Melhores momentos

AULA PASSADA

Recorrências

$$\begin{aligned} T(1) &= 1 \\ T(n) &= T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 6n + 5 \quad \text{para } n = 2, 3, 4, 5, \ldots \end{aligned}$$

n	1	2	3	4	5	6	7	8	9	10
$n\lceil \lg n \rceil$	0	2	6	8	15	18	21	24	36	40
$T(n)$	1	19	43	67	97	127	157	187	223	259
$20\,n\lfloor \lg n \rfloor$	0	40	60	160	200	240	280	480	540	600

Conclusão:

$$\boxed{T(n) \text{ é } \Theta(n \lg n).}$$

Conclusão da conclusão

> O consumo de tempo do MERGE-SORT é $\Theta(n \lg n)$
> no pior caso.

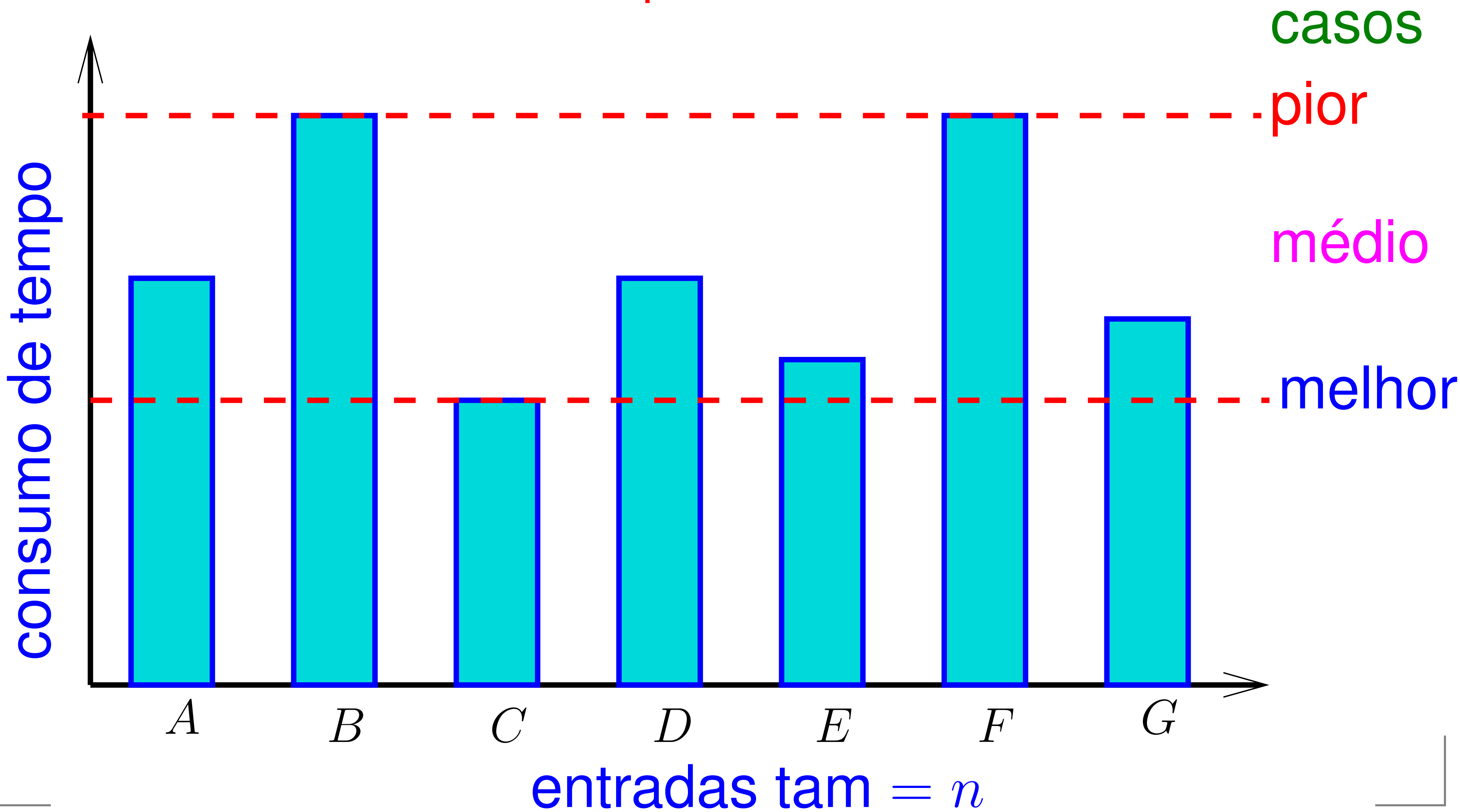
Exercício. Mostre que:

> O consumo de tempo do MERGE-SORT é $\Theta(n \lg n)$.

Hmmmm... Qual a diferença entra as duas afirmações?

Lembra?

Número mínimo, médio ou máximo?
Melhor caso, caso médio, pior caso?



AULA 5

Recorrências com notação O

CLRS 4.1–4.2

AU 3.9, 3.11

Classe O da solução de uma recorrência

Não faço questão de solução exata: basta solução assintótica (em notação O, se possível Θ)

Classe O da solução de uma recorrência

Não faço questão de solução exata: basta solução assintótica (em notação O, se possível Θ)

Exemplo:

$$
\begin{aligned}
G(1) &= 1 \\
G(n) &= 2G(n/2) + 7n + 2 \quad \text{para } n = 2, 4, 8, 16, \ldots
\end{aligned}
$$

Solução exata: $G(n) = 7n \lg n + 3n - 2$

Solução assintótica: $G(n) = \mathrm{O}(n \lg n)$ $\quad$ ($G(n) = \Theta(n \lg n)$)

Classe O da solução de uma recorrência

Não faço questão de solução exata: basta solução assintótica (em notação O, se possível Θ)

Exemplo:

$$
\begin{aligned}
G(1) &= 1 \\
G(n) &= 2G(n/2) + 7n + 2 \quad \text{para } n = 2, 4, 8, 16, \ldots
\end{aligned}
$$

Solução exata: $G(n) = 7n \lg n + 3n - 2$

Solução assintótica: $G(n) = \mathrm{O}(n \lg n)$ $\quad$ ($G(n) = \Theta(n \lg n)$)

Em geral, é mais fácil obter e provar solução assintótica que solução exata

Dica prática (sem prova)

A solução da recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(\lfloor n/2 \rfloor) + 7n + 2 \quad \text{para } n = 2, 3, 4, 5, \ldots \end{aligned}$$

está na mesma classe Θ que a solução de

$$\begin{aligned} T'(1) &= 1 \\ T'(n) &= 2T'(n/2) + n \quad \text{para } n = 2, 2^2, 2^3, \ldots \end{aligned}$$

Dica prática (sem prova)

A solução da recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(\lfloor n/2 \rfloor) + 7n + 2 \quad \text{para } n = 2, 3, 4, 5, \ldots \end{aligned}$$

está na mesma classe Θ que a solução de

$$\begin{aligned} T'(1) &= 1 \\ T'(n) &= 2T'(n/2) + n \quad \text{para } n = 2, 2^2, 2^3, \ldots \end{aligned}$$

e na mesma classe Θ que a solução de

$$\begin{aligned} T''(4) &= 10 \\ T''(n) &= 2T''(n/2) + n \quad \text{para } n = 2^3, 2^4, 2^5, \ldots \end{aligned}$$

Recorrências com O do lado direito

A “recorrência”

$$T(n) = 2T(n/2) + \mathrm{O}(n)$$

representa todas as recorrências da forma $T(n) = 2T(n/2) + f(n)$ em que $f(n)$ é $\mathrm{O}(n)$.

Recorrências com O do lado direito

A "recorrência"

$$T(n) = 2T(n/2) + \mathrm{O}(n)$$

representa todas as recorrências da forma
$T(n) = 2T(n/2) + f(n)$ em que $f(n)$ é $\mathrm{O}(n)$.

