Algoritmos em Grafos

Prof. Fábio Dias

18 de novembro de 2024

UFC CAMPUS QUIXADÁ

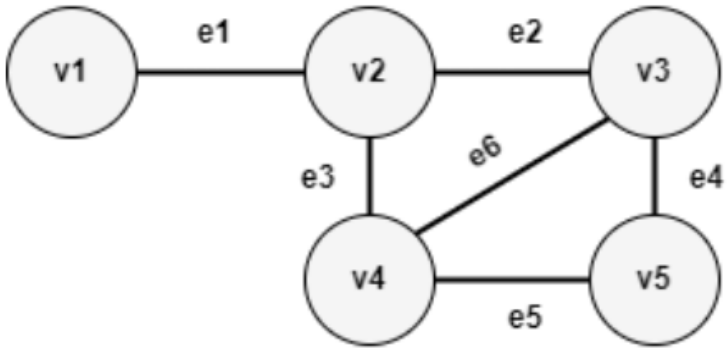
Sumário

UFC CAMPUS QUIXADÁ

Definição

Um grafo $G = (V, E)$ é definido pelo par de conjuntos V e E, onde:

- V - conjunto não vazio: os vértices ou nós do grafo;
- E - conjunto de pares ordenados (v, w), v e $w \in V$: as arestas do grafo.



Dado dois vértices do grafo, irá existir uma aresta entre eles se houver uma ligação entre esses vértices.

PROFESSOR, QUE TIPO DE LIGAÇÃO???? 🤔

UFC CAMPUS QUIXADÁ

Definição

Figura: Facebook: Relações entre pessoas.

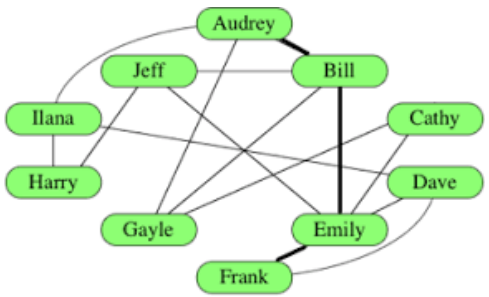
Definição

Figura: Problema do caminho mínimo: vértices locais, arestas entre esses locais.

Exemplos

Vamos definir matemáticamente o grafo $G = (V, E)$ do Facebook:

- $V = \{p : p \text{ é uma pessoa } \}$
- $E = \{(v, w) :< v \text{ é amigo de } w >\}$

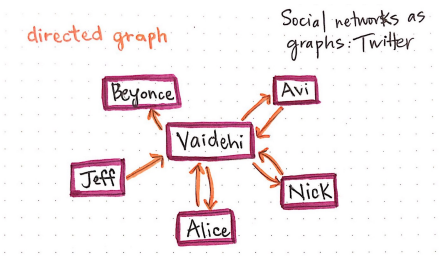


- Cada aresta será representada pelos par de vértices que ela é **incidente**;
- Quando a relação entre os vértices são simétrica, ou seja, v é amigo de w SSE w é amigo de v, dizemos que o grafo é não direcionado.

Grafos Direcionados

- $V = \{p : p \text{ é uma pessoa } \}$
- $E = \{(v, w) :< v \text{ segue } w >\}$

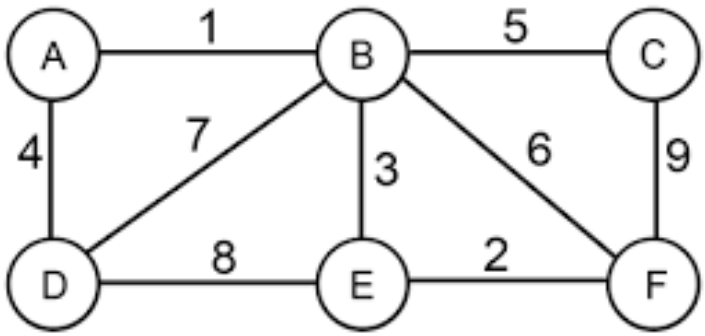
Nesse caso, se existir o arco (aqui as arestas são chamadas de arcos) (v, w), pode não existir o arco (w, v).



Nesse caso, chamamos de grafo direcionado ou orientado ou digrafo.

