Algoritmos Recursivos e Resolvendo Recorrência

Projeto e Análise de Algoritmo — QXD0041

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Prof. Fabio Dias
fabiodias@ufc.br

Universidade Federal do Ceará

2º semestre/2024

Algoritmos Recursivo

Busca Binária

Dado um vetor v de tamanho n e a chave de busca x, retorna o índice onde a chave se encontra no vetor.

```c
int busca_binaria(int v[], int ini, int fim, int x) {
  if(fim < ini) return -1;
  int i = (fim + ini)/2;
  if(v[i] == x) return i;
  else if(x < v[i]) return busca_binaria(v, ini, i - 1);
  else return return busca_binaria(v, i + 1, fim);
}
```

- Qual o tempo de execução desse algoritmo?

Algoritmos Recursivos

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Recorrência expressam a complexidade de algoritmos recursivos como, por exemplo, os algoritmos de divisão e conquista.

Algoritmos Recursivos

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Recorrência expressam a complexidade de algoritmos recursivos como, por exemplo, os algoritmos de divisão e conquista.
- A função de complexidade recursiva não diz muito sobre o quanto um algoritmo é ou não eficiente.

Algoritmos Recursivos

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Recorrência expressam a complexidade de algoritmos recursivos como, por exemplo, os algoritmos de divisão e conquista.
- A função de complexidade recursiva não diz muito sobre o quanto um algoritmo é ou não eficiente.
- Quando o algoritmo é recursivo, em geral, a função de complexidade, se preocupa em contar as quantidade de chamadas recursivas.

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

Removendo a Recursão

- E preciso saber resolver as recorrências para que possamos efetivamente determinar a complexidade dos algoritmos recursivos.

Removendo a Recursão

- E preciso saber resolver as recorrências para que possamos efetivamente determinar a complexidade dos algoritmos recursivos.
- Resolver uma recorrência significa encontrar uma fórmula fechada para $T(n)$, ou seja, encontrar uma função equivalente a $T(n)$.

Removendo a Recursão

- É preciso saber resolver as recorrências para que possamos efetivamente determinar a complexidade dos algoritmos recursivos.
- Resolver uma recorrência significa encontrar uma fórmula fechada para $T(n)$, ou seja, encontrar uma função equivalente a $T(n)$.
- Não é necessário achar uma solução exata. Basta encontrar uma função $f(n)$ tal que $T(n) = O(f(n))$.

Removendo a Recursão

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- E preciso saber resolver as recorrências para que possamos efetivamente determinar a complexidade dos algoritmos recursivos.
- Resolver uma recorrência significa encontrar uma fórmula fechada para $T(n)$, ou seja, encontrar uma função equivalente a $T(n)$.
- Não é necessário achar uma solução exata. Basta encontrar uma função $f(n)$ tal que $T(n) = O(f(n))$.
- Existem métodos para remover a recursão: Método Iterativo, Árvore de Recorrência e Teorema Mestre.

Método Iterativo

- A idéia desse método é expandir a recorrência e escrevê-la como um somatório de termos que dependem apenas de n e das condições iniciais.

Método Iterativo

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- A idéia desse método é expandir a recorrência e escrevê-la como um somatório de termos que dependem apenas de n e das condições iniciais.
- Será necessário conhecer limitantes ou fórmulas de vários somatórios. **Matemática Discreta!!!!!!!!!!!!!!!**

Método Iterativo - Busca Binária

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

Método Iterativo - Busca Binária

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

- Assuma que n é uma potencia de 2, ou seja, $n = 2^k$, para inteiro k.

Método Iterativo - Busca Binária

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

- Assuma que n é uma potencia de 2, ou seja, $n = 2^k$, para inteiro k.
- Substitua os valores fora da recursão usando a notação assintótica.

