Universidade Eduardo Mondlane
Faculdade de Engenharia
Departamento de Engenharia Eletrotécnica
Engenharia Informática

Estrutura de Dados e Algoritmos – EDA

Por:

- **Dr. Alfredo Covele**
- **Eng. Cristiliano Maculuve**

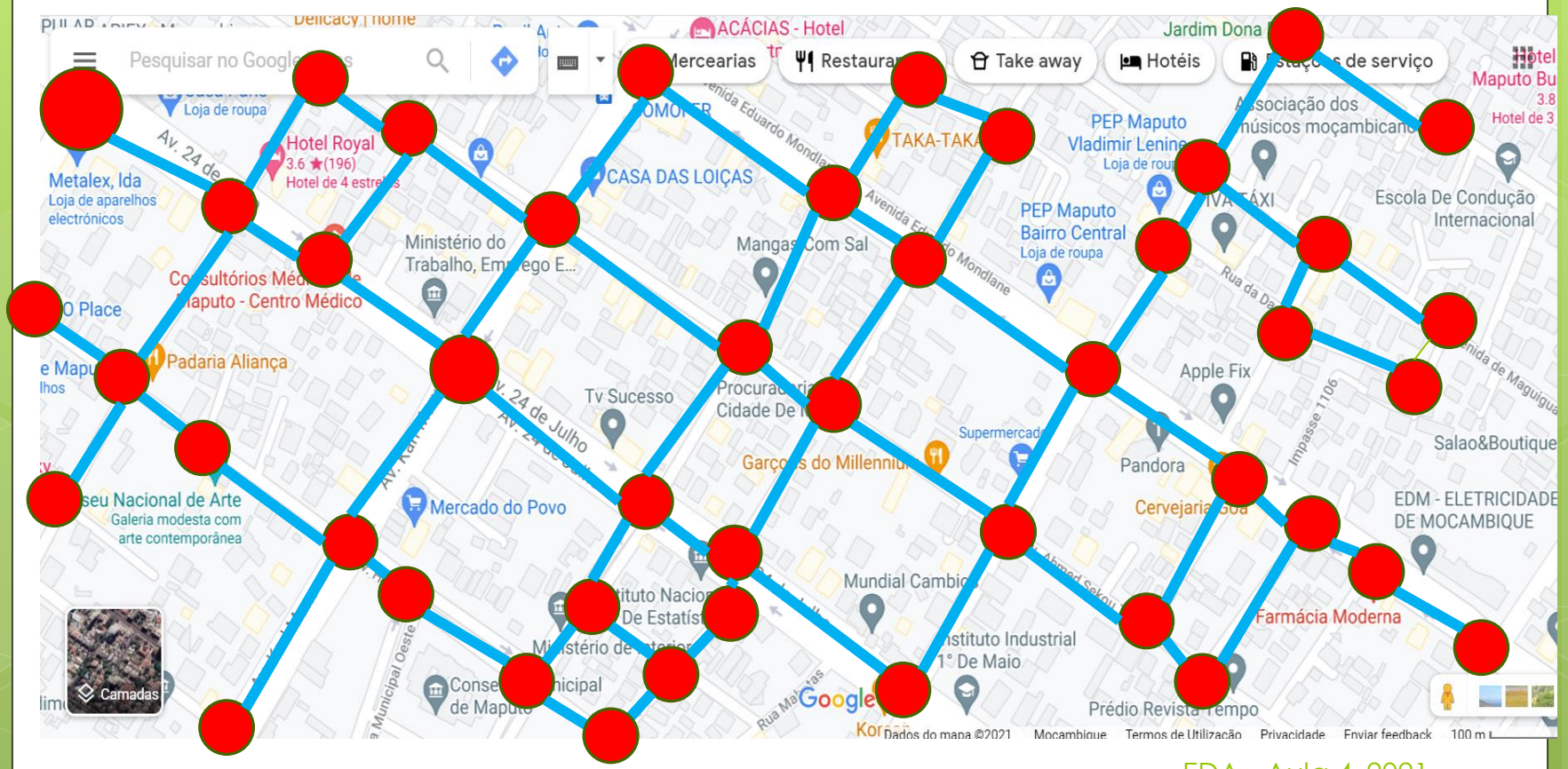
Novembro 2021

Tópicos

No.	Designação
1	Grafos ❑ Conceitos básicos ❑ Modelação ❑ Implementação em java

Grafos

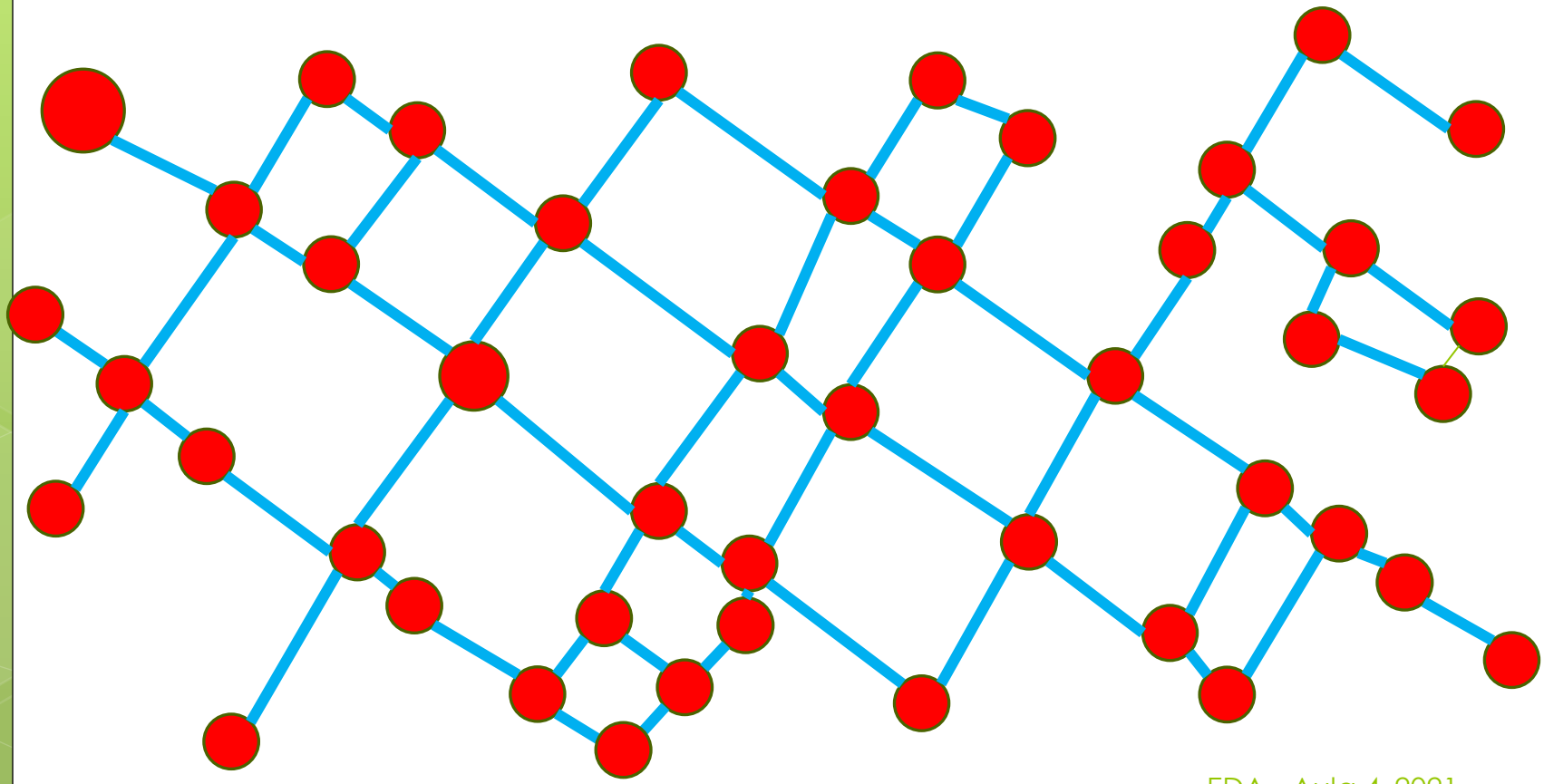
Como modelar as ruas de uma cidade?