Grafos Ponderados

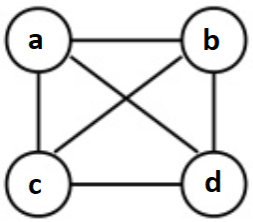
- Existe uma função $c : E \rightarrow R$;
- Para toda arestas $(v, w) \in E$, $c(v, w) \in R$ é o peso, custo, valor da aresta (v, w);

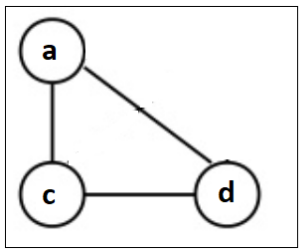


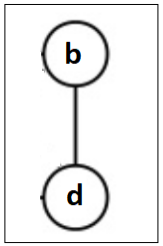
Em alguns caso, os vértices também podem ter um valor associado.

Definições

- Um subgrafo $G' = (V', E')$ de um grafo $G = (V, E)$ é uma parte do grafo, ou seja, $V' \subseteq V$ e $E' \subseteq E$:

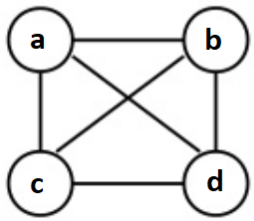


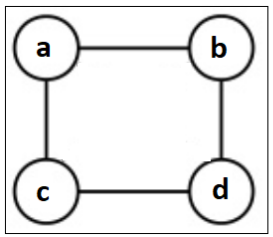


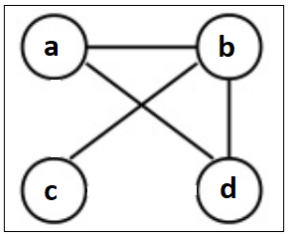


Definições

- Um subgrafo $G' = (V', E')$ é gerador quando $V' = V$ e $E' \subseteq E$:





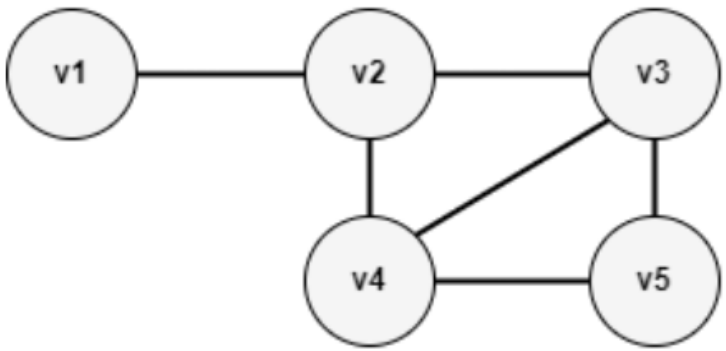


Caminho no Grafo

Passeio em grafo: Um passeio em um grafo é uma sequência de vértices dotada da seguinte propriedade: se v e w são vértices consecutivos na sequência, então (v, w) é uma aresta do grafo. Ex: $< v1, v2, v4, v2, v3 >$.

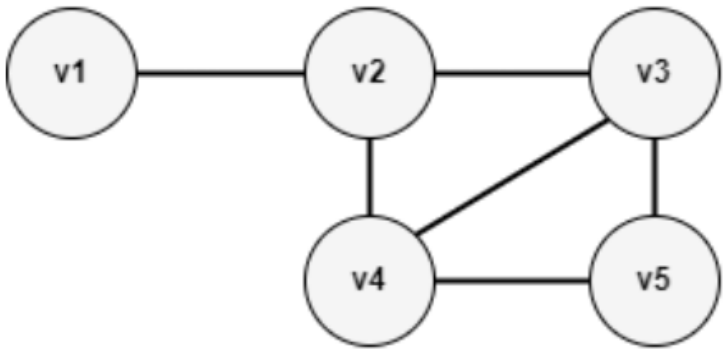
Caminho em grafo: Um caminho em um grafo é um passeio sem arestas repetidas, ou seja, é um passeio em que as arestas são todas diferentes entre si. Ex: $< v1, v2, v4, v3 >$.

Um Caminho Simples é um caminho sem repetição de vértices.



