Modelagem e solução de um problema usando grafos (versão: 11/9/23)

Disciplina: Algoritmos em grafos
Professor: Vinícius Dias (viniciusdias@ufla.br)

- Esta especificação pode ser alterada ao longo do semestre, mas sempre no sentido de detalhar os requisitos. De toda forma, se isso acontecer, eu aviso.
- Se tiver alguma dúvida ou quiser discutir alguma ideia, me procure.

1. Introdução

Este trabalho tem o objetivo de consolidar conceitos teóricos e aspectos práticos relacionados à solução de problemas computacionais usando grafos. Esta atividade é individual e consiste no desenvolvimento de um mini-projeto que envolva a modelagem e solução de um problema usando grafos e sobre uma base de dados (grafo) extraído de aplicações reais. A seguir uma especificação detalhada do trabalho, leia com atenção.

2. O que deve ser entregue

1. **Relatório (1 arquivo `relatorio.pdf`):** Você deve submeter uma documentação de até 3 páginas com o seguinte conteúdo:

   Cabeçalho. Nome do estudante, número de matrícula, nome da disciplina, código da disciplina e nome do professor.

   Definição e apresentação da base de dados: Você deve escolher uma base de dados real que é (ou que pode ser) modelada usando grafos. O seu relatório deve explicar todos os aspectos pertinentes a essa fonte de dados com relação ao trabalho, por exemplo: O que o grafo representa, isto é, o que são os vértices e arestas? Quais as características desse grafo? Ele é não-direcionado ou direcionado? Ele é simples ou não? Ele possui atributos ou não? Se sim, quais? Caracterize o grafo também quantitativamente: quantos vértices, quantas arestas, qual é o grau médio dos vértices?

   Eu vou deixar aqui duas referências onde vocês podem buscar essas bases de dados reais que são modeladas usando grafos: (i) SNAP-datasets em <http://snap.stanford.edu/data/index.html> e (ii) Network-repository em <https://networkrepository.com/>. Você pode usar outras fontes desde que a metodologia de coleta ou referência esteja devidamente atendida. Faz parte da atividade consultar, ler, entender e apresentar a base de dados escolhida.

   Limitação do escopo e definição do problema: Depois de definido grafo, você deve limitar o escopo do que você pretende estudar/resolver sobre o grafo e definir o problema de forma clara e objetiva. Lembre-se que, em um problema bem definido, a entrada e a saída esperada estão sempre bem especificadas (inclusive em termos de formato). É importante também incluir exemplos que ilustrem a sua proposta, para que o objetivo da implementação fique claro.

   Sua solução usando algoritmos: Aqui você deve mostrar a metodologia usada para resolver o problema: tratamento dos dados, algoritmos utilizados, eventuais modificações que precisaram ser feitas em algoritmos clássicos, etc. Veja que esta etapa **não envolve nenhum tipo de descrição de implementação como códigos em linguagens de programação.** Sua descrição aqui é em alto nível, abusando de pseudocódigos, diagramas, descrições textuais, etc.

Ferramenta/software desenvolvido: Dada a descrição em alto nível da solução acima, descreva os aspectos práticos que possibilitaram a implementação da solução descrita acima: linguagem de programação utilizada, arquitetura (como os elementos do software se comunicam?), instruções de compilação e uso, etc.

Resultados: Utilize a sua implementação e mostre alguns resultados produzidos por ela, indicando sempre como interpretar os resultados e qual é a relavância do resultado produzido.

Limitações: Nesta última seção liste e discuta as limitações do seu trabalho.

Guia para um bom relatório: utilize ilustrações (afinal, uma das vantagens de grafos é sua capacidade de representação gráfica), utilize pseudocódigos para descrever algoritmos, sempre caracterize os algoritmos utilizados do ponto de vista das operações realizadas (ordem de crescimento assintótico), forneça exemplos de como o problema escolhido pode ser relevante, seja sucinto e objetivo na descrição do seu software (deixe detalhes como trechos de código para o repositório com o código), faça uma análise inteligente e crítica dos resultados (não basta apresentar, é preciso indicar como o resultado deve ser interpretado), estude como os parâmetros de entrada (tamanho do grafo e outros parâmetros) afetam o tempo de execução da ferramenta sempre que for pertinente.

2. **Implementação (arquivos fonte, entrada e makefile):**

   Linguagem. Implemente usando sua linguagem de preferência, desde que as bibliotecas padrão das linguagens sejam utilizadas e não seja necessário o download de uma grande quantidade de dependências para funcionar. Certifique-se de que o trabalho pode ser facilmente reproduzido.

