Tópicos de Programação

Arthur Casals
(arthur.casals@usp.br)

IME - USP

Aula 8:

- Fundamentos de Estruturas de Dados

Na aula passada…

Grafos:

› O que são?

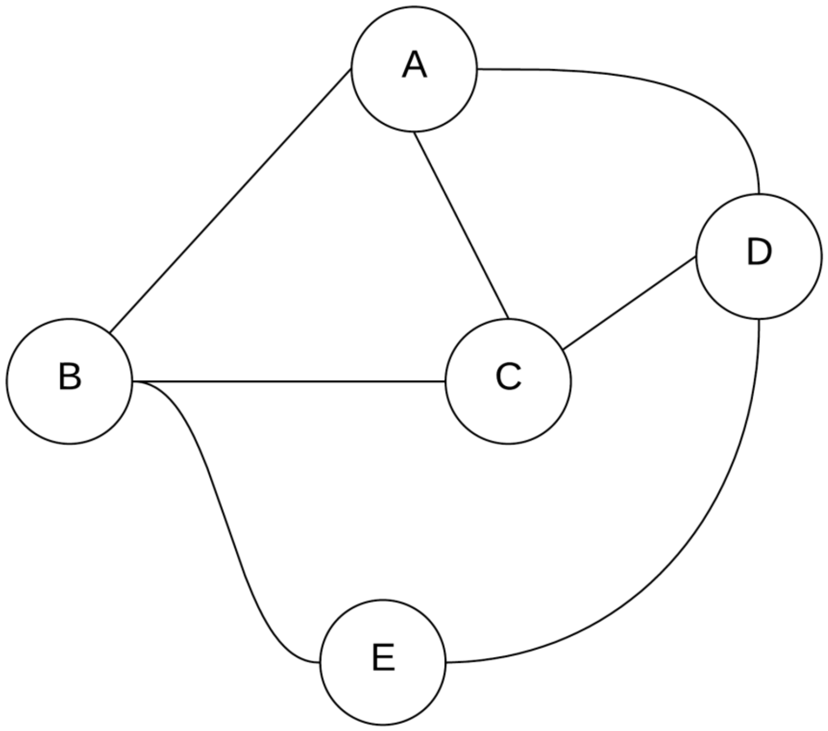
› Servem para quê?

* referência: ver material complementar sobre grafos, disponível na página do curso

Na aula passada...

Grafos:

- Um conjunto de vértices conectados por arestas



Na aula passada...

Grafos

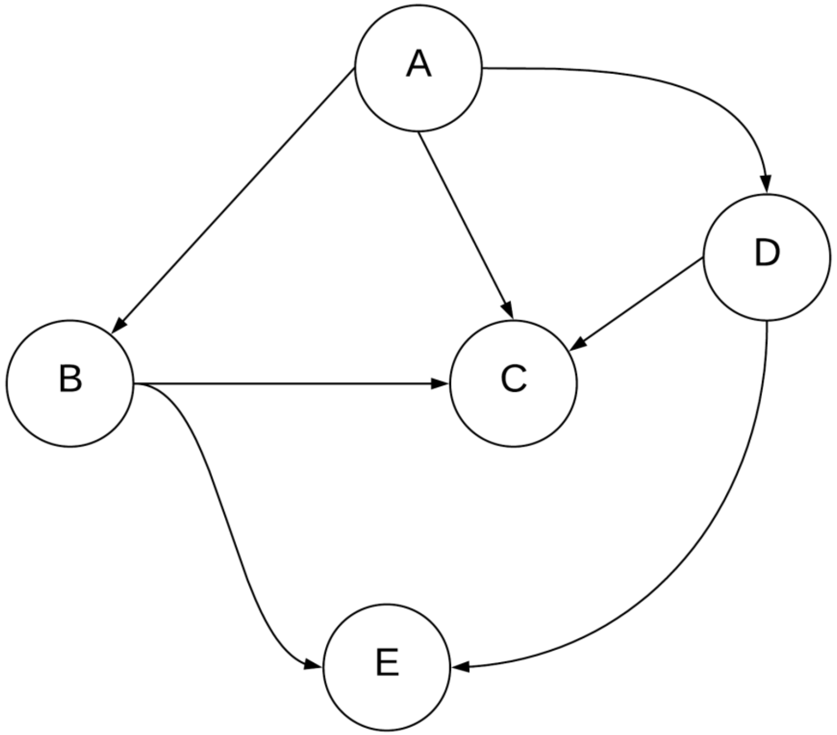
"Um grafo G(V, A) é uma estrutura matemática constituída pelos conjuntos:

- V, finito e não vazio de n vértices, e
- A, contendo m arestas, que são pares não ordenados de elementos de V"

Na aula passada...

Grafos:

- Se existir ordem nas arestas do grafo, este é chamado de **orientado, ou dirigido, ou digrafo.**



Na aula passada...

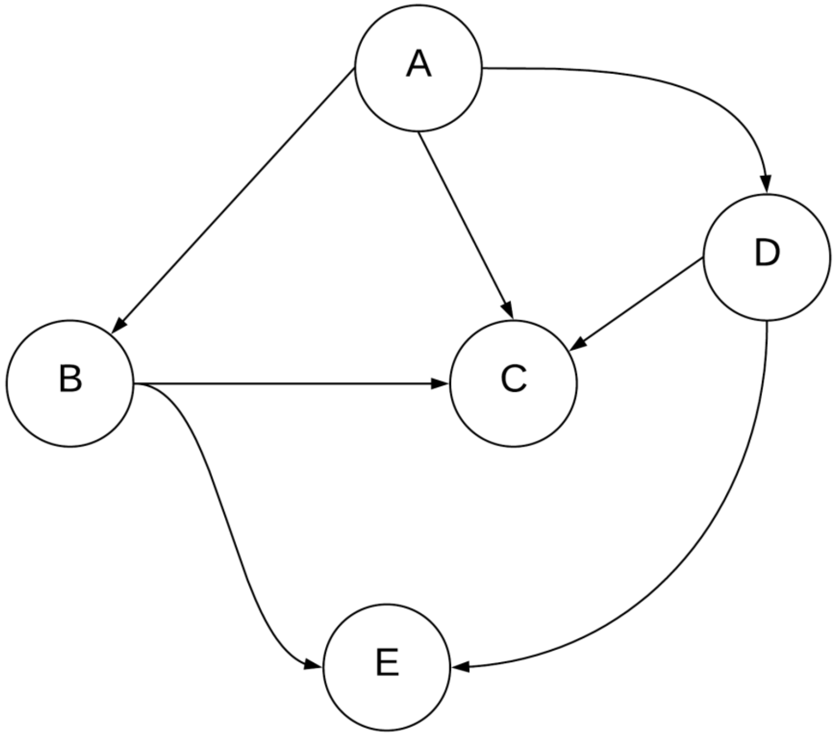
Grafos:

- **Vértices adjacentes:** são os vértices conectados por uma aresta
- **Grau de um vértice:** número de vértices adjacentes
  - Grau máximo: $\Delta(G)$
  - Grau mínimo: $\delta(G)$
- **Arco:** par ordenado de vértices (ponta inicial, ponta final)

Na aula passada...

Grafos:

- **Subgrafo:** Um subgrafo G' de um grafo G ($G' \subseteq G$) é qualquer grafo tal que:

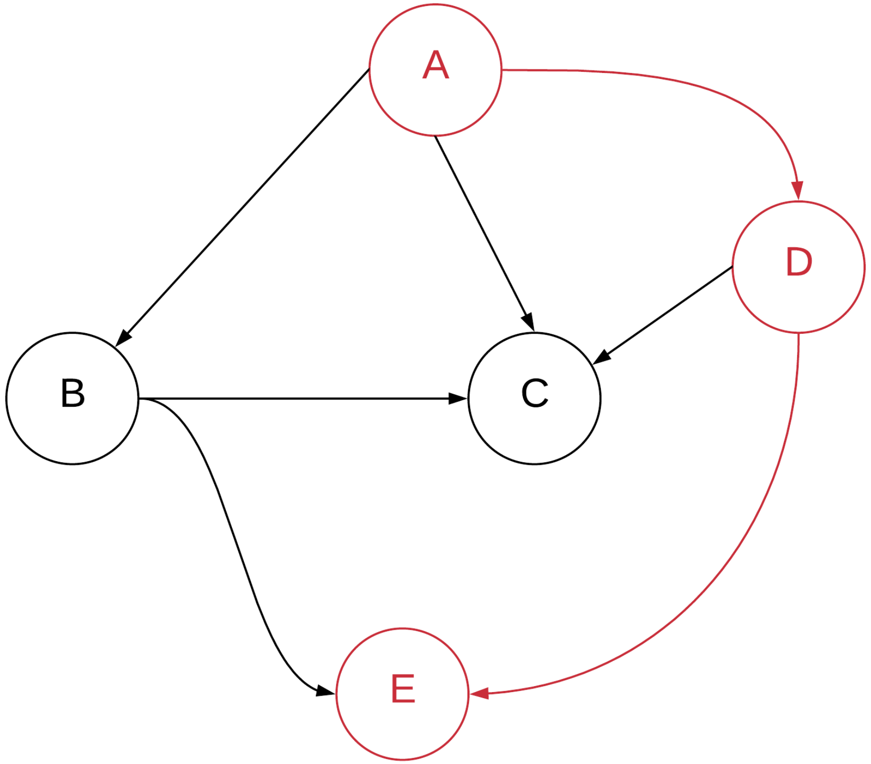


- $V_{G'} \subseteq V_G$
- $A_{G'} \subseteq A_G$
- $(v, w) \in A_{G'} \Rightarrow (v, w) \in V_{G'} x V_{G'}$

Na aula passada...

Grafos:

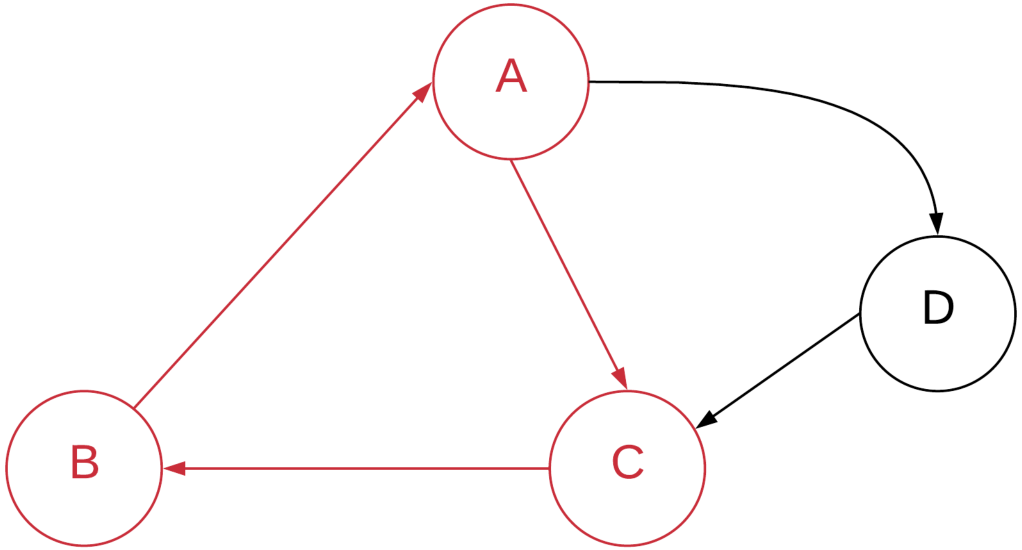
- **Passeio:** Qualquer sequência $(s_0, s_1, ..., s_k)$ tal que $(s_i, s_{i+1}) \in A_G$, $0 \leq i < k$



Na aula passada...

