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 3: As mensagens secretas da Aliança Rebelde

Valor: 10 pontos
Data de entrega: 30 de Outubro de 2020

Introdução

O seu plano para auxiliar a Aliança Rebelde foi um sucesso! Com os planos de dominação do imperador sendo entregues antecipadamente para a Aliança, eles podem colocar o seu plano de defesa em prática, de modo a evitar os ataques do imperador Vader. Para que a estratégia de defesa seja alcançada e sincronizada entre todas as civilizações, uma série de mensagens devem ser transmitidas. Com essas mensagens sendo transmitidas, há uma nova esperança de que o império possa ser, finalmente, derrotado!

Há, no entanto, um receio de que as forças do império podem estar tentando interceptar os meios de comunicação interplanetárias. Desse modo, as mensagens transmitidas passam por um processo de criptografia, ou seja: a mensagem enviada possui palavras que não correspondem a mensagem verdadeira, podendo ser lida corretamente apenas por quem é da Aliança.

De modo a automatizar o processo de encriptação e decriptação das mensagens, você, caro(a) aluno(a), tem uma fundamental e última missão: desenvolver o sistema de transmissão de mensagens encriptadas da Aliança Rebelde. Nesse sistema, você deverá implementar uma árvore binária, que é a base para o sistema de encriptação das mensagens da Aliança. Vamos lá, caro(a) aluno(a), essa é a sua última missão! Que a força esteja com você!

Detalhes do problema

O objetivo deste trabalho é praticar os conceitos relacionados a implementação e manipulação de árvores binárias para resolver o problema apresentado a seguir: dado um idioma de entrada, as mensagens da Aliança Rebelde a serem transmitidas devem ser encriptadas seguindo a sua disposição numa árvore binária, de modo que uma mensagem em texto comum seja transformada num texto "ilegível" por quem não conhece a criptografia. Do mesmo modo, uma mensagem criptografada deve ser descriptografada também seguindo a disposição numa árvore binária.

Além disso, o idioma utilizado pela Aliança é muito dinâmico, na qual as palavras mudam constantemente. Assim, é também necessário substituir uma palavra por outra. Considerando a árvore binária, isso consiste na pesquisa e remoção da palavra a ser substituída, bem como a inserção da nova palavra. Lembre-se de tomar todos os cuidados necessários na remoção para manter a árvore binária consistente!

Desse modo, neste trabalho será necessário realizar quatro tarefas principais, que são listadas a seguir:

- **Adição de palavras do idioma**: A adição de novas palavras do idioma consiste em construir uma árvore binária, dadas as palavras do idioma, seguindo a ordem que elas são informadas. A construção da árvore segue a ordenação alfabética de modo que, dado um nó existente na árvore n_1 e a inserção de um novo nó n_2, caso o nó n_2 tenha ordem alfabética anterior ao nó n_1, ele deve ser inserido à esquerda do nó, caso contrário, ele deve ser inserido à direita do nó.
- **Substituição de palavras do idioma**: A substituição de palavras do idioma consiste em três passos:
  - Pesquisar a palavra a ser substituída;

  - Remover da árvore binária a palavra a ser substituída do idioma;
  - Adição da nova palavra na árvore binária;

- **Encriptação de uma mensagem**: Considerando que a árvore já está construída, a **encriptação** de uma mensagem ocorre no mapeamento das palavras para a sua posição na árvore, considerando a **visitação pré-ordem**. Exemplificando: suponha que uma mensagem seja "absha ut alehsa". Suponha que, pela visitação pré-ordem, a palavra "absha" seja a de ordem 1, a palavra "ut" seja a de ordem 4, e a palavra "alehsa" seja a de ordem 3. Assim, a mensagem encriptada será " 1 4 3 ".

- **Desencriptação**: Do mesmo modo, dado que a árvore binária está construída, deve ser possível realizar a "desencriptação" de uma mensagem criptografada. Desse modo, considerando a mensagem criptografada "1 4 3", a mensagem no idioma comum da Aliança será "absha ut alehsa".

