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

Estruturas de Dados 2020/1
Prof. Luiz Chaimowicz, Raquel O. Prates
Monitores: Álisson R. Svaigen e Michele B. Pinheiro

Trabalho Prático 2: A ordenação da estratégia de dominação do imperador

Valor: 10 pontos
Data de entrega: 15 de outubro de 2020

Introdução

Devido ao seu ótimo desempenho ao desenvolver o sistema para gerenciar as naves do imperador Vader, você ganhou a confiança dele! O imperador destituiu o general Falura do posto, o entregando a você, caro(a) aluno(a)! Parabéns, você é o novo braço direito de Vader! Tendo como aliado alguém de tamanha destreza e inteligência, o imperador deseja expandir os seus domínios, conquistando as civilizações intergalácticas que ainda resistem ao Novo Império. Como se sabe, o universo é muito vasto e, dessa maneira, há um número muito grande de civilizações conhecidas para se dominar.

Constantemente, o imperador recebe informações de seus aliados de novas civilizações para conquistar. No entanto, esses dados estão todos desorganizados. Para organizar seu plano de dominação, Vader recorre novamente às suas habilidades de resolução de problemas! Ele deseja que você estabeleça "ordem no caos", traçando um plano de dominação baseado em dois aspectos: distância da civilização com relação ao planeta sede do Novo Império; e tamanho da população da civilização.

Devido aos conhecimentos obtidos na disciplina de Estrutura de Dados, você logo identifica que essa tarefa se trata de um problema de ordenação e, consequentemente, o método de ordenação utilizado pode influenciar diretamente no tempo que o algoritmo leva para ordenar um conjunto de dados suficientemente grande. E eis que, então, o grande *plot twist* é revelado: você, caro(a) aluno(a), é um agente infiltrado da Aliança Rebelde! Finalmente, você terá a oportunidade de enganar o imperador Vader e ajudar a Aliança!

Para isso, você irá implementar métodos de ordenação com complexidade assintótica maior (ou seja, "mais lento") para o imperador, enquanto métodos de ordenação com complexidade assintótica menor (ou seja, "mais rápido") devem ser implementados para a Aliança, de modo que eles recebam os planos de dominação primeiro. Que a força esteja com você!

Detalhes do problema

O objetivo deste trabalho é praticar os conceitos relacionados aos diferentes algoritmos de ordenação para resolver o problema apresentado a seguir: o imperador Vader deseja executar um plano de dominação de civilizações. Para tal, é necessário que, dado um conjunto de civilizações, elas sejam ordenadas seguindo dois diferentes aspectos, apresentados a seguir:

- **Distância da civilização**: Prioritariamente, as civilizações devem ser dominadas de acordo com a sua distância para o quartel general do imperador, de modo que uma civilização mais próxima deve ser dominada primeiro. Desse modo, dada duas civilizações c_1 e c_2, com distâncias d_1 e d_2 respectivamente, se $d_1 < d_2$, c_1 deve ser dominada antes de c_2. Caso $d_1 > d_2$, c_2 deve ser dominada antes de c_1. Caso $d_1 = d_2$, o critério de desempate será pela análise do tamanho da população da civilização;
- **Tamanho da população**: caso haja empate na distância de duas civilizações, deve-se realizar a ordenação das civilizações empatadas de acordo com o tamanho da população, de modo que civilizações com populações maiores sejam dominadas primeiro. Desse modo, dada duas civilizações

c_1 e c_2, com populações p_1 e p_2 respectivamente, se $p_1 > p_2$, c_1 deve ser dominada antes de c_2. Caso $p_1 < p_2$, c_2 deve ser dominada antes de c_1.

