - Queremos provar que conseguimos calcular a^n , para n > 0. Por hipótese de indução, sei calcular $a^{\lfloor n/2 \rfloor}$. Então, calculo a^n da seguinte forma:

$$a^n = \begin{cases} a^{\lfloor \frac{n}{2} \rfloor^2} & \text{se} \quad n \quad \% \quad 2 = 0 \\ a \times a^{\lfloor \frac{n}{2} \rfloor^2} & \text{se} \quad n \quad \% \quad 2 = 1 \end{cases}$$

Solução 2 – Indução Forte

```
/* Entrada: A base a e o expoente n.
   Saída: O valor de a elevado a n. */
double exponenciacao(double a, int n) {
  double an;
  if (n == 0) {
     return 1; /* caso base */
  } else {
     double aux = exponenciacao(a, n / 2);
     an = aux * aux;
     if (n % 2 == 1) {
       an = an * a;
     }
     return an;
  }
}
```

Solução 2 – Complexidade

- Qual o número de operações executadas?
  - n=0:

```
/* Entrada: A base a e o expoente n.
   Saída: O valor de a elevado a n. */
double exponenciacao(double a, int n) {
  double an;
  if (n == 0) {
      return 1; /* caso base */
  } else {
      double aux = exponenciacao(a, n / 2);
      an = aux * aux;
      if (n % 2 == 1) {
        an = an * a;
      }
      return an;
  }
}
```

Solução 2 – Complexidade

- Qual o número de operações executadas?
  - n=0:
    - Teste
  - n > 0:
    -

```
/* Entrada: A base a e o expoente n.
    Saída: O valor de a elevado a n. */
double exponenciacao(double a, int n) {
  double an;
  if (n == 0) {
      return 1; /* caso base */
  } else {
      double aux = exponenciacao(a, n / 2);
      an = aux * aux;
      if (n % 2 == 1) {
        an = an * a;
      }
      return an;
  }
}
```

Solução 2 – Complexidade

- Qual o número de operações executadas?
  - n=0:
    - Teste – c_1
  - n > 0:
    - teste + atribuição + multiplicação + atribuição + teste + multiplicação + atribuição – c_2

```
/* Entrada: A base a e o expoente n.
   Saída: O valor de a elevado a n. */
double exponenciacao(double a, int n) {
  double an;
  if (n == 0) {
      return 1; /* caso base */
  } else {
      double aux = exponenciacao(a, n / 2);
      an = aux * aux;
      if (n % 2 == 1) {
        an = an * a;
      }
      return an;
  }
}
```

Solução 2 – Complexidade

- Qual o número de operações executadas?
  - Seja T (n) o número de operações executadas pelo algoritmo para calcular a^n. Então a relação de recorrência deste algoritmo é:

$$T(n) = \begin{cases} c1 & \text{se} \quad n = 0 \\ T(\lfloor \frac{n}{2} \rfloor) + c2 & \text{se} \quad n > 0 \end{cases}$$

```
/* Entrada: A base a e o expoente n.
   Saída: O valor de a elevado a n. */
double exponenciacao(double a, int n) {
  double an;
  if (n == 0) {
     return 1; /* caso base */
  } else {
     double aux = exponenciacao(a, n / 2);
     an = aux * aux;
     if (n % 2 == 1) {
       an = an * a;
     }
     return an;
  }
}
```

Solução 2 – Complexidade

- Qual o número de operações executadas?
  - $T(n) = \begin{cases} c1 & \text{se} \quad n = 0 \\ T(\lfloor \frac{n}{2} \rfloor) + c2 & \text{se} \quad n > 0 \end{cases}$
  - e $T(n) \in \Theta(\log n)$

```
/* Entrada: A base a e o expoente n.
    Saída: O valor de a elevado a n. */
double exponenciacao(double a, int n) {
  double an;
  if (n == 0) {
      return 1; /* caso base */
  } else {
      double aux = exponenciacao(a, n / 2);
      an = aux * aux;
      if (n % 2 == 1) {
        an = an * a;
      }
      return an;
  }
}
```

Exercício – Máximo e Mínimo

- Problema:
  - Dado um conjunto S de $n \geq 2$ números reais, determinar o maior e o menor elemento de S.
  - Um algoritmo incremental para esse problema faz $2n - 3$ comparações entre elementos do vetor:
    - Fazemos 1 comparação no caso base, em que n=2. Nesse caso comparamos um ao outro.
    - Fazemos 2 no passo, testando o menor e o maior em (n-1) com o elemento em n
      - São 2 comparações em cada uma das n-1 chamadas, exceto a primeira (n=2), em que há somente uma.

Exercício – Máximo e Mínimo

- Incremental:
  - Base: ???

Exercício – Máximo e Mínimo

- Incremental:
  - Base: n=2
    - ???

