Introdução à Análise de Algoritmos
Exercício Programa 01 - Márcio Moretto Ribeiro

Mirela Mei - 11208392

17 de Novembro de 2021

1 Introdução

Algoritmo de Seleção
A ordenação por seleção consiste em selecionar em cada etapa o maior (ou o menor) elemento e alocá-lo em sua posição correta dentro da futura lista ordenada. Durante a execução, a lista com n registros é decomposta em duas sublistas, uma contendo os itens já ordenados e a outra com os elementos ainda não ordenados. No início, a sublista ordenada está vazia e a desordenada contém todos os elementos, no final do processo a sublista ordenada apresentará (n-1) elementos e a desordenada terá um elemento.
O algoritmo é composto por dois laços, um externo e outro interno. O laço externo serve para controlar o índice inicial e o interno percorre todo o vetor. Na primeira iteração do laço externo o índice começa de 0 e cada iteração ele soma uma unidade até o final do vetor e o laço mais interno percorre o vetor começando desse índice externo + 1 até o final do vetor.

Complexidade de tempo caso médio: $\Theta(n)$
Complexidade de tempo pior caso: $\Theta(n^2)$

Merge Sort
O merge sort é um exemplo de algoritmo de ordenação por comparação do tipo dividir-para-conquistar. Sua ideia básica consiste em dividir (o problema em vários subproblemas e resolver esses subproblemas através da recursividade) e conquistar (após todos os subproblemas terem sido resolvidos ocorre a conquista que é a união das resoluções dos subproblemas). Como o algoritmo Merge Sort usa a recursividade, há um alto consumo de memória e tempo de execução, tornando esta técnica não muito eficiente em alguns problemas.
O merge sort geralmente é usado para odenar listas ligadas. O acesso aleatório de uma lista ligada faz com que alguns outros algoritmos (como quicksort ou heapsort) tenham um desempenho pior ou até impossível. A sua eficiência é a mesma para melhor, pior e caso médio, independentemente de como os dados do array estão organizados a ordenação será eficaz.

$$\mathrm{T(n)}= \begin{cases} \Theta(1) & se\ n = 1 \\ 2T(n/2) + \Theta(n) & se\ n > 1 \end{cases}$$

Complexidade de tempo: $\Theta(n \log n)$

Complexidade de espaço: $\Theta(\log n)$

2 Objetivos

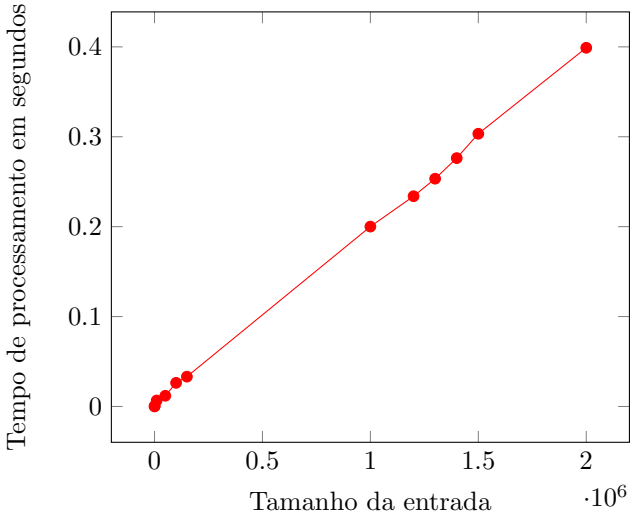
Ao comparar os tempos de processamento, é possível notar que, para o primeiro algoritmo (Merge Sort), os valores do tempo crescem em uma escala (n log n). A função merge é executada em um tempo O(n). O passo da divisão é executado em tempo constante, independente do tamanho do subarray. O passo da combinação intercala n elementos, levando também tempo 0(n). Em arrays de 100 elementos, os valores ficaram próximos de 0.0000551960 segundos, ao aumentar a escala para 100000 elementos, o tempo é de aproximadamente 0.0262811519 segundos, ocorrendo um aumento de 0,02 segundos. Já para valores na escala dos milhões, o tempo de execução é de aproximadamente 0,3 segundos.

Para o segundo algoritmo, é necessário considerar a probabilidade de dois elementos serem comparados durante o algoritmo para cada par de elementos, assumindo uma rotação aleatória. A partir disso, pode-se derivar o número médio de comparações. Ao assumir que o campo que se deseja selecionar o i-ésimo menor elemento seja uma permutação aleatória de [1,...,n]. Os elementos de pivô também podem ser vistos como uma dada permutação aleatória. Durante a execução, sempre se escolhe o próximo pivô possível dessa permutação, portanto eles são escolhidos uniformemente de maneira aleatória, pois cada elemento tem a mesma probabilidade de ocorrer que o próximo elemento possível na permutação. Compara-se apenas dois elementos i e j (com i menor j) Se e somente se um deles for escolhido como primeiro pivô elemento do intervalo [min(k,i), max(k,j)]. Se outro elemento desse intervalo for escolhido primeiro, eles nunca serão comparados. Todos os três casos precisam de um número linear de comparações esperadas. Isso mostra que o algoritmo possui um tempo de execução esperado em O(n) e o pior caso está em O(n2).
Ao observar os valores de teste, nota-se que o tempo começa em 0.0000551960 segundos para arrays de 100 elementos, aumentando o array para 100000 elementos, tem-se um aumento de aproximadamente 0,02 segundos. Ao escalar para 2000000 elementos, o tempo aproximado é de 0.3989968300 segundos.

3 Resultados

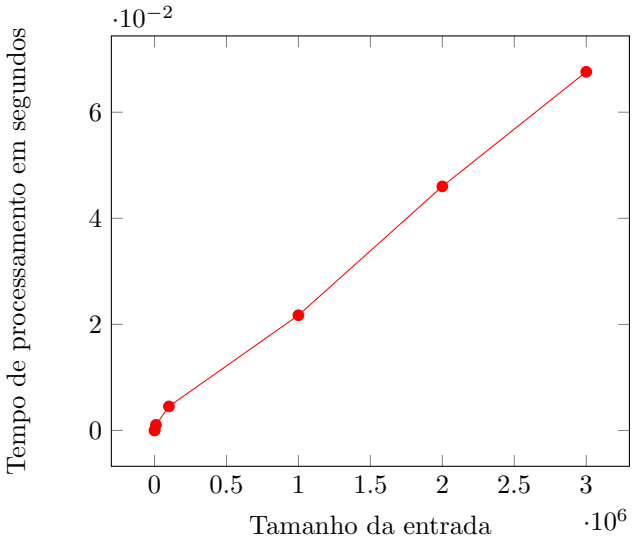
3.1 Primeiro algoritmo de seleção (Merge Sort)

Tamanho da entrada	Tempo de processamento (seg)
100	0.0000551960
1000	0.0006233950
10000	0.0042548059
100000	0.0262811519
1000000	0.2000931650
2000000	0.3989968300



3.2 Segundo algoritmo de seleção

Tamanho da entrada	Tempo de processamento (seg)
100	0.0000027680
1000	0.0000194230
10000	0.0010555180
100000	0.0045349621
1000000	0.0217231568
2000000	0.0460131876
3000000	0.0676212140



4 Conclusão

Ao comparar os dois algoritmos, é evidente que o segundo obteve uma eficiência quase 100 vezes melhor em relação ao tempo de processamento. A partir disso, comprova-se a análise de complexidade dos dois, estando visível também nos gráficos e tabelas acima.