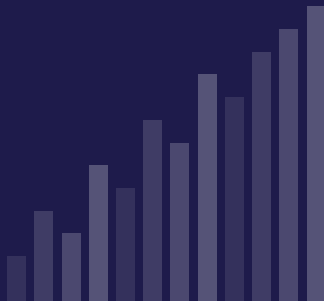

AULA 7 · BLOCO 1: FUNDAMENTOS, ORDENAÇÃO E CUSTO

Subarranjo máximo: três algoritmos, três ordens

Seção 4.1

Estruturas de dados e seus algoritmos · José Dihego · IFBA



Roteiro da aula

Ao final, você será capaz de

- ▶ Reduzir o problema de compra e venda de ações ao subarranjo máximo
- ▶ Resolver por força bruta, por divisão e conquista e pelo algoritmo linear
- ▶ Analisar o algoritmo de divisão e conquista pela recorrência $2T(n/2) + \Theta(n)$
- ▶ Medir $n^2/2$, $n \lg n$ e n operações



Teoria

O problema, a transformação, o subarranjo que cruza o meio



Prática em C

Divisão e conquista em C com o vetor da Figura 4.3



Experimento

Força bruta, divisão e conquista e Kadane lado a lado

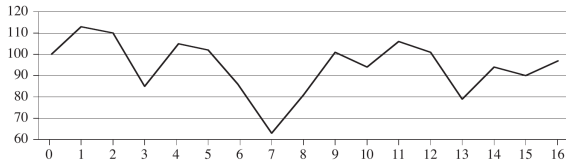


Conexões

Finanças, bioinformática, visão computacional e séries temporais

Comprar barato, vender caro

- ▶ Escolher o dia de compra e o de venda para maximizar o lucro.
- ▶ Nem sempre o mínimo global vem antes do máximo global.
- ▶ **Transformação:** olhe as **variações** diárias
 $A[i] = \text{preço}_i - \text{preço}_{i-1}$.
- ▶ O lucro de comprar no dia $i - 1$ e vender no dia j é $A[i] + \dots + A[j]$.



Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Price	100	113	110	85	105	102	86	63	81	101	94	106	101	79	94	90	97
Change		13	-3	-25	20	-3	-16	-23	18	20	-7	12	-5	-22	15	-4	7

Figura 4.1 de Cormen et al. (2009) — preço de uma ação ao longo de 17 dias e a variação diária.

O subarranjo máximo

- ▶ Força bruta: testar os $\binom{n}{2} + n$ pares (i, j) , com somas acumuladas: $\Theta(n^2)$.
- ▶ Só faz sentido se há valores negativos; se todos são positivos, a resposta é o vetor inteiro.
- ▶ Divisão e conquista: o melhor subarranjo está **à esquerda** do meio, **à direita** ou **cruzando** o meio.

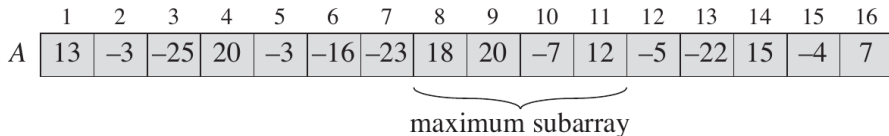


Figura 4.3 de Cormen et al. (2009) — o subarranjo $A[8..11]$ tem a maior soma: 43.

O subarranjo que cruza o meio

O subarranjo que cruza é **melhor sufixo da esquerda** + **melhor prefixo da direita**: dois laços simples, $\Theta(n)$ no total. É isso que torna a etapa de combinação linear.

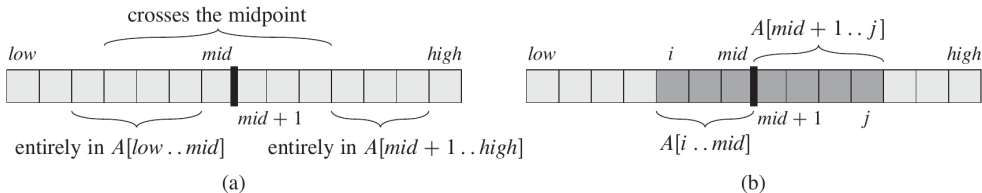


Figura 4.4 de Cormen et al. (2009) — (a) as três posições possíveis. (b) quem cruza o meio é um sufixo de $A[low \dots mid]$ mais um prefixo de $A[mid + 1 \dots high]$.

