Análise Empírica do Algoritmo Shellsort

Raquel M. de Souza[1]*, **Fabiano de S. Oliveira**[1]†, **Paulo Eustáquio D. Pinto**[1]†

[1]Universidade do Estado do Rio de Janeiro, Brasil

raquelmarcolino25@gmail.com, {fabiano.oliveira,pauloedp}@ime.uerj.br

Abstract. *This work aims at the study of the time complexity of the Shellsort algorithm from an empirical perspective. The theoretical performance of this algorithm depends on a sequence of integers used. Several classical sequences are studied in the literature, for most of which the time complexity is known. However, for some sequences, the exact time complexity is still open. Our approach is to study such classical sequences empirically to confirm the known time complexities and to gather evidences for those still unknown.*

Resumo. *O objetivo deste trabalho é estudar a complexidade de tempo do algoritmo de ordenação Shellsort de um ponto de vista empírico. O desempenho teórico deste algoritmo depende de uma sequência de inteiros usada. Diversas sequências clássicas são estudadas na literatura, para a maioria das quais a complexidade de tempo é conhecida. No entanto, para algumas, ainda não se conhece a complexidade de tempo justa. Nossa abordagem foi a de estudar tais sequências clássicas empiricamente para ratificar as complexidades de tempo conhecidas e juntar evidências sobre aquelas desconhecidas.*

1. Introdução

O algoritmo *Shellsort* é um algoritmo de ordenação por comparação proveniente de uma generalização do algoritmo de ordenação por inserção (*InsertionSort*). A ordenação por Shellsort constitui-se de execuções sequenciais de InsertionSort sobre subconjuntos tomados dos n elementos do vetor de entrada, sendo que o último subconjunto é aquele de todos os elementos (ou seja, é uma execução padrão do *InsertionSort*). Esta estratégia tem potencial de obter melhor desempenho quando (i) as ordenações anteriores à última atuam sobre subconjuntos de elementos de tamanhos controlados, de modo que o custo do pior caso para estas execuções iniciais tenham peso reduzido no custo total; e (ii) os elementos escolhidos para serem pré-ordenados são tais que evitam o pior caso quadrático do último *InsertionSort*, de modo que a diminuição do custo na última execução compense as ordenações parciais anteriores. A motivação de Shell [Shell 1959] ao introduzir esta ideia em 1959 foi a de tentar fazer com que o *InsertionSort*, um algoritmo de ordenação local (isto é, que faz uso de memória auxiliar constante, que não é o caso, por exemplo, do *MergeSort*) com complexidade de tempo de pior caso $\Theta(n^2)$, fosse tão eficiente quanto possível para um algoritmo de ordenação por comparação (como é o *MergeSort*, com tempo ótimo de $\Theta(n \log n)$).

Mais especificamente, o *Shellsort* escolhe tais subconjuntos de elementos que serão pré-ordenados através da noção de *sequência de passos*. Se $S = p_1, p_2, p_3, \ldots$, com

*Projeto financiado por CAPES.

†Projeto parcialmente financiado por FAPERJ.

$p_1 = 1$, é a sequência de passos a ser usada num algoritmo de *Shellsort* e p_k é o maior passo menor que n, então o Shellsort é descrito pelo Algoritmo 1.

Entrada: $V[1..n]$ de inteiros
Resultado: V ordenado
para $j \leftarrow k$ **até** 1 **passo** -1 **faça**
  para $i \leftarrow 1$ **até** p_j **faça**
    InsertionSort(V, $\{ i, i + p_j, i + 2p_j, \ldots, i + tp_j \}$), com $t = \left\lfloor \dfrac{n - i}{p_j} \right\rfloor$

Algoritmo 1: Pseudo-código do Shellsort. A chamada InsertionSort(V, S) ordena os elementos de V nas posições contidas em S por *InsertionSort*.

Desde sua introdução, o *Shellsort* foi objeto de investigação por diversos autores (ver Tabela 1). Note que, para algumas sequências, a complexidade de tempo resultante foi mostrada ser $o(n^2)$, como por exemplo, aquelas estudadas em [Frank and Lazarus 1960, Hibbard 1963, Pratt 1972] de tempo $O(n^{3/2})$ e em [Incerpi and Sedgewick 1983] de tempo $O(n^{1+\sqrt{8 \cdot \ln(5/2)/\ln(n)}})$. Note também que algumas destas sequências possuem sua complexidade de pior caso exata desconhecida.

Neste trabalho, é feita uma análise empírica da complexidade de tempo de várias das sequências da Tabela 1 com objetivo inicial de confirmar aquelas complexidades já conhecidas. Com este passo, validamos o processo de aferição empírica da complexidade. Em seguida, passamos a usar o mesmo método para conjecturar as complexidades de tempo desconhecidas. Em especial, analisamos a complexidade de caso médio das diversas sequências, desconhecidas por análise quase na totalidade.

