# Trabalho Prático 1
# O problema de recomendação de amigos

**Luís Eduardo Oliveira Lizardo**

[1]Projeto e Análise de Algoritmos
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais (UFMG)

lizardo@dcc.ufmg.br

## 1. Introdução

Recomendação de novos amigos é um problema comum em redes sociais, pois permite aos usuários expandirem as suas relações de amizade na rede e também a aumentarem as suas interações. Neste trabalho as amizades dos usuários de uma rede social são representadas por um grafo não direcionado, onde os vértices são os usuários e as arestas indicam se há uma relação de amizade entre eles.

O objetivo deste trabalho é (1) determinar se todos os usuários da rede social estão conectados, seja através de uma conexão direta ou indiretamente através de outros amigos, e (2) calcular um subgrafo de conexões cuja função de qualidade $r$ seja máxima. Esta função de qualidade representa a configuração de amizade ideal para a rede social e é dada por

$$r = \frac{\sum_{i=1}^{E} Friendship_i}{\sum_{i=1}^{E} Distance_i}, \forall i \in E$$

onde *Friendship* e *Distance* são métricas que quantificam as relações de amizades dos usuários e indicam, respectivamente, o nível de amizade entre dois usuários e uma suposta distância física entre eles.

O restante deste relatório está organizado da seguinte forma: a Seção 2 explica dois algoritmos utilizados na solução. A Seção 3 descreve a solução do problema e faz a análise de complexidade do programa. A Seção 4 apresenta a avaliação experimental do trabalho e avalia a solução do problema em relação ao tempo de execução. A Seção 5 conclui o trabalho.

## 2. Referencial Teórico

Neste trabalho foram utilizados dois algoritmos clássicos de teoria dos grafos para solucionar o problema: o algoritmo de Busca em Largura (BFS do inglês Breadth-First Search) e o algoritmo de Prim para calcular a Árvore Geradora Mínima/Máxima. As subseções seguintes darão uma breve explicação sobre os algoritmos, mas mais detalhes podem ser encontrados em [Cormen et al. 2001].

### 2.1. Busca em Largura (BFS)

Este é um algoritmo de busca em grafos que expande e examina sistematicamente todos os vértices. O BFS começa por um vértice e explora todos os vértices vizinhos a este. Então, para cada um desses vértices mais próximos, ele explora os seus vizinhos ainda

inexplorados e assim por diante, até que termine ao encontrar o vértice alvo da busca, ou quando terminar de explorar o grafo inteiro.

O BFS apresenta, no pior caso, aquele em que todos os vértices e arestas são explorados pelo algoritmo, complexidade de tempo igual a $O(E+V)$, onde $E$ é o número de arestas do grafo e $V$ o número de vértices.

### 2.2. Algoritmo de Prim

O algoritmo de Prim é um algoritmo guloso que calcula a árvore geradora mínima/máxima de um grafo não direcionado, ponderado e conectado. A árvore geradora mínima/máxima é um subgrafo do grafo original que conecta todos os vértices e cuja soma dos pesos das arestas é mínima/máxima.

Este algoritmo funciona selecionando um vértice para iniciar o subgrafo e depois adiciona os demais vértices escolhendo sempre aquele cuja aresta até o subgrafo tem peso mínimo/máximo.

A complexidade de tempo do algoritmo de Prim depende da estrutura de dados utilizada para determinar qual vértice, cuja aresta tem o menor/maior peso entre as demais, será adicionado ao subgrafo. No caso deste trabalho, foi implementado uma fila de prioridades máxima utilizando um Heap binário. Desta forma a complexidade de tempo do algoritmo de Prim é $O(V \lg V + E \lg V)$, onde $E$ é o número de arestas do grafo e $V$ o número de vértices.

