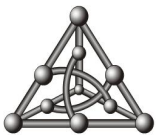


UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

Implementação Algorítmica

Atividade 5 — Detecção de redes de relacionamento

1 Descrição

Aplicativos de relacionamento usam grafos para modelar redes de interesse mútuo entre seus usuários. Nesses grafos, o conjunto dos vértices representam as pessoas da rede e é dividido em duas partes. As arestas, por sua vez, sempre ligam um vértice de uma parte a um vértice da outra parte. Dessa forma, uma aresta indica que as duas pessoas, representadas pelos vértices ligados as suas extremidades, têm interesse uma na outra. Ou seja, o grafo que representa uma rede de relacionamentos deve ser necessariamente **bipartido**. Assim, se $G = (V, E)$ é um grafo bipartido, então existem subconjuntos disjuntos V_1 e V_2 de V tais que $V = V_1 \cup V_2$ e toda aresta $e = uv$ é tal que $u \in V_1$ e $v \in V_2$ ou vice-versa. Ou seja, G é bipartido com partição dos vértices V_1 e V_2.

Uma carcterística importante de uma rede de relacionamentos é que ela não contém ciclos de comprimento ímpar. Um **ciclo** é uma sequência de vértices $v_1 v_2 \cdots v_k v_1$, tal que $v_i v_{i+1}$ é uma aresta de G, para todo $i = 1, \ldots, k-1$ e $v_k v_1$ também é uma aresta de G. Dizemos que um ciclo é **ímpar** se seu número de arestas (ou vértices) é ímpar. No exemplo anterior, $v_1 v_2 \cdots v_k v_1$ é um ciclo de comprimento ímpar, ou simplesmente ciclo ímpar, se k é ímpar.

Nesta atividade você deve gerar grafos com arestas aleatórias e verificar, para cada um deles, se eles podem representar uma rede de relacionamentos. Observe, conforme mencionado acima, que uma das formas é verificar se o grafo em questão não contém um ciclo ímpar.

2 Programa, entrada e saída

Você deve desenvolver e implementar os algoritmos conforme a descrição da seção 1.

Você deve construir conjuntos de dados de entrada da seguinte forma. Primeiro, determine um número de vértices n para um grafo. Por exemplo, o número de vértices pode variar como $n = 10, 20, 30, \ldots, 990, 1000$. Para cada valor n do número de vértices de um grafo G, você precisa sortear arestas e adicioná-las ao conjunto de arestas de G. Para todos os pares de vértices u, v em V, com $1 \leq u, v \leq n$, você deve usar algum processo aleatório para determinar se a aresta uv pertence ou não ao grafo, com alguma probabilidade. Você pode usar uma moeda confiável e assim um par de vértices u, v vai possuir uma aresta que os conecta com probabilidade $1/2$. Apesar dessa estratégia ser correta, ela produz grafos muito densos, com muitas arestas. Então talvez seja melhor usar uma moeda tendenciosa que adiciona uma aresta ligando u, v com probabilidade menor, $1/100$ por exemplo (o que significa que a probabilidade da aresta não ser sorteada é $99/100$). Depois de escolhida uma probabilidade que será usada em todos os experimentos, você deve mostrá-la na primeira linha da saída.

Dado que um grafo com n vértices, mais as arestas sorteadas, foi construído (usando listas de adjacências), você deve computar e imprimir o número de vértices do grafo, seu número de arestas e informar se o grafo é ou não bipartido.

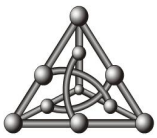
2.1 Exemplo de entrada e saída

Um exemplo para casos de teste com $10 \leq n \leq 300$, é mostrado a seguir. Na primeira linha, mostramos qual probabilidade foi usada para sortear arestas dos grafos. Depois, uma linha apresentando o rótulo para o número de vértices do grafo (**V**), o número de arestas (**E**) e para a classificação do grafo (**bipartido**). Depois disso, as informações de cada grafo do conjunto de testes é impressa: seu número de vértices, seu número de arestas e se é ou não bipartido.