Melhor: representa todas as recorrências do tipo

$$\begin{aligned} T'(n) &\le a \quad \text{para } n = k, k+1, \ldots, 2k-1 \\ T'(n) &\le 2T'(\lfloor n/2 \rfloor) + b\,n \quad \text{para } n \ge 2k \end{aligned}$$

quaisquer que sejam a, $b > 0$ e $k > 0$
(poderíamos tomar $n_0 = 1$; veja ex ??.I)

Recorrências com O do lado direito

A "recorrência"

$$T(n) = 2T(n/2) + \mathrm{O}(n)$$

também representa todas as recorrências do tipo

$$\begin{aligned} T''(n) &= a \quad \text{para } n = 2^{k-1} \\ T''(n) &\leq 2T''(n/2) + b\,n \quad \text{para } n = 2^k, 2^{k+1}, \ldots \end{aligned}$$

quaisquer que sejam a, $b > 0$ e $k > 0$

Recorrências com O do lado direito

A “recorrência”

$$T(n) = 2T(n/2) + \mathrm{O}(n)$$

também representa todas as recorrências do tipo

$$\begin{array}{lcll} T''(n) & = & a & \text{para } n = 2^{k-1} \\ T''(n) & \leq & 2T''(n/2) + b\,n & \text{para } n = 2^{k}, 2^{k+1}, \ldots \end{array}$$

quaisquer que sejam a, $b > 0$ e $k > 0$

As soluções exatas vão depender de a, b, k;
mas todas estarão na mesma classe O (no caso, em $\mathrm{O}(n \lg n)$)

Exemplo com O

Recorrência com O do lado direito

Na análise do consumo de tempo do MERGE-SORT tinhamos:

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

$$T(1) = \mathrm{O}(1)$$

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \mathrm{O}(n) \quad \text{para } n = 2, 3, 4, \ldots$$

Solução assintótica: $T(n)$ é $\mathrm{O}(n \lg n)$.

Exemplo com Θ

Recorrências com Θ do lado direito

Na análise do consumo de tempo do MERGE-SORT poderíamos ter feito:

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

$$T(1) = \Theta(1)$$

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) \quad \text{para } n = 2, 3, 4, \ldots$$

Solução assintótica: $T(n)$ é $\Theta(n \lg n)$.

Dica prática

Suponha dada uma “recorrência” como

$$T(n) = 2T(n/2) + \mathrm{O}(n).$$

Para ter uma idéia da classe O a que T pertence, resolva a recorrência

$$\begin{aligned} T''(2^0) &= 1 \\ T''(n) &\leq 2T''(n/2) + n \quad \text{para } n = 2^1, 2^2, 2^3, \ldots \end{aligned}$$

Mais dicas práticas

recorrência	condição	solução
$T(n) = T(n-1) + 4n^3$		$\Theta(n^{3+1})$
$T(n) = 6T(n-1) + 4n^3$		$\Theta(6^n)$
$T(n) = aT(n/5) + 4n^3$	$a < 5^3$	$\Theta(n^3)$
$T(n) = aT(n/5) + 4n^3$	$a = 5^3$	$\Theta(n^3 \log n)$
$T(n) = aT(n/5) + 4n^3$	$a > 5^3$	$\Theta(n^{\log_5 a})$

Veja AU, sec 3.11, p.151

Mais dicas práticas (continuação)

No lugar de

- "$n/5$", posso escrever "$\lfloor n/5 \rfloor$" ou "$\lceil n/5 \rceil$"
- "5", posso escrever qualquer $b > 1$
- "4", posso escrever qualquer número real
- "$4n^3$" posso escrever qualquer polinômio de grau 3
- "3", posso escrever qualquer inteiro $k \geq 0$
- "6", posso escrever qualquer número $a > 1$

Mais dicas práticas ainda

A mesma coisa, escrita de maneira um pouco diferente (Master Theorem, CLRS, sec. 4.3, p.73):

Suponha $T(n) = aT(n/5) + f(n)$ para algum $a \geq 1$. Então, em geral,

$$
\begin{array}{lll}
\text{se } f(n) = \mathrm{O}(n^{\log_5 a - \epsilon}) & \text{então} & T(n) = \Theta(n^{\log_5 a}) \\
\text{se } f(n) = \Theta(n^{\log_5 a}) & \text{então} & T(n) = \Theta(n^{\log_5 a} \lg n) \\
\text{se } f(n) = \Omega(n^{\log_5 a + \epsilon}) & \text{então} & T(n) = \Theta(f(n))
\end{array}
$$

para qualquer $\epsilon > 0$

Teorema Ban-Ban-Ban

Teorema Mestre (Master Theorem, CLRS, sec. 4.3, p.73):

Suponha

$$T(n) = a\,T(n/b) + f(n)$$

para algum $a \geq 1$ e $b > 1$ e onde n/b significa $\lceil n/b \rceil$ ou $\lfloor n/b \rfloor$.
Então, em geral,

se $f(n) = \mathrm{O}(n^{\log_b a - \epsilon})$ então $T(n) = \Theta(n^{\log_b a})$
se $f(n) = \Theta(n^{\log_b a})$ então $T(n) = \Theta(n^{\log_b a} \lg n)$
se $f(n) = \Omega(n^{\log_b a + \epsilon})$ então $T(n) = \Theta(f(n))$

para qualquer $\epsilon > 0$.

Exercícios

Exercício 9.A
A que classe O pertencem as solução de recorrência do tipo $T(n) = T(n/3) + \mathrm{O}(1)$?

Exercício 9.B
Seja T a função definida pela recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 4T(\lfloor n/2 \rfloor) + n \qquad \text{para } n = 2, 3, 4, 5, \ldots \end{aligned}$$

A que ordem Θ pertence T?

Exercício 9.C [CLRS 4.2-1]
Seja T a função definida pela recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 3T(\lfloor n/2 \rfloor) + n \qquad \text{para } n = 2, 3, 4, 5, \ldots \end{aligned}$$

A partir da árvore da recorrência, adivinhe a que classe Θ pertence $T(n)$. Prove a delimitação pelo método da substituição.

Mais exercícios

Exercício 9.D
Resolva a recorrência

$$\begin{aligned} T(1) &= 1 \\ T(n) &= 2T(\lceil n/2 \rceil) + 7n + 2 \quad \text{para } n = 2, 3, 4, 5, \ldots \end{aligned}$$

Exercício 9.E
Resolva a "recorrência" $T(n) = T(n-2) + \mathrm{O}(n)$.

Exercício 9.F
Resolva a "recorrência" $T(n) = 5T(n-1) + \mathrm{O}(n)$.

Técnicas em projeto de algoritmos

Programming Pearls: Algorithm Design Techniques,

Jon Bentley, Addison-Wesley, 1986

Segmento de soma máxima

Um segmento de um vetor $A[1\,..\,n]$ é um 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:

$$
\begin{array}{c|c|c|c|c|c|c|c|c|c|c|}
 & 1 & & & & & & & & & n \\
\cline{2-11}
A & 31 & -41 & 59 & 26 & -53 & 58 & 97 & -93 & -23 & 84 \\
\cline{2-11}
\end{array}
$$

Segmento de soma máxima

Um segmento de um vetor $A[1\,..\,n]$ é um 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.

Segmento de soma máxima

Problema (versão simplificada): Determinar a soma máxima de um segmento de um dado um vetor $A[1\,..\,n]$.

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 soma máxima é 187.

Algoritmo café-com-leite

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

SEG-MAX-3 (A, n)

$somamax \leftarrow 0$
$e \leftarrow 0 \quad d \leftarrow -1 \quad \rhd\ A[e\,..\,d]$ é vazio
para $i \leftarrow 1$ **até** n **faça**
    para $f \leftarrow i$ **até** n **faça**
        $soma \leftarrow 0$
        para $k \leftarrow i$ **até** f **faça**
            $soma \leftarrow soma + A[k]$
        se $soma > somamax$ **então**
            $somamax \leftarrow soma \quad e \leftarrow i \quad d \leftarrow f$
devolva e, d **e** $somamax$

Correção

Relação invariante chave:

(i0) na linha 3 vale que: $A[e \mathinner{.\,.} d]$ é um segmento de soma máxima com $e < i$.

	e		i				d			n
A	31	−41	59	26	−53	58	97	−93	−23	84

Correção

Relação invariante chave:

(i0) na linha 3 vale que: $A[e \,.\,.\, d]$ é um segmento de soma máxima com $e < i$.

	e		i				d			n
A	31	−41	59	26	−53	58	97	−93	−23	84

Mais relações invariantes:

(i1) na linha 3 vale que:

$$somamax = A[e] + A[e+1] + A[e+2] + \cdots + A[d];$$

(i2) na linha 6 vale que:

$$soma = A[i] + A[i+1] + A[i+2] + \cdots + A[k-1].$$

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	$=$	$(n+1)+n+(n-1)+\cdots+1$	$= \Theta(n^2)$
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

