Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Departamento de Ciências de Computação
Disciplina de Estrutura de Dados III (SCC0607)

Docente
Profa. Dra. Cristina Dutra de Aguiar
cdac@icmc.usp.br

Monitores
Caio Capocasali da Silva
caio.capocasali@usp.br *ou telegram: @CaioCapocasali*
Renan Banci Catarin
renanbcatarin@usp.br *ou telegram: @Reckat*
Renan Trofino Silva
renan.trofino@usp.br *ou telegram: @renan823*

Primeiro Trabalho Prático

Este trabalho tem como objetivo aprofundar conceitos relacionados a grafos.

O trabalho deve ser feito por 2 alunos, que são os mesmos alunos do trabalho introdutório. Qualquer mudança deve ser autorizada pela docente. A solução deve ser proposta exclusivamente pelos alunos com base nos conhecimentos adquiridos nas aulas. Consulte as notas de aula e o livro texto quando necessário.

Programa

Descrição Geral. Implemente um programa em C/ C++ por meio do qual o usuário possa obter dados de um arquivo binário de entrada, gerar um grafo direcionado a partir deste e realizar investigações interessantes dentro do contexto de redes sociais.

Importante. A definição da sintaxe de cada comando bem como sua saída devem seguir estritamente as especificações definidas em cada funcionalidade. Para especificar a sintaxe de execução, considere que o programa seja chamado de "programaTrab". Essas orientações devem ser seguidas uma vez que a correção do funcionamento do programa se dará de forma automática. De forma geral, a primeira entrada da entrada padrão é sempre o identificador de suas funcionalidades, conforme especificado a seguir.

Modularização. É importante modularizar o código. Trechos de programa que aparecerem várias vezes devem ser modularizados em funções e procedimentos.

Descrição Específica. O programa deve oferecer as seguintes funcionalidades:

[11] Permita a recuperação dos dados, de todos os registros, armazenados em arquivos de dados no formato binário e a geração de um grafo contendo esses dados na forma de um conjunto de vértices V e um conjunto de arestas A. Os arquivos de dados no formato binário devem seguir os mesmos formatos dos arquivos de dados gerados no trabalho prático introdutório e no primeiro trabalho prático. O grafo deve ser um grafo direcionado e deve representar quem segue quem (ou seja, qual pessoa segue qual pessoa).

A representação do grafo deve ser, obrigatoriamente, na forma de listas de adjacências. As listas de adjacências consistem tradicionalmente em um vetor de |V| elementos que são capazes de apontar, cada um, para uma lista linear, de forma que o i-ésimo elemento do vetor aponta para a lista linear de arestas que são adjacentes ao vértice i. Cada elemento do vetor deve representar o **nomeUsuario de uma pessoa que segue.** Os vértices do vetor devem ser ordenados de forma crescente de acordo com o nomeUsuario. De acordo com a especificação do trabalho prático introdutório, o campo *nomeUsuario* não pode assumir valores nulos ou valores repetidos. Cada elemento da lista linear deve armazenar o rótulo do vértice (ou seja, o **nomeUsuario de uma pessoa que é seguida**), a data de início que a pessoa que segue começou a seguir a pessoa que é seguida, a data de fim que a pessoa que segue começou a seguir a pessoa que é seguida e o grau de amizade. Os elementos de cada lista linear devem ser ordenados de forma crescente de acordo com o nome do usuário da pessoa que é seguida. Se houver empate, a ordenação deve ser feita de forma crescente de acordo com a data de início.

Entrada do programa para a funcionalidade [11]:

`11 arquivoPessoa.bin arquivoIndexaPessoa.bin arquivoSegueOrdenado.bin`

onde:

- `arquivoPessoa.bin` é o arquivo de dados **pessoa** no formato binário, o qual é gerado de acordo com as especificações da **Descrição do Arquivo Pessoa** definidas no trabalho prático introdutório.
- `arquivoIndexaPessoa.bin` é o arquivo de índice primário que indexa o arquivo de dados **pessoa**, o qual é gerado de acordo com as especificações da **Descrição do Arquivo de Índice Primário** definidas no trabalho prático introdutório.
- `arquivoSegueOrdenado.bin` é o arquivo de dados **segue** no formato binário gerado pela funcionalidade [9] conforme as especificações descritas no primeiro trabalho prático, o qual encontra-se ordenado de acordo com o campo *idPessoaQueSegue*.