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 entrada será composta por **N** linhas, tal que $0 < \mathbf{N} \leq 10000$). Em cada linha haverá uma operação a ser realizada pelo seu sistema, cujo primeiro caractere da linha indica qual é a operação correspondente, conforme segue:

- **i**: indica uma operação de inserção de palavra na árvore. Para esta operação, uma linha de entrada é constituída pelo caractere 'i' e uma string s, tal que o tamanho de $s \leq 15$ caracteres, sendo a palavra a ser inserida. Para esta operação, não deve ser impresso nada na saída do programa;

- **s**: indica uma operação de substituição de palavra na árvore. Para esta operação, uma linha de entrada é constituída pelo caractere 's'; uma string $s1$, tal que o tamanho de $s1 \leq 15$ caracteres, sendo a palavra a ser substituida; e uma string $s2$, tal que o tamanho de $s2 \leq 15$ caracteres, sendo a nova palavra a ser inserida na árvore. Para esta operação, não deve ser impresso nada na saída do programa;

- **e**: indica uma operação de encriptação de uma mensagem. Para esta operação, uma linha de entrada é constituída pelo caractere 'e'; um inteiro i, tal que $i \leq 20$, que indica o número de palavras que compõem a mensagem; e uma mensagem M, constituída por i palavras. Para esta operação, a mensagem encriptada deve ser impressa na saída do programa.

- **d**: indica uma operação de desencriptação de uma mensagem. Para esta operação, uma linha de entrada é constituída pelo caractere 'd'; um inteiro i, tal que $i \leq 20$, que indica o número de palavras que compõem a mensagem; e uma mensagem M, constituída por i palavras. Para esta operação, a mensagem desencriptada deve ser impressa na saída do programa.

O final da entrada é indicado por `EOF` (`CTRL + D` no terminal).

Informações adicionais. Para este problema, você pode assumir que o sistema é coeso, sendo as seguintes premissas sempre verdadeiras:

- Não haverá a solicitação de inserção de uma palavra caso ela já esteja presente na árvore;

- Não haverá a solicitação de substituição de uma palavra que não está na árvore. Ainda para esta operação, não haverá como palavra substituta uma que já está na árvore;

- Para as solicitações de encriptação, todas as palavras da mensagem a ser encriptada constarão na árvore a ser criada;

- Para as solicitações de desencriptação, todas as palavras de código pré-ordem são possíveis de serem alcançados na árvore.

