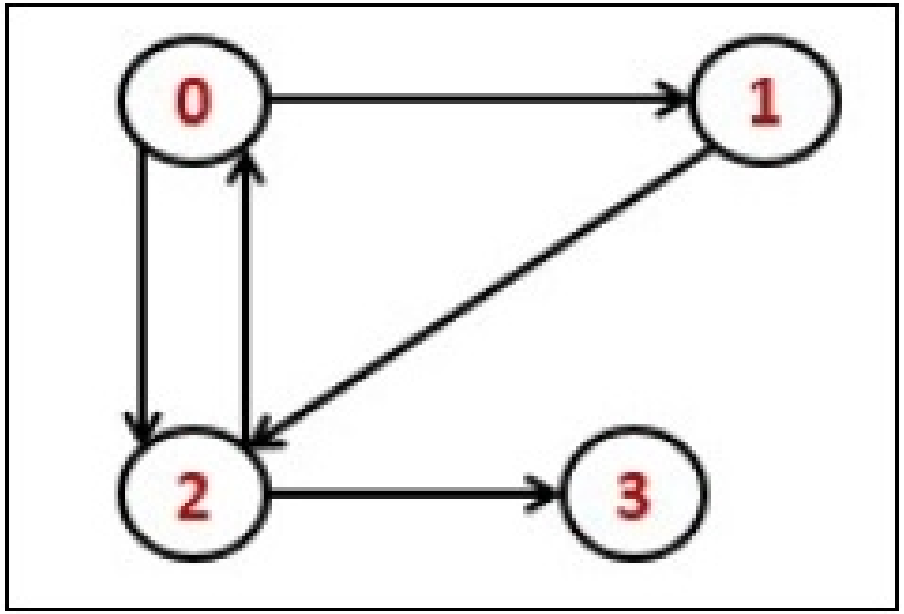
Conceitos de Grafos e Representação em Memória

ANDRÉ KAWAMOTO

Fechos

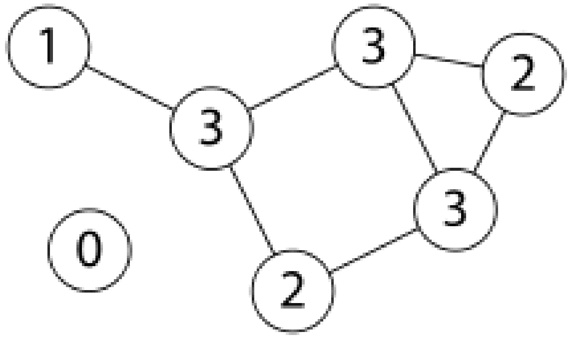


- Fecho Transitivo Direto - FTD
  - Conjunto de vértices que pode ser atingido a partir de um vértice V.
  - Note que o próprio vértice pertence ao FTD

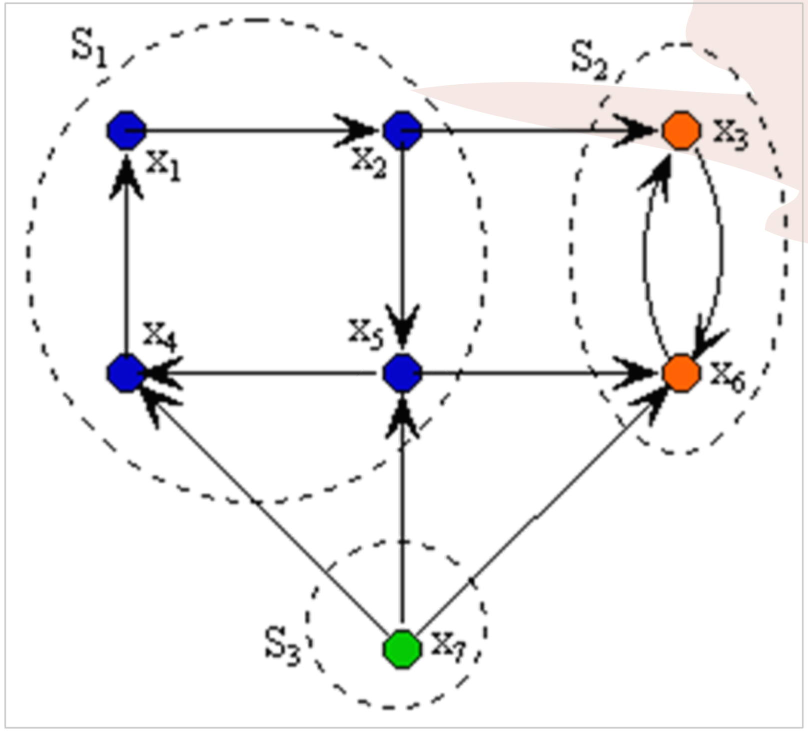
- Fecho transitivo inverso – FTI
  - Conjunto de vértices a partir dos quais é possível atingir um vértice v
  - O próprio vértice pertence ao FTI

