Universidade Federal de Minas Gerais
Instituto de Ciências Exatas
Departamento de Ciências da Computação

Professores: Luiz Chaimowicz e Raquel Prates
Monitores: Abner S. Nascimento e Alysson R. da Silva
Estruturas de Dados 2020.2

Belo Horizonte, 12 de Fevereiro de 2021

Trabalho Prático II: Retornando para casa

Trabalho Individual. Valor: 10 pontos **Entrega: 4 de Março de 2021**

Objetivo: O objetivo deste trabalho é consolidar conhecimento e colocar em prática os métodos de ordenação vistos nas aulas. Para isso, deverão implementar alguns deles e fazer uma análise comparativa, discutindo os resultados obtidos. Observem que para este trabalho, **a análise dos algoritmos terá o mesmo peso do que sua implementação em termos de nota**.

1 Introdução

A primeira fase da missão Extração-Z foi um sucesso! Com a quantidade de recursos coletados, será possível construir tecnologias de ponta para purificação de mananciais, reflorestamento, coleta de fontes de energia renováveis, dentre muitas outras aplicações. Tem início uma nova era de desenvolvimento para a humanidade, impulsionada pelos seus esforços junto ao Instituto de Ciências Exoplanetárias. Todas as bases do programa poderão finalmente ser recolhidas, terão seus robôs catalogados e estarão prontas para dar início à viagem de retorno.

Entretanto, se levar os robôs inicialmente já foi um desafio, trazê-los de volta será um processo ainda mais delicado. São centenas de milhares de bases, espalhadas por sistemas planetários não só da via láctea, mas também de outras galáxias do aglomerado. Além disso, como as bases e naves que carregam os recursos estão mais pesadas, agora mais energia será gasta para transportá-las pelo espaço.

Diante dessas dificuldades, o Instituto de Ciências Exoplanetárias concluiu que a melhor forma de organizar a logística de retorno é priorizar as bases mais distantes, isto é, que requerem **mais** saltos hiperespaciais para serem alcançadas. As equipes de engenharia do instituto têm preferências por determinados algoritmos de ordenação e não conseguiram entrar ainda em um consenso a respeito de qual escolher para organizar a grande quantidade de informações na ordem correta de prioridade. Você deve implementar os diferentes algoritmos de ordenação apresentados pelas equipes, analisar as características de cada um, vantagens e desvantagens. Sua análise será valiosa para que a decisão final seja tomada, então, boa sorte!

2 Especificações

Você trabalhará com dados que contém o nome da base e a distância em saltos hiperespaciais necessários para alcançar cada uma. Essas informações estão dispostas em um arquivo com **200 mil linhas**, na forma: '`nome distância`'. Cada uma das equipes de engenharia propôs uma técnica diferente de ordenação que você deve implementar. São elas:

1. Insertion sort (i.e. método de inserção);
2. Merge sort **ou** heap sort **(não é um requisito implementar ambos e não receberão ponto extra caso optem por isso)**;
3. Quick sort.

Além dos algoritmos sugeridos pelas equipes, você também deve implementar modificações em algumas das técnicas originais, bem como pesquisar e implementar um método alternativo de ordenação, conforme os critérios abaixo:

1. Quick sort modificado: procure desenvolver uma versão levemente diferente em relação à do ponto anterior. Você pode escolher uma das melhorias apresentadas nas aulas, ou pesquisar uma nova, como uma versão sem chamadas recursivas, ou com estratégia diferente para a escolha do pivô. **Na documentação, você deve explicar o que essa melhoria faz no Quick sort e também a razão de sua escolha.**

2. Um algoritmo de ordenação de sua escolha, **que não tenha sido trabalhado nas aulas**, com exceção de métodos triviais de força-bruta, ou aleatórios, como *Bogus Sort*. **Na documentação, você deverá explicar o algoritmo, o motivo de sua escolha, e a sua análise.**

2.1 Análise e Comparação

Cada algoritmo implementado deverá ser testado com o conteúdo de três arquivos de entrada. Um dos arquivos de entrada contém o conteúdo ordenado em ordem crescente, outro contém o conteúdo ordenado em ordem decrescente, e o último contém o conteúdo ordenado aleatoriamente. Todos os arquivo de entrada contém 200.000 linhas. Para cada um deles, você deve testar os algoritmos variando o tamanho **N** da entrada. O valor de **N** é a quantidade de linhas a serem lidas. Ele deve seguir a sequência de valores abaixo:

- 100, 500, 1.000, 5.000, 10.000, 50.000, 100.000 e 200.000.