Divisão e conquista

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

A mesma recorrência do merge sort: caso 2 do teorema mestre, $\Theta(n \lg n)$.

Exercício 4.1-5: existe solução $\Theta(n)$. Guarde o melhor subarranjo que *termina* em j : ou é $A[j]$ sozinho, ou é o melhor que termina em $j - 1$ mais $A[j]$ (algoritmo de Kadane).

MAX-SUBARRANJO(A, lo, hi)

```

1  se  $hi = lo$  então
2      retorne ( $lo, hi, A[lo]$ )                                // caso base
3  senão
4       $mid \leftarrow \lfloor (lo + hi) / 2 \rfloor$ 
5       $(e_i, e_f, e_s) \leftarrow \text{MAX-SUBARRANJO}(A, lo, mid)$ 
6       $(d_i, d_f, d_s) \leftarrow \text{MAX-SUBARRANJO}(A, mid+1, hi)$ 
7       $(c_i, c_f, c_s) \leftarrow \text{MAX-CRUZANDO}(A, lo, mid, hi)$ 
8      se  $e_s \geq d_s$  e  $e_s \geq c_s$  então
9          retorne ( $e_i, e_f, e_s$ )
10     senão se  $d_s \geq e_s$  e  $d_s \geq c_s$  então
11         retorne ( $d_i, d_f, d_s$ )
12     senão
13         retorne ( $c_i, c_f, c_s$ )
    
```



Prática em C

- ▶ Função que devolve início, fim e soma (um struct)
- ▶ Dois laços para o subarranjo que cruza o meio
- ▶ Teste com o vetor da Figura 4.3: soma 43 em $A[8..11]$

Subarranjo máximo por divisão e conquista (1/3)

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  typedef struct { int ini, fim; long soma; } Faixa;
5
6  /* FIND-MAX-CROSSING-SUBARRAY: o melhor subarranjo que cruza o meio. */
7  Faixa cruzando(const int A[], int lo, int meio, int hi) {
8      long s = 0, esq = -__LONG_MAX__, dir = -__LONG_MAX__;
9      int i_max = meio, j_max = meio + 1;
10     for (int i = meio; i >= lo; i--) {           // melhor sufixo de A[lo..meio]
11         s += A[i];
12         if (s > esq) { esq = s; i_max = i; }
13     }
```

Subarranjo máximo por divisão e conquista (2/3)

```
14     s = 0;
15     for (int j = meio + 1; j <= hi; j++) {           // melhor prefixo de A[meio+1..hi]
16         s += A[j];
17         if (s > dir) { dir = s; j_max = j; }
18     }
19     return (Faixa){i_max, j_max, esq + dir};
20 }
21
22 Faixa maximo(const int A[], int lo, int hi) {
23     if (lo == hi) return (Faixa){lo, hi, A[lo]};    // caso base: um elemento
24     int meio = (lo + hi) / 2;
25     Faixa e = maximo(A, lo, meio), d = maximo(A, meio + 1, hi);
26     Faixa c = cruzando(A, lo, meio, hi);
27     if (e.soma >= d.soma && e.soma >= c.soma) return e;
28     if (d.soma >= e.soma && d.soma >= c.soma) return d;
29     return c;
```

Subarranjo máximo por divisão e conquista (3/3)

