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, 09 de março de 2021

Trabalho Prático III: Um resgate inesperado

Trabalho Individual. Valor: 10 pontos **Entrega: 28 de março de 2021**

1 Introdução

Finalmente, após a jornada pela coleta dos recursos e organização da logística para a viagem de retorno, a frota da missão de exploração interplanetária do composto-Z pôde entrar em curso de volta para a Terra. Toda a humanidade celebrou os feitos do Instituto de Ciências Exoplanetárias e as equipes de pesquisa ficaram muito satisfeitas com seu trabalho. Contudo, nem tudo são boas notícias. No meio do caminho, piratas mercenários de um sistema planetário próximo conseguiram bloquear os sinais vindos da central de controle. Assim, eles mantém as naves imobilizadas no espaço profundo e ganham tempo para chegar até elas para roubar o precioso carregamento de composto-Z.

Felizmente, a missão conseguiu restabelecer uma comunicação de emergência. As naves podem utilizar um canal especial para enviar informações importantes ao centro de controle da missão e podem também receber mensagens com comandos para tirar-lhes dessa enrascada. Por segurança, as mensagens do canal são codificadas para confundir os terríveis, porém não muito espertos, piratas. Cada letra é representada por um código alfanumérico que decodifica o caminho a ser percorrido numa árvore binária. Apenas a central conhece a árvore que foi utilizada para codificar as mensagens de cada nave. Sua tarefa é construir um sistema decodificador/codificador para auxiliar o comando da missão nesse processo.

2 Especificações

Você receberá uma sequência de caracteres que deverão ser dispostos numa árvore binária, conforme o valor alfabético de cada letra. O alfabeto inclui as **26 letras latinas maiúsculas sem acentuações** e ainda **um caractere de espaço**. As letras devem ser dispostas na árvore na sequência em que aparecem e de modo que, em cada nó, uma letra menor esteja sempre à esquerda e uma letra maior, sempre à direita. O caractere espaço deve ser sempre considerado como menor que todas as demais letras. Para a sequência '`FMB LO`', por exemplo, a árvore resultante está ilustrada na Figura 1.

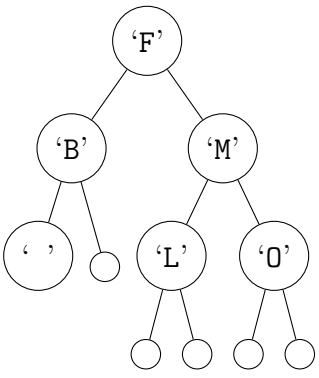

Figura 1: Exemplo de árvore resultante para uma sequência '`FMB LO`'

.

Uma mensagem é codificada **letra por letra**. Cada código consiste em algarismos '`0..9`' e um caractere '`x`', que indica a raiz. Todos os códigos são iniciados por '`x`', seguido de uma sequência de algarismos que determina o caminho a ser percorrido na árvore: se o algarismo for ímpar, segue-se para a subárvore esquerda; caso contrário, segue-se para a direita. Caso a sequência não possua nenhum algarismo, então, entende-se que o valor resultante corresponde à letra da raiz. O fim do código é delimitado por um novo caractere **x** ou pelo fim da frase. Para o exemplo da Figura 1, uma codificação válida para a frase '`BOM BOLO FOFO`' seria:

```
x1x24x8x11x3x00x63x84x31xx62xx88
```

Note que **uma mesma letra** pode ser representada de **várias formas**, como o caso da letra '`B`' no exemplo, representada como '`x1`' e '`x3`'. Tanto 3 quanto 1 são algarismos ímpares que traçam um caminho válido da raiz até o caractere '`B`'. Assim, ao codificar uma mensagem, é importante escolher **aleatoriamente** os algarismos pares e ímpares para compor a trajetória, para garantir variabilidade na codificação. Seu sistema deve ser capaz de codificar e decodificar mensagens, detectando automaticamente cada caso.