Método Iterativo - Busca Binária

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

- Assuma que n é uma potencia de 2, ou seja, $n = 2^k$, para inteiro k.
- Substitua os valores fora da recursão usando a notação assintótica.

$$T(n) = T(\tfrac{n}{2}) + c \qquad \text{Como } T(\tfrac{n}{2}) = T(\tfrac{n}{4}) + c$$

Método Iterativo - Busca Binária

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

- Assuma que n é uma potencia de 2, ou seja, $n = 2^k$, para inteiro k.
- Substitua os valores fora da recursão usando a notação assintótica.

$$\begin{array}{lcll} T(n) & = & T(\frac{n}{2}) + c & \text{Como } T(\frac{n}{2}) = T(\frac{n}{4}) + c \\ & = & T(\frac{n}{4}) + 2c & \text{Como } T(\frac{n}{4}) = T(\frac{n}{8}) + c \end{array}$$

Método Iterativo - Busca Binária

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

- Assuma que n é uma potencia de 2, ou seja, $n = 2^k$, para inteiro k.
- Substitua os valores fora da recursão usando a notação assintótica.

$$\begin{aligned} T(n) &= T(\tfrac{n}{2}) + c && \text{Como } T(\tfrac{n}{2}) = T(\tfrac{n}{4}) + c \\ &= T(\tfrac{n}{4}) + 2c && \text{Como } T(\tfrac{n}{4}) = T(\tfrac{n}{8}) + c \\ &= T(\tfrac{n}{8}) + 3c && \text{Como } T(\tfrac{n}{8}) = T(\tfrac{n}{16}) + c \end{aligned}$$

Método Iterativo - Busca Binária

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

- Assuma que n é uma potencia de 2, ou seja, $n = 2^k$, para inteiro k.
- Substitua os valores fora da recursão usando a notação assintótica.

$$\begin{array}{lcll} T(n) & = & T(\frac{n}{2}) + c & \text{Como } T(\frac{n}{2}) = T(\frac{n}{4}) + c \\ & = & T(\frac{n}{4}) + 2c & \text{Como } T(\frac{n}{4}) = T(\frac{n}{8}) + c \\ & = & T(\frac{n}{8}) + 3c & \text{Como } T(\frac{n}{8}) = T(\frac{n}{16}) + c \\ & = & T(\frac{n}{16}) + 4c & \end{array}$$

Método Iterativo - Busca Binária

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

- Assuma que n é uma potencia de 2, ou seja, $n = 2^k$, para inteiro k.
- Substitua os valores fora da recursão usando a notação assintótica.

$$\begin{array}{lcll} T(n) & = & T(\frac{n}{2}) + c & \text{Como } T(\frac{n}{2}) = T(\frac{n}{4}) + c \\ & = & T(\frac{n}{4}) + 2c & \text{Como } T(\frac{n}{4}) = T(\frac{n}{8}) + c \\ & = & T(\frac{n}{8}) + 3c & \text{Como } T(\frac{n}{8}) = T(\frac{n}{16}) + c \\ & = & T(\frac{n}{16}) + 4c & \\ & ... & ... & \end{array}$$

Método Iterativo - Busca Binária

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

- Assuma que n é uma potencia de 2, ou seja, $n = 2^k$, para inteiro k.
- Substitua os valores fora da recursão usando a notação assintótica.

$$\begin{array}{lcll} T(n) & = & T(\frac{n}{2}) + c & \text{Como } T(\frac{n}{2}) = T(\frac{n}{4}) + c \\ & = & T(\frac{n}{4}) + 2c & \text{Como } T(\frac{n}{4}) = T(\frac{n}{8}) + c \\ & = & T(\frac{n}{8}) + 3c & \text{Como } T(\frac{n}{8}) = T(\frac{n}{16}) + c \\ & = & T(\frac{n}{16}) + 4c & \\ & ... & ... & \\ & = & T(\frac{n}{2^i}) + ic & \text{Na } (i+1)\text{-ésima chamada recursiva.} \end{array}$$

Método Iterativo - Busca Binária

Na $(i+1)$-ésima chamada recursiva:

$$T(n) = T(\frac{n}{2^i}) + ic.$$

- Quando o algoritmo deixa de fazer chamada recursivas??????

