INSTITUTO FEDERAL
Ceará

CAA
Grafos

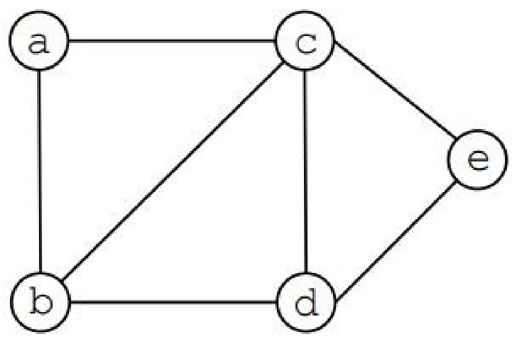
Paulo Amora

paulo.amora@ifce.edu.br

Grafos

- Def: Um Grafo é um par G(V,E) onde
  - V é um conjunto de elementos chamados **Vértices**
  - E é um conjunto finito de pares de vértices não-ordenados chamado **Arestas** (Edges)
- Ex: V={a,b,c,d,e}

  E={(a,b);(a,c);(b,c);(b,d);(c,d);(c,e);(d,e)}

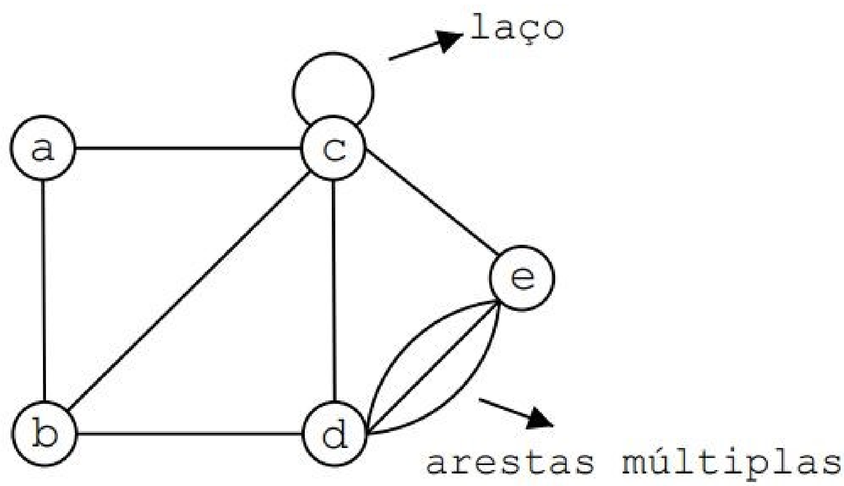


Grafos

- Dada uma aresta e = (a, b), dizemos que os vértices a e b são os **extremos** da aresta *e* e que *a* e *b* são vértices **adjacentes**.
- Dizemos também que a aresta *e* é **incidente** aos vértices *a* e *b*, e que os vértices *a* e *b* são incidentes à aresta *e*.

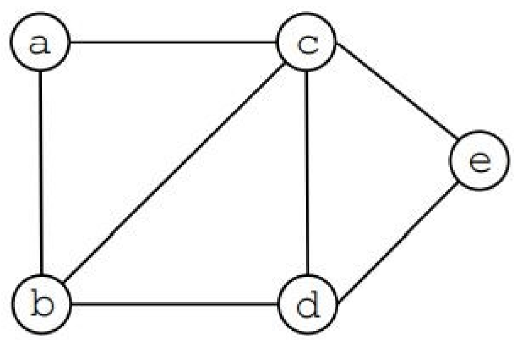
Grafos

- Dizemos que um grafo é **simples** quando não possui laços ou arestas múltiplas.
- Um **laço** é uma aresta com extremos idênticos e **arestas múltiplas** são duas ou mais arestas com o mesmo par de vértices como extremos.



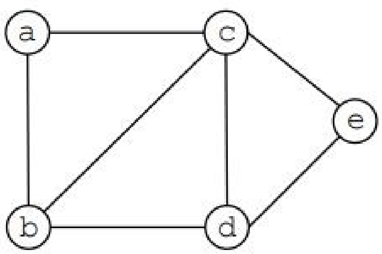
Grafos

- Denotamos por |V| e |E| a quantidade de elementos dos conjuntos de vértices e arestas de um grafo G, respectivamente. No exemplo abaixo temos |V| = 5 e |E| = 7.
- O tamanho do grafo G é dado por |V| + |E|.

