Aula 10 – Divisão e Conquista

Norton T. Roman & Luciano A. Digiampietri
digiampietri@usp.br
@digiampietri

2023

Construção Incremental

- No método de construção incremental, temos os seguintes passos:

Construção Incremental

- No método de construção incremental, temos os seguintes passos:
  - Inicialmente, resolvemos o problema para um subconjunto dos elementos da entrada

Construção Incremental

- No método de construção incremental, temos os seguintes passos:
  - Inicialmente, resolvemos o problema para um subconjunto dos elementos da entrada
  - Então adicionamos os demais elementos um a um

Construção Incremental

- No método de construção incremental, temos os seguintes passos:
  - Inicialmente, resolvemos o problema para um subconjunto dos elementos da entrada
  - Então adicionamos os demais elementos um a um
  - Em muitos casos, se os elementos forem adicionados em uma ordem ruim, o algoritmo não será eficiente.

Construção Incremental

- No método de construção incremental, temos os seguintes passos:
  - Inicialmente, resolvemos o problema para um subconjunto dos elementos da entrada
  - Então adicionamos os demais elementos um a um
  - Em muitos casos, se os elementos forem adicionados em uma ordem ruim, o algoritmo não será eficiente.
- Exemplo:
  - Cálculo recursivo de $n!$

Divisão e Conquista

- Na divisão e conquista, o problema principal é decomposto em subproblemas menores

Divisão e Conquista

- Na divisão e conquista, o problema principal é decomposto em subproblemas menores
  - Combinando então as respostas de cada um desses subproblemas

Divisão e Conquista

- Na divisão e conquista, o problema principal é decomposto em subproblemas menores
  - Combinando então as respostas de cada um desses subproblemas
- É mais um paradigma de projeto de algoritmos baseado no princípio da indução

Divisão e Conquista

- Na divisão e conquista, o problema principal é decomposto em subproblemas menores
  - Combinando então as respostas de cada um desses subproblemas
- 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 para Conquistar

Divisão e Conquista

Dividir para Conquistar

- **Dividir** o problema em determinado número de subproblemas

Divisão e Conquista

Dividir para Conquistar

- **Dividir** o problema em determinado número de subproblemas
- **Conquistar** os subproblemas, resolvendo-os recursivamente

Divisão e Conquista

Dividir para Conquistar

- **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 (caso base)

Divisão e Conquista

Dividir para Conquistar

- **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 (caso base)
- **Combinar** as soluções fornecidas pelos subproblemas, a fim de produzir a solução para o problema original

Divisão e Conquista

Dividir para Conquistar

- A busca binária recursiva utiliza essa técnica?

```
Entrada: arranjo arr, elemento x

Se o arranjo tiver 1 elemento,
                    compare com x

Se x=arr[meio], meio é o índice
           do elemento procurado
Se x<arr[meio], repete a busca
   no subarranjo de ini a meio-1
Senão
    repete a busca no subarranjo
         de meio+1 ao fim de arr
```

Divisão e Conquista

Dividir para Conquistar

- A busca binária recursiva utiliza essa técnica?
- Dividir:
  - Divide o problema em sub-problemas?

```
Entrada: arranjo arr, elemento x

Se o arranjo tiver 1 elemento,
                  compare com x

Se x=arr[meio], meio é o índice
           do elemento procurado
Se x<arr[meio], repete a busca
   no subarranjo de ini a meio-1
Senão
    repete a busca no subarranjo
         de meio+1 ao fim de arr
```

Divisão e Conquista

Dividir para Conquistar

- A busca binária recursiva utiliza essa técnica?
- Dividir:
  - Divide o problema em sub-problemas?

```
Entrada: arranjo arr, elemento x

Se o arranjo tiver 1 elemento,
                  compare com x

Se x=arr[meio], meio é o índice
           do elemento procurado
Se x<arr[meio], repete a busca
   no subarranjo de ini a meio-1
Senão
    repete a busca no subarranjo
         de meio+1 ao fim de arr
```

Divisão e Conquista

Dividir para Conquistar

- A busca binária recursiva utiliza essa técnica?
- Dividir:
  - Divide o problema em sub-problemas?
- Conquistar:
  - Resolve os sub-problemas recursivamente?

```
Entrada: arranjo arr, elemento x

Se o arranjo tiver 1 elemento,
                   compare com x

Se x=arr[meio], meio é o índice
            do elemento procurado
Se x<arr[meio], repete a busca
   no subarranjo de ini a meio-1
Senão
    repete a busca no subarranjo
          de meio+1 ao fim de arr
```

Dividir para Conquistar

- A busca binária recursiva utiliza essa técnica?
- Dividir:
  - Divide o problema em sub-problemas?
- Conquistar:
  - Resolve os sub-problemas recursivamente?

```
Entrada: arranjo arr, elemento x

Se o arranjo tiver 1 elemento,
                  compare com x


Se x=arr[meio], meio é o índice
           do elemento procurado
Se x<arr[meio], repete a busca
   no subarranjo de ini a meio-1
Senão
    repete a busca no subarranjo
         de meio+1 ao fim de arr
```

Divisão e Conquista

Dividir para Conquistar

- Combinar:
  - Forma a solução final a partir da combinação das soluções dos sub-problemas?

```
Entrada: arranjo arr, elemento x

Se o arranjo tiver 1 elemento,
                  compare com x

Se x=arr[meio], meio é o índice
           do elemento procurado
Se x<arr[meio], repete a busca
   no subarranjo de ini a meio-1
Senão
    repete a busca no subarranjo
         de meio+1 ao fim de arr
```

Divisão e Conquista

Dividir para Conquistar

- Combinar:
  - Forma a solução final a partir da combinação das soluções dos sub-problemas?

```
Entrada: arranjo arr, elemento x

Se o arranjo tiver 1 elemento,
                  compare com x

Se x=arr[meio], meio é o índice
           do elemento procurado
Se x<arr[meio], repete a busca
   no subarranjo de ini a meio-1
Senão
    repete a busca no subarranjo
         de meio+1 ao fim de arr
```

Divisão e Conquista

Dividir para Conquistar

- 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

```
Entrada: arranjo arr, elemento x

Se o arranjo tiver 1 elemento,
                  compare com x

Se x=arr[meio], meio é o índice
           do elemento procurado
Se x<arr[meio], repete a busca
   no subarranjo de ini a meio-1
Senão
    repete a busca no subarranjo
         de meio+1 ao fim de arr
```

Ex: Exponenciação

Solução 1: Indução Fraca

- Calcule a^n para todo real a e inteiro $n \geq 0$

Ex: Exponenciação

Solução 1: Indução Fraca

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

Ex: Exponenciação

Solução 1: Indução Fraca

- Calcule a^n para todo real a e inteiro $n \geq 0$
- Caso base:
  - $n = 0 \Rightarrow a^0 = 1$

