Universidade de São Paulo
Instituto de Matemática e Estatística
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação
MAC110 - Introdução à Computação

Análise de diferentes algoritmos de ordenação

Aluno(a): João Viktor Souza Almeida

Professor: Roberto Hirata Junior

Resumo

O relatório a seguir visa analisar "alguns"algoritmos de ordenação e como suas eficiências mudam dadas diferentes condições iniciais, tais como porcentagem de ordenação e número de elementos da lista a ser ordenada.

Metodologia

Na criação do relatório, os testes dos algoritmos foram realizados utilizando a linguagem de programação Python. Durante a execução dos algoritmos, foi utilizada uma máquina com as seguintes configurações:

- Processador (CPU): Ryzen 5 3350G
- Memória (RAM): 16GB DDR4 @ 3200MHz
- Armazenamento: 256GB SSD
- Placa de Vídeo (GPU): Radeon Vega 11
- Sistema Operacional: Windows 11 Pro

Além disso, durante a execução dos algoritmos, foram desabilitados o máximo de programas possíveis a fim de minimizar as interferências nos resutlados dos testes.

Para computar a média de execução, foi invocada uma função *timeMe*, na qual eram armazenados os resultados das 10 itinerações do mesmo algoritmo e, no final, retornava a média e o desvio padrão dos resultados.

Para fins simplificatórios, foram ignorados possíveis margens de erros da função que retorna a lista embaralhada. Contudo, essa margem de erro torna-se inotável durante os testes devido ao tamanho das listas utilizadas.

Palavras-chave: insertion, bubble, counting, selection, algoritmos, ordenação, análise;

Conteúdo

1 Algoritmo de seleção

O algoritmo de Seleção é um algoritmo de ordenação no qual, a cada itineração, o menor elemento da lista é garantido estar na posição correta. Ou seja, na primeira itineração, assegura-se que o menor elemento ficará na primeira posição da lista, na segunda itineração, o segundo elemento, e assim por diante.

Acerca do seu desempenho, o algoritmo em questão possui um "desempenho"quadrático, isto é, para n elementops da lista, o algoritmo realizará $\frac{n(n-1)}{2}$ comparações a fim de ordernar totalmente a lista. Uma observação pertinente é que a quantidade de comparações a ser feita independe da porcentagem de ordenação da lista.

2 Algoritmo de bolha

O algoritmo de bolha é um algoritmo cuja complexidade é, assim como o anterior, quadrática. Sua principal característica é que, na n-ésima passagem pela lista, o algoritmo assegura estarem ordenados os n últimos elementos da lista.

Exemplo: Seja M uma lista [4,3,2,1,0]. Na primeira passagem, o maior valor da lista (4) estará em seu lugar correto, ou seja, M será [3,2,1,0,4]. Na segunda passagem, o segundo maior elemento também estará posicionado em sua posição correta; assim, M será [2,1,0,3,4].

O algoritmo continuará realizando "isto"até que, na 5^{o} passagem, (pois a lista possui 5 elementos), a lista estará totalmente ordenada. Portanto, assim como no algoritmo anterior, o Bolha é eficiente apenas para listas com tamanhos pequenos.

3 Algoritmo de inserção

O algoritmo de inserção é um cuja caracteristica principal é a asseguração da ordenação da lista a cada O algoritmo de inserção, ao contrário dos outros já comentados, possui uma complexidade que varia dependendo do quão ordenada a lista já se encontra.

4 Algoritmo de contagem

Diferentemente dos algoritmos outrora apresentados, o algoritmo em tópico possui uma complexidade linear, isto é, o tempo levado para ordernar uma lista, ao contrário dos apresentados, não cresce de maneira quadrática.

Em relação ao uso de memória, contudo, há uma grande desvantagem no algoritmo, uma vez que a lista auxiliar utilizada possuirá $\max lista - \min lista$ elementos; ou seja, caso o maior elemento seja 5000 e o menor 1000, a lista auxiliar terá um tamanho de 4000 elementos.

Dessa forma, o algoritmo possui um melhor proveito se utilizado para listas com pouca variação de tamanho entre os seus elementos.

Conclusões

Os algoritmos em questão, apesar de possuírem diferentes desempenhos dependendo da ordenação da lista, possuem um baixo proveito com listas muito grande. Além disso, é notável que o método importado do Python contém diversos algoritmos, que são usados dependendo do contexto. Portanto, a fim de ordenar uma lista com maior rapidez, é notável que a função built-in do Python deve, pelo menos para listas grandes, ser usada em relação aos outros métodos outrora discutidos.