**UNIVERSIDADE FEDERAL DE SANTA CATARINA**
**Curso de Engenharia Mecatrônica**
**EMB5632 – Sistemas Operacionais**
**Prof. Dr. Giovani Gracioli**
**2020/02**

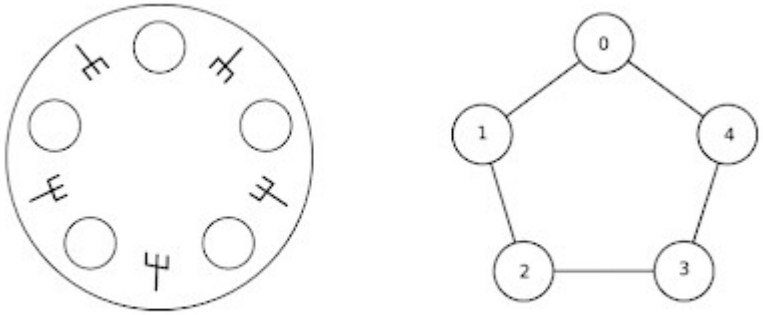
# Problema do Bar dos Filósofos [2]

1) **Apresentação do Problema**

O problema do Bar dos Filósofos [1] é uma generalização do problema dos Jantar dos Filósofos, que foi originalmente proposto por E. W. Dijkstra em 1971. O Jantar do Filósofos é um problema muito emblemático para a Ciência da Computação, especialmente pelo fato de representar a essência de muitos problemas de sincronização entre processos ou threads. Na natureza do problema está o controle de acesso aos recursos compartilhados através de exclusão mútua.

O problema do jantar dos filósofos está representado na Figura ao lado. Cinco filósofos estão sentados ao redor de uma mesa. Em frente a cada um deles estão um prato e um garfo. Como filosofar é uma tarefa cansativa, os filósofos alternam períodos em que estão comendo e filosofando. Para comer, um filósofo precisa pegar os garfos da esquerda e da direita. Entretanto, os garfos são compartilhados com os filósofos também da esquerda e da direita. Depois de comer, o filósofo coloca os dois garfos, da direita e da esquerda, de volta na mesa, possibilitando assim que outro filósofo possa comer. Vale salientar que um filósofo precisa obrigatoriamente dos dois garfos para comer. **Quantos filósofos podem comer simultaneamente nestre problema?** Uma solução para o problema deve garantir que vários filósofos possam comer simultaneamente, além de impedir deadlocks e starvation.

O problema do jantar dos filósofos pode ser representado através de um grafo cíclico, onde cada processo/thread (filósofo) precisa dos recursos ao seu lado para executar sua região crítica (comer ou processar), conforme mostra a Figura abaixo.




(a) Problem setting (b) Formalization as graph

Em 1984, Chandy e Misra propuseram uma generalização para o problema do jantar dos filósofos, chamado o Bar dos Filósofos [1].

A principal diferença entre os dois problemas está nas restrições do grafo. No problema do Bar dos Filósofos, não existe nenhuma restrição com relação à estrutura do grafo, como distribuição ou grau de conectividade. Os filósofos (processos/threads) são representados pelos nodos (vértices) do grafo, enquanto as garrafas (recursos compartilhados) são representadas pelas arrestas do grafo. Os pontos fundamentais devem ser levados em conta nesta problema: (1) a estrutura do grafo é arbitrária; e (2) o número de garrafas requeridas pelos filósofos pode variar. Em cada rodada, cada filósofo escolhe um subconjunto de suas garrafas adjancentes para a "bebedeira". O subconjunto de garrafas não necessariamente é o mesmo a cada rodada. Se o filósofo escolher todas as garrafas adjacentes em um grafo circular, pode-se simular o problema do jantar dos filósofos. Os estados dos filósofos são TRANQUILO, COM SEDE, e BEBENDO, semelhante aos estados filosofando, com fome, e comendo do problema do jantar dos filósofos. Cada filósofo pode requerer no mínimo uma e no máximo n garrafas, sendo n o número de arestas associadas a cada vértice.

