

Trabalho Prático: Método de Ordenação

Equipe: Henrique Augusto Rodrigues, João Vitor Gonzaga Jota, Filipe Arthur Ferreira Silva
Orientador: Sílvio Jamil Ferzoli Guimarães

Henrique Augusto Rodrigues*

2022/04/05

Resumo

A ordenação baseada em comparação é um problema clássico da computação e já bastante estudado, e para resolvê-la, há diversos algoritmos já propostos como seleção, inserção, quicksort, dentre outros.

Palavras-chaves: PAA, Projeto, Análise, Algoritmo, Ordenação.

Introdução

A ordenação baseada em comparação é um problema clássico da computação e já bastante estudado, e para resolvê-la, há diversos algoritmos já propostos como seleção, inserção, quicksort, dentre outros. Seja $X = x_1 \ x_2 \ \dots \ x_n$ um vetor de n elementos. Um algoritmo de ordenação aplicado a X irá rearranjar os elementos de forma a colocá-los em alguma ordem, como não-decrescente, por exemplo. Assim, uma forma de rearranjar X é baseada na análise de todas as permutações possíveis. Neste trabalho, espera-se que sejam implementados um método que encontre todas as permutações de X e três métodos clássicos, como Seleção, Quicksort e outra a escolha, e que sejam analisados os tempos, "teóricos" e "práticos", para estas ordenações. O tempo "teórico" pode ser medido em termos do número de comparações necessárias para a ordenação, já o tempo "prático" será medido em termos de tempo de execução. É importante destacar que a análise do tempo de execução envolve a execução dos métodos para várias instâncias (de tamanho diferentes). Os métodos de ordenação escolhidos pelo grupo foram, Quick Sort, Merge Sort e Insertion Sort.

*henrique.rodrigues@sga.pucminas.br

1 Problemas

Ordenação baseada em comparação.

1.1 Resolução

Na tentativa de resolver o problema, o grupo optou por três métodos de ordenação, quick sort, merge sort e insertion sort. Foram feitos cálculos téóricos e práticos na tentativa de, mostrar qual é o melhor método.

1.2 Resultados e Discussão

Discutiremos e mostraremos nesta seção, os resultados obtidos pelos autores. Como esperado, o quick sort obteve o melhor resultado dentre os três listados na seção anterior.

1.3 Quick Sort - Henrique Augusto Rodrigues

Pior Caso para o Quick Sort:

$T^*(n) = (n)$, é tão ruim quanto o algoritmo de inserção. Ilustração do pior caso, são feitas 28 comparações. [1 2 3 4 5 6 7 8]

[1 2 3 4 5 6 7] 8

[1 2 3 4 5 6] 7 8

[1] 2 3 4 5 6 7 8

1 2 3 4 5 6 7 8

Melhor Caso para o Quick Sort: (1) $T^*(n) = T^*(n/2) + T^*(n/2) + n$ (2) $S(n) = 2S(n/2) + n$ (2) Quando k atinge $\lg n$, temos $S(n) = nS(1) + n\lg n$. Supondo que $S(1) = 1$, teremos $S(n) = n + n\lg n$. Segue daí que $n\lg n \leq S(n) \leq 2n\lg n$ e portanto $S(n) = (n\lg n)$. Condição: $T^*(n) \leq n\lg n$

$T^*(n) = (n\lg n)$, O Teorema Mestre confirma que a solução da recorrência a cima está em $(n\lg n)$.

1.4 Merge Sort - Filipe Arthur Ferreira Silva

A contagem de operações efetuada pelo mergesort pode ser definida pela seguinte relação de recorrência: $T(n) = 1 + 2T(n/2) + 2 + f(n)$, $n > 1$, $n = 1$ //sendo $n = \text{fim} - \text{inicio}$ No caso, o primeiro valor constante 1 (no primeiro e segundo caso) se deve ao teste que é feito no if. O valor constante 2 no primeiro caso se deve ao cálculo do valor “meio” ($\text{inicio} + \text{fim}/2$). $2T(n/2)$ se refere às duas chamadas recursivas do método, cada chamada com uma metade do vetor. Por fim, $f(n)$ se refere à chamada do método intercalar, igual a: $f(n) = 2 + 1 + 3 + 3 + 2n/2 + 3n/2 + 2 + 2n = 9n/2 + 11$ Sendo, respectivamente, 2 operações para o cálculo de $n/2$, 1 operação para calcular $n/2$, 3 operações em cada malloc (+, * e a operação de alocação), $2n/2$ operações para inicializar o 1º subarray, $3n/2$ operações para inicializar o 2º subarray, 1 inserção da sentinela 0x7FFFFFFF em cada subarray e por fim $2n$ operações para remontar o vetor (sendo 2 = comparação entre $a1[i]$ e $a2[j]$ + inserção do menor valor no vetor). Inserindo a fórmula encontrada na relação de recorrência: $T(n) = 2T(n/2) + 9n/2 + 14$, $n > 1$, $n = 1$ Convertendo a relação de recorrência em um somatório: $i=0 \lg n - 1 (2i + 9n/2 + 14)$ Desenvolvendo o somatório, temos: $i=0 \lg n - 12i$

$+ i=0 \lg n - 19n/2 + i=0 \lg n - 114 = 2 \lg n - 1 - 1 \lg n - 2 + \lg n * 9n^2 + \lg n * 14$ O método, portanto, possui ordem de complexidade $O(n \log n)$.

1.5 Insertion Sort - João vitor Gonzaga Jota

Melhor Caso InsertionSort -(comparações) A comparações caso o vetor para cada elemento no vetor. Esteja ordenado, apenas uma comparação então:

$$\sum_{i=0}^{n-1} 1 = 1 * (n - 1 - 0 - 1) = \sum_{i=0}^{n-1} n \text{ Sendo a ordem de complexidade } O(n).$$

Pior Caso Insertionsort (comparações) $n - 1$ comparações caso o vetor esteja inversamente ordenado, $n - i$ comparações cada elemento no vetor. Então:

$$\sum_{i=0}^{n-1} n - 1 = (n - 1)((0 - 1 - 0 + 1))/2 = (n^2)/2 = O(n^2).$$

Conclusão

O presente trabalho teve como objetivo tentar resolver o problema de ordenação por comparação comparando o quick sort, merge sort e o insertion sort, comprovando que, o método de inserção quick sort é o melhor entre os três, como trabalho futuro, o grupo propõe que, faça a análise entre os demais métodos de ordenação.

Referências

Kleinberg, Jon. Tardos, Éva. Algorithm Design, Boston San Francisco New York, Matt Goldstein.