Algoritmos e Estruturas de Dados

3ª Série

Contar triângulos numa rede

N. 50471 Brian Melhorado

N. 50543 Arthur Oliveira

Licenciatura em Engenharia Informática e de Computadores

Semestre de Verão 2022/2023

11/06/2023

Índice

1. Introdução

A contagem de triângulos é essencial para avaliar o efeito de clustering em redes. Seja G um grafo que represente uma rede, por exemplo social, com n participantes em pares de amigos. Então, é expectável que o número de triângulos seja muito maior neste grafo do que num grafo aleatório. O motivo é que se A e B são amigos e A é também amigo de C, então existe uma maior probabilidade, em geral, de que B e C sejam amigos. Deste modo, a contagem do número de triângulos é uma métrica que pode ser utilizada avaliar este efeito. A contagem de triângulos também pode ser utilizada para classificar a evolução temporal de uma rede, visto que à medida que a rede evolui, o número de triângulos se densifica.
O problema é descrito por:
• um conjunto V de n vértices;
• uma lista E de m ligações entre vértices, em que cada ligação é descrita por um par composto pela identificação de dois vértices.
O objetivo deste trabalho é, portanto, a realização de uma aplicação que:
• calcule o número de triângulos numa rede social;
• liste, para cada vértice, o número total de triângulos a que pertence;
• dado um determinado k, obtenha os k vértices que pertençam a mais triângulos.

Os grupos deverão utilizar as redes presentes em https://snap.stanford.edu/data/ . Existem redes orientadas e não orientadas e com formatos diferentes. Como tal, deverá realizar uma função de leitura genérica, que receba como parâmetro o reader, a string que descreve o separador de linha, o número de linhas iniciais a não considerar, e se é um grafo orientado ou não. Por exemplo, o seguinte excerto de código

```
val myReader=createReader("exemplo.txt")
val graph=readFile(myReader, ",",0, false)
```

descreve como deve ser parametrizada a função para o exemplo que consta no enunciado.
Exemplo:
Considere que o ficheiro de entrada (assuma que representa uma rede não orientada), que designaremos por
exemplo.txt em que cada número é um identificador de um vértice e cada linha descreve uma aresta, tem o seguinte
conteúdo:

```
1,2
2,3
3,5
5,4
4,3
5,2
4,1
6,7
6,8
6,9
9,10
10,6
```

A aplicação deverá inicialmente receber o nome do ficheiro de input (ex: exemplo.txt) e o nome do ficheiro de um dos ficheiros de output (ex: listCounting.txt). De seguida deve retornar o número de total de triângulos e listar para o ficheiro de output especificado, para cada vértice, o número total de triângulos a que pertence. Por fim deve possibilitar que o utilizador ou insira um k é um nome de um outro ficheiro de output, que lista para esse novo ficheiro os k vértices que façam parte do maior número de triângulos existentes, ou termine a aplicação.
Um exemplo de execução da aplicação, é o seguinte:
>countingTrianglesKt exemplo.txt listCounting.txt
> 3
> Introduza um número positivo k ou 0 se quiser terminar a aplicação
> 2 kCounting.txt
> Introduza um número positivo k ou 0 se quiser terminar a aplicação
> 0
O que deverá produzir os seguintes ficheiros de texto:
//listCounting.txt
1 -> 0
2 -> 1
3 -> 2
4 -> 1
5 -> 2
6 -> 1
7 -> 0
8 -> 0
9 -> 1
10 -> 1
//kCounting.txt
3 -> 2
5 -> 2

2. Agrupar palavras

Esta secção tem como intuito abordar o problema a resolver. Vamos dividi-la em três partes:

• **Analise do problema**, onde será apresentada uma descrição do problema, assim como também as operações necessárias a implementar e os métodos utilizados para o resolver

• **Estrutura de Dados**, onde serão ilustradas as principais estruturas de dados (ADT) utilizadas na solução do problema, com alguns exemplos da sua utilização.

•**Algoritmos e análise da complexidade espacial,** que vai constar da descrição dos principais algoritmos utilizados na implementação das operações usadas para o desenvolvimento da solução, assim como também a analise da sua complexidade espacial.

2.1 Análise do problema

O problema consiste em analisar redes sociais representadas por grafos e realizar as seguintes tarefas:

1. Calcular o número de triângulos na rede social.
2. Listar, para cada vértice, o número total de triângulos aos quais ele pertence.
3. Dado um valor k, obter os k vértices que pertencem ao maior número de triângulos.

Operações Necessárias

É necessário implementar uma função genérica de leitura que receba como parâmetros o leitor, o separador de linha, o número de linhas iniciais a serem ignoradas e se o grafo é orientado ou não.

Métodos Utilizados para Resolver o Problema

Para resolver o problema descrito, podem ser utilizados os seguintes métodos:

1. Algoritmo de Contagem de Triângulos:

- Este algoritmo percorre todos os triângulos possíveis no grafo e conta quantos deles existem.