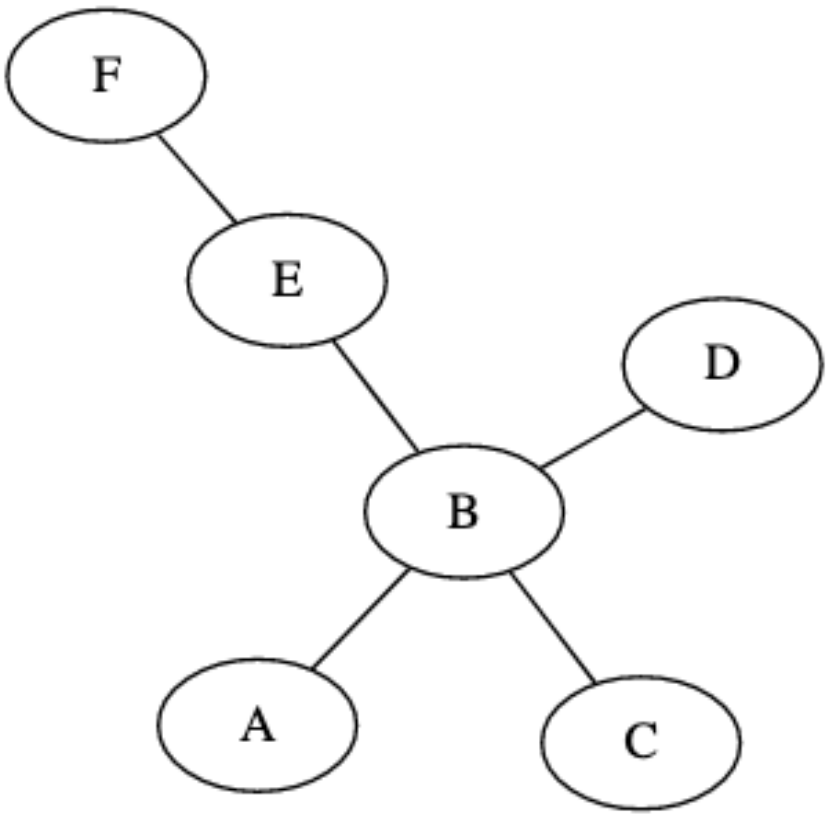
Ciclo no Grafo

Ciclo em grafos: Um ciclo é um caminho com origem e destino iguais.
Ex: $C_1 =< v2, v3, v5, v4, v2 >$ e $C_2 =< v2, v3, v4, v2 >$.



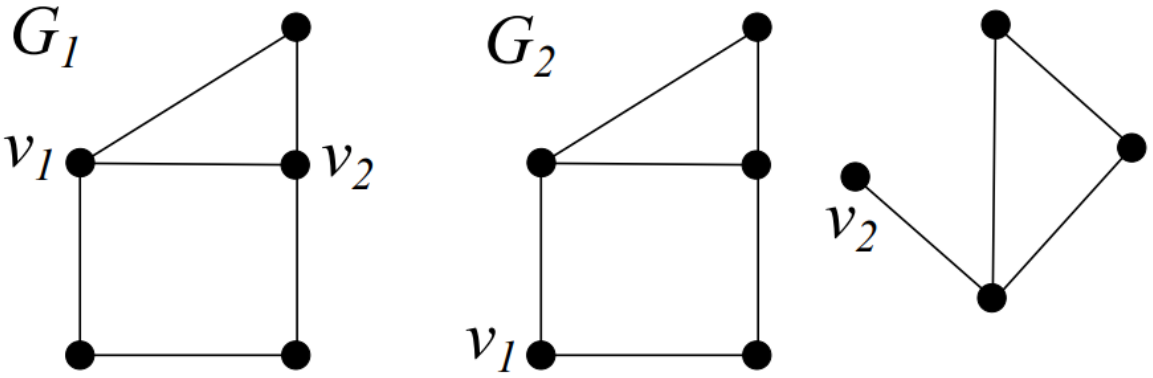
Grafo Acíclico

Grafos Acíclico: Um grafo que não contém ciclos.



Grafo Conexo e Desconexo

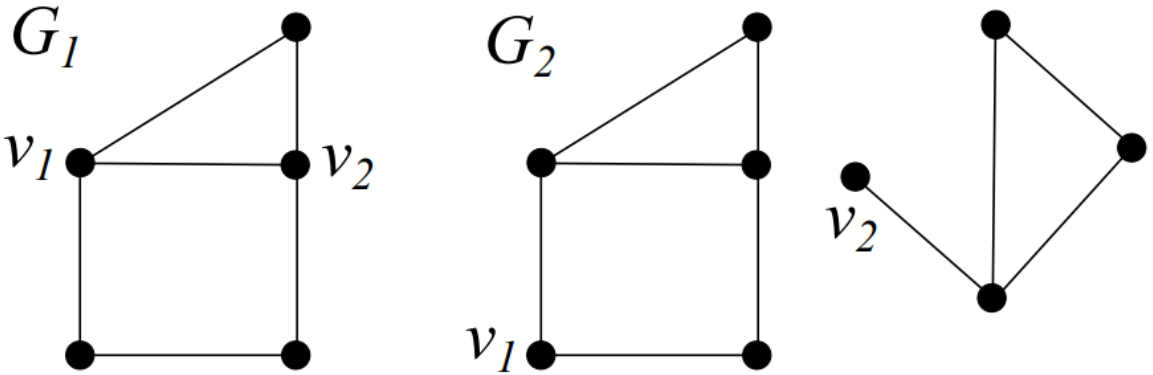
Considere os grafos abaixo:



E possível achar um caminho entre os vértices v_1 e v_2?