Ou seja, você deve testar cada algoritmo implementado lendo inicialmente os 100 primeiros elementos; depois, em um outro teste, os 500 primeiros elementos (incluindo os anteriores também); depois, em mais um teste os 1.000 primeiros elementos; assim por diante, até que se alcance $N = 200.000$. **Deve-se fazer esses testes para cada um dos três arquivos para cada um dos algoritmos implementados.**

Você deve **obrigatoriamente** criar três gráficos de barras, um para cada arquivo de entrada. Ou seja, um gráfico será utilizado para expressar os resultados obtidos para o arquivo que representa a ordem **crescente**, outro para a ordem **decrescente**, e outro para o caso onde os elementos estão ordenados em ordem aleatória. Em sua análise, procure identificar quais casos favorecem ou pioram a performance de cada algoritmo. O gráfico deve apresentar barras comparativas para todos os tamanhos de entrada testados. A Figura 1 pode ser utilizada como referência.

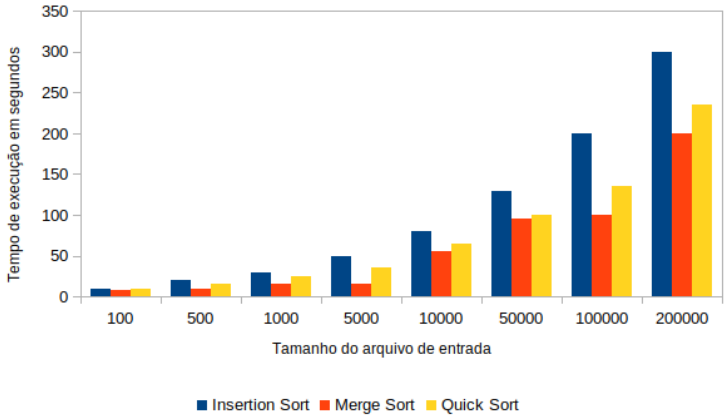

Figura 1: Exemplo de gráfico utilizado para representar os resultados obtidos de um dos três arquivos de entrada. Você deve produzir um gráfico como este para cada um dos 3 arquivos.

Além dos três gráficos obrigatórios, você pode fazer quaisquer outros gráficos e análises que julgar relevantes e que contribuam para o seu trabalho.

Por fim, você deve explicar os resultados obtidos levando em consideração a **complexidade assintótica** dos algoritmos; as **vantagens** e **desvantagens** de cada um; se são ou não **estáveis**; e como se comportam em relação à entrada, *i.e.*, como a ordem dos elementos influencia na quantidade de operações efetuadas.

3 Entrada e Saída

Apesar de ser necessário avaliar seu programa para três arquivos de entrada diferentes, você **DEVE** programar seu sistema para ler apenas um deles a cada execução. Dessa forma você pode modularizar melhor os seus testes e execuções.

Seu programa deve, pois, receber como entrada o nome de um arquivo e a quantidade N de linhas a serem consideradas. Na Figura 1a, há um exemplo das informações das bases: seus nomes e distâncias, respectivamente, distribuídos ao longo das linhas e separados por espaços. A saída esperada consiste nas 7 primeiras linhas ordenadas conforme **a distância**, **da mais afastada (maior), à mais próxima (menor)**. Note que, para algoritmos **não-estáveis**, a saída pode ser diferente em caso de empate entre dois elementos.

```
Chuborilia 11
Rochoth 203
Thanyria 45
Metur 21
Gneimia 203
Brosie 11
Thorix 9
```

(a) Primeiras linhas da entrada

```
Rochoth 203
Gneimia 203
Thanyria 45
Metur 21
Brosie 11
Chuborilia 11
Thorix 9
```

(b) Saída esperada

Figura 1: Exemplos para o arquivo de entrada e a saída esperada, considerando $N = 7$.

A documentação do trabalho deve conter os requisitos descritos na Subseção 2.1. Você pode usar o método que preferir para medição do tempo de execução de cada algoritmo, seja ele via código[1], ou utilitários de linha de comando[2]. Será fornecida uma entrada de exemplo, mas seu trabalho poderá ser testado com arquivos diferentes.

4 Entregáveis

Você deve utilizar a linguagem C ou C++ para a implementação dos algoritmos. Como no trabalho anterior, métodos das bibliotecas-padrão podem ser usadas livremente para operações externas aos algoritmos principais, como por exemplo o uso de ifstream e strings para leitura de arquivo. Contudo, o uso de algoritmos e eventuais estruturas auxiliares, como *heaps*, pré-implementados pelas bibliotecas-padrão da linguagem ou terceiros é **terminantemente vetado**. Você pode reaproveitar as estruturas implementadas no trabalho anterior e está proibido de utilizar estruturas disponíveis como *vector*, *list*, e *queue*. Com exceção do tipo *string*, utilize apenas as variáveis de tipos primitivos e derivados para criar **suas próprias implementações** para todas as classes, estruturas, e algoritmos.