Grafos:

- **Caminho:** Qualquer passeio no qual nenhum dos vértices ocorre mais de uma vez



(A, C, B, A): ***<u>não é caminho!</u>***

Na aula passada...

Grafos:

- **Grafo conexo:** G é conexo se para todos $u \in V_G$ e $v \in V_G$ existe caminho entre u e v

- **Componente conexa:** dado $G' \subseteq G$, G' é componente conexa de G se para todos $u \in V_{G'}$ e $v \in V_{G'}$ existe algum caminho entre u e v

Na aula passada...

Grafos:

› Representação computacional: bolinhas?

- Matriz de adjacência: relacionamento entre **vértices**
- Matriz de incidência: relacionamento entre **vértices e arestas**
- Lista de adjacência: explicita vizinhos de cada **vértice**

Fundamentos de estruturas de dados

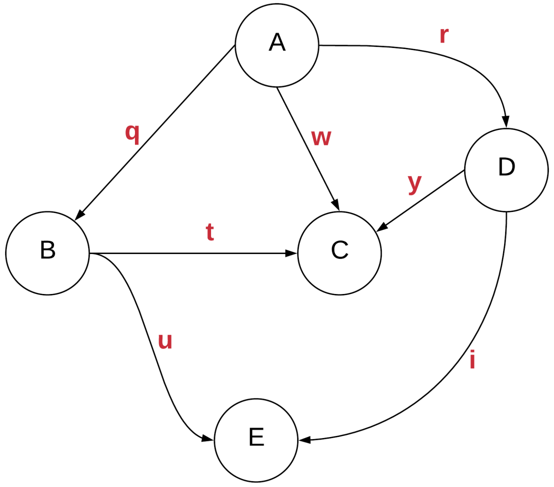
Dígrafos: implementação em C*

› **Precisamos definir:**

- Vértice

- Arco (o que é / como criar)

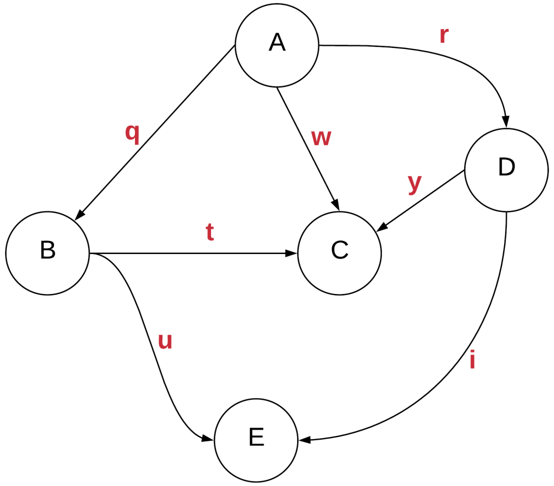
- Operações (?)



*adaptado de: http://wiki.icmc.usp.br/images/2/21/Alg2_02.Grafos_ED.pdf

Fundamentos de estruturas de dados

Digrafos: implementação em C*

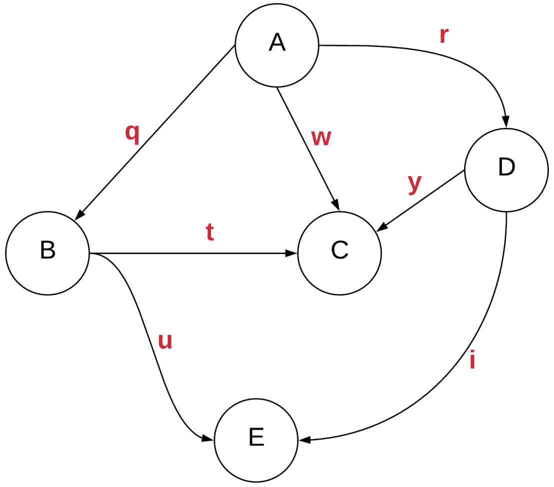



```
#define Vertice char

typedef struct {

        Vertice ponta_inicial;

        Vertice ponta_final;

} ARCO;
```

Fundamentos de estruturas de dados

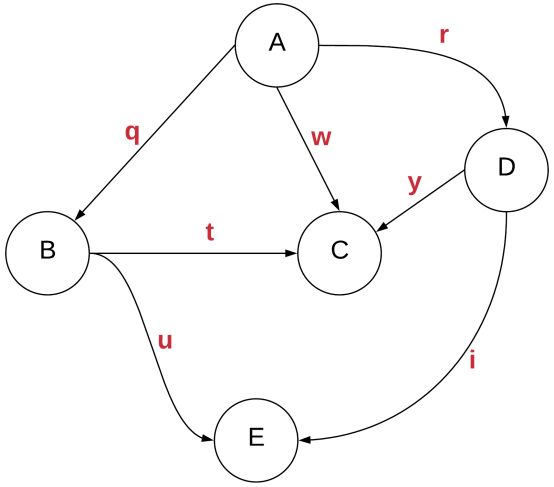
Dígrafos: implementação em C*

```
ARCO CriaArco (Vertice a, Vertice b) {
        ARCO e;
        e.ponta_inicial = a;
        e.ponta_final = b;
        return e;
}
```



Fundamentos de estruturas de dados

Dígrafos: implementação em C*



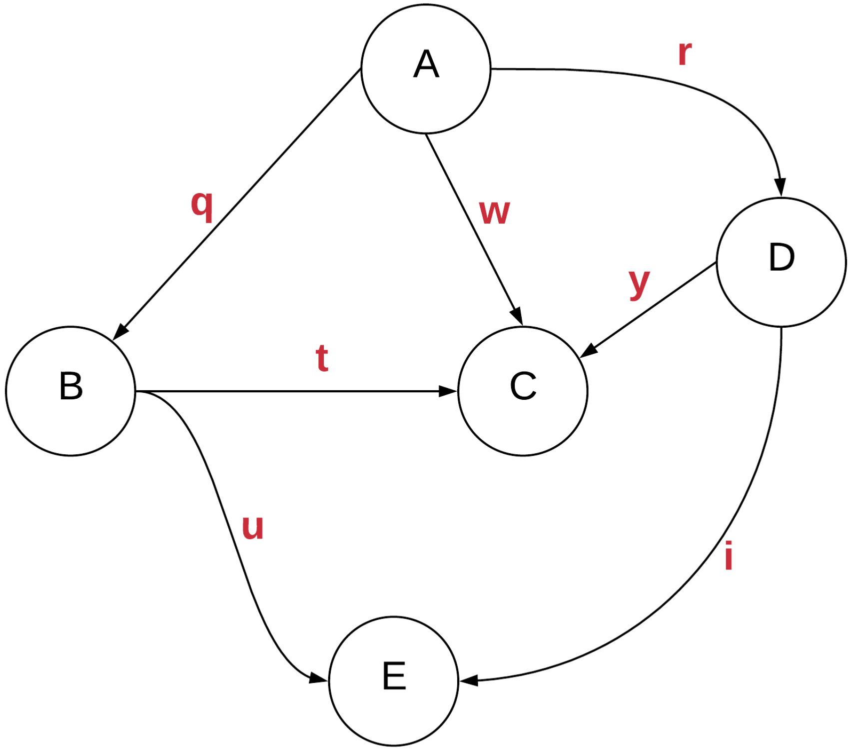
› Operações:

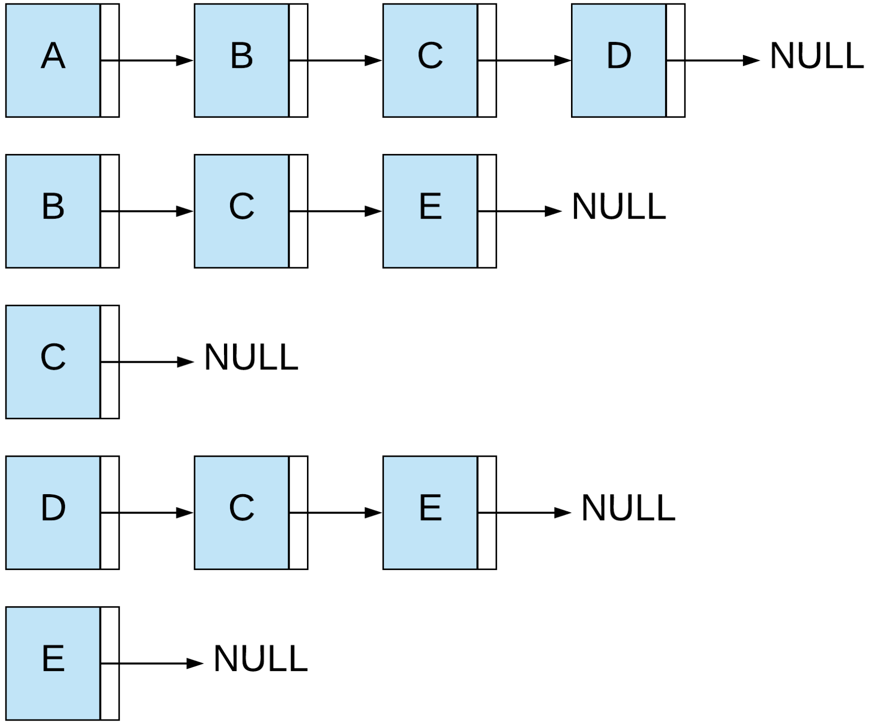
- IniciaGrafo(G)
- DestroiGrafo(G)
- InsereArco(ARCO)
- BuscaArco(ARCO)
- RemoveArco(ARCO)
- CopiaGrafo(G)

Fundamentos de estruturas de dados

Grafos: Representação computacional -> Implementação (?)

