Grafos

Prática

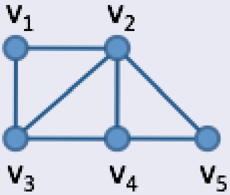
Profs. Andre Gustavo, Salles Magalhaes

Departamento de Informática
Universidade Federal de Viçosa

INF 333 - 2024/1

Matriz de adjacência

A matriz de adjacência de G é a matriz $\mathbf{A}(G) = [a_{ij}]$, sendo a_{ij} o número de arestas unindo os vértices v_i e v_j



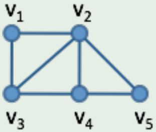
	v_1	v_2	v_3	v_4	v_5
v_1	0	1	1	0	0
v_2	1	0	1	1	1
v_3	1	1	0	1	0
v_4	0	1	1	0	1
v_5	0	1	0	1	0

- Para grafos simples, sem loops e múltiplas arestas, pode-se usar uma matriz de booleanos.

Exercício #1

Ler um grafo e armazená-lo numa matriz de adjacência.
Em seguida imprimir os adjacentes de cada vértice.

Exemplo



Entrada

```
5 7
1 2
3 2
3 1
4 5
4 3
2 4
5 2
```

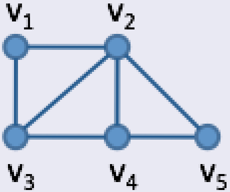
Saída

```
1: 2 3
2: 1 3 4 5
3: 1 2 4
4: 2 3 5
5: 2 4
```

- a primeira linha indica o número de vértices (5) e arestas (7)
- as 7 linhas seguintes indicam as arestas (não direcionadas)

Lista de adjacência

A lista de adjacência de G é uma lista de listas lineares, uma para cada vértice, contendo seus vértices adjacentes



v_1	v_2	v_3	
v_2	v_1	v_3	v_4
v_3	v_1	v_2	v_4
v_4	v_2	v_3	v_5
v_5	v_2	v_4	

Exercício #2

Ler um grafo e armazená-lo numa lista de adjacências.
Em seguida imprimir os adjacentes de cada vértice.

Exemplo

- teste com o mesmo exemplo do exercício anterior
- os resultados devem ser os mesmos para as duas estruturas (se necessário, ordene as listas)

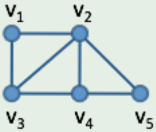
```
BP (Grafo G, vertice v, bool visitado[ ]) {
    visitado[v] = true
    escrever(v), processar(v), ...
    para cada w adjacente a v
        se !visitado[w]
            BP(G,w,visitado)
}
```

- *G*: grafo a ser percorrido
- *v*: vértice inicial da busca
- *visitado*[ ]: array de booleanos, inicialmente todos *false*

Exercício #3

Ler um grafo e armazená-lo numa lista de adjacências.
Ler um vértice e fazer uma busca em profundidade a partir dele, imprimindo os vértices na ordem em que são encontrados.

Exemplo



Entrada

```
5 7    //#v #a
1 2    //arestas
3 2
3 1
4 5
4 3
2 4
5 2
1      //inicio da busca
4      //inicio da busca
0      //fim
```

Saída

```
1: 1 2 3 4 5
4: 4 2 1 3 5
```

Busca em largura (a partir de um dado vértice)

```
BL (Grafo G, vertice v, bool visitado[]) {
    Fila F
    F.insere(v)
    visitado[v] = true
    enquanto !F.vazia()
        v ← F.remove()
        escrever(v), processar(v), ...
        para cada w adjacente a v
            se !visitado[w]
                F.insere(w)
                visitado[w] = true
}
```

Exercício #4

Ler um grafo e armazená-lo numa lista de adjacências.
Ler um vértice e fazer uma busca em largura a partir dele, imprimindo os vértices na ordem em que são encontrados.

Exemplo

- teste com o mesmo exemplo do exercício anterior
- a saída será:

```
1: 1 2 3 4 5
4: 4 2 3 5 1
```

- veja a diferença na ordem de visita das diferentes buscas

Exercício #5

Ler um grafo e armazená-lo numa lista de adjacências. Fazer uma busca contando o número de componentes do grafo.

Dica: a busca identifica um componente; uma nova busca a partir de um vértice não visitado identifica um novo componente.

Exercício #7

Ler um grafo e armazená-lo numa lista de adjacências. Fazer uma busca para verificar se o grafo é ou não bipartido.

Dica: “colorir” os vértices durante a busca: colorir o inicial com cor 1; os adjacentes de vértice de cor 1 são coloridos com cor 2 e vice-versa; se algum adjacente já estiver colorido com a mesma cor não é bipartido; as cores representam as partições.

Exercício #8

No exercício anterior, imprimir as partições do grafo.

Exercício #9

Resolva os problemas disponiveis no Vjudge (para esta pratica)