Técnicas e Análise de Algoritmos: divisão e conquista

Prof. Jefferson T. Oliva

Algoritmos e Estrutura de Dados II (AE23CP)
Engenharia de Computação
Departamento Acadêmico de Informática (Dainf)
Universidade Tecnológica Federal do Paraná (UTFPR)
Campus Pato Branco

Sumário

- Divisão e Conquista
  - Exemplo 1: encontrar o maior valor
  - Exemplo 2: potenciação
  - Exemplo 3: problema da mochila

Divisão e Conquista

- Ideia básica
  - **Divisão**: dividir o problema a ser resolvido em subproblemas menores e independentes
  - **Conquista**: encontrar soluções para as partes recursivamente
  - **Combinação**: combinar as soluções obtidas em uma solução global

Divisão e Conquista

Divisao_e_Conquista(x)

```
if x é pequeno ou simples then
    return resolver(x)
else
    decompor x em conjuntos menores x_0, x_1, ... x_n
    for (i ← 0 até n) do
        y_i ← Divisao_e_Conquista(x_i)
    end for
    combinar y_i's
end if
return y
```

Divisão e Conquista

Exemplo 1: encontrar o maior valor

- O problema consiste em encontrar o maior elemento de um array $A[1..n]$

Solução Ingênua

$max \leftarrow A[1]$
for $i \leftarrow 2$ até n **do**
  if $A[i] > max$ **then**
    $max \leftarrow A[i]$
  end if
end for
return max

- Complexidade de tempo e de espaço: $\Theta(n)$

- Solução por divisão e conquista

Maxim(A, x, y)

if $y - x \leq 1$ **then**
    return $max(A[x], A[y])$
else
    $m \leftarrow (x + y)/2$
    $v1 \leftarrow Maxim(A[x..m])$
    $v2 \leftarrow Maxim(A[m + 1..y])$
end if
return $max(v1, v2)$

- A complexidade de tempo $T(n)$ do algoritmo é uma fórmula de recorrência
  - $T(n) = c$, para $n <= 2$, onde c é uma constante
  - $T(n) = 2T(n/2) + c$ para $n > 2$
- A complexidade de espaço $T(n)$ do algoritmo também é uma fórmula de recorrência
  - $T(n) = n + c$, para $n <= 2$ (já que o vetor possui tamanho n)
  - $T(n) = 2T(n/2) + c$ para $n > 2$
- Logo, por meio do método mestre (caso 1), para a função foi definida a complexidade de tempo e de espaço na ordem $\Theta(n)$

Divisão e Conquista

Exemplo 2: potenciação

- Solução Ingênua

potencia(a, n)

$p \leftarrow a$
for i $\leftarrow$ 2 até n **do**
    $p \leftarrow p \times a$
end for
return p

- Complexidade de tempo: $O(n)$

- Complexidade de espaço: $\Theta(1)$

Divisão e Conquista

Exemplo 2: potenciação

- Solução divisão e conquista

potencia(a, n)

if $n = 0$ **then**
    return 1
else if $n = 1$ **then**
    return a
else
    if n é par **then**
        $x \leftarrow potencia(a, n/2)$
        return $x * x$
    else
        $x \leftarrow potencia(a, (n-1)/2)$
        return $x * x * a$
    end if
end if

- A complexidade $T(n)$ do algoritmo é uma fórmula de recorrência, tanto para a análise de tempo quanto de espaço
  - $T(0) = T(1) = c$
  - $T(n) = T(n/2) + c$ para $n > 1$
- Logo, por meio do método mestre (caso 2), para a função foi definida a complexidade de $\Theta(\log n)$
- Pior caso: $O(\log n)$

- Se o vetor tiver o tamanho igual a 1, a função verifica se o peso do item não irá extrapolar a capacidade da mochila
    - Se a capacidade não for extrapolada, subtraia-a com o peso do item, já que o mesmo será adicionado na mochila e, em seguida, retorne o custo do item
    - Caso contrário, apenas retorne 0

- Caso o tamanho do vetor seja maior que 1
  1. **Divisão**: divida o vetor ao meio
  2. **Conquista 1**: tente incluir, na mochila, um objeto da primeira metade recursivamente
  3. **Conquista 2**: tente incluir, na mochila, um objeto da primeira metade recursivamente
  4. **Combinação**: some o resultado das duas metades

- Solução divisão e conquista

mochilaDQ(P, C, b, i, f)

```
if i = f and b − P[i] ≥ 0 then
    b ← b − P[i]
    return C[i]
else if i = f and b − P[i] < 0 then
    return 0
else
    m ← (i + f)/2
    return
    mochilaDQ(P, C, b, i, m) + mochilaDQ(P, C, b, m + 1, f)
end if
```

