Universidade Federal do Espírito Santo – Departamento de Informática
Estruturas de Dados (INF09292)
1º Trabalho Prático
Período: 2018/2
Profª Patrícia Dockhorn Costa
Email: pdcosta@inf.ufes.br

Especificação do Trabalho 1
NetMap

Data de Entrega:04/10/2018

1 INTRODUÇÃO

Este trabalho tem como objetivo praticar o uso de tipos abstratos de dados e estruturas do tipo Lista, bem como avaliar, de forma prática, os conceitos ensinados no curso de Estruturas de Dados I. Também tem como caráter contextualizar com problemas no mundo real e sugerir soluções para esses problemas.

2 DESCRIÇÃO DO TRABALHO

NetMap é uma rede que permite a passagem de dados entre terminais distribuídos. Assim como a internet, NetMap é composta por roteadores que se conectam através de enlaces que habilitam a passagem de dados pela rede. Conectados a roteadores, estão os terminais. Cada terminal pode estar conectado a somente um roteador. Já um roteador pode estar conectado a vários terminais e/ou ainda a outro(s) roteador(es). A figura 1 ilustra os componentes de um NetMap e a relação entre eles.

Como pode ser visto na figura 1, neste NetMap existem cinco roteadores e sete terminais. Os roteadores 1 e 3 estão conectados diretamente por um enlace, assim como os roteadores 2 e 3, roteadores 3 e 5 e roteadores 3 e 4. O NetMap pode ser implementado usando um conjunto de listas encadeadas: basicamente, implementa-se uma lista de Roteadores, onde cada célula contém os dados do roteador (nome e operadora).

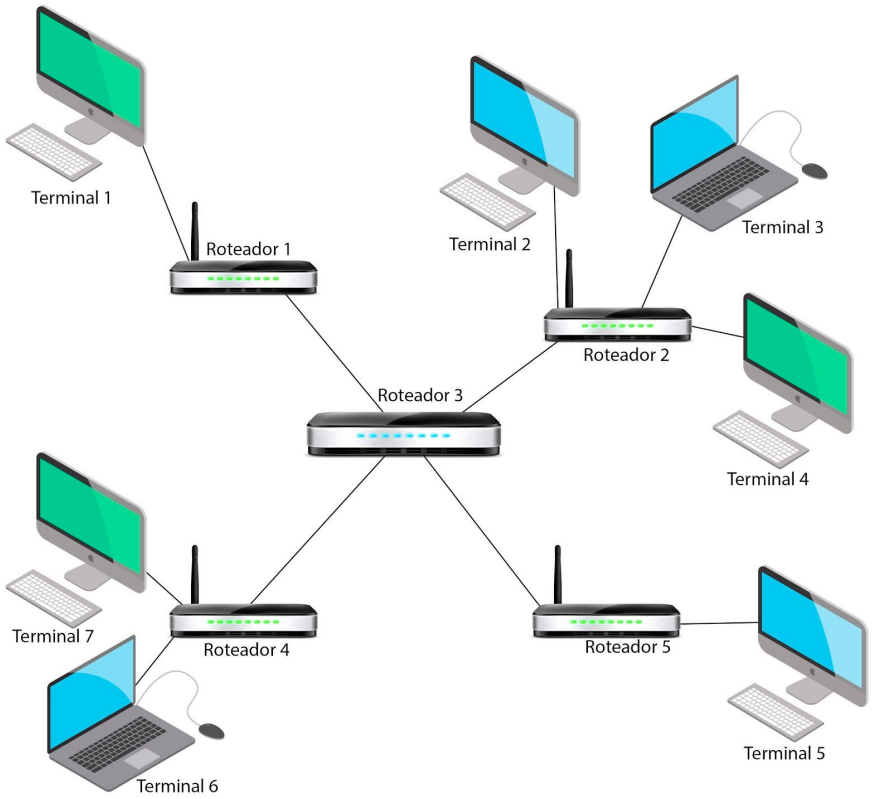

Figura 1 - Exemplo de NetMap com todos os roteadores interligados.