Método Iterativo - Busca Binária

Na $(i+1)$-ésima chamada recursiva:

$$T(n) = T(\frac{n}{2^i}) + ic.$$

- Quando o algoritmo deixa de fazer chamada recursivas??????
- As chamadas recursivas finalizam quando ocorre o caso base.

Método Iterativo - Busca Binária

Na $(i+1)$-ésima chamada recursiva:

$$T(n) = T(\frac{n}{2^i}) + ic.$$

- Quando o algoritmo deixa de fazer chamada recursivas??????
- As chamadas recursivas finalizam quando ocorre o caso base.
- Ou seja, ocorre quando $\frac{n}{2^i} = 1$.

Método Iterativo - Busca Binária

Na $(i+1)$-ésima chamada recursiva:

$$T(n) = T(\frac{n}{2^i}) + ic.$$

- Quando o algoritmo deixa de fazer chamada recursivas??????
- As chamadas recursivas finalizam quando ocorre o caso base.
- Ou seja, ocorre quando $\frac{n}{2^i} = 1$.
- Portanto $T(\frac{n}{2^i}) = T(1) = O(1)$.

Método Iterativo - Busca Binária

Na $(i+1)$-ésima chamada recursiva:

$$T(n) = T(\frac{n}{2^i}) + ic.$$

- Quando o algoritmo deixa de fazer chamada recursivas??????
- As chamadas recursivas finalizam quando ocorre o caso base.
- Ou seja, ocorre quando $\frac{n}{2^i} = 1$.
- Portanto $T(\frac{n}{2^i}) = T(1) = O(1)$.

$$T(n) = O(1) + ic.$$

- Ufa! Removemos a recorrência.

Método Iterativo - Busca Binária

Na $(i+1)$-ésima chamada recursiva:

$$T(n) = T(\frac{n}{2^i}) + ic.$$

- Quando o algoritmo deixa de fazer chamada recursivas??????
- As chamadas recursivas finalizam quando ocorre o caso base.
- Ou seja, ocorre quando $\frac{n}{2^i} = 1$.
- Portanto $T(\frac{n}{2^i}) = T(1) = O(1)$.

$$T(n) = O(1) + ic.$$

- Ufa! Removemos a recorrência.
- Para qual o valor de i que isso ocorre?????

Método Iterativo - Busca Binária

UNIVERSIDADE
FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Na $(i+1)$-ésima chamada recursiva:

$$T(n) = T(\frac{n}{2^i}) + ic.$$

- Quando o algoritmo deixa de fazer chamada recursivas??????
- As chamadas recursivas finalizam quando ocorre o caso base.
- Ou seja, ocorre quando $\frac{n}{2^i} = 1$.
- Portanto $T(\frac{n}{2^i}) = T(1) = O(1)$.

$$T(n) = O(1) + ic.$$

- Ufa! Removemos a recorrência.
- Para qual o valor de i que isso ocorre?????

$$\frac{n}{2^i} = 1 \implies 2^i = n \implies \lg^{2^i} = \lg n \implies i = \lg n.$$

Método Iterativo - Busca Binária

$$T(n) = O(1) + ic = c + c \lg n = O(\lg n)$$

MergeSort

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

Dado um vetor v de tamanho n....

```
void mergeSort(int vet[], int ini, int fim) {
  if (ini < fim){
    int meio = (ini + fim)/2;
    mergeSort(vet, ini, meio);
    mergeSort(vet, meio + 1, fim);
    merge(vet, ini, meio, fim);
  }
}
```

- Qual o tempo de execução desse algoritmo?