- A complexidade de tempo do algoritmo é uma fórmula de recorrência
  - $T(1) = c$
  - $T(n) = 2T(n/2) + c$ para $n > 1$
- A complexidade de espaço
  - $T(1) = n + c$
  - $T(n) = 2T(n/2) + c$ para $n > 1$
- Logo, por meio do método mestre (caso 1), para a função foi definida a complexidade de $\Theta(n)$
- **Aviso**: a solução do problema da mochila utilizando divisão e conquista apresentada em aula pode não gerar solução ótima!

Divisão e Conquista

- Outros exemplos de aplicação:
  - Ordenação por intercalação (*mergesort*)
  - Distância Euclidiana
  - Busca binária
  - Mediana
  - Quicksort

Divisão e Conquista

- Vantagens:
  - Altamente paralelizáveis
  - Fácil implementação
  - Simplificação de problemas complexos
- Desvantagens:
  - Necessidade de memória auxiliar
  - Repetição de subproblemas

Referências I

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; Clifford, S.
Algoritmos: teoria e prática.
Elsevier, 2012.

Horowitz, E., Sahni, S. Rajasekaran, S.
Computer Algorithms.
Computer Science Press, 1998.

Szwarcfiter, J.; Markenzon, L.
Estruturas de Dados e Seus Algoritmos.
LTC, 2010.

Ziviani, N.
Projeto de Algoritmos - com implementações em Java e C++.
Thomson, 2007.

Mergesort

- Se o vetor tiver o tamanho igual a 1, a função apenas retorna o elemento
- Caso o tamanho do vetor seja maior que 1
  1. **Divisão**: divida o vetor ao meio
  2. **Conquista 1**: ordene a primeira metade recursivamente
  3. **Conquista 2**: ordene a segunda metade recursivamente
  4. **Combinação**: intercale as duas metades

Mergesort

mergesort(A, p, r)

```
if p < r then
    q = (p + r)/2
    mergesort(A, p, q)
    mergesort(A, q + 1, r)
    merge(A, p, q, r)
end if
```

Mergesort

merge(A, p, q, r)

```
for i = p to q do
    B[i] = A[i]
end for
for j = q + 1 to r do
    C[j − q] = A[j] /*{Se for uma linguagem em que o primeiro elemento de
    vetores é acessado na posição 0, então C[j − q − 1] = A[j]}*/
end for
i = p
j = q
for k = p to r do
    if B[i] ≤ C[j] then
        A[k] = B[i]
        i = i + 1
    else
        A[k] = B[j]
        j = j − 1
    end if
end for
```

Mergesort

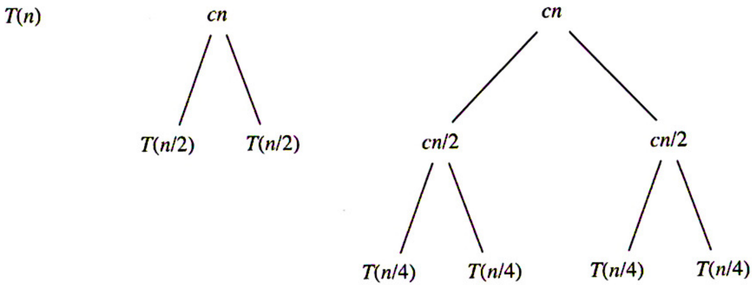
mergesort(A, p, r)

```
if p < r then
    q = (p + r)/2
    mergesort(A, p, q)
    mergesort(A, q + 1, r)
    merge(A, p, q, r)
end if
```

- A complexidade $T(n)$ do algoritmo é uma fórmula de recorrência
  - $T(1) = c$
  - $T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + cn$ para $n > 1$
- Logo, a complexidade do *mergesort* é $\Theta(n \log n)$
- Pior caso: $O(n \log n)$

