Análise de Algoritmos

CLRS 4.1 e 4.2

Essas transparências foram adaptadas das transparências do Prof. Paulo Feofiloff e do Prof. José Coelho de Pina.

Mergesort

```
MERGESORT (A, p, d)
1   se p < d
2       então q ← ⌊(p + d)/2⌋
3             MERGESORT (A, p, q)
4             MERGESORT (A, q + 1, d)
5             INTERCALA (A, p, q, d)
```

linha	consumo máximo na linha
1	$\Theta(1)$
2	$\Theta(1)$
3	$T(\lceil n/2 \rceil)$
4	$T(\lfloor n/2 \rfloor)$
5	$\Theta(n)$

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n)$$

Mergesort

$T(n) :=$ consumo de tempo máximo quando $n = d - p + 1$

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) \quad (1)$$

Solução: $T(n)$ é $\Theta(???)$.

Receita:

- Substitua a notação assintótica por função da classe.
- Restrinja-se a n potência de 2.
- Estipule que na base o valor de é 1.
- Use expansão ou árvore de recorrência para determinar um “chute” de solução.
- Confira se o chute está correto.

Expansão

n potência de 2 e $k = \lg n$

Expansão

n potência de 2 e $k = \lg n$

$$T(n) = 2T(n/2) + n$$

Expansão

n potência de 2 e $k = \lg n$

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \\
&= 2\big(2T(n/2^2) + n/2\big) + n \;=\; 2^2 T(n/2^2) + 2n
\end{aligned}
$$

Expansão

n potência de 2 e $k = \lg n$

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \\
&= 2\big(2T(n/2^2) + n/2\big) + n \;=\; 2^2 T(n/2^2) + 2n \\
&= 2^2\big(2T(n/2^3) + n/2^2\big) + 2n \;=\; 2^3 T(n/2^3) + 3n
\end{aligned}
$$

Expansão

n potência de 2 e $k = \lg n$

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \\
&= 2\big(2T(n/2^2) + n/2\big) + n \;=\; 2^2 T(n/2^2) + 2n \\
&= 2^2\big(2T(n/2^3) + n/2^2\big) + 2n \;=\; 2^3 T(n/2^3) + 3n \\
&= \ldots \;=\; 2^k T(n/2^k) + kn
\end{aligned}
$$

Expansão

n potência de 2 e $k = \lg n$

$$
\begin{aligned}
T(n) &= 2T(n/2) + n \\
&= 2\big(2T(n/2^2) + n/2\big) + n \;=\; 2^2 T(n/2^2) + 2n \\
&= 2^2\big(2T(n/2^3) + n/2^2\big) + 2n \;=\; 2^3 T(n/2^3) + 3n \\
&= \ldots \;=\; 2^k T(n/2^k) + kn \\
&= n + n\lg n \;=\; \Theta(n \lg n)
\end{aligned}
$$

Conclusão: O MERGESORT consume $\Theta(n \lg n)$ unidades de tempo.

Exemplos

- $T(n) = T(\lfloor n/2 \rfloor) + \Theta(1)$
- $T(n) = T(n-1) + \Theta(n)$
- $T(n) = 3T(\lfloor n/2 \rfloor) + \Theta(n)$
- $T(n) = 2T(\lfloor n/2 \rfloor) + \Theta(n^2)$

Exemplos

- $T(n) = T(\lfloor n/2 \rfloor) + \Theta(1)$ $\qquad \Theta(\lg n)$
- $T(n) = T(n-1) + \Theta(n)$ $\qquad \Theta(n^2)$
- $T(n) = 3T(\lfloor n/2 \rfloor) + \Theta(n)$ $\qquad \Theta(n^{\lg 3})$
- $T(n) = 2T(\lfloor n/2 \rfloor) + \Theta(n^2)$ $\qquad \Theta(n^2)$

Segmento de soma máxima

Um segmento de um vetor $A[1\,..\,n]$ é qualquer subvetor da forma $A[e\,..\,d]$.

Problema: Dado um vetor $A[1\,..\,n]$ de números inteiros, determinar um segmento $A[e\,..\,d]$ de soma máxima.

Entra:

	1									n
A	31	−41	59	26	−53	58	97	−93	−23	84

Segmento de soma máxima

Um segmento de um vetor $A[1\,..\,n]$ é qualquer subvetor da forma $A[e\,..\,d]$.