Além disso, cada célula desta lista de roteadores contém uma lista encadeada indicando quais são os roteadores conectados a ele, ou seja, representando os enlaces. Para implementar os terminais, pode-se implementar uma lista de terminais, na qual cada terminal possui um ponteiro para o seu roteador. Isso pode ser observado na seguinte figura (figura 2):

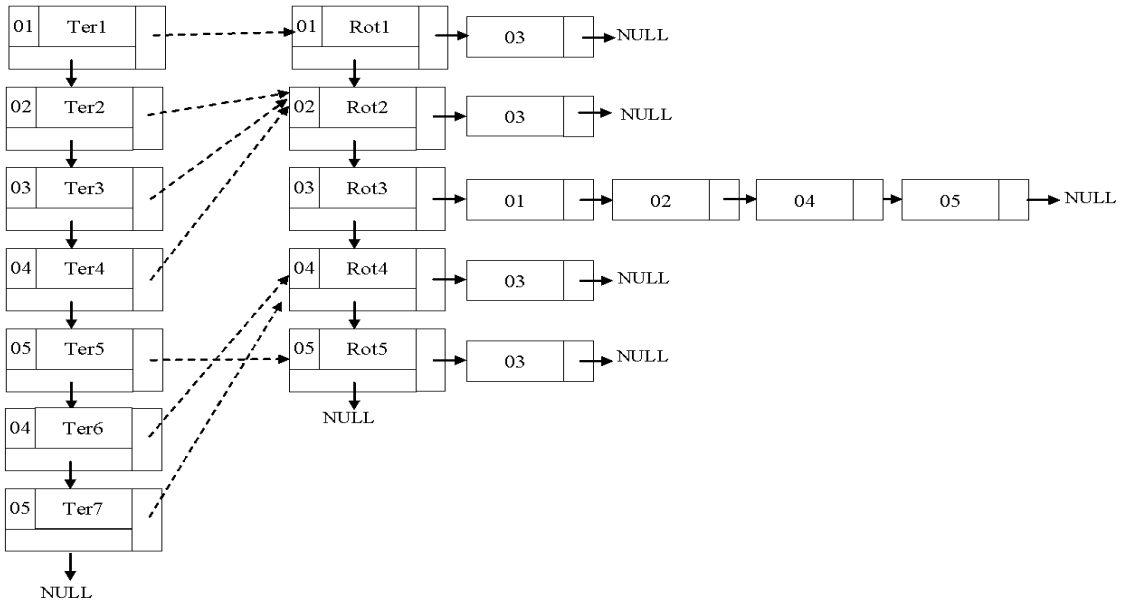

Figura 2 - Exemplo de listas de terminais e Roteadores, bem como as ligações entre eles.

Um NetMap não precisa ter todos os roteadores interligados, ou seja, a listas encadeadas que indicam quais são os roteadores ligados a eles podem ser nulas. A figura 3 ilustra um exemplo com 4 roteadores e 6 terminais. Os roteadores 1 e 3 estão conectados por um enlace, bem como os roteadores 2 e 3, porém o roteador 4 não está conectado a nenhum outro, assim este não forma um enlace com nenhum outro roteador.