Neste trabalho, o problema deve ser resolvido em duas versões diferentes, **na qual você deve escolher, no mínimo, dois métodos diferentes para cada versão**:

- **Plano para a Aliança Rebelde**: nesta versão, os algoritmos de ordenação utilizados devem possuir a menor complexidade assintótica (recomenda-se utilizar algoritmos de ordenação com complexidade assintótica eficiente). Desse modo, o seu tempo de execução será o menor possível.
- **Plano para o imperador**: nesta versão, os algoritmos de ordenação utilizados devem possuir complexidade assintótica maior do que os considerados na primeira versão (recomendável que sejam algoritmos com complexidade assintótica quadrática), de modo que, caso as versões fossem executadas ao mesmo tempo, o imperador receberia o plano de dominação depois da Aliança Rebelde.

Sugestões de estratégias a serem seguidas para a resolução deste Trabalho Prático

Atenção: esta seção apresenta sugestões de estratégias para resolução do problema proposto. As abordagens aqui tratadas não são, necessariamente, as únicas possíveis para resolução, e sim, apenas um ponto de partida. Como citado anteriormente, a sua solução deve levar em consideração a ordenação através de duas chaves comparativas: 1) a distância da civilização; e 2) o tamanho da população. A segunda chave comparativa deve ser considerada apenas para os casos em que há um empate na primeira chave comparativa. A seguir, vamos apresentar duas sugestões de resolução.

A primeira sugestão refere-se a implementação de algoritmos de ordenação que, no momento da comparação de chaves, realize uma dupla comparação. Em outras palavras, deve-se implementar os algoritmos clássicos com uma modificação na etapa de comparação, na qual, caso a primeira comparação (com a primeira chave) der um resultado de igualdade, deve-se imediatamente realizar uma segunda comparação, levando em consideração a segunda chave comparativa.

A segunda sugestão diz respeito a aplicar duas iterações de ordenação para o conjunto de entrada. Numa primeira iteração, deve-se realizar a ordenação por meio de um dado algoritmo, considerando uma das chaves. Após, numa segunda iteração, realiza-se novamente uma nova ordenação, considerando a chave comparativa que não foi tratada na primeira iteração.

Análise Experimental

Para demonstrar que as versões implementadas terão um tempo de resposta diferente, você deverá realizar uma análise experimental dos algoritmos implementados. Lembre-se: Para cada uma das versões, você deve implementar, **no mínimo, dois algoritmos de ordenação**.

A análise experimental consiste em coletar o tempo de execução do seu algoritmo, dado um conjunto de entrada (Este link[1] traz exemplos de como calcular o tempo de execução de um algoritmo em C e C++). Dessa maneira, você deve coletar o tempo de execução considerando todas as combinações de execução possíveis, e então, analisar qual é a combinação que representa uma maior diferença de tempo de resposta entre o sistema do imperador Vader e o da Aliança Rebelde. Em outras palavras, você deve analisar e verificar qual algoritmo representa um maior tempo de execução para o imperador e qual representa um menor tempo de execução para a Aliança Rebelde.

É importante ressaltar que, embora espera-se que os tempos de execução sejam diferentes, a **saída produzida deve ser a mesma** para ambas as versões, sendo a única diferença o tempo de execução de ambos. Para demonstrar as diferenças relacionadas ao tempo de execução, serão disponibilizados 10 arquivos de entrada, na qual você deve executá-los em ambas as versões e apresentar uma análise dos tempos por meio de tabelas e/ou gráficos. Na Tabela a seguir, são apresentados os arquivos de entrada e o respectivo número de civilizações, representados em cada um deles.

[1] `https://www.geeksforgeeks.org/measure-execution-time-with-high-precision-in-c-c/`

Arquivo de entrada	Número de civilizações
0.in	50
1.in	100
2.in	500
3.in	1000
4.in	10000
5.in	100000
6.in	250000
7.in	500000
8.in	1000000
9.in	2000000

O download dos arquivos de teste pode ser feito no seguinte link: `shorturl.at/nwGKW`

Entrada e Saída

Entrada e saída. Neste trabalho, a entrada será a padrão do sistema (`stdin`). A saída também será a padrão do sistema (`stdout`). A primeira linha da entrada é composta por um valor inteiro **N** ($0 < \mathbf{N} \leq 2000000$), que indica o total de civilizações para domínio. A seguir, haverão **N** linhas compostas por três informações, que caracterizam as civilizações:

- Um arranjo de caracteres (string) n_c, que indica o nome do planeta da civilização;
- Um valor inteiro (*int*) d_c, que indica a distância da civilização para o quartel general do Novo Império;
- Um valor inteiro (*int*) p_c, que indica o tamanho da população da civilização.