Grafo Conexo e Desconexo

Considere os grafos abaixo:



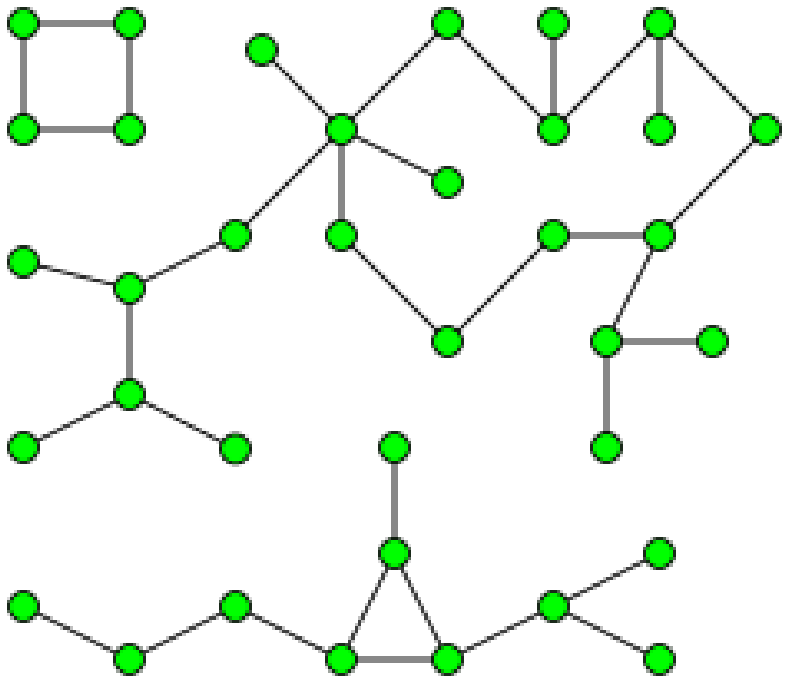
E possível achar um caminho entre os vértices v_1 e v_2?

Grafo Conexo: Um grafo é dito conexo se existir pelo menos um caminho entre cada par de vértices do grafo. Caso contrário, o grafo é chamado de desconexo.

O grafo G_1 acima é conexo, e o grafo G_2 é desconexo.

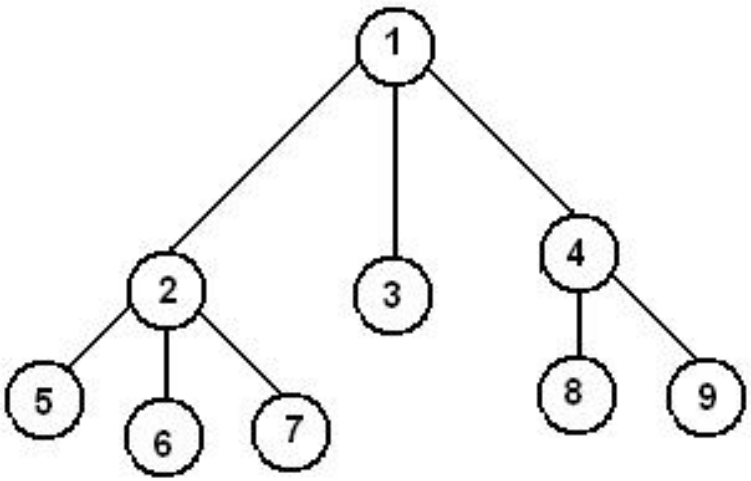
Componentes em Grafo Desconexo

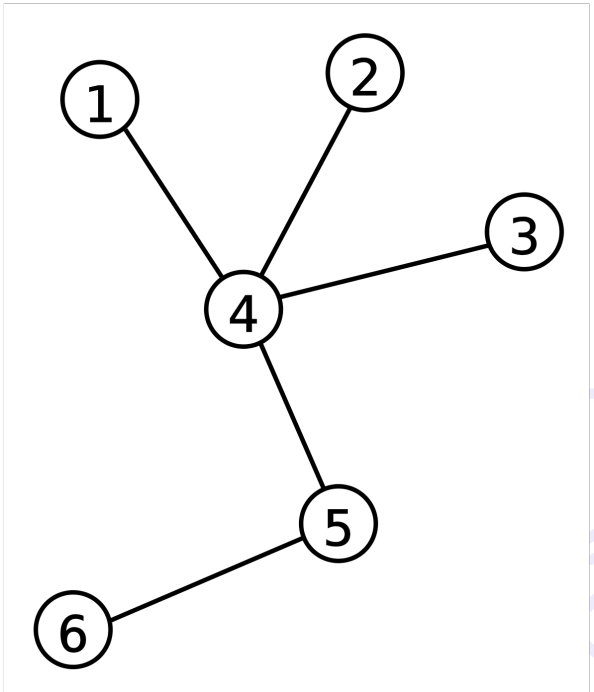
Cada um dos subgrafos conexos maximais de um grafo desconexo é chamado de uma **componente** do grafo. Ou seja, uma componente é um subgrafo conexo que não esteja estritamente contido em outros subgrafos conexos.