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

SEG-MAX-2 (A, n)

$\quad somamax \leftarrow 0$

$\quad e \leftarrow 0 \quad d \leftarrow -1 \quad \rhd\ A[e\,.\,.\,d]$ é vazio

$\quad$ **para** $i \leftarrow 1$ **até** n **faça**

$\qquad soma \leftarrow 0$

$\qquad$ **para** $f \leftarrow i$ **até** n **faça**

$\qquad\quad soma \leftarrow soma + A[f]$

$\qquad\quad$ **se** $soma > somamax$ **então**

$\qquad\qquad somamax \leftarrow soma \quad e \leftarrow i \quad d \leftarrow f$

$\quad$ **devolva** e, d **e** $somamax$

Correção

Relação invariante chave:

(i0) na linha 3 vale que: $A[e\,..\,d]$ é um segmento de soma máxima com $e < i$.

	e						d	i		n
A	31	-41	59	26	-53	58	97	-93	-23	84

Mais relações invariante:

(i1) na linha 3 vale que:

$$somamax = A[e] + A[e+1] + A[e+2] + \cdots + A[d];$$

(i2) na linha 5 vale que:

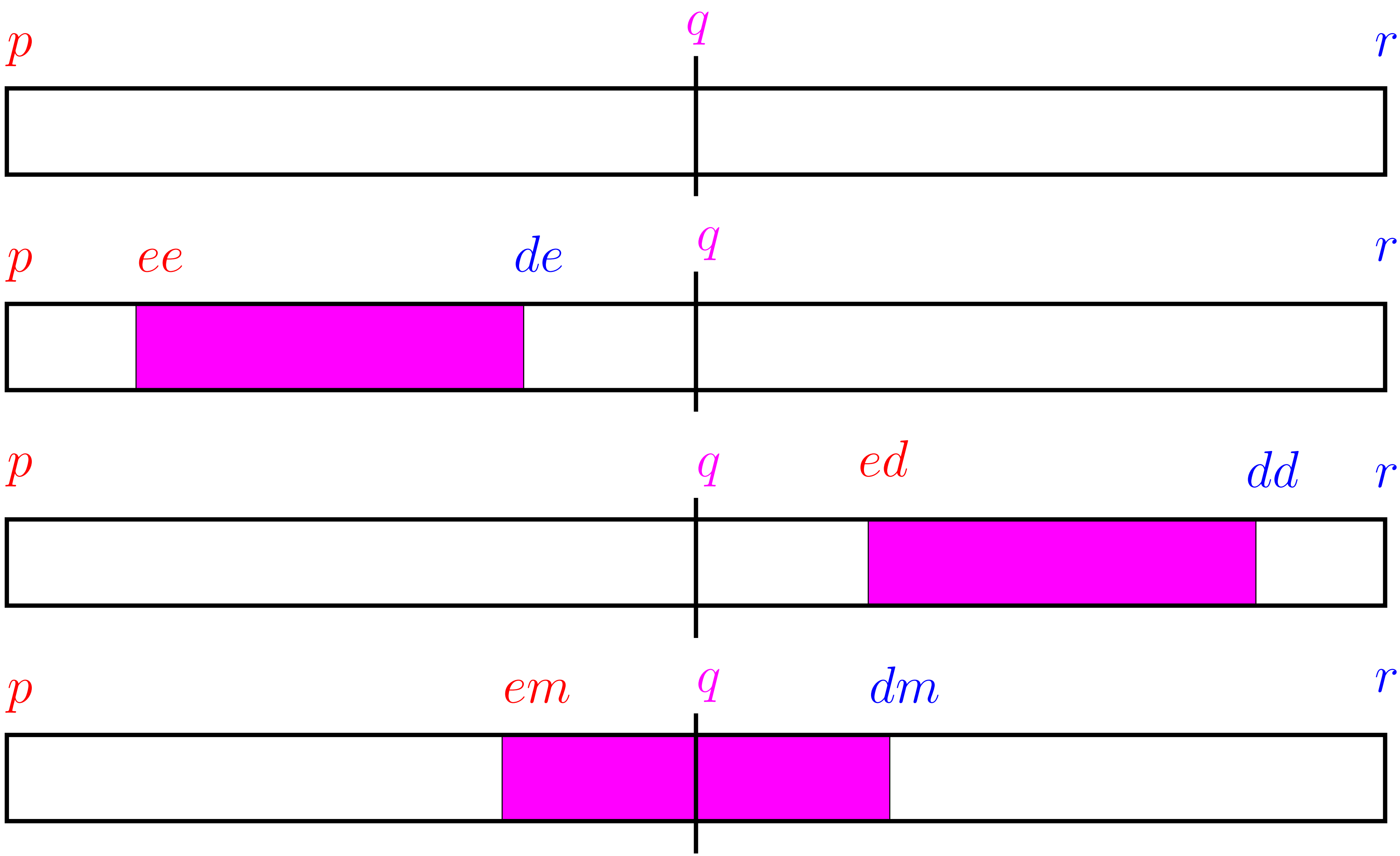
$$soma = A[i] + A[i+1] + A[i+2] + \cdots + A[f-1];$$

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

Algoritmo determina soma máxima de um seg. de $A[p \,.\,.\, r]$.

SEG-MAX-DC (A, p, r)

se $p = r$ **então devolva** $\max(0, A[p])$
$q \leftarrow \lfloor (p + r)/2 \rfloor$
$maxesq \leftarrow$ SEG-MAX-DC(A, p, q)
$maxdir \leftarrow$ SEG-MAX-DC$(A, q + 1, r)$
$max2esq \leftarrow soma \leftarrow A[q]$
para $i \leftarrow q - 1$ **decrescendo até** p **faça**
    $soma \leftarrow soma + A[i]$
    $max2esq \leftarrow \max(max2esq, soma)$
$max2dir \leftarrow soma \leftarrow A[q + 1]$
para $f \leftarrow q + 2$ **até** r **faça**
    $soma \leftarrow soma + A[f]$
    $max2dir \leftarrow \max(max2dir, soma)$
$maxcruz \leftarrow max2esq + max2dir$
devolva $\max(maxesq, maxcruz, maxdir)$

Correção

Verifique que:

- $maxesq$ é a soma máxima de um segmento de $A[p \mathinner{..} q]$;
- $maxdir$ é a soma máxima de um segmento de $A[q+1 \mathinner{..} r]$; e
- $maxcruz$ é a soma máximo de um segmento da forma $A[i \mathinner{..} f]$ com $i \leq q$ e $q+1 \leq f$.

Conclua que o algoritmo devolve a soma máxima de um segmento de $A[p \mathinner{..} r]$.

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	$=$	$T(\lceil \frac{n}{2} \rceil)$	$= T(\lceil \frac{n}{2} \rceil)$
4	$=$	$T(\lfloor \frac{n}{2} \rfloor)$	$= T(\lfloor \frac{n}{2} \rfloor)$
5	$=$	1	$= \Theta(1)$
6	$=$	$\lceil \frac{n}{2} \rceil + 1$	$= \Theta(n)$
7-8	$=$	$\lceil \frac{n}{2} \rceil$	$= \Theta(n)$
9	$=$	1	$= \Theta(1)$
10	$=$	$\lfloor \frac{n}{2} \rfloor + 1$	$= \Theta(n)$
11-12	$=$	$\lfloor \frac{n}{2} \rfloor$	$= \Theta(n)$
13-14	$=$	2	$= \Theta(1)$
total	$=$	$T(\lceil \frac{n}{2} \rceil) + T(\lfloor \frac{n}{2} \rfloor) + \Theta(4n+4)$	

Consumo de tempo

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

Na análise do consumo de tempo do SEG-MAX-DC chegamos a (já manjada) recorrências com Θ do lado direito:

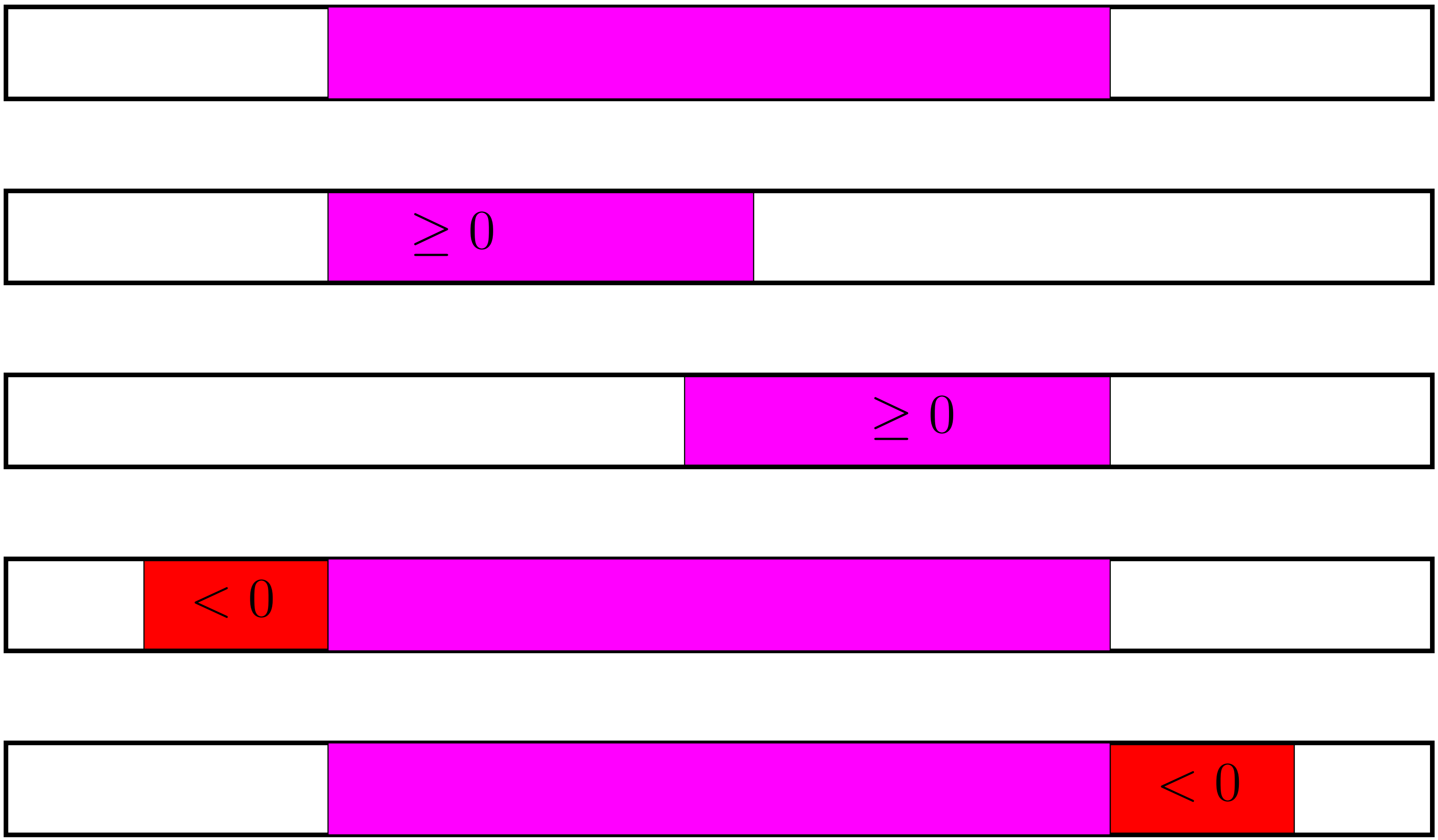
$$T(1) = \Theta(1)$$

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) \quad \text{para } n = 2, 3, 4, \ldots$$

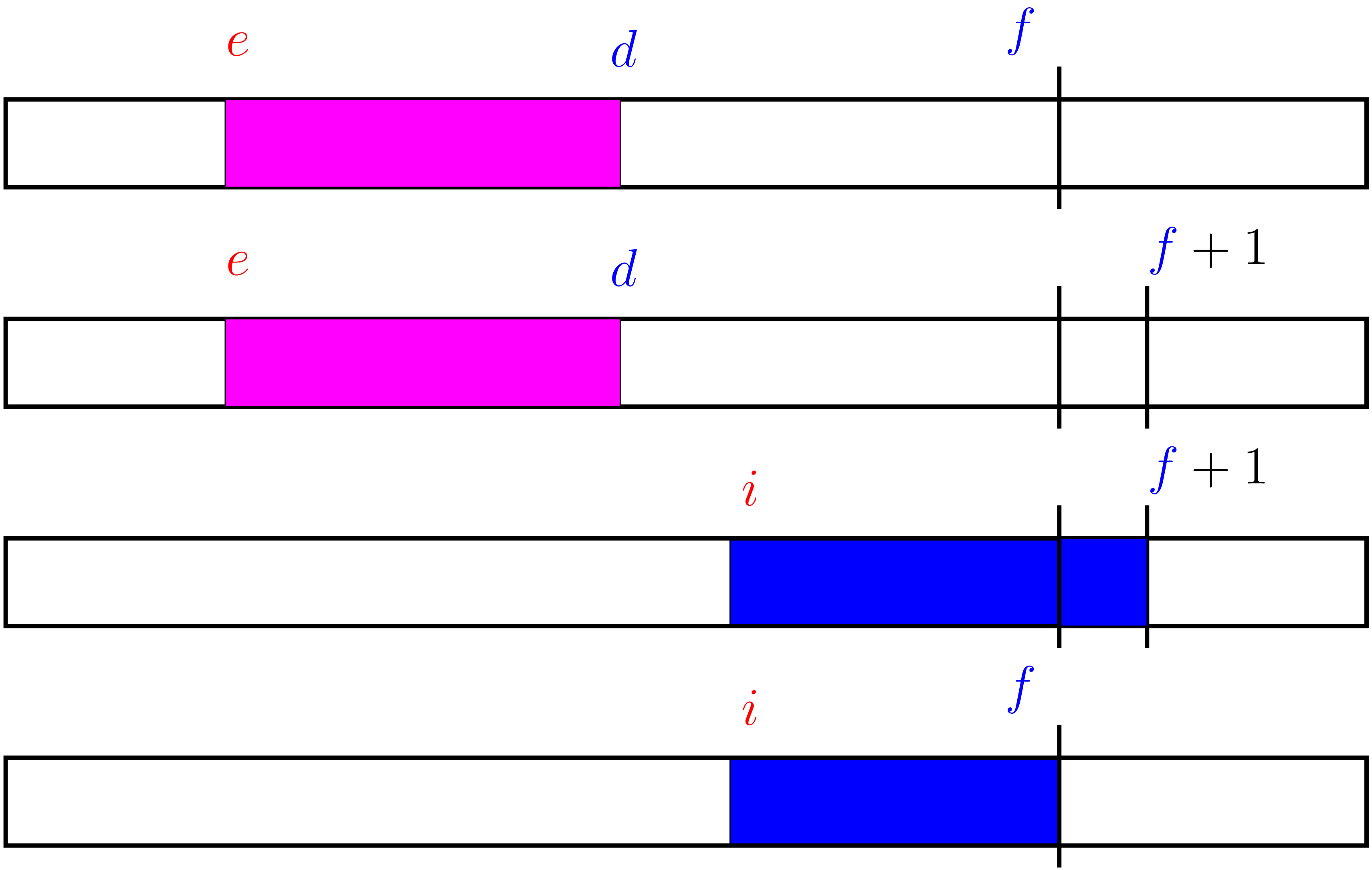
Solução aproximada/assintótica: $T(n)$ é $\Theta(n \lg n)$.

Cara da solução

Solução



Solução indutiva



Algoritmo linear

Algoritmo determina um segmento de soma máxima de $A[1\,..\,n]$ (por Jay Kadane).

SEG-MAX-1 (A, n)

$somamax \leftarrow 0$
$e \leftarrow 0 \quad d \leftarrow -1 \quad \rhd\ A[e\,..\,d]$ é vazio
$soma \leftarrow 0$
$i \leftarrow 1 \quad f \leftarrow -1 \quad \rhd\ A[e\,..\,f]$ é vazio
enquanto $f < n$ **faça**
    $f \leftarrow f + 1$
    $soma \leftarrow soma + A[f]$
    se $soma < 0$
        então $soma \leftarrow 0 \quad i \leftarrow f + 1$
    senão se $soma > somamax$
        então $somamax \leftarrow soma \quad e \leftarrow i \quad d \leftarrow f$
devolva e, d **e** $somamax$

Correção

Relação invariante chave:

(i0) na linha 5 vale que: $A[e \mathinner{.\,.} d]$ é segmento de soma máxima com $d \le f$.

