Na 2ª unidade estudaremos os algoritmos de Ordenação.
O exercício é em DUPLA.

A cada aula será passado um ou mais dos seguintes algoritmos para que vocês implementem e façam a medição do tempo de execução do mesmo.

1. **Insertion Sort**
2. **Selection Sort**
3. **Bubble Sort**
4. **HeapSort**
5. MergeSort
6. QuickSort

As implementações devem ser feitas por vocês, cópias serão zeradas.

A cada algoritmo o código deve ser INCREMENTADO com a função equivalente ao mesmo.

O programa quando iniciar a execução deve mostrar ao usuário os algoritmos disponíveis e perguntar o número do algoritmo que o usuário deseja executar.

Em seguida deve perguntar qual arquivo dentre os SETE arquivos de números (que serão explicados em seguida) o usuário deseja ordenar.

Após a escolha do usuário o algoritmo de ordenação deve ser executado e ao final da execução deve ser mostrado o tempo.

Para cada algoritmo vocês devem medir através do código quanto tempo ele levou para ordenar um conjunto de números.

Vocês receberam 7 arquivos de números que devem ser lidos, ordenados e registrados o tempo de ordenação para cada um dos arquivos (os arquivos encontram-se neste link)

1. Arquivo com MIL números desordenados
2. Arquivo com CINCO MIL números desordenados
3. Arquivo com DEZ MIL números desordenados
4. Arquivo com VINTE MIL números desordenados
5. Arquivo com CINQUENTA MIL números desordenados

6. Arquivo com SETENTA E CINCO MIL números desordenados
7. Arquivo com CEM MIL números desordenados

O tempo de execução NÃO pode incluir o tempo de leitura, ele deve marcar APENAS o tempo de ORDENAÇÃO.

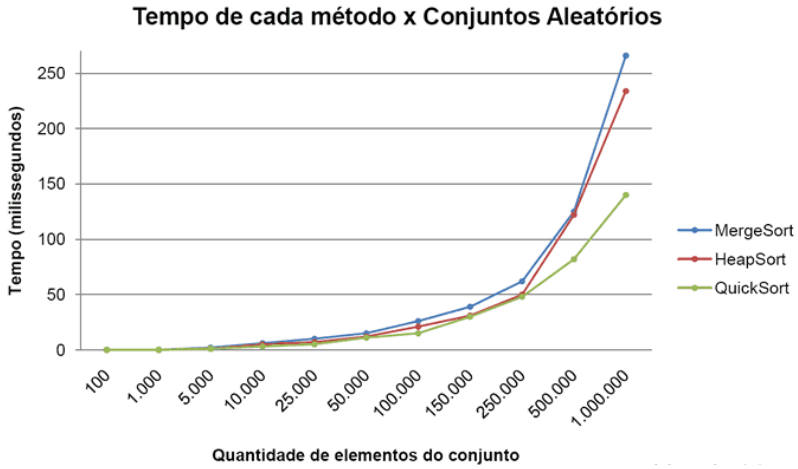
Os tempos precisam ser medidos utilizando o mesmo computador de preferência sem nenhuma outra coisa aberta para que a comparação seja mais próxima da realidade.

Vocês devem preencher a seguinte tabela e gerar gráficos comparativos para os algoritmos:

1. Gráfico do tamanho da entrada x o Tempo para cada Algoritmo
2. Gráfico que mostre a cada "rodada" os tempos pela quantidade (arquivos) os de todos algoritmos implementados

	1000	5000	10000	20000	50000	75000	100000
Insertion Sort							
Selection Sort							
Bubble Sort							
HeapSort							
MergeSort							
QuickSort							

Exemplo de gráfico:



Para cada algoritmo implementado CADA membro da dupla preparar um documento (de texto) explicando:

1. A complexidade do algoritmo,
    a. qual o melhor caso?
    b. qual o pior caso?
    c. qual o caso médio?
2. Comentar sobre o resultado dos gráficos/tabelas e a percepção de vocês sobre os tempos em comparação com os outros algoritmos

Todos os arquivos devem ser entregues no GOOGLE CLASSROOM nas datas pré-definidas.

Caso tenha atraso na entrega o exercício valerá apenas 50% das notas.

Caso tenha erro no formato e/ou nome dos arquivos, os alunos perderão 25% das notas.

As implementações devem ser em C.

As duplas que quiserem também podem implementar em HASKELL valendo uma bonificação de 10% para cada algoritmo. No caso de implementar em HASKELL e em C cada membro apresenta um dos códigos.

No caso de implementar os algoritmos em Haskell vocês devem medir o tempo de execução em Haskell e em C e incluir os comentários no documento.

Além de criar as tabelas e gráficos com as informações das execuções em Haskell e em C

A tabela deve ser preenchida a cada novo algoritmo e deve ser entregue junto com o código e os vídeos.

Os DOIS membros devem entregar os exercícios no GOOGLE CLASSROOM

TODOS os arquivos devem seguir o seguinte formato de nome:

Supondo que os nomes da dupla são:

- Jõao Silva Santos

- Maria Eduarda Santana

O nome do arquivo deveria ser:
jss_mes.formato_do_arquivo

Arquivos fora do padrão serão desconsiderados.

No final cada aluno deve entregar um ZIP (jss_mes.zip) com os seguintes arquivos (os nomes dos arquivos seguem o exemplo da entrega de Jõao):

- jss_mes_sort.c
- jss_mes_tabela_c.jpg
    - Contendo TODOS os algoritmos
- jss_mes_algoritmos_grafico_c.jpg
    - Contendo TODOS os algoritmos
- jss_sort.doc

Caso também tenha sido implementado o algoritmo em Haskell o arquivo também deve conter os arquivos:

- jss_mes_sort.hs
- jss_mes_sort_tabela_haskell.jpg
    - Contendo TODOS os algoritmos
- jss_mes_algoritmos_grafico_haskell.jpg
    - Contendo TODOS os algoritmos