Organize seu código-fonte em arquivos separados conforme a responsabilidade de cada objeto e entidade, com nomenclatura condizentes ao que executam. A fim de padronizar a compilação e o

[1]Sugestão de como medir tempo em C:
http://wurthmann.blogspot.com/2015/04/medir-tempo-de-execucao-em-c.html

[2]Sugestões de ferramentas para medição de tempo na linha de comando
Linux/Mac OS: https://www.hostinger.com.br/tutoriais/comando-time-linux
Windows: http://howtoserver.com.br/tempo-de-execucao-powershell/

projeto, você **DEVE utilizar** a estrutura de projeto abaixo e o '`Makefile`' disponível no *Moodle*:

```
- TP2
    |- src
    |- obj
    |- bin
    |- include
    Makefile
```

A pasta '`TP1`' é a raiz do projeto; a pasta '`bin`' deve conter os executáveis gerados após a compilação; '`src`' deve armazenar arquivos de código ('`*.c`', '`*.cpp`', ou '`*.cc`'); e '`include`', os cabeçalhos (*headers*) do projeto, com extensão '`*.h`' ou '`*.hpp`'. O executável do seu programa deverá receber como parâmetro o nome do arquivo de entrada e o valor de N, isto é, a quantidade de bases que serão consideradas, conforme o exemplo a seguir:

```
run.out dados.txt 1000
```

No qual, '`dados.txt`' é o primeiro argumento recebido pelo seu programa (armazenado dentro de '`argv[1]`') e 1000 é o segundo (armazenado em '`argv[2]`').

Procure seguir boas práticas de programação: cada bloco deve ser indentado apropriadamente; utilize nomes descritivos para variáveis; e, se julgar necessário, deixe breves comentários em pontos relevantes do código. Evite comentários triviais, como:

```
x = 2 // atribui o valor 2 a x
```

4.1 Documentação

A documentação do trabalho deve ser entregue em formato **pdf** e também deverá seguir o modelo que está postado no Moodle. Ele deve conter **todos** os itens descritos abaixo. Procure escrever de forma sucinta e objetiva.

- Título, nome, e matrícula.
- Introdução com apresentação geral dos algoritmos implementados.
- Instruções para compilação e execução (caso estas demandem passos adicionais).
- Sobre os algoritmos vistos em sala, comente brevemente sua implementação, bem como a complexidade, estabilidade e comportamento em relação à entrada. Eventuais casos especiais que possam não ter sido previstos na descrição do trabalho e a maneira como foram tratados também devem ser explicados, assim como todas as decisões tomadas ao longo do desenvolvimento.
- Em relação ao **quicksort modificado** explique que melhoria fez no algoritmo, como a implementou, se gera algum impacto na complexidade dele e, em caso afirmativo, qual. Além disso, explique por que escolheu esta melhoria em particular.
- Em relação ao **novo algoritmo**, explique como ele funciona (não é necessário adicionar o código ao texto, basta explicar a ideia geral); decisões relacionadas à implementação; apresente sua complexidade; se é estável ou não; se é adaptável ou não (*i.e.*, impactado pela ordem de entrada dos elementos); suas vantagens e desvantagens. Além disso, explique por que escolheu esse algoritmo em particular.
- Comparações entre os algoritmos e seus tempos de execução, apresentadas com gráficos e tabelas.
- Conclusões, observações e considerações finais.
- Bibliografia com as fontes consultadas para realização do trabalho.

4.2 Submissão

Todos os arquivos relacionados ao trabalho devem ser submetidos na atividade designada para tal no *Moodle* dentro do prazo estipulado. A entrega deve ser feita **em um único arquivo**, com nomenclatura '`nome_sobrenome_matricula.zip`'. Este deve conter um arquivo '`pdf`', para a documentação, e uma pasta '`projeto`', contendo tanto o código-fonte e um '`Makefile`' para compilação.

5 Considerações Finais

1. Leia **atentamente** o documento de especificação, pois o descumprimento de quaisquer requisitos obrigatórios aqui descritos causará penalizações na nota final.

2. Os principais aspectos avaliados na correção serão:

   - Corretude na ordenação das saídas;
   - Conteúdo da documentação e estudos de complexidade;
   - Apresentação dos dados e corretude da análise;
   - Boas práticas de implementação e organização do código.

3. Certifique-se de garantir que seu arquivo foi submetido corretamente no sistema.

4. Em caso de dúvida, não hesite em perguntar nos fóruns de discussão da disciplina ou aos monitores. Encorajamos a participação através do fórum para estimular a comunicação da turma no regime de ensino remoto emergencial.

5. **Plágio é CRIME**. Trabalho onde o plágio for identificado serão **automaticamente anulados** e as medidas administrativas cabíveis serão tomadas. Discussões a respeito do trabalho entre colegas são permitidas. É permitido consultar fontes externas, desde que exclusivamente para fins didáticos e devidamente registradas na sessão de bibliografia da documentação. **Cópia e compartilhamento de código não são permitidos**.

6. Procure começar o trabalho com antecedência, pois atrasos sofrerão penalização de $2^d - 1$ pontos, com $d =$ dias de atraso.

Bom trabalho!