Um grafo (não orientado)

- Um grafo é conexo quando existe um caminho entre cada par de vértices.
- Caso não exista, o grafo é desconexo



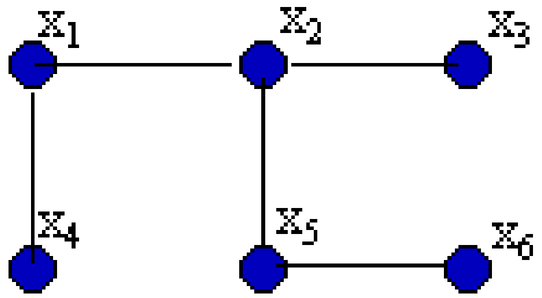
Um DiGrafo (grafo orientado)



- Um Grafo Orientado é Fortemente Conexo se ele contém um caminho entre todo par de vértices
- Se não for Fortemente Conexo, há **pelo menos** dois **subgrafos** fortemente conexos, disjuntos em relação aos vértices e maximais em relação à inclusão.
- Cada um destes subgrafos é dito ser uma **componente fortemente conexa** de G

Árvore

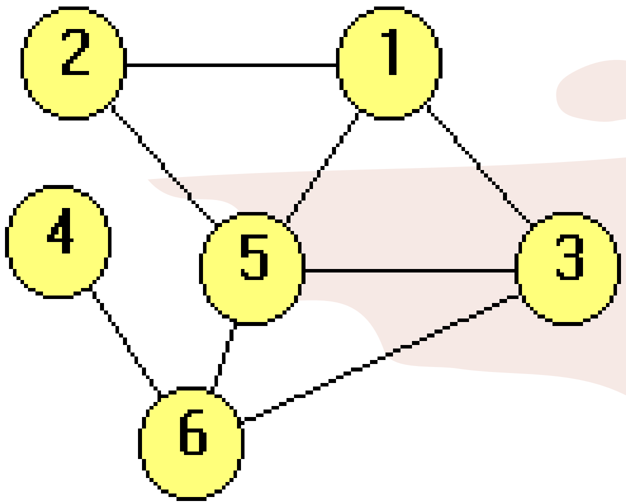
- Árvore é um grafo conexo e sem ciclos, logo:
  - G possui n-1 arestas;
  - A adição de uma aresta exatamente um ciclo;
  - A remoção de uma aresta torna o grafo desconexo (todas as arestas são pontes);
  - todo par de vértices de G é unido por uma e somente um caminho simples