As informações acima mencionadas são separadas por um caractere de espaço. O final da entrada é indicado por `EOF` (`CTRL + D` no terminal).

A saída esperada consiste na ordenação das civilizações de entrada, seguindo as premissas apontadas na seção "Detalhes do problema". Cada linha de saída deve apresentar os dados de uma civilização da mesma maneira da entrada: n_c d_c p_c

Informações adicionais. Para este problema, você pode assumir que o sistema é coeso, não havendo civilizações com o mesmo tamanho populacional, dado que elas estão a uma mesma distância.

Exemplo

Exemplo de Entrada	Exemplo de Saída
5 amxnahseha 10 1000 znamsjeahs 1 5000 wiajamznaj 10 7501 amajwhazzz 10 7500 laksjahsnn 40 35000	 znamsjeahs 1 5000 wiajamznaj 10 7501 amajwhazzz 10 7500 amxnahseha 10 1000 laksjahsnn 40 35000

Entregáveis

Código-fonte. A implementação poderá ser feita utilizando as linguagens C ou C++. Não será permitido o uso da *Standard Library* do C++ ou de bibliotecas externas que implementem as estruturas

de dados ou os algoritmos. **A implementação das estruturas e algoritmos utilizados neste trabalho deve ser sua**. A utilização de *Makefile*[2] é **obrigatória** para este trabalho.

Aplique boas práticas de programação e organize seu código-fonte em arquivos, classes e funções de acordo com o significado de cada parte. A separação de responsabilidades é um dos princípios da engenharia de software: cada função deve realizar apenas uma tarefa e cada classe deve conter apenas métodos condizentes com sua semântica.

Documentação. A documentação de seu programa **deverá** estar em formato **PDF**, **seguir** o modelo de trabalhos acadêmicos da SBC (que pode ser encontrado online[3]) e ser **sucinta, não ultrapassando 12 páginas**.

A documentação deverá conter **todos** os seguintes tópicos:

- Cabeçalho. Título do trabalho, nome e número de matrícula do autor.
- Introdução. Apresentação do problema abordado e visão geral sobre o funcionamento do programa.
- Implementação. Nesse trabalho, é **essencial** que você descreva qual estratégia utilizou para garantir a ordenação de acordo com os dois campos comparativos (distância e população). Além disso, deve ser feita a descrição sobre a implementação do programa. Devem ser detalhados o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, compilador utilizado, bem como decisões tomadas relativas aos casos e detalhes que porventura estejam omissos no enunciado;
- Instruções de compilação e execução. Instruções de como compilar e executar o programa.
- Análise de complexidade. Estudo da **complexidade de tempo e espaço** dos algoritmos de melhor e pior caso desenvolvido utilizando o formalismo da notação assintótica.
- Comparação de tempo de execução dos algoritmos. Apresentar a comparação de tempo de execução das duas versões desejadas neste trabalho, exibindo por meio de tabelas e/ou gráficos como eles se comportam e se diferenciam de acordo com o tamanho dos 10 arquivos de entrada. É necessário realizar uma análise textual que discute os resultados obtidos e qual a relação por eles expressas de acordo com a complexidade assintótica de tempo dos algoritmos implementados. Este link[4] traz exemplos de como calcular o tempo de execução de um algoritmo em C e C++. Utilize essa medição apenas para realizar os seus experimentos, não é necessário deixá-los na versão final de código fonte entregue;
- Conclusão. Resumo do trabalho realizado, conclusões gerais sobre os resultados e eventuais dificuldades ou considerações sobre seu desenvolvimento.
- Bibliografia. Fontes consultadas para realização do trabalho.