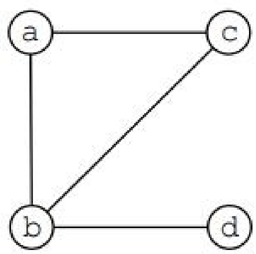


Grafos

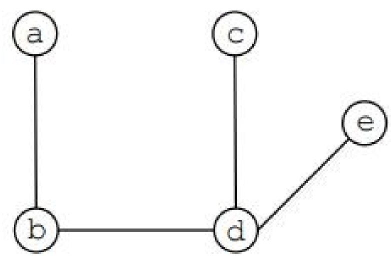
- Um **subgrafo** H = (V′,E′) de um grafo G = (V,E) é um grafo tal que $V' \subseteq V$, $E' \subseteq E$.
- Um **subgrafo gerador** de G é um subgrafo H com V′ = V.



Grafo G



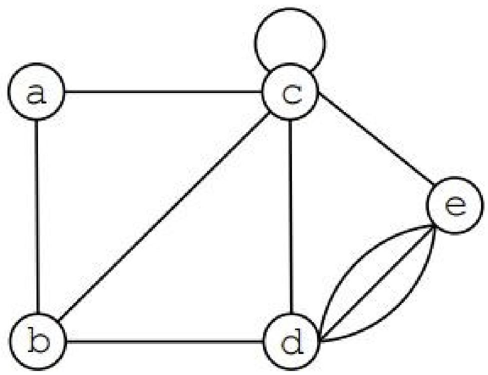
Subgrafo não gerador



Subgrafo gerador

Grafos

- O **grau** de um vértice v, denotado por d(v) é o número de arestas incidentes a v, com laços contados duas vezes.



d(a)=2

d(b)=3

d(c)=6

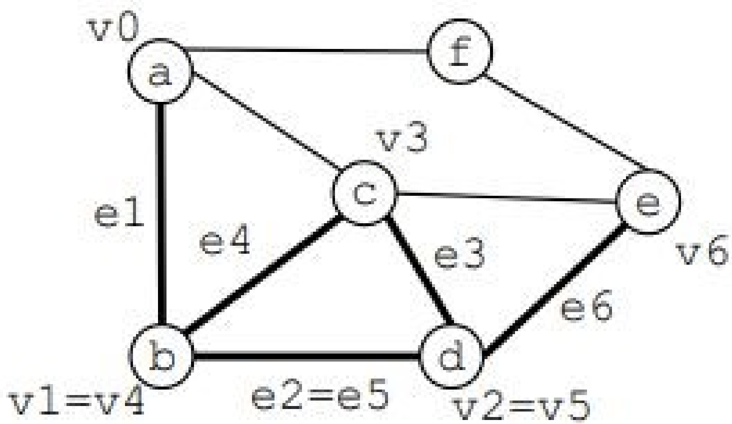
d(d)=5

d(e)=4

- Lema do aperto de mão:
  - Para todo grafo, temos que a soma de todos os graus é igual a 2x a quantidade de arestas

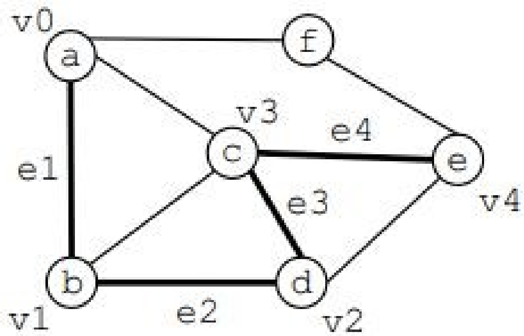
Grafos

- Um **caminho** P de v_0 a v_n no grafo G é uma sequência finita e não vazia (v_0, e_1, v_1, . . . , e_n, v_n) cujos elementos são alternadamente vértices e arestas e tal que, para todo $1 \leq i \leq n$, v_{i-1} e v_i são os extremos de e_i.
- O comprimento do caminho P e dado pelo seu número de arestas, ou seja, n

