Análise Empírica de Tempo de Execução de Algoritmo ShellSort e Comparação com HeapSort

Bruno Oliveira Souza Santos

Universidade Federal de Minas Gerais (UFMG)
Belo Horizonte - MG - Brasil

brunooss@ufmg.br

1 Introdução

O objetivo da prática é comparar o algoritmo de ordenação ShellSort, com valores para h e para sua atualização escolhidos previamente, com o algoritmo HeapSort, cuja complexidade é conhecida, sendo este da ordem de O(nlog(n)), a fim de comparar seus respectivos tempos de execução.

Inicialmente, é determinado um número n de iterações. Em cada iteração, são gerados 10 vetores com um tamanho específico m, que serão ordenados utilizando os algoritmos ShellSort e HeapSort. Em seguida, os tempos médios de execução de cada algoritmo são calculados e verificados, a fim de analisar a eficácia do valor h escolhido no algoritmo ShellSort em comparação com o algoritmo HeapSort.

Para a realização do teste em questão, é selecionado o valor inicial de h = 1 e, posteriormente, o h será multiplicado por 2 sucessivamente, para cada iteração. Tal escolha foi feita intuitivamente, a partir da observação dos métodos de ordenação por divisão e conquista, a exemplo do próprio HeapSort a ser considerado.

2 Análise de Complexidade

2.1 Complexidade de Tempo

A complexidade de tempo do algoritmo ShellSort é determinada pelo número de comparações e trocas realizadas durante o processo de ordenação. O algoritmo utiliza uma estratégia de divisão e conquista, onde o vetor é dividido em subvetores menores e ordenados individualmente, de modo que a quantidade de subvetores é diretamente relacionada à escolha de h. A complexidade de tempo do algoritmo ShellSort depende do valor de h escolhido e, por tal razão, será realizada uma verificação empírica do seu tempo de execução para tamanhos pré-determinados de vetores.

A complexidade de tempo do algoritmo HeapSort é determinada pelo número de comparações e trocas realizadas durante a construção do heap e a extração dos elementos de uma árvore binária completa, em que cada nó é maior ou igual aos seus nós filhos. Esse processo é repetido até que todos os elementos estejam em ordem. A complexidade de tempo do HeapSort é sempre $O(nlogn)$, onde n é o tamanho do vetor, independentemente da ordem inicial dos elementos. Dessa forma, este método de ordenação servirá como referência e como base de comparação para o ShellSort.

Tamanho do Vetor	Média Execução ShellSort	Média de Execução HeapSort
m = 100000	90.494 ms	35.324 ms
m = 200000	181.272 ms	78.302 ms
m = 300000	289.515 ms	113.43 ms
m = 400000	388.730 ms	159.658 ms
m = 500000	523.855 ms	204.882 ms
m = 600000	629.402 ms	244.638 ms
m = 700000	696.285 ms	295.232 ms
m = 800000	855.399 ms	333.552 ms
m = 900000	965.730 ms	383.001 ms
m = 1000000	1095.480 ms	425.740 ms
m = 1100000	1216.390 ms	474.086 ms
m = 1200000	1360.000 ms	517.730 ms
m = 1300000	1447.580 ms	566.780 ms
m = 1400000	1577.070 ms	619.620 ms
m = 1500000	1749.930 ms	675.586 ms

3 Resultados e Análises

A partir do código desenvolvido em C++ para implementar os algoritmos ShellSort e HeapSort, foram realizadas análises empíricas de tempo de execução, de forma que a média dos tempos de execução de cada algoritmo para cada tamanho de vetor estão disponibilizadas na tabela a seguir:

4 Conclusão

Com base nos resultados obtidos, é possível inferir que o algoritmo HeapSort apresenta um desempenho superior em relação ao ShellSort em termos de tempo de execução, considerando os tamanhos de vetor analisados. A complexidade de tempo do HeapSort, que é sempre O(n log n), independentemente da ordem inicial dos elementos, contribui para esse resultado favorável. Por outro lado, a complexidade de tempo do ShellSort, para h sendo multiplicado por 2 a cada iteração, tem seu valor aproximadamente multiplicado por 2, à medida que o vetor também tem seu tamanho dobrado.

Dessa forma, observa-se que o HeapSort é mais eficiente em termos de tempo, uma vez que o crescimento de seu tempo de execução ocorre de forma logarítmica