MergeSort

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

```
void merge(int vet[], int ini, int meio, int fim) {
  int i = 0, j = meio + 1, k = 0;
  int aux[fim - ini + 1];
  while (i <= meio && j <= fim) {
    if(vet[i] < vet[j]){
      aux[k] = vet[i];
      i++;
    }else{
      aux[k] = vet[j];
      j++;
    }
    k++;
  }
  if(i == meio) {
    while(j <= fim) {
      aux[k] = vet[j];
      j++;
      k++;
    }
  }else{
    while(i <= meio) {
      aux[k] = vet[i];
      i++;
      k++;
    }
```

Método Iterativo - MergeSort

Universidade Federal do Ceará
Campus Quixadá

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

Método Iterativo - MergeSort

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

$$T(n) \quad = \quad 2T(\tfrac{n}{2}) + cn$$

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

Método Iterativo - MergeSort

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

$$\begin{aligned} T(n) &= 2T(\tfrac{n}{2}) + cn \\ &= 2(2T(\tfrac{n}{4}) + c\tfrac{n}{2}) + cn \end{aligned}$$

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

Método Iterativo - MergeSort

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

$$\begin{aligned} T(n) &= 2T(\tfrac{n}{2}) + cn && \text{Como } T(\tfrac{n}{2}) = 2T(\tfrac{n}{4}) + c\tfrac{n}{2} \\ &= 2(2T(\tfrac{n}{4}) + c\tfrac{n}{2}) + cn \\ &= 4T(\tfrac{n}{4}) + 2cn && \text{Como } T(\tfrac{n}{4}) = 2T(\tfrac{n}{8}) + c\tfrac{n}{4} \end{aligned}$$

Método Iterativo - MergeSort

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

$$\begin{aligned} T(n) &= 2T(\tfrac{n}{2}) + cn && \text{Como } T(\tfrac{n}{2}) = 2T(\tfrac{n}{4}) + c\tfrac{n}{2} \\ &= 2(2T(\tfrac{n}{4}) + c\tfrac{n}{2}) + cn && \\ &= 4T(\tfrac{n}{4}) + 2cn && \text{Como } T(\tfrac{n}{4}) = 2T(\tfrac{n}{8}) + c\tfrac{n}{4} \\ &= 4(2T(\tfrac{n}{8}) + c\tfrac{n}{4}) + 2cn && \end{aligned}$$

Método Iterativo - MergeSort

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

$$\begin{aligned} T(n) &= 2T(\tfrac{n}{2}) + cn & \text{Como } T(\tfrac{n}{2}) = 2T(\tfrac{n}{4}) + c\tfrac{n}{2} \\ &= 2(2T(\tfrac{n}{4}) + c\tfrac{n}{2}) + cn & \\ &= 4T(\tfrac{n}{4}) + 2cn & \text{Como } T(\tfrac{n}{4}) = 2T(\tfrac{n}{8}) + c\tfrac{n}{4} \\ &= 4(2T(\tfrac{n}{8}) + c\tfrac{n}{4}) + 2cn & \\ &= 8T(\tfrac{n}{8}) + 3cn & \text{Como } T(\tfrac{n}{8}) = 2T(\tfrac{n}{16}) + c\tfrac{n}{8} \end{aligned}$$

Método Iterativo - MergeSort

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

$$\begin{aligned} T(n) &= 2T(\tfrac{n}{2}) + cn \\ &= 2(2T(\tfrac{n}{4}) + c\tfrac{n}{2}) + cn \\ &= 4T(\tfrac{n}{4}) + 2cn \\ &= 4(2T(\tfrac{n}{8}) + c\tfrac{n}{4}) + 2cn \\ &= 8T(\tfrac{n}{8}) + 3cn \\ &= 8(2T(\tfrac{n}{16}) + c\tfrac{n}{8}) + 3cn \end{aligned}$$