- Representação computacional: **lista de adjacência**





Fundamentos de estruturas de dados

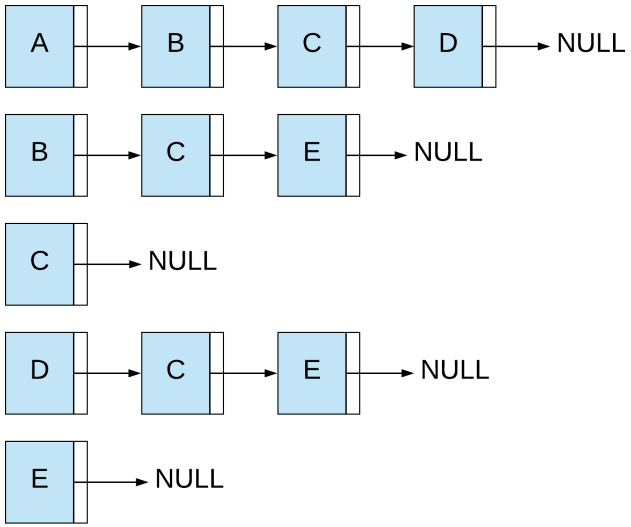
Dígrafos: implementação em C*

› **Lista de Adjacência:**

- Vértice

- Link (apontando de um vértice para outro)

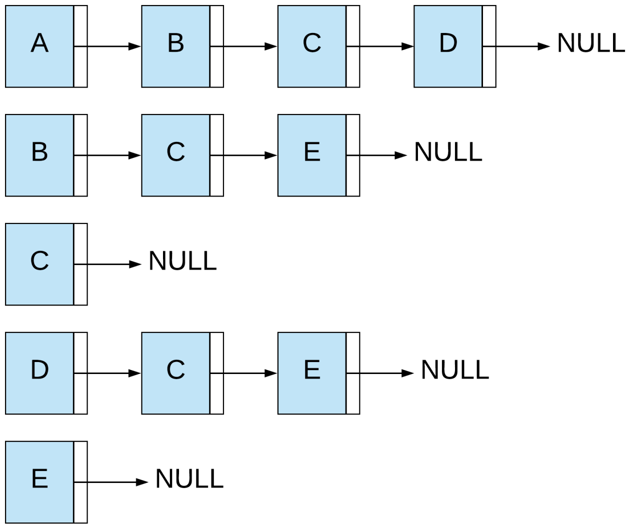
- Cada **Elemento** = Vértice + Link



*adaptado de: http://wiki.icmc.usp.br/images/2/21/Alg2_02.Grafos_ED.pdf

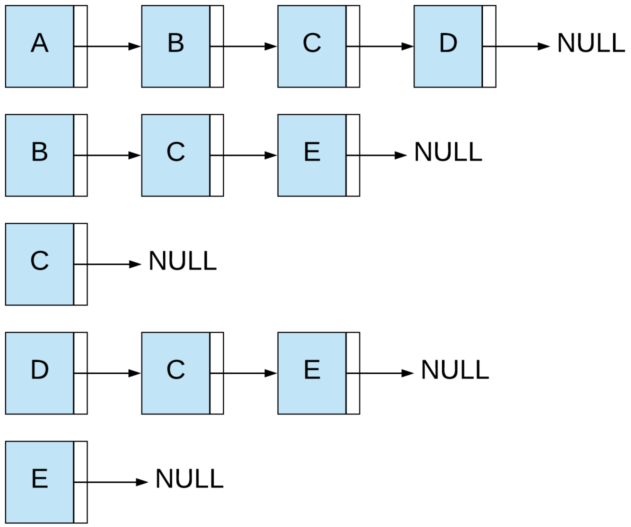
Fundamentos de estruturas de dados

Dígrafos: implementação em C*




```
typedef struct Elemento *Link;

struct Elemento {

    Vertice w;

    Link proximo;

}
```

Fundamentos de estruturas de dados



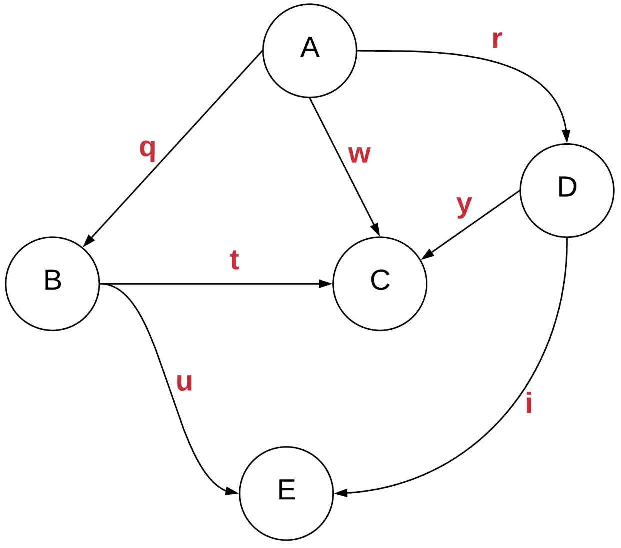
Dígrafos: implementação em C*

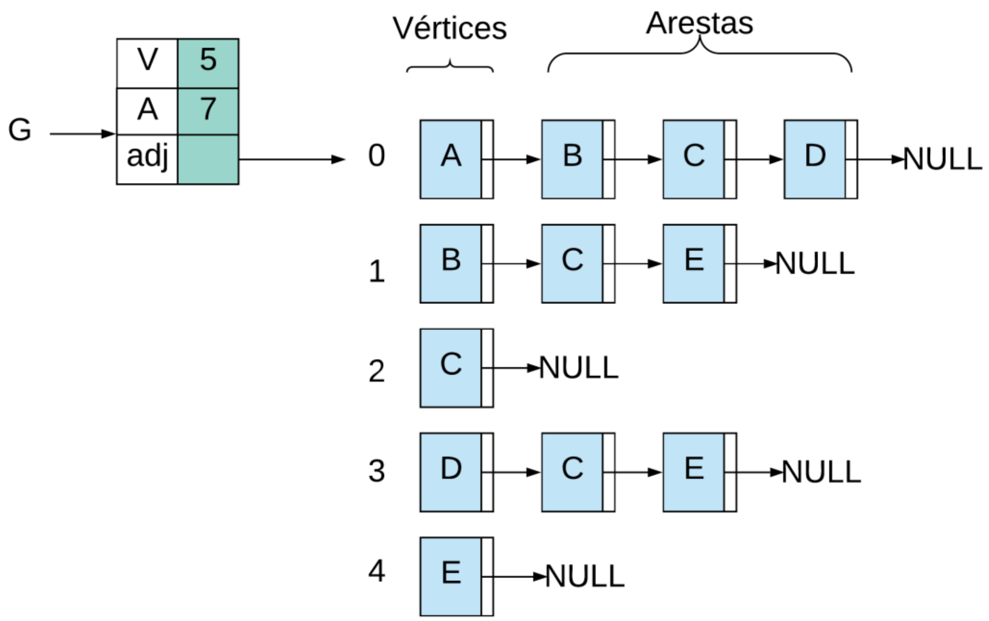
```
struct digrafo {
        int V; //numero de vertices
        int A; //numero de arestas
        Link *adj; //lista de adjacencia
}
typedef struct digrafo *Digrafo;
```

Fundamentos de estruturas de dados

Grafos:

- Criando o grafo a partir da **lista de adjacência**

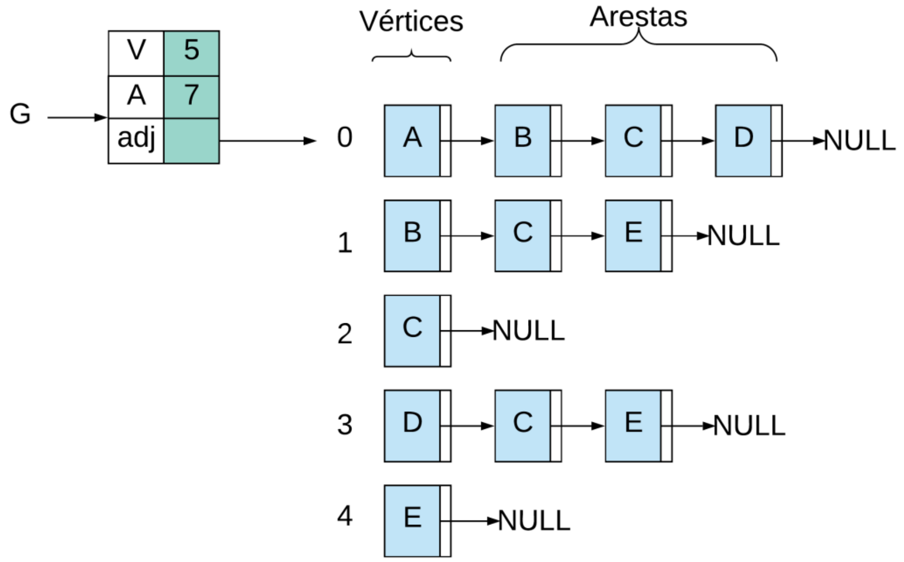




Fundamentos de estruturas de dados

› Criando o grafo a partir da **lista de adjacência:**

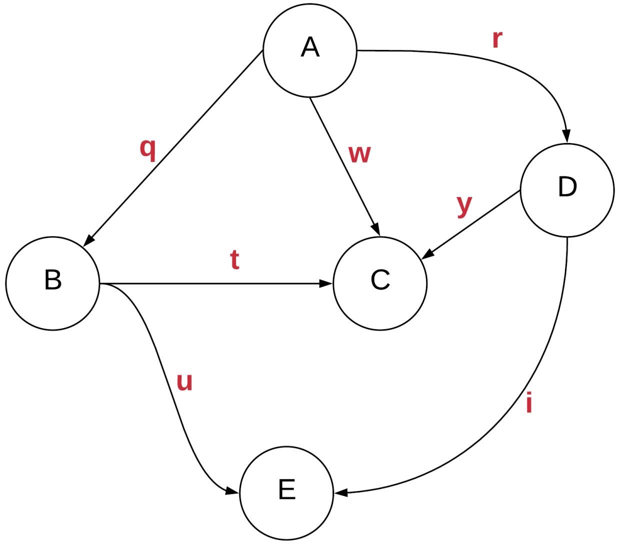
- **Para cada vértice (armazenado em ?)**
  - Enquanto tiver próximo:
    - Cria o vértice referenciado, se não existir
    - Cria um *link* do vértice atual para o vértice referenciado
- (...ou quando acabarem vértices/ arestas)

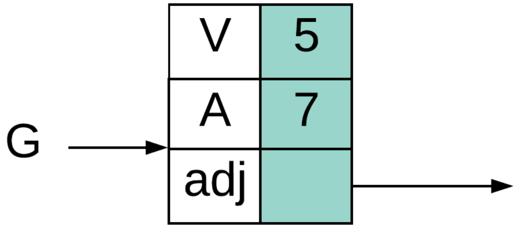


Fundamentos de estruturas de dados

Grafos:

- Criando o grafo a partir da **matriz de incidência**





$$
\begin{array}{c c}
 & \begin{array}{ccccc} A & B & C & D & E \end{array} \\
\begin{array}{c} q \\ w \\ r \\ t \\ y \\ u \\ i \end{array} &
\begin{bmatrix}
-1 & 1 & 0 & 0 & 0 \\
-1 & 0 & 1 & 0 & 0 \\
-1 & 0 & 0 & 1 & 0 \\
0 & -1 & 1 & 0 & 0 \\
0 & 0 & 1 & -1 & 0 \\
0 & -1 & 0 & 0 & 1 \\
0 & 0 & 0 & -1 & 1
\end{bmatrix}
\end{array}
$$

Fundamentos de estruturas de dados

Grafos - recapitulando:

- Definição formal
- Grafos orientados (dígrafos)
- Vértices adjacentes
- Grau de um vértice
- Arco
- Subgrafo
- Passeio / Caminho
- Grafo conexo / Componente conexa
- Representações computacionais

Fundamentos de estruturas de dados

Grafos – mais definições:

- **Passeio fechado:** É um passeio que começa e termina no mesmo vértice
- **Circuito:** É um passeio fechado que não contém **arestas** repetidas
- **Circuito simples:** É um circuito que não contém **vértices** repetidos a não ser pelo primeiro/último

Fundamentos de estruturas de dados

Grafos:

› **Ciclos:**

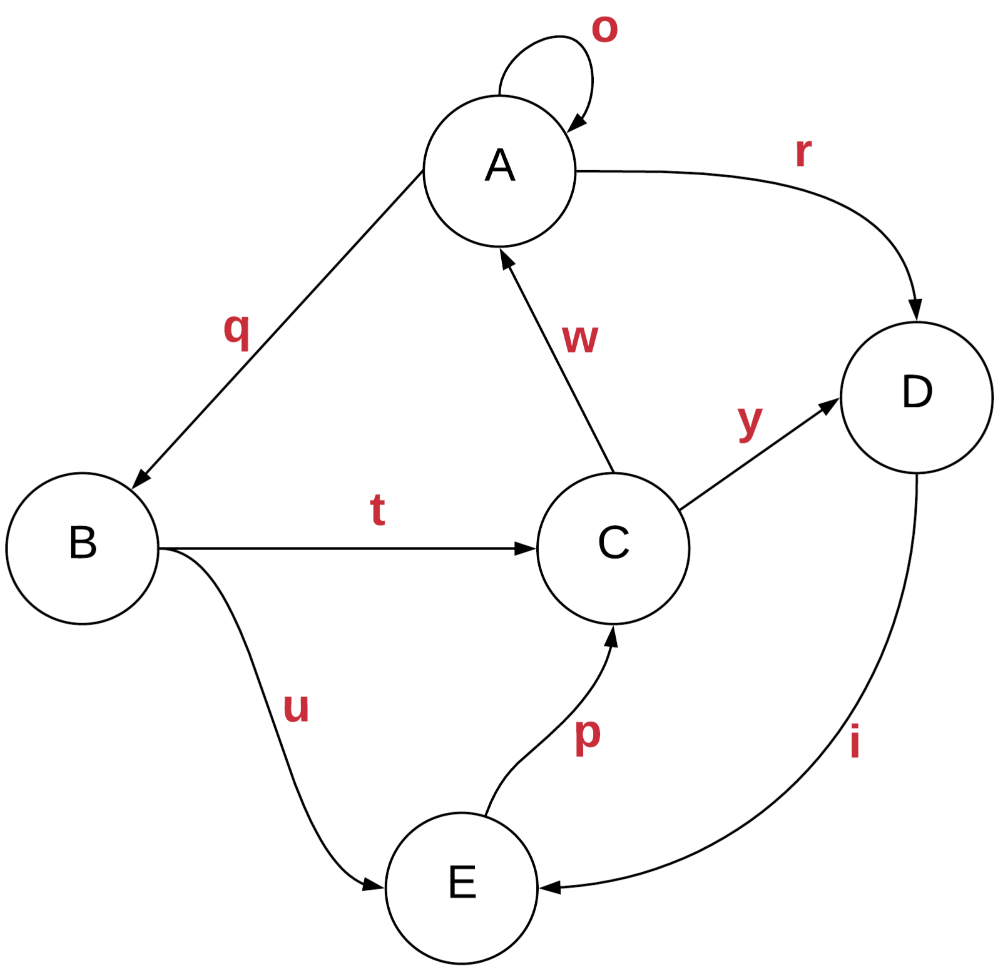
- Um ciclo é um caminho de comprimento maior ou igual a 1 entre um vértice e ele mesmo
- Um grafo que não possui ciclos é *acíclico*
- Um caminho sem ciclos é um caminho simples

Fundamentos de estruturas de dados

Grafos:

› **Ciclos:**

- <A,D,E,C,A>
- <A,B,C,A>
- <A,B,E,C,A>
- <D,E,C,D>
- <D,E,C,A,D>
- <A,A>



Fundamentos de estruturas de dados

Grafos:

› **Conexão:**

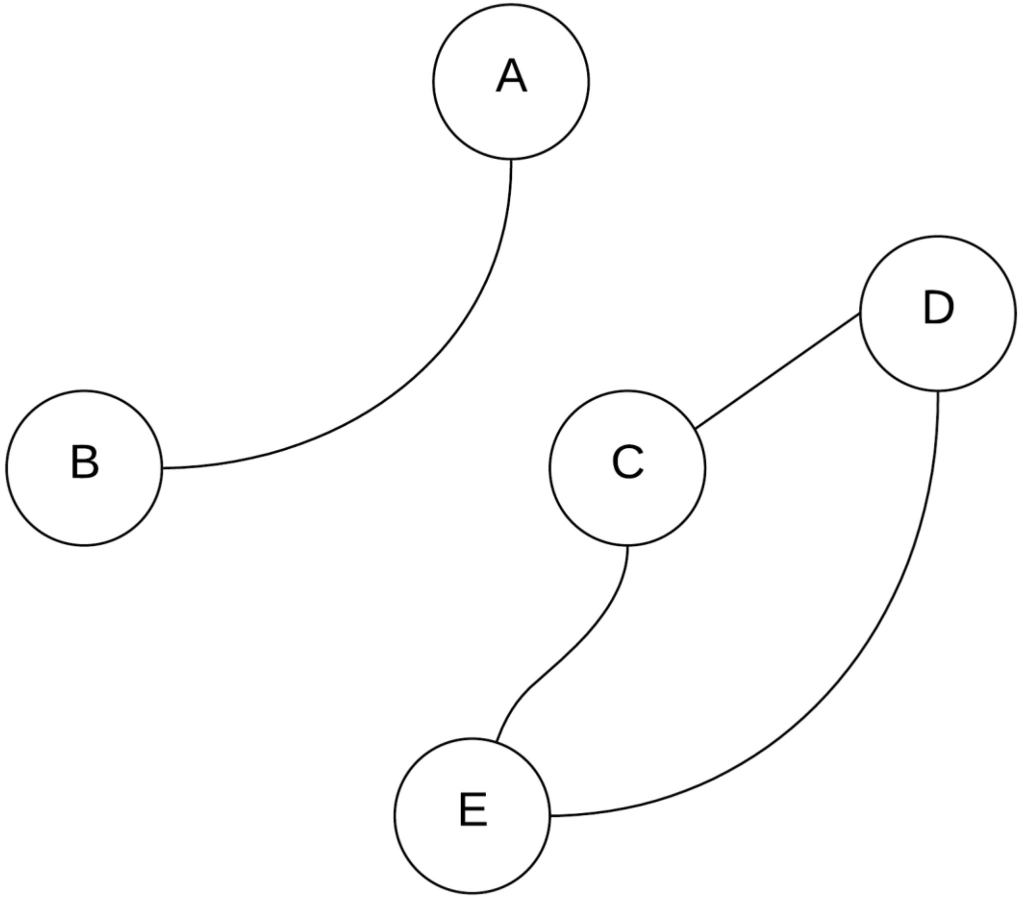
- Um **grafo não-direcionado** é *conexo* se e somente se existe um *caminho* entre quaisquer dois de seus vértices
- Um **dígrafo** é *fortemente conexo* se e somente se existe um *caminho* entre quaisquer dois de seus vértices

Fundamentos de estruturas de dados

Grafos:

› **Conexão:**

- Exemplo de grafo desconexo

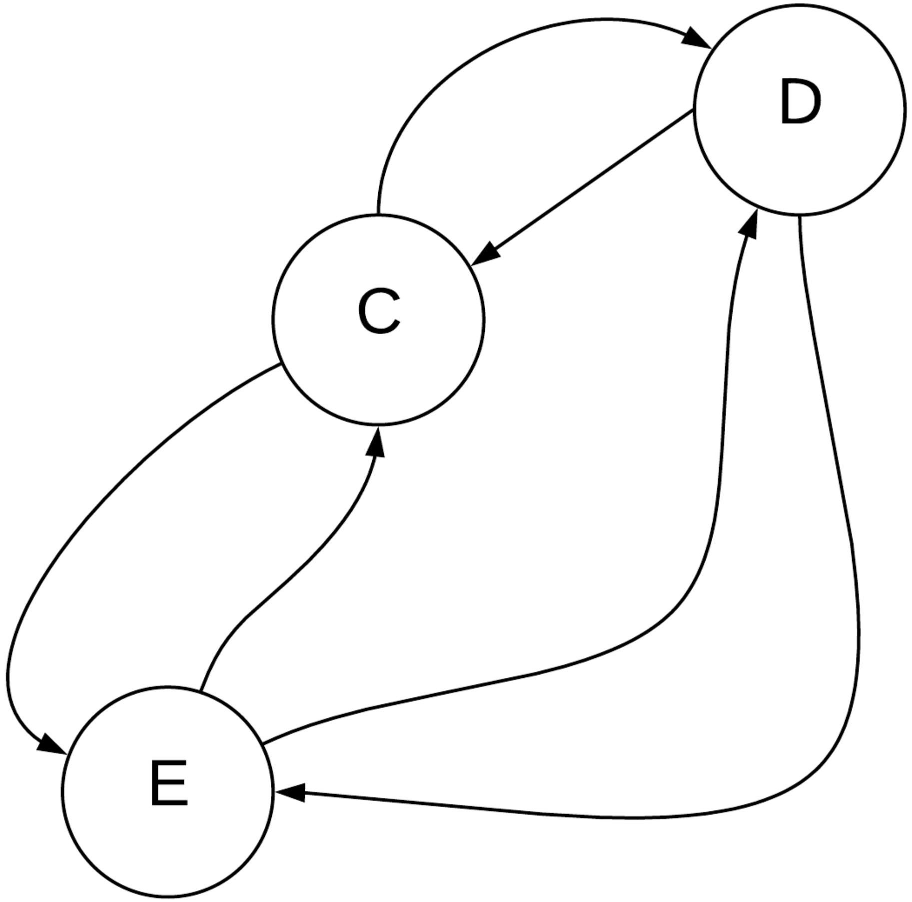


Fundamentos de estruturas de dados

Grafos:

› **Dígrafo:**

- Exemplo de dígrafo fortemente conexo

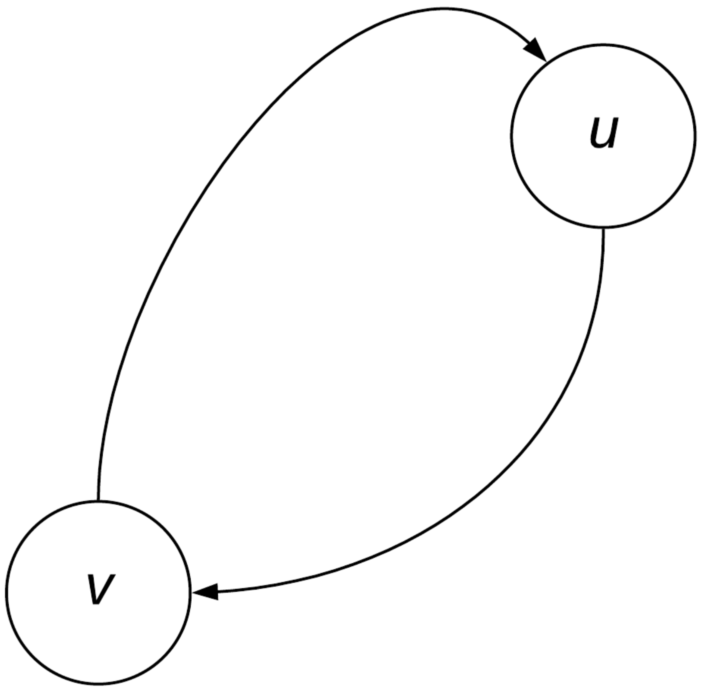


Fundamentos de estruturas de dados

Grafos:

› Para quaisquer vértices u e v do dígrafo:

- Existe um <u>passeio</u> de u a v
- Existe um <u>passeio</u> de v a u



Fundamentos de estruturas de dados

Grafos:

› **Conexão:**

- Dois vértices de um grafo são conexos se existir um passeio entre eles
- Um grafo é dito conexo quando qualquer par de seus vértices é conexo
- Se um grafo é conexo, quaisquer dois vértices podem ser conectados por um caminho simples

Fundamentos de estruturas de dados

Grafos:

› **Conexão:**

- Se dois vértices u e v fazem parte de um circuito e uma aresta entre u e v é removida, ainda existe um caminho entre u e v

Fundamentos de estruturas de dados

Grafos:

› **Conexão:**

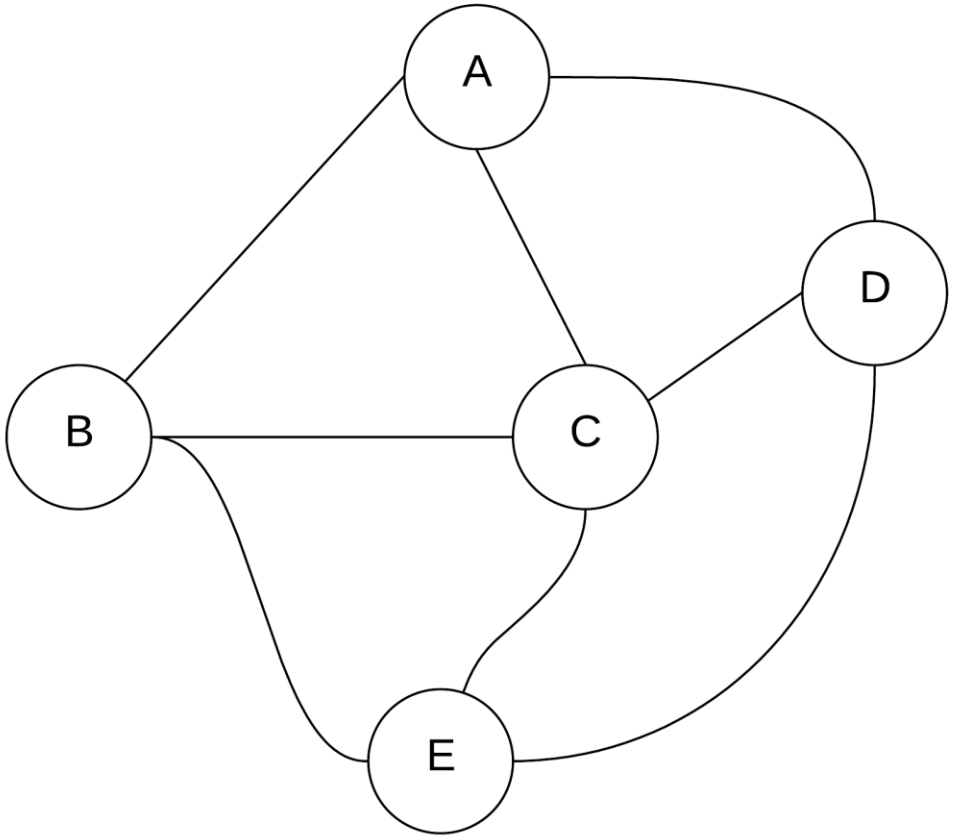
- Um grafo H é um **componente conexo** do grafo G quando:
  - H é subgrafo de G
  - H é conexo
  - H não é subgrafo de nenhum outro grafo conexo

Fundamentos de estruturas de dados

Grafos:

› **Componente conexos: (?)**

- <A,D,E,C,A>
- <A,B,C,A>
- <A,B,E,C,A>
- <D,E,C,D>
- <A,A>

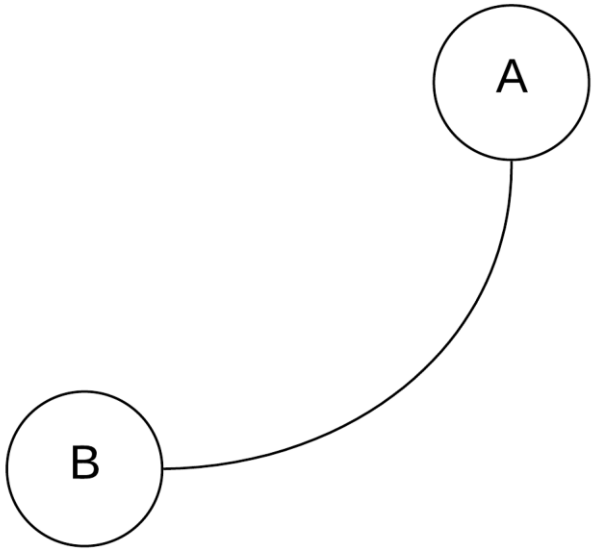


Fundamentos de estruturas de dados

Grafos:

› Conexão:

- Qualquer grafo é formado pela união de seus componentes conexos



Fundamentos de estruturas de dados

Grafos:

- Todas as definições vistas usam, direta ou indiretamente, o conceito de *passeio*
- *Como passear por um grafo?*

Fundamentos de estruturas de dados

Grafos: Métodos de Passeio

- Formas sistemáticas para explorar o grafo, com o objetivo de se obter informações sobre sua estrutura
- Algumas dificuldades:
  - Ausência de nó referencial
  - Visitas repetidas a um mesmo nó
  - *Eficiência* do método de passeio

Fundamentos de estruturas de dados

Grafos: Métodos de Passeio

- **Largura:** Todos os vértices localizados a uma distância d de um vértice u, escolhidos aleatoriamente, são percorridos antes dos vértices localizados a uma distância $d + 1$ de u

- **Profundidade:** Para um vértice u aleatório, visita-se um de seus vértices adjacentes. Para este vértice, visita-se um outro vértice adjacente, até que o vértice mais distante de u tenha sido visitado. Os vértices adjacentes a este vértice são visitados da mesma forma, e assim por diante, regressivamente. O processo é repetido até que todos os vértices do grafo tenham sido visitados.

Fundamentos de estruturas de dados

Grafos: Métodos de Passeio

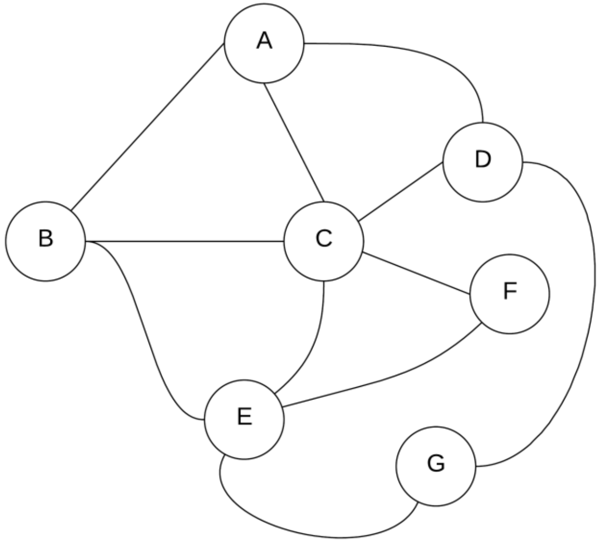
› **Distância:**

- A medida de distância *d* entre dois vértices pode ser definida por: número de arestas entre dois vértices, soma de pesos associados a cada aresta entre os vértices, etc.

Fundamentos de estruturas de dados

Grafos: Métodos de Passeio

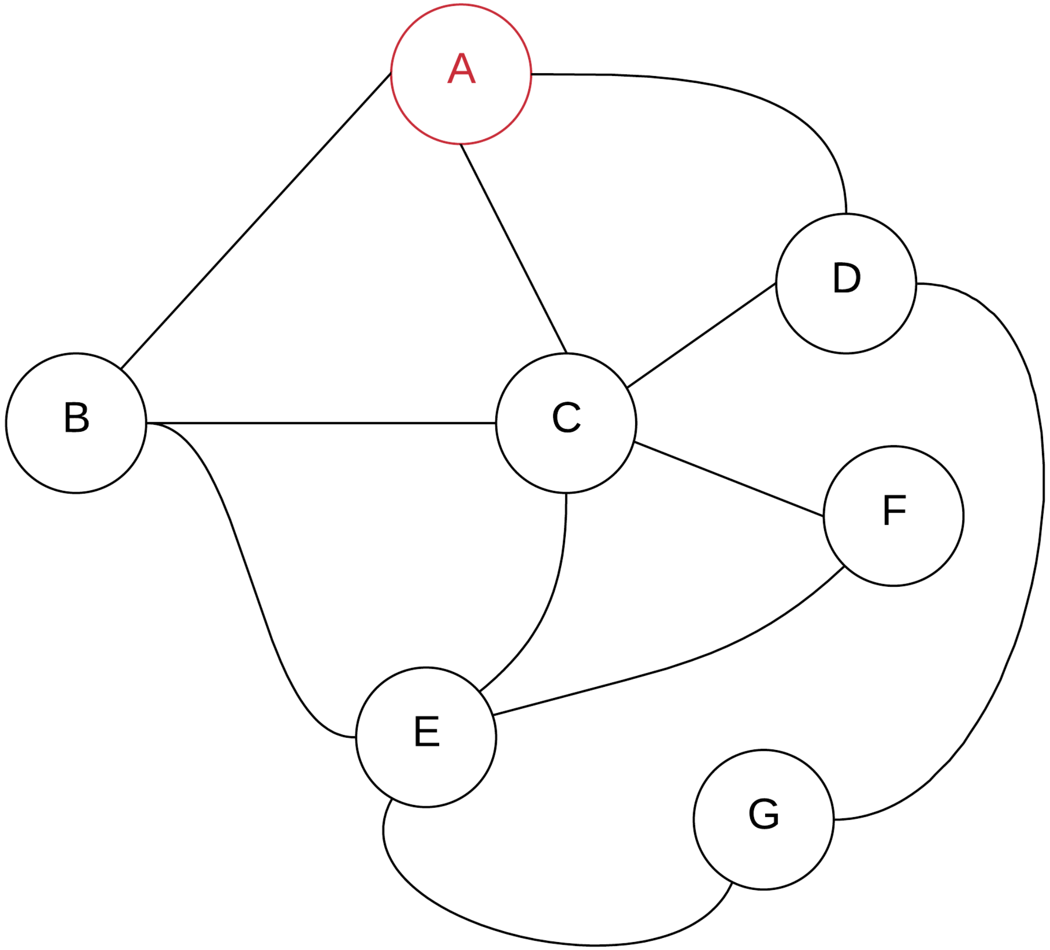
- **Largura:** Todos os vértices localizados a uma distância *d* de um vértice *u*, escolhidos aleatoriamente, são percorridos antes dos vértices localizados a uma distância *d + 1* de *u*



