



Centro Universitário FEI

Projeto de CA-2330

Relatório - Parte 3

Grupo Nº 11

João Pedro Rosa Cezarino - R.A: 22.120.021-5

Lucca Bonsi Guarreschi - R.A: 22.120.016-5

Vitor Martins Oliveira - R.A: 22.120.067-8

**São Bernardo do Campo
2020**

1 Grafos Bipartidos

Um grafo é chamado bipartido quando seu conjunto de vértices V puder ser dividido em dois subconjuntos $V1$ e $V2$, tais que cada aresta de G une um vértice de $V1$ a outro de $V2$.

Cada aresta de G possui um extremo em X e outro em Y (Os extremos deverão estar em conjuntos diferentes).

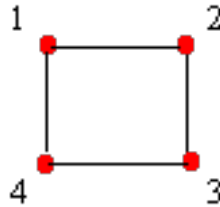


Figura 1: $X = 1, 2$ e $Y = 4, 3$

Para demonstrar que um grafo G é bipartido basta mostrar uma bipartição de VG na qual os extremos estejam em conjuntos diferentes.

2 Grafos Bipartidos Completos

Um grafo é dito bipartido completo se for completo e se todo vértice de X é adjacente a todo vértice de Y . Portanto, X e Y são independentes. Usualmente um grafo bipartido completo é denotado por $K_{m,n}$, onde m é o número de vértices em $V1$ e n é o número de vértices em $V2$.

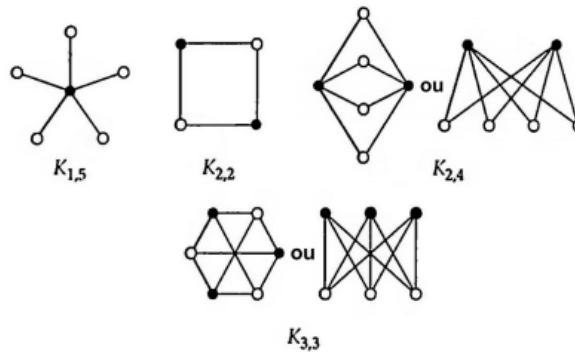


Figura 2

Um grafo bipartido completo $K_{m,n}$ tem $m * n$ arestas

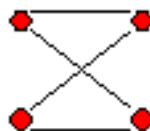


Figura 3: $K_{2,2}$ / 4 Arestas

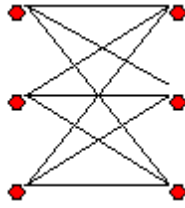


Figura 4: $K_{3,3}$ / 9 Arestas

O grafo G_6 é um $K_{3,3}$, ou seja, um grafo bipartido completo que contém dois conjuntos de 3 vértices cada. Ele é completo pois todos os vértices de um conjunto estão ligados a todos os vértices do outro conjunto.

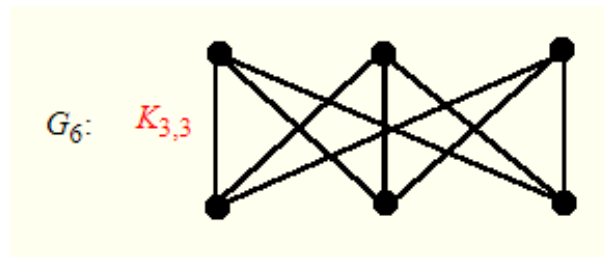


Figura 5

3 Descrição do Programa

Inicialmente o arquivo onde se encontra a matriz de adjacência "A.txt" é aberto e a leitura de cada linha da matriz é realizada, com o objetivo de retirar os espaços (" \n") e transformar o conteúdo em números inteiros (do tipo "int"). Após esse processo, o programa verifica e retira (se houver a presença) os espaços em branco desnecessários na matriz.

```

arq = open('A.txt', 'r') ## Abre o arquivo no modo "read" ##
lista = arq.readlines() ## Lê o arquivo ".txt" e cria uma lista com seu conteúdo ##
lista_final = []
arq.close()

for x in lista: ## Esse "for" lê a lista e o arquivo e adiciona o seus elementos a outra lista, removendo as quebras de linhas ##
    x = x.rstrip('\n')
    lista_final.append(x)

for num in lista_final: ## Remove os espaços em branco do final da lista, caso tenha algum ##
    if '' in lista_final:
        lista_final.remove('')

Matriz1 = [[int(num) for num in line.split(' ')] for line in lista_final] ## Transforma a lista em uma matriz ##
Matriz2 = [[int(num) for num in line.split(' ')] for line in lista_final]