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

Como $T(\frac{n}{4}) = 2T(\frac{n}{8}) + c\frac{n}{4}$

Como $T(\frac{n}{8}) = 2T(\frac{n}{16}) + c\frac{n}{8}$

Método Iterativo - MergeSort

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

$$\begin{aligned} T(n) &= 2T(\tfrac{n}{2}) + cn \\ &= 2(2T(\tfrac{n}{4}) + c\tfrac{n}{2}) + cn \\ &= 4T(\tfrac{n}{4}) + 2cn \\ &= 4(2T(\tfrac{n}{8}) + c\tfrac{n}{4}) + 2cn \\ &= 8T(\tfrac{n}{8}) + 3cn \\ &= 8(2T(\tfrac{n}{16}) + c\tfrac{n}{8}) + 3cn \\ &= 16T(\tfrac{n}{16}) + 4cn \end{aligned}$$

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

Como $T(\frac{n}{4}) = 2T(\frac{n}{8}) + c\frac{n}{4}$

Como $T(\frac{n}{8}) = 2T(\frac{n}{16}) + c\frac{n}{8}$

Método Iterativo - MergeSort

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

$$\begin{aligned}
T(n) &= 2T(\tfrac{n}{2}) + cn \\
&= 2(2T(\tfrac{n}{4}) + c\tfrac{n}{2}) + cn \\
&= 4T(\tfrac{n}{4}) + 2cn \\
&= 4(2T(\tfrac{n}{8}) + c\tfrac{n}{4}) + 2cn \\
&= 8T(\tfrac{n}{8}) + 3cn \\
&= 8(2T(\tfrac{n}{16}) + c\tfrac{n}{8}) + 3cn \\
&= 16T(\tfrac{n}{16}) + 4cn \\
\ldots &\quad \ldots
\end{aligned}$$

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

Como $T(\frac{n}{4}) = 2T(\frac{n}{8}) + c\frac{n}{4}$

Como $T(\frac{n}{8}) = 2T(\frac{n}{16}) + c\frac{n}{8}$

Método Iterativo - MergeSort

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

$$\begin{array}{lcll}
T(n) & = & 2T(\frac{n}{2}) + cn & \text{Como } T(\frac{n}{2}) = 2T(\frac{n}{4}) + c\frac{n}{2} \\
 & = & 2(2T(\frac{n}{4}) + c\frac{n}{2}) + cn & \\
 & = & 4T(\frac{n}{4}) + 2cn & \text{Como } T(\frac{n}{4}) = 2T(\frac{n}{8}) + c\frac{n}{4} \\
 & = & 4(2T(\frac{n}{8}) + c\frac{n}{4}) + 2cn & \\
 & = & 8T(\frac{n}{8}) + 3cn & \text{Como } T(\frac{n}{8}) = 2T(\frac{n}{16}) + c\frac{n}{8} \\
 & = & 8(2T(\frac{n}{16}) + c\frac{n}{8}) + 3cn & \\
 & = & 16T(\frac{n}{16}) + 4cn & \\
 & ... & ... & \\
 & = & 2^i T(\frac{n}{2^i}) + icn & \text{Na } (i+1)\text{-ésima chamada recursiva.}
\end{array}$$

Método Iterativo - MergeSort

- As chamadas recursivas finalizam quando ocorre o caso base, ou seja, quando $\frac{n}{2^i} = 1$.
- Portanto $T(\frac{n}{2^i}) = T(1) = O(1)$.
- Para qual o valor de i que isso ocorre?????

$$\frac{n}{2^i} = 1 \implies 2^i = n \implies \lg^{2^i} = \lg n \implies i = \lg n.$$

Substituindo:

$$T(n) = 2^i T(\frac{n}{2^i}) + icn = 2^i c + icn = c2^{\lg n} + cn\lg n = cn + cn\lg n = O(n\lg n)$$

Árvore de Recorrência

- Ideia similar do método iterativo, mas de maneira gráfica.
- Permite visualizar melhor o que acontece quando a recorrência é iterada.
- É mais fácil organizar as contas.