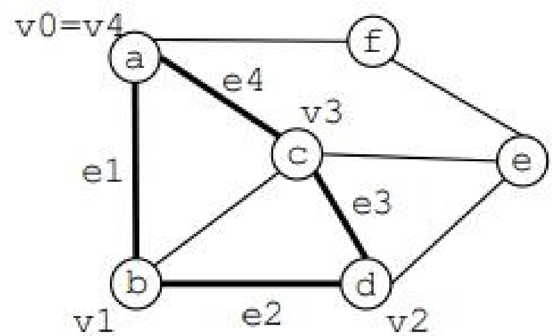


Grafos

- Um **caminho simples** é um caminho em que não há repetição de vértices e nem de arestas na sequência.
- Um **ciclo** ou **caminho fechado** é um caminho em que $v_0 = v_n$.



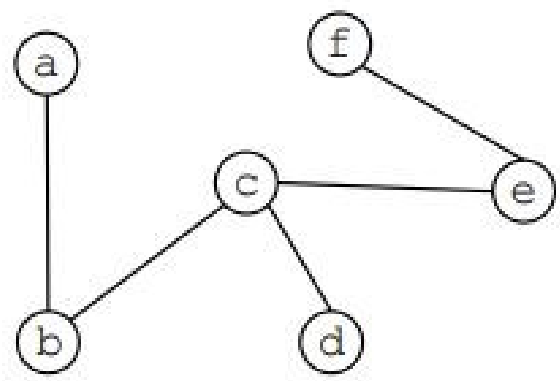
Caminho Simples



Ciclo

Grafos

- Um grafo G é uma árvore se é conexo e não possui ciclos (acíclico).
- As seguintes afirmações são equivalentes:
  - G é uma árvore.
  - G é conexo e possui exatamente $|V| - 1$ arestas.
  - G é conexo e a remoção de qualquer aresta desconecta o grafo (minimal conexo).
  - Para todo par de vértices (u, v) de G, existe um único caminho de u a v em G.

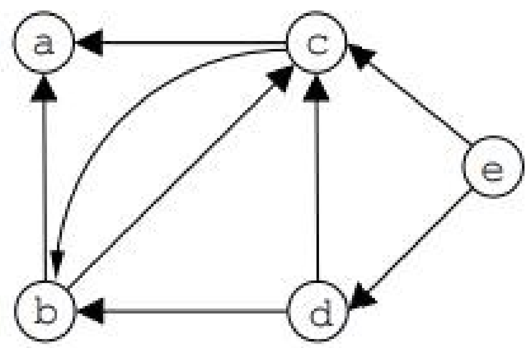


Grafos

- Ex:
- Floresta: grafo acíclico (não precisa ser conexo). Cada componente é uma árvore.
- Grafo completo: para todo par de vértices (u, v) a aresta (u, v) pertence ao grafo. Em outras palavras, todos os vértices são adjacentes
- Grafo bipartido: possui uma bipartição (A,B) do conjunto de vértices tal que toda aresta tem um extremo em A e outro em B.
- Grafo planar: pode ser desenhado no plano de modo que arestas se interceptam apenas nos extremos.

Grafos

- Grafos Orientados
- Todas as definições vistas são para grafos **não orientados**
- Um **grafo orientado** é definido de forma semelhante, com a diferença que as arestas (às vezes chamadas de arcos) consistem de pares ordenados de vértices



Grafos

- Se e = (u, v) é uma aresta de um grafo orientado G, então dizemos que e sai de u e entra em v.
- O grau de saída d+(v) de um vértice v é o número de arestas que saem de v. O grau de entrada d−(v) de v é o número de arestas que entram em v.

