SMOOTHSORT: o elo perdido da Ordenação

André Luiz Pfitzner
Paulo E. Duarte Pinto
Rosa Maria E. M. da Costa

Universidade do Estado do Rio de Janeiro – IME/DICC
20550-013, Rua São Francisco Xavier, 524, Maracanã, Rio de Janeiro, RJ
E-mail: andrepfitzner@gmail.com,
pauloedp@ime.uerj.br,
rcosta@ime.uerj.br

RESUMO

O *Smoothsort* [3] é um método de ordenação desenvolvido por Dijkstra, fortemente inspirado no *Heapsort*, com uma particularidade especialíssima de ter complexidade de melhor caso O(n) e de pior caso O(n log n), sendo n o número de dados a serem ordenados. O melhor caso ocorre quando os dados já estão ordenados. Além disso, a complexidade varia de forma suave, quando se passa da situação de dados totalmente ordenados para a de dados aleatórios. Essa característica é única no conjunto dos métodos de ordenação importantes e utilizados na prática. Por exemplo, três destacados métodos: *Heapsort, Mergesort e Quicksort,* têm complexidade de melhor caso O(n log n) [1]. Como a situação de ter que ordenar conjuntos totalmente ordenados ou quase ordenados ocorre frequentemente nas aplicações reais, seria de se esperar que o método em referência ocupasse um lugar de destaque tanto na teoria quanto na prática de algoritmos. Entretanto isso não ocorre. O método é praticamente desconhecido e a literatura produzida sobre o mesmo é pífia. Recentemente, Pfitzner [4] apresentou uma coleção de ferramentas voltadas para o ensino/aprendizagem do método e de comparação com os demais métodos, confirmando a boa performance esperada para o *Smoothsort*. Ou seja, o método parece ser um elo perdido na evolução dos algoritmos de ordenação.

Neste artigo, fazemos uma breve descrição do método e apresentamos alguns resultados de análise do algoritmo, que complementam e melhoram a análise resumida feita pelo autor, na descrição original do *Smoothsort [3]*.

Para ordenar dados guardados em um vetor, o *Smoothsort* cria, inicialmente, uma "floresta de Leonardo". Essa floresta é constituída de árvores com links virtuais, sendo cada uma delas um *Heap*, tal como no *Heapsort*. Os números de elementos das árvores da floresta criada são números de Leonardo, análogos aos números de Fibonacci, e representados pela seqüência 1 1 3 5 9 ..., onde cada número, a partir do segundo, é a soma dos anteriores acrescida de 1. Na floresta criada, as raízes estão ordenadas e os tamanhos das árvores são decrescentes. Isso vale também para as subárvores de uma árvore. A Figura 1 ilustra a floresta com 3 árvores criadas na primeira parte do método, para 13 chaves. Após a criação das árvores, que é feita examinando-se os dados da esquerda para a direita, começa-se um processo de destruição da floresta, agora da direita para a esquerda, mantendo-se sempre o princípio de que a raiz da árvore mais à direita é a maior de todas as chaves do vetor. O algoritmo funciona com complexidade linear no caso dos dados já estarem ordenados, porque nenhuma troca é feita, tanto na criação da floresta, quanto na sua destruição.

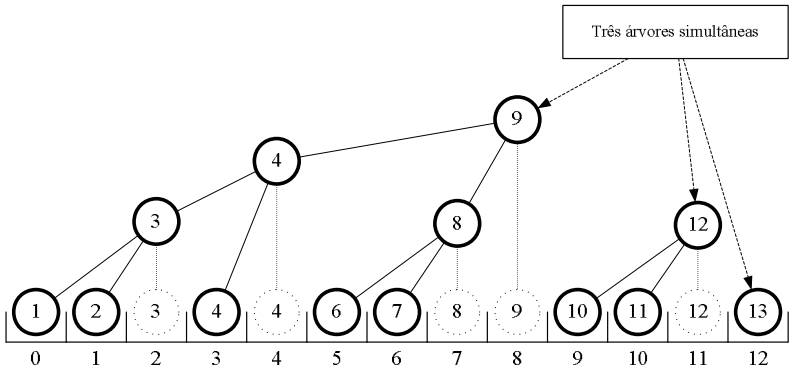

Figura 1 - Três árvores na floresta

Alguns dos resultados da análise do algoritmo são apresentados nos teoremas 1 a 4, enunciados e sucintamente explicados a seguir.

Teorema 1: "O número de árvores de Leonardo criadas pelo algoritmo *Smoothsort* **é mínimo".**
Isso decorre da demonstração, feita a partir de [2] de que o conjunto dos k primeiros números de Leonardo L_1,L_2,...L_k (excluindo-se L_0), para qualquer k, é um "conjunto de moedas totalmente guloso". Desta forma, para um vetor de tamanho n, os tamanhos das árvores que compõem a floresta de Leonardo são determinados gulosamente: a árvore mais à esquerda tem número de elementos igual ao maior número de Leonardo contido em n, e assim sucessivamente.

Teorema 2: "A memória adicional necessária para o *Smoothsort* **é O(log n)".**
Isso resulta da prova de que o número de árvores da floresta criada no primeiro passo do algoritmo é O(log n).

Teorema 3: "A complexidade de melhor caso do *Smoothsort* **é O(n)".**
Este resultado deriva do fato de que, na criação e destruição da floresta de Leonardo, todas as árvores são naturalmente Heaps, sem necessidade de nenhuma troca de chaves.

Teorema 4: "A complexidade de pior caso do *Smoothsort* **é O(n log n)".**
O pior caso ocorre quando o vetor começa inversamente ordenado e é essencialmente equivalente à criação de um Heap, acrescentando um elemento de cada vez. A complexidade desse processo é O(n log n).

Vários testes realizados comprovam a mudança "suave' da complexidade de O(n) para O(n log n), à medida que aumenta a desorganização inicial do vetor. Portanto, parece ser interessante a "rehabilitação" do *Smoothsort*, sobretudo para tentar buscar superar a grande dificuldade na sua implementação, talvez a principal razão do seu abandono.

Palavras-chave*: Algoritmos, Ordenação*
Referências

[1] Cormen, T. H.; Leiserson, C. E.; Rivest R. L.; Stein, C. (2001). "Introduction to Algorithms". Segunda edição, MIT Press/McGraw-Hill.

[2] Cowen, L.J.; Cowen, R.; Steinberg A. (2008) Totally Greedy Coin Sets and Greedy Obstructions. "The electronic Journal of Combinatorics" 15 (2008), #R90.

[3] Dijkstra, E. W. (1981) Smoothsort, an alternative for sorting in situ http://www.cs.utexas.edu/users/EWD/ewd07xx/EWD796.PDF, visitado em outubro de 2008.

[4] Pfitzner, A.L. (2009) "O Algoritmo de Ordenação Smoothsort e suas Aplicações", Monografia de Graduação, IME-DICC/UERJ, RJ, 2009.