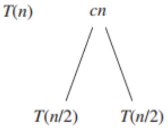
Árvore de Recorrência - MergeSort

Universidade Federal do Ceará
Campus Quixadá

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

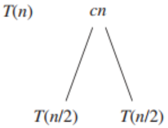
Árvore de Recorrência - MergeSort

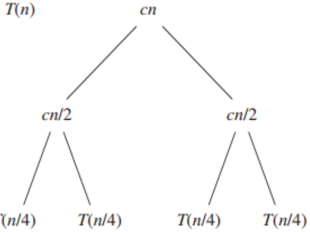
$$T(n) = \begin{cases} O(1) & , n = 1 \\ 2T(\frac{n}{2}) + O(n) & , n > 1 \end{cases}$$



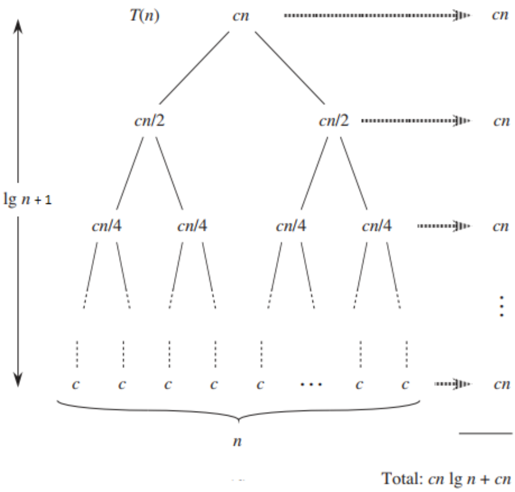
Árvore de Recorrência - MergeSort

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





Árvore de Recorrência - MergeSort



$$T(n) = cn + cn \lg n = O(n \lg n)$$

Árvore de Recorrência - Resumo

- O número de nós em cada nível da arvore é o número de chamadas recursivas.
- Em cada nó indicamos o tempo gasto naquele nó que não corresponde a chamadas recursivas.
- Na coluna mais a direita indicamos o tempo total naquele nível que não corresponde a chamadas recursivas.
- Somando ao longo da coluna determina-se a solução da recorrência.

Teorema Mestre

- Veremos agora um resultado que descreve soluções para recorrências da forma

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

onde $a \geq 1$ e $b > 1$ são constantes.

Teorema Mestre

- Veremos agora um resultado que descreve soluções para recorrências da forma

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

onde $a \geq 1$ e $b > 1$ são constantes.

- O caso base é omitido na definição e convenciona-se que é uma constante para valores pequenos.

Teorema Mestre

- Veremos agora um resultado que descreve soluções para recorrências da forma

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

onde $a \geq 1$ e $b > 1$ são constantes.

- O caso base é omitido na definição e convenciona-se que é uma constante para valores pequenos.

- A expressão $\frac{n}{b}$ pode indicar tanto $\lfloor \frac{n}{b} \rfloor$ quanto $\lceil \frac{n}{b} \rceil$.

Teorema Mestre

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Veremos agora um resultado que descreve soluções para recorrências da forma

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

onde $a \geq 1$ e $b > 1$ são constantes.

- O caso base é omitido na definição e convenciona-se que é uma constante para valores pequenos.
- A expressão $\frac{n}{b}$ pode indicar tanto $\lfloor \frac{n}{b} \rfloor$ quanto $\lceil \frac{n}{b} \rceil$.
- **O Teorema Mestre não fornece a resposta para todas as recorrências da forma acima.**

Teorema Mestre

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

- Veremos agora um resultado que descreve soluções para recorrências da forma

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

  onde $a \geq 1$ e $b > 1$ são constantes.

- O caso base é omitido na definição e convenciona-se que é uma constante para valores pequenos.
- A expressão $\frac{n}{b}$ pode indicar tanto $\lfloor \frac{n}{b} \rfloor$ quanto $\lceil \frac{n}{b} \rceil$.
- **O Teorema Mestre não fornece a resposta para todas as recorrências da forma acima.**
- Formado por três situações.