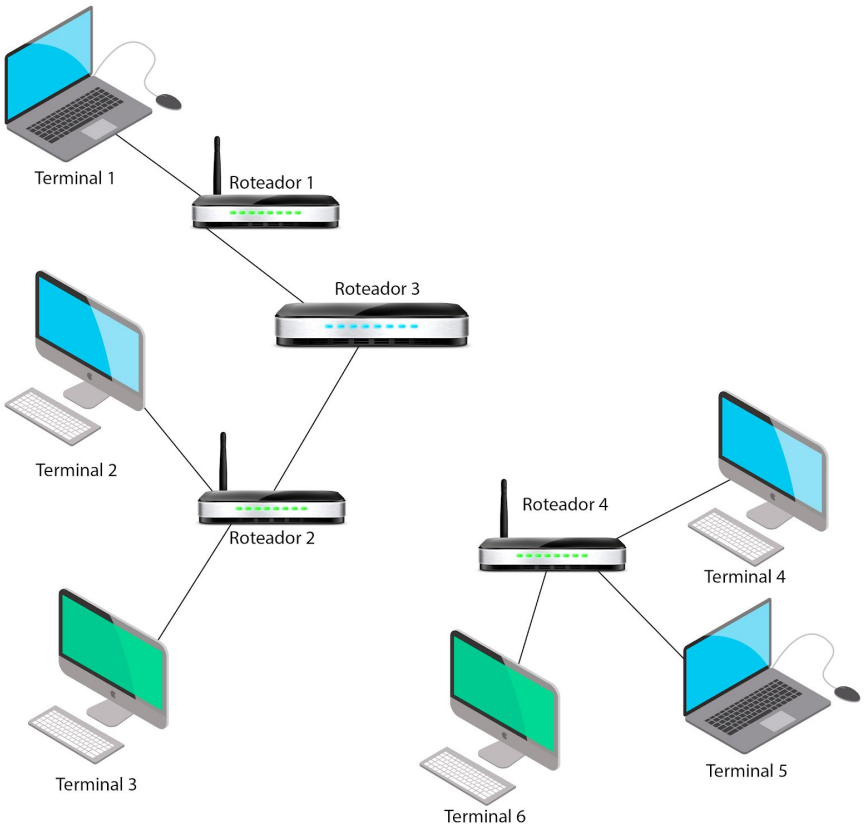

Figura 3 - Exemplo de NetMap com roteadores sem interligação.

2.1 Funcionamento Detalhado do Programa

Nesse trabalho, você deverá implementar estruturas com listas. Considere que cada roteador é uma estrutura contendo os campos nome (ex.: roteador1) e operadora do roteador (e.x., Telemar, GVT , NET, etc.), e que não há roteadores ou terminais com os mesmos nomes. De maneira similar, considere também que os terminais são estruturas contendo os campos nome (ex.: ter1) e localização do

terminal(ex.: Vitória). O NetMap não aceita nomes compostos, tanto para nomes quanto para cidades. As seguintes operações devem ser implementadas[1]:

- **CadastraRoteador**(Roteador): Cadastra um novo roteador ao NetMap.
- **CadastraTerminal**(Terminal): Cadastra um novo terminal ao NetMap.
- **RemoveRoteador**(Roteador): Remove o roteador do NetMap. (Note que todas as relações deste roteador com outros devem ser excluídas.)
- **ConectaTerminal**(Terminal, Roteador): Conecta um terminal a um roteador, ambos já cadastrados no NetMap.
- **DesconectaTerminal**(Terminal): Desconecta o terminal do seu respectivo roteador.
- **RemoveTerminal**(Terminal): Remove o terminal do NetMap.
- **ConectaRoteadores**(Roteador1, Roteador2): Cria um enlace entre dois roteadores já cadastrados.
- **DesconectaRoteadores**(Roteador1, Roteador2): Remove o enlace entre dois roteadores já cadastrados.
- **FrequenciaTerminal**(localização): Imprime o número de terminais cadastrados de uma determinada localização.
- **FrequenciaOperadora**(operadora): Imprime o número de roteadores cadastrados de uma determinada operadora.
- **EnviarPacotesDados**(Terminal1, Terminal2): Verifica se é possível enviar dados, por exemplo do Terminal1 para o Terminal2. Retorna SIM se for possível e NAO caso contrário. Por exemplo, EnviarPacotesDados(Terminal1, Terminal5) retorna SIM para o exemplo da figura 1, e NAO para o exemplo da figura 3.
- **ImprimeNetMap**():imprime todos os terminais e roteadores cadastrados em forma de grafo. Será salvo num arquivo ".dot", contendo as ligações dos roteadores e terminais. (Depois será usado o GraphViz[2] para gerar a imagem do NetMap e visualizar graficamente a rede de ligações.)