Fundamentos de estruturas de dados

Grafos: Passeio em Largura

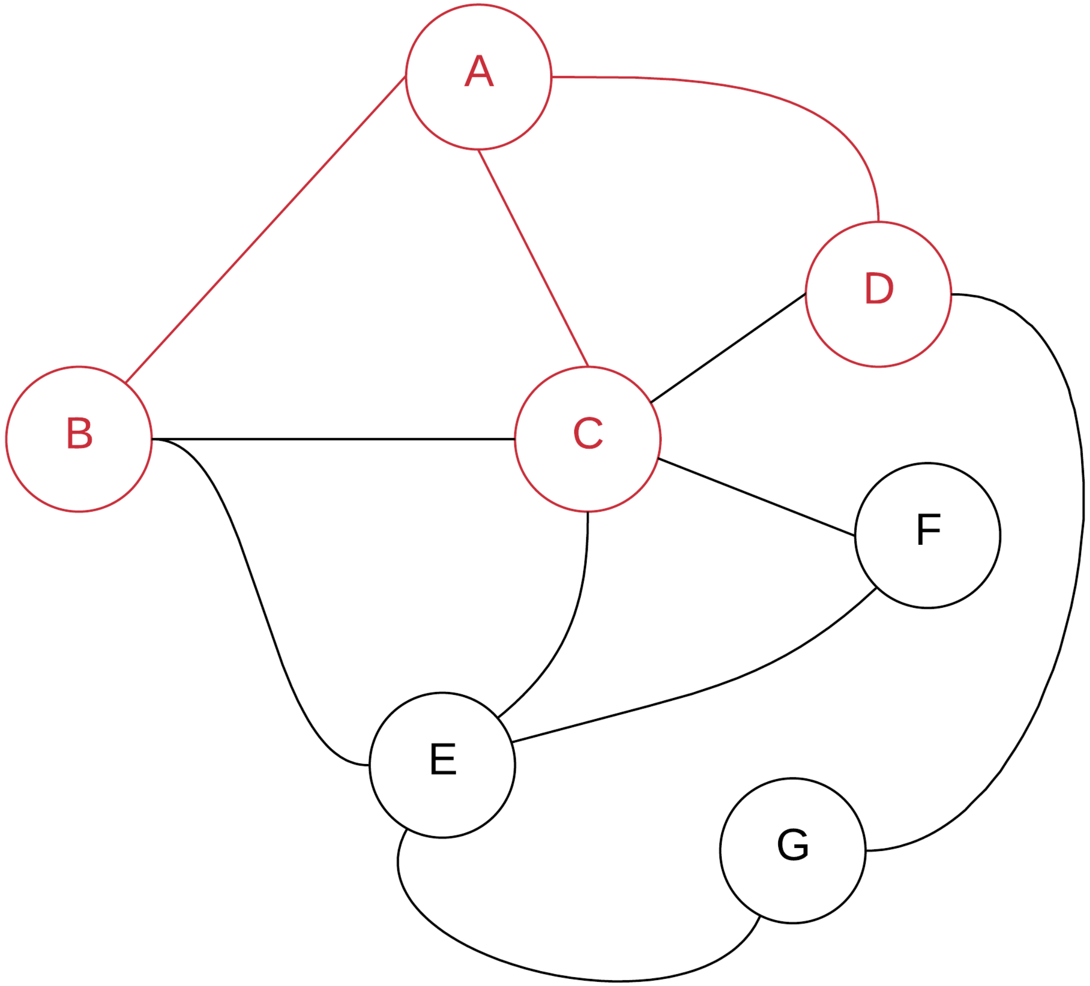
- Vértice inicial: A
- Vértices adjacentes: B, C, D ($d = 1$)



Fundamentos de estruturas de dados

Grafos: Passeio em Largura

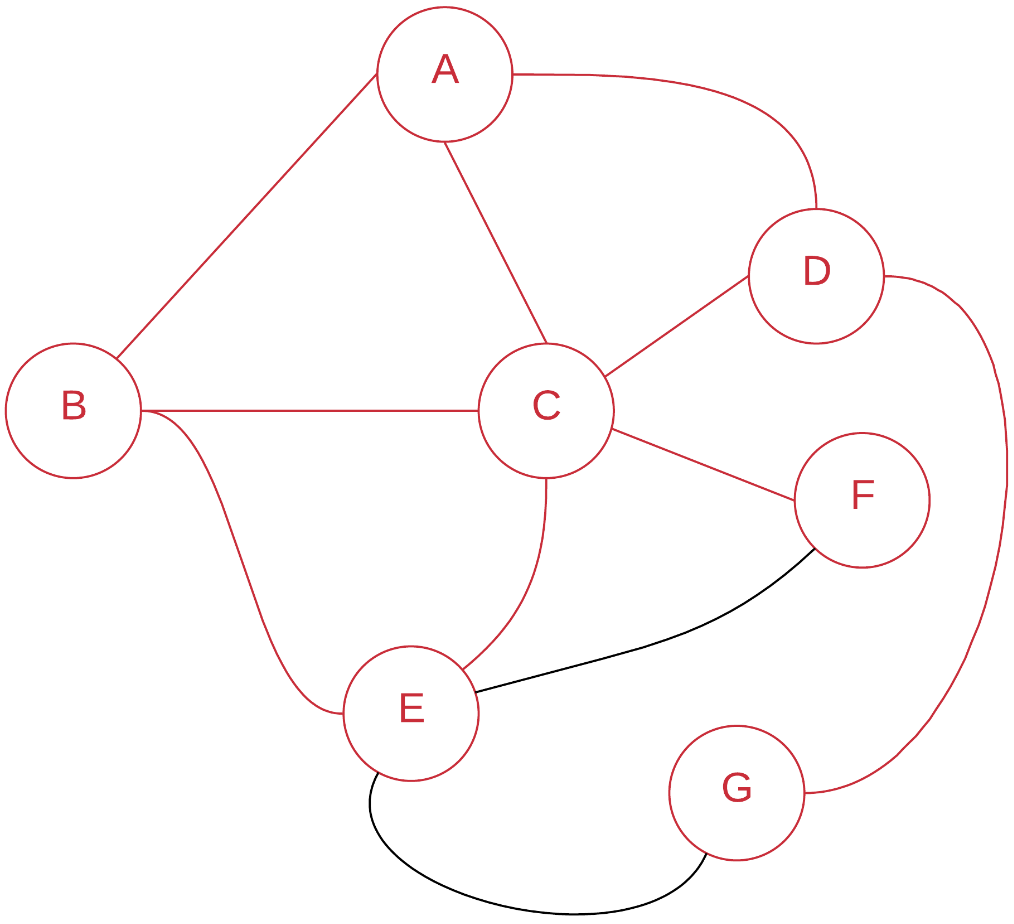
- Vértices visitados: B, C, D ($d = 1$)
- Próximos: C, E, F, G ($d = 2$)



Fundamentos de estruturas de dados

Grafos: Passeio em Largura

- Vértices visitados: C, E, F, G
  (d = 2)
- Próximos: F, G
  (d = 3)

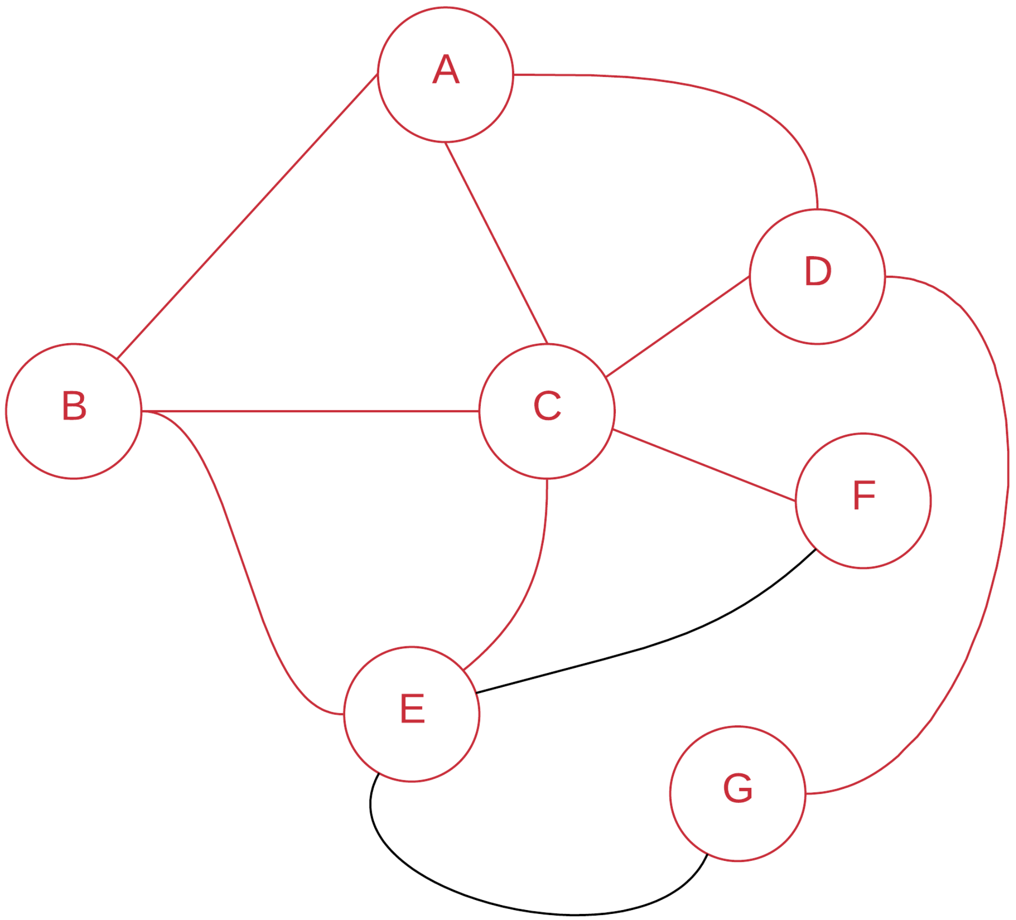


Fundamentos de estruturas de dados

Grafos: Passeio em Largura

- F e G já foram visitados
- EF e EG ainda não foram visitados

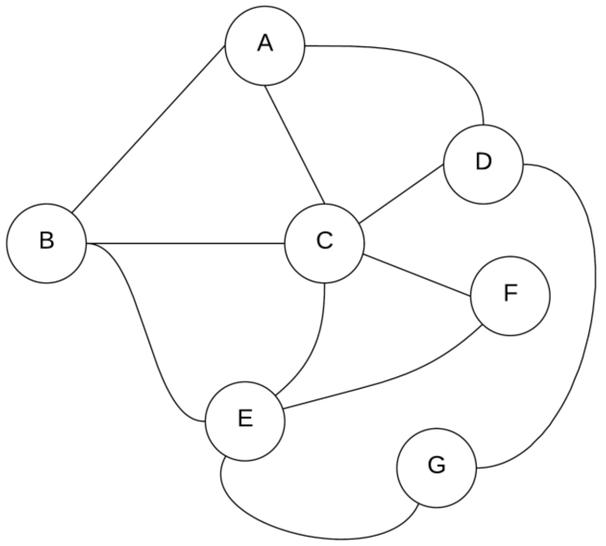
=> Para que todas as arestas sejam visitadas, os vértices tiveram que ser visitados mais de uma vez!