Teorema Mestre

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

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

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

Teorema Mestre

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

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

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

1. Se $f(n) = O(n^{\log_b^a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b^a})$.

Teorema Mestre

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

1. Se $f(n) = O(n^{\log_b^a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b^a})$.

Teorema Mestre

1. Se $f(n) = O(n^{\log_b^a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b^a})$.

Exemplo: $T(n) = 9T(\frac{n}{3}) + n$.

Teorema Mestre

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

1. Se $f(n) = O(n^{\log_b^a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b^a})$.

Exemplo: $T(n) = 9T(\frac{n}{3}) + n$.

- $a = 9$, $b = 3$ e $f(n) = n$, portanto, as constantes e a função satisfazem as condições do teorema mestre.

Teorema Mestre

1. Se $f(n) = O(n^{\log_b^a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b^a})$.

Exemplo: $T(n) = 9T(\frac{n}{3}) + n$.

- $a = 9$, $b = 3$ e $f(n) = n$, portanto, as constantes e a função satisfazem as condições do teorema mestre.
- $\log_3^9 = 2$. Logo, $n^{\log_3^9} = n^2$.

Teorema Mestre

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

1. Se $f(n) = O(n^{\log_b^a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b^a})$.

Exemplo: $T(n) = 9T(\frac{n}{3}) + n$.

- $a = 9$, $b = 3$ e $f(n) = n$, portanto, as constantes e a função satisfazem as condições do teorema mestre.
- $\log_3^9 = 2$. Logo, $n^{\log_3^9} = n^2$.
- Como $f(n) = O(n^{\log_3^9 - \epsilon})$, para $\epsilon = 1$, portanto, pelo item 1 do Teorema Mestre, $T(n) = \Theta(n^2)$.

Teorema Mestre

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

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

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

1. Se $f(n) = O(n^{\log_b^a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b^a})$.

2. Se $f(n) = \Theta(n^{\log_b^a})$, então $T(n) = \Theta(n^{\log_b^a} \log n)$.

Teorema Mestre

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

2. Se $f(n) = \Theta(n^{\log_b^a})$, então $T(n) = \Theta(n^{\log_b^a} \log n)$.

Teorema Mestre

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

2. Se $f(n) = \Theta(n^{\log_b^a})$, então $T(n) = \Theta(n^{\log_b^a} \log n)$.

Exemplo: $T(n) = T(\frac{2n}{3}) + 1$.

Teorema Mestre

2. Se $f(n) = \Theta(n^{\log_b^a})$, então $T(n) = \Theta(n^{\log_b^a} \log n)$.

Exemplo: $T(n) = T(\frac{2n}{3}) + 1$.

- $a = 1$, $b = \frac{3}{2}$ e $f(n) = 1$, portanto, as constantes e a função satisfazem as condições do teorema mestre.

Teorema Mestre

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

2. Se $f(n) = \Theta(n^{\log_b^a})$, então $T(n) = \Theta(n^{\log_b^a} \log n)$.

Exemplo: $T(n) = T(\frac{2n}{3}) + 1$.

- $a = 1$, $b = \frac{3}{2}$ e $f(n) = 1$, portanto, as constantes e a função satisfazem as condições do teorema mestre.
- $\log_{3/2}^1 = 0$. Logo, $n^{\log_{3/2}^1} = n^0 = 1$.

Teorema Mestre

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

2. Se $f(n) = \Theta(n^{\log_b^a})$, então $T(n) = \Theta(n^{\log_b^a} \log n)$.

Exemplo: $T(n) = T(\frac{2n}{3}) + 1$.