Observe que o primeiro grafo do exemplo abaixo tem 10 vértices e não tem aresta alguma. Ainda assim, por definição, esse grafo é bipartido.

```
Probabilidade de sortear uma aresta: 0.01

   V        E      bipartido
----------------------------
  10        0      sim
  20        1      sim
  30        6      sim
  40        2      sim
  50       12      sim
  60       15      sim
  70       25      sim
  80       27      sim
  90       42      sim
 100       47      sim
 110       51      sim
 120       80      não
 130       93      não
 140       86      sim
 150      109      não
 160      134      não
 170      159      sim
 180      160      não
 190      176      não
 200      210      sim
 210      209      não
 220      241      não
 230      273      sim
 240      282      não
 250      300      não
 260      334      não
 270      372      não
 280      402      não
 290      404      não
 300      436      não
```



3 Entrega

Instruções para entrega da sua atividade:

1. **O que entregar?**

   Um arquivo compactado a ser entregue deve conter o seguinte:

   - programa(s) desenvolvido(s) (e um arquivo **`Makefile`**, se for o caso),
   - tabela com as informações dos grafos gerados e testados (em texto), de acordo com a formatação apresentada na seção 2.1.

   Compacte todos esses arquivos com o compactador de sua preferência e entregue um único arquivo (com extensão **`.tgz`**, **`.bz2`**, **`.zip`**, **`.rar`**, ...).

2. **Forma de entrega**

   A entrega será realizada diretamente no Sistema (AVA/UFMS), na disciplina de Implementação Algorítmica – T01. Um fórum de discussão deste trabalho já se encontra aberto. Após abrir uma sessão digitando seu *login* e sua senha, vá até a sessão "Atividades" e escolha "Entrega da atividade 5". Você pode entregar a atividade quantas vezes quiser até às **23 horas e 59 minutos** do dia **21 de novembro de 2022**. A última versão entregue é aquela que será corrigida. Encerrado o prazo, não serão mais aceitos trabalhos.

3. **Linguagem de programação**

   Use a linguagem de programação de sua preferência para desenvolver esta atividade.

4. **Atrasos**

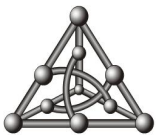
   Trabalhos atrasados não serão aceitos. Não deixe para entregar seu trabalho na última hora. Para prevenir imprevistos como queda de energia, problemas com o sistema, e/ou falha de conexão com a internet, sugerimos que a entrega do trabalho seja feita pelo menos um dia antes do prazo determinado.

5. **Erros**

   Trabalhos com erros de compilação/interpretação receberão nota ZERO. Faça todos os testes necessários para garantir que seu programa está livre de erros de compilação/interpretação.

6. **Arquivo com o programa fonte**

   Seu(s) arquivo(s) contendo o(s) fonte(s) do(s) programa(s) na linguagem escolhida deve(m) estar bem organizado(s). Um programa tem de ser muito bem compreendido por uma pessoa. Verifique se seu programa tem a indentação adequada, se não tem linhas muito longas, se tem variáveis com nomes significativos, entre outros. Não esqueça que um programa bem descrito e bem organizado é a chave de seu sucesso. Também não esqueça da documentação de seu programa.



7. **Conduta Ética**

   O trabalho deve ser feito INDIVIDUALMENTE/COM SEU GRUPO. Cada estudante tem responsabilidade sobre cópias de seu trabalho, mesmo que parciais. Não compartilhe seu programa ou trechos de seu programa. Você pode consultar seus colegas para esclarecer dúvidas e discutir idéias sobre o trabalho, ao vivo ou no fórum de discussão da disciplina, mas NÃO copie o programa!

   Trabalhos considerados plagiados terão nota ZERO.