Problema: Dado um vetor $A[1\,..\,n]$ de números inteiros, determinar um segmento $A[e\,..\,d]$ de soma máxima.

Entra:

	1									n
A	31	-41	59	26	-53	58	97	-93	-23	84

Sai:

	1		3				7			n
A	31	-41	59	26	-53	58	97	-93	-23	84

$A[e\,..\,d] = A[3\,..\,7]$ é segmento de soma máxima.
$A[3\,..\,7]$ tem soma 187.

Algoritmo café-com-leite

Determina um segmento de soma máxima de $A[1\,..\,n]$.

```
SEG-MAX-3 (A, n)
 1   somamax ← 0
 2   e ← 0    d ← −1    ▷ A[e..d] é vazio
 3   para i ← 1 até n faça
 4       para f ← i até n faça
 5           soma ← 0
 6           para k ← i até f faça
 7               soma ← soma + A[k]
 8           se soma > somamax então
 9               somamax ← soma    e ← i    d ← f
10   devolva e, d e somamax
```

Consumo de tempo

Se a execução de cada linha de código consome 1 unidade de tempo o consumo total é:

linha		todas as execuções da linha	
1-2	$=$	2	$= \Theta(1)$
3	$=$	$n+1$	$= \Theta(n)$
4	$=$	$(n+1)+n+(n-1)+\cdots+2$	$= \Theta(n^2)$
5	$=$	$n+(n-1)+\cdots+1$	$= \Theta(n^2)$
6	$=$	$(2+\cdots+(n+1))+(2+\cdots+n)+\cdots+2$	$= \Theta(n^3)$
7	$=$	$(1+\cdots+n)+(1+\cdots+(n-1))+\cdots+1$	$= \Theta(n^3)$
8	$=$	$n+(n-1)+(n-2)+\cdots+1$	$= \Theta(n^2)$
9	$\leq$	$n+(n-1)+(n-2)+\cdots+1$	$= \mathrm{O}(n^2)$
10	$=$	1	$= \Theta(1)$
total	$=$	$\Theta(2n^3+3n^2+n+2)+\mathrm{O}(n^2)$	$= \Theta(n^3)$

Algoritmo arroz-com-feijão

Determina um segmento de soma máxima de $A[1\,..\,n]$.