Ex: Exponenciação

Solução 1: Indução Fraca

- Calcule a^n para todo real a e inteiro $n \geq 0$
- Caso base:
  - $n = 0 \Rightarrow a^0 = 1$
- Hipótese de indução:

Ex: Exponenciação

Solução 1: Indução Fraca

- Calcule a^n para todo real a e inteiro $n \geq 0$
- Caso base:
  - $n = 0 \Rightarrow a^0 = 1$
- Hipótese de indução:
  - Para qualquer inteiro $n > 0$ e real a sei calcular a^{n-1}

Ex: Exponenciação

Solução 1: Indução Fraca

- Calcule a^n para todo real a e inteiro $n \geq 0$
- Caso base:
  - $n = 0 \Rightarrow a^0 = 1$
- Hipótese de indução:
  - Para qualquer inteiro $n > 0$ e real a sei calcular a^{n-1}
- Passo:

Ex: Exponenciação

Solução 1: Indução Fraca

- Calcule a^n para todo real a e inteiro $n \geq 0$
- Caso base:
  - $n = 0 \Rightarrow a^0 = 1$
- Hipótese de indução:
  - Para qualquer inteiro $n > 0$ e real a sei calcular a^{n-1}
- Passo:
  - $a^n = a \times a^{n-1}$. Pela H.I., sei calcular a^{n-1}, logo sei calcular a^n

Ex: Exponenciação

Solução 1: Indução Fraca

- Então...

```
double exp(double a, int n)
{
  if (n == 0) return 1;
  return(a * exp(a,n-1));
}
```

Ex: Exponenciação

Solução 1: Indução Fraca

- Então...
- E qual a complexidade dessa solução?

```
double exp(double a, int n)
{
  if (n == 0) return 1;
  return(a * exp(a,n-1));
}
```

$$T(n) = \begin{cases} & \text{se } n = 0 \\ & \text{para } n \geq 1 \end{cases}$$

Ex: Exponenciação

Solução 1: Indução Fraca

- Então...
- E qual a complexidade dessa solução?

```
double exp(double a, int n)
{
  if (n == 0) return 1;
  return(a * exp(a,n-1));
}
```

$$T(n) = \begin{cases} 0 & \text{se } n = 0 \\ & \text{para } n \geq 1 \end{cases}$$

Ex: Exponenciação

Solução 1: Indução Fraca

- Então...
- E qual a complexidade dessa solução?

```
double exp(double a, int n)
{
  if (n == 0) return 1;
  return(a * exp(a,n-1));
}
```

$$T(n) = \begin{cases} 0 & \text{se } n = 0 \\ T(n-1) & \text{para } n \geq 1 \end{cases}$$

Ex: Exponenciação

Solução 1: Indução Fraca

- Então...
- E qual a complexidade dessa solução?

```
double exp(double a, int n)
{
  if (n == 0) return 1;
  return(a * exp(a,n-1));
}
```

$$T(n) = \begin{cases} 0 & \text{se } n = 0 \\ T(n-1) + 1 & \text{para } n \geq 1 \end{cases}$$

Ex: Exponenciação

Solução 1: Indução Fraca

- E...

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

```
double exp(double a, int n)
{
  if (n == 0) return(1);
  return(a * exp(a,n-1));
}
```

Ex: Exponenciação

Solução 1: Indução Fraca

- E...

$$
\begin{aligned}
T(n) &= T(n-1)+1 \\
&= T(0)+\sum_{i=1}^{n} 1
\end{aligned}
$$

```
double exp(double a, int n)
{
  if (n == 0) return(1);
  return(a * exp(a,n-1));
}
```

Ex: Exponenciação

Solução 1: Indução Fraca

- E...

$$
\begin{aligned}
T(n) &= T(n-1)+1 \\
&= T(0)+\sum_{i=1}^{n} 1 \\
&= 0+n
\end{aligned}
$$

```
double exp(double a, int n)
{
  if (n == 0) return(1);
  return(a * exp(a,n-1));
}
```

Ex: Exponenciação

Solução 1: Indução Fraca

- E...

$$
\begin{aligned}
T(n) &= T(n-1)+1 \\
&= T(0)+\sum_{i=1}^{n} 1 \\
&= 0+n \\
&= n
\end{aligned}
$$

```
double exp(double a, int n)
{
  if (n == 0) return(1);
  return(a * exp(a,n-1));
}
```

Ex: Exponenciação

Solução 2: Indução Forte

- Calcule a^n para todo real a e inteiro $n \geq 0$

Ex: Exponenciação

Solução 2: Indução Forte

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

Ex: Exponenciação

Solução 2: Indução Forte

- Calcule a^n para todo real a e inteiro $n \geq 0$
- Caso base:
  - $n = 0 \Rightarrow a^0 = 1$

Ex: Exponenciação

Solução 2: Indução Forte

- Calcule a^n para todo real a e inteiro $n \geq 0$
- Caso base:
  - $n = 0 \Rightarrow a^0 = 1$
- Hipótese de indução:

Ex: Exponenciação

Solução 2: Indução Forte

- Calcule a^n para todo real a e inteiro $n \geq 0$
- Caso base:
  - $n = 0 \Rightarrow a^0 = 1$
- Hipótese de indução:
  - Para qualquer inteiro $n \geq 0$ e real a sei calcular $a^k, 0 \leq k \leq n - 1$

Ex: Exponenciação

Solução 2: Indução Forte

- Calcule a^n para todo real a e inteiro $n \geq 0$
- Caso base:
  - $n = 0 \Rightarrow a^0 = 1$
- Hipótese de indução:
  - Para qualquer inteiro $n \geq 0$ e real a sei calcular $a^k, 0 \leq k \leq n-1$
- Passo:

Ex: Exponenciação

Solução 2: Indução Forte

- Calcule a^n para todo real a e inteiro $n \geq 0$
- Caso base:
  - $n = 0 \Rightarrow a^0 = 1$
- Hipótese de indução:
  - Para qualquer inteiro $n \geq 0$ e real a sei calcular $a^k, 0 \leq k \leq n-1$
- Passo:
  - Vamos calcular a^n. Como ficaria o cálculo de a^n se soubéssemos $a^{\frac{n}{2}}$ (de fato, $a^{\lfloor \frac{n}{2} \rfloor}$)?

