SCC-503 - Algoritmos e Estruturas de Dados II

Professor: Robson L. F. Cordeiro – robson@icmc.usp.br
Estagiário PAE: Didier Vega – davo@icmc.usp.br
Monitor: Gustavo Smarito – gusmarito@grad.icmc.usp.br

Projeto 1 – Grafos

O programa a ser desenvolvido neste projeto deve seguir rigorosamente os formatos de entrada e saída definidos, uma vez que todos os projetos serão submetidos a um **corretor automático**.

Implemente sua atividade com seus colegas de grupo sem compartilhar, olhar código dos outros grupos, ou buscar na Internet. Procure usar apenas os conceitos vistos nas aulas. Os critérios de correção dos exercícios são:

1. (60%) O programa funciona corretamente nas tarefas descritas para todos os casos de teste e processamento correto das entradas e saídas;
2. (20%) Qualidade da solução desenvolvida, estruturação do código e implementação do TAD, bom uso das técnicas de programação e eficiência da solução em termos de espaço e tempo;
3. (10%) Boa endentação e uso de comentários no código, além de boa estruturação e modularização;
4. (10%) Documentação externa: relatório curto de tipo **readme.txt** explicando as técnicas utilizadas para solucionar cada sub-problema.

O programa deve ser desenvolvido em no máximo **grupos de três (3) alunos**. Quaisquer programas similares terão **nota zero**, independente de qual for o original e qual for a cópia. Podem ser utilizadas as linguagens de programação C ou C++. **Data de entrega: 04/04/2014** via Tidia-ae (Atividades)**.**

Problema 1: Localização de Hospital

Gotham é uma cidade com um alto índice de acidentes de trânsito e conflitos nas ruas. Isto tem elevado à demanda por um bom hospital de urgências para a cidade. O prefeito de Gotham quer resolver esse problema. Para isso, foi contratado um especialista em redes complexas das empresas Wayne para assessorar na localização do hospital.

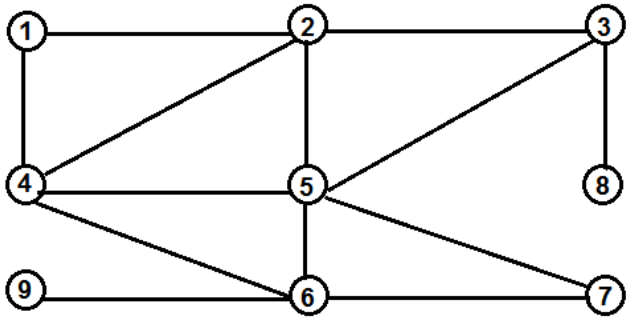

Figura 1: Exemplo de representação da cidade: os vértices correspondem aos quarteirões e as arestas às vias que conectam esses quarteirões

Suponha que a cidade é modelada por meio de um grafo G = (V, A), conforme ilustrado na Figura1, onde cada vértice representa um quarteirão e as arestas representam as ruas que conectam esses quarteirões. O problema consiste em determinar um local apropriado para o hospital (no caso, um quarteirão) cuja maior distância de todos os caminhos mínimos para os demais vértices seja a menor possível. Em outras palavras, para cada vértice $\boldsymbol{u}$ calculamos $\mathbf{E}(\boldsymbol{u})$, o tamanho do maior caminho mínimo de $\boldsymbol{u}$ aos demais vértices. Essa medida é conhecida como *Eccentricity*[1]. Após calcular $\mathbf{E}(\boldsymbol{u})$ para todos os vértices, escolhemos o vértice (ou vértices em caso de empate) com o menor valor de $\mathbf{E}(\boldsymbol{u})$ do grafo, obtendo assim o vértice central (*Central Vertex*[2]).

Desse modo, o hospital deverá estar localizado no vértice central do cálculo anterior. Dado o grafo da cidade, o prefeito quer que sejam calculados o(s) vértice(s) central(is) do grafo onde deveria ser construído o hospital da cidade.