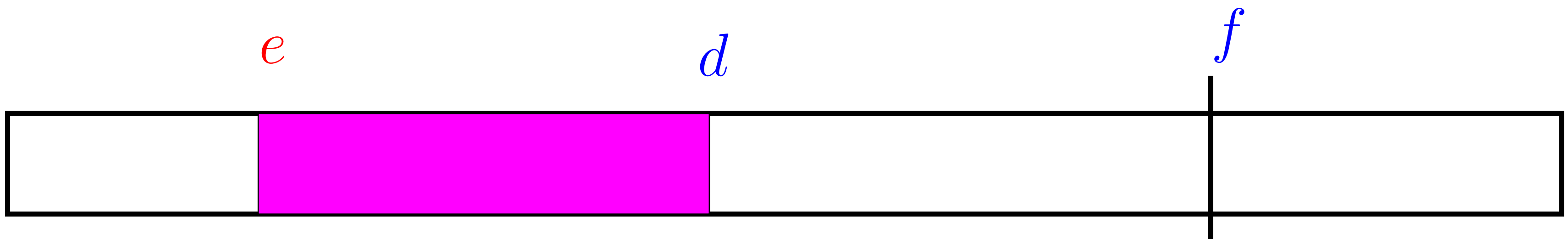
			e	d	f					n
A	31	−41	59	26	−53	58	97	−93	−23	84

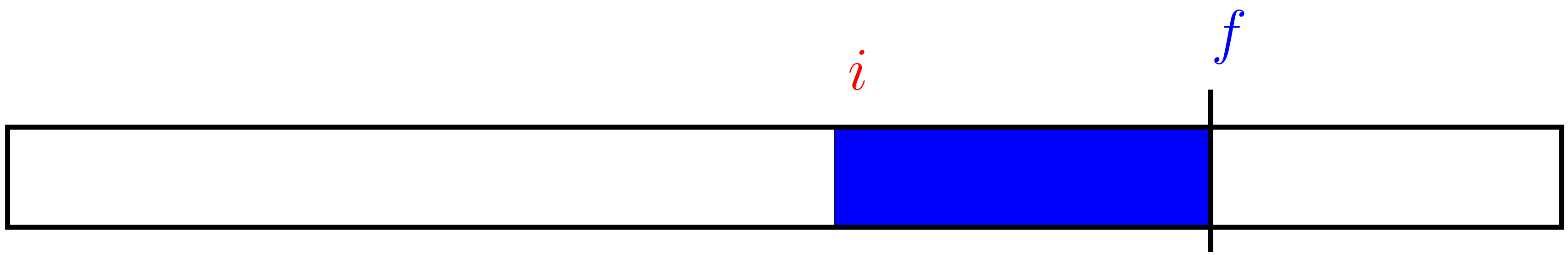
Mais uma relação invariante:

(i1) na linha 5 vale que:

$$somamax = A[e] + A[e+1] + A[e+2] + \cdots + A[d].$$

Invariantes (i0) e (i3)





Correção (continuação)

Mais relações invariantes.
Na linha 5 vale que:

(i2) $A[i\,..\,f]$ é segmento de soma máxima

com termino em f;
(i3) $soma = A[i] + A[i+1] + A[i+2] + \cdots + A[f]$;
(i4) $A[i] + A[i+1] + \cdots + A[k] \geq 0$
para todo $k = i, i+1, \ldots, f$;

Na linha 9 vale que:

(i5) $soma = A[i] + A[i+1] + \cdots + A[f] < 0$

As relações invariantes (i4) e (i5) implicam que na linha 9:

(i6) $A[k] + A[k+1] + \cdots + A[f] < 0$
para todo $k = i, i+1, \ldots, f$;

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	$= \Theta(1)$
3-4	$=$	2	$= \Theta(1)$
5	$=$	$n+1$	$= \Theta(n)$
6-7	$=$	n	$= \Theta(n)$
8	$=$	n	$= \Theta(n)$
9-10	$=$	n	$= \Theta(n)$
11	$\leq$	n	$= \mathrm{O}(n)$
12	$=$	1	$= \Theta(1)$
total	$=$	$\Theta(4n+3) + \mathrm{O}(n)$	$= \Theta(n)$

Conclusões

O consumo de tempo do algoritmo SEG-MAX-3 é $\Theta(n^3)$.

O consumo de tempo do algoritmo SEG-MAX-2 é $\Theta(n^2)$.

O consumo de tempo do algoritmo SEG-MAX-DC é $\Theta(n \lg n)$.

O consumo de tempo do algoritmo SEG-MAX-1 é $\Theta(n)$.

Técnicas

- Evitar recomputações. Usar espaço para armazenar resultados a fim de evitar recomputá-los (SEG-MAX-2, SEG-MAX-1, programação dinâmica).
- Pré-processar os dados. HEAPSORT pré-processa os dados armazenando-os em uma estrutura de dados a fim de realizar computações futuras mais eficientemente.
- Divisão-e-conquista. MERGE-SORT e SEG-MAX-DC utilizam uma forma conhecidade dessa técnica.
- Algoritmos incrementais/varredura. Solução de um subproblema é estendida a uma solução do problema original (SEG-MAX-1).
- Delimitação inferior. Projetistas de algoritmos só dormem em paz quando sabem que seus algoritmos são o melhor possível (SEG-MAX-1).

Análise experimental de algoritmos

"O interesse em experimentação, é devido ao reconhecimento de que os resultados teóricos, freqüentemente, não trazem informações referentes ao desempenho do algoritmo na prática."

Análise experimental de algoritmos

Segundo D.S. Johnson, pode-se dizer que existem quatro motivos básicos que levam a realizar um trabalho de implementação de um algoritmo:

- usar o código em uma aplicação particular, cujo propósito é descrever o impacto do algoritmo em um certo contexto;
- proporcionar evidências da superioridade de um algoritmo;
- melhor compreensão dos pontos fortes e fracos e do desempenho das operações algorítmicas na prática; e
- produzir conjecturas sobre o comportamento do algoritmo no caso-médio sob distribuições específicas de instâncias onde a análise probabilística direta é muito difícil.

Ambiente experimental

A plataforma utilizada nos experimentos é um PC rodando Linux Debian ?.? com um processador Pentium II de 233 MHz e 128MB de memória RAM .

Os códigos estão compilados com o gcc versão ?? e opção de compilação ??.

As instâncias são obtidas utilizando-se o gerador `gera.c` disponível em

`...~coelho/mac5711-2006/tarefa1/`.

As implementações comparadas neste experimento são SEG-MAX-3, SEG-MAX-2 e SEG-MAX-1.

Ambiente experimental

A estimativa do tempo é calculada utilizando-se:

```
#include <time.h>
[...]
clock_t start, end;
double time;

start = clock();

[...implementação...]

end  = clock();
time = ((double)(end - start))/CLOCKS_PER_SEC;
```

O tempo de entrada/saída é computado separadamente.