```

Figura 6

Após a abertura do arquivo, duas variáveis e duas listas são inicializadas. A função "adjacentes(y)" verifica quais vértices do grafo são adjacentes entre si e retorna "True" caso eles sejam adjacentes. A matriz então é percorrida com o objetivo de checar se os vértices podem ser divididos em dois grupos distintos.

```
def bipartido(matriz):
    result = ""
    Bipartido = True
    u = []
    v = []

    def adjacentes(v):
        if len(v) > 1:
            for i in range(len(v)):
                for j in range(len(v)):
                    z = v[i]
                    k = v[j]
                    if matriz[z][k] != 0:
                        return True
            return False

    for i in range(len(matriz)):
        if i not in v:
            u.append(i)
            for j in range(len(matriz)):
                if j > i:
                    if matriz[i][j] > 0:
                        if j not in v:
                            v.append(j)
```

Figura 7

Após as instruções acima, o programa passa a verificar se o grafo é bipartido ou não. Caso ele seja, uma bipartição será exibida no terminal.

```
    if adjacentes(v):
        Bipartido = False

    if Bipartido:
        result += ("O grafo é bipartido, e possui bipartições em u = {")
        for i in range(len(u)):
            if i < len(u)-1:
                result += ("V{}".format(u[i]+1) + ", ")
            else:
                result += ("V{}".format(u[i]+1) + " e v = {")

        for j in range(len(v)):
            if j < len(v)-1:
                result += ("V{}".format(v[j] + 1) + ", ")
            else:
                result += ("V{}".format(v[j] + 1) + "}")
        result += ("\n")
```

Figura 8

Após a verificação da bipartição, o programa verifica se o grafo pode ser bipartido completo ou não.

```
def bipartidoCompleto(u, v, matriz):
    resultado = ""
    BipCompleto = True

    for i in range(len(u)):
        cont = 0
        for j in range(len(v)):
            k = u[i]
            z = v[j]
            if matriz[k][z] > 0:
                cont += 1
        if cont < len(v):
            BipCompleto = False
            break

    if BipCompleto:
        resultado += ("O grafo é bipartido completo, pois cada vértice com bipartição em u se conecta a todos os vértices com bipartição em v")
    else:
        resultado += ("O grafo não é bipartido completo, pois não são todos os vértices com bipartição em u que se conectam a todos os vértices com bipartição em v")

    result = bipartidoCompleto(u, v, matriz)
    return result

else:
    return ("O grafo não é bipartido, pois possui vértices que se conectam a vértices adjacentes\n")
```

Figura 9

Por fim, os resultados são impressos no terminal, como nos exemplos abaixo:

```
***** RESULTADOS *****

O grafo não é bipartido, pois possui vértices que se conectam a vértices adjacentes

*****

----- INTEGRANTES -----
Nome: Lucca Bonsi Guarreschi / R.A:22.120.016-5
Nome: Vitor Martins Oliveira / R.A:22.120.067-8
Nome: João Pedro Rosa Cezarino / R.A: 22.120.021-5
Turma: 010
```

Figura 10

```
***** RESULTADOS *****

O grafo não é bipartido, pois possui vértices que se conectam a vértices adjacentes

*****

----- INTEGRANTES -----
Nome: Lucca Bonsi Guarreschi / R.A:22.120.016-5
Nome: Vitor Martins Oliveira / R.A:22.120.067-8
Nome: João Pedro Rosa Cezarino / R.A: 22.120.021-5
Turma: 010
```

Figura 11

```
***** RESULTADOS *****

O grafo é bipartido, e possui bipartições em u = {V1, V3, V5, V7} e v = {V2, V4, V6, V8}
O grafo não é bipartido completo, pois não são todos os vértices com bipartição em u que se conectam a todos os
vértices com bipartição em v

*****

----- INTEGRANTES -----
Nome: Lucca Bonsi Guarreschi / R.A:22.120.016-5
Nome: Vitor Martins Oliveira / R.A:22.120.067-8
Nome: João Pedro Rosa Cezarino / R.A: 22.120.021-5
Turma: 010
```

Figura 12