2.1 Entrada e Saída

A entrada consiste num único arquivo '`entrada.txt`', com varias linhas que representam comandos. Cada linha contem duas partes separadas por '`:`', onde a primeira parte é um comando que representa o que deve ser feito com a segunda parte. Existem três comandos, sendo eles o comando '`A`', '`D`', e '`C`'. O comando '`A`' significa que a segunda parte da linha deve ser **carregada na arvore de transliteração**, o comando '`D`' significa que a segunda parte da linha deve ser **decodificada**, e por fim o comando '`C`' significa que a segunda parte da linha deve ser **codificada**.

Ao executar um comando do tipo '`D`' ou '`C`', deve-se imprimir no terminal (stdout) o resultado do comando. Na Figura 2, encontra-se um exemplo de arquivo de entrada e a saída correspondente. A árvore obtida a partir da sequência da entrada na Figura 2a está ilustrada na Figura 3.

```
A: LPESTVRAJQKGWYX IDNZCMHBOFU
D: x7x487x405x430
C: SUCESSO
C: TUDO OK
D: x4x77x45x5
D: x46x9236x053
C: NAO
D: x6026x276x802x57x847
C: CODIGO Z
C: MISSAO
C: REBOOT
D: x0x57x08241x84x9
```

(a) Exemplo de entrada.

```
ERRO
x84x82229x1325x3x40x86x616
x280x68069x590x210x391x452x944
PANE
SIM
x67x17x870
VOLTAR
x1301x232x930x9254x585x614x379x8266268
x457x5694x48x46x57x230
x661x1x73235x470x290x084
PAUSE
```

(b) Exemplo de saída esperada.

Figura 2: Exemplo de arquivo de entrada e a saída esperada.

3 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**. Não utilize instruções de sistema operacional,