Resultados experimentais

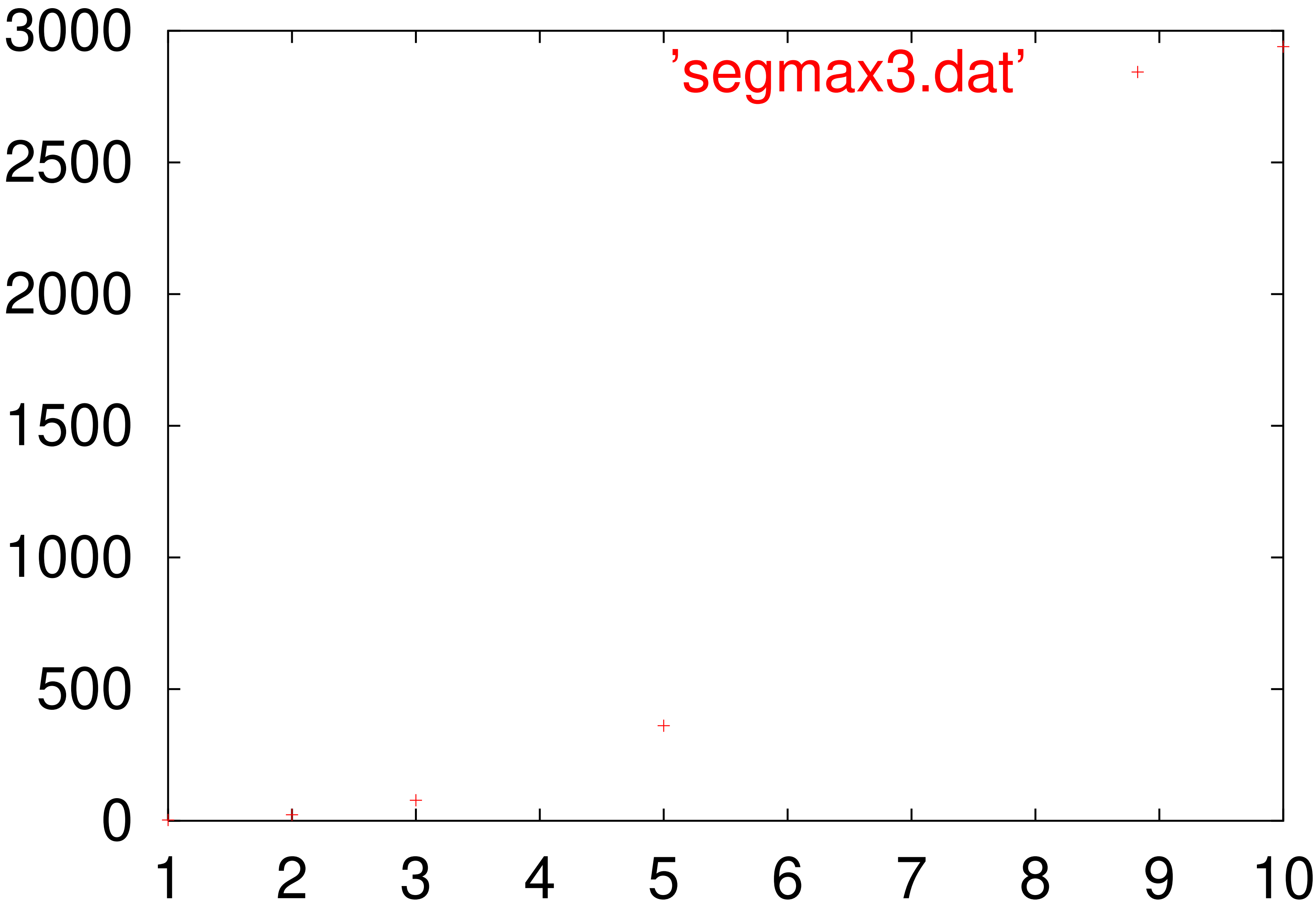
n	SEG-MAX-2		SEG-MAX-3	
	`clock`	`user time`	`clock`	`user time`
1000	0.01	0.01	2.91	2.92
2000	0.03	0.03	23.00	23.04
3000	0.08	0.08	77.58	1m17.57
5000	0.24	0.24	360.78	6m0.45s
10000	1.16	1.62	2940	49m27.32
20000	4.8	4.82	?	?
50000	42.8	42.96	?	?
100000	188.0	3m7.58s	?	?
200000	911.74	15m10.34s	?	?
400000	?	64m22.880s	?	?

Resultados experimentais

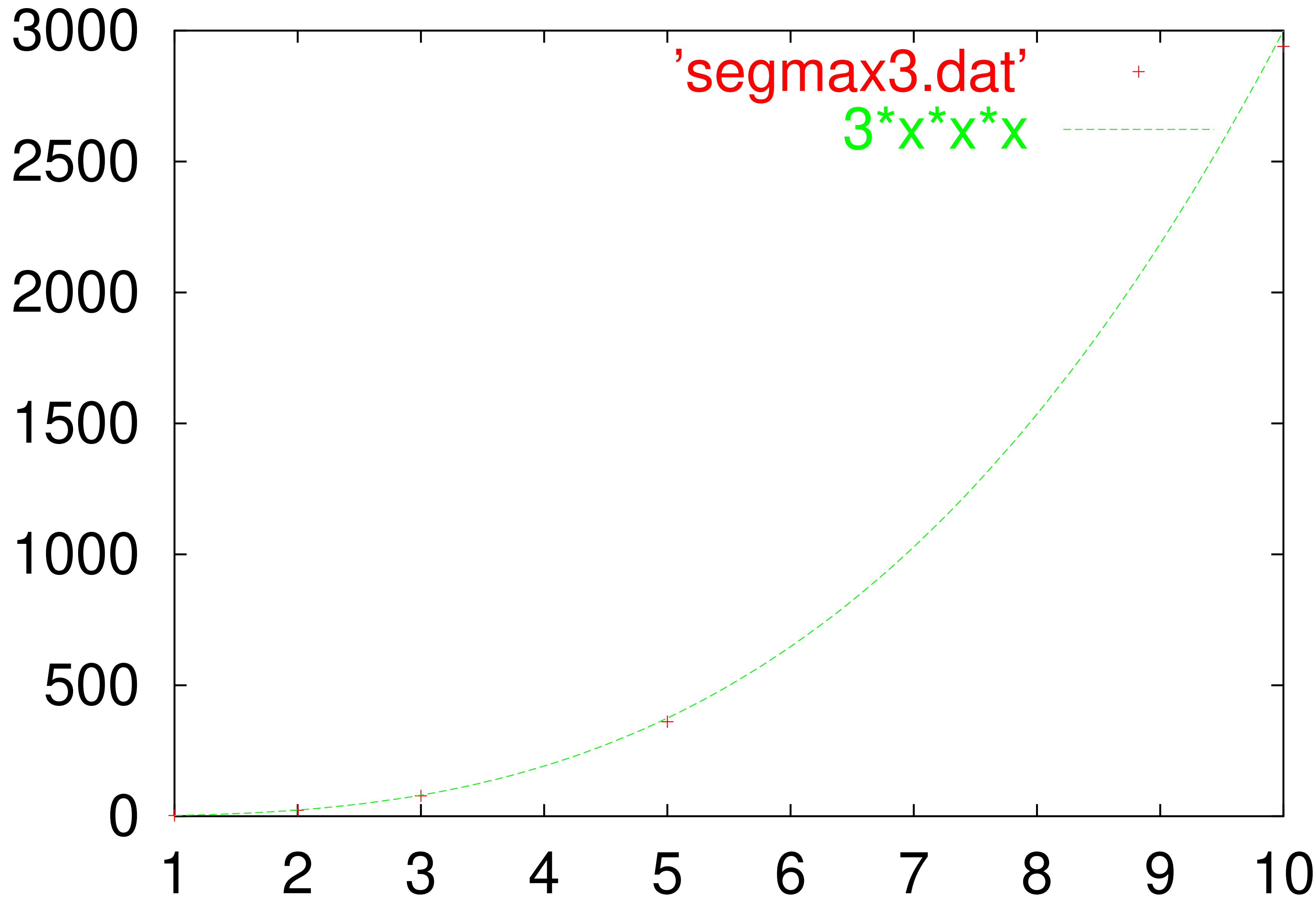
n (M)	SEG-MAX-1 A		SEG-MAX-1 B	
	`clock`	`user time`	`clock`	`user time`
1	0.06	0.11	0.08	0.07
2	0.11	0.23	0.15	0.06
3	0.16	0.28	0.30	0.12
6	0.32	0.63	0.56	0.29
8	0.43	0.80	0.71	0.43
10	0.55	1.03	1.00	0.52
15	0.82	1.55	1.37	0.64
20	1.08	2.03	2.17	0.95
25	1.37	2.52	2.44	1.38
30	1.70	3.13	3.28	1.68

Todos os tempos estão em segundos.