Ex: Exponenciação

Solução 2: Indução Forte

- Passo:

Ex: Exponenciação

Solução 2: Indução Forte

- Passo:
  - Podemos escrever a^n como

$$a^n = \begin{cases} (a^{\lfloor \frac{n}{2} \rfloor})^2, & \text{se } n \text{ par} \\ \end{cases}$$

Ex: Exponenciação

Solução 2: Indução Forte

- Passo:
  - Podemos escrever a^n como

$$a^n = \begin{cases} (a^{\lfloor \frac{n}{2} \rfloor})^2, & \text{se } n \text{ par} \\ a \times (a^{\lfloor \frac{n}{2} \rfloor})^2 & \text{se } n \text{ ímpar} \end{cases}$$

Ex: Exponenciação

Solução 2: Indução Forte

- Passo:
  - Podemos escrever a^n como

$$a^n = \begin{cases} (a^{\lfloor \frac{n}{2} \rfloor})^2, & \text{se } n \text{ par} \\ a \times (a^{\lfloor \frac{n}{2} \rfloor})^2 & \text{se } n \text{ ímpar} \end{cases}$$

  - Pela H.I., sei calcular $a^{\lfloor \frac{n}{2} \rfloor}$, então sei calcular a^n

Ex: Exponenciação

Solução 2: Indução Forte

```
double exp(double a,int n){
  double aux;
  if (n == 0) return 1;
  else {
    aux = exp(a,n/2);
    aux = aux * aux;
    if (n % 2 == 1)
      aux = aux * a;
    return aux;
  }
}
```

Ex: Exponenciação

Solução 2: Indução Forte

- E qual a complexidade disso?

```
double exp(double a,int n){
  double aux;
  if (n == 0) return 1;
  else {
    aux = exp(a,n/2);
    aux = aux * aux;
    if (n % 2 == 1)
      aux = aux * a;
    return aux;
  }
}
```

$$T(n) = \begin{cases} & \text{se } n = 0 \\ & \text{para } n \geq 1 \end{cases}$$

Ex: Exponenciação

Solução 2: Indução Forte

- E qual a complexidade disso?

```
double exp(double a,int n){
  double aux;
  if (n == 0) return 1;
  else {
    aux = exp(a,n/2);
    aux = aux * aux;
    if (n % 2 == 1)
      aux = aux * a;
    return aux;
  }
}
```

$$T(n) = \begin{cases} 0 & \text{se } n = 0 \\ & \text{para } n \geq 1 \end{cases}$$

Ex: Exponenciação

Solução 2: Indução Forte

- E qual a complexidade disso?

```
double exp(double a,int n){
  double aux;
  if (n == 0) return 1;
  else {
    aux = exp(a,n/2);
    aux = aux * aux;
    if (n % 2 == 1)
      aux = aux * a;
    return aux;
  }
}
```

$$T(n) = \begin{cases} 0 & \text{se } n = 0 \\ T(\frac{n}{2}) & \text{para } n \geq 1 \end{cases}$$

Ex: Exponenciação

Solução 2: Indução Forte

- E qual a complexidade disso?

```
double exp(double a,int n){
  double aux;
  if (n == 0) return 1;
  else {
    aux = exp(a,n/2);
    aux = aux * aux;
    if (n % 2 == 1)
      aux = aux * a;
    return aux;
  }
}
```

$$T(n) = \begin{cases} 0 & \text{se } n = 0 \\ T(\frac{n}{2}) + 2 & \text{para } n \geq 1 \end{cases}$$

Ex: Exponenciação

Solução 2: Indução Forte

- Mas $T(n) = T(\frac{n}{2}) + 2$ é também a complexidade da busca binária

Ex: Exponenciação

Solução 2: Indução Forte

- Mas $T(n) = T(\frac{n}{2}) + 2$ é também a complexidade da busca binária
- Que já vimos ser $T(n) = O(log_2(n))$

Ex: Exponenciação

Solução 2: Indução Forte

- Mas $T(n) = T(\frac{n}{2}) + 2$ é também a complexidade da busca binária
- Que já vimos ser $T(n) = O(log_2(n))$
- Então, em sua versão incremental, a exponenciação é $O(n)$, enquanto que em sua versão por divisão e conquista é $O(log_2(n))$

Ex: Exponenciação

Solução 2: Indução Forte

- Mas $T(n) = T(\frac{n}{2}) + 2$ é também a complexidade da busca binária
- Que já vimos ser $T(n) = O(log_2(n))$
- Então, em sua versão incremental, a exponenciação é $O(n)$, enquanto que em sua versão por divisão e conquista é $O(log_2(n))$
  - Lembrando que $n > log_2(n)$, para $n \geq 1$.

Ex: Exponenciação

Solução 2: Indução Forte

- Mas $T(n) = T(\frac{n}{2}) + 2$ é também a complexidade da busca binária
- Que já vimos ser $T(n) = O(log_2(n))$
- Então, em sua versão incremental, a exponenciação é $O(n)$, enquanto que em sua versão por divisão e conquista é $O(log_2(n))$
  - Lembrando que $n > log_2(n)$, para $n \geq 1$.
  - Mas isso, claro, vai depender das constantes multiplicativas

Ex: Mínimo e Máximo

Solução 1: Indução Fraca

- Dado um arranjo S de $n \geq 2$ números reais, determine o maior e o menor elemento de S

Ex: Mínimo e Máximo

Solução 1: Indução Fraca

- Dado um arranjo S de $n \geq 2$ números reais, determine o maior e o menor elemento de S
- Caso base:

Ex: Mínimo e Máximo

Solução 1: Indução Fraca

- Dado um arranjo S de $n \geq 2$ números reais, determine o maior e o menor elemento de S
- Caso base:
  - $n = 2$: Compare um com o outro e veja qual o maior

Ex: Mínimo e Máximo

Solução 1: Indução Fraca

- Dado um arranjo S de $n \geq 2$ números reais, determine o maior e o menor elemento de S
- Caso base:
  - $n = 2$: Compare um com o outro e veja qual o maior
- Hipótese de indução:

Ex: Mínimo e Máximo

Solução 1: Indução Fraca

- Dado um arranjo S de $n \geq 2$ números reais, determine o maior e o menor elemento de S
- Caso base:
  - $n = 2$: Compare um com o outro e veja qual o maior
- Hipótese de indução:
  - Sei o maior e o menor dentre os $n - 1$ primeiros elementos do arranjo

Ex: Mínimo e Máximo

Solução 1: Indução Fraca

- Passo:

Ex: Mínimo e Máximo

Solução 1: Indução Fraca

- Passo:
  - Pela H.I., consigo calcular o maior e o menor entre os $n - 1$ primeiros elementos de S