---

[1] As operações listadas aqui não representam as assinaturas das funções da implementação. Essas operações apenas indicam as principais funcionalidades que devem ser implementadas pelo seu programa, sem detalhes da implementação. Portanto, os retornos e os parâmetros das funções devem ser definidos pelo seu programa.

[2] Graphviz (abreviação de Graph Visualization Software) é um pacote de ferramentas de código aberto iniciado pela AT & T Labs Research para desenhar gráficos especificados em scripts de linguagem DOT. Um pequeno tutorial de como instalar e usar estará disponível no site (https://inf.ufes.br/~pdcosta/ensino/) em breve. Mais informações: https://www.graphviz.org/

A implementação da estrutura deverá ser feita utilizando alocação dinâmica de memória (ponteiros). Você deve fazer testes de consistência se essas operações podem ser aplicadas (teste de pré-condições) e deve imprimir mensagens de sucesso ou falha (por exemplo, a chamada de função *RemoveRoteador(Rot7)* no exemplo acima deve gerar uma mensagem de erro pois não há Roteador7 cadastrado com esse nome.

Além disso, procure escrever funções auxiliares que facilitem a implementação das operações acima, evitando a repetição desnecessária de código. Por exemplo, funções de manipulação de listas, pesquisa de id por nome, etc. Organize o seu sistema usando o conceito de **Tipos Abstratos de Dados** discutido em sala de aula. Outro fato importante de observar é sobre os cuidados para evitar vazamento de memória.

2.2 Entrada e Saída

O seu programa (NetMap) deverá ler os dados de entrada a partir de um arquivo, cujo nome é passado como parâmetro na linha de comando (Faz parte do trabalho descobrir como manipular arquivos e strings em C). Exemplo de execução do programa a partir da linha de comando:

./NetMap entrada.txt

O arquivo de entrada é basicamente uma lista de comandos (um por linha) em formato texto. O último comando é a palavra FIM, que indica o final do arquivo. O formato a ser usado é exemplificado abaixo:

Exemplo de arquivo de entrada (entrada.txt)

```
CADASTRAROTEADOR Rot1 Telemar
CADASTRAROTEADOR Rot2 Gvt
CADASTRAROTEADOR Rot3 Claro
CADASTRAROTEADOR Rot4 Telemar
CADASTRAROTEADOR Rot5 Gvt
CADASTRAROTEADOR Rot6 Gvt
CADASTRATERMINAL Ter1 Vitoria
CADASTRATERMINAL Ter2 VilaVelha
CADASTRATERMINAL Ter3 Vitoria
```

```
CADASTRATERMINAL Ter4 Colatina
CADASTRATERMINAL Ter5 Mimoso
CADASTRATERMINAL Ter6 Vitoria
CADASTRATERMINAL Ter7 Serra
CADASTRATERMINAL Ter8 Serra
REMOVEROTEADOR Rot10
REMOVEROTEADOR Rot6
REMOVETERMINAL Ter8
CONECTAROTEADORES Rot1 Rot3
CONECTAROTEADORES Rot2 Rot3
CONECTAROTEADORES Rot4 Rot3
CONECTAROTEADORES Rot5 Rot3
CONECTAROTEADORES Rot2 Rot5
CONECTATERMINAL Ter1 Rot1
CONECTATERMINAL Ter7 Rot4
CONECTATERMINAL Ter6 Rot4
CONECTATERMINAL Ter2 Rot2
CONECTATERMINAL Ter3 Rot2
IMPRIMENETMAP
CONECTATERMINAL Ter4 Rot2
CONECTATERMINAL Ter5 Rot2
CONECTATERMINAL Ter9 Rot1
DESCONECTAROTEADORES Rot2 Rot5
DESCONECTATERMINAL Ter5
CONECTATERMINAL Ter5 Rot5
FREQUENCIAOPERADORA Gvt
FREQUENCIATERMINAL Vitoria
ENVIARPACOTESDADOS Ter1 Ter4
IMPRIMENETMAP
FIM
```

Os comandos FREQUENCIAOPERADORA, FREQUENCIATERMINAL, ENVIARPACOTESDADOS, e IMPRIMENETMAP deverão imprimir as informações em um dos três arquivos de saída(conforme será especificado adiante), a ser criado pelo seu programa. O nome do arquivo de saída geral deverá ser "*saida.txt"*. Observe que todas as mensagens de erro deverão ser impressas no arquivo "*log.txt"* e que haverá um arquivo *"saida.dot"* que será usado para gerar a imagem do NETMAP (apenas para o comando IMPRIMENETMAP).