Árvores

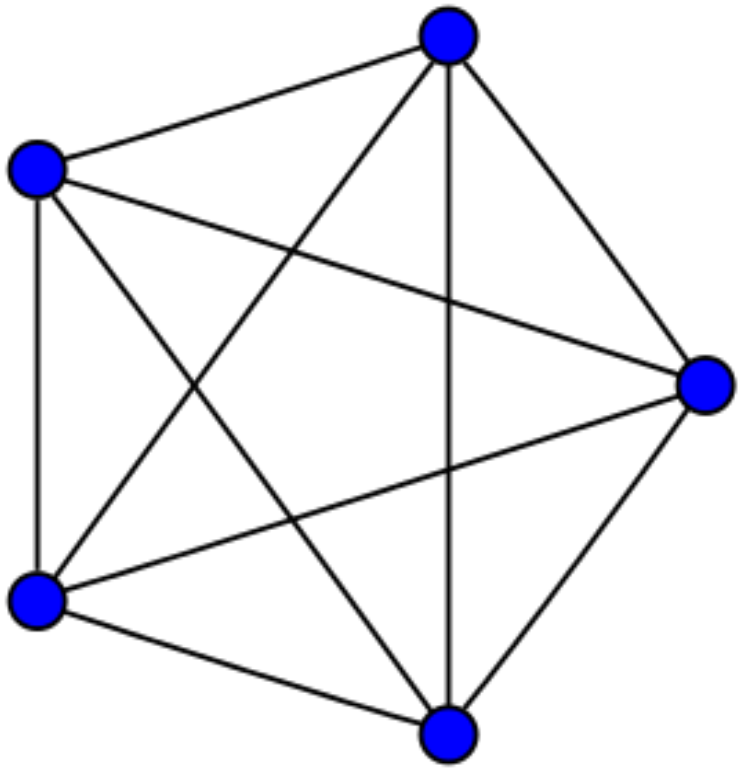
Uma árvore é um grafo conexo acíclico.





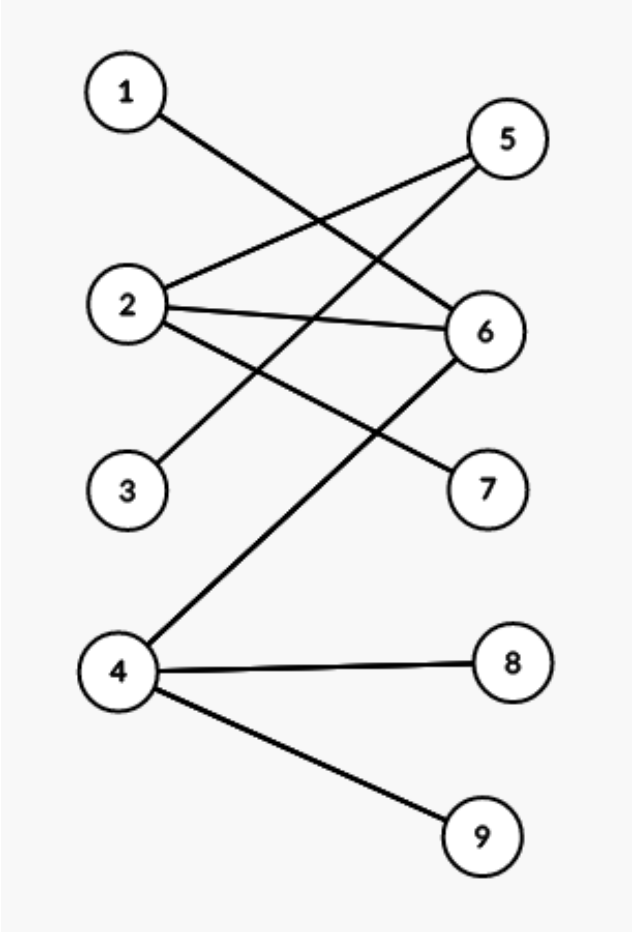
Grafo Completo

Um grafo é dito ser completo quando há uma aresta entre cada par de seus vértices.