```
30 }
31
32 int main(void) {
33     int n;
34     if (scanf("%d", &n) != 1 || n <= 0) return 1;
35     int *A = malloc(n * sizeof(int));
36     for (int k = 0; k < n; k++)
37         if (scanf("%d", &A[k]) != 1) return 1;
38     Faixa f = maximo(A, 0, n - 1);
39     // + 1: índices como no livro (1..n)
40     printf("soma %ld de A[%d..%d]\n", f.soma, f.ini + 1, f.fim + 1);
41     free(A);
42     return 0;
43 }
```



Experimento

- ▶ Mesma resposta, custos muito diferentes
- ▶ $n = 256$ a 16.384 , valores em $[-100, 100]$
- ▶ O programa aborta se os três discordarem

🧪 EXPERIMENTO

n^2 , $n \lg n$ **ou** n ?

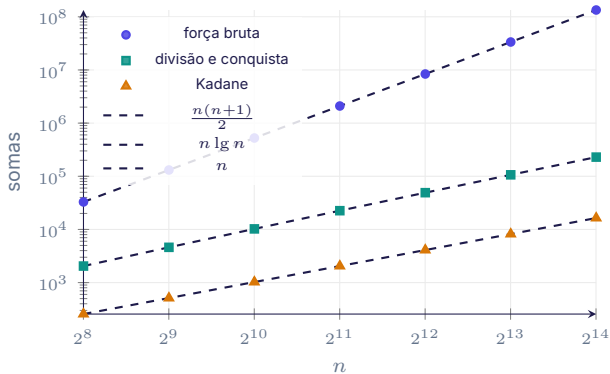
Hipótese. Contando somas de elementos: força bruta = $n(n + 1)/2$; divisão e conquista = $n \lg n$ (cada nível da recursão percorre o vetor uma vez); Kadane = n . Os três devem dar a **mesma** soma máxima.

Método

1. Vetores aleatórios com valores entre -100 e 100
2. $n = 2^8, \dots, 2^{14}$
3. Contador ops em cada soma
4. Verificação cruzada das três respostas

🧪 EXPERIMENTO

Três algoritmos, mesma resposta



n	medido	razão
256,0	32.896	1,000
512,0	131.328	1,000
1.024	524.800	1,000
2.048	2.098.176	1,000
4.096	8.390.656	1,000
16.384	134.225.920	1,000

$$\text{razão} = \text{medido} \div \frac{n(n+1)}{2}$$

✓ desvio máx. 0,0%

Para $n = 16.384$: 134 milhões de somas na força bruta, 229 mil na divisão e conquista, 16 mil no Kadane.

EXPERIMENTO

O que o experimento mostrou

As contagens são **exatamente** $n(n+1)/2$, $n \lg n$ e n , e as três respostas coincidem em todos os testes. Divisão e conquista melhora muito a força bruta, mas uma ideia melhor (programação dinâmica, aula 33) chega a $\Theta(n)$: **nenhum paradigma é a última palavra.**

>_ Reproduza: `gcc -O2 aula07_experimento.c -o exp -lm && ./exp`



Conexões

- ▶ Finanças
- ▶ Bioinformática
- ▶ Visão computacional
- ▶ Monitoramento

Onde isso aparece na computação

 Finanças

Melhor janela de compra e venda, maior sequência de lucro ou de perda (*drawdown*) numa carteira: o problema desta aula com dados reais.

 Bioinformática

Atribuindo pesos a aminoácidos ou nucleotídeos, o segmento de soma máxima localiza regiões transmembranares em proteínas e ilhas CpG no DNA.

 Visão computacional

A versão 2D, o retângulo de soma máxima, encontra a região mais clara ou mais "ativa de uma imagem em $O(n^3)$, com Kadane nas colunas.

 Monitoramento

Em séries temporais (CPU, batimentos, tráfego), subtrair a média e achar o subarranjo máximo localiza a janela de maior anomalia.

Para casa

Exercícios

1. (4.1-1) O que MAX-SubARRANJO devolve se todos os elementos são negativos?
2. (4.1-2) Escreva a força bruta e prove que é $\Theta(n^2)$.
3. (4.1-3) Implemente as duas versões e ache o n_0 em que a divisão e conquista passa a vencer.
4. (4.1-5) Implemente o algoritmo linear e prove sua corretude com um invariante.
5. Aplique o algoritmo às variações diárias de uma ação real (dados abertos da B3) e diga quando comprar e vender.

Leitura

Cormen et al., seção 4.1.

Código

josedihego.net/estruturas