2. Medida empírica da complexidade de tempo

A ferramenta EMA (*EMpirical analysis of Algorithms*) foi desenvolvida para estimar a complexidade de tempo e espaço de algoritmos [Oliveira 2016]. Mais especificamente, a proposta do EMA é fornecer a complexidade que mais se ajusta aos resultados de experimentos planejados e executados pelo próprio EMA. Os experimentos estão sujeitos a entradas do usuário (como tempo/espaço máximo permitido de execução) que influenciam na qualidade da estimação. A entrada do EMA consiste de: (i) um programa $\mathcal{A}$ que será analisado; (ii) a lista $\mathcal{V} = v_1, v_2, \ldots, v_r$ de variáveis das quais a complexidade de tempo/espaço de $\mathcal{A}$ dependa (para o caso do Shellsort, $\mathcal{V} = n$); e (iii) um programa $\mathcal{B}$ que tem por entrada uma valoração $V_1, \ldots, V_r$ das respectivas variáveis $v_1, \ldots, v_r$ e produz uma entrada $\mathcal{I}$ para $\mathcal{A}$ na qual $v_i = V_i$ para todo $1 \leq i \leq r$. O processamento do EMA segue as seguintes etapas: (i) (*calibração*) para todo $1 \leq i \leq r$, determina valores $V_i^{\max}$ de modo que o tempo de execução de $\mathcal{A}$ fique limitado a certa constante e que o espaço de memória utilizado não ultrapasse a memória física livre do sistema para toda entrada $\mathcal{I}$ na qual $v_i \leq V_i^{\max}$; (ii) (*simulações*) define-se um subconjunto $S_i \subseteq \{ 1, \ldots, V_i^{\max} \}$ para todo $1 \leq i \leq r$ com valores de v_i que serão analisados: para cada $(V_1, \ldots, V_r) \in S_1 \times \cdots \times S_r$, $\mathcal{B}$ é usado para gerar uma entrada $\mathcal{I}$ de $\mathcal{A}$ na qual $v_i = V_i$ para $1 \leq i \leq r$, e, em seguida, $\mathcal{A}$ é executado tendo $\mathcal{I}$ por entrada, armazenando ao final o tempo de execução e o espaço máximo de memória ocupado. Portanto, forma-se uma base de dados com $\prod_{i=1}^{r} |S_i|$ execuções; e (iii) (*análise*) por meio de várias regressões não-lineares, estima-se uma forma funcional para a complexidade de tempo/espaço.

Para o caso de uma variável, se $T(n)$ é a função que representa o recurso (tempo ou espaço) requerido por $\mathcal{A}$ sob entrada na qual $v_1 = n$, o EMA determinará os parâmetros que minimizam a soma dos quadrados dos resíduos da equação

$$T(n) = a_0 n^{a_1} a_2^{n^{a_3}} \log n^{a_4} \log^{a_5} n, \text{ com } a_0, a_2 > 0.$$

Tabela 1. Sequências de passos estudadas na literatura para o Shellsort.

Termo geral [Autor Ano]	**Sequência**	**Pior caso**
$\left\lfloor \frac{n}{2^k} \right\rfloor, k \geq 1$ [Shell 1959]	$\left\lfloor \frac{n}{2} \right\rfloor, \left\lfloor \frac{n}{4} \right\rfloor, ..., 1$	$\Theta(n^2)$ [Frank and Lazarus 1960]
$2^k - 1, k \geq 1$ [Hibbard 1963]	$1, 3, 7, 15, 31, 63, ...$	$\Theta(n^{3/2})$ [Pratt 1972]
$2^p 3^q$, $p \in \mathbb{N}$ e $q \in \mathbb{N}$ [Pratt 1972]	$1, 2, 3, 4, 6, 8, ...$	$\Theta(n \log^2 n)$ [Pratt 1972]
$a_k = 3 \cdot a_{k-1} + 1, k \geq 2$ e $a_1 = 1$ [Knuth 1973]	$1, 4, 13, 40, 121, ...$	$\Theta(n^{3/2})$ [Pratt 1972]
$\prod_{0<k<r \quad k \neq ((r^2+r)/2)-q} a_k$, onde $r = \left\lfloor \sqrt{2q + \sqrt{2q}} \right\rfloor$ e $a_k = \min\Big(m \in \mathbb{N} : m \geq \left(\frac{5}{2}\right)^{k+1}, \forall p : 0 \leq p < k \Rightarrow \text{mdc}(a_p, m) = 1\Big)$ [Incerpi and Sedgewick 1983]	$1, 3, 7, 21, 48, ...$	$O(n^{1+\sqrt{8 \ln (5/2)/ \ln n}})$ [Incerpi and Sedgewick 1983]
$4^k + 3 \cdot 2^{k-1} + 1, k \geq 2$ [Sedgewick 1986]	$1, 8, 23, 77, 281, ...$	$O(n^{4/3})$ [Sedgewick 1986]
$a_k = \max\left(\left\lfloor \frac{5a_{k-1}}{11} \right\rfloor, 1\right), k \geq 2$ e $a_1 = n$ [Gonnet and Baeza-Yates 1991]	$\left\lfloor \frac{5n}{11} \right\rfloor, \left\lfloor \frac{5}{11} \left\lfloor \frac{5n}{11} \right\rfloor \right\rfloor, ..., 1$	em aberto
$\left\lceil \frac{9^k - 4^k}{5 \cdot 4^{k-1}} \right\rceil, k \geq 1$ [Tokuda 1992]	$1, 4, 9, 20, 46, ...$	em aberto
Sequência obtida empiricamente [Ciura 2001]	$1, 4, 10, 23, 57, 132, 301, 701, 1750$	em aberto