Ex: Mínimo e Máximo

Solução 1: Indução Fraca

- Passo:
  - Pela H.I., consigo calcular o maior e o menor entre os $n-1$ primeiros elementos de S
  - Se o n-ésimo elemento for maior que o maior em $n-1$, então ele é o maior de todos. Senão, o resultado de $n-1$ é o maior

Ex: Mínimo e Máximo

Solução 1: Indução Fraca

- Passo:
  - Pela H.I., consigo calcular o maior e o menor entre os $n-1$ primeiros elementos de S
  - Se o n-ésimo elemento for maior que o maior em $n-1$, então ele é o maior de todos. Senão, o resultado de $n-1$ é o maior
  - Se o n-ésimo elemento for menor que o menor em $n-1$, então ele é o menor de todos. Senão, o resultado de $n-1$ é o menor

Ex: Mínimo e Máximo

Solução 1: Indução Fraca

```
typedef struct {
  int min;
  int max;
} MM;

MM minMax(double s[], int n) {
  MM resp;
  if (n==2) {
    if (s[0]>s[1]) {
      resp.min = s[1];
      resp.max = s[0];
    } else {
      resp.min = s[0];
      resp.max = s[1];
    }
  }
  else {
      resp = minMax(s,n-1);
      if (s[n-1] > resp.max)
            resp.max = s[n-1];
      if (s[n-1] < resp.min)
            resp.min = s[n-1];
  }
  return resp;
}
```

Ex: Mínimo e Máximo

Solução 1: Indução Fraca

```
typedef struct {
  int min;
  int max;
} MM;

MM minMax(double s[], int n) {
  MM resp;
  if (n==2) {
    if (s[0]>s[1]) {
      resp.min = s[1];
      resp.max = s[0];
    } else {
      resp.min = s[0];
      resp.max = s[1];
    }
  }
  else {
      resp = minMax(s,n-1);
      if (s[n-1] > resp.max)
            resp.max = s[n-1];
      if (s[n-1] < resp.min)
            resp.min = s[n-1];
  }
  return resp;
}
```

- E qual a complexidade disso?

Ex: Mínimo e Máximo

Solução 1: Indução Fraca

```
MM minMax(double s[], int n) {
  MM resp;
    if (n==2) {
    if (s[0]>s[1]) {
      resp.min = s[1];
      resp.max = s[0];
    } else {
      resp.min = s[0];
      resp.max = s[1];
    }
  }
else {
     resp = minMax(s,n-1);
     if (s[n-1] > resp.max)
           resp.max = s[n-1];
     if (s[n-1] < resp.min)
           resp.min = s[n-1];
  }
  return resp;
}
```

$$T(n) = \begin{cases} & \text{se } n = 2 \\ & \text{se } n > 2 \end{cases}$$

Ex: Mínimo e Máximo

Solução 1: Indução Fraca

```
MM minMax(double s[], int n) {
  MM resp;
    if (n==2) {
    if (s[0]>s[1]) {
      resp.min = s[1];
      resp.max = s[0];
    } else {
      resp.min = s[0];
      resp.max = s[1];
    }
  }
else {
    resp = minMax(s,n-1);
    if (s[n-1] > resp.max)
          resp.max = s[n-1];
    if (s[n-1] < resp.min)
          resp.min = s[n-1];
  }
  return resp;
}
```

$$T(n) = \begin{cases} 1 & \text{se } n = 2 \\ & \text{se } n > 2 \end{cases}$$

Ex: Mínimo e Máximo

Solução 1: Indução Fraca

```
MM minMax(double s[], int n) {
  MM resp;
    if (n==2) {
    if (s[0]>s[1]) {
      resp.min = s[1];
      resp.max = s[0];
    } else {
      resp.min = s[0];
      resp.max = s[1];
    }
  }
```

```
else {
      resp = minMax(s,n-1);
      if (s[n-1] > resp.max)
            resp.max = s[n-1];
      if (s[n-1] < resp.min)
            resp.min = s[n-1];
  }
  return resp;
}
```

$$T(n) = \begin{cases} 1 & \text{se } n = 2 \\ T(n-1) & \text{se } n > 2 \end{cases}$$

Ex: Mínimo e Máximo

Solução 1: Indução Fraca

```
MM minMax(double s[], int n) {
  MM resp;
    if (n==2) {
    if (s[0]>s[1]) {
      resp.min = s[1];
      resp.max = s[0];
    } else {
      resp.min = s[0];
      resp.max = s[1];
    }
  }
  else {
      resp = minMax(s,n-1);
      if (s[n-1] > resp.max)
            resp.max = s[n-1];
      if (s[n-1] < resp.min)
            resp.min = s[n-1];
    }
    return resp;
  }
```

$$T(n) = \begin{cases} 1 & \text{se } n = 2 \\ T(n-1) + 2 & \text{se } n > 2 \end{cases}$$

Ex: Mínimo e Máximo

Solução 1: Indução Fraca

```
MM minMax(double s[], int n) {
  MM resp;
  if (n==2) {
    if (s[0]>s[1]) {
      resp.min = s[1];
      resp.max = s[0];
    } else {
      resp.min = s[0];
      resp.max = s[1];
    }
  }
  else {
      resp = minMax(s,n-1);
      if (s[n-1] > resp.max)
            resp.max = s[n-1];
      if (s[n-1] < resp.min)
            resp.min = s[n-1];
    }
    return resp;
  }
```

E $T(n) \in O(n)$ (idem à busca sequencial)

Ex: Mínimo e Máximo

Solução 2: Indução Forte

- Dado um arranjo S de $n \geq 2$ números reais, determine o maior e o menor elemento de S

Ex: Mínimo e Máximo

Solução 2: Indução Forte

- Dado um arranjo S de $n \geq 2$ números reais, determine o maior e o menor elemento de S
- Caso base:

Ex: Mínimo e Máximo

Solução 2: Indução Forte

- Dado um arranjo S de $n \geq 2$ números reais, determine o maior e o menor elemento de S
- Caso base:
  - $n = 2$: Compare um com o outro e veja qual o maior

Ex: Mínimo e Máximo

Solução 2: Indução Forte

- Dado um arranjo S de $n \geq 2$ números reais, determine o maior e o menor elemento de S
- Caso base:
  - $n = 2$: Compare um com o outro e veja qual o maior
- Hipótese de indução:

Ex: Mínimo e Máximo

Solução 2: Indução Forte

- Dado um arranjo S de $n \geq 2$ números reais, determine o maior e o menor elemento de S
- Caso base:
  - $n = 2$: Compare um com o outro e veja qual o maior
- Hipótese de indução:
  - Sei encontrar o maior e o menor elemento em sub-arranjos de tamanho $2 \leq k \leq n - 1$

Ex: Mínimo e Máximo

Solução 2: Indução Forte

- Passo:

Ex: Mínimo e Máximo

Solução 2: Indução Forte

- Passo:
  - Vejamos para um arranjo de n elementos. Pela H.I., sei o menor o maior elemento em sub-arranjos de tamanho $\lceil \frac{n}{2} \rceil$

Ex: Mínimo e Máximo

Solução 2: Indução Forte

- Passo:
  - Vejamos para um arranjo de n elementos. Pela H.I., sei o menor o maior elemento em sub-arranjos de tamanho $\lceil \frac{n}{2} \rceil$
  - Então, se n for par, o maior elemento será o maior dentre as respostas de $\lfloor \frac{n}{2} \rfloor$ que correspondem às duas metades do arranjo. O mesmo vale para o menor.