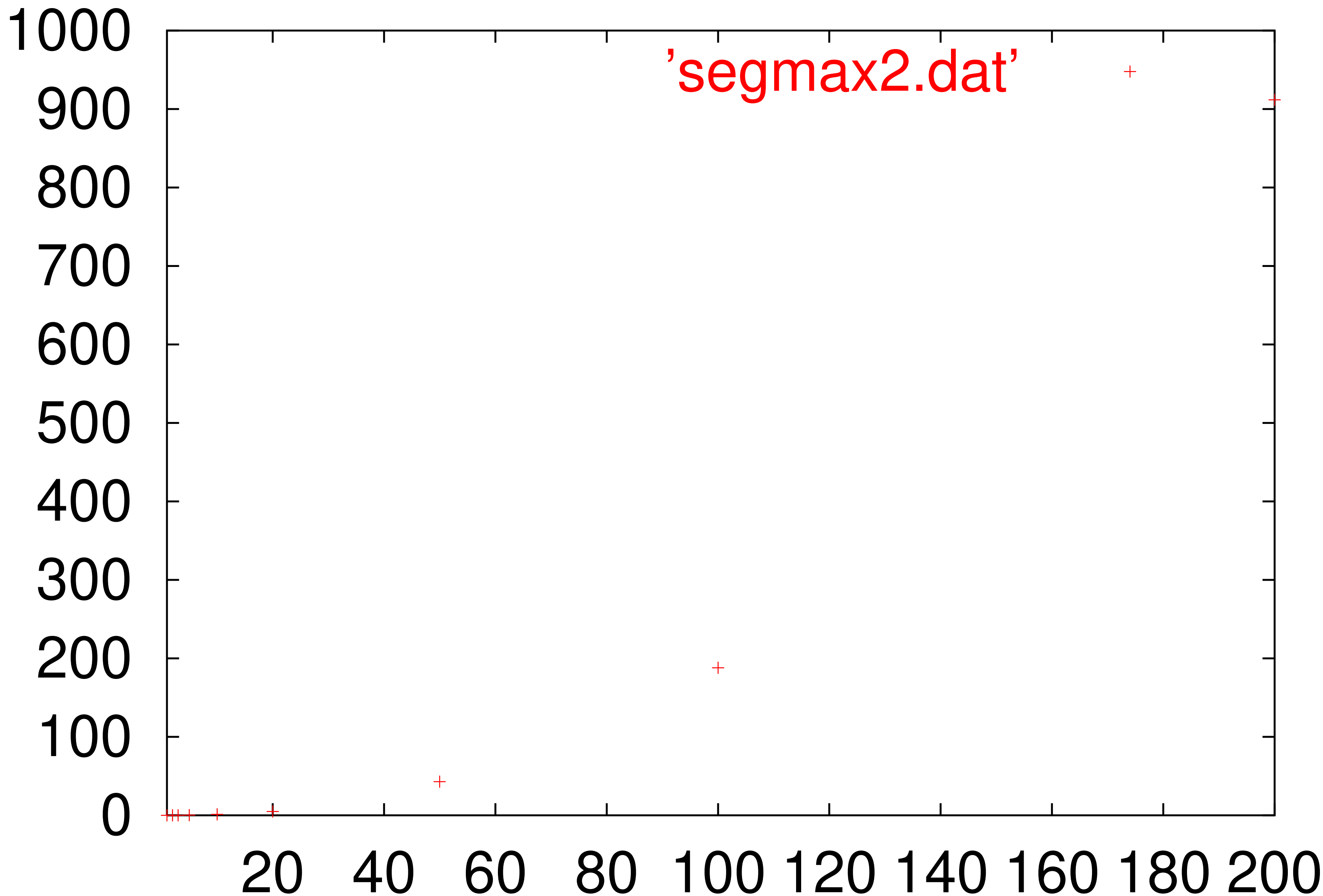
SEG-MAX-3



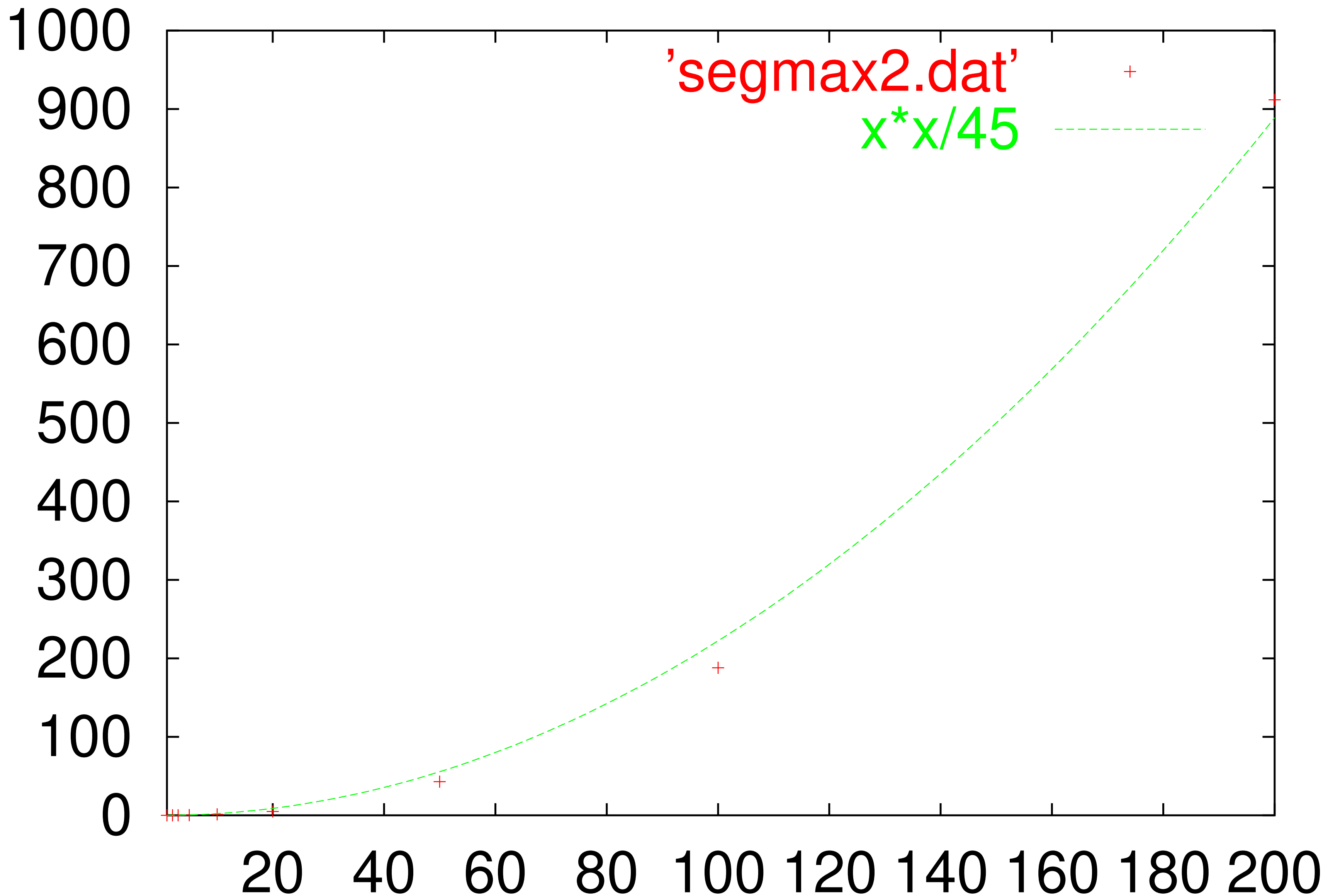
SEG-MAX-3



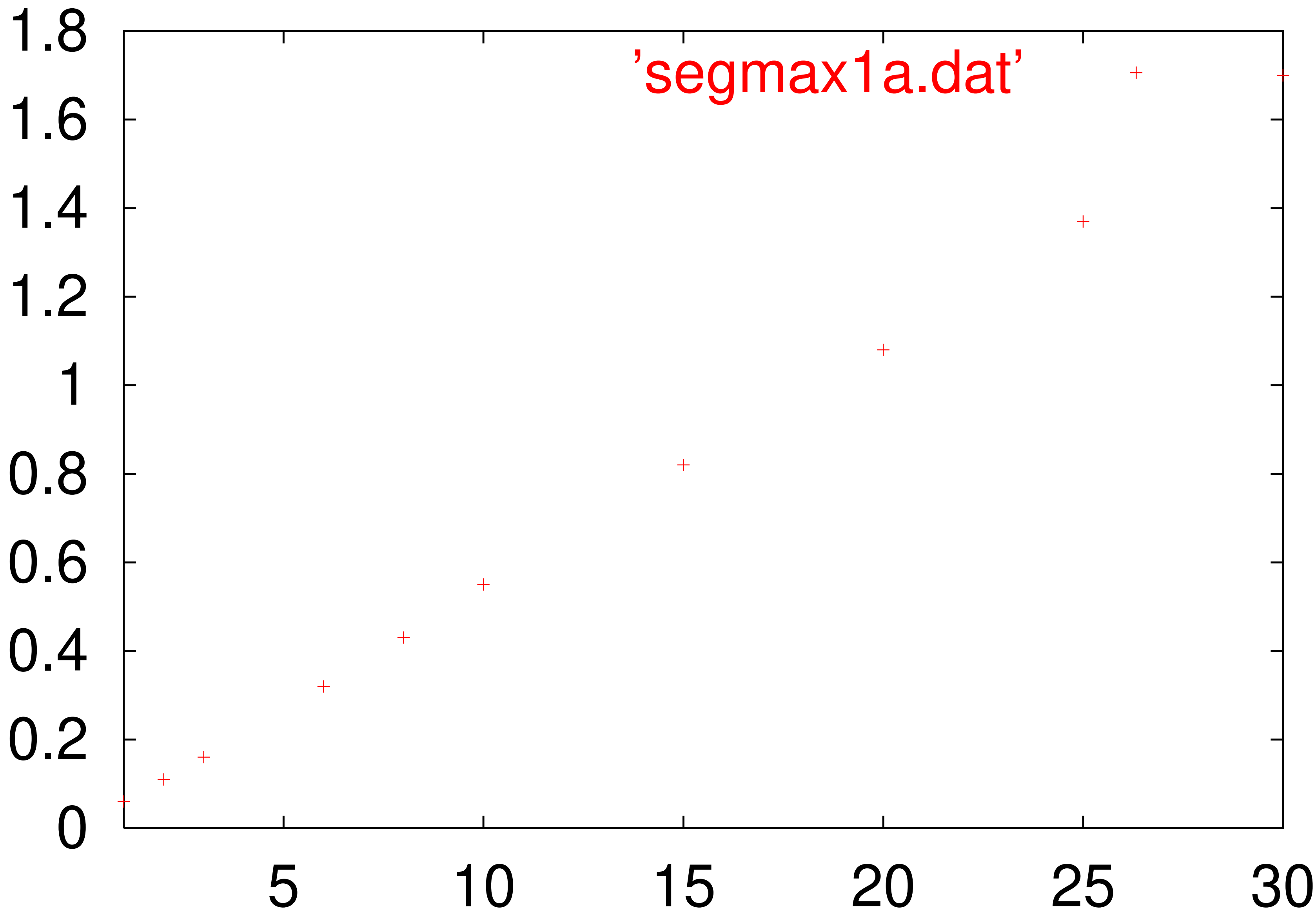
SEG-MAX-2


1000
900
800
700
600
500
400
300
200
100
0
20
40
60
80
100
120
140
160
180
200
'segmax2.dat'

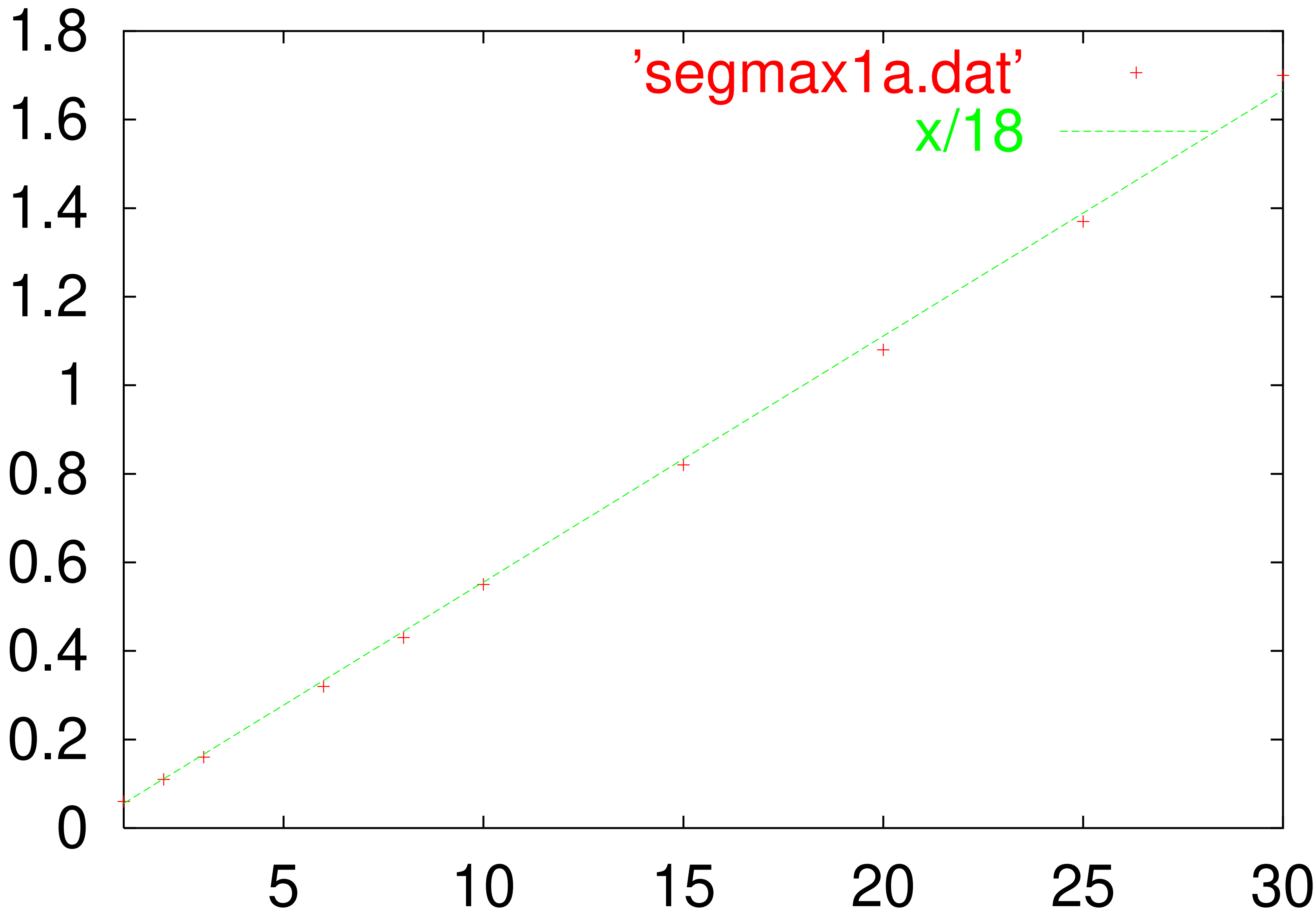
SEG-MAX-2


1000
900
800
700
600
500
400
300
200
100
0
20
40
60
80
100
120
140
160
180
200
'segmax2.dat'
x*x/45

SEG-MAX-1



SEG-MAX-1



Exercícios

Exercício 10.A
Nos algoritmos convencionamos o segmento vazio como sendo o de soma máxima se $A[1\,..\,n]$ tem apenas números negativos. Suponha que em vez disso tivéssemos decidido que nesse caso o maior elemento de $A[1\,..\,n]$ é o segmento de soma máxima. Como isto alteraria s quatro algoritmos?

Exercício 10.B
Suponha que desejamos determinar um segmento como soma mais próxima de zero, em vez do de soma máxima. Projete um algoritmo para esta tarefa. Qual é o consumo de tempo do seu algoritmo? E se estivéssemos interessados em um algoritmo para encontrar um segmento de soma mais próxima de um dado valor?

Exercício 10.C