Fundamentos de estruturas de dados

Grafos: Métodos de Passeio

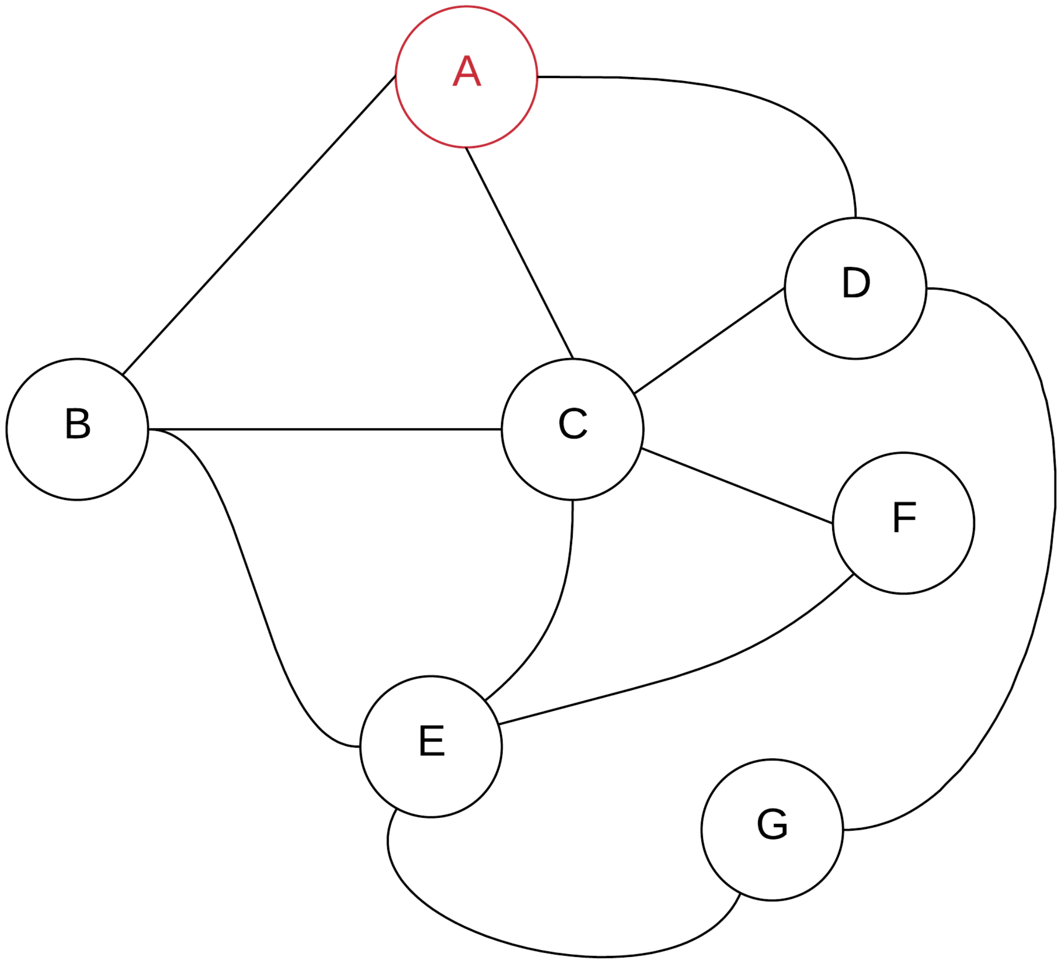
- **Profundidade:** Para um vértice *u* aleatório, visita-se um de seus vértices adjacentes. Para este vértice, visita-se um outro vértice adjacente, até que o vértice mais distante de *u* tenha sido visitado. Os vértices adjacentes a este vértice são visitados da mesma forma, e assim por diante, regressivamente. O processo é repetido até que todos os vértices do grafo tenham sido visitados.



Fundamentos de estruturas de dados

Grafos: Passeio em Profundidade

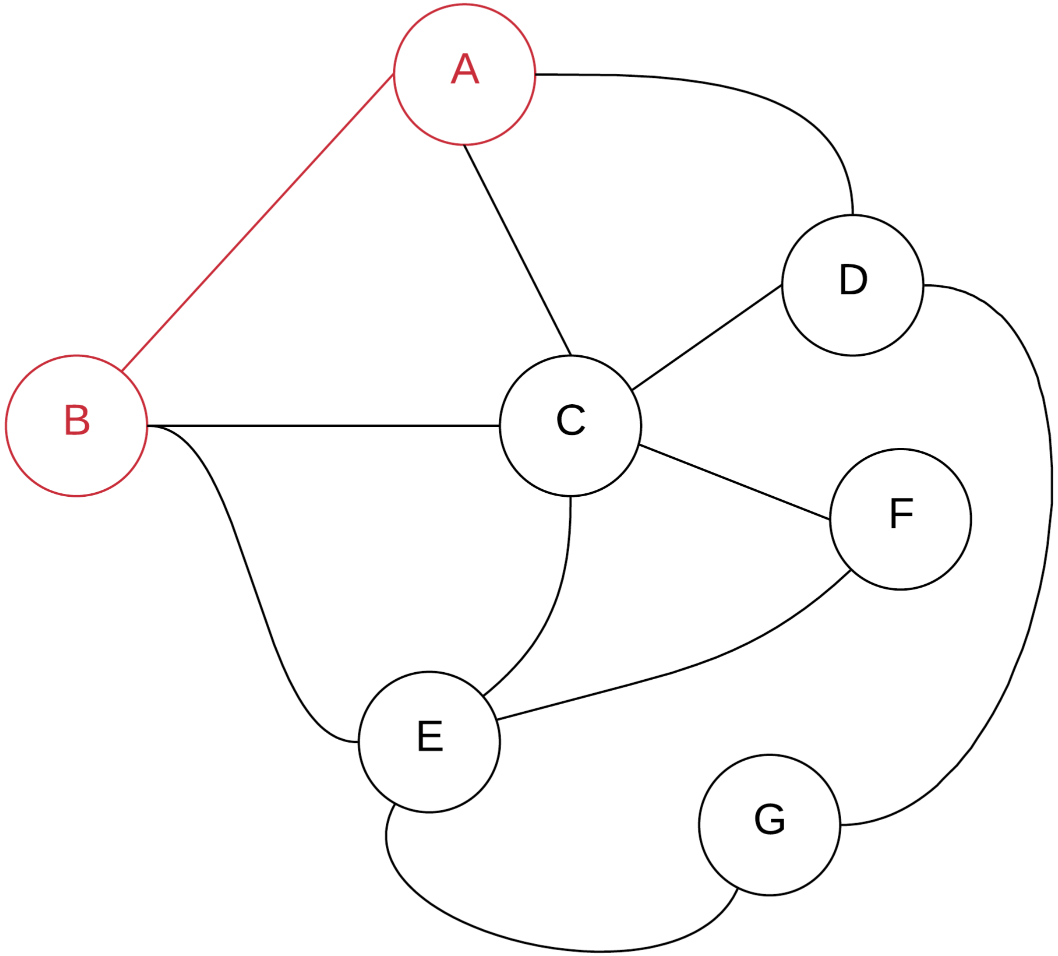
- Vértice inicial: A
- Próximo vértice: B (d = 1)



Fundamentos de estruturas de dados

Grafos: Passeio em Profundidade

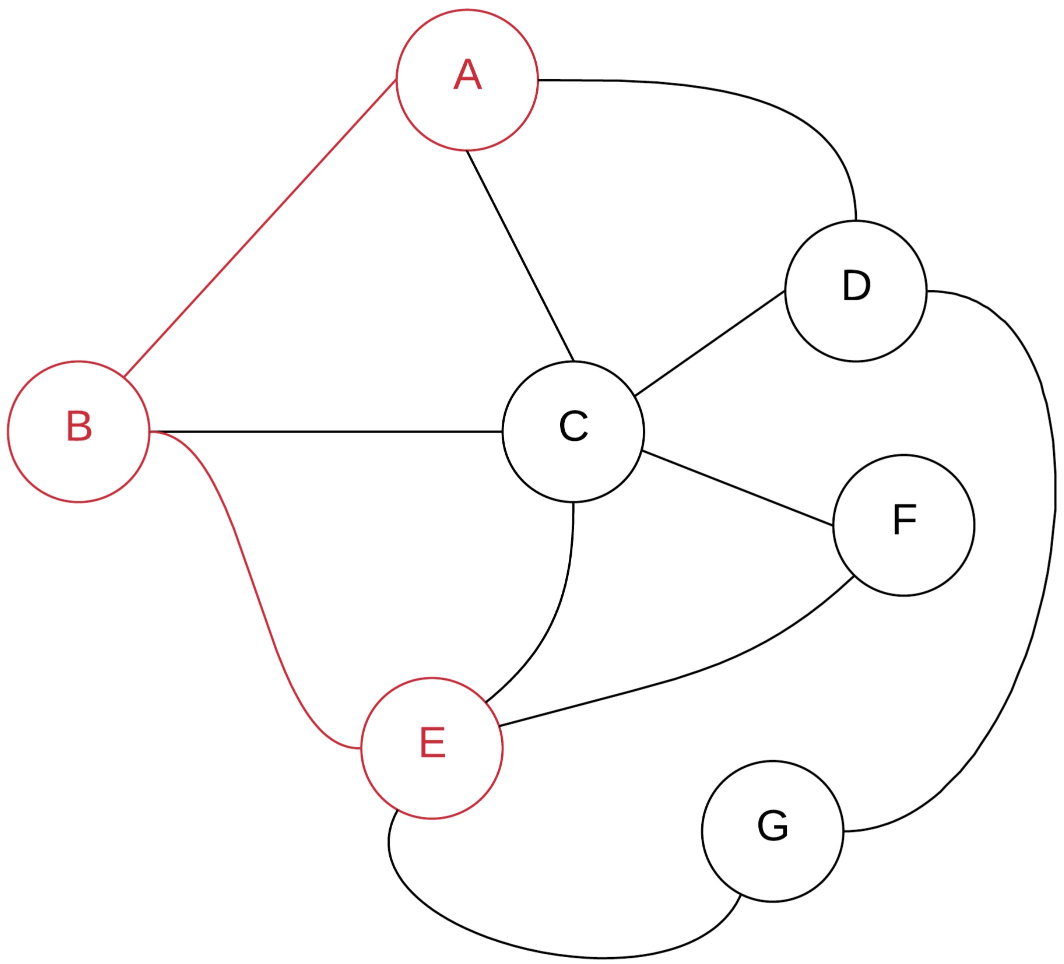
- Vértice visitado: B
- Próximo vértice: E (d = 2)



