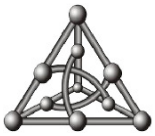


UNIVERSIDADE FEDERAL DE MATO GROSSO DO SUL
Faculdade de Computação

Implementação Algorítmica

Atividade 3 — Grafos

1 Descrição

Grafos são estruturas de dados amplamente usadas em computação e algoritmos que trabalham com grafos são fundamentais na área. Percurso ou busca em um grafo significa visitar sistematicamente as arestas do grafo, bem como os seus vértices. Um algoritmo que realiza uma busca em um grafo pode descobrir muito sobre sua estrutura. Muitos algoritmos começam por realizar uma busca em seu grafo de entrada com o objetivo de obter essa informação estrutural. Outros algoritmos são descritos como reelaborações simples de algoritmos que realizam buscas em grafos. Técnicas de busca em grafos são o coração da área de algoritmos para grafos.

Seja $G = (V, E)$ um grafo sem orientação nas arestas. Seja v um vértice em G. Dizemos que o **grau** de v em G é o número de vizinhos que v possui em G, ou seja, é o número de arestas que incidem no vértice v. Por exemplo, se v é vizinho de x, y e z em G, isto é, se as arestas vx, vy e vz pertencem a E, então o grau de v é 3.

Sejam s e v vértices de um grafo G. Um caminho de menor comprimento de s a v em G é chamado um **menor caminho** ou **caminho mais curto** de s a v. Lembre-se que o algoritmo BFS encontra corretamente o caminho mais curto a partir de um vértice origem s em V para cada vértice v em V. Dentre todos os caminhos mais curtos a partir de s em G, aquele que tem maior comprimento determina o **diâmetro** do grafo G. Assim, o diâmetro é o comprimento do maior caminho mais curto com origem em s no grafo G.

Nesta atividade, você tem de implementar um algoritmo que gera grafos e extrai algumas propriedades de cada um deles.

2 Programa, entrada e saída

Você deve construir conjuntos de dados de entrada da seguinte forma. Primeiro, determine um número de vértices n para um grafo $G(V, E)$. Por exemplo, o número de vértices pode variar como $n = 10, 20, 30, \ldots, 990, 1000$. Dessa maneira, $V = \{v_0, v_1, \ldots, v_{n-1}\}$. Para todos os pares de vértices u, v em V, onde $u \neq v$, você deve usar um processo aleatório para determinar se a aresta uv pertence ou não ao conjunto E, com alguma probabilidade. Uma maneira seria 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. Por isso você deverá usar uma moeda tendenciosa que adiciona uma aresta ligando u, v com probabilidade p menor, por exemplo, $p = 1/4$ (o que significa que a probabilidade da aresta não ser sorteada é $3/4$).

Dado que um grafo com n vértices, mais as arestas sorteadas, foi construído, você deve computar e imprimir o número de vértices (`V`) do grafo, o número de arestas (`E`) do grafo, seu grau

mínimo (**gmin**), seu grau máximo (**gmax**), a média dos graus (**gmed**) dos vértices do grafo e o seu diâmetro (**diam**). Considere sempre a origem $s = v_0$.

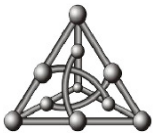
2.1 Exemplo de entrada e saída

A entrada do seu algoritmo consiste de quatro valores:

- ini: um número inteiro positivo que indica o número de vértices inicial para fazer os experimentos;
- fim: um número inteiro positivo maior que ini indicando o número de vértices do último experimento;
- stp: um número inteiro positivo, indicando o *step* (incremento) entre um experimento e o próximo;
- p: um número real tal que $0 < p \leq 0.25$ indicando a probabilidade de incluir uma aresta no grafo.

Um exemplo de execução para a entrada ini $= 10$, fim $= 200$, stp $= 10$ e $p = 0.1$ é mostrado a seguir:

V	E	gmin	gmax	gmed	diam
10	2	0	1	0.4	0
20	19	0	5	1.9	7
30	44	0	6	2.9	5
40	74	0	9	3.7	4
50	113	1	9	4.5	4
60	172	1	11	5.7	4
70	240	1	14	6.9	4
80	304	2	15	7.6	4
90	397	3	17	8.8	4
100	514	5	19	10.3	3
110	603	4	21	11.0	3
120	788	5	21	13.1	3
130	844	6	25	13.0	3
140	988	5	28	14.1	3
150	1085	7	26	14.5	3
160	1295	6	27	16.2	3
170	1384	7	28	16.3	3
180	1644	8	30	18.3	3
190	1813	7	28	19.1	3
200	2051	9	31	20.5	3



3 Entrega

1. **O que entregar?**

   - O código-fonte. Se o seu programa consiste de vários arquivos-fonte, compacte-os com o compactador de sua preferência e entregue um único arquivo (com extensão **`.tgz`**, **`.bz2`**, **`.zip`**, **`.rar`**, ...);
   - Um arquivo `readme.txt` com instruções para executar o seu programa em um ambiente Linux.

2. **Forma de entrega**

   A entrega será realizada diretamente no Sistema (AVA/UFMS), na disciplina de Implementação Algorítmica – T01. Você pode entregar a atividade quantas vezes quiser até às **23 horas e 59 minutos** do dia **26 de novembro de 2024**. A última versão entregue é aquela que será corrigida. Encerrado o prazo, trabalhos não serão mais aceitos.

   NÃO HAVERÁ PRORROGAÇÃO DE PRAZO.

3. **Linguagem de programação**

   O programa deve ser implementado em uma das seguintes linguagens: C/C++, Python ou Java.

4. **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.

5. **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.

6. **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.