Os comandos FREQUENCIAOPERADORA e FREQUENCIATERMINAL imprimem em linhas do arquivo "*saida.txt"* o número de roteadores de uma determinada operadora e o número de terminais de uma determinada localização, respectivamente. O comando ENVIARPACOTESDADOS imprime em uma linha o

comando junto com os dois terminais e SIM se for houver a possibilidade de enviar dados de um determinado terminal para outro e NAO caso contrário.

ex.: `ENVIARPACOTESDADOS Ter1 Ter4: SIM`

O comando IMPRIMENETMAP irá salvar no arquivo *"saida.dot"* o estado atual do NETMAP no momento em que foi pedido, sendo que a cada vez que for pedido para imprimir de novo, deverá ser impresso no mesmo arquivo ".dot", mas separada por um comentário "//intermediario" entre o estado anterior e o novo (vide exemplo para melhor entendimento).

Considerando o arquivo de entrada dado acima, espera-se os seguintes arquivos de saída: *"log.txt"* (possíveis erros ou outras mensagens, separados por uma linha em branco), *"saida.txt"* (com as informações de saida dos comandos, separados por uma linha em branco) e um *"saida.dot"* (código usando a linguagem DOT, e assim usar o GraphViz para gerar uma imagem do NetMap).

Arquivo de saída "log.txt" para o arquivo entrada.txt

```
Erro: Roteador Rot10 inexistente no NetMap

Erro: Terminal Ter9 inexistente no NetMap
```

Arquivo de saída "saida.txt" para o arquivo entrada.txt

```
FREQUENCIAOPERADORA Gvt: 2

FREQUENCIATERMINAL Vitoria: 3

ENVIARPACOTESDADOS Ter1 Ter4: SIM
```

Arquivo de saída "saida.dot" para o arquivo entrada.txt

```
strict graph {
     ter1 -- rot1;
     ter2 -- rot2;
     ter3 -- rot2;
     ter4;
     ter5;
     ter6 -- rot4;
     ter7 -- rot4;
     rot1 -- rot3;
     rot2 -- rot3;
     rot2 -- rot5;
     rot3 -- rot1;
```

```
     rot3 -- rot2;
     rot3 -- rot4;
     rot3 -- rot5;
     rot4 -- rot3;
     rot5 -- rot3;
     rot5 -- rot2;
}
//intermediario

strict graph {
     ter1 -- rot1;
     ter2 -- rot2;
     ter3 -- rot2;
     ter4 -- rot2;
     ter5 -- rot4;
     ter6 -- rot4;
     ter7 -- rot5;
     rot1 -- rot3;
     rot2 -- rot3;
     rot3 -- rot1;
     rot3 -- rot2;
     rot3 -- rot4;
     rot3 -- rot5;
     rot4 -- rot3;
     rot5 -- rot3;
}
```