- $a = 1$, $b = \frac{3}{2}$ e $f(n) = 1$, portanto, as constantes e a função satisfazem as condições do teorema mestre.
- $\log_{3/2}^1 = 0$. Logo, $n^{\log_{3/2}^1} = n^0 = 1$.
- Como $f(n) = \Theta(n^{\log_b^a}) = \Theta(1)$, portanto, pelo item 2 do Teorema Mestre, $T(n) = \Theta(\log n)$.

Teorema Mestre

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

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

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

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

Teorema Mestre

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

Teorema Mestre

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

Exemplo: $T(n) = 3T(\frac{n}{4}) + n \lg n$.

Teorema Mestre

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

Exemplo: $T(n) = 3T(\frac{n}{4}) + n \lg n$.

- $a = 3$, $b = 4$ e $f(n) = n \lg n$, portanto, as constantes e a função satisfazem as condições do teorema mestre.

Teorema Mestre

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

Exemplo: $T(n) = 3T(\frac{n}{4}) + n \lg n$.

- $a = 3$, $b = 4$ e $f(n) = n \lg n$, portanto, as constantes e a função satisfazem as condições do teorema mestre.
- $\log_4^3 = 0.79248...$. Logo $n^{\log_4^3} = n^{0.79248...}$.

Teorema Mestre

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

Exemplo: $T(n) = 3T(\frac{n}{4}) + n \lg n$.

- $a = 3$, $b = 4$ e $f(n) = n \lg n$, portanto, as constantes e a função satisfazem as condições do teorema mestre.
- $\log_4^3 = 0.79248....$ Logo $n^{\log_4^3} = n^{0.79248...}$.
- Fazendo $\epsilon = 1 - \log_4^3 = 0.20751...$, temos que $f(n) = \Omega(n^{\log_4^3 + \epsilon})$.

Teorema Mestre

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

Exemplo: $T(n) = 3T(\frac{n}{4}) + n \lg n$.

- $a = 3$, $b = 4$ e $f(n) = n \lg n$, portanto, as constantes e a função satisfazem as condições do teorema mestre.
- $\log_4^3 = 0.79248....$ Logo $n^{\log_4^3} = n^{0.79248...}$.
- Fazendo $\epsilon = 1 - \log_4^3 = 0.20751...$, temos que $f(n) = \Omega(n^{\log_4^3 + \epsilon})$.
- Como $af(\frac{n}{b}) = 3(\frac{n}{4} \lg \frac{n}{4}) = \frac{3}{4}n(\lg n - \lg 4)) \leq \frac{3}{4}n \lg n \leq \frac{3}{4}f(n)$, Logo, $c = \frac{3}{4}$.

Teorema Mestre

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

Exemplo: $T(n) = 3T(\frac{n}{4}) + n \lg n$.

- $a = 3$, $b = 4$ e $f(n) = n \lg n$, portanto, as constantes e a função satisfazem as condições do teorema mestre.
- $\log_4^3 = 0.79248...$. Logo $n^{\log_4^3} = n^{0.79248...}$.
- Fazendo $\epsilon = 1 - \log_4^3 = 0.20751...$, temos que $f(n) = \Omega(n^{\log_4^3 + \epsilon})$.
- Como $af(\frac{n}{b}) = 3(\frac{n}{4} \lg \frac{n}{4}) = \frac{3}{4}n(\lg n - \lg 4)) \leq \frac{3}{4}n \lg n \leq \frac{3}{4}f(n)$, Logo, $c = \frac{3}{4}$.
- Portanto, pelo item 3 do Teorema Mestre, $T(n) = \Theta(n \lg n)$.

Teorema Mestre

UNIVERSIDADE FEDERAL DO CEARÁ
CAMPUS QUIXADÁ

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

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

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

1. Se $f(n) = O(n^{\log_b^a - \epsilon})$ para alguma constante $\epsilon > 0$, então $T(n) = \Theta(n^{\log_b^a})$.

2. Se $f(n) = \Theta(n^{\log_b^a})$, então $T(n) = \Theta(n^{\log_b^a} \log n)$.

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