Saída caso o programa seja executado com sucesso:

Para cada elemento i do vetor ($1 \leq i \leq n$), exiba cada elemento j ($1 \leq j \leq m$) da lista linear correspondente da seguinte forma. Exiba o elemento i do vetor. Na mesma linha, exiba o elemento j da lista linear correspondente. Mude de linha. Adicionalmente, pule uma linha quando for exibir o próximo elemento i do vetor. Deve haver uma vírgula e um espaço em branco entre cada saída mostrada na saída padrão. Valores nulos devem ser representados por NULO.

Exemplo com metadados:

$nomeUsuarioQueSegue_1$, $nomeUsuarioQueESeguida_{11}$, $dataInicioQueSegue_{11}$, $dataFimQueSegue_{11}$, $grauAmizade_{11}$
$nomeUsuarioQueSegue_1$, $nomeUsuarioQueESeguida_{12}$, $dataInicioQueSegue_{12}$, $dataFimQueSegue_{12}$, $grauAmizade_{12}$
...
$nomePessoaQueSegue_1$, $nomeUsuarioQueESeguida_{1m}$, $dataInicioQueSegue_{1m}$, $dataFimQueSegue_{1m}$, $grauAmizade_{1m}$
...
$nomeUsuarioQueSegue_n$, $nomeUsuarioQueESeguida_{n1}$, $dataInicioQueSegue_{n1}$, $dataFimQueSegue_{n1}$, $grauAmizade_{n1}$
$nomeUsuarioQueSegue_n$, $nomeUsuarioQueESeguida_{n2}$, $dataInicioQueSegue_{n2}$, $dataFimQueSegue_{n2}$, $grauAmizade_{n2}$
...
$nomePessoaQueSegue_n$, $nomeUsuarioQueESeguida_{nm}$, $dataInicioQueSegue_{nm}$, $dataFimQueSegue_{nm}$, $grauAmizade_{nm}$

Mensagem de saída caso algum erro seja encontrado:

```
Falha na execução da funcionalidade.
```

Exemplo de execução (são mostrados apenas alguns elementos)**:**

```
./programaTrab
11 arquivoPessoa.bin arquivoIndexaPessoa.bin arquivoSegueOrdenado.bin
PIETRA6, JULIA5, 15/06/2025, NULO, 2
PIETRA6, LUCASCARDOSO, 11/04/2024, NULO, 2
PULE UMA LINHA
SCONCEICAO, ARTHUROLIVEIRA, 28/09/2022, NULO, 1
SCONCEICAO, MARIA60, 11/08/2021, NULO, 1
SCONCEICAO, MAYA83, 23/04/2022, 04/03/2024, 2
PULE UMA LINHA
```

[12] Gere o grafo transposto do grafo gerado na funcionalidade [11]. Um grafo transposto $G^T = (V, A^T)$ de um grafo direcionado $G = (V, A)$ é um grafo que contém todos os vértices de G e cujas arestas são definidas da seguinte forma: $A^T = \{ (u, v) : (v, u) \in A \}$, ou seja, A^T consiste das arestas de G com suas direções invertidas. A representação do grafo deve, obrigatoriamente, ser na forma de listas de adjacências.

As listas de adjacências consistem tradicionalmente em um vetor de |V| elementos que são capazes de apontar, cada um, para uma lista linear, de forma que o i-ésimo elemento do vetor aponta para a lista linear de arestas que são adjacentes ao vértice i. Cada elemento do vetor deve representar o **nomeUsuario de uma pessoa que é seguida**. Os vértices do vetor devem ser ordenados de forma crescente de acordo com o nomeUsuario. De acordo com a especificação do trabalho prático introdutório, o campo *nomeUsuario* não pode assumir valores nulos ou valores repetidos. Cada elemento da lista linear deve armazenar o rótulo do vértice (ou seja, o **nomeUsuario de uma pessoa que segue**), a data de início que a pessoa começou a ser seguida, a data de fim que a pessoa parou de ser seguida e o grau de amizade. Os elementos de cada lista linear devem ser ordenados de forma crescente de acordo com o nome do usuário da pessoa que segue. Se houver empate, a ordenação deve ser feita de forma crescente de acordo com a data de início.

