UNIVERSIDADE FEDERAL ALFENAS (UNIFAL)
Bacharelado em Ciência da Computação

| Disciplina | Método de realização | Data de apresentação |
|---|---|---|
| DCE529 - AEDS 3 | Presencial | 22/10/2024 às 8h00 |

Professor
Iago Augusto de Carvalho (iago.carvalho@unifal-mg.edu.br)

## Trabalho prático 3 - Personagens mais centrais

O objetivo deste terceiro trabalho é utilizar algoritmos em grafos para resolver um problema de centralidade em uma rede complexa construída a partir dos livros de Game of Thrones.

Uma rede complexa pode ser entendida como um grande grafo sem uma estrutura definida, cujas conexões não podem ser definidas de uma forma simples. Os vértices destas redes podem ser categorizados de acordo com sua importância relativa. Diz-se que quanto mais importante é um vértice, maior é sua centralidade.

Existem diversas métricas de centralidade de vértices em redes complexas, como o *betweeness*, o *closeness*, o *eigenvector*, o índice de centralidade de *Katz* e a centralidade harmônica, dentre outros. Neste trabalho, estamos interessados no índece *betweeness*.

O *betweeness* quantifica o número de vezes que um vértice faz parte do caminho mínimo entre outros dois vértices de um grafo. Quanto mais vezes este vértice é utilizado nos caminhos mínimos de outros vértices, maior é o seu índice *betweeness*. Matematicamente, o valor de *betweeness* $B_v$ de um vértice $v \in V$ pode ser definido como

$$B_v = \sum_{s \neq v \neq t \in V} \frac{\delta_{st}(v)}{\delta_{st}},$$

onde $\delta_{st}$ é o número total de caminhos mínimos no grafo e $\delta_{st}(v)$ é o número destes caminhos que passam pelo vértice $v$. Note que $\delta_{st}$ é igual ao número de pares de vértices no grafo e $\delta_{st}(v)$ não considera o número de caminhos iniciados ou finalizados em $v$.

---

**Construção da rede**: A rede deve ser desenvolvida a partir dos livros de Game of Thrones disponíveis aqui neste repositório. São disponibilizados todos os **5** livros existentes até o momento e a rede deve ser construída utilizando um deles como base.

Esta rede será definida como um grafo não-direcionado com pesos. Cada vértice desta rede será um personagem do livro. Além disso, o peso da aresta será igual a força do relacionamento entre dois personagens diferentes.

A força do relacionamento entre dois personagens, neste caso, será medida por quantas vezes seus nomes aparecem próximos uns dos outros no livro. Para este trabalho, vocês devem considerar uma distância de até **50 palavras**, tanto para frente quanto para trás. Deste modo, quanto mais vezes o nome de dois personagens aparecem próximos um do outro no livro, mais forte será seu relacionamento.

Para a construção da rede, deve-se considerar um total de 10 diferentes personagens do livro, a escolher entre os 15 abaixo:

| | | | | |
|---|---|---|---|---|
| Arya | Bran | Brienne | Catelyn | Cersei |
| Daenerys | Jaime | Melisandre | Petyr | Robert |
| Sam | Sansa | Theon | Tyrion | Varys |

Ao fim, é esperado que vocês tenham uma rede parecida com a rede da Figura 1.

---

**O que deve ser desenvolvido**: Neste trabalho cada grupo deverá implementar um algoritmo para ler **um dos livros** disponibilizados e montar a rede complexa de relacionamento entre os personagens conforme descrito acima.

Como o *betweeness* utiliza o caminho mais curto, deve-se, ao fim da construção da rede, alterar o peso de suas arestas. Então, o grupo deverá desenvolver um algoritmo que inverta os custos das arestas, isto é, este