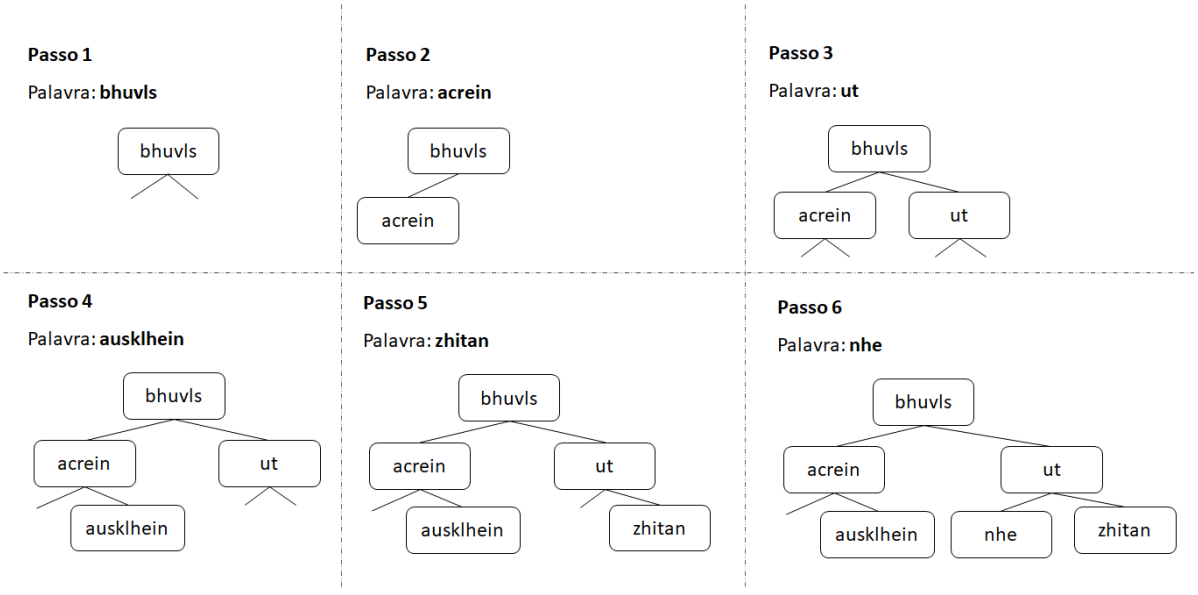
Exemplo

Exemplo de Entrada	Exemplo de Saída
i bhuvis	
i acrein	
i ut	
i ausklhein	
i zhitan	
i nhe	
i baiean	
i skhuls	
e 4 skhuls ut nhe acrein	7 5 6 2
d 5 1 2 3 5 7	bhuvis acrein ausklhein ut skhuls
s skhuls aarain	
e 4 aarain ut nhe acrein	3 6 7 2

Exemplo passo a passo

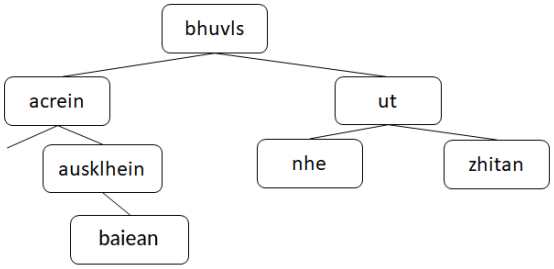
Para clarificar as definições do problema apresentados na seção anterior, vamos apresentar a resolução passo a passo do exemplo de entrada a saída supracitado.

As 8 primeiras linhas do exemplo de entrada são compostos pela inserção de palavras. Desse modo, devemos construir a árvore binária. Assim, a construção da árvore, passo-a-passo, ocorreria conforme exibido nas imagens a seguir:



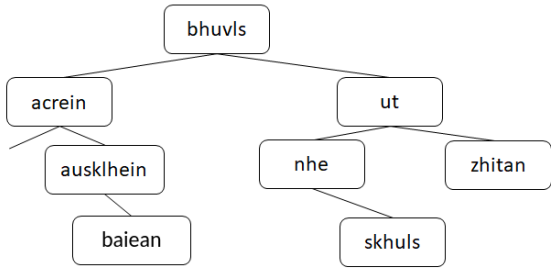
Passo 7

Palavra. **baiean**

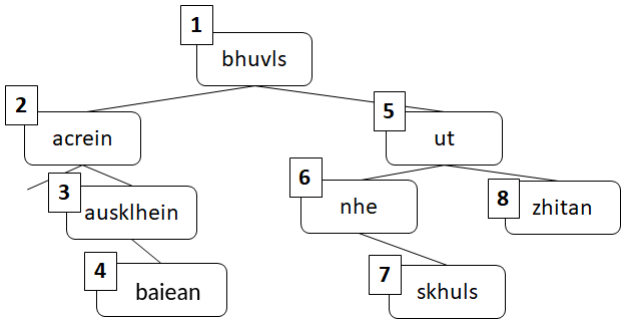


Passo 8

Palavra: **skhuls**



Com a configuração atual da árvore, numa visitação pré-ordem teríamos os códigos exibidos na figura a seguir.



Seguindo, a próxima linha do exemplo de entrada solicita a encriptação da seguinte mensagem:

```
skhuls ut nhe acrein
```

A mensagem encriptada seria dada respectivamente pelos "valores" da visitação pré-ordem. Sendo:

```
7 5 6 2
```

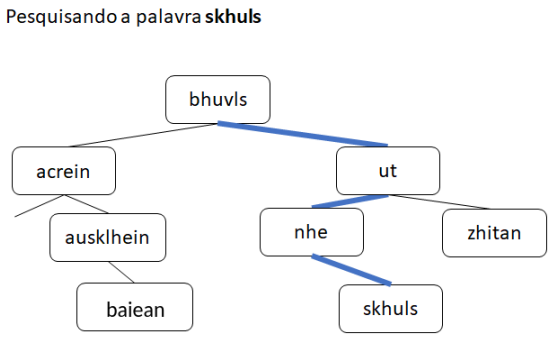
Na próxima linha, há a solicitação de uma desencriptação para a mensagem:

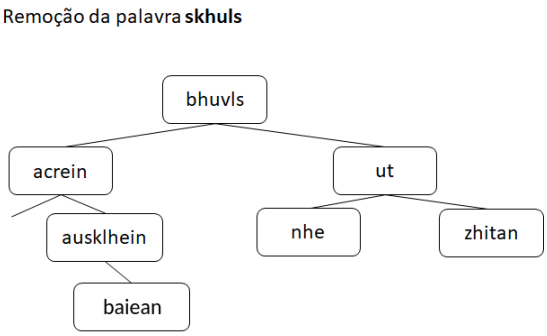
```
1 2 3 5 7
```

A mensagem desencriptada correspondente, seguindo o código pré-ordem, seria:

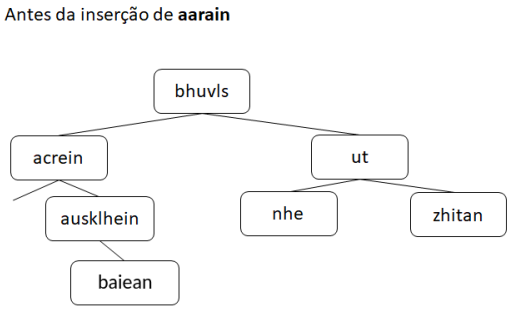
```
bhuvis acrein ausklhein ut skhuls
```

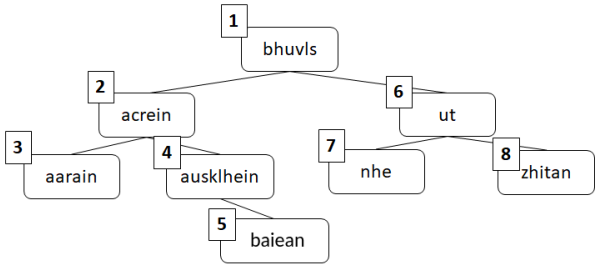
A próxima linha solicita a substituição da palavra "skhuls" pela palavra "aarain". Para isso, primeiramente devemos pesquisar e remover a palavra "skhuls". A imagem a seguir apresenta as pesquisas a partir da raiz, indicadas pelas linhas azuis, bem como a sua remoção.





Após a remoção, deve-se realizar a inserção da palavra "aarain", conforme apresenta a imagem a seguir. Note que, na árvore da direita, já estão indicados os valores da visitação pré-ordem.





Finalizando, a última linha do exemplo de entrada refere-se a uma encriptação da palavra:

```
aarain ut nhe acrein
```

Pela nova árvore, a mensagem encriptada é:

```
3 6 7 2
```

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*[1] é **obrigatória** para este trabalho. O seu Makefile **deve obrigatoriamente gerar** um arquivo executável cujo nome seja **tp3**, para que então se possa avaliar o seu código de maneira automatizada.

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.

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

Documentação. A documentação de seu programa **deverá** estar em formato **PDF**, **seguir** o modelo de trabalhos acadêmicos da SBC (que pode ser encontrado online[2]) e ser **sucinta, não ultrapassando 10 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 detalhadamente 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.
- 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.

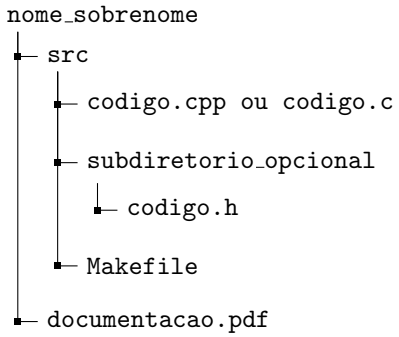


Figura 1: Estrutura de diretórios do entregável do TP3

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 **30 de Outubro de 2020**.

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

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`[3], o `lstlisting`[4], e o `algorithm2e`[5] (para pseudocódigos), e algumas para Google Docs (`Code Blocks`[6], Code Pretty[7]).

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[8] 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[9]!

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