Problema do Máximo Subarray
Uma Análise de sua Complexidade

Matheus Garay Trindadel[1], **Guilherme de Freitas Gaiardo**[1]

[1]Departamento de Eletrônica e Computação – Universidade Federal de Santa Maria (UFSM)
97.105-900 – Santa Maria – RS – Brazil

{mtrindade,ggaiardo}@inf.ufsm.br

Abstract. *This meta-paper describes the style to be used in articles and short papers for SBC conferences. For papers in English, you should add just an abstract while for the papers in Portuguese, we also ask for an abstract in Portuguese ("resumo"). In both cases, abstracts should not have more than 10 lines and must be in the first page of the paper.*

Resumo. *Este meta-artigo descreve o estilo a ser usado na confecção de artigos e resumos de artigos para publicação nos anais das conferências organizadas pela SBC. É solicitada a escrita de resumo e abstract apenas para os artigos escritos em português. Artigos em inglês deverão apresentar apenas abstract. Nos dois casos, o autor deve tomar cuidado para que o resumo (e o abstract) não ultrapassem 10 linhas cada, sendo que ambos devem estar na primeira página do artigo.*

1. Introdução

Em computação é recorrente encontrar problemas que, devido a limitações dos computares, não se é possível resolver em tempo hábil para certas entradas. Muitas vezes, algoritmos completamente funcionais para entradas pequenas se tornam inutilizáveis para entradas grandes. No intuito de prever o desempenho de algoritmos, podemos usar técnicas para estimar o crescimento da quantidade de instruções necessárias para resolver um dado problema em função do tamanho da entrada. Vale ressaltar que essas técnicas são independentes de arquitetura e simplesmente mostram como o custo computacional se comporta.

Para exemplificar esse conceito, esse artigo faz uma análise computacional do problema clássico do máximo subarray. Nessa análise, serão apresentadas apresentados três algoritmos que resolvem esse problema. Cada um desses algoritmos possui uma complexidade, i.e., um crescimento do custo computacional, diferente. Essas complexidades serão demonstradas usando diversas técnicas, dependendo da estrutura do algoritmo.

2. Problema do Subarray Máximo

O clássico problema do subarray máximo consiste do seguinte: dado um array de valores contendo valores negativos, encontrar o subarray que, dentro do conjunto de subarrays do array original, se somarmos os valores de seus elementos, possui a maior soma. Por exemplo, podemos tomar o array A apresentado na sequência:

$$A = [13, -3, -25, -3, -16, -23, 18, 20, -7, 12, -5, -22, 15, -4, 7]$$

Observando o array, percbemos que o subarray com a soma máxima consite em:

$$[18, 20, -7, 12]$$

A soma desse subarray corresponde a 43. Nenhum outro subarray possui uma soma superior a esta. O problema então consiste em escrever um algoritmo para analisar o conjunto de subarrays e achar o que maximiza a soma de seus elementos.

3. Soluções para o Problem

A seguir, serão discutidas diversas formas de resolver o problema. Além disso, uma extensiva análise da solução será feita para demonstrar a complexidade em cada caso.

3.1. Força Bruta

A solução talvez mais óbvia e simples consiste em, exaustivamente, calcular todos os subarrays e compará-los entre si. O algoritmo consiste em dois laços aninhados para montar todos os subarrays. A soma é computada e salva. Se a soma for maior que a maior soma corrente (ou não existir soma corrente), ela passa ser a maior soma e o subarray máximo passa a ser o subarray que gerou esta soma. A solução é melhor apresentada no seguinte pseudocódigo:

Algorithm 1 Força Bruta

```
1:  function MAXSUBARRAY(A)
2:      maxSoma ← −∞                                   ▷ 1
3:      n ← len(A)                                     ▷ 1
4:      i0 ← −1                                        ▷ 1
5:      j0 ← −1                                        ▷ 1
6:      for i ∈ [1 : n] do                             ▷ n + 1
7:          soma ← 0                                   ▷ n
8:          for j ∈ [i : n] do                         ▷ Σ_{i=1}^{n+1} i = (n²+3n+2)/2
9:              soma ← soma + A[j]                     ▷ Σ_{i=1}^{n} i = (n²+n)/2
10:             if soma > maxSoma then                 ▷ Σ_{i=1}^{n} i = (n²+n)/2
11:                 maxSoma ← soma                     ▷ Σ_{i=1}^{n} t_j
12:                 i0 ← i                             ▷ Σ_{i=1}^{n} t_j
13:                 j0 ← j                             ▷ Σ_{i=1}^{n} t_j
14:             end if
15:         end for
16:     end for
17:     return A[i0 : j0]                              ▷ 1
18: end function
```

4. Figures and Captions

Figure and table captions should be centered if less than one line (Figure 1), otherwise justified and indented by 0.8cm on both margins, as shown in Figure 2. The caption font must be Helvetica, 10 point, boldface, with 6 points of space before and after each caption.

Figura 1. A typical figure

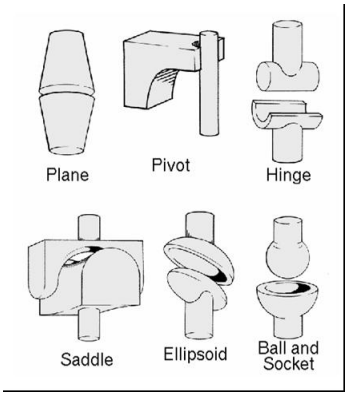

Figura 2. This figure is an example of a figure caption taking more than one line and justified considering margins mentioned in Section 4.

In tables, try to avoid the use of colored or shaded backgrounds, and avoid thick, doubled, or unnecessary framing lines. When reporting empirical data, do not use more decimal digits than warranted by their precision and reproducibility. Table caption must be placed before the table (see Table 1) and the font used must also be Helvetica, 10 point, boldface, with 6 points of space before and after each caption.

5. Images

All images and illustrations should be in black-and-white, or gray tones, excepting for the papers that will be electronically available (on CD-ROMs, internet, etc.). The image resolution on paper should be about 600 dpi for black-and-white images, and 150-300 dpi for grayscale images. Do not include images with excessive resolution, as they may take hours to print, without any visible difference in the result.

Tabela 1. Variables to be considered on the evaluation of interaction techniques

	Chessboard top view	Chessboard perspective view
Selection with side movements	6.02 ± 5.22	7.01±6.84
Selection with in-depth movements	6.29±4.99	12.22±11.33
Manipulation with side movements	4.66± 4.94	3.47±2.20
Manipulation with in-depth movements	5.71 ±4.55	5.37 ±3.28

6. References

Bibliographic references must be unambiguous and uniform. We recommend giving the author names references in brackets, e.g. [Knuth 1984], [Boulic and Renault 1991], and [Smith and Jones 1999].

The references must be listed using 12 point font size, with 6 points of space before each reference. The first line of each reference should not be indented, while the subsequent should be indented by 0.5 cm.

Referências

Boulic, R. and Renault, O. (1991). 3d hierarchies for animation. In Magnenat-Thalmann, N. and Thalmann, D., editors, *New Trends in Animation and Visualization*. John Wiley & Sons ltd.

Knuth, D. E. (1984). *The TEX Book*. Addison-Wesley, 15th edition.

Smith, A. and Jones, B. (1999). On the complexity of computing. In Smith-Jones, A. B., editor, *Advances in Computer Science*, pages 555–566. Publishing Press.