Relatório

Trabalho Pratico Ordenação

RA: 318211348

Nome: Samuel Fellipe Penna

Introdução

- A ideia deste trabalho é como fazer os métodos de ordenação para que se tenha um entendimento dos processos e de qual método é o mais viável para tal projeto.

Implementação

- Na implementação foi utilizada a estrutura de vetores para a criação das ordens crescente,decrescente e randômica.
- Para estabelecer o randômico foi utilizado o “random” do net framework (que após utilizar ele gera um número aleatório entre o número mínimo e máximo que o usuário colocar).
- Para verificar o tempo de ordenação dos vetores foi utilizado o comando Stopwatch (que inicia uma contagem de tempo

que o comando leva para ser executado) e o TimeSpan (que anota o tempo que o stopwatch contou). Foram criadas as classes que retornam os métodos de ordenação.

Analise

- O programa retorna o tempo gasto para ordenar os 3 vetores criados por 5 métodos diferentes (Selection, Insertion, Bubble, Merge e Quick Sort).
- Abaixo na Tabela 1, estão os tempos de cada método para ordenar um vetor de 10.000 posições. E em seguida um gráfico (Gráfico 1) para posteriores análises.

Tabela 1 – Dados extraídos do programa

Método	Vetor	Tempo (ms)
Merge	Vetor A Vetor B Vetor C	251209 189979 185245
Bubble	Vetor A Vetor B Vetor C	3424855 3321669 3144001
Insertion	Vetor A	3397

		Vetor B Vetor C	688 683
	Quick	Vetor A Vetor B Vetor C	915827 1048459 977586
	Selection	Vetor A Vetor B Vetor C	1593642 1813689 1813689

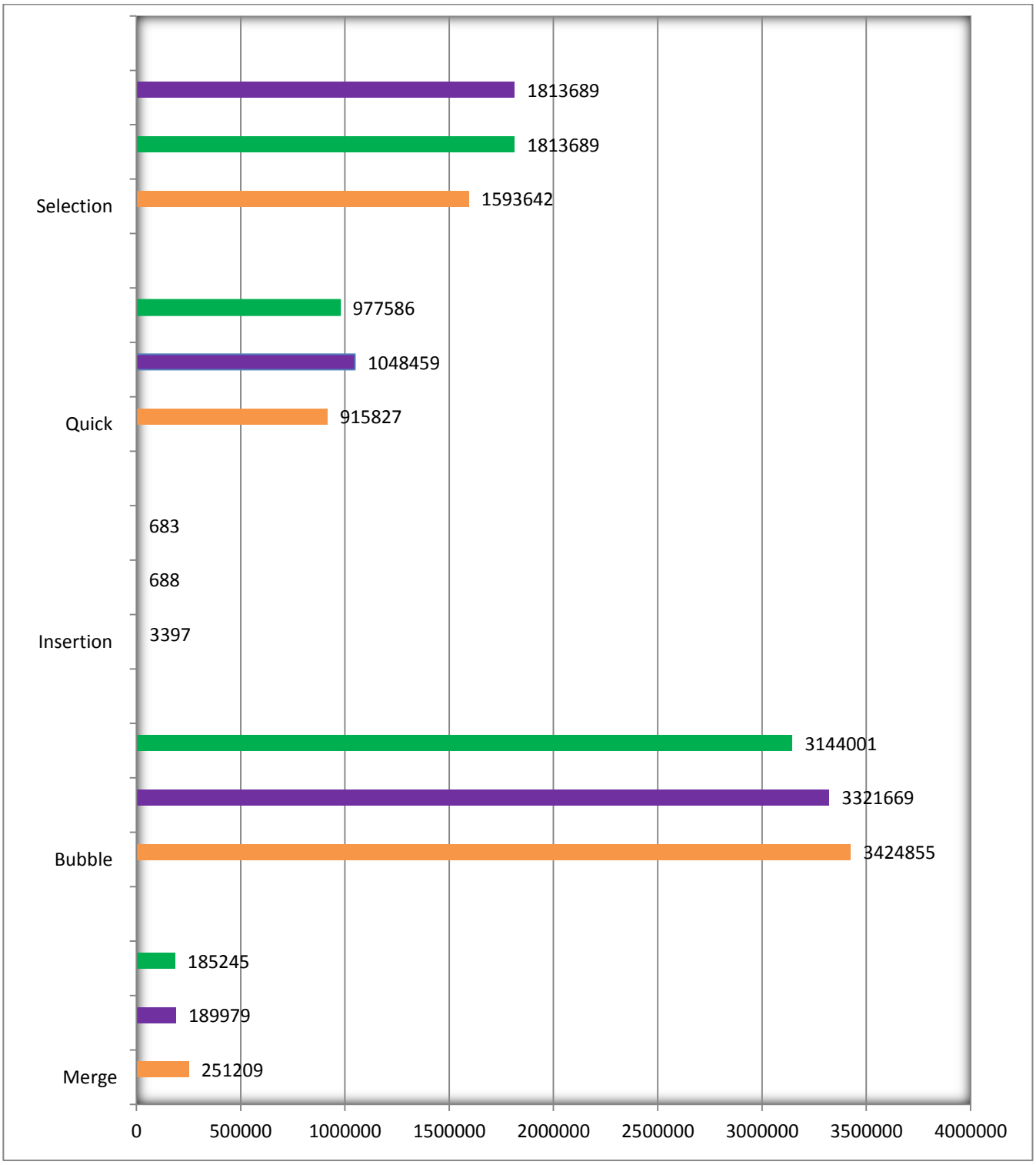

Gráfico 1 – Comparativo entre os métodos

Comparando o Vetor A(vetor crescente) nos 5 métodos comparados, o Insertion Sort foi o que se apresentou mais eficiente

Como observado o método mais eficiente no contexto demonstrado é o Insertion Sort. . Esse resultado se deve ao fato do Insertion Sort comparar os elementos de um vetor de forma global, identificando rapidamente que o vetor já está ordenado como deveria, não realizando permutações.

Após o Insertion os métodos mais eficientes para ordenar os vetores propostos foram os métodos eficientes, Merge e Quick Sort, respectivamente. Como abordado na introdução são bons métodos em cenários de ordenação de valores alto.

O Merge Sort, como podemos observar, devido a recursividade ser sua principal ferramenta, seu melhor resultado vem ao lidar com estruturas lineares aleatórias.
O algoritmo Quick Sort, ao subdividir o vetor e fazer inserções diretas utilizando um valor de referência (pivô), reduz seu tempo de execução, mas, as quantidades de comparações (leitura) e, principalmente, trocas (escrita) ainda são muito altas.

O método Bubble Sort apresenta o pior desempenho, resultando em altos valores de comparações e trocas.

Conclusão

O trabalho foi interessante para aprender diferentes maneiras de ordenação existentes.
A maior dificuldade foi no merge sort pois mesmo implementando diferentes tipos de métodos dele eu não consegui entender como o código funcionava e o motivo qual ele não estava ordenando
meus vetores.
Como resultado final foi visto que o quick sort é bem mais rápido do que os outros métodos de ordenação.