Exercício – Máximo e Mínimo

- Incremental:
  - Base: n=2
    - Compara um com o outro, vê qual o maior e o menor
  - Hipótese de indução:
    - ???

Exercício – Máximo e Mínimo

- Incremental:
  - Base: n=2
    - Compara um com o outro, vê qual o maior e o menor
  - Hipótese de indução:
    - Sei o maior e o menor dentre os n-1 primeiros elementos do vetor
  - Passo:
    - ???

Exercício – Máximo e Mínimo

- Incremental:
  - Base: n=2
    - Compara um com o outro, vê qual o maior e o menor
  - Hipótese de indução:
    - Sei o maior e o menor dentre os n-1 primeiros elementos do vetor
  - Passo:
    - Se o n-ésimo elemento for maior que o maior em n-1, ele é o maior de todos, senão é o em n-1
    - Se o n-ésimo elemento for menor que o menor em n-1, ele é o menor de todos, senão é o em n-1

Exercício – Máximo e Mínimo

```
public static double[] maiorMenor(double s[], int n) {
    double[] resp;
    if (n==2) {
        resp = new double[2];
        if (s[0]>s[1]) {
            resp[0] = s[0];
            resp[1] = s[1];
        }
        else {
            resp[0] = s[1];
            resp[1] = s[0];
        }
    }
    else {
        resp = maiorMenor(s,n-1);
        if (s[n-1] > resp[0]) resp[0] = s[n-1];
        if (s[n-1] < resp[1]) resp[1] = s[n-1];
    }
    return(resp);
}
```

Exercício – Máximo e Mínimo

- Problema:
  - Dado um conjunto S de $n \geq 2$ números reais, determinar o maior e o menor elemento de S.
  - Será que um algoritmo de divisão e conquista seria melhor?
    - Projete um algoritmo de divisão e conquista para esse problema que faz um número menor de comparações.

Exercício – Máximo e Mínimo

- Divisão e Conquista:
  - Base:

Exercício – Máximo e Mínimo

- Divisão e Conquista:
  - Base: n=2
    - ???

Exercício – Máximo e Mínimo

- Divisão e Conquista:
  - Base: n=2
    - Compara um com o outro, vê qual o maior e o menor
  - Hipótese de indução (forte):
    - ???

Exercício – Máximo e Mínimo

- Divisão e Conquista:
  - Base: n=2
    - Compara um com o outro, vê qual o maior e o menor
  - Hipótese de indução (forte):
    - Sei qual o maior e menor elemento em sub-vetores de $\lfloor n/2 \rfloor$
  - Passo:
    - ???

Exercício – Máximo e Mínimo

- Divisão e Conquista:
  - Base: n=2
    - Compara um com o outro, vê qual o maior e o menor
  - Hipótese de indução (forte):
    - Sei qual o maior e menor elemento em sub-vetores de $\lfloor n/2 \rfloor$
  - Passo:
    - Se n par, o maior será o maior dentre as respostas de $\lfloor n/2 \rfloor$. O mesmo vale para o menor.
    - Se n ímpar, o maior será o maior dentre s[n] e o maior dos dois $\lfloor n/2 \rfloor$. O mesmo vale para o menor.

Exercício – Máximo e Mínimo

- Divisão e Conquista:
  - Complexidade: ???

Exercício – Máximo e Mínimo

- Divisão e Conquista:
  - Complexidade: ???

$$T(n) = \begin{cases} c_1, & n=2 \\ 2T(n/2) + c_2, & n>2 \end{cases}$$

  - Implementação: Exercício
  - Qual a complexidade disso?
    - Você espera ser menor que O(n)?

Divisão e Conquista

- A complexidade de tempo de algoritmos divisão e conquista para um entrada de tamanho n é:
  - $T(n) = Dividir(n) + Conquistar(n) + Combinar(n)$.
  - Para entradas pequenas, isto é, para $n \leq c$, c pequeno, podemos assumir que $T(n) = O(1)$.
- Vamos supor que o problema seja dividido em a subproblemas, cada um com 1/b do tamanho original.
  - Como fica a "Conquista"?