Problema 2: Engarrafamento de vias

Além disso, o prefeito de Gotham sabe que precisa ampliar algumas das vias congestionadas da cidade, assim como foi feito em Metrópolis. Desse modo, melhora-se o trânsito na cidade, evitam-se mais acidentes e permite-se que a polícia possa perseguir de modo eficaz os capangas do Curinga. Para isso, foi utilizada uma medida que permite identificar as vias com maior carga ou concentração de fluxo de informação nos grafos. A medida consiste em calcular o número de caminhos mínimos que passam por cada aresta do grafo, sendo uma variação da conhecida medida *Betweenness centrality*[3]. De modo prático, definimos neste trabalho a carga $\boldsymbol{\beta}(\boldsymbol{u_{ij}})$ como o número de caminhos mínimos entre todos os possíveis pares de vértices **(s, t)** que passam pela aresta $\boldsymbol{u_{ij}}$. As arestas com maior carga representam as vias chaves com maior engarrafamento e devem, portanto, ser ampliadas ou recapeadas.

[1] http://en.wikipedia.org/wiki/Eccentricity_(graph_theory)

[2] http://en.wikipedia.org/wiki/Eccentricity_(graph_theory)

[3] http://en.wikipedia.org/wiki/Betweenness_centrality

Dado o grafo da cidade, o prefeito quer encontrar as X vias em que deveriam ser feitas as obras de ampliação, usando a medida de carga anteriormente explicada.

Resolução

Escreva um programa que recebe por entrada padrão o nome do arquivo a ser lido. O arquivo do grafo sempre estará localizado na mesma pasta do executável. O programa deverá apresentar o(s) quarteirão(ões)/vértice(s) onde será construído o hospital da cidade. Em seguida, por entrada padrão, recebe um número inteiro X que indica quantas vias o prefeito quer ampliar/arrumar. O programa apresentará por saída padrão as X vias selecionadas.
Utilize como estrutura para armazenar o grafo uma **lista de adjacência: lista ligada ou arranjo**.
No caso de empates: no problema 1, apresente os empates em ordem crescente de índice; no problema 2, apresente todos em ordem crescente de índice.

Entrada de dados

O programa receberá por entrada padrão o nome de um arquivo de texto, localizado na mesma pasta, e o número X de ruas a serem ampliadas. O arquivo contém a lista de arestas do grafo no formato:

<n Vértices>\n
<m Arestas>\n
<vi_1 [espaço] vj_1>\n
...
<vi_M [espaço] vj_M>

Para o grafo da Figura1, um exemplo de arquivo seria:

```
9
13
1 2
2 3
2 4
3 5
4 1
4 5
4 6
5 6
5 7
2 5
6 7
9 6
3 8
```

Importante: Neste projeto serão considerados apenas grafos **conexos não direcionados**. Assim, para o exemplo anterior temos que a entrada da aresta 1 2 implica que também exista 2 1.

Arquivo de saída

O arquivo apresentará o vértice (ou vértices no caso de empate) onde deveria ser localizado o hospital e também a seleção de X arestas (vias) a serem ampliadas.

<vi[espaço]vj **...** >\n /* vértice(s) onde deveria ser localizado o hospital */
<u_i[espaço]u_j >\n (1)
<u_k[espaço]u_l>\n (2)
...
<u_y[espaço]u_z>\n (X)
/* as X vias pedidas pelo problema 2 */

Por exemplo, para o arquivo anterior com X = 2, temos a seguinte saída:

5
3 5
3 8
6 9

ATENÇÃO

- Não deverá ser utilizada qualquer variável global. Também, serão **desconsiderados** os trabalhos que utilizam como estrutura de dados uma Matriz (de adjacência, incidência ou representação estrela).

- Não poderão ser utilizadas bibliotecas com funções prontas (a não ser aquelas para entrada, saída e alocação dinâmica de memória).

- Todos os arquivos do projeto (Makefile, arquivos .h, arquivos .c, etc.) deverão ser armazenados no mesmo diretório. Esse diretório deverá ser compactado com extensão .zip, com o nome ***NUSP_projeto1.zip***, onde NUSP é o seu número USP. No arquivo **readme.txt** deve-se colocar os nomes e números USP dos integrantes do grupo

- Dúvidas conceituais deverão ser colocadas nos horários de atendimento.

Referências

- M.E.J Newman, **Networks An Introduction**. Oxford University Press. New York, 2010.
- Sholmo Havlin and Reuven Cohen, **Complex Networks: structure, robustness and function**. Cambridge University Press. New York, 2010.
- Barrat, M. Barthelemy and A. Vespignani, **Dynamical Processes on Complex Networks**. Cambridge University Press. New York, 2008.