O código-fonte e a documentação devem ser organizados como demonstrado pela árvore de diretórios na Figura 1 (Veja na próxima página). O diretório raiz deve ser nomeado de acordo seu **nome e último sobrenome**, separado por *underscore*, por exemplo, o trabalho de "Kristoff das Neves Björgman" seria entregue em um diretório chamado `kristoff_bjorgman`. Este diretório principal deverá conter um subdiretório chamado `src`, que por sua vez conterá os códigos (`.cpp, .c, .h, .hpp`) na estrutura de diretórios desejada. A documentação **em formato PDF** deverá ser incluída no diretório raiz do trabalho. Evite o uso de caracteres especiais, acentos e espaços na nomeação de arquivos e diretórios.

O diretório deverá ser submetido em um único arquivo **'nome_sobrenome.zip'**, (onde nome e sobrenome seguem as mesmas diretrizes para o nome do diretório, explicado acima) através do Moodle da disciplina até as **23:59** do dia **15 de outubro de 2020**.

[2] `https://opensource.com/article/18/8/what-how-makefile`

[3] `http://www.sbc.org.br/documentos-da-sbc/summary/169-templates-para-artigos-e-capitulos-de-livros/878-modelosparapublicaodeartigos`

[4] `https://www.geeksforgeeks.org/measure-execution-time-with-high-precision-in-c-c/`

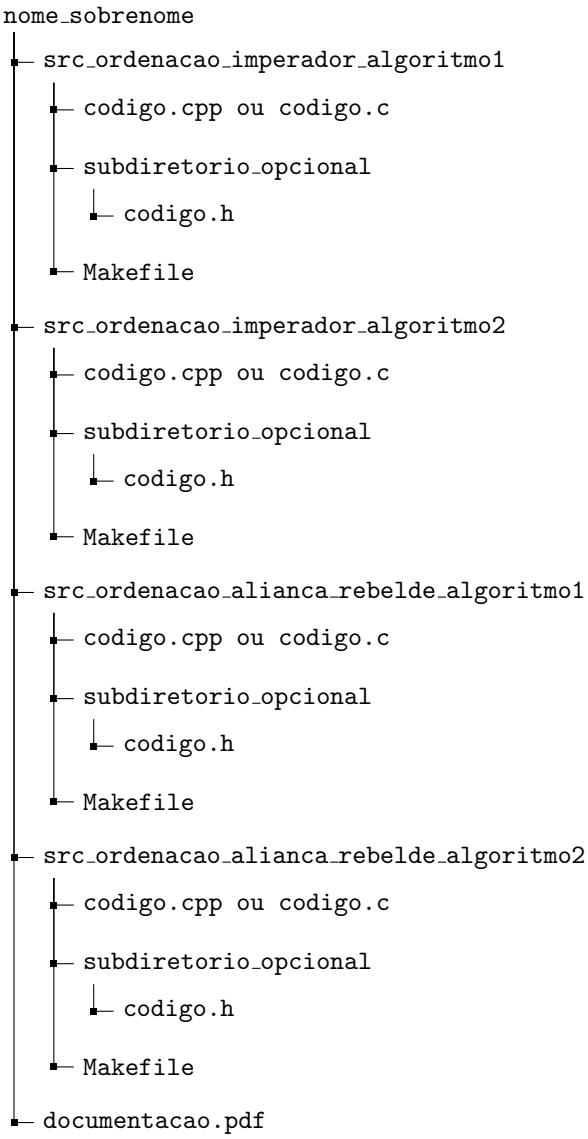

Figura 1: Estrutura de diretórios do entregável do TP2

Considerações Finais

Algumas considerações finais importantes:

- **Preste bastante atenção nos detalhes da especificação**. Cada detalhe ignorado acarretará em perda de pontos.
- O que será avaliado no trabalho:

  Boas práticas de programação: se o código está bem organizado e identado, com comentários explicativos, possui variáveis com nomes intuitivos, modularizado, etc.

