Universidade Federal de Minas Gerais
Departamento de Ciência da Computação
Algoritmos e Estruturas de Dados III
2º Semestre de 2012

Trabalho Prático 4
Problemas NP-Completo e Programação Paralela

Esse trabalho tem como objetivo levar o aluno a lidar com dois tópicos avançados de Algoritmos e Estruturas de Dados, problemas NP-Completo e programação concorrente. Espera-se que o aluno consiga modelar um problema de alto custo computacional e mostrar que não é possível implementar uma solução que consiga resolver entradas não triviais de maneira ótima. Ainda, o aluno deve implementar um algoritmo paralelo para resolver o problema em questão exercitando alguns conceitos básicos de programação concorrente.

Problema

Você acaba de ser contratado por uma rede social online exclusiva para cozinheiros, o Facecook. Os donos da rede desejam realizar uma festa entre os seus membros para celebrar o seu primeiro aniversário e querem que você desenvolva um algoritmo que os ajude na seleção do maior número de convidados possível. Há porém uma restrição bastante peculiar, todos os convidados devem ser amigos entre si no Facecook. Os donos da rede querem promover uma festa bem íntima e acreditam que a melhor maneira de se fazer isso é convidar apenas pessoas que já se conheciam previamente.

Para facilitar a compreensão deste problema, imagine que as relações entre os membros do Facecook são dadas pelo grafo da Figura 1. Neste grafo, um vértice representa um membro. Dois membros são amigos no Facecook se e somente se existe uma aresta entre os dois vértices que os representam. A Figura 2 apresenta em azul o maior grupo de usuários tal que todos sejam amigos no Facecook.

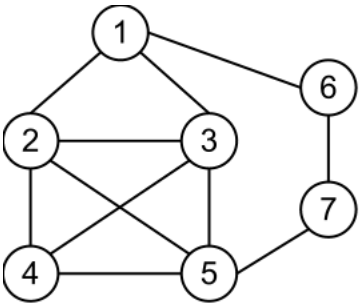

Figura 1: Rede de membros do Facecook. Vértices representam usuários da rede e arestas representam a amizade entre eles.

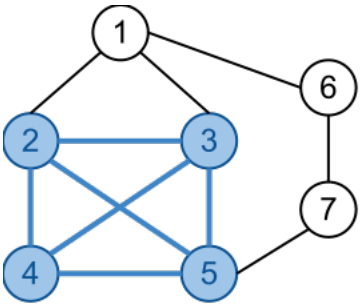

Figura 2: Os membros 2, 3, 4 e 5 formam o maior grupo de usuários que pode ser convidado para a festa de aniversário da rede

Mesmo você dizendo que o problema acima é extremamente caro computacionalmente, os donos da rede insistem que você pode escrever um programa para resolvê-lo rapidamente. Ainda, eles lhe ofereceram um poderoso servidor com múltiplos processadores argumentando que você pode

resolver o problema para entradas muito grandes desde que você paralelize o seu algoritmo. Dessa forma, pede-se

1. Mostre que o problema acima é NP-Completo e, portanto, não pode ser resolvido de forma ótima para entradas grandes.

2. Desenvolva um algoritmo baseado em backtracking para encontrar o maior número de usuários que pode ser convidado para a festa ao mesmo tempo. Calcule a complexidade do algoritmo em termos de memória e espaço.

3. Crie uma versão paralelizada do algoritmo anterior utilizando a biblioteca pthreads[1]. Discuta se esta implementação paralela pode ajudar o algoritmo a executar para entradas maiores, como grafos de 20, 100 ou 1000 vértices.

4. Realize as seguintes análises sobre o desempenho das duas versões do algoritmo

   (a) Descubra o maior grafo para o qual os algoritmos conseguem retornar uma solução em tempo viável.

   (b) Dado o tamanho da entrada descoberto no item anterior, estime o tempo de execução para entradas maiores. Quanto tempo levaria se você adicionasse mais um vértice ao grafo? E se você dobrasse o número de vértices? Qual o maior tamanho da entrada que você conseguiria tratar se você deixasse o seu programa executando pelos próximos 100 anos?

   (c) Mostre através de um gráfico de *tempo x tamanho da entrada* como o seu algoritmo se comporta para diferentes números de threads. Crie apenas um grafo com múltiplas séries e fixe para cada série o número de threads. Discuta como o número de threads influenciou no tempo de execução. Há um limite do número de threads para o qual o algoritmo deixa de executar mais rápido? Para gerar esse gráfico, execute o seu programa em um computador com, no mínimo, 2 núcelos de processamento. Nos laboratórios do departamento, há diversas máquinas dual core que você pode utilizar em seus experimentos.