2. Algoritmo de Listagem de Triângulos por Vértice:

- Para cada vértice no grafo, é necessário percorrer suas arestas e verificar se os vértices adjacentes formam um triângulo com ele.
- Esse processo é repetido para todos os vértices no grafo.

3. Ordenação dos Vértices pelo Número de Triângulos:

- Uma Priority queue que usa um compare que vê qual dos vértices possui a maior quantidade de triângulos.

4. Leitura Genérica do Grafo:

- Implementar uma função que possa ler diferentes formatos de arquivos de grafo, considerando se o grafo é orientado ou não, o separador de linha e o número de linhas iniciais a serem ignoradas.
- É necessário extrair as informações de vértices e arestas do arquivo e criar a estrutura de dados adequada para representar o grafo.

2.2 Estruturas de Dados

Existem duas principais estruturas de dados envolvidas:

1. Grafo (Lista de Adjacências): O grafo é representado utilizando uma estrutura de dados que armazena a informação de quais vértices estão conectados uns aos outros. A representação é feita com um HashMap. Nesta estrutura, cada vértice do grafo é representado por um elemento da lista, e cada elemento contém um conjunto (ou HashSet) dos vértices vizinhos. Essa estrutura permite um acesso eficiente aos vizinhos de cada vértice e é utilizada para calcular o número de triângulos.

2. Triângulos: Durante o cálculo do número de triângulos, é necessário armazenar temporariamente os triângulos encontrados. Pode-se utilizar uma estrutura de dados adicional para armazenar essas informações, como uma lista de triângulos, onde cada triângulo é representado por um conjunto (ou HashSet) de vértices. Essa estrutura permite verificar a existência de triângulos e também pode ser utilizada para listar os vértices que pertencem a mais triângulos.

2.3 Algoritmos e análise espacial

Se a entrada do problema é um HashMap e a lista de adjacências é um HashSet, a complexidade espacial será influenciada por ambas as estruturas de dados.

Vamos considerar a seguinte representação:

Entrada: Um HashMap onde as chaves representam os vértices e os valores são conjuntos (HashSet) contendo os vizinhos de cada vértice. O espaço necessário para armazenar a entrada será proporcional ao número de vértices (V) e ao número total de conexões (E).

Lista de Adjacências: Cada vértice no HashMap contém um HashSet que representa seus vizinhos. O espaço necessário para armazenar a lista de adjacências será proporcional ao número de vértices (V) e ao número médio de vizinhos por vértice.

Portanto, a complexidade espacial do problema, considerando a entrada como um HashMap e a lista de adjacências como um HashSet, seria aproximadamente:

Espaço de Entrada: O espaço de entrada será proporcional ao número de vértices (V) e ao número total de conexões (E). A complexidade espacial da entrada será O(V + E).

Espaço Auxiliar (Lista de Adjacências): O espaço auxiliar necessário para a lista de adjacências será proporcional ao número de vértices (V) e ao número médio de vizinhos por vértice. A complexidade espacial da lista de adjacências será influenciada pela distribuição das conexões no grafo.

Portanto, a complexidade espacial total do problema será em torno de O (V + E), levando em consideração o espaço necessário para a entrada do HashMap e o espaço auxiliar utilizado pelo HashSet na representação da lista de adjacências.

3. Avaliação Experimental

Para fazer a avaliação experimental utilizamos os três ficheiros facebookcombined.txt (não orientado), wikivote.txt (orientado), artist_edges.txt (não orientado), disponibilizados pela professora.

Características da máquina onde foram realizados os testes:
Processador: AMD Ryzen 7 5700U with Radeon Graphics 1.80 GHz
RAM:16,0 GB (13,8 GB usable)
Tipo de sistema Sistema operativo de 64 bits, processador baseado em x64

Ficheiro	facebookcombined	wikivote	artist_edges
Número de arestas	88.234	103,689	1,380,293
Tempo de execução	25.777s	3.3706s	32m 56.597s

Tabela 1: Resultados do tempo de execução considerando várias amostras.

Através das avaliações é possível ver um crescimento no tempo de execução da função, sendo que quanto maior for o número de arestas nos ficheiros a serem analisados maior o tempo de execução obtido nas análises. Além disso, também é possível observar que no caso dos grafos orientados, o tempo de execução é menor, em relação aos grafos não orientados.

4. Conclusões

Neste trabalho, propusemos uma solução eficiente para o problema de contagem de triângulos em redes sociais. Utilizamos estruturas de dados como HashMap, HashSet e uma priority queue para calcular o número de triângulos e identificar os vértices mais conectados. A solução mostrou-se adequada para avaliar o efeito de clustering em redes.

Referências

[1] “Disciplina: Algoritmos e Estruturas de Dados - 2223SV,” Moodle 2022/23. [Online]. Available: https://2223.moodle.isel.pt. [Accessed: 15-05-2023].

[2] Introduction to Algorithms, 3º Edition. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. MIT Press.