Ex: Mínimo e Máximo

Solução 2: Indução Forte

- Passo:
  - Vejamos para um arranjo de n elementos. Pela H.I., sei o menor o maior elemento em sub-arranjos de tamanho $\lceil \frac{n}{2} \rceil$
  - Então, se n for par, o maior elemento será o maior dentre as respostas de $\lfloor \frac{n}{2} \rfloor$ que correspondem às duas metades do arranjo. O mesmo vale para o menor.
  - Se n for ímpar, o maior elemento será o maior dentre as respostas de $\lfloor \frac{n}{2} \rfloor$ e $\lceil \frac{n}{2} \rceil$ que correspondem às duas partes do arranjo. O mesmo vale para o menor.

Ex: Mínimo e Máximo

Solução 2: Indução Forte

- E qual a complexidade desse algoritmo?

Ex: Mínimo e Máximo

Solução 2: Indução Forte

- E qual a complexidade desse algoritmo?

$$T(n) = \begin{cases} O(1) & \text{se } n = 2 \\ 2T(\frac{n}{2}) + O(1) & \text{se } n > 2 \end{cases}$$

Ex: Mínimo e Máximo

Solução 2: Indução Forte

- E qual a complexidade desse algoritmo?

$$T(n) = \begin{cases} O(1) & \text{se } n = 2 \\ 2T(\frac{n}{2}) + O(1) & \text{se } n > 2 \end{cases}$$

- Temos então que $T(n) \in O(n)$

Ex: Mínimo e Máximo

Solução 2: Indução Forte

- E qual a complexidade desse algoritmo?

$$T(n) = \begin{cases} O(1) & \text{se } n = 2 \\ 2T(\frac{n}{2}) + O(1) & \text{se } n > 2 \end{cases}$$

- Temos então que $T(n) \in O(n)$
  - A demonstração fica por sua conta

Ex: Mínimo e Máximo

Solução 2: Indução Forte

- E qual a complexidade desse algoritmo?

$$T(n) = \begin{cases} O(1) & \text{se } n = 2 \\ 2T(\frac{n}{2}) + O(1) & \text{se } n > 2 \end{cases}$$

- Temos então que $T(n) \in O(n)$
  - A demonstração fica por sua conta
- Não houve melhora em relação à versão anterior

Ex: Mínimo e Máximo

Solução 2: Indução Forte

- E qual a complexidade desse algoritmo?

$$T(n) = \begin{cases} O(1) & \text{se } n = 2 \\ 2T(\frac{n}{2}) + O(1) & \text{se } n > 2 \end{cases}$$

- Temos então que $T(n) \in O(n)$
  - A demonstração fica por sua conta
- Não houve melhora em relação à versão anterior
  - Você esperava ser menor que $O(n)$?

Ex: Mínimo e Máximo

Código - Indução Forte

```
MM minMax(double s[], int ini, int fim) {
  MM resp, resp2;
  int meio;
  if (ini==fim) {
      resp.min = s[ini];
      resp.max = s[ini];
  } else {
    meio = (ini + fim)/2;
    resp = minMax(s,ini, meio);
    resp2 = minMax(s,meio+1,fim);
    if (resp.min>resp2.min)
          resp.min = resp2.min;
    if (resp.max<resp2.max)
          resp.max = resp2.max;
  }
  return resp;
}
```

Ex: Mínimo e Máximo

Código - Indução Forte

```
MM minMax(double s[], int ini, int fim) {
  MM resp, resp2;
  int meio;
  if (ini==fim) {
      resp.min = s[ini];
      resp.max = s[ini];
  } else {
    meio = (ini + fim)/2;
    resp = minMax(s,ini, meio);
    resp2 = minMax(s,meio+1,fim);
    if (resp.min>resp2.min)
          resp.min = resp2.min;
    if (resp.max<resp2.max)
          resp.max = resp2.max;
  }
  return resp;
}
```

$$T(n) = \begin{cases} \\ \end{cases}$$

Ex: Mínimo e Máximo

Código - Indução Forte

```
MM minMax(double s[], int ini, int fim) {
  MM resp, resp2;
  int meio;
  if (ini==fim) {
      resp.min = s[ini];
      resp.max = s[ini];
  } else {
    meio = (ini + fim)/2;
    resp = minMax(s,ini, meio);
    resp2 = minMax(s,meio+1,fim);
    if (resp.min>resp2.min)
          resp.min = resp2.min;
    if (resp.max<resp2.max)
          resp.max = resp2.max;
  }
  return resp;
}
```

$$T(n) = \begin{cases} & \text{se } n = 1 \\ & \end{cases}$$

Ex: Mínimo e Máximo

Código - Indução Forte

```
MM minMax(double s[], int ini, int fim) {
  MM resp, resp2;
  int meio;
  if (ini==fim) {
      resp.min = s[ini];
      resp.max = s[ini];
  } else {
    meio = (ini + fim)/2;
    resp = minMax(s,ini, meio);
    resp2 = minMax(s,meio+1,fim);
    if (resp.min>resp2.min)
          resp.min = resp2.min;
    if (resp.max<resp2.max)
          resp.max = resp2.max;
  }
  return resp;
}
```

$$T(n) = \begin{cases} 0 & \text{se } n = 1 \\ & \end{cases}$$

Ex: Mínimo e Máximo

Código - Indução Forte

```
MM minMax(double s[], int ini, int fim) {
  MM resp, resp2;
  int meio;
  if (ini==fim) {
      resp.min = s[ini];
      resp.max = s[ini];
  } else {
    meio = (ini + fim)/2;
    resp = minMax(s,ini, meio);
    resp2 = minMax(s,meio+1,fim);
    if (resp.min>resp2.min)
          resp.min = resp2.min;
    if (resp.max<resp2.max)
          resp.max = resp2.max;
  }
  return resp;
}
```

$$T(n) = \begin{cases} 0 & \text{se } n = 1 \\ 2 * T(\frac{n}{2}) + 2 & \text{se } n > 1 \end{cases}$$

