Universidade Federal de Ouro Preto – UFOP
Instituto de Ciências Exatas e Biológicas – ICEB
Departamento de Computação – DECOM
Disciplina: Teoria dos Grafos
Professor: Marco Antonio M. Carvalho

Trabalho Prático – O Problema do Brigadista em Grafos

Avaliação: Código-fonte comentado, resultados corretos para instâncias disponibilizadas, desempenho (vide tabela de estratos).

Estrato	Desempenho	Nota
1	Melhor	110%
2	10%	100%
3	10%-15%	85%
4	15%-25%	75%
5	25%-35%	70%
6	35%-50%	65%
7	>50%	30%

Pontos extras: 10% de acréscimo para o código de melhor desempenho;
Data de entrega: 27 de novembro, até as 10:00 via *e-mail*. Trabalhos atrasados ou incompletos não serão aceitos.

Referência

O documento abaixo foi utilizado como base para elaboração deste trabalho, incluindo enunciado e instâncias.

RAMOS, Natanael. Um estudo computacional do problema do brigadista em grafos. 2018. 1 recurso online (82 p.). Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação, Campinas, SP. Disponível em: <http://www.repositorio.unicamp.br/handle/REPOSIP/331841>. Acesso em: 18 fev. 2018.

Enunciado

Este trabalho consiste na implementação de um método para solução do problema de **Caminho do Brigadista em Grafos** (ou *Firefighter Problem*, FFP). Este é um modelo determinístico e em tempo discreto para simular a propagação e contenção de incêndios em grafos. Informalmente, este problema consiste em ajudar os brigadistas por meio da determinação de uma estratégia de contenção de incêndios na qual o dano causado seja minimizado, especialmente dado que o número de brigadistas disponíveis é limitado.

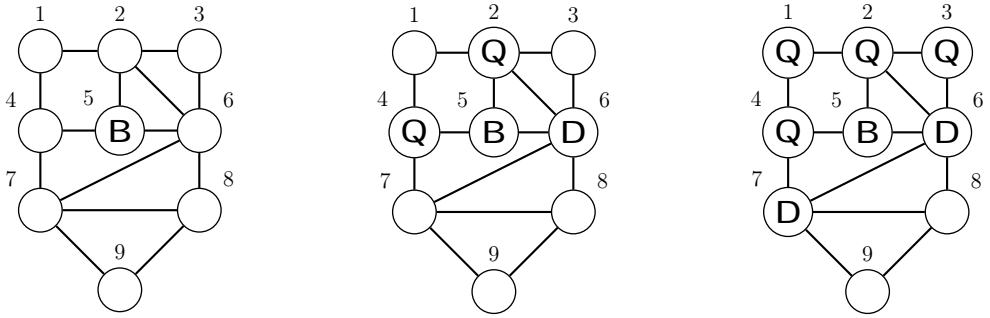
Formalmente, são dados um inteiro D representando a quantidade de brigadistas disponíveis, um grafo não direcionado e não ponderado $G = (V, E)$ e um subconjunto de vértices $B \subseteq V$, que representam os focos de incêndio. Então, inicia-se nos elementos de B um processo iterativo de propagação e contenção de fogo através dos vértices de G, em rodadas discretas, o qual termina quando não existem mais vértices que possam ser queimados ou defendidos. Idealmente, quando o fogo está contido.

O objetivo ao resolver o FFP é maximizar o número de vértices não queimados quando o fogo é contido, com a restrição de que no máximo D vértices podem ser protegidos contra o fogo por rodada. Aplicações práticas do FFP, além da obtenção de estratégias para minimização de danos causados por incêndios, podem ser encontradas em áreas como no espalhamento de vírus em computador, na propagação de informação em

uma rede social e também na disseminação de doenças entre populações

Durante o processo de contenção, cada um dos vértices de *G* pode estar em um de três estados: **intocado**, **queimado** ou **defendido**. Inicialmente, todos os vértices estão intocados. Então, na primeira rodada, os vértices de *B*, denominados focos de incêndio, passam para o estado de queimados. Em cada uma das rodadas subsequentes, cada um dos *D* brigadistas disponíveis pode escolher um vértice intocado para defender. Note que um brigadista pode escolher qualquer vértice intocado para defender, não havendo a necessidade de "trafegar" pelas arestas do grafo. Em seguida, o fogo se propaga de vértices queimados para seus adjacentes intocados.