Divisão e Conquista

- Vamos supor que o problema seja dividido em a subproblemas, cada um com 1/b do tamanho original.
  - Como fica a "Conquista"? R: aT(n/b)
  - Se levamos D(n) para dividir o problema em subproblemas e C(n) para combinar as soluções dados aos subproblemas, então tem-se a recorrência T(n) tal que:

$$T(n) = \begin{cases} O(1) & \text{se} \quad n \leq c \\ aT(n/b) + D(n) + C(n) & \text{caso contrário} \end{cases}$$

Divisão e Conquista

- Expressão geral de recorrência de um algoritmo de divisão e conquista:
  - $T(n) = aT(n/b) + f(n)$
- Onde:
  - a representa o número de subproblemas obtidos na divisão
  - n/b representa seu tamanho
  - f(n) é a função que dá a complexidade das etapas de divisão e combinação
    - $f(n) = D(n) + C(n)$

Divisão e Conquista

Teorema Mestre (CLRS)

Sejam $a \geq 1$ e $b > 1$ constantes, seja $f(n)$ uma função e seja $T(n)$ definida para os inteiros não-negativos pela relação de recorrência

$$T(n) = aT(n/b) + f(n)$$

Então $T(n)$ pode ser limitada assintoticamente da seguinte maneira:

1. Se $f(n) \in O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a + \epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplo

- T(n) = 9T(n/3) + n
  - ???

1. Se $f(n) \in O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a + \epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplo

- T(n) = 9T(n/3) + n
  - a = 9
  - b = 3
  - Texto as alternativas

1. Se $f(n) \in O(n^{\log_b a-\epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a+\epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplo

- T(n) = 9T(n/3) + n
  - a = 9
  - b = 3
  - Texto as alternativas
    - Note que $n^{\log_3 9}$ aparece em todas as 3 alternativas

1. Se $f(n) \in O(n^{\log_b a-\epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a+\epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplo

- T(n) = 9T(n/3) + n
  - a = 9
  - b = 3
  - Testo as alternativas
    - Note que $n^{\log_3 9}$ aparece em todas as 3 alternativas
    - $n^{\log_3 9} = n^2$
    - $f(n) \in O(n^{\log_3 9 - \epsilon}), \epsilon > 0$?

1. Se $f(n) \in O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a + \epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplo

- T(n) = 9T(n/3) + n
  - a = 9
  - b = 3
  - Testo as alternativas
    - Note que $n^{\log_3 9}$ aparece em todas as 3 alternativas
    - $n^{\log_3 9} = n^2$
    - $f(n) \in O(n^{\log_3 9 - \epsilon}), \epsilon > 0$?
    - Pelo caso 1:

$$n \in O(n^{2-1}) \rightarrow n \in O(n), \epsilon = 1 \rightarrow T(n) \in \Theta(n^2)$$

1. Se $f(n) \in O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a + \epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplo

- $T(n) = T(2n/3) + 1$

1. Se $f(n) \in O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a + \epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplo

- T(n) = T(2n/3) + 1
  - A = 1
  - B = 3/2
  - f(n) = 1
  - $n^{\log_{3/2} 1} = n^0 = 1$
  - Pelo caso 2:

$$1 \in \Theta(n^0) \rightarrow 1 \in \Theta(1) \rightarrow T(n) \in \Theta(n^0 \log(n)) \rightarrow T(n) \in \Theta(\log(n))$$

1. Se $f(n) \in O(n^{\log_b a-\epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a+\epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplo

- T(n) = T(2n/3) + 1
  - A = 1
  - B = 3/2
  - f(n) = 1
  - $n^{\log_{3/2} 1} = n^0 = 1$
  - Onde se encaixa?

1. Se $f(n) \in O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a + \epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplo

- $T(n) = 3T(n/4) + n \times \log(n)$
  - ???

1. Se $f(n) \in O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a + \epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplo

- $T(n) = 3T(n/4) + n \times \log(n)$
  - A = 3
  - B = 4
  - f(n) = nlog(n)
  - ???

1. Se $f(n) \in O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a + \epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplo

- $T(n) = 3T(n/4) + n \times \log(n)$
  - A = 3
  - B = 4
  - f(n) = nlog(n)
  - $n^{\log_b a} = n^{\log_4 3} = n^{0{,}793}$
  - $f(n) \in O(n^{\log_b a - \epsilon}) \rightarrow n \log(n) = O(n^{\log_4 3 - \epsilon}) = \Theta(n^{0{,}793 - \epsilon}), \epsilon > 0\ ?$
    - ???

1. Se $f(n) \in O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a + \epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplo

- T(n) = 3T(n/4) + n × log(n)
  - A = 3
  - B = 4
  - f(n) = nlog(n)
  - $n^{\log_b a} = n^{\log_4 3} = n^{0,793}$
  - $f(n) \in O(n^{\log_b a - \epsilon}) \rightarrow n \log(n) = O(n^{\log_4 3 - \epsilon}) = \Theta(n^{0,793 - \epsilon}), \epsilon > 0$?
    - Não, pois isso implica que $n \log(n) = O(n^c), c < 1$

1. Se $f(n) \in O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a + \epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplo

- T(n) = 3T(n/4) + n × log(n)
  - A = 3
  - B = 4
  - f(n) = nlog(n)
  - $n^{\log_b a} = n^{\log_4 3} = n^{0{,}793}$
  - $f(n) \in \Theta(n^{\log_b a}) \rightarrow n \log(n) = \Theta(n^{\log_4 3}) = \Theta(n^{0{,}793})$?
    - ???