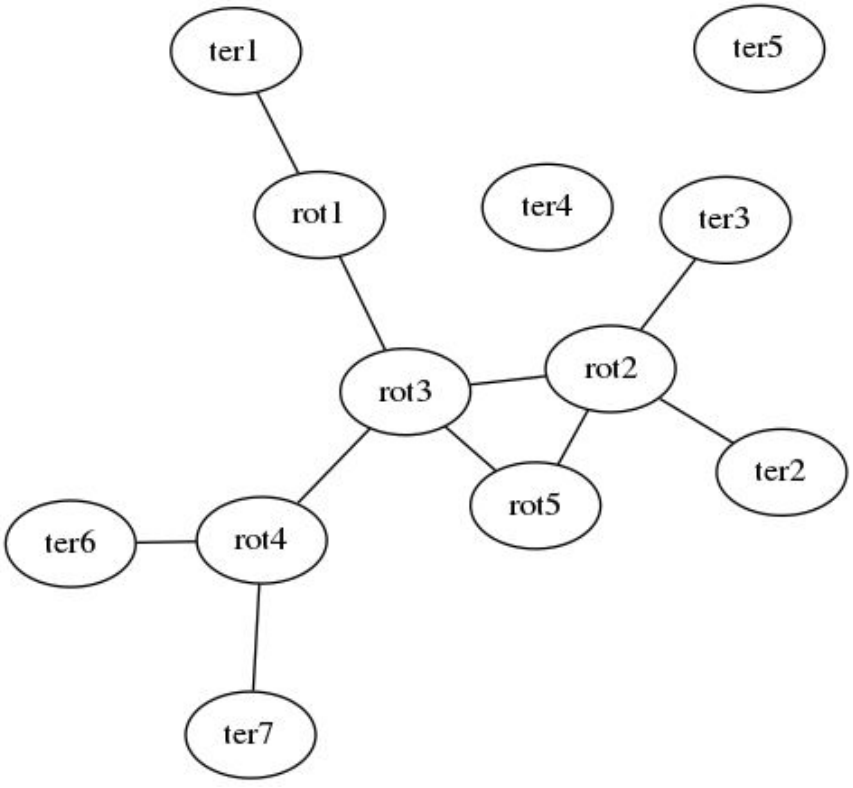

Figura 4 - Exemplo de “grafo” esperado na saída intermediária. (comando neato)

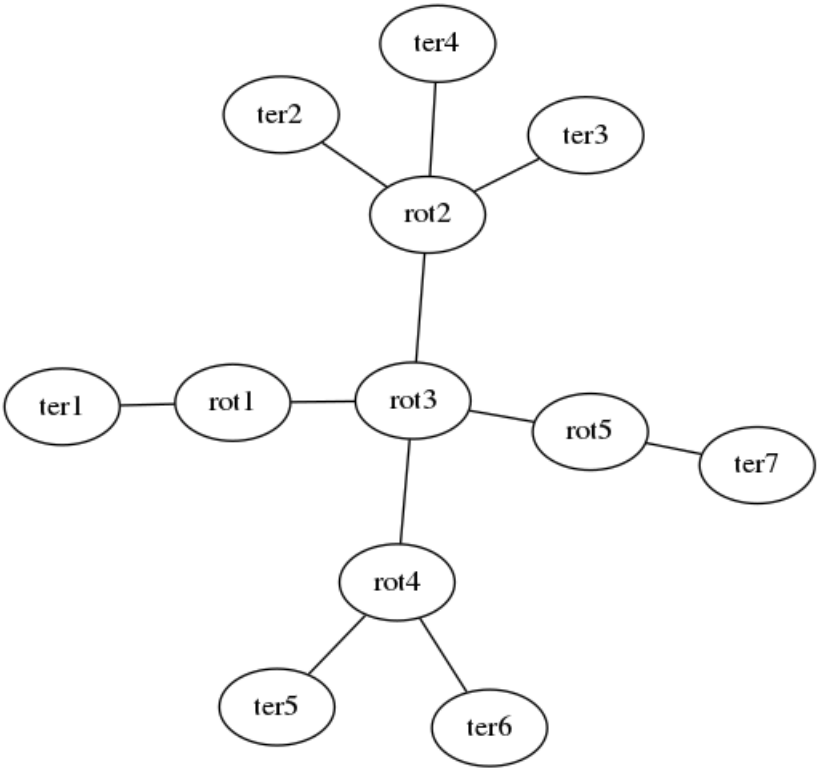

Figura 5 - Exemplo de “grafo” esperado na saída final. (comando neato)