Entrada do programa para a funcionalidade [12]:

`12 arquivoPessoa.bin arquivoIndexaPessoa.bin arquivoSegueOrdenado.bin`

onde:

- `arquivoPessoa.bin` é o arquivo de dados **pessoa** no formato binário, o qual é gerado de acordo com as especificações da **Descrição do Arquivo Pessoa** definidas no trabalho prático introdutório.
- `arquivoIndexaPessoa.bin` é o arquivo de índice primário que indexa o arquivo de dados **pessoa**, o qual é gerado de acordo com as especificações da **Descrição do Arquivo de Índice Primário** definidas no trabalho prático introdutório.
- `arquivoSegueOrdenado.bin` é o arquivo de dados **segue** no formato binário gerado pela funcionalidade [9] conforme as especificações descritas no primeiro trabalho prático, o qual encontra-se ordenado de acordo com o campo *idPessoaQueSegue*.

Saída caso o programa seja executado com sucesso:

Para cada elemento i do vetor ($1 \leq i \leq n$), exiba cada elemento j ($1 \leq j \leq m$) da lista linear correspondente da seguinte forma. Exiba o elemento i do vetor. Na mesma linha, exiba o elemento j da lista linear correspondente. Mude de linha. Adicionalmente, pule uma linha quando for exibir o próximo elemento i do vetor. Deve haver uma vírgula e um espaço em branco entre cada saída mostrada na saída padrão. Valores nulos devem ser representados por NULO.

Exemplo com metadados:

$nomeUsuarioQueESeguido_1$, $nomeUsuarioQueSegue_{11}$, $dataInicioQueSegue_{11}$, $dataFimQueSegue_{11}$, $grauAmizade_{11}$
$nomeUsuarioQueESeguido_1$, $nomeUsuarioQueSegue_{12}$, $dataInicioQueSegue_{12}$, $dataFimQueSegue_{12}$, $grauAmizade_{12}$
...
$nomeUsuarioQueESeguido_1$, $nomeUsuarioQueSegue_{1m}$, $dataInicioQueSegue_{1m}$, $dataFimQueSegue_{1m}$, $grauAmizade_{1m}$
...
$nomeUsuarioQueESeguido_n$, $nomeUsuarioQueSegue_{n1}$, $dataInicioQueSegue_{n1}$, $dataFimQueSegue_{n1}$, $grauAmizade_{n1}$
$nomeUsuarioQueESeguido_n$, $nomeUsuarioQueSegue_{n2}$, $dataInicioQueSegue_{n2}$, $dataFimQueSegue_{n2}$, $grauAmizade_{n2}$
...
$nomeUsuarioQueESeguido_n$, $nomeUsuarioQueSegue_{nm}$, $dataInicioQueSegue_{nm}$, $dataFimQueSegue_{nm}$, $grauAmizade_{nm}$

Mensagem de saída caso algum erro seja encontrado:

```
Falha na execução da funcionalidade.
```

Exemplo de execução (são mostrados apenas alguns elementos)**:**

```
./programaTrab
12 arquivoPessoa.bin arquivoIndexaPessoa.bin arquivoSegueOrdenado.bin
ARTHUROLIVEIRA, CAUAMOURA, 29/11/2021, NULO, 2
ARTHUROLIVEIRA, RYANMENDES, 19/07/2025, NULO, 0
ARTHUROLIVEIRA, SCONCEICAO, 28/09/2022, NULO, 1
PULE UMA LINHA
MARIA60, AMONTEIRO, 22/10/2022, NULO, 0
MARIA60, ENZO7, 08/10/2025, NULO, 1
MARIA60, ISABELA5, 24/01/2023, NULO, 1
MARIA60, LARISSACASTRO, 06/12/2021, NULO, NULO
PULE UMA LINHA
```

[13] Determine o caminho mais curto entre cada **pessoa que segue** e uma determinada **celebridade** definida como parâmetro de entrada. Deve ser listado na saída padrão o caminho mais curto, de forma que sejam exibidas somente as respostas referentes às pessoas que ainda seguem as celebridades. Considere que, durante a execução do algoritmo, qualquer decisão a respeito da ordem de inserção ou análise dos vértices/arestas deve ser feita considerando a ordem alfabética: primeiro ordena pelo nomeUsuario; em caso de empate, ordena pela data de início; em caso de empate, ordena pela data de fim.