1. Se $f(n) \in O(n^{\log_b a-\epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a+\epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplo

- T(n) = 3T(n/4) + n × log(n)
  - A = 3
  - B = 4
  - f(n) = nlog(n)
  - $n^{\log_b a} = n^{\log_4 3} = n^{0{,}793}$
  - $f(n) \in \Theta(n^{\log_b a}) \rightarrow n \log(n) = \Theta(n^{\log_4 3}) = \Theta(n^{0{,}793})$?
    - Não, pois isso implica que $n \log(n) = \Theta(n^c), c < 1$

1. Se $f(n) \in O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a + \epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplo

- T(n) = 3T(n/4) + n × log(n)
  - A = 3
  - B = 4
  - f(n) = nlog(n)
  - $n^{\log_b a} = n^{\log_4 3} = n^{0,793}$
  - $f(n) \in \Omega(n^{\log_b a + \epsilon}) \rightarrow n \log(n) = \Omega(n^{\log_4 3 + \epsilon}) = \Omega(n^{0,793 + \epsilon}), \epsilon > 0\,?$
    - ???

1. Se $f(n) \in O(n^{\log_b a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a + \epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplo

- T(n) = 3T(n/4) + n × log(n)
  - A = 3
  - B = 4
  - f(n) = nlog(n)
  - $n^{\log_b a} = n^{\log_4 3} = n^{0{,}793}$
  - $f(n) \in \Omega(n^{\log_b a+\epsilon}) \rightarrow n \log(n) = \Omega(n^{\log_4 3+\epsilon}) = \Omega(n^{0{,}793+\epsilon}), \epsilon > 0\,?$
    - Sim, se fizermos $\epsilon \approx 2$, teremos $n \log(n) = \Omega(n^1)$
  - $af(n/b) \leqslant cf(n), c < 1\,?$

1. Se $f(n) \in O(n^{\log_b a-\epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a+\epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplo

- T(n) = 3T(n/4) + n × log(n)
  - A = 3
  - B = 4
  - f(n) = nlog(n)
  - $n^{\log_b a} = n^{\log_4 3} = n^{0,793}$
  - $af(n/b) \leqslant cf(n), c < 1$?
    - $3((n/4)\log(n/4)) \leqslant cn\log(n), c < 1$
    - $(3/4)n\log(n/4) \leqslant cn\log(n), c < 1$ (para c=3/4 é verdade)
  - Assim, pelo caso 3, $T(n) \in \Theta(n\log(n))$

1. Se $f(n) \in O(n^{\log_b a-\epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) \in \Theta(n^{\log_b a})$
2. Se $f(n) \in \Theta(n^{\log_b a})$, então $T(n) \in \Theta(n^{\log_b a} \log n)$
3. Se $f(n) \in \Omega(n^{\log_b a+\epsilon})$, para alguma constante $\epsilon > 0$ e se $af(n/b) \leq cf(n)$, para alguma constante $c < 1$ e para n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre – Exemplos

- Exemplos onde o Teorema Mestre se aplica
  - (Caso 1) $T(n) = 4T(n/2) + n \log(n)$, $T(1) = 1$.
  - (Caso 2) $T(n) = 2T(n/2) + n$, $T(1) = 1$.
  - (Caso 3) $T(n) = T(n/2) + n \log(n)$, $T(1) = 1$.

Teorema Mestre – Exemplos

- Exemplos onde o Teorema Mestre não se aplica
  - $T(n) = T(n - 1) + n \log(n)$, $T(1) = 1$.
  - $T(n) = T(n - a) + T(a) + n$, $T(b) = 1$.
    (para inteiros $a \geq 1$, $b \leq a$, a e b inteiros)
  - $T(n) = T(\alpha n) + T((1 - \alpha)n) + n$, $T(1) = 1$.
    (para $0 < \alpha < 1$)
  - $T(n) = T(n - 1) + \log(n)$, $T(1) = 1$.
  - $T(n) = 2\,T(n/2) + n \log(n)$, $T(1) = 1$.

Exercícios

- Use o Teorema Mestre para encontrar solução para as seguintes recorrências:
  - $T(n) = 4T(n/2) + n \log(n)$
  - $T(n) = 2T(n/2) + n.$
  - $T(n) = T(n/2) + n \log(n)$
  - $T(n) = T(n/2) + \Theta(1)$
    [Recorrência da exponenciacão - divisão e conquista]
  - $T(n) = 4T(n/2) + n.$
  - $T(n) = 4T(n/2) + n^2$.
  - $T(n) = 4T(n/2) + n^3$.