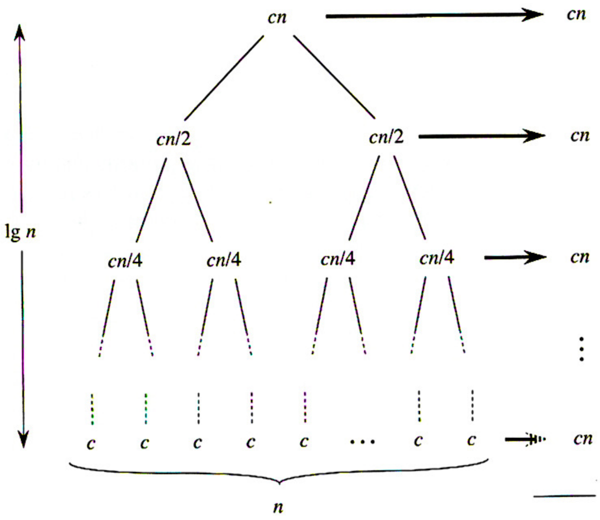
Mergesort

- Construção da árvore de recursão da recorrência do Mergesort $T(n) = 2T(n/2) + cn$



Mergesort

- Custo total: $cn \log n + cn = \Theta(n \log n)$



Funcionamento do Mergesort

- Entrada

	p				q				r
A	22	33	55	77	99	11	44	66	88

- Saída

	p				q				r
A	11	22	33	44	55	66	77	88	99

Funcionamento do Mergesort

	p				q				r
A	66	33	55	44	99	11	77	22	88

Funcionamento do Mergesort

	p				q				r
A	66	33	55	44	99	11	77	22	88

	p		q		r				
A	66	33	55	44	99				

Funcionamento do Mergesort

	p				q				r
A	66	33	55	44	99	11	77	22	88

	p		q		r				
A	66	33	55	44	99				

	p	q	r						
A	66	33	55						

Funcionamento do Mergesort

	p				q				r
A	66	33	55	44	99	11	77	22	88

	p		q		r				
A	66	33	55	44	99				

	p	q	r						
A	66	33	55						

	p	r							
A	66	33							

Funcionamento do Mergesort

	p				q				r
A	66	33	55	44	99	11	77	22	88

	p		q		r				
A	66	33	55	44	99				

	p	q	r						
A	66	33	55						

	p	r							
A	66	33							

	p = r								
A	66								

Funcionamento do Mergesort

	p				q				r
A	66	33	55	44	99	11	77	22	88

	p		q		r				
A	66	33	55	44	99				

	p	q	r						
A	66	33	55						

	p	r							
A	66	33							

	p = r								
A		33							

Funcionamento do Mergesort

	p				q				r
A	33	66	55	44	99	11	77	22	88

	p		q		r				
A	33	66	55	44	99				

	p	q	r						
A	33	66	55						

	p	r							
A	33	66							

Funcionamento do Mergesort

	p				q				r
A	33	66	55	44	99	11	77	22	88

	p		q		r				
A	33	66	55	44	99				

	p	q	r						
A	33	66	55						

			p = r						
A			55						

Funcionamento do Mergesort

	p				q				r
A	33	66	55	44	99	11	77	22	88

	p		q		r				
A	33	66	55	44	99				

	p	q	r						
A	33	66	55						

Funcionamento do Mergesort

	p				q				r
A	33	55	66	44	99	11	77	22	88

	p		q		r				
A	33	55	66	44	99				

	p	q	r						
A	33	55	66						

Funcionamento do Mergesort

	p				q				r
A	33	55	66	44	99	11	77	22	88

	p		q		r				
A	33	55	66	44	99				

				p	r				
A				44	99				

Funcionamento do Mergesort

	p				q				r
A	33	55	66	44	99	11	77	22	88

	p		q		r				
A	33	55	66	44	99				

				p	r				
A				44	99				

Funcionamento do Mergesort

	p				q				r
A	33	55	66	44	99	11	77	22	88

	p		q		r				
A	33	55	66	44	99				

				p	r				
A				44	99				

				p = r					
A				44					

Funcionamento do Mergesort

	p				q				r
A	33	55	66	44	99	11	77	22	88

	p		q		r				
A	33	55	66	44	99				

				p	r				
A				44	99				

					p = r				
A					99				

Funcionamento do Mergesort

	p				q				r
A	33	55	66	44	99	11	77	22	88

	p		q		r				
A	33	55	66	44	99				

				p	r				
A				44	99				

Funcionamento do Mergesort

	p				q				r
A	33	55	66	44	99	11	77	22	88

	p		q		r				
A	33	44	55	66	99				

Funcionamento do Mergesort

	p				q				r
A	33	44	55	66	99	11	77	22	88

Funcionamento do Mergesort

	p				q				r
A	33	44	55	66	99	11	77	22	88

						p	q		p
A						11	77	22	88

Funcionamento do Mergesort

	p				q				r
A	33	44	55	66	99	11	77	22	88

						p	q		p
A						11	22	77	88

. . .

Funcionamento do Mergesort

	p				q				r
A	11	22	33	44	55	66	77	88	99