● Vértices —— Arestas

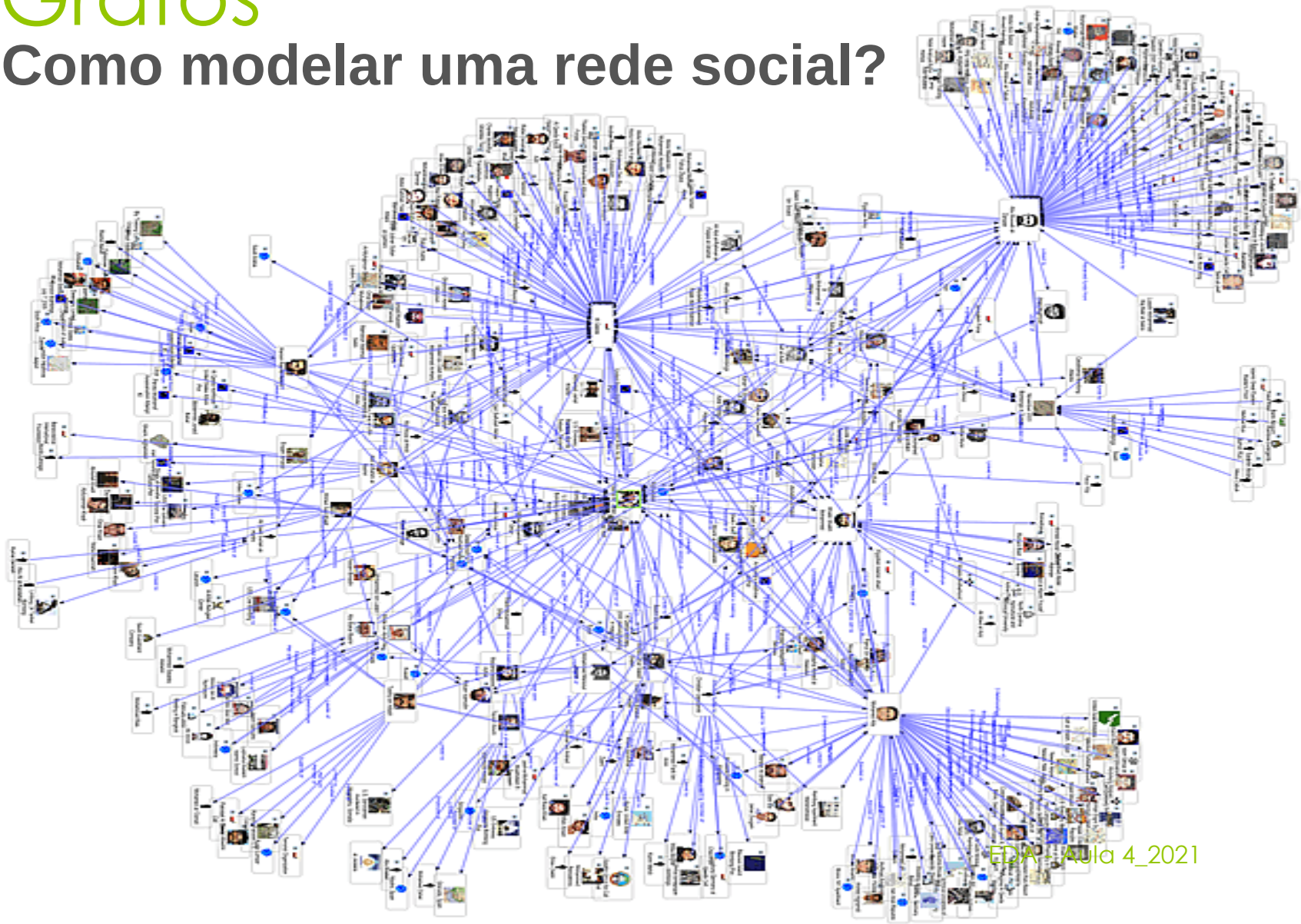
Grafos

Como modelar as ruas de uma cidade?



Grafos

Como modelar uma rede social?



Grafos

Como modelar uma rede social?

Numa rede social é possível alcançar amigos das pessoas que temos relação directa.

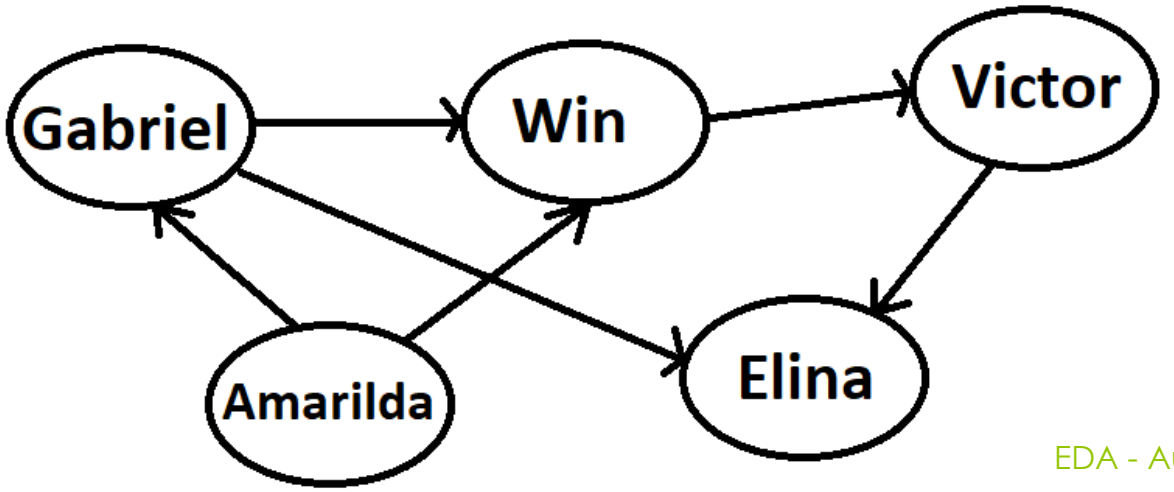
Grafos

O que são grafos?

São estruturas que permitem codificar relacionamentos entre pares de objectos.

Os objectos são os **vértices** (**nós**) do grafo