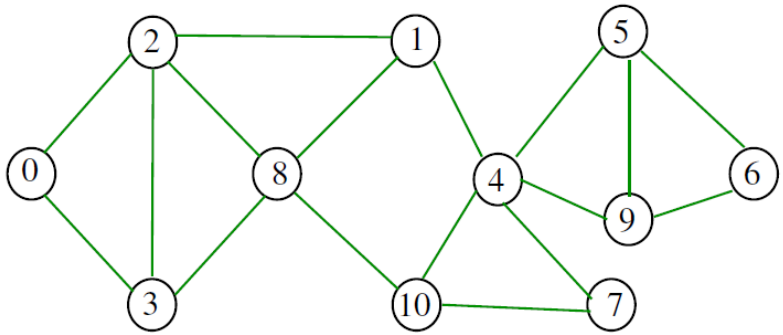
Grafo Bipartite

Acima tem um tipo de grafo chamado de Grafo Bipartidos: Grafos cujos vértices podem ser particionado em dois conjuntos disjuntos, tais que só existe arestas entre esse conjuntos;



Principais Propriedades em Grafos

- Dizemos que um vértice v é vizinho (adjacente) de w, se existe a aresta (v, w). Dizemos que a aresta (u, v) é *incidente* aos vértices u e v;
- Vizinhança de um vértice v ($N(v)$) é um subconjunto do conjunto de vértices V contendo os vizinhos de v;



- $N(2) = \{0, 1, 3, 8\}$, $N(4) = \{1, 5, 7, 9, 10\}$.
- *Grau* do vértice em grafos não direcionados é o número de arestas incidentes ao vértice, ou seja, $grau(v) = |N(v)|$.
- *Grau de entrada (saída)* de um vértice em um grafo direcionado é o número de arcos que entram (saem) dele.

Complexidade Computacional em Grafos

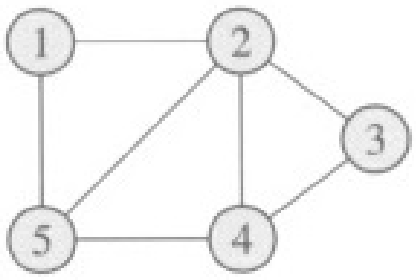
- O tempo de execução para um algoritmo que recebe a entrada sendo um Grafo, estará em função do tamanho de V e E, ou seja, $|V|$ e $|E|$;
- Um algoritmo com complexidade $O(VE)$, ou $O(V^2 \log E)$;
- Na complexidade, por simplicidade, não iremos colocar as $||$.

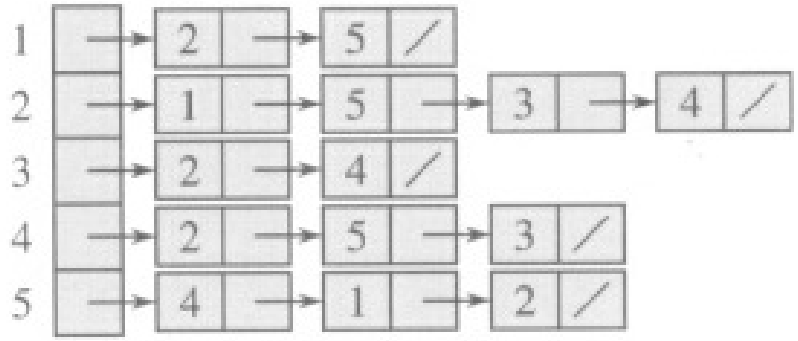
Representações de Grafos

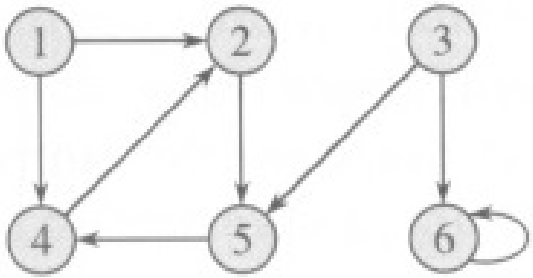
- Podemos representar um grafo, com uma estrutura que represente os conjuntos V e E;
- Para representar V, basta usamos a nomenclatura de índices: $1, ..., |V|$;
- As duas principais formas de representação de arestas são:
  - Lista de Adjacências;
  - Matriz de Adjacências.
- Qualquer desses modos se aplicar a grafos não direcionado quanto direcionado.