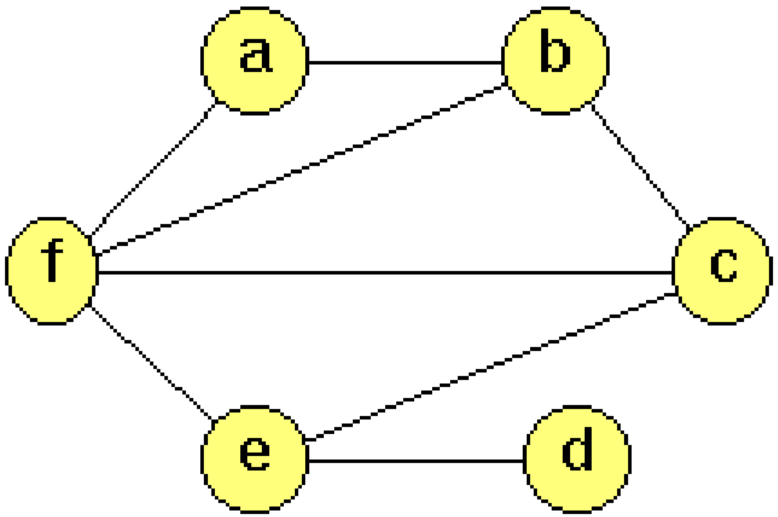


Isomorfismo

- Sejam dois grafos
  - $G_1(V_1,A_1)$
  - $G_2(V_2,A_2)$

- Um **isomorfismo** de G_1 sobre G_2 é um mapeamento bijetivo
  - $f: V_1 \leftrightarrow V_2$
  - tal que a aresta $(x,y) \in A_1$ se e somente se a aresta $(f(x),f(y)) \in A_2$, para todo $x,y \in V_1$.





Isomorfismo

- f = { (a,2), (b,1), (c,3), (d,4), (e,6), (f,5) }

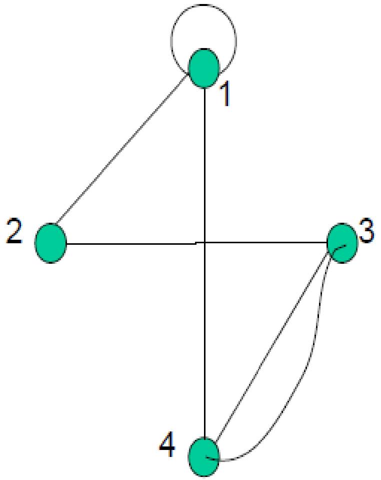
Representação de Grafos em Memória

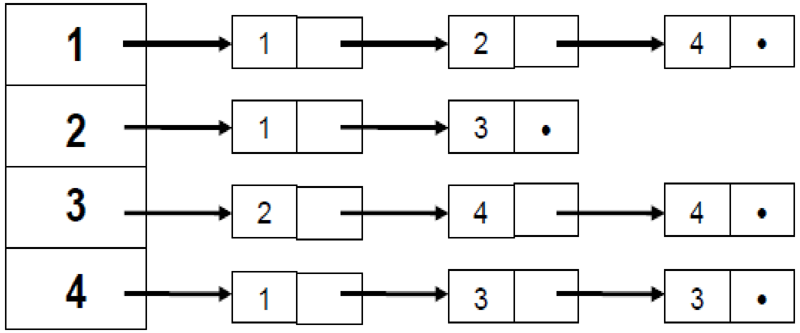
- Um grafo pode ser armazenado em memória de diversas maneiras, as formas mais comuns são:
  - Listas
  - Matriz

- A estrutura escolhida obviamente influencia a eficiência de algoritmos, por isso a análise de eficiência é feita em função da cardinalidade dos conjuntos V e E