   Ambiente. Os testes serão executados em Linux/UNIX. Portanto, garanta que seu código compila e roda corretamente nesse sistema operacional. A melhor forma de garantir que seu trabalho rode em Linux é escrever e testar o código nele. Você pode também usar o subsistema do Windows para Linux (WSL)[1], que é uma forma rápida com menos instalações. Você também pode fazer o download de uma variante de Linux como o Ubuntu[2] e instalá-lo em seu computador ou diretamente ou por meio de uma máquina virtual como o VirtualBox [3]. Há vários tutoriais sobre como instalar Linux disponíveis na Internet.

   Makefile. Esse arquivo é obrigatório e usado para compilar/preparar (`$ make`) e executar (`$ make run`) o seu trabalho. Uma referência introdutória sobre como usar *makefiles* para compilação pode ser encontrada no rodapé[4].

   Amostra do grafo. Forneça uma amostra do grafo original (isto é, um subgrafo) apenas para efeito de teste. Este arquivo TXT deve acompanhar a submissão (veja abaixo).

   Link para o grafo completo. Como o grafo completo pode ser expressivo e inviável de ser adicionado em um repositório e submetido no Campus Virtual. Você deve apenas informar em um arquivo TXT o link para o grafo completo, isto é, onde ele pode ser obtido. Lembre-se de que a permissão de acesso para esse recurso precisa ser **pública**.

   Qualidade do Código. Seu código deve ser bem escrito: (1) dê nomes a variáveis, funções e estruturas que façam sentido; (2) divida a implementação em módulos que tenham um significado bem definido; (3) acrescente comentários sempre que julgar apropriado; (4) não é necessário parafrasear o código, mas é interessante acrescentar descrições de alto nível que ajudem outras

[1] <https://learn.microsoft.com/pt-br/windows/wsl/install>
[2] <http://www.ubuntu.com>
[3] <https://www.virtualbox.org>
[4] <https://bit.ly/2PrgIJk>

pessoas a entender como sua implementação funciona; (5) evite utilizar variáveis globais; (6) mantenha as funções concisas: seres humanos não são muito bons em manter uma grande quantidade de informações na memória ao mesmo tempo. Funções muito grandes, portanto, são mais difíceis de entender; (7) lembre-se de indentar o código: escolha uma forma de indentar (tabulações ou espaços) e mantenha-se fiel a ela, misturar duas formas de indentação pode fazer com que o código fique ilegível quando você abri-lo em um editor de texto diferente do que foi utilizado originalmente. (8) evite linhas de código muito longas. Nem todo mundo tem um monitor tão grande quanto o seu. Uma convenção comum adotada em vários projetos é não passar de 80 caracteres de largura.

3. Como deve ser entregue

O trabalho deve ser entregue no Campus Virtual da disciplina na forma de um único arquivo zipado (formato `.zip`) contendo documentação, implementação e makefile organizadas em um diretório chamado "mini-projeto-grafos":

```
mini-projeto-grafos/
  |- relatorio.pdf                    # relatório com o conteúdo acima
  |- src/                             # arquivos fonte
  |- makefile                         # define como usar o software
  |- amostra-grafo.txt                # amostra reduzida (até 1KB) do grafo
  |- link-para-grafo-completo.txt     # referência para o grafo completo
```

Você deve compactar esse diretório e seu conteúdo em um arquivo único chamado `mini-projeto-grafos-<nome>-<sobrenome>.zip`. Arquivos em outros formatos (RAR, por exemplo) que não `.zip` não serão aceitos. Por exemplo, para a Fulana da Silva: `mini-projeto-grafos-fulana-silva.zip`.

4. Avaliação

Este trabalho será avaliado de acordo com os seguintes critérios:

- O estudante seguiu a especificação do trabalho: formatos dos arquivos, nome dos arquivos, linguagem de programação, entrada e saída, makefile, etc.;
- Qualidade da documentação: completude, texto bem escrito e formatado, metodologia e análise de resultados bem feitas, etc.;
- Qualidade da implementação: código indentado, bem comentado, variáveis legíveis, código modularizado, correto, alocação dinâmica correta, etc.

5. Observações gerais

- Leia esta especificação com cuidado;
- Essa especificação não é isenta de erros e ambiguidades. Portanto, se tiverem problemas para entender o que está escrito aqui: pergunte!
- Comece o trabalho o quanto antes;
- Apenas um arquivo deve ser entregue, compactado e no formato `.zip`;
- Seu trabalho deve ser compatível com ambiente Linux;

- **Seja honesto.** Você não aprende nada copiando código de terceiros nem pedindo a outra pessoa que faça o trabalho por você. Se a cópia for detectada, sua nota será zerada e o colegiado será devidamente informado para que providências possam ser tomadas.