Entrada do programa para a funcionalidade [13]:

`13 arquivoPessoa.bin arquivoIndexaPessoa.bin arquivoSegueOrdenado.bin "nomeUsuarioDaCelebridade"`

onde:

- `arquivoPessoa.bin` é o arquivo de dados **pessoa** no formato binário, o qual é gerado de acordo com as especificações da **Descrição do Arquivo Pessoa** definidas no trabalho prático introdutório.
- `arquivoIndexaPessoa.bin` é o arquivo de índice primário que indexa o arquivo de dados **pessoa**, o qual é gerado de acordo com as especificações da **Descrição do Arquivo de Índice Primário** definidas no trabalho prático introdutório.
- `arquivoSegueOrdenado.bin` é o arquivo de dados **segue** no formato binário gerado pela funcionalidade [9] conforme as especificações descritas no primeiro trabalho prático, o qual encontra-se ordenado de acordo com o campo *idPessoaQueSegue*.
- `nomeUsuarioDaCelebridade` é o nome de usuário da celebridade passado como parâmetro. Ele é representado entre aspas duplas (") por ser do tipo *string*.

Saída caso o programa seja executado com sucesso:

`Para cada caminho mais curto i (1 ≤ i ≤ n), exiba os dados de cada aresta j presente neste caminho da seguinte forma. Escreva em uma linha específica o nome do usuário da pessoa que segue, o nome do usuário da pessoa que é seguida, a data de início, a data de fim e o grau de amizade. Deve haver uma vírgula e um espaço em branco entre cada saída mostrada na saída padrão. Valores nulos devem ser representados por NULO. Note que o último nome do usuário da pessoa que é seguida de cada caminho mais curto deve ser o nomeUsuarioDaCelebridade passado como parâmetro. Adicionalmente, pule uma linha quando for exibir o próximo caminho mais curto i. Caso a pessoa que segue não seguir a celebridade, então deve ser escrito "NAO SEGUE A CELEBRIDADE".`

<u>Exemplo com metadados</u>:

$nomeUsuarioQueSegue_1$, $nomeUsuarioQueESeguida_{11}$, $dataInicioQueSegue_{11}$, $dataFimQueSegue_{11}$, $grauAmizade_{11}$

$nomeUsuarioQueSegue_1$, $nomeUsuarioQueESeguida_{12}$, $dataInicioQueSegue_{12}$, $dataFimQueSegue_{12}$, $grauAmizade_{12}$

...

nomePessoaQueSegue$_1$, nomeUsuarioDaCelebridade, dataInicioQueSegue$_{1m}$, dataFimQueSegue$_{1m}$, grauAmizade$_{1m}$
...
nomeUsuarioQueSegue$_n$, nomeUsuarioQueESeguida$_{n1}$, dataInicioQueSegue$_{n1}$, dataFimQueSegue$_{n1}$, grauAmizade$_{n1}$
nomeUsuarioQueSegue$_n$, nomeUsuarioQueESeguida$_{n2}$, dataInicioQueSegue$_{n2}$, dataFimQueSegue$_{n2}$, grauAmizade$_{n2}$
...
nomePessoaQueSegue$_n$, nomeUsuarioDaCelebridade, dataInicioQueSegue$_{nm}$, dataFimQueSegue$_{nm}$, grauAmizade$_{nm}$

Mensagem de saída caso algum erro seja encontrado:

```
Falha na execução da funcionalidade.
```

Exemplo de execução (são mostrados apenas alguns elementos)**:**

```
./programaTrab
13 arquivoPessoa.bin arquivoIndexaPessoa.bin arquivoSegueOrdenado.bin
"ARTHUROLIVEIRA"
EVELYN7, SCONCEICAO, 07/08/2020, NULO, 2
SCONCEICAO, ARTHUROLIVEIRA, 28/09/2022, NULO, 1
PULE UMA LINHA
STEPHANY6, MARIAH8, 31/05/2021, NULO, NULO
MARIAH8, NINA18, 11/10/2013, NULO, NULO
NINA18, ARTHUROLIVEIRA, 03/02/2024, NULO, 1
PULE UMA LINHA
```

