UNIVERSIDADE FEDERAL DE SÃO JOÃO DEL-REI - UFSJ
COMPUTAÇÃO PARALELA 2024/2
RAFAEL SACHETTO OLIVEIRA

Trabalho prático 1 - Em Grupo (máximo 3 pessoas)

1 Objetivo

O objetivo deste trabalho é desenvolver uma solução paralela utilizando MPI (Message Passing Interface) para encontrar os vizinhos comuns de pares de vértices em um grafo. O trabalho visa explorar a decomposição de tarefas em um problema de grafos e a comunicação entre processos, otimizando o tempo de execução em relação à versão sequencial.

2 Descrição do Problema

Dado um grafo não-direcionado $G = (V, E)$, onde V é o conjunto de vértices e E é o conjunto de arestas, o problema consiste em determinar, para cada par de vértices u e v, o número de vizinhos comuns, ou seja, os vértices que são adjacentes tanto a u quanto a v. Em termos matemáticos, isso significa encontrar a interseção entre os conjuntos de vizinhos de u e v, denotado como $N(u) \cap N(v)$.

Exemplo

Se o grafo tiver os vértices u, v, w, e as arestas (u, v), (v, w), (u, w), os vizinhos comuns entre u e v seriam o vértice w.

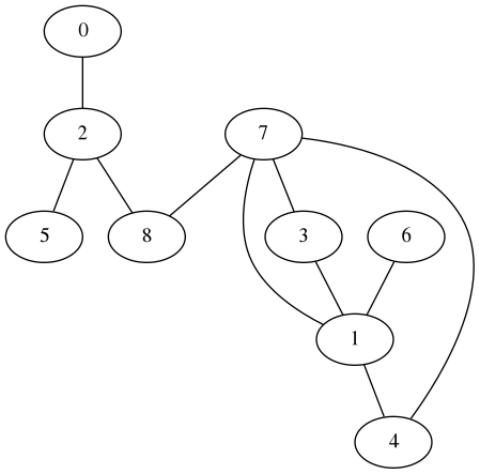
Entrada

A entrada para o seu programa será um arquivo que descreve o grafo no formato conhecido como *edgelist*. O nome do arquivo de entrada deve ser lido pelo seu programa da linha de comando. Neste formato, são dados dois a dois os índices de vértices conectados por arestas.

Por exemplo, se o arquivo de entrada contém:

```
0 2 1 4 3 1 2 5 7 8 2 8 6 1 4 7 1 7 7 3
```

O grafo resultante seria:



Saída

A saída produzida deve ser um arquivo texto (ASCII) no seguinte formato:

- Cada linha corresponde às informações sobre um par distinto de vértices.
- A linha possui três valores inteiros, i, j e k e indica que entre os vértices vi e vj existem k vizinhos em comum.
- Apenas i, j, k ou j, i, k devem aparecer no arquivo de saída, nunca os dois.
- Se o número de vizinhos em comum entre dois vértices é zero, o par correspondente não deve aparecer no arquivo de saída.

Para a entrada do exemplo acima, a saída deve ser equivalente ao seguinte:

```
0 5 1
0 8 1
1 3 1
1 4 1
1 7 2
1 8 1
```

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

O nome do arquivo de saída produzido deve ser o nome do arquivo de entrada, mas com a extensão .cng (a extensão anterior, se existente, deve ser removida). Por exemplo, a saída produzida para o arquivo `grafo.edgelist` deve ficar no arquivo `grafo.cng`.

3 Tarefas

Versão Sequencial

Implementar uma versão sequencial que encontra os vizinhos comuns para cada par de vértices u e v. Para cada par de vértices, o algoritmo deve comparar seus conjuntos de vizinhos e determinar os vizinhos comuns.

Versão Paralela (usando MPI)

Desenvolver uma versão paralela utilizando MPI, onde o cálculo dos vizinhos comuns para diferentes pares de vértices é distribuído entre vários processos MPI.

- Dividir o trabalho entre os processos de maneira eficiente, garantindo que cada processo seja responsável por uma parte do grafo ou por um conjunto específico de pares de vértices.
- Implementar a comunicação entre os processos MPI para compartilhar e reunir os resultados parciais, utilizando funções como `MPI_Send`, `MPI_Recv`, ou `MPI_Gather`.

Divisão do Problema

Uma possível abordagem para a paralelização é a divisão de pares de vértices entre os processos. Cada processo pode receber um subconjunto de pares (u, v) para calcular os vizinhos comuns. Outra opção é dividir o grafo em subgrafos e atribuir a cada processo um subgrafo, sendo responsável por calcular os vizinhos comuns envolvendo vértices dentro de seu subgrafo e, em alguns casos, interagir com outros processos para verificar as relações entre vértices que pertencem a subgrafos diferentes.

Avaliação de Desempenho

- Comparar o tempo de execução da solução sequencial com a versão paralela.
- Realizar testes com grafos de diferentes tamanhos e estruturas (grafo esparso, grafo denso) e com diferentes números de processos MPI.
- Utilizar funções de medição de tempo, como `MPI_Wtime()`, para coletar dados e criar gráficos que mostrem a relação entre o número de processos e o desempenho.

Relatório

- Explicar a estratégia utilizada para paralelizar o problema.
- Analisar o desempenho da solução paralela, indicando melhorias ou possíveis otimizações.
- Incluir uma comparação detalhada entre a versão sequencial e paralela, utilizando gráficos e tabelas que mostrem a evolução do tempo de execução com o aumento do número de processos e do tamanho do grafo.
- Discutir as dificuldades encontradas no desenvolvimento da solução e as lições aprendidas.

4 Critérios de Avaliação

- Correção e eficiência da solução paralela.
- Escalabilidade e desempenho da implementação com diferentes tamanhos de grafos e números de processos.

- Qualidade do código e da divisão de tarefas entre os processos.
- Clareza e profundidade da análise de desempenho no relatório.

5 Dicas

- Utilize técnicas de decomposição que permitam um bom balanceamento de carga entre os processos, evitando que alguns processos fiquem ociosos.
- Certifique-se de utilizar uma estratégia de comunicação eficiente para minimizar o overhead de comunicação entre os processos.
- Ao testar a implementação, comece com grafos pequenos para verificar a correção do código antes de realizar testes de desempenho com grafos maiores.