Ex: Mínimo e Máximo

Código - Indução Forte

```
MM minMax(double s[], int ini, int fim) {
  MM resp, resp2;
  int meio;
  if (fim - ini < 2){
    if (ini==fim) {
        resp.min = s[ini];
        resp.max = s[ini];
    }else{
      if (s[ini]>s[fim]) {
        resp.min = s[fim];
        resp.max = s[ini];
      } else {
        resp.min = s[ini];
        resp.max = s[fim];
      }
    }
  }
  else {
    meio = (ini + fim)/2;
    resp = minMax(s,ini, meio);
    resp2 = minMax(s,meio+1,fim);
    if (resp.min>resp2.min)
          resp.min = resp2.min;
    if (resp.max<resp2.max)
          resp.max = resp2.max;
  }
  return resp;
}
```

Ex: Mínimo e Máximo

Código - Indução Forte

```
MM minMax(double s[], int ini, int fim) {
  MM resp, resp2;
  int meio;
  if (fim - ini < 2){
    if (ini==fim) {
        resp.min = s[ini];
        resp.max = s[ini];
    }else{
      if (s[ini]>s[fim]) {
        resp.min = s[fim];
        resp.max = s[ini];
      } else {
        resp.min = s[ini];
        resp.max = s[fim];
      }
    }
  }
  else {
    meio = (ini + fim)/2;
    resp = minMax(s,ini, meio);
    resp2 = minMax(s,meio+1,fim);
    if (resp.min>resp2.min)
          resp.min = resp2.min;
    if (resp.max<resp2.max)
          resp.max = resp2.max;
  }
  return resp;
}
```

Ex: Mínimo e Máximo

Código - Indução Forte

```
MM minMax(double s[], int ini, int fim) {
  MM resp, resp2;
  int meio;
  if (fim - ini < 2){
    if (ini==fim) {
        resp.min = s[ini];
        resp.max = s[ini];
    }else{
      if (s[ini]>s[fim]) {
        resp.min = s[fim];
        resp.max = s[ini];
      } else {
        resp.min = s[ini];
        resp.max = s[fim];
      }
    }
  }
  else {
    meio = (ini + fim)/2;
    resp = minMax(s,ini, meio);
    resp2 = minMax(s,meio+1,fim);
    if (resp.min>resp2.min)
          resp.min = resp2.min;
    if (resp.max<resp2.max)
          resp.max = resp2.max;
  }
  return resp;
}
```

$$T(n) = \begin{cases} 0 & \text{se } n = 1 \\ 1 & \text{se } n = 2 \\ 2 * T(\frac{n}{2}) + 2 & \text{se } n > 2 \end{cases}$$

Divisão e Conquista

- A complexidade de tempo de algoritmos divisão e conquista, para uma entrada de tamanho n, é:

Divisão e Conquista

- A complexidade de tempo de algoritmos divisão e conquista, para uma entrada de tamanho n, é:
  - $T(n) = Dividir(n) + Conquistar(n) + Combinar(n)$

Divisão e Conquista

- A complexidade de tempo de algoritmos divisão e conquista, para uma 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 $T(n) = \Theta(1)$ (caso base)

Divisão e Conquista

- A complexidade de tempo de algoritmos divisão e conquista, para uma 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 $T(n) = \Theta(1)$ (caso base)
- Vamos supor que o problema seja dividido em a subproblemas, cada um com $\frac{1}{b}$ do tamanho original

Divisão e Conquista

- A complexidade de tempo de algoritmos divisão e conquista, para uma 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 $T(n) = \Theta(1)$ (caso base)
- Vamos supor que o problema seja dividido em a subproblemas, cada um com $\frac{1}{b}$ do tamanho original
- Como fica a "Conquista" $C(n)$?

Divisão e Conquista

- A complexidade de tempo de algoritmos divisão e conquista, para uma 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 $T(n) = \Theta(1)$ (caso base)
- Vamos supor que o problema seja dividido em a subproblemas, cada um com $\frac{1}{b}$ do tamanho original
- Como fica a "Conquista" $C(n)$?
  - $C(n) = aT\left(\frac{n}{b}\right)$

Divisão e Conquista

- Se levamos $D(n)$ para dividir o problema em subproblemas e $C(n)$ para combinar suas soluções, então tem-se a recorrência $T(n)$:

Divisão e Conquista

- Se levamos $D(n)$ para dividir o problema em subproblemas e $C(n)$ para combinar suas soluções, então tem-se a recorrência $T(n)$:

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

Divisão e Conquista

- A expressão geral de recorrência de um algoritmo de divisão e conquista é então

$$T(n) = aT(\frac{n}{b}) + f(n)$$

Divisão e Conquista

- A expressão geral de recorrência de um algoritmo de divisão e conquista é então

$$T(n) = aT(\frac{n}{b}) + f(n)$$

- Onde:

Divisão e Conquista

- A expressão geral de recorrência de um algoritmo de divisão e conquista é então

$$T(n) = aT(\frac{n}{b}) + f(n)$$

- Onde:
  - a representa o número de subproblemas obtidos na divisão

Divisão e Conquista

- A expressão geral de recorrência de um algoritmo de divisão e conquista é então

$$T(n) = aT(\frac{n}{b}) + f(n)$$

- Onde:
  - a representa o número de subproblemas obtidos na divisão
  - $\frac{n}{b}$ representa seu tamanho

Divisão e Conquista

- A expressão geral de recorrência de um algoritmo de divisão e conquista é então

$$T(n) = aT(\frac{n}{b}) + f(n)$$

- Onde:
  - a representa o número de subproblemas obtidos na divisão
  - $\frac{n}{b}$ representa seu tamanho
  - $f(n)$ é a função que dá a complexidade das etapas de divisão e combinação

Divisão e Conquista

- A expressão geral de recorrência de um algoritmo de divisão e conquista é então

$$T(n) = aT(\frac{n}{b}) + f(n)$$

- Onde:
  - a representa o número de subproblemas obtidos na divisão
  - $\frac{n}{b}$ 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

- Relação:

$$T(n) = aT(\frac{n}{b}) + f(n)$$

Divisão e Conquista

- Relação:

$$T(n) = aT(\frac{n}{b}) + f(n)$$

- Haveria um meio mais fácil de calcularmos a complexidade desse tipo de expressão?

Divisão e Conquista

- Relação:

$$T(n) = aT(\frac{n}{b}) + f(n)$$

- Haveria um meio mais fácil de calcularmos a complexidade desse tipo de expressão?

