Universidade Federal de Itajubá – UNIFEI

ECOM02 – Teoria dos Grafos – Prof. Edmilson Marmo Moreira
Descrição do primeiro trabalho de Implementação em C/C++
Segundo semestre de 2018

QUANTIDADE DE MEMBROS NO GRUPO:

O trabalho será realizado pelo mesmo grupo do trabalho de pesquisa sobre Operações e Conectividade em Grafos.

FORMA DE APRESENTAÇÃO:

O trabalho deverá ser enviado através do SIGAA - *Sistema Integrado de Gestão de Atividades Acadêmicas* da UNIFEI.

Todo o código deve estar contido em um único arquivo fonte denominado: `ECOM02.cpp`

DATA PARA ENTREGA E AVALIAÇÃO:

O data máxima para o envio do trabalho será: ~~**26 de setembro de 2018**~~

03 de outubro de 2018.

Caso haja alguma alteração na composição dos grupos, o grupo tem até o dia **18 de setembro de 2018** para informar a situação, por *e-mail*.

No dia ~~**27 de setembro de 2018**~~ **04 de outubro de 2018**, será realizado um teste para avaliar o conhecimento adquirido pelos alunos. O teste é individual e delimitará a nota máxima de cada membro do grupo.

A nota individual do trabalho não ultrapassará em 30% a nota obtida por cada aluno no teste do dia ~~27/09/2018~~ 04/10/2018.

DESCRIÇÃO DO TRABALHO:

O trabalho consiste na implementação das operações estudadas no trabalho de pesquisa.

Todas as funções deverão ser desenvolvidas no mesmo arquivo com o código principal que será disponibilizado.

Para que fique claro como as funções deverão ser adicionadas ao arquivo do programa principal, o exemplo a seguir ilustra a implementação de uma função que verifica se um grafo (representado através de uma matriz de adjacência) é regular. A função já está contextualizada no código que será fornecido.

```
/*
  Arcabouço para implementação do Trabalho de
  ECOM02 - Teoria dos Grafos
*/

#include <stdio.h>

#define MAX 100
#define INF 2000000

int grafo[MAX][MAX]; // representação matricial dos grafos
// Funcao para ler a matriz
void leGrafo (int g[][MAX], int ordem, int tamanho)
```

```
{
  for (int i=0; i < ordem; i++) {
    for (int j=0; j < tamanho; j++) {
      scanf("%d", &g[i][j]);
    }
  }
}
// Funções desenvolvidas no trabalho

// Verifica se grafo é regular com Matriz de Adjacências
bool regular(int ordem)
{
  int grau=0, grau_aux;
  for (int i=0; i < ordem; i++)
    grau += (grafo[0][i] != 0 ? 1 : 0);
  for (int i=1; i < ordem; i++) {
    grau_aux=0;
    for (int j=0; j < ordem; j++) {
      grau_aux += (grafo[i][j] != 0 ? 1 : 0);
    }
    if (grau != grau_aux)
      return false;
  }
  return true;
}

int main()
{
 int operacao, ordem, tamanho;
/*
  Declaração de outras variáveis necessárias ao trabalho
  int a, b, p;
  int i, j, k;
 int Soma;
*/
  while (true) {
    scanf("%d", &operacao);
    if (!operacao)
      break;
   switch (operacao) {
     case 1: // Caso exemplo - Verificar se o Grafo é regular
       scanf("%d", &ordem);
       leGrafo(grafo, ordem, ordem);
       if (regular(ordem))
         printf("Grafo regular\n");
       else
         printf("Grafo nao regular\n");
       break;
     case 2:
       break;
     case 3:
       break;
     case 4:
       break;
    }
  }
 return (0);
```

```
}
```

Observem que o valor 1 do comando *switch* representa a função de verificação se o grafo é regular utilizando como representação uma Matriz de Adjacência. Os demais casos serão descritos a seguir. Além disso, os dados específicos para a função escolhida são lidos dentro do respectivo "case" e passados como parâmetros para a função específica.

Cuidado com o padrão de entrada e o padrão de saída que será solicitado. É muito importante que estes padrões sejam respeitados. Isso será um dos critérios de correção do trabalho. Cada caso a seguir indica a função que deverá ser desenvolvida.

Lembrem-se, não haverá grafos com ordem e tamanho maior que 100 e **todos os grafos serão representados pela matriz de adjacência**.

CASO 2: ***Calcular a Excentricidade de um vértice***

A função deste caso deverá calcular excentricidade de um vértice. A primeira linha de entrada contém 2 números *N* e *V*, correspondentes à ordem do grafo e ao Vértice cuja excentricidade será calculada. As *V* linhas seguintes representam os elementos da Matriz de Adjacência, sendo *V* elementos em cada linha. Os vértices são numerados de 0 até *V*-1.