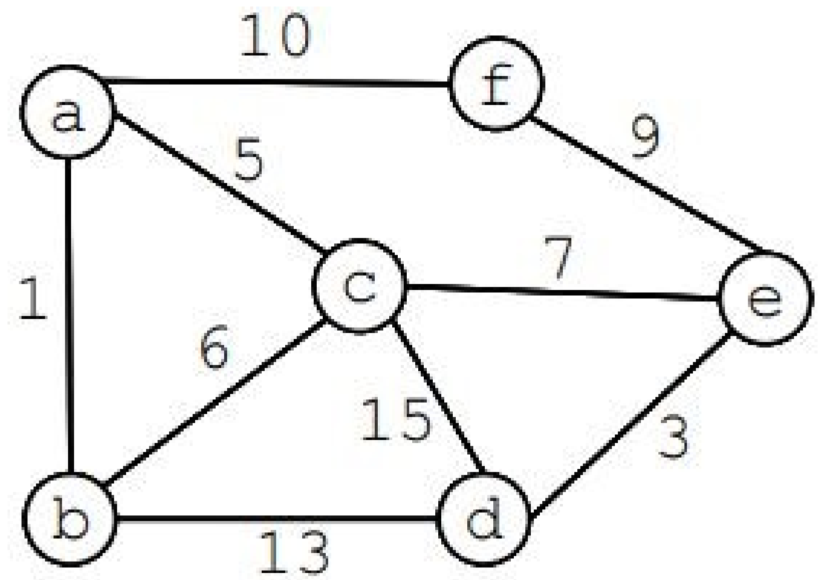
Teorema.

Para todo grafo orientado $G = (V, E)$ temos:

$$\sum_{v \in V} d^{+}(v) = \sum_{v \in V} d^{-}(v) = |E|.$$

Grafos

- Grafos ponderados
- Um grafo (orientado ou não) é ponderado se a cada aresta e do grafo está associado um valor real c(e), o qual denominamos custo (ou peso) da aresta.



Grafos

- Grafos são estruturas abstratas que podem modelar diversos problemas do mundo real.
- Por exemplo, um grafo pode representar conexões entre cidades por estradas ou uma rede de computadores.
- O interesse em estudar algoritmos para problemas em grafos é que conhecer um algoritmo para um determinado problema em grafos pode significar conhecer algoritmos para diversos problemas reais.

Grafos

- Caminho mínimo: dado um conjunto de cidades, as distancias entre elas e duas cidades A e B, determinar um caminho (trajeto) mais curto de A até B.
- Árvore Geradora de Peso Mínimo: dado um conjunto de computadores, onde cada par de computadores pode ser ligado usando uma quantidade de fibra ótica, encontrar uma rede interconectando-os que use a menor quantidade de fibra ótica possível.
- Emparelhamento máximo: dado um conjunto de pessoas e um conjunto de vagas para diferentes empregos, onde cada pessoa é qualificada para certos empregos e cada vaga deve ser ocupada por exatamente uma pessoa, encontrar um modo de empregar o maior numero possível de pessoas.

Grafos

- Problema do Caixeiro Viajante: dado um conjunto de cidades, encontrar um ciclo que sai de uma cidade, passa por todas as cidades e volta para a cidade inicial tal que a distância total a ser percorrida seja menor possível.
- Problema Chinês do Correio: dado o conjunto das ruas de um bairro, encontrar um caminho que passa por todas as ruas voltando ao ponto inicial tal que a distância total a ser percorrida seja menor possível.

