Universidade Federal de Minas Gerais — Estruturas de Dados 2019/2
Instituto de Ciências Exatas — Prof. Carlos H. C. Teixeira, Luiz Chaimowicz, Olga N. Goussevskaia
Departamento de Ciência da Computação — Monitores: Álisson R. Svaigen e Matheus A. Diniz

Trabalho Prático 2: O problema da agenda de viagens de Rick Sanchez

Valor: 10 pontos
Data de entrega: 10 de novembro de 2019

Introdução

Está lembrado de Rick Sanchez? Ele ficou muito contente com a sua solução computacional para o problema das medições nos recipientes! Agora, ele precisa de você para resolver uma outra situação: como ele é um cientista de renome intergalático, recebe constantemente convites para visitar e dar palestras em diferentes planetas. Ele geralmente aceita o convite, desde que possa decidir o dia e o tempo que permanecerá no planeta, visitando-o. Além disso, Rick sempre aloca um tempo **T**, em minutos, para visitação de planetas durante um mês.

Rick gostaria de automatizar a sua agenda de viagens interplanetárias. Para isso, ele pediu para você desenvolver um programa que, dado uma especificação de **P** planetas e o tempo t que ele deve permanecer em cada um, informe como saída uma agenda de visitação dos planetas em cada mês, dado que o tempo total de visitação em um mês não ultrapasse o tempo **T**.

Como todo cientista, Rick tem suas manias: ele quer visitar o máximo de planetas possíveis no primeiro mês, ou seja, os k planetas de menor tempo de visitação. Para o próximo mês, caso haja planetas ainda não visitados, pretende novamente visitar o máximo possível, considerando os $\{P - k\}$ planetas restantes, e assim sucessivamente. Além disso, dados os k planetas, a visitação deve acontecer seguindo a ordem alfabética em k.

Rick ainda lhe propõe um grande desafio: ele garante que é possível organizar quais planetas devem ser visitados em cada mês com um algoritmo de ordenação **estável** de complexidade de tempo $O(n \log_2 n)$. Já para indicar a ordem de visitação dos planetas no mês, Rick garante que é possível fazer este processamento com complexidade de tempo $O(n \times k)$, dado que k é tamanho da cadeia de caracteres que identifica um planeta.

Detalhes do problema

O objetivo deste trabalho é praticar os conceitos relacionados a algoritmos clássicos de ordenação. A seguir, são esclarecidos detalhes relevantes que compõem o problema:

- **T** indica o tempo em minutos alocados por Rick para visitar planetas durante 1 mês, tal que $T > 0$;
- São especificados **P** planetas, tal que $0 < P \leq 150000$. Cada especificação de um planeta $p \in P$ possui:
  - Um nome composto por uma cadeia de x caracteres (o tamanho da cadeia é o mesmo para todos os planetas), tal que $x < \log_2 P$;
  - Um tempo inteiro t em minutos, tal que $0 < t \leq T$;
- O primeiro mês de visitação deve ter índice 1. Para cada mês de visitação, o tempo total não deve exceder **T**;

- Dados dois meses de visitação subsequentes, M_1 e M_2, o número de planetas visitados em M_1 deve ser sempre maior ou igual o número de planetas visitados em M_2;
- Para qualquer planeta p_1 agendado para o mês M_1, seu tempo t de visitação deve ser sempre menor ou igual qualquer tempo de visitação de um planeta p_2 de um mês M_2;
- Para um determinado mês M com k planetas agendados, o programa deve indicar para Rick que a visitação deve ocorrer seguindo a ordem alfabética dos nomes dos k planetas.

Entrada e Saída

Os formatos desejados de entrada e saída são:

Entrada. Neste trabalho, a entrada será a padrão do sistema (`stdin`). A primeira linha de uma instância de entrada do problema é composta por 3 valores inteiros T, P, x, correspondendo, respectivamente, ao tempo em minutos para visitação de planetas durante 1 mês, o número total de planetas e o total de caracteres de cada nome de planeta. Após a primeira linha, haverão P linhas com informações sobre cada planeta. Cada uma das P linhas são compostas por um inteiro t e uma cadeia de caracteres de tamanho x, que representam, respectivamente, o tempo de visitação do planeta e o seu nome.

Saída. Neste trabalho, a saída será a padrão do sistema (`stdout`), e ela representará a agenda de visitação dos planetas. Ela deve ser composta por P linhas, sendo que cada uma deve possuir um inteiro M, uma cadeia de caracteres cp e um inteiro t, indicando, respectivamente, o mês de visitação, o planeta visitado e o tempo de visitação daquele planeta. Note que, na saída, os meses devem ser listados de maneira ordenada. Além disso, para um determinado mês M, os planetas devem aparecer de maneira alfabeticamente ordenada.

Exemplos

Exemplo de Entrada	Exemplo de Saída
720 5 2 60 ag 60 aa 240 dc 700 be 90 zz	1 aa 60 1 ag 60 1 dc 240 1 zz 90 2 be 700

Exemplo de Entrada	Exemplo de Saída
300 9 3 290 aaa 10 xgf 100 dcz 20 aba 20 abb 10 fff 150 ggd 150 ggc 140 xax	1 aba 20 1 abb 20 1 dcz 100 1 fff 10 1 xax 140 1 xgf 10 2 ggc 150 2 ggd 150 3 aaa 290

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. É expressamente proibido utilizar rotinas prontas de ordenação**. Os códigos devem ser executáveis em um computador com *Linux*. Caso não possua um computador com *Linux*, teste seu trabalho em um dos computadores do laboratório de graduação do CRC[1]. 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** exclusivamente o modelo de trabalhos acadêmicos da SBC (que pode ser encontrado online[3]) e ser **sucinta**, não excedendo o limite de 7 páginas. 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. Descrição sobre a implementação do programa. **Devem ser justificadas as escolhas dos algoritmos de ordenação implementados** (pode-se utilizar o material de aula da disciplina). 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. **Não devem ser inseridos trechos de código fonte na documentação**.
- Instruções de compilação e execução. Instruções de como compilar e executar o programa.
- Análise de complexidade. Estudo da complexidade de tempo e espaço do algoritmo de melhor e pior caso desenvolvido utilizando o formalismo da notação assintótica.
- 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. 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 **10 de novembro de 2019**.

[1] `https://crc.dcc.ufmg.br/infraestrutura/laboratorios/linux`

[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`

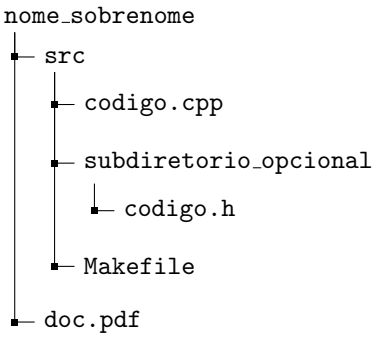

Figura 1: Estrutura de diretórios do entregável do TP2

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 está código bem organizado e indentado, com comentários explicativos, possui variáveis com nomes intuitivos, modularizado, etc.

  Implementação correta dos algoritmos: se a árvore e o processo de decodificação foram implementados 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*.

2. **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[4] no Google Docs/MS Word.

3. **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[5]!

[4] `https://support.office.com/en-ie/article/insert-footnotes-and-endnotes-61f3fb1a-4717-414c-9a8f-015a5f3ff4cb`
[5] `https://blog.scielo.org/blog/2013/11/11/etica-editorial-e-o-problema-do-autoplagio/#.XORgbdtKg5k`