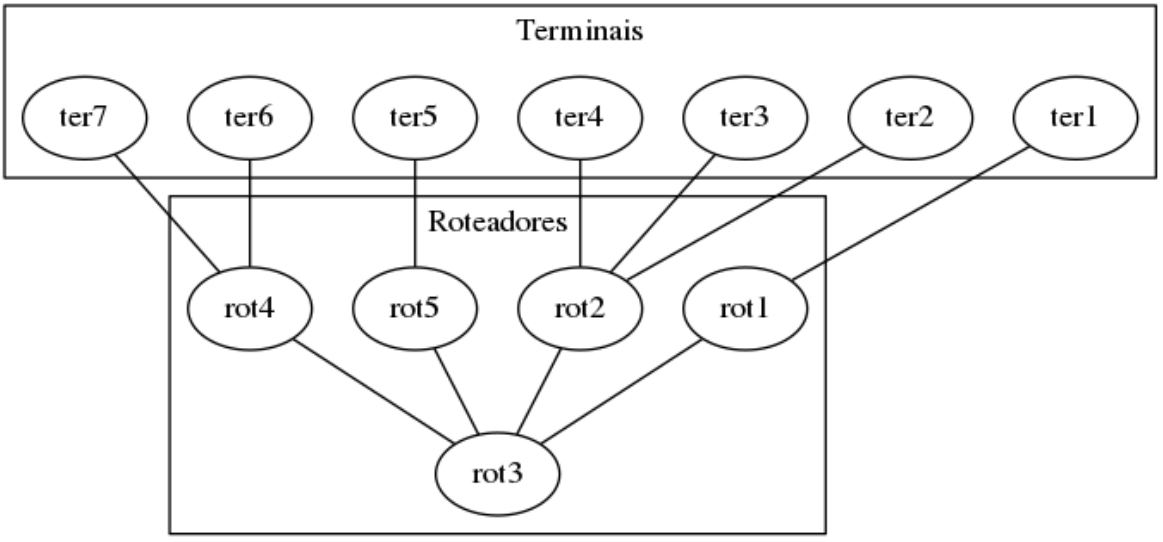

Figura 6- Exemplo de “grafo” feito no Graphviz esperado na saída final.
(usando subgraph cluster e labels no .dot)

3 REGRAS GERAIS

A implementação deverá seguir os conceitos de modularização e abstração (TAD) apresentados em sala. O trabalho terá uma componente subjetiva que será avaliada pelo professor para verificar o grau de uso dos conceitos ensinados.

Portanto, além de funcionar, o código deverá estar bem escrito para que o aluno obtenha nota máxima.

É extremamente recomendado utilizar algum programa para fazer as comparações do resultado final do programa, isto é, os arquivos de saída gerados, poderão ser comparados com o arquivo de saída esperada (fornecido pelo professor) utilizando o comando diff (Ao menos para os arquivos "*saida.txt*" e "*log.txt*"). O meld é uma alternativa gráfica para o diff, se você preferir. Esse programa faz uma comparação linha a linha do conteúdo de 2 arquivos. Diferenças na formatação poderão impossibilitar a correção.

3.1 Regras importantes

O trabalho deverá ser feito em grupo de **2** pessoas (poderá haver exceções caso seja necessário). Cada grupo deverá desenvolver individualmente o trabalho e os alunos deverão necessariamente conhecer e dominar todos os trechos de código criados.

Cada grupo deverá trabalhar independente dos outros não sendo permitido a cópia ou compartilhamento de código. O professor e monitores irão fazer verificação automática de plágio. Trabalhos identificados como iguais, em termos de programação, não serão tolerados e serão penalizados com a nota zero. Isso também inclui o grupo que forneceu o trabalho, sendo portanto, de sua obrigação a proteção de seu trabalho contra cópias ilícitas. Proteja seu trabalho e não esqueça cópias do seu código nas máquinas de uso comum principalmente(ex. laboratório).

3.2 Material a entregar

3.2.1 Em arquivo PDF: Documentação do trabalho (utilizando as normas ABNT Ufes[3]), que deve conter:

- **Capa:** deve conter os seguintes itens: Título, Autoria e Data.
- **Introdução:** descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa (em termos de módulos, arquivos, etc.).

