Introdução à Programação Paralela — Primeiro Trabalho

2021

1 Grafos

Na área de redes complexas representamos um sistema por elementos e suas relações. Por exemplo, pessoas e suas relações de amizade, ou proteínas e suas relações de interação, ou aeroportos e a existência de um vôo direto entre dois aeroportos.

A representação é conhecida matematicamente como um *grafo*. Um grafo, representado por $G(V, E)$ consiste em um conjunto de *vértices* ou *nós* V e um conjunto de *arestas* ou *ligações* E. Para o nosso propósito, o conjunto V é simplesmente um conjunto de identificadores de nós, com o identificador sendo um número inteiro entre 0 e $N-1$, onde $N = |V|$ é o número de vértices,[1] isto é, $V = \{0, 1, 2, \ldots, N-1\}$. O conjunto E é composto de pares (i, j) onde $i, j \in V$. Aqui assumimos que as ligações não têm direção, isto é, a presença de (i, j) em E é equivalente à presença de (j, i) e indica tanto que i está ligado a j quanto que j está ligado a i. Simplificações adicionais que assumiremos aqui são que não existem ligações entre um vértice e ele mesmo; que uma ligação $(i, j) \in E$ é única, isto é, não existem múltiplas ligações entre o mesmo par de vértices; e que as ligações são binárias: ou elas existem ou não, sem termos ligações mais fortes ou mais fracas.

Grafos densos e esparsos Dizemos que um grafo é *denso* quando uma fração significativa de todas as ligações possíveis está presente; caso contrário, o grafo é *esparso*. Mais precisamente, o grafo é denso se $|E| \propto N^2$ e esparso se $|E| \ll N^2$. Em redes complexas, frequentemente temos $|E| \propto N$ [pois isso implica um custo por vértice (em termos de número de ligações) constante, independente de N].

2 Grau e clube dos ricos

Chamamos de *grau* de um vértice o número de ligações que ele tem. Representamos o grau do vértice i por k_i. Em muitas redes, existe um conjunto pequeno de vértices que tem grau muito maior do que a média. Esses vértices são denominados *hubs*, e têm um papel bastante importante na estrutura da rede e em dinâmicas que ocorrem entre os vértices.

O chamado *coeficiente de clube dos ricos* (*rich-club coefficient*) é uma tentativa de quantificar a tendência desses vértices de grau maior estarem conectados uns com os outros (existem outras formas de fazer isso).

Primeiro definimos o conjunto

$$\mathcal{R}(k) = \{i \in V \mid k_i > k\},$$

isto é, o conjunto de todos os vértices que têm grau maior do que k. Seja $n_k \equiv |\mathcal{R}(k)|$. O coeficiente de clube dos ricos para o grau k é definido como:

$$r(k) = \begin{cases} \frac{1}{n_k(n_k-1)} \sum_{i,j \in \mathcal{R}(k)} \mathbf{1}_E(i, j) & \text{se } n_k > 1, \\ 1 & \text{se } n_k \leq 1, \end{cases}$$

onde $\mathbf{1}_E(i, j)$ é a função característica do conjunto das arestas E e vale 1 se $(i, j) \in E$ e 0 caso contrário.

[1] Se C é um conjunto, então $|C|$ é a sua *cardinalidade*, isto é, o seu número de elementos (para conjuntos finitos).

3 Representação de grafos

Existem diversas formas de representar um grafo para fazermos cálculos sobre ele. Aqui vamos discutir as três mais comuns. Para isso, vamos usar um exemplo com 5 vértices, sendo que os vértices 1, 2, 3 e 4 estão ligados ao vértice 0 e os vértice 2 e 3 são ligados entre si, com nenhuma outra ligação presente.

3.1 Matriz de adjacência

Podemos representar um grafo através da sua denominada *matriz de adjacência*, $\mathbf{A}$, que é uma matriz tal que o seu elemento na linha i, coluna j, a_{ij} vale 1 se os vértices i e j são ligados, e 0 caso contrário (supondo que as linhas e colunas são numeradas a partir de zero, como os vértices).

Para o nosso grafo de exemplo, a matriz de adjacência será:

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

3.2 Lista de adjacência

Na representação por lista[2] de adjacência, o grafo é representado através das listas de vizinhos para cada um dos vértices, isto é, para cada vértice i guardamos os identificadores de seus vizinhos.

