Algoritmos e Estruturas de Dados III

Método mestre para análise de algoritmos recursivos

Prof. Dr. Carlos Alberto Ynoguti

Inatel
Instituto Nacional de Telecomunicações

Objetivos

Ao final desta atividade, os alunos deverão estar aptos a:

- Conhecer o método mestre para análise de algoritmos recursivos
- Aplicá-lo à análise de alguns casos

Introdução

- Muitos algoritmos importantes são implementados de forma recursiva. Exemplo: mergesort, quicksort, algoritmos para operação em árvores
- Existem vários métodos para fazer a análise de complexidade deste tipo de algoritmo, como a árvore de recursão e o método mestre
- Vamos focar agora no método mestre, que é uma “receita de bolo” para resolver este problema

O método mestre

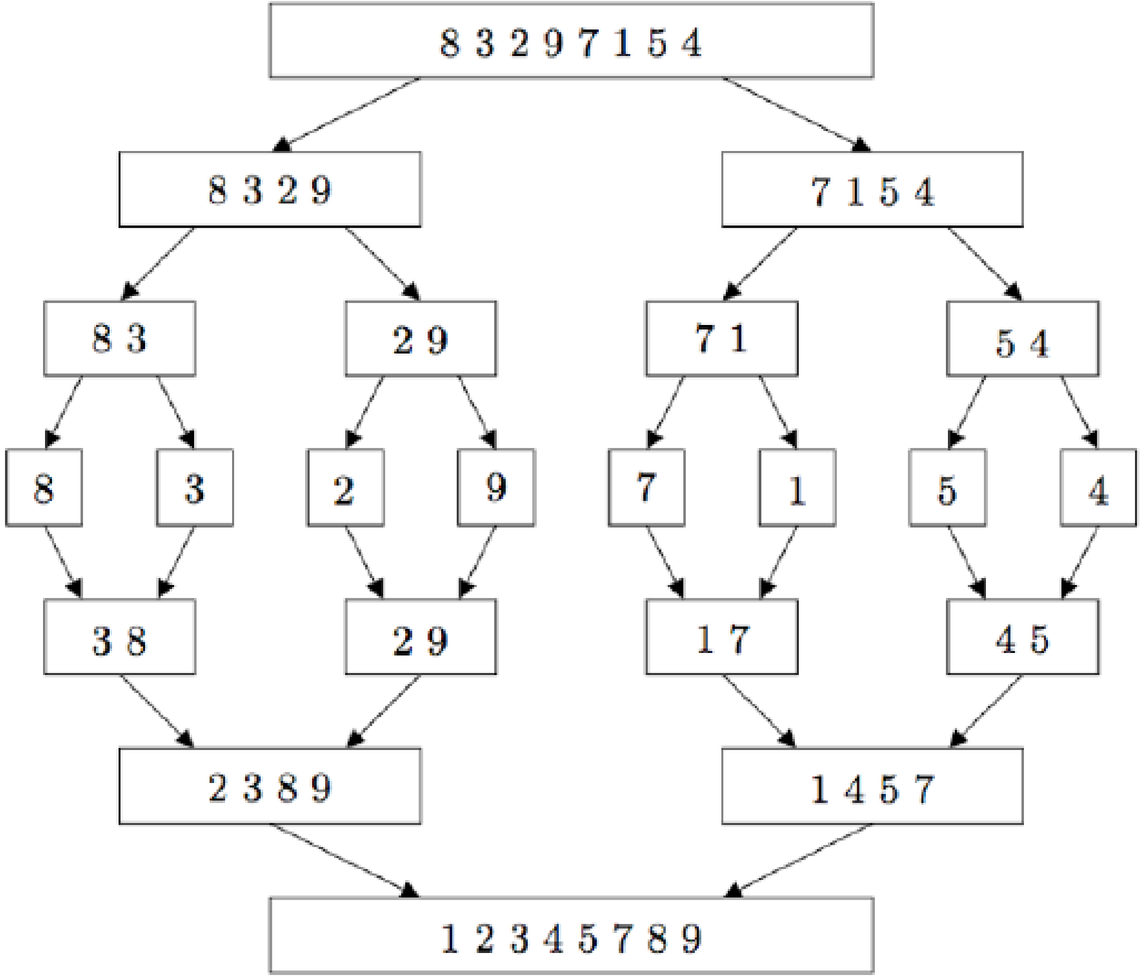
A fórmula padrão para o método mestre é:

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

onde:

- a: número de subproblemas que surgem da divisão
- n/b: tamanho de cada subproblema
- $f(n)=O(n^d)$: custo para criar os subproblemas e combinar seus resultados

Exemplo: mergesort



Exemplo: mergesort

```
merge(v,p,q,r)
    n1 = q-p+1, n2 = r-q
    para i = 0 até n1-1
        L[i] = v[p+i]
    para i=0 até n2-1
        R[i] = v[q+i+1]
    L[n1] = INT_MAX
    R[n2] = INT_MAX
    i = 0, j = 0
    para k = p até r
        se(L[i]<=R[j])
            v[k] = L[i]
            i++
        senão
            v[k] = R[j]
            j++
```

```
mergesort(v,p,r)
    se(p < r)
        q = (p + r)/2
        mergesort(v,p,q)
        mergesort(v,q+1,r)
        merge(v,p,q,r)
```

- a=2: dividimos o vetor inicial em 2 partes a cada chamada recursiva.
- n/b = n/2: a cada vez dividimos o vetor ao meio, de forma que o tamanho de cada subproblema é n/2, e portanto b=2.
- d = 1, pois o custo para fazer a divisão é O(1), e a rotina merge tem complexidade O(n), e desta forma, o custo para fazer a divisão e juntar os resultados é O(n).

Fórmulas para o algoritmo mestre

Caso 1:

$O(n^d \log(n))$ se $a = b^d$

Caso 2:

$O(n^d)$ se $a < b^d$

Caso 3:

$O(n^{\log_b a})$ se $a > b^d$

Para o mergesort

Caso 1:

$O(n^d \log(n))$ se $a = b^d$

Caso 2:

$O(n^d)$ se $a < b^d$

Caso 3:

$O(n^{\log_b a})$ se $a > b^d$

a = 2

b = 2

d = 1

Caso 1

$O(n^1 \log(n)) = O(\mathrm{n} \log(n))$

Exercício

Determine a complexidade dos algoritmos cujos tempos de execução são dados por:

1. $T(n) = 3T\left(\frac{n}{2}\right) + O(n^2)$
2. $T(n) = 16T\left(\frac{n}{4}\right) + O(n)$