   (d) Imagine que você possui um cluster de computadores à sua disposição. Estime o número de processadores necessário para resolver, com o mesmo tempo, um problema com 1 vértice a mais do que a entrada descoberta no item (a). Quantos processadores você precisaria para resolver um problema 2 vezes maior com o mesmo tempo?

Entrada e Saída

Você deverá criar duas versões do seu programa, uma sequencial e outra paralela. Ambas as versões devem ler de um arquivo de entrada um ou mais grafos representando a rede de usuários do Facecook e escrever em um arquivo de saída o número de convidados para cada uma das entradas. A versão paralela do seu programa deve receber um parâmetro adicional, o número de threads que deve executar em paralelo.

Seguem abaixo dois exemplos de execução para a versão sequencial e paralela do algoritmo respectivamente:

```
./tp4-seq input.txt output.txt
```

[1] https://computing.llnl.gov/tutorials/pthreads/

```
./tp4-par input.txt output.txt 3
```

O arquivo de entrada possui um inteiro K na primeira linha onde K é o número de instâncias a serem simuladas. Em seguida as K instâncias são definidas da seguinte forma. A primeira linha possui um inteiro, N, que indica o número de usuários da rede. Em seguida a matriz de adjacência que representa a rede de usuários é dada nas N linhas seguintes. Cada linha $0 \leq i < N$ contem as conexões do usuário i com os outros membros da rede. Caso i seja amigo de j, a j-ésima entrada da linha i será igual a 1. Caso contrário ela será igual a 0. O arquivo de saída possui uma única linha com K inteiros separados por espaço, tal que cada inteiro é o número de convidados para a festa. A entrada e saída da Figura 1 é mostrado abaixo:

Entrada:

```
1
7
0 1 1 0 0 1 0
1 0 1 1 1 0 0
1 1 0 1 1 0 0
0 1 1 0 1 0 0
0 1 1 1 0 0 1
1 0 0 0 0 0 1
0 0 0 0 1 1 0
```

Saída:

```
4
```

Entrega

- A data de entrega desse trabalho é **22/12/2012**.
- A penalização por atraso obedece à seguinte fórmula $2^{d-1}/0.32\%$, onde d são os dias úteis de atraso.
- Submeta apenas um arquivo chamado <numero_matricula>_<nome>.zip. Não utilize espaços no nome do arquivo. Ao invés disso utilize o caractere '_'.
- Não inclua arquivos compilados ou gerados por IDEs. **Apenas** os arquivos abaixos devem estar presentes no arquivo zip.
  - Makefile
  - Arquivos fonte (*.c e *.h)
  - Documentacao.pdf
- Não inclua **nenhuma pasta**. Coloque todos os arquivos na raiz do zip.
- Siga rigorosamente o formato do arquivo de saida descrito na especificação. Tome cuidado com whitespaces e formatação dos dados de saída
- NÃO SERÁ NECESSÁRIO ENTREGAR DOCUMENTAÇÃO IMPRESSA!
- Será adotada **média harmônica** entre as notas da **documentação e da execução**, o que implica que a nota final será 0 se uma das partes não for apresentada.

Documentação

A documentação não deve exceder 10 páginas e deve conter pelo menos os seguintes itens:

- Uma **introdução** do problema em questão.
- **Modelagem e solução proposta** para o problema. O algoritmo deve ser explicado de forma clara, possivelmente através de pseudo-código e esquemas ilustrativos.
- **Análise de complexidade** de tempo e espaço da solução implementada.
- **Experimentos** variando-se o tamanho da entrada e quaisquer outros parâmetros que afetem significavelmente a execução.
- Especificação da(s) **máquina(s) utilizada(s)** nos experimentos realizados.
- Uma breve **conclusão** do trabalho implementado.

Código

- O código deve ser obrigatoriamente escrito na **linguagem C**. Ele deve compilar e executar corretamente nas máquinas Linux dos laboratórios de graduação.
- O utilitário ***make*** deve ser utilizado para auxiliar a compilação, um arquivo *Makefile* deve portanto ser incluído no código submetido.
- As estruturas de dados devem ser **alocadas dinamicamente** e o código deve ser **modularizado** (divisão em múltiplos arquivos fonte e uso de arquivos cabeçalho .h)
- **Variavéis globais** devem ser evitadas.
- Parte da correção poderá ser feita de forma automatizada, portanta **siga rigorosamente os padrões de saída especificados**, caso contrário sua nota pode ser prejudicada.
- Os arquivos executáveis devem ser chamados tp4-seq e tp4-par.
- **Legibilidade e boas práticas** de programação serão avaliadas.