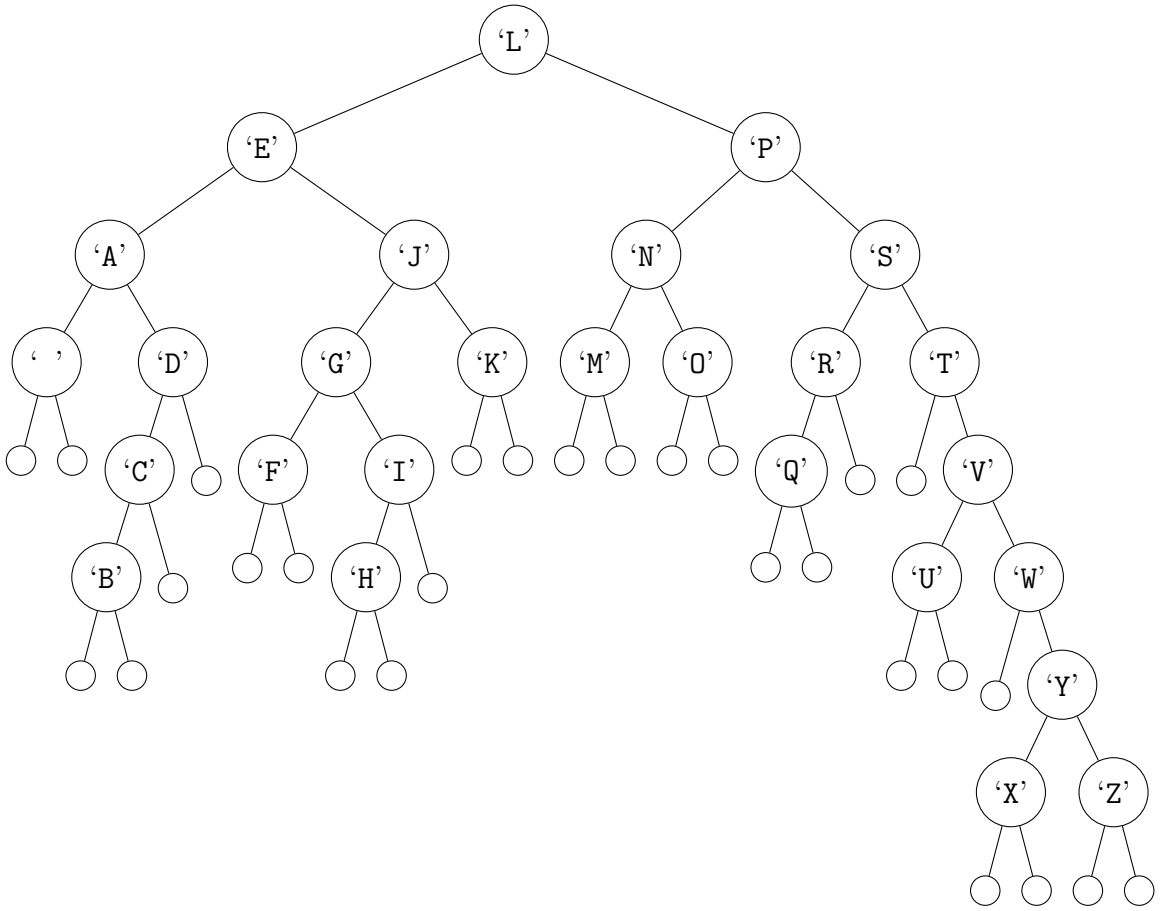

Figura 3: Exemplo de árvore resultante para a sequência da Figura 2a

.

como '`system("pause")`'. 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. Códigos-fonte que não atenderem a essas especificações **não serão avaliados**. Caso deseje, você pode reaproveitar implementações de **seus** trabalhos anteriores.

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 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 caminho para o arquivo de entrada**, conforme o exemplo a seguir:

```
run.out caminho/para/entrada.txt
```

No qual, '`caminho/para/entrada.txt`' é o primeiro argumento recebido pelo seu programa (armazenado dentro de '`argv[1]`'). Você **NÃO** deve deixar o **nome do arquivo**, nem o **caminho para ele** escritos no código de forma alguma: o executável final deve obter o caminho para o arquivo **exclusiva-**

mente via linha de comando, por meio de '`argv[1]`'. Utilize apenas a saída padrão e **não gere arquivos de saída adicionais**.

O descumprimento de quaisquer requisitos de entrada/saída acima especificados gerará perda de grande parte da nota relativa ao código.

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
```

3.1 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.
- Comentários sobre a compilação e execução **apenas caso necessário**. Instruções de compilação divergentes das que foram especificada ou que exijam o uso de softwares adicionais (IDEs, sistemas operacionais, etc) além de um compilador ou auxiliar (make) serão **desconsideradas**.
- Descrição das estruturas e classes implementadas, incluindo seus atributos e métodos, bem como a análise de complexidade desses últimos.
- Conclusões, e considerações finais. Bibliografia com as fontes consultadas.

3.2 Submissão

Todos os arquivos relacionados ao trabalho devem ser submetidos na atividade designada para tal no *Moodle* dentro do prazo estipulado. Atrasos sofrerão penalização de $2^d - 1$ pontos, com $d =$ dias de atraso. A entrega deve ser feita **em um único arquivo**, com nomenclatura '`nome_sobrenome_matricula.zip`' (atenção à extensão .zip, formatos de compactação alternativos não são indicados). O arquivo zip deve conter um arquivo '`pdf`', para a documentação, e uma pasta '`projeto`', contendo tanto o código-fonte e um '`Makefile`' para compilação. **O descumprimento das especificações de submissão acarretará em penalizações na pontuação**.

4 Considerações Finais

1. Leia **atentamente** o documento de especificação, pois o descumprimento de quaisquer requisitos aqui descritos causará penalizações ou anulação da na nota final, a depender da gravidade.
2. Os principais aspectos avaliados na correção serão:
   - Corretude na codificação/decodificação das entradas;
   - Conteúdo da documentação e corretude dos estudos de complexidade;
   - 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**. Trabalhos nos quais suspeita de plágio for identificada 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 em HIPÓTESE ALGUMA**.

Bom trabalho!