Data: Goiânia, 30 de agosto de 2022

Faculdade: Senai Fatesg / Professor: Luís Mário

Aluno: Wenderson

Aluno: Witor Sather Santos de Queiroz

Aluno: Erlan Dias

Relatório Descritivo Trabalho de Busca e Ordenação - Dicionários

Relatório descritivo sobre a execução e observações realizadas durante o desenvolvimento do trabalho acerca de cada algoritmo desenvolvido, explicitando o funcionamento de cada algoritmo bem como a realização de uma análise sobre quais algoritmos se comportaram melhor na resolução do problema.

Algoritmos de ordenação usados:

- SelectionSort
- BubbleSort
- InsertionSort
- MergeSort
- QuickSort

SelectionSort - Ordenação por Seleção

O SelectionSort vasculha repetidamente a lista de itens, selecionando um elemento de cada vez e colocando-o na posição correta da sequência. A principal vantagem do SelectionSort é que ela funciona bem em uma lista pequena. Além disso, por ser um algoritmo de ordenação de local, não precisa de armazenamento temporário além do necessário para guardar a lista original. A principal desvantagem é sua baixa eficiência em listas grandes. Assim como o BubbleSort, ele exige n^2 números de passos para cada n elementos. Adicionalmente, o seu desempenho é facilmente influenciado pela ordem inicial dos itens antes do processo de triagem. Devido a isso, esse tipo seleção é adequado apenas para uma lista em que poucos elementos estejam em ordem aleatória.

BubbleSort – Ordenação por Bolha

O BubbleSort troca repetidamente elementos adjacentes que não estão em ordem até que toda lista de itens esteja em sequência. Dessa maneira, os itens flutuam na lista conforme os seus valores, indo do maior (no caso de ordenação crescente) pro final ao fim de cada iteração.

A principal vantagem desse algoritmo é que sua implementação é fácil e conhecida. Além disso, no BubbleSort, os elementos são trocados de lugar sem utilizar

armazenamento temporário, o que faz o requerimento de espaço ser mínimo. A principal desvantagem é o fato de que não apresenta bons resultados quando a lista contém muitos itens. Isso porque esse tipo de ordenação exige n^2 passos de processamento para cada número n de elementos que serão ordenados. Portanto, o BubbleSort indicada para o ensino acadêmico, mas não para aplicações na vida real.

InsertionSort – Ordenação por Inserção

O InsertionSort varre a lista repetidamente e, a cada vez, insere um item da sequência desordenada na posição correta.

A principal vantagem da ordenação por inserção é a sua simplicidade, além de mostrar um bom desempenho em listas pequenas. É um algoritmo de ordenação de local, logo, o requerimento de espaço é mínimo. A desvantagem é que não possui um desempenho tão bom quanto outros algoritmos de ordenação. Com n^2 passos necessários para funcionar, o InsertionSort também não funciona bem com listas grandes. No entanto, é particularmente útil com listas de poucos itens.

MergeSort – Ordenação por Intercalação

O MergeSort realiza a ordenação dividindo uma sequência de elementos em sequencias menores recursivamente, que serão ordenadas e depois serão combinados de forma ordenada.

As principais vantagens deste algoritmo são a facilidade na implementação e a sua complexidade (O(n log2 n)). A principal desvantagem é a utilização de funções recursivas que acarretam em um maior uso de memória.

QuickSort – Ordenação por Troca de Partição

O QuickSort trabalha com o princípio da divisão-e-conquista. Primeiramente, ela divide a lista de itens em duas sub-listas com base em um elemento pivô. Todos os elementos na primeira sub-lista são dispostos de maneira que sejam menores do que o pivô, enquanto que todos os elementos na segunda sub-lista são dispostos para serem maiores que o pivô. O mesmo processo de particionamento e organização é executado repetidamente nas sub-listas resultantes até que toda a lista seja organizada.

O QuickSort é considerado por alguns o melhor algoritmo de ordenação por causa da sua vantagem significante em termos de eficiência, uma vez que funciona bem com uma lista grande de itens. Por ordenar no próprio local, também não há necessidade de espaço adicional de armazenamento. A leve desvantagem que ela apresenta é que seu pior desempenho é similar à média de desempenho dos outros algoritmos descritos acima. Entretanto, é importante notar que esse pior caso é muito raro. De um modo mais geral, o QuickSort produz o método de organização mais eficiente e amplamente utilizado para organizar uma lista de qualquer tamanho.

Tempo de Execução Nano Segundos

Máquina Witor Sather

Processador: Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.19 GHz

Memória RAM: 20.0 GB

Tipo de Sistema: 64 bits