[3] Disponível em: http://repositorio.ufes.br/bitstream/10/1533/1/Normalizacao e apresentacao de trabalhos cientificos e academicos.pdf e também fisicamente na livraria da UFES: http://edufes.ufes.br/items/show/325

- **Implementação:** descrição da implementação do programa. Devem ser detalhadas as estruturas de dados utilizadas (de preferência com diagramas ilustrativos), o funcionamento das principais funções utilizadas incluindo pré e pós condições, o formato de entrada e saída de dados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado. Modularize o seu programa usando a técnica de tipos abstratos de dados, como discutido em sala de aula.
- **Conclusão:** comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
- **Bibliografia:** bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet se for o caso.

3.2.2 Por email (edd20182@gmail.com):

- O assunto da mensagem deve ser ed20182:trab1:<nome1>:<nome2>
  - Por exemplo: ed20182:trab1:<joaosilva>:<mariacosta>
- Documentação do trabalho (em formato PDF, **vide item 3.2.1**).
- Todos os arquivos .c e .h criados (exigido código muito bem documentado!).
- O makefile[4].

3.3 Pontuação

Trabalhos entregues após o prazo perderão nota. O trabalho será pontuado de acordo com sua implementação e os critérios da tabela 1 a seguir. Os pontos descritos na tabela 1 não são independentes entre si, isto é, alguns itens são pré-requisitos para obtenção da pontuação dos outros. Por exemplo, o item “Saídas Corretas” só é válida se também o trabalho estiver compilando e executando. Código com falta de legibilidade e modularização pode perder ponto conforme informado na tabela. Erros gerais de funcionamento, lógica ou **outros** serão descontados como um todo.

[4] Há um pequeno guia em: https://inf.ufes.br/~pdcosta/ensino/2016-2-estruturas-de-dados/material/GuiaRapido_EDI.pdf

Item	Quesitos	Pontos
Trabalho escrito	- Documento PDF de acordo com as normas ABNT, contendo todas as informações sobre o trabalho.	+2
Código e execução	- O Trabalho compilou e executou corretamente.	+2
Conceitos de ED	-TAD e Listas implementadas corretamente.	+3
Resultados	-Saídas corretas foram geradas para cada arquivo de teste.	+3
Legibilidade e Modularização	Pode perder ponto caso não faça: -Uso de comentários -Identação do código -Uso de funções -Uso de tipos de dados definidos pelo usuário e respectivos arquivos (.c e .h)	-3
Vazamento de Memória	-Perde pontos caso esqueça de liberar a memória ou acesso indevido de memória.	-2
Formatação	-Perde pontos caso a saída do programa não respeite a formatação sugerida neste documento.	-2
Atraso	-Perde 1 ponto para cada dia de atraso da entrega.	-1
Plágio	Caso seja constatado plágio, ZERO.	**0**

Tabela 1 - Critérios de avaliação e pontuação.

4 CONSIDERAÇÕES FINAIS

Não utilizar acentos no trabalho. Como referência para a correção do trabalho, o programa deverá compilar e executar nos computadores do LABGRAD. Na seção 2.2, no arquivo de erros "*log.txt*", considere que possa haver erro de leitura ou gravação de arquivo. Segue abaixo um exemplo:

Arquivo de saída "log.txt" para o arquivo entrada.txt

```
Erro: I/O, nao foi possivel abrir o arquivo "entrada.txt"
```

5 ERRATAS

Este documento descreve de maneira geral o trabalho a ser implementado. Este documento poderá sofrer correções caso seja constatado algum erro ou inconsistência. Verifique periodicamente este documento, e procure por marcações deste tipo (em amarelo). Qualquer alteração nas regras do trabalho será comunicada em sala e/ou pelo site. É responsabilidade do aluno frequentar as aulas e se manter atualizado em relação às especificações do trabalho. Caso seja notada qualquer tipo de inconsistência nos arquivos de testes disponibilizados, comunique imediatamente ao professor e/ou monitores para que ela seja corrigida.

BOM TRABALHO!

Have a safe journey in NetMap browsing!

(monitores de E.D. I)