O Teorema Mestre

Divisão e Conquista

Teorema Mestre

- 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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre: Exemplo

- $T(n) = 9T(n/3) + n$
  - $a = 9, b = 3, f(n) = n$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 9T(n/3) + n$
  - $a = 9, b = 3, f(n) = n$
- Testamos as opções

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 9T(n/3) + n$
  - $a = 9, b = 3, f(n) = n$
- Testamos as opções
  - Note que $n^{log_3 9} = n^2$ 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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 9T(n/3) + n$
  - $a = 9, b = 3, f(n) = n$
- Testamos as opções
  - Note que $n^{log_3 9} = n^2$ 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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

- Temos que $n \in O(n^{log_3 9 - \epsilon})$, para $\epsilon = 1$ (Caso 1)

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 9T(n/3) + n$
  - $a = 9, b = 3, f(n) = n$
- Testamos as opções
  - Note que $n^{log_3 9} = n^2$ 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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

  - Temos que $n \in O(n^{log_3 9 - \epsilon})$, para $\epsilon = 1$ (Caso 1)
  - Ou seja, $n \in O(n^{2-1}) = O(n)$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 9T(n/3) + n$
  - $a = 9, b = 3, f(n) = n$
- Testamos as opções
  - Note que $n^{log_3 9} = n^2$ 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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

  - Temos que $n \in O(n^{log_3 9 - \epsilon})$, para $\epsilon = 1$ (Caso 1)
  - Ou seja, $n \in O(n^{2-1}) = O(n)$
  - Então, pelo teorema, $T(n) \in \Theta(n^2)$

Divisão e Conquista

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = T(2n/3) + 1$
  - $a = 1, b = \frac{3}{2}, f(n) = 1$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = T(2n/3) + 1$
  - $a = 1, b = \frac{3}{2}, f(n) = 1$
- Testamos a 1ª opção:

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = T(2n/3) + 1$
  - $a = 1, b = \frac{3}{2}, f(n) = 1$
- Testamos a 1ª opção:
  - $n^{log_{3/2}1} = n^0 = 1$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre: Exemplo

- $T(n) = T(2n/3) + 1$
  - $a = 1, b = \frac{3}{2}, f(n) = 1$
- Testamos a 1ª opção:
  - $n^{log_{3/2}1} = n^0 = 1$
  - $f(n) = O(n^{log_b a - \epsilon})$?

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = T(2n/3) + 1$
  - $a = 1, b = \frac{3}{2}, f(n) = 1$
- Testamos a 1ª opção:
  - $n^{log_{3/2} 1} = n^0 = 1$
  - $f(n) = O(n^{log_b a - \epsilon})$?
  - Se for, então $1 = O(n^{0-\epsilon})$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = T(2n/3) + 1$
  - $a = 1, b = \frac{3}{2}, f(n) = 1$
- Testamos a 1ª opção:
  - $n^{log_{3/2}1} = n^0 = 1$
  - $f(n) = O(n^{log_b a - \epsilon})$?
  - Se for, então $1 = O(n^{0-\epsilon})$
  - $\Rightarrow 1 = O(\frac{1}{n^\epsilon}), \epsilon > 0$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = T(2n/3) + 1$
  - $a = 1, b = \frac{3}{2}, f(n) = 1$
- Testamos a 1ª opção:
  - $n^{log_{3/2} 1} = n^0 = 1$
  - $f(n) = O(n^{log_b a - \epsilon})$?
  - Se for, então $1 = O(n^{0-\epsilon})$
  - $\Rightarrow 1 = O(\frac{1}{n^\epsilon}), \epsilon > 0$
  - Não, pois isso implica $1 \leq c\frac{1}{n^\epsilon}, \epsilon > 0$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = T(2n/3) + 1$
  - $a = 1, b = \frac{3}{2}, f(n) = 1$
- Testamos a 2ª opção:

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre: Exemplo

- $T(n) = T(2n/3) + 1$
  - $a = 1, b = \frac{3}{2}, f(n) = 1$
- Testamos a 2ª opção:
  - $n^{log_{3/2}1} = n^0 = 1$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = T(2n/3) + 1$
  - $a = 1, b = \frac{3}{2}, f(n) = 1$
- Testamos a 2ª opção:
  - $n^{log_{3/2} 1} = n^0 = 1$
  - Pelo Caso 2, temos que $1 \in \Theta(n^0) \Rightarrow 1 \in \Theta(1)$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = T(2n/3) + 1$
  - $a = 1, b = \frac{3}{2}, f(n) = 1$
- Testamos a 2ª opção:
  - $n^{log_{3/2} 1} = n^0 = 1$
  - Pelo Caso 2, temos que $1 \in \Theta(n^0) \Rightarrow 1 \in \Theta(1)$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

- Então $T(n) \in \Theta(n^0 log\ n) \Rightarrow T(n) \in \Theta(log\ n)$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$
- Testamos a 1ª opção:

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$
- Testamos a 1ª opção:
  - $n^{log_b a} = n^{log_4 3} = n^{0,793}$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$
- Testamos a 1ª opção:
  - $n^{log_b a} = n^{log_4 3} = n^{0{,}793}$
  - $f(n) = O(n^{log_b a - \epsilon})$?

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$
- Testamos a 1ª opção:
  - $n^{log_b a} = n^{log_4 3} = n^{0,793}$
  - $f(n) = O(n^{log_b a - \epsilon})$? Se for, então $n \times log\ n = O(n^{log_4 3 - \epsilon})$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$
- Testamos a 1ª opção:
  - $n^{log_b a} = n^{log_4 3} = n^{0{,}793}$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

  - $f(n) = O(n^{log_b a - \epsilon})$? Se for, então $n \times log\ n = O(n^{log_4 3 - \epsilon})$
  - $\Rightarrow n \times log\ n = O(n^{0{,}793 - \epsilon}), \epsilon > 0$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$
- Testamos a 1ª opção:
  - $n^{log_b a} = n^{log_4 3} = n^{0,793}$
  - $f(n) = O(n^{log_b a - \epsilon})$? Se for, então $n \times log\ n = O(n^{log_4 3 - \epsilon})$
  - $\Rightarrow n \times log\ n = O(n^{0,793 - \epsilon}), \epsilon > 0$
  - Não, pois isso implica $n \times log\ n = O(n^c), c < 1$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$
- Testamos a 2ª opção:

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$
- Testamos a 2ª opção:
  - $f(n) = \Theta(n^{log_b a})$?