Para o nosso grafo de exemplo, teríamos as listas:

$$[\{1,2,3,4\},\{0\},\{0,3\},\{0,2\},\{0\}]$$

3.3 Lista de arestas

Na representação por lista de arestas, simplesmente guardamos uma lista com todas as arestas do grafo.

Para o nosso exemplo, teríamos

$$\{(0,1),(0,2),(0,3),(0,4),(2,3)\}$$

4 Descrição do trabalho

Você deve escrever um programa que, dado um grafo de entrada, calcule e escreva em um arquivo os valores de $r(k)$ para $k = 0 \ldots k_{\max} - 1$, onde $k_{\max}$ é o maior grau presente no grafo.

Escolha uma representação de grafo e uma implementação dos cálculos que levem em consideração necessidade de bom desempenho, considerando que as redes utilizadas serão esparsas, com $|E| \propto N$, mas N pode ser grande.

Leituras e escritas em arquivo são demoradas, e o tempo é bastante dependente do desempenho do sistema de arquivos no computador. Por essa razão, ao fazer avaliação de desempenho neste trabalho você deve temporizar apenas a parte do cálculo, deixando de fora a leitura do arquivo de entrada e a escrita dos resultados.

[2] O uso do termo “lista” nesta situação não implica que a implementação necessariamente precisa usar listas ligadas. Uma lista aqui é apenas uma sequência de valores.

4.1 Entradas

O grafo a ser processado por seu programa será fornecido em um arquivo, com o formato descrito abaixo. O nome desse arquivo deve ser lido como primeiro parâmetro da linha de comando (`argv[1]`).

Esta recomendação é importante! Não se deve ler o nome do arquivo com um `scanf` ou similar, mas sim da linha de comando. Também, uma vez fornecido o comando com o nome do arquivo a carregar, o programa não deve esperar mais nada nem imprimir qualquer mensagem que não seja uma mensagem final dizendo o tempo de processamento, mas apenas gerar a saída especificada, a não ser que haja erro na leitura do arquivo.

O arquivo de entrada tem o seguinte formato:

- A primeira linha tem um valor inteiro positivo que é o número de vértices no grafo.
- A segunda linha tem um inteiro positivo que é o número de arestas no grafo.
- Cada uma das linhas seguintes representa uma aresta e consiste em dois valores inteiros, que são os identificadores dos vértices que essa aresta conecta. Os identificadores são de 0 a $N-1$, onde N é o número de vértices do grafo.

Para o nosso exemplo, o arquivo de entrada seria, por exemmplo:

```
5
5
0 1
0 2
0 3
0 4
2 3
```

Note que não especificamos uma ordem para as arestas. Por exemplo, o seguinte arquivo descreve o mesmo grafo do exemplo:

```
5
5
3 0
3 2
0 1
0 4
2 0
```

No entanto, garantimos que cada aresta irá aparecer apenas uma vez no arquivo. Os arquivos fornecidos terão a extensão `.net`, por exemplo `ex015.net`.

4.2 Saídas

O programa deve gerar um arquivo de saída com o mesmo nome do arquivo de entrada *mas com a extensão trocada para* `.rcb`. Por exemplo, se o arquivo de rede se chamava `ex015.net` então o arquivo de saída se chamará `ex015.rcb`.

O formato desse arquivo é de um valor de $r(k)$ por linha, com o valor de $r(0)$ na primeira linha, $r(1)$ na segunda, e assim por diante.

Os valores devem ser escritos com 5 casas decimais de precisão depois da vírgula.

4.3 O que entregar

Você deve entregar **o código fonte** (C, C++ ou Fortran) do seu programa, que deve estar todo em um único arquivo (não use compilação separada). Esse código fonte deve:

1. Incluir comentários descrevendo as decisões tomadas que têm impacto no desempenho (por exemplo, representação do grafo e forma de cálculo do coeficiente). Se você fez experimentos para escolher a melhor forma, inclua nos comentários alguns resultados ilustrativos.

2. Ser um código "limpo", isto é, sem resíduos de versões anteriores ou pedaços de código usados apenas para depuração.

3. Ser adequadamente formatado. Diversos ambientes de programação fazem a formatação para você. Caso não use um ambiente que faça a formatação, você pode usar `clang-format` ou então `indent` (procure na Internet).