[14] Determine o comprimento do caminho para que uma determinada pessoa definida como parâmetro de entrada ouça a fofoca que ela mesmo contou. Considere que, durante a execução do algoritmo, qualquer decisão a respeito da ordem de inserção ou análise dos vértices/arestas deve ser feita considerando a ordem alfabética: primeiro ordena pelo nomeUsuario; em caso de empate, ordena pela data de início; em caso de empate, ordena pela data de fim. Outro aspecto é que, mesmo que existam mais do que um caminho relacionado à pessoa definida como parâmetro de entrada, apenas a resposta relativa ao primeiro caminho deve ser listada na saída padrão.

Entrada do programa para a funcionalidade [14]:

```
14 arquivoPessoa.bin arquivoIndexaPessoa.bin arquivoSegueOrdenado.bin "nomeUsuarioQueGerouFofoca"
```

onde:

- `arquivoPessoa.bin` é o arquivo de dados **pessoa** no formato binário, o qual é gerado de acordo com as especificações da **Descrição do Arquivo Pessoa** definidas no trabalho prático introdutório.

- `arquivoIndexaPessoa.bin` é o arquivo de índice primário que indexa o arquivo de dados **pessoa**, o qual é gerado de acordo com as especificações da **Descrição do Arquivo de Índice Primário** definidas no trabalho prático introdutório.

- `arquivoSegueOrdenado.bin` é o arquivo de dados **segue** no formato binário gerado pela funcionalidade [9] conforme as especificações descritas no primeiro trabalho prático, o qual encontra-se ordenado de acordo com o campo *idPessoaQueSegue*.

- `nomeUsuarioQueGerouFofoca` é o nome de usuário da pessoa que gerou a fofoca e que é passado como parâmetro. Ele é representado entre aspas duplas (") por ser do tipo *string*.

Saída caso o programa seja executado com sucesso:

```
A saída deve ser exibida na saída padrão da seguinte forma. Deve ser exibido o comprimento do caminho caso a fofoca retorne para a pessoa ou "A FOFOCA NAO RETORNOU".
```

Mensagem de saída caso algum erro seja encontrado:

```
Falha na execução da funcionalidade.
```

Exemplo de execução (são mostrados apenas alguns elementos)**:**

```
./programaTrab
14 arquivoPessoa.bin arquivoIndexaPessoa.bin arquivoSegueOrdenado.bin "ARTHUROLIVEIRA"
5

./programaTrab
14 arquivoPessoa.bin arquivoIndexaPessoa.bin arquivoSegueOrdenado.bin "STEPHANY6"
A FOFOCA NAO RETORNOU
```

Restrições

As seguintes restrições têm que ser garantidas no desenvolvimento do trabalho.

[1] O arquivo de dados deve ser gravado em disco no **modo binário**. O modo texto não pode ser usado.

[2] Os dados do registro descrevem os nomes dos campos, os quais não podem ser alterados. Ademais, todos os campos devem estar presentes na implementação, e nenhum campo adicional pode ser incluído. O tamanho e a ordem de cada campo deve obrigatoriamente seguir a especificação.

[3] Deve haver a manipulação de valores nulos, conforme as instruções definidas.

[4] Não é necessário realizar o tratamento de truncamento de dados.

[5] Devem ser exibidos avisos ou mensagens de erro de acordo com a especificação de cada funcionalidade.

[6] Os dados devem ser obrigatoriamente escritos campo a campo. Ou seja, não é possível escrever os dados registro a registro. Essa restrição refere-se à entrada/saída, ou seja, à forma como os dados são escritos no arquivo.

[7] O(s) aluno(s) que desenvolveu(desenvolveram) o trabalho prático deve(m) constar como comentário no início do código (i.e. NUSP e nome do aluno). Para trabalhos desenvolvidos por mais do que um aluno, não será atribuída nota ao aluno cujos dados não constarem no código fonte.

[8] Todo código fonte deve ser documentado. A **documentação interna** inclui, dentre outros, a documentação de procedimentos, de funções, de variáveis, de partes do

código fonte que realizam tarefas específicas. Ou seja, o código fonte deve ser documentado tanto em nível de rotinas quanto em nível de variáveis e blocos funcionais.