  Implementação correta dos algoritmos: se as estruturas abstratas de dados foram implementadas de forma correta e resolvem o problema aqui descrito.

Conteúdo da documentação: se todo o conteúdo necessário está presente, reflete o que foi implementado e está escrito de forma coerente e coesa.

- Após submeter no Moodle seu arquivo **'.zip'**, faça o download dele e certifique-se que não está corrompido. Não será dada segunda chance de submissão para arquivos corrompidos.
- Em caso de dúvidas, **não hesite em perguntar** no Fórum de Discussão no Moodle ou procurar os monitores da disciplina – estamos aqui para ajudar!
- **PLÁGIO É CRIME**: caso utilize códigos disponíveis online ou em livros, **referencie** (inclua comentários no código fonte e descreva a situação na documentação). Trabalhos onde o plágio for identificado serão devidamente penalizados: o aluno terá seu trabalho anulado e as devidas providências administrativas serão tomadas. Discussões sobre o trabalho entre colegas são encorajadas, porém compartilhamento de código ou texto é plágio e as regras acima também se aplicam.
- Em caso de atraso na entrega, serão descontados $2^d - 1$ pontos, onde d é o número de dias (corridos) de atraso arredondado para cima.
- Comece o trabalho o mais cedo possível. Você nunca terá tanto tempo pra fazê-lo se começar agora!

Bom trabalho!

Apêndices

A Dicas para a documentação

O objetivo desta seção é apresentar algumas dicas para auxiliar na redação da documentação do trabalho prático.

1. **Sobre *Screenshots***: ao incluir *screenshots* (imagens da tela) em sua documentação, evite utilizar o fundo escuro. Muitas pessoas preferem imprimir documentos para lê-los e imagens com fundo preto dificultam a impressão e visualização. Recomenda-se o uso de fundo branco com caracteres pretos, para *screenshots*. Evite incluir trechos de código e/ou pseudocódigos utilizando *screenshots*. Leia abaixo algumas dicas de como incluir código e pseudocódigo em seu texto.

2. **Sobre códigos e pseudocódigos**: Ao incluir este tipo de texto em sua documentação procure usar a ferramenta adequada para que a formatação fique a melhor possível. Existem várias ferramentas para LaTeX, como o `minted`[5], o `lstlisting`[6], e o `algorithm2e`[7] (para pseudocódigos), e algumas para Google Docs (`Code Blocks`[8], Code Pretty[9]).

3. **Sobre URLs e referências**: evite utilizar URLs da internet como referências. Geralmente URLs são incluídas como notas de rodapé. Para isto basta utilizar o comando `\footnote{\url{}}` no LaTeX, ou ativar a opção nota de rodapé[10] no Google Docs/MS Word.

4. **Evite o Ctrl+C/Ctrl+V**: encoraja-se a modularização de código, porém a documentação é única e só serve para um trabalho prático. Reuso de documentação é auto-plágio[11]!

[5] `https://www.overleaf.com/learn/latex/Code_Highlighting_with_minted`
[6] `https://www.overleaf.com/learn/latex/Code_listing`
[7] `https://en.wikibooks.org/wiki/LaTeX/Algorithms`
[8] `https://gsuite.google.com/marketplace/app/code_blocks/100740430168`
[9] `https://chrome.google.com/webstore/detail/code-pretty/igjbncgfgnfpbnifnnlcmjfbnidkndnh?hl=en`
[10] `https://support.office.com/en-ie/article/insert-footnotes-and-endnotes-61f3fb1a-4717-414c-9a8f-015a5f3ff4cb`
[11] `https://blog.scielo.org/blog/2013/11/11/etica-editorial-e-o-problema-do-autoplagio/#.XORgbdtKg5k`