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$
- Testamos a 2ª opção:
  - $f(n) = \Theta(n^{log_b a})$?
  - Se for, então $n \times log\ n = \Theta(n^{log_4 3}) = \Theta(n^{0,793})$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times \log n$
  - $a = 3, b = 4, f(n) = n \times \log n$
- Testamos a 2ª opção:
  - $f(n) = \Theta(n^{\log_b a})$?
  - Se for, então $n \times \log n = \Theta(n^{\log_4 3}) = \Theta(n^{0,793})$
  - Não, pois isso implica $n \times \log n = \Theta(n^c), c < 1$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$
- Testamos a 3ª opção:

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times \log n$
  - $a = 3, b = 4, f(n) = n \times \log n$
- Testamos a 3ª opção:
  - $f(n) = \Omega(n^{\log_b a+\epsilon})$?

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$
- Testamos a 3ª opção:
  - $f(n) = \Omega(n^{log_b a+\epsilon})$?
  - Se for, então $n \times log\ n = \Omega(n^{log_4 3+\epsilon}) = \Omega(n^{0,793+\epsilon}), \epsilon > 0$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$
- Testamos a 3ª opção:
  - $f(n) = \Omega(n^{log_b a + \epsilon})$?

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

  - Se for, então $n \times log\ n = \Omega(n^{log_4 3 + \epsilon}) = \Omega(n^{0,793+\epsilon}), \epsilon > 0$
  - Fazendo $\epsilon \approx 0,2$, temos que $n \times log\ n = \Omega(n^1)$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$
- Testamos a 3ª opção:
  - E $af(n/b) \leq cf(n)$?

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$
- Testamos a 3ª opção:
  - É $af(n/b) \leq cf(n)$?
  - $\Rightarrow 3\frac{n}{4} log\ \frac{n}{4} \leq cn\ log\ n$?

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$
- Testamos a 3ª opção:
  - E $af(n/b) \leq cf(n)$?
  - $\Rightarrow 3\frac{n}{4} log\ \frac{n}{4} \leq cn\ log\ n$?
  - $\Rightarrow \frac{3}{4} n\ log\ \frac{n}{4} \leq cn\ log\ n$.

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$
- Testamos a 3ª opção:
  - E $af(n/b) \leq cf(n)$?
  - $\Rightarrow 3\frac{n}{4} log\ \frac{n}{4} \leq cn\ log\ n$?
  - $\Rightarrow \frac{3}{4} n\ log\ \frac{n}{4} \leq cn\ log\ n$. Para $c = \frac{3}{4}$ isso é verdadeiro

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Teorema Mestre: Exemplo

- $T(n) = 3T(n/4) + n \times log\ n$
  - $a = 3, b = 4, f(n) = n \times log\ n$
- Testamos a 3ª opção:
  - E $af(n/b) \leq cf(n)$?
  - $\Rightarrow 3\frac{n}{4} log\ \frac{n}{4} \leq cn\ log\ n$?
  - $\Rightarrow \frac{3}{4} n\ log\ \frac{n}{4} \leq cn\ log\ n$. Para $c = \frac{3}{4}$ isso é verdadeiro
  - Portanto $T(n) = \Theta(n\ log\ n)$

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 todo n suficientemente grande, então $T(n) \in \Theta(f(n))$

Divisão e Conquista

Exemplos onde o Teorema Mestre se aplica

- $T(n) = 4T(n/2) + n \log n, \; T(1) = 1$
  - Caso 1

Divisão e Conquista

Exemplos onde o Teorema Mestre se aplica

- $T(n) = 4T(n/2) + n\,log\,n,\; T(1) = 1$
  - Caso 1
- $T(n) = 2T(n/2) + n,\; T(1) = 1$
  - Caso 2

Divisão e Conquista

Exemplos onde o Teorema Mestre se aplica

- $T(n) = 4T(n/2) + n\,log\,n,\ T(1) = 1$
  - Caso 1
- $T(n) = 2T(n/2) + n,\ T(1) = 1$
  - Caso 2
- $T(n) = T(n/2) + n\,log\,n,\ T(1) = 1$
  - Caso 3

Divisão e Conquista

Exemplos onde o Teorema Mestre não se aplica

- $T(n) = T(n-1) + n \log n, \; T(1) = 1$

Divisão e Conquista

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 $a \geq 1, \; b \leq a$)

Divisão e Conquista

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 $a \geq 1,\ b \leq a$)
- $T(n) = T(\alpha n) + T((1-\alpha)n) + n,\ T(1) = 1$ (para $0 < \alpha < 1$)

Divisão e Conquista

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 $a \geq 1,\ b \leq a$)
- $T(n) = T(\alpha n) + T((1-\alpha)n) + n,\ T(1) = 1$ (para $0 < \alpha < 1$)
- $T(n) = T(n-1) + log\, n,\ T(1) = 1$

Divisão e Conquista

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 $a \geq 1, \; b \leq a$)
- $T(n) = T(\alpha n) + T((1-\alpha)n) + n, \; T(1) = 1$ (para $0 < \alpha < 1$)
- $T(n) = T(n-1) + log \, n, \; T(1) = 1$
- $T(n) = 2T(n/2) + n \, log \, n, \; T(1) = 1$

Divisão e Conquista

Teorema Mestre: Observação

- Trabalhamos até então com relações de recorrência do tipo

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

Divisão e Conquista

Teorema Mestre: Observação

- Trabalhamos até então com relações de recorrência do tipo

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

- Contudo, a relação não está bem definida, pois n/b pode não ser inteiro

Divisão e Conquista

Teorema Mestre: Observação

- Trabalhamos até então com relações de recorrência do tipo
$$T(n) = aT(n/b) + f(n)$$
- Contudo, a relação não está bem definida, pois n/b pode não ser inteiro
- De fato, relaxamos a definição, mas o correto seria usar $T(\lceil n/b \rceil)$ ou $T(\lfloor n/b \rfloor)$

Divisão e Conquista

Teorema Mestre: Observação

- Trabalhamos até então com relações de recorrência do tipo
$$T(n) = aT(n/b) + f(n)$$
- Contudo, a relação não está bem definida, pois n/b pode não ser inteiro
- De fato, relaxamos a definição, mas o correto seria usar $T(\lceil n/b \rceil)$ ou $T(\lfloor n/b \rfloor)$
  - Não importa qual usar, pois isso não afeta o comportamento assintótico da recorrência

Referências

- Ziviani, Nivio. Projeto de Algoritmos: com implementações em Java e C++. Cengage. 2007.
- Cormen, Thomas H., Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford. Introduction to Algorithms. 2a ed. MIT Press, 2001.
- Gersting, Judith L. Fundamentos Matemáticos para a Ciência da Computação. 3a ed. LTC. 1993.

Aula 10 – Divisão e Conquista

Norton T. Roman & Luciano A. Digiampietri
digiampietri@usp.br
@digiampietri

2023