Exemplo de entrada:

```
7 3
0 1 0 0 0 0 0
1 0 1 0 0 0 0
0 1 0 1 0 0 0
0 0 1 0 1 1 1
0 0 0 1 0 0 0
0 0 0 1 0 0 0
0 0 0 1 0 0 0
```

Saída esperada:

```
3
```

CASO 3: ***Calcular o Raio de um grafo***

Para o grafo, representado por sua matriz de adjacência, a função deste caso deverá calcular o valor do raio do grafo.

A primeira linha de entrada contém um número inteiro *V*, correspondente à ordem do grafo. As *V* linhas seguintes representam os elementos da Matriz de Adjacência, sendo *V* elementos em cada linha.

Exemplo de entrada:

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

Saída esperada:

```
2
```

CASO 4: ***Calcular o Diâmetro de um grafo***

Para o grafo, representado por sua matriz de adjacência, a função deste caso deverá calcular o valor do diâmetro do grafo.

A primeira linha de entrada contém um número inteiro *V*, correspondente à ordem do grafo. As *V* linhas seguintes representam os elementos da Matriz de Adjacência, sendo *V* elementos em cada linha.

Exemplo de entrada:

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

Saída esperada:

```
4
```

CASO 5: ***Calcular o centro de um grafo***

Para o grafo, representado por sua matriz de adjacência, a função deste caso deverá identificar os vértices que fazem parte do centro do grafo.

A primeira linha de entrada contém um número inteiro *V*, correspondente à ordem do grafo. As *V* linhas seguintes representam os elementos da Matriz de Adjacência, sendo *V* elementos em cada linha. A saída deverá apresentar os índices dos vértices que fazem parte do centro do grafo em uma mesma linha e separados por um espaço.

Exemplo de entrada:

```
5
0 1 1 1 1
1 0 1 1 1
1 1 0 1 1
1 1 1 0 1
1 1 1 1 0
```

Saída esperada:

```
0 1 2 3 4
```

CASO 6: ***Realizar a fusão de dois vértices de um grafo***

Para o grafo, representado por sua matriz de adjacência, a função deste caso deverá gerar um novo grafo resultante da fusão de dois vértices.

A primeira linha de entrada contém três números inteiros *V, A* e *B*, correspondente à ordem do grafo e aos dois vértices que serão fundidos. As *V* linhas seguintes representam os elementos da Matriz de Adjacência, sendo *V* elementos em cada linha. A saída deverá apresentar a matriz de adjacência do grafo resultante. Os vértices são numerados de 0 até *V*-1.

Exemplo de entrada:

```
6 2 3
0 1 1 0 0 1
1 0 1 1 1 0
1 1 0 1 0 0
0 1 1 0 0 0
0 1 0 0 0 0
1 0 0 0 0 0
```

Saída esperada:

```
0 2 1 1 1
2 0 1 0 0
1 1 0 0 0
1 0 0 0 0
1 0 0 0 0
```

CASO 7: ***Realizar a contração de dois vértices de um grafo***

Para o grafo, representado por sua matriz de adjacência, a função deste caso deverá gerar um novo grafo resultante da fusão de dois vértices.

A primeira linha de entrada contém três números inteiros *V, A* e *B*, correspondente à ordem do grafo e aos dois vértices que serão fundidos. As *V* linhas seguintes representam os elementos da Matriz de Adjacência, sendo *V* elementos em cada linha. A saída deverá apresentar a matriz de adjacência do grafo resultante. Os vértices são numerados de 0 até *V*-1.

Exemplo de entrada:

```
6 2 3
0 1 1 0 0 1
1 0 1 1 1 0
1 1 0 1 0 0
0 1 1 0 0 0
0 1 0 0 0 0
1 0 0 0 0 0
```

Saída esperada:

```
0 1 1 1 1
1 0 1 0 0
1 1 0 0 0
1 0 0 0 0
1 0 0 0 0
```

CASO 8: ***Realizar a inserção de um vértice em um aresta de um grafo***

Para o grafo, representado por sua matriz de adjacência, a função deste caso deverá gerar um novo grafo resultante da inserção de um vértice em uma aresta.

A primeira linha de entrada contém três números inteiros *V, A* e *B*, correspondente à ordem do grafo e aos dois vértices correspondentes à aresta que receberá o vértice. As *V* linhas seguintes representam os elementos da Matriz de Adjacência, sendo *V* elementos em cada linha. A saída deverá apresentar a matriz de adjacência do grafo resultante. Os vértices são numerados de 0 até *V*-1. O novo vértice terá a identificação do maior índice do grafo resultante.

Exemplo de entrada:

```
2 0 1
0 1
1 0
```

Saída esperada:

```
0 0 1
0 0 1
1 1 0
```