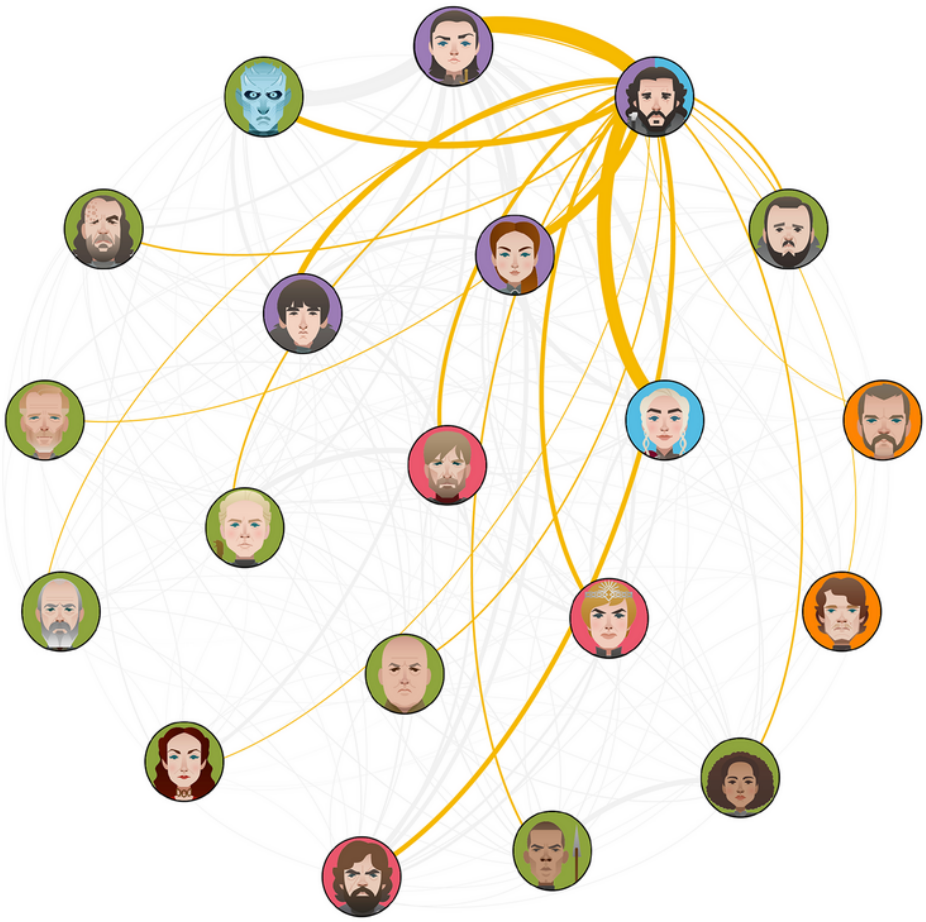

Figura 1: Exemplo de rede de resultado. Quanto mais grossa a aresta, maior é seu peso

algoritmo fará com que a aresta de maior peso torne-se a aresta de menor peso. De forma similar, a aresta de menor peso deverá passar a ser a de maior peso. Este algoritmo deverá manter a proporcionalidade entre os pesos das arestas.

Por fim, deverá ser realizado o cálculo de *betweeness* para calcular, em ordem, quais são os personagens mais centrais do livro de acordo com esta métrica. Para isto, deverão ser implementados:

- Uma estrutura de dados para grafos
  - Matriz de adjacência ou lista de adjacência
- Um algoritmo de caminho mínimo
- O cálculo de *betweeness*

**O que deve ser entregue**: Cada grupo deverá desenvolver um documento *.pdf* contendo as seguintes sessões:

1. Introdução (introduzir e definir o problema de centralidade em grafos)
2. Algoritmos (descrever os algoritmos utilizados e analisar sua complexidade)
3. Resultados (exibir a centralidade de cada um dos vértices do grafo, qual é o vértice mais central e uma **figura** do grafo mostrando os vértices, sua topologia e sua centralidade)
4. Descrição do Makefile e instruções de compilação e execução dos códigos desenvolvidos. A descrição do Makefile só é necessária para códigos desenvolvidos em C ou C++

A figura da rede deve ser semelhante Figura 1. Entretanto, não é necessário desenhar o rosto dos personagens.

Além disso, os grupos também deverão montar uma apresentação de slides (também em formato *.pdf*) para apresentação em sala de aula no dias 22/10, sendo que a apresentação deverá durar entre 7 e 12 minutos.

Por fim, deverá ser entregue o código desenvolvido. Os algoritmo poderão ser implementados utilizando C, C++ ou python. O código deverá ser entregue em um único arquivo *.zip* contendo um cabeçalho com o nome dos integrantes do grupo.

**Método de entrega**: Todos os três arquivos deverão ser entregues no Moodle da disciplina até as 8h00 do dia 22/10.

**Método de avaliação**: A apresentação corresponderá por 30% da nota total. De forma complementar, o outro documento *.pdf* corresponderá também por 30% da nota e o código corresponderá por 40% da nota.

Na apresentação, serão avaliados:

- Adequação ao tempo
- Postura dos apresentadores
- Assertividade na fala
- Corretude do conteúdo apresentado
- Uso correto da língua portuguesa
- Qualidade dos slides

No documento *.pdf* com a descrição do problema, dos algoritmos e os resultados, serão avaliados:

- Uso correto da língua portuguesa
- Qualidade e clareza na apresentação dos algoritmos
- Análise correta das complexidades dos algoritmos
- Qualidade geral da comparação dos resultados
- Adequação do relatório aos templates disponibilizados

No código serão avaliados:

- A qualidade e clareza do código
- Comentários explicativos
- Execução correta dos algoritmos
- Adequação do Makefile proposto (caso desenvolvido em C ou C++)