3. Resultados Obtidos e Conclusão

A Tabela 2 sumariza as complexidades de tempo obtidas. Na fase das simulações, o EMA utilizou a faixa $2 \times 10^6 \leq n \leq 25 \times 10^6$, amostrando 100 pontos neste intervalo ($|S_1| = 100$). Na análise de caso médio, foram feitos 30 sorteios com distribuição uniforme de vetores para cada valor de n amostrado, tomando-se a média dos tempos obtidos. Os experimentos foram conduzidos em um computador com processador Intel Core(TM) $i7 - 3770$ CPU @ 3.40GHz de 8 núcleos e 16GB de memória RAM. Como conclusões, notamos que as complexidades empíricas concordam com aquelas analíticas existentes, sugerindo as complexidades empíricas como candidatas a analíticas quando estas são desconhecidas. Além disso, notamos que o *Shellsort* atinge na prática o objetivo original de ser ótimo (ou próximo de) para várias sequências, ao menos no seu caso médio.

Este trabalho foi desenvolvido como projeto de graduação, que se transformou recentemente em estudo de mestrado. Visamos caracterizar o pior caso do *Shellsort* de maneira algorítmica, de forma que as complexidades empíricas obtidas neste caso sejam

referentes ao pior caso, e não ao caso médio. O objetivo é acumular evidências para o preenchimento da lacuna de conhecimento do tempo de pior caso do *Shellsort* para algumas sequências. Os resultados parciais nesta linha de investigação parecem promissores.

Tabela 2. Complexidades analisadas.

Sequência [Autor Ano]	**Existente**	**Obtida**
	Pior Caso Analítico	***Empírico***
Shell, 1959	$\Theta(n^2)$ [Frank and Lazarus 1960]	$\Theta(n^2)$
Pratt, 1971	$\Theta(n \lg^2 n)$ [Pratt 1972]	$\Theta(n \lg^2 n)$
	Caso Médio Analítico	***Empírico***
Pratt, 1971	$\Theta(n \lg^2 n)$ [Pratt 1972]	$\Theta(n \lg^2 n)$
	Caso Médio Empírico	***Empírico***
Shell, 1959	$\Theta(n^{1,226})$ [Shell 1959]	$\Theta(n \lg^3 n)$
Hibbard, 1963	$\Theta(n^{1,26})$ ou $\Theta(n \lg^2 n)$ [Knuth 1973]	$\Theta(n^{1,203})$
Knuth, 1973	$\Theta(n^{1,25})$ ou $\Theta(n \lg^2 n)$ [Weiss 1991]	$\Theta(n \lg^3 n)$
Incerpi e Sedgewick, 1985	em aberto	$\Theta(n \lg n)$
Sedgewick, 1986	$\Theta(n^{7/6})$ [Weiss 1991]	$\Theta(n \lg^2 n)$
demais autores (a partir de 1991)	em aberto	$\Theta(n \lg n)$

Referências

Ciura, M. (2001). Best increments for the average case of shellsort. In *Fundamentals of Computation Theory*, pages 106–117. Springer.

Frank, R. M. and Lazarus, R. B. (1960). A high-speed sorting procedure. *Communications of the ACM*, 3(1):20–22.

Gonnet, G. H. and Baeza-Yates, R. A. (1991). Handbook of algorithms and data structures: in pascal and c.

Hibbard, T. N. (1963). An empirical study of minimal storage sorting. *Communications of the ACM*, 6(5):206–213.

Incerpi, J. and Sedgewick, R. (1983). Improved upper bounds on shellsort. In *Foundations of Computer Science, 1983., 24th Annual Symposium on*, pages 48–55. IEEE.

Knuth, D. E. (1973). The art of computer programming 3: Sorting and searching.

Oliveira, F. S. (2016). EMA - WebPage. `http://fabianooliveira.ime.uerj.br/ema`. [última vez acessado: 07 de Junho de 2016].

Pratt, V. R. (1972). Shellsort and sorting networks. Technical report, DTIC Document.

Sedgewick, R. (1986). A new upper bound for shellsort. *Journal of Algorithms*, 7(2):159–173.

Shell, D. L. (1959). A high-speed sorting procedure. *Communications of the ACM*, 2(7):30–32.

Tokuda, N. (1992). An improved Shellsort. In *12th World Computer Congress on Algorithms, Software, Architecture-Information Processing*, pages 449–457. N.-Holland.

Weiss, M. A. (1991). Short note empirical study of the expected running time of shellsort. *The Computer Journal*, 34(1):88–91.