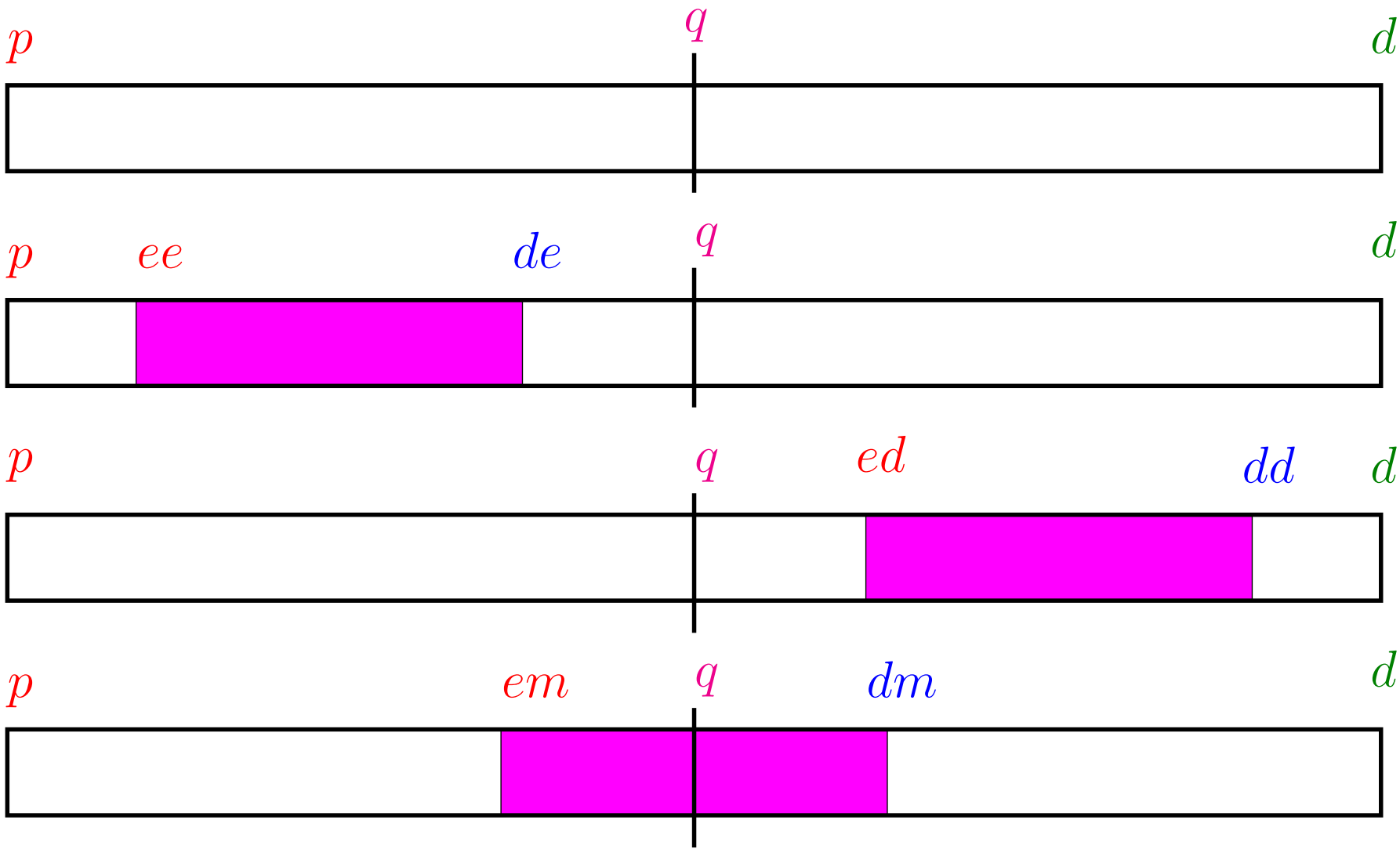
```
SEG-MAX-2 (A, n)
1   somamax ← 0
2   e ← 0   d ← −1   ▷ A[e..d] é vazio
3   para i ← 1 até n faça
4       soma ← 0
5       para f ← i até n faça
6           soma ← soma + A[f]
7           se soma > somamax então
8               somamax ← soma   e ← i   d ← f
9   devolva e, d e somamax
```

Consumo de tempo

Se a execução de cada linha de código consome 1 unidade de tempo o consumo total é:

linha		todas as execuções da linha	
1-2	$=$	2	$= \Theta(1)$
3	$=$	$n+1$	$= \Theta(n)$
4	$=$	n	$= \Theta(n)$
5	$=$	$(n+1)+n+\cdots+2$	$= \Theta(n^2)$
6	$=$	$n+(n-1)+\cdots+1$	$= \Theta(n^2)$
7	$=$	$n+(n-1)+\cdots+1$	$= \Theta(n^2)$
8	$\leq$	$n+(n-1)+\cdots+1$	$= \mathrm{O}(n^2)$
9	$=$	1	$= \Theta(1)$
total	$=$	$\Theta(3n^2+2n+2)+\mathrm{O}(n^2)$	$= \Theta(n^2)$

Solução de divisão-e-conquista



Algoritmo de divisão-e-conquista

Setermina soma máxima de um seg. de $A[p\,..\,d]$.

```
SEG-MAX-DC (A, p, d)
 1   se p = d então devolva max(0, A[p])
 2   q ← ⌊(p + d)/2⌋
 3   maxesq ← SEG-MAX-DC(A, p, q)
 4   maxdir ← SEG-MAX-DC(A, q + 1, d)
 5   max2esq ← soma ← A[q]
 6   para i ← q − 1 decrescendo até p faça
 7         soma ← soma + A[i]
 8         max2esq ← max(max2esq, soma)
 9   max2dir ← soma ← A[q + 1]
10   para f ← q + 2 até d faça
11         soma ← soma + A[f]
12         max2dir ← max(max2dir, soma)
13   maxcruz ← max2esq + max2dir
14   devolva max(maxesq, maxcruz, maxdir)
```

Algoritmo de divisão-e-conquista

Convença-se que o algoritmo anterior funciona. Ou seja, que ele de fato devolve a resposta correta.

Analise o consumo de tempo do algoritmo.