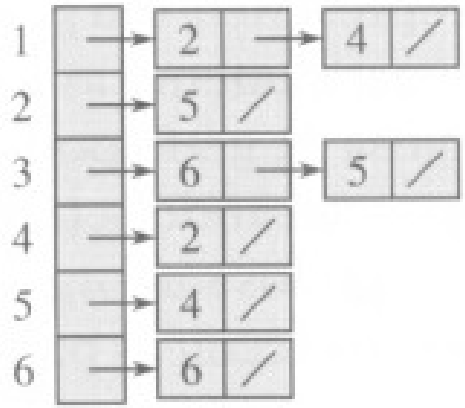
Lista de Adjacência

- O grafo consiste de um vetor de $|V|$ listas, uma para cada vértices;
- Para cada $v \in V$, a lista $adj[v]$ contém todos os vértices vizinhos de v;

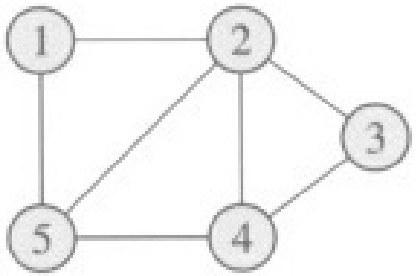


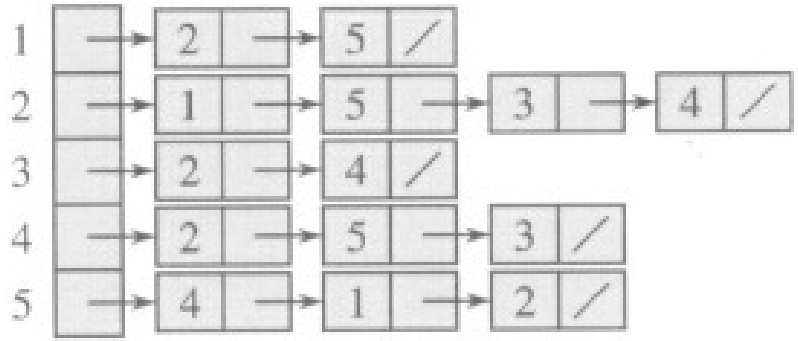






Lista de Adjacência



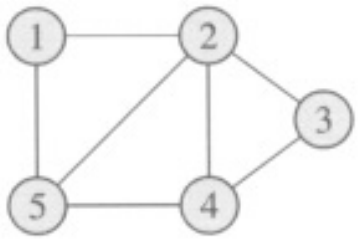



```cpp
#include <list>
...
int N = 5;
list<int> Adj[N + 1];
Adj[1].push_back(2);
Adj[1].push_back(5);
Adj[2].push_back(1);
Adj[2].push_back(5);
Adj[2].push_back(3);
Adj[2].push_back(4);
Adj[3].push_back(2);
Adj[3].push_back(4);
...
```

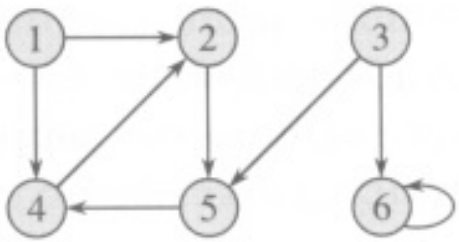
UFC CAMPUS QUIXADÁ

Matriz de Adjacência

- O grafo consiste de uma matriz de dimensão $|V|x|V|$, onde cada célula da matriz irá conter 1 ou 0;
- $matriz[v][w]$ será 1 se o vértice w é vizinho de v, 0 caso contrário;



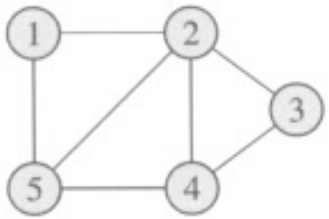
	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

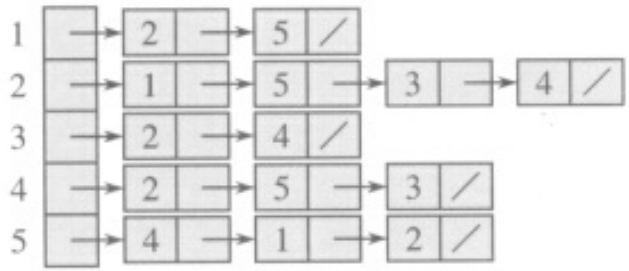


	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

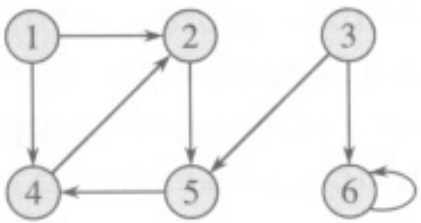
Para grafos ponderados, a matriz irá armazenar o valor da aresta.