Grafos

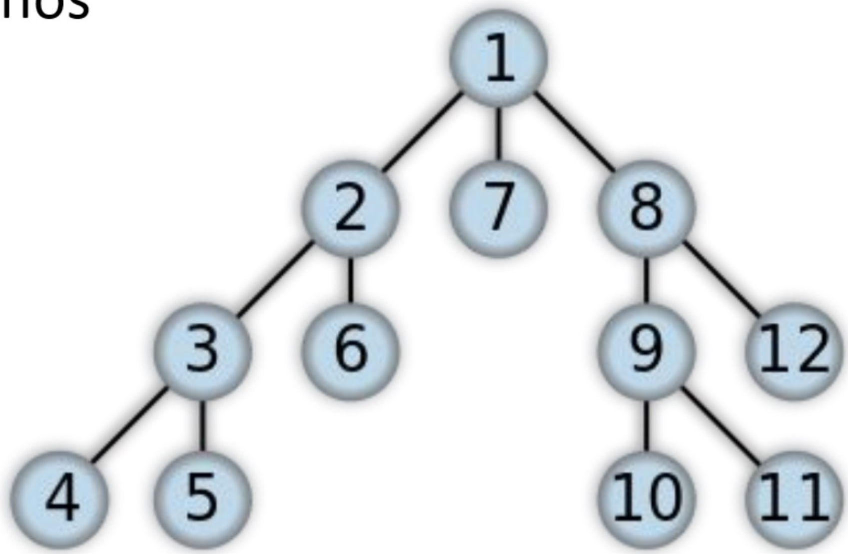
- Caminhamento
- Existem várias formas de percorrer um grafo, para poder por exemplo, detectar se há um ciclo ou descobrir se existe um caminho entre dois vértices
- Vamos trabalhar dois algoritmos em prática:
    - Busca em profundidade (DFS)
    - Busca em largura (BFS)

Grafos

- Busca em profundidade
- Percorre o grafo seguindo um caminho até o extremo e depois retornando para processar os outros vizinhos

Grafos

- Busca em profundidade
- Percorre o grafo seguindo um caminho até o extremo e depois retornando para processar os outros vizinhos



Grafos

- Busca em profundidade
- Para implementar busca em profundidade utilizamos uma estrutura de dados auxiliar (pilha)

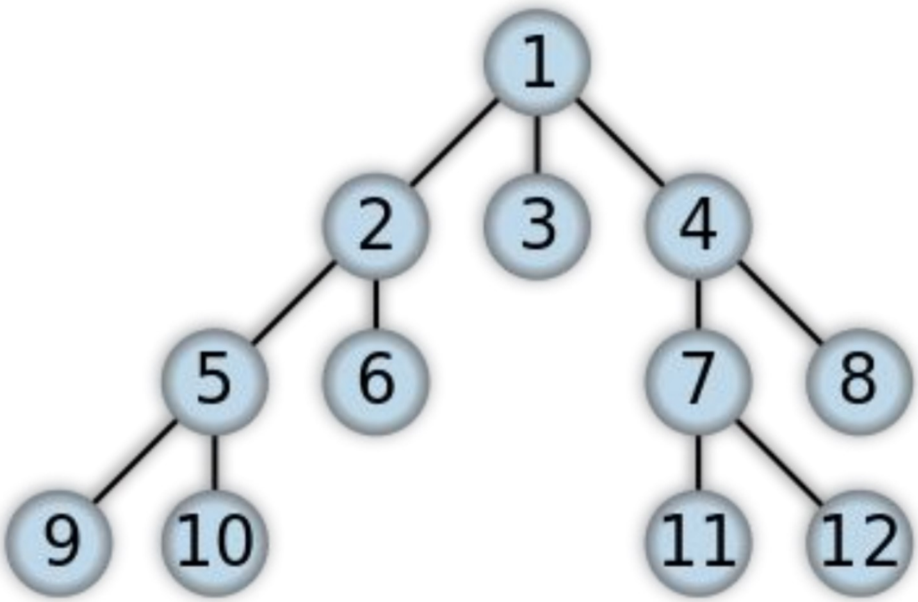
```
DFS (Vértice inicio):
    S é uma pilha
    S.push(inicio)
    enquanto S não é vazia
      v = S.pop()
      se v não foi visitado:
         visita(v)
         para cada aresta a ligada a v:
            S.push(a.w)
```

Grafos

- Busca em largura
- Percorre o grafo visitando os vizinhos e descendo após visitar todos os vizinhos

Grafos

- Busca em largura
- Percorre o grafo visitando os vizinhos e descendo após visitar todos os vizinhos



Grafos

- Busca em largura
- Para implementar busca em profundidade utilizamos uma estrutura de dados auxiliar (fila)

```
BFS (Vértice inicio):
    Q é uma fila
    Q.enfileira(inicio)
    enquanto Q não é vazia
      v = Q.desenfileira()
      visita(v)
      para cada aresta a ligada a v:
         se v não foi visitado:
            Q.enfileira(a.w)
```