## 2) Descrição da Implementação

O trabalho consiste em escrever um programa C++ para simular o problema do Bar dos Filósofos. Utilize classes para representar os filósofos e outras representações que achar necessária. Cada filósofo, ao passar de tranquilo para com sede, deve sortear o número de garrafas requeridas, podendo variar de 2 até n, sendo n o número de arrestas adjacentes ao vértice.

Para simular os estados, utilize os seguintes tempos. **Tempo tranquilo**: de 0 a 2 segundos (escolhido aleatoriamente). **Tempo com sede**: tempo até conseguir todas as garrafas. **Tempo bebendo**: 1 segundo. Após beber todas as garrafas, o filósofo retorna a tranquilo.

Dado um grafo G = (V, E), é possível representar G através de M, uma matriz quadrada de ordem V, onde Mij = 1 se o vértice i possuir uma aresta ao vértice j, ou Mij = 0 caso contrário.

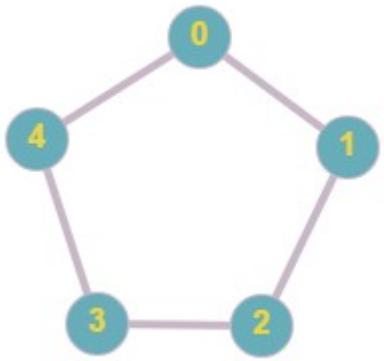
O programa deve ser capaz de ler um arquivo .txt onde cada linha do arquivo representa uma linha de M e cada caractere da linha assume os valores 0 ou 1, indicando as adjacências entre vértices. Considere que o índice de cada caractere nas linhas representa o n-ésimo vértice de G, ou seja, uma coluna de M. É indicado que o programa receba como argumento o local do arquivo contendo a matriz de adjacências.

## 3) Exemplos

Abaixo são exemplificados alguns grafos para o problema e suas respectivas representações no arquivo de entrada.

- Exemplo 1: Problema do Jantar dos Filósofos

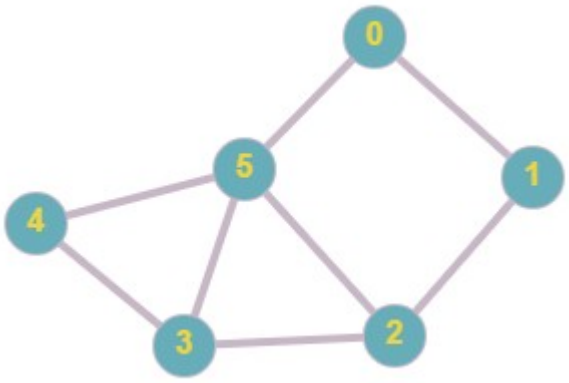
```
/* Jantar do Filosofos */
0, 1, 0, 0, 1,   //vértice 0
1, 0, 1, 0, 0,   //vértice 1
0, 1, 0, 1, 0,   //vértice 2
0, 0, 1, 0, 1,   //vértice 3
```