Lista de Adjacência

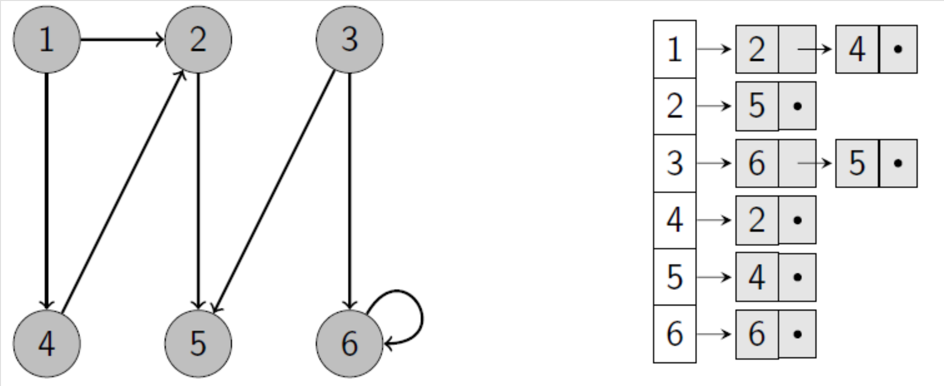
- Um *array* de tamanho |V|, em que cada posição correspondente a um dos vértices do grafo
- Cada posição do array aponta para uma lista com todos os vértices adjacentes ao vértice daquela posição

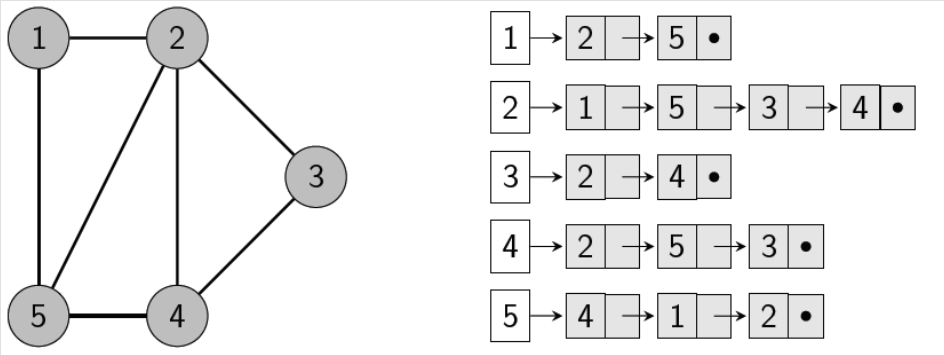




Lista de Adjacência

- Exemplos para grafos orientados e não orientados





Considerações Sobre Listas de Adjacência

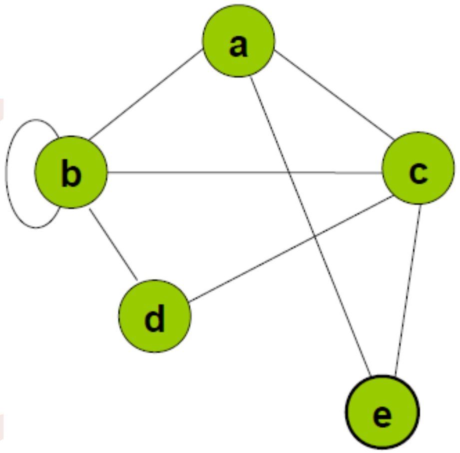
- A lista associada a um vértice pode ser vazia.
- Em grafos não orientados, pode-se evitar a repetição na representação de arestas adotando-se algum critério de ordenação
- Adequada na representação de grafos esparsos (a quantidade de arestas é bem menor que a de vértices)
- Ineficiente na busca de uma aresta no grafo
- Adaptável para grafos valorados

Matriz de Adjacência

- Uma matriz A de |V| x |V|
- Os vértices são associados às linhas e às colunas da matriz
- Cada elemento da matriz indica se existe ou não aresta entre os vértices
- Matriz de adjacência
  - A[i][j] = 1 , se existe aresta entre vértices *i* e *j*
  - A[i][j] = 0 , caso contrário

