Projeto e Análise de Algoritmos

Recorrência - Aula 5

Diane Castonguay
diane@inf.ufg.br

Instituto de Informática
Universidade Federal de Goiás

Dividir e Conquistar: Dividir

Dividir
o problema

em
um determinado número de

subproblemas

Dividir e Conquistar: Conquistar

Conquistar os subproblemas resolvendo-os recursivamente.

Resolver os subproblemas de tamanho sufficiente pequeno direitamente

Dividir e Conquistar: Combinar

Combinar
as soluções
dos subproblemas
a fim
de formar
a solução
para o problema original.

```
/*
 * Ordena um vetor usando a tecnica de dividir e conquistar
 * Chamada inicial: MERGE-SORT(A, 1, comprimento[A])
 */
MERGE-SORT(A, p, r)
1.      se p < r
2.          então q ← ⌊(p+r)/2⌋                    //Dividir
3.                MERGE-SORT(A, p, q)      //Conquistar
4.                MERGE-SORT(A, q+1, r)    //Conquistar
5.                MERGE(A, p, q, r)        //Combinar
6.          //senão o subvetor A[p..r] é vazio (p>r)
7.          //ou é de comprimento um (p=r).
8.          //Em ambos casos, ele é considerado ordenado.
9.      fim-se
```

```
/*
 * Combina os vetores ordenados A[p..q] e A[q+1..r]
 * em um vetor ordenado A[p..q]
 */
MERGE(A, p, q, r)
1. n1 ← q – p + 1     // comprimento de A[p..q]
2. n2 ← r – q         // comprimento de A[q+1..r]
3. criar os vetores L [1..n1+1] e R [1..n2+1]
4. para i ← 1 até n1 faça
5.         L[i] ← A[p + i – 1]
6. fim-para
7. para j ← 1 até n2 faça
8.         R[j] ← A[q + j]
9. fim-para
10. L[n1+1] ← ∞
11. R[n2+1] ← ∞
```

```
12. i ← 1
13. j ← 1
14. para k ← p até r faça
15.             se L[i] ≤ R[j]
16.                 então    A[k] ← L[i]
17.                          i ← i + 1
18.                 senão    A[k] ← R[j]
19.                          j ← j + 1
20.             fim-se
21. fim-para
```

Dividir e Conquistar: Complexidade

tamanho da entrada: n

Dividir: divisão em a subproblemas de tamanho n/b

custo: D(n)

Conquistar:

Uso da recursão

problemas pequenos ($n \leq c$) resolvidos direitamente

custo: constante, $\Theta(1)$

Combinar: custo : C(n)

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

MERGE-SORT: Complexidade

tamanho da entrada: n

Dividir:

divisão em 2 subvetores de tamanho n/2, linha 2

custo: $D(n) = \Theta(1)$

Conquistar:

Uso da recursão, linhas 3 e 4.

custo: ?

problemas pequenos ($n \leq 1$) resolvidos direitamente

custo: constante, $\Theta(1)$

Combinar: MERGE, custo : $C(n) = \Theta(n)$

MERGE-SORT: Complexidade

tamanho da entrada: n

Total: $T(n) = \begin{cases} \Theta(1) & \text{se } n \leq 1 \\ 2T(n/2) + \Theta(n) & \text{senão} \end{cases}$

Como resolver tal equação???
Como saber qual é a complexidade de MERGE-SORT???