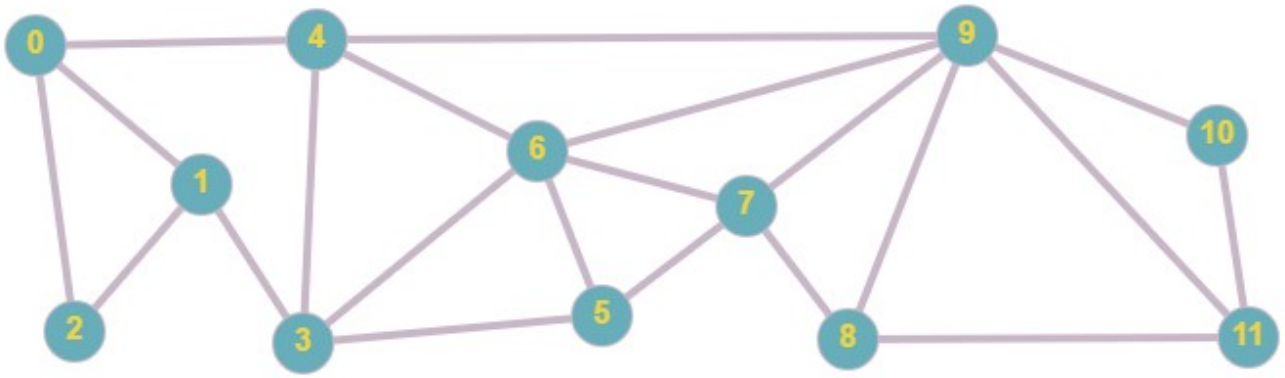
1, 0, 0, 1, 0,  //vértice 4

- Exemplo 2: Grafo 6 nós com baixa conectividade (máx 3 arestas por nó)

/* Grafo 6 nós baixa conectividade */
0, 1, 0, 0, 0, 1,  //vértice 0
1, 0, 1, 0, 0, 0,  //vértice 1
0, 1, 0, 1, 0, 1,  //vértice 2
0, 0, 1, 0, 1, 1,  //vértice 3
0, 0, 0, 1, 0, 1,  //vértice 4
1, 0, 1, 1, 1, 0,  //vértice 5




- Exemplo 3: Grafo 12 nós com alta conectividade  (máx 6 arestas por nó)




/* Grafo 12 nós alta conectividade */
0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,  //vértice 0
1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,  //vértice 1
1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,  //vértice 2
0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,  //vértice 3
1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0,  //vértice 4
0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0,  //vértice 5
0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0,  //vértice 6
0, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 0,  //vértice 7
0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1,  //vértice 8
0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1,  //vértice 9
0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,  //vértice 10
0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,  //vértice 11

Cada filósofo deve beber 6 vezes para os exemplos 1 e 2 e deve beber 3 vezes para o exemplo 3 (uma simulação deve demorar pouco mais de 2 minutos).

Ao final da execução mostrar os tempos de cada filósofo levou em cada estado (tranquilo, com sede e bebendo).

Tempo total para execução e tempo de espera (com sede) médio de cada filósofo para ver a starvation.

Será considerado correto o programa que não apresentar deadlock durante a execução e os tempos médios de espera dos filósofos seja equilibrada (não houve starvation).

## 4) Avaliação

A avaliação consiste na apresentação do programa desenvolvido em um relatório escrito. No relatório, a dupla deve descrever o problema, conter uma descrição do algoritmo para a solução, descrição do projeto e implementação do software (incluindo diagramas UML) e resultados, no mínimo, dos 3 exemplos usados neste documento.

Com relação ao código, serão avaliados os seguintes quesitos:

- Projeto orientado a objeto, clareza e organização do código
- Uso dos recursos da linguagem C++ (std::threads, classes, etc)
- Uso de programação estruturada (-30%)
- Uso da linguagem C ao invés de linguagem C++ (-30%)
- Uso de variável global (-5) – Atributo estático de classe não é considerado variável global
- Vazamento de memória (-30%) - Usar valgrind para detectar vazamento

Plágio não será tolerado em nenhuma hipótese ao longo dos trabalhos, acarretando em nota 0 a todos os envolvidos e envio do caso para a coordenação encaminhar as medidas cabíveis.

## 5) Formato de Entrega

Todos os arquivos utilizados na implementação do trabalho e relatório devem ser entregues em um único arquivo .zip ou .tar.gz na atividade do moodle. Deve ser anexado um arquivo Makefile para compilar o código.

## 6) Data de Entrega

A data e hora da entrega do trabalho estão especificadas na tarefa do moodle. A data e hora da apresentação serão definidas pelo professor.

## 7) Referências

[1] K. M. CHANDY and J. MISRA. The Drinking Philosophers Problem. ACM Transactions on Programming Languages and Systems, Vol. 6, No. 4, October 1984, Pages 632-646. Disponível em: https://www.cs.utexas.edu/users/misra/scannedPdf.dir/DrinkingPhil.pdf.
[2] Prof. Marcial Fernández. Programação Concorrente e Paralela. Universidade Estadual do Ceará (UECE).