Matriz de Adjacência

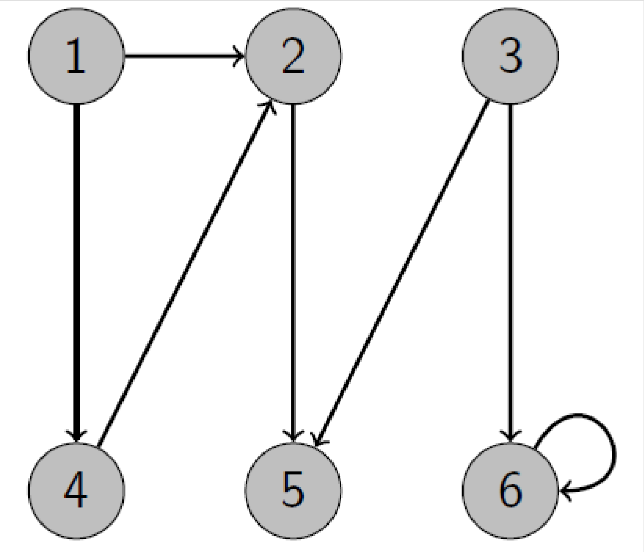
- Em um grafo, a matriz é simétrica
- A diagonal do grafo é preenchida com 0's – exceto se existir loop
- A quantidade de 1's na matriz é 2 * |E|



	a	b	c	d	e
a	0	1	1	0	1
b	1	1	1	1	0
c	1	1	0	1	1
d	0	1	1	0	0
e	1	0	1	0	0

Matriz de Adjacência

- A matriz não é simétrica
- As colunas da linha “i” indicam em quais vértices o elemento “i” incide
- As linhas da coluna “j” indicam quais vértices incidem sobre o elemento “j”



$$A = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

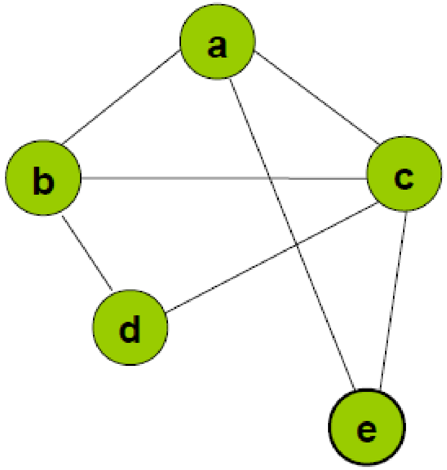
Considerações sobre Matriz de Adjacência

- Ocupam mais memória
- Preferível quando o grafo é denso - $|E|$ é próximo de $|V|^2$
- Útil para verificar se há aresta entre 2 vértices rapidamente (tempo constante)

Matriz de Incidência

- Matriz A de tamanho |V| x |E|
  - A[i][j] = 1, se vértice *i* incide sobre aresta *j*
  - A[i][j] = 0, caso contrário

Matriz de Incidência



	{a,b}	{a,c}	{a,e}	{b,c}	{b,d}	{c,d}	{c,e}
a	1	1	1	0	0	0	0
b	1	0	0	1	1	0	0
c	0	1	0	1	0	1	1
d	0	0	0	0	1	1	0
e	0	0	1	0	0	0	1

Considerações Sobre Matriz de Incidência

- São matrizes esparsas que exigem bastante espaço para armazenamento
- Não são muito utilizadas, salvo em alguns casos específicos de programação inteira e hipergrafos

Qual representação usar?

- Em grafos com muitos vértices e poucas arestas, a matriz de adjacência fica esparsa
- Se optar por usar matriz de adjacência, em grafos não-orientados é possível utilizar apenas a diagonal superior ou inferior
- O uso de listas tem um custo adicional para verificar incidência/adjacência entre 2 vértices
- Em grafos pequenos, matrizes podem ser vantajosas
- Ambas representações podem ser adaptadas para representar grafos valorados/rotulados

Para pensar

- Quanto tempo para calcular o grau de saída de um vértice
  - Usando matriz de adjacência
  - Usando lista de adjacência

- E o grau de entrada?
- Sugira 1 aplicação prática para o Fecho Transitivo (responder no moodle)

Atividade moodle

- Criar uma classe base para grafos