## 3. Solução Proposta

Esta seção descreve a representação do grafo da rede social e apresenta a abordagem do problema divida em duas etapas. A primeira etapa verifica se todos os usuários da rede social estão conectados e, a segunda calcula o subgrafo cuja configuração apresenta a função de qualidade máxima. Também são apresentadas, nesta seção, as complexidades de tempo e espaço da solução.

### 3.1. Representação da Rede Social

A rede social foi modelada como um grafo não direcionado *G = (V, E)*, onde as arestas *E* indicam se há relações entre os usuários *V*. O grafo foi implementado com uma lista de adjacências. A complexidade de espaço dessa representação é $\theta(V + E)$.

### 3.2. Validar a rede social

Para determinar se a rede social é valida, ou seja, se todos os usuários estão conectados direta ou indiretamente através de outros amigos, é preciso verificar se o grafo, que representa a rede social, é conexo.

Esta verificação foi feita com o BFS, rodando a partir de um vértice qualquer do grafo. O BFS marca todos os vértices explorados, então, se todos os vértices foram marcados, o grafo é conexo.

### 3.3. Maximizar a função de qualidade

Neste trabalho não foi possível, em tempo polinomial, calcular o valor de $r$ diretamente a partir do grafo da rede social. Desta forma, foi implementado um processo invertido, onde

é feito uma busca binária no conjunto possível de valores que $r$ pode assumir e verificado no grafo se o valor é atingível.

Para verificar se $r$ é atingível, reescrevemos a função de qualidade em função das métricas de cada aresta e do $r$ procurado:

$$r = \frac{\sum_{i=1}^{E} Friendship_i}{\sum_{i=1}^{E} Distance_i}, \forall i \in E$$

$$r = \frac{F_1 + F_2 + ... + F_n}{D_1 + D_2 + ... + D_n}$$

$$F_1 + F_2 + ... + F_n = r(D_1 + D_2 + ... + D_n)$$

$$F_1 + F_2 + ... + F_n = (rD_1) + (rD_2) + ... + (rD_n)$$

$$(F_1 - rD_1) + (F_2 - rD_2) + ... + (F_n - rD_n) = 0$$

$$\sum_{i=1}^{E}(Friendship_i - r \times Distance_i) = 0 \quad (1)$$

A Equação 1 pode ser interpretada, então, como um conjunto de arestas cujo $Friendship_i - r \times Distance_i$ somam zero caso o valor de $r$ seja atingível. Se o valor do somatório dessa equação for maior que zero, significa que o conjunto de arestas podem atingir uma função de qualidade maior que o $r$ procurado. Caso o somatório seja menor que zero, então $r$ está acima do valor máximo atingível.

A única condição imposta a essa formulação é que o conjunto de arestas necessariamente precisam manter o grafo conexo. Para garantir essa condição, adicionamos ao conjunto todas as arestas que pertencem a árvore geradora máxima do grafo, considerando como pesos de cada aresta $Friendship - r \times Distance$. Para maximizar o somatório, são adicionadas ainda ao conjunto todas as arestas com pesos positivos, mas que não pertencem a árvore geradora máxima.

O *lowerbound* da busca binária é zero. Já o *upperbound* é o somatório dos *Friendships* de todas as arestas, dividido pelo número de arestas. A escolha desse *upperbound* é devido ao fato do menor valor de *Distance* ser 1.

### 3.4. Complexidade da solução

Na primeira etapa, para verificar se o grafo é conexo, é executado um BFS, cuja complexidade de tempo é $O(E + V)$.

Na segunda etapa é feita uma busca binária, na qual cada busca executa o algoritmo de Prim para achar a árvore geradora máxima. A complexidade de tempo do algoritmo de Prim implementado é $O(V \lg V + E \lg V)$, conforme explicado na Seção 2.2. A busca binária executa no máximo $\log(upperbound - lowerbound)$ vezes, que é igual ao $\log(upperbound)$ já que *lowerbound* é zero.