Fundamentos de estruturas de dados

Grafos: Passeio em Profundidade

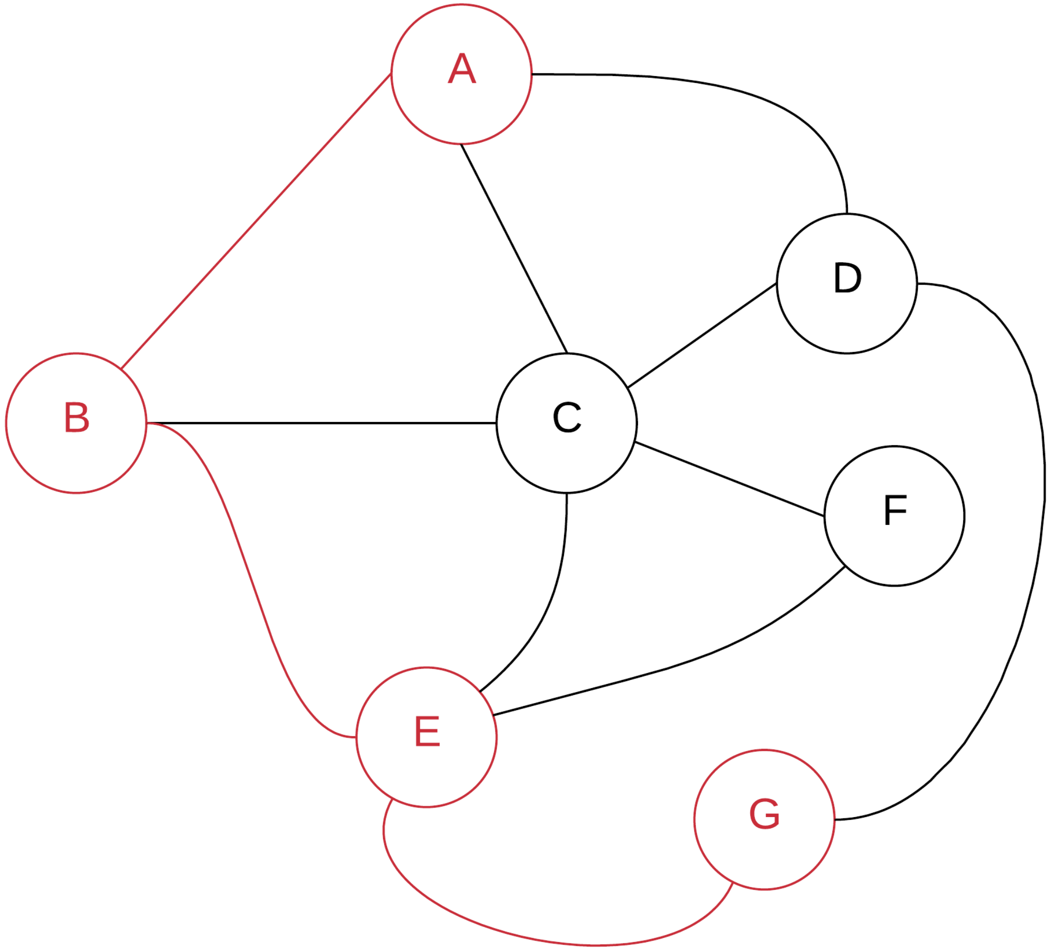
- Vértice visitado: E
- Próximo vértice: G ($d = 3$)



Fundamentos de estruturas de dados

Grafos: Passeio em Profundidade

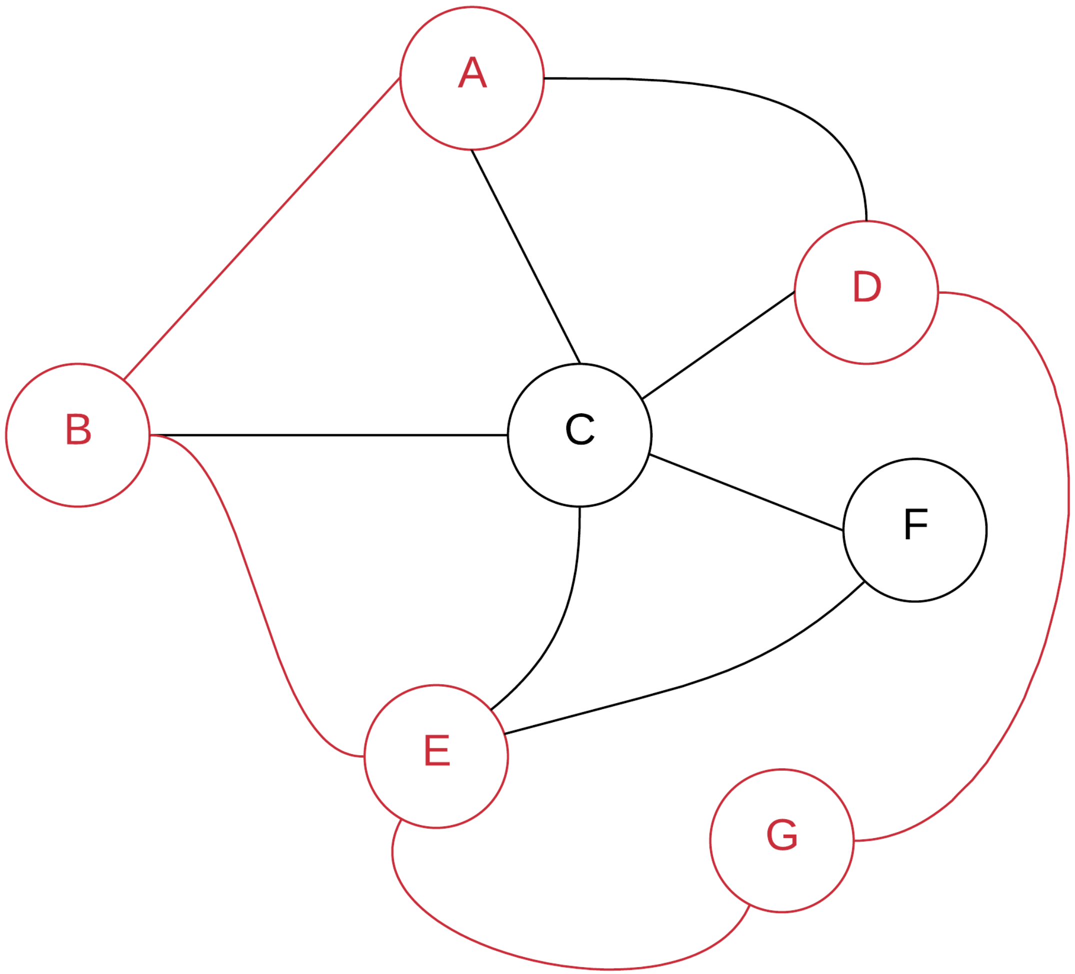
- Vértice visitado: G
- Próximo vértice: D ($d = 4$)



Fundamentos de estruturas de dados

Grafos: Passeio em Profundidade

- Vértice visitado: D
- Arestas percorridas: 4
- Vértices visitados: 5

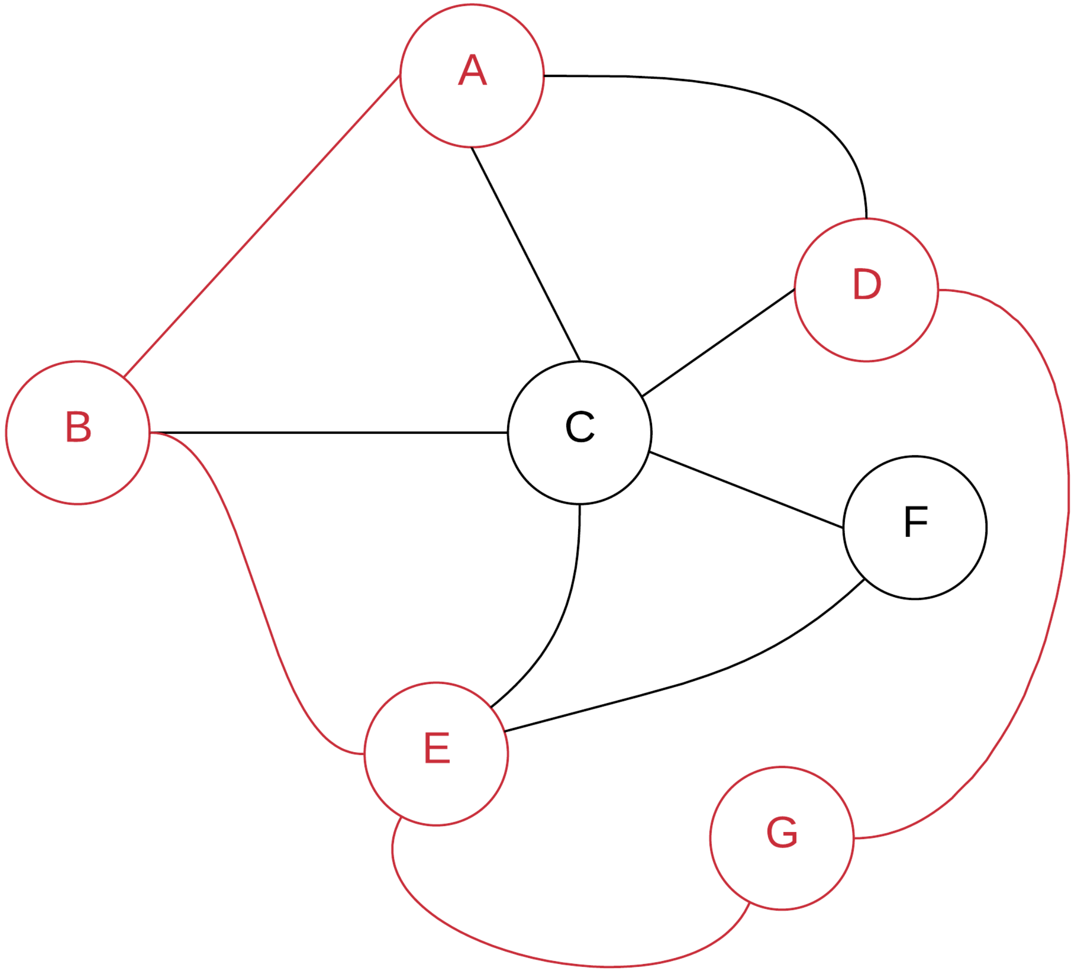


Fundamentos de estruturas de dados

Grafos: Passeio em Profundidade

- Vértice visitado: D
- Arestas percorridas: 4
- Vértices visitados: 5

=> Funciona para **achar** D?



Fundamentos de estruturas de dados

Grafos:

- **Passeio: para quê fazer?**

  - É necessário conhecer todas as arestas. Exemplo: achar a menor distância entre dois pontos.
  - É necessário conhecer todos os vértices. Exemplo: buscar um determinado elemento.

Fundamentos de estruturas de dados

Grafos:

› **Ciclo Hamiltoniano:**

- É um <u>**circuito simples**</u> que contém todos os vértices do grafo, e cada vértice é visitado somente uma vez.

- Utilização: visitar os amigos!

(No futuro: Problema do Caixeiro-viajante)

Fundamentos de estruturas de dados

Grafos:

› **Ciclo Euleriano:**

- É um <u>circuito</u> que contém todos os vértices e arestas do grafo. Por ser um circuito, cada aresta é percorrida apenas uma vez.

- Se um grafo possui um Ciclo Euleriano, então todos os seus vértices possuem um número **par** de arestas incidentes (**grau** ou **valência**).

Fundamentos de estruturas de dados

Grafos:

› **Ciclo Euleriano:**

- Se todos os vértices de um **grafo conexo** possuem um grau par, então o grafo possui um Ciclo Euleriano.
- Um Caminho Euleriano é um caminho entre dois vértices que contém todos os vértices do grafo, e que passa por todas as arestas exatamente uma vez

Fundamentos de estruturas de dados

Grafos: Problemas utilizando Ciclo Euleriano*

- **Problema do Circuito Hamiltoniano:** dada uma representação gráfica de um grafo, determinar, caso exista, um caminho que passe por todos os vértices, sem repetir vértice, começando e terminando no mesmo vértice.
- **Problema do Circuito Euleriano:** dada uma representação gráfica de um grafo, determinar, caso exista, um caminho que passe por todas as arestas, sem repetir aresta começando e terminando no mesmo vértice.

* fonte: material de referência sobre grafos, disponível no site da disciplina