No problema do retângulo de soma máxima é dada uma matriz $A[1\,..\,n, 1\,..\,n]$ de números inteiros e deseja-se determinar um retângulo de soma máxima. Projete um algoritmo para este problema. Qual o consumo de tempo do seu algoritmo?

Exercício 10.D
Modifique o algoritmo SEG-MAX-DC para que seu consumo de tempo seja linear.

Mais exercícios

Exercício 10.E
Demonstre que qualquer algoritmo para o problema do segmento de soma máxima deve examinar cada um dos n elementos do vetor. (Algoritmos para alguns problemas podem, corretamente, ignorar alguns dos dados; o algoritmo de Boyer e Moore para busca de padrões é um exemplo.)

Exercício 10.F
Projete um algoritmo que recebe um vetor $A[1\,..\,n]$ de números inteiros e um número inteiro m e devolve um índice i tal que $1 \leq i \leq n - m$ e a soma $A[i] + \cdots + A[i+m]$ seja próxima de zero. Qual o consumo de tempo do seu algoritmo.

Exercício 10.G
Resolva a recorrência

$$\begin{aligned} T(1) &= 0 \\ T(n) &= 2\,T(n/2) + n \quad \text{para } n = 2, 4, 8, \ldots \end{aligned}$$

Demonstre por indução que a sua solução está correta.