A Figura 1 apresenta um exemplo do processo iterativo que constitui a resolução de uma instância do FFP, considerando apenas um brigadista disponível (i.e., *D*=1). Vértices rotulados por *Q* ou *B*, no caso de focos de incêndio, estão queimados. Vértices rotulados com *D* estão defendidos e aqueles sem rótulos estão intocados. O número próximo de cada vértice é o seu identificador único. Na Figura 1(a) somente o vértice 5 está queimado, pois é um foco de incêndio. Na próxima rodada, Figura 1(b), o vértice 6 é defendido pelo único brigadista disponível e os demais vértices rotulados por *Q* são queimados, uma vez que não foram defendidos e são vizinhos do vértice 5, queimado na rodada anterior. O processo termina na Figura 1(c) visto que o fogo foi contido, nesse caso, por uma estratégia em que foram defendidos o vértice 6 na rodada 1 e o 7 na rodada 2. Nesse exemplo, a solução ótima foi obtida, com quatro vértices salvos (i.e. não queimados): 6, 7, 8 e 9.



(a) Rodada 0 (b) Rodada 1 (c) Rodada 2

Formato da entrada

São disponibilizadas 10 instâncias para testes preliminares. Um conjunto de testes surpresa poderá ser utilizado na hora da apresentação do trabalho. Cada arquivo possui o seguinte formato:

- Número de vértices (*n*)
- Número de arestas (*m*)
- Número de brigadistas (*D*)
- Número de focos de incêndio (|*B*|)
- Índices dos elementos do conjunto *B*
- Para cada aresta, o vértice de origem e o vértice de destino.

Exemplo

```
9
14
1
1
5
1 2
1 4
2 3
```

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

Formato da Saída

Os programas devem imprimir ao final de sua execução, o número de vértices queimados e, para cada vértice defendido, uma linha com o número do vértice e a rodada em que ele foi defendido.

Abordagens

Exceto pelas restrições indicadas na seção abaixo, serão aceitas quaisquer abordagens, ficando a avaliação condicionada à apresentação do trabalho, em que os alunos deverão demonstrar domínio sobre a abordagem empregada. Também não será aceita a implementação do método descrito no material que serve de referência para este trabalho.

Restrições

Não são permitidos recursos computacionais como programação paralela e distribuída, entretanto, a escolha da linguagem de programação e a utilização de outros recursos da própria linguagem são livres.

Não são permitidos recursos de programação matemática (Programação Linear, Programação Inteira, etc.).

O algoritmo deve ser determinístico, ou seja, a resposta para uma determinada instância deve ser sempre a mesma, além de o código não “chutar” a resposta. Algoritmos não determinísticos serão desconsiderados.

Cada algoritmo poderá executar por até 2 minutos. Algoritmos mais rápidos com resultados iguais ou melhores serão beneficiados na avaliação.

Entregáveis

Código fonte criado (um único arquivo, sem bibliotecas ou projetos), e relatório com os resultados obtidos para as instâncias. Não é necessário incluir no relatório a descrição do método ou código fonte.

Máquinas para Testes

Serão tomadas como referência as máquinas do laboratório de ensino COM30 no sistema operacional Ubuntu. Portanto, os resultados reportados devem ser referentes àqueles computadores.

Como Medir o Tempo?

Em C++, adicione o trecho de código abaixo para medir o tempo. Recomenda-se medir o tempo depois da leitura dos dados e após o término da geração da solução.

```
#include <chrono>
using namespace std::chrono;

int main(){

  duration<double> time_span2;
```

```
  high_resolution_clock::time_point t1 = high_resolution_clock::now();
  //gerar a solução aqui
  high_resolution_clock::time_point t2 = high_resolution_clock::now();
  duration<double> time_span = duration_cast<duration<double> >(t2 - t1);

  return 0;
}
```