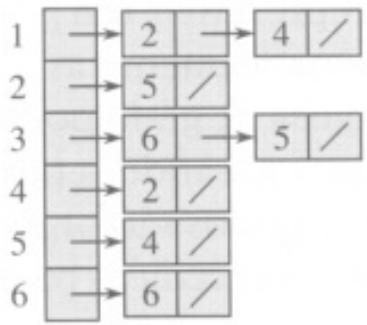
Ambas





	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0





	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

UFC CAMPUS QUIXADÁ

Qual é a melhor????

- Quem consome mais memória?
- Qual o custo para saber se um vértice w é vizinho de v?
- Qual o custo para listar todos os vizinho de um vértices?

UFC CAMPUS QUIXADÁ

Quem consome mais memória?

- Observe primeiro que $|E| = O(|V|^2)$;
- Em geral a Matriz de Adjacência consome mais memória:
  - Matriz: $\Theta(|V|^2)$ e Lista: $\Theta(|V| + |E|)$;
  - Principalmente para grafos esparsos;
- Grafos esparsos são aqueles para os quais $|E|$ é muito menor que $|V|^2$;
- Grafos densos são aqueles para os quais $|E|$ está próximo de $|V|^2$;
- Para grafos completos, ou seja, grafos com arestas entre todos os vértices, qual a diferença nas duas em relação a memória?
- Mas, em grafos não ponderados, podemos criar uma matriz de bit.

UFC CAMPUS QUIXADÁ

Qual o custo para saber se um vértice w é vizinho de v?

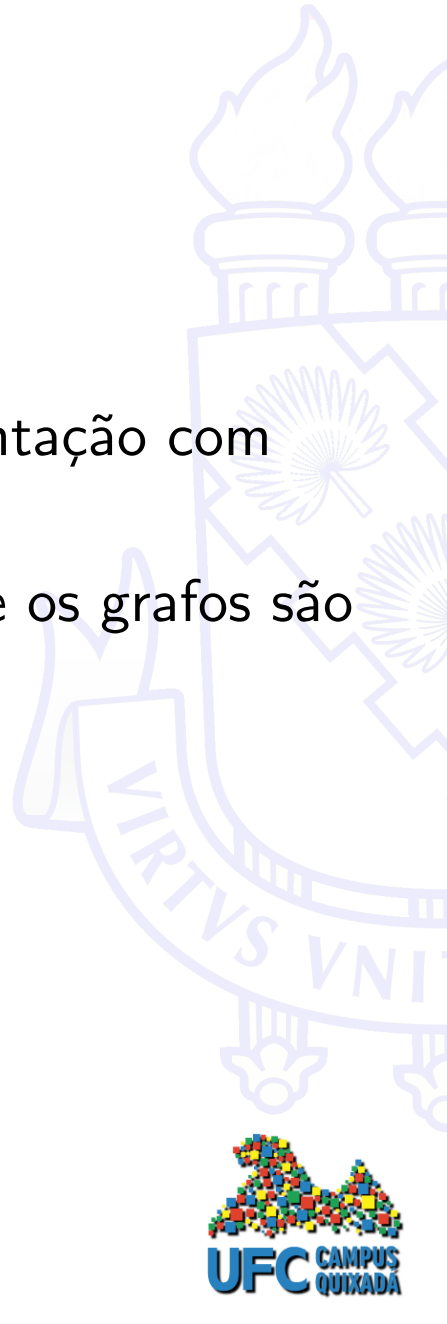
- Matriz de Adjacência: $O(1)$;
- Lista de Adjacência: $O(|N(v)|)$;
- Portanto, a Matriz de Adjacência é mais indicada em aplicações que precisamos saber rapidamente se há uma aresta conectando dois vértices fornecido;

Qual o custo para listar todos os vizinho de um vértices?

- Matriz de Adjacência: $O(|V|)$;
- Lista de Adjacência: $O(|N(v)|)$;
- Portanto, a lista de Adjacência é mais indicada em aplicações que normalmente precisamos saber todos os vizinhos de um vértices;

Qual a melhor?

- Depende da aplicação;
- A maioria dos algoritmos em grafos utilizam a representação com Lista de Adjacência;
- A matriz é mais simples e por isso, em aplicações onde os grafos são pequenos, ela é mais preferível;
- Claro, se memória não for problema, tenha as duas.



UFC CAMPUS QUIXADÁ

Perguntas?!