[9] A implementação deve ser realizada usando as linguagens de programação C e C++. As funções das bibliotecas <stdio.h> devem ser utilizadas para operações relacionadas à escrita e leitura dos arquivos. A implementação não pode ser feita em qualquer outra linguagem de programação. O programa executará no [run.codes].

Material para Entregar

Arquivo compactado. Deve ser preparado um arquivo .zip contendo:

- Código fonte do programa devidamente documentado.
- Makefile para a compilação do programa.

Vídeo gravado.

- Um vídeo gravado pelos integrantes do grupo, o qual deve ter, no máximo, 5 minutos de gravação. O vídeo deve explicar o trabalho desenvolvido. Ou seja, o grupo deve apresentar: cada funcionalidade e uma breve descrição de como a funcionalidade foi implementada. Todos os integrantes do grupo devem participar do vídeo, sendo que o tempo de apresentação dos integrantes deve ser balanceado. Ou seja, o tempo de participação de cada integrante deve ser aproximadamente o mesmo. O uso da webcam é obrigatório.

Instruções para fazer o arquivo makefile. No [run.codes] tem uma orientação para que, no makefile, a diretiva “all” contenha apenas o comando para compilar seu programa e, na diretiva “run”, apenas o comando para executá-lo. Adicionalmente, para utilizar a função binarioNaTela, é necessário usar a flag -lmd. Assim, a forma mais simples de se fazer o arquivo makefile é:

```
all:
	gcc -o programaTrab *.c -lmd
run:
```

```
./programaTrab
```

Lembrando que *.c já engloba todos os arquivos .c presentes no arquivo zip. Adicionalmente, no arquivo Makefile é importante se ter um *tab* nos locais colocados acima, senão ele pode não funcionar.

Instruções de entrega.

O programa deve ser submetido via [run.codes]:

- página: https://runcodes.icmc.usp.br/
- Código de matrícula: **3YE6**

O vídeo gravado deve ser submetido por meio da página da disciplina no e-disciplinas, no qual o grupo vai informar o nome de cada integrante, o número do grupo e um link que contém o vídeo gravado. Ao submeter o link, verifique se o mesmo pode ser acessado. Vídeos cujos links não puderem ser acessados receberão nota zero. Vídeos corrompidos ou que não puderem ser corretamente acessados receberão nota zero.

Critério de Correção

Critério de avaliação do trabalho. Na correção do trabalho, serão ponderados os seguintes aspectos.

- Corretude da execução do programa.
- Atendimento às especificações do registro de cabeçalho e dos registros de dados.
- Atendimento às especificações da sintaxe dos comandos de cada funcionalidade e do formato de saída da execução de cada funcionalidade.
- Qualidade da documentação entregue. A documentação interna terá um peso considerável no trabalho.
- Vídeo. Integrantes que não participarem da apresentação receberão nota 0 no trabalho correspondente.

Casos de teste no [run.codes]. Juntamente com a especificação do trabalho, serão disponibilizados 70% dos casos de teste no [run.codes], para que os alunos possam avaliar o programa sendo desenvolvido. Os 30% restantes dos casos de teste serão utilizados nas correções.

Restrições adicionais sobre o critério de correção.

- O uso excessivo de ferramentas de inteligência artificial acarretará a atribuição de nota zero (0) ao trabalho.
- A não execução de um programa devido a erros de compilação implica que a nota final da parte do trabalho será igual a zero (0).
- O não atendimento às especificações do registro de cabeçalho e dos registros de dados implica que haverá uma diminuição expressiva na nota do trabalho.
- O não atendimento às especificações de sintaxe dos comandos de cada funcionalidade e do formato de saída da execução de cada funcionalidade implica que haverá uma diminuição expressiva na nota do trabalho.
- A ausência da documentação implica que haverá uma diminuição expressiva na nota do trabalho.
- A realização do trabalho prático com alunos de turmas diferentes implica que haverá uma diminuição expressiva na nota do trabalho.
- A inserção de palavras ofensivas nos arquivos e em qualquer outro material entregue implica que a nota final da parte do trabalho será igual a zero (0).
- Em caso de plágio, as notas dos trabalhos envolvidos serão zero (0).

Bom Trabalho!