Dessa forma, a complexidade de tempo da solução do problema é:

$$O(E + V) + O(\log(upperbound)) \times O(V \lg V + E \lg V)$$

. E a complexidade de espaço é:

$$\theta(V + E)$$

.

## 4. Avaliação Experimental

Nesta seção é avaliada a convergência da busca binária e também o tempo de execução da solução em relação ao tamanho do grafo.

Os testes foram realizados no sistema operacional Ubuntu 14.04, rodando um notebook com processador 3rd Generation Intel® Core™ i7-3630QM (2.40GHz 6MB Cache), 8GB RAM, HHD 1 TB S-ATA (5,400 rpm). Para cada teste, foram feitas 5 execuções e retirado a média.

### 4.1. Convergência da busca binária

O gráfico da Figura 1 mostra como a Função de Qualidade $r$ converge na busca binária. Para um *upperbound* inicial de aproximadamente 25000, foi necessário 47 iterações para a convergência.

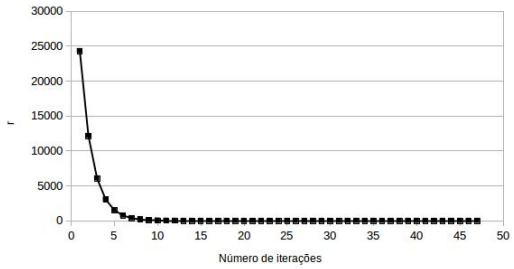


**Figura 1. Convergência da busca binária.**

### 4.2. Tempo de execução

O tempo de execução da solução proposta foi avaliado em dois tipos diferentes de grafos: denso, onde $E = V^2$ e esparso, onde $E = 2V$. Estes grafos foram gerados com valores aleatórios de *Friendship* e *Distance*, e o número de vértices variando de 20 a 200. A Figura 2 mostra o gráfico do tempo de execução da solução para grafos densos, e a Figura 3 para grafos esparsos.

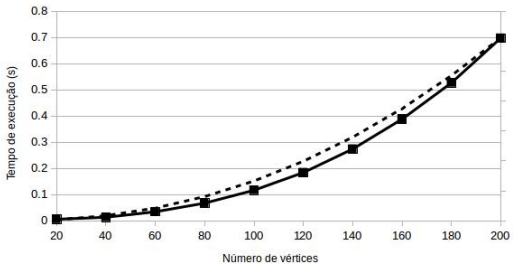


**Figura 2. Grafo denso, $E = V^2$**

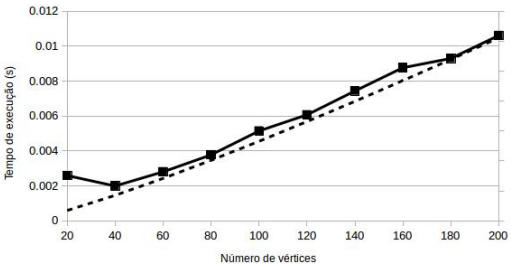


**Figura 3. Grafo esparso, $E = 2V$**

Conforme podemos observar nos gráficos, em grafos densos, a solução apresentou um tempo de execução crescente proporcional a aproximadamente $V^2 \lg V$ (linha pontilhada no gráfico), que é a maior parcela assintótica da complexidade de tempo da solução. Já em grafos esparsos, a solução apresentou um tempo de execução crescente proporcional a $2V \lg V$ (linha pontilhada no gráfico).

## 5. Conclusão

O objetivo deste trabalho foi determinar se um dado grafo não direcionado é conexo, e caso seja, qual é o subconjunto de arestas que mantém o grafo conexo que maximiza uma determinada função de qualidade.

No trabalho foi feito uma Busca em Largura para determinar se o grafo é conexo e utilizado o algoritmo de Prim para gerar a árvore geradora máxima que contém as arestas que maximizam a função de qualidade.

## Referências

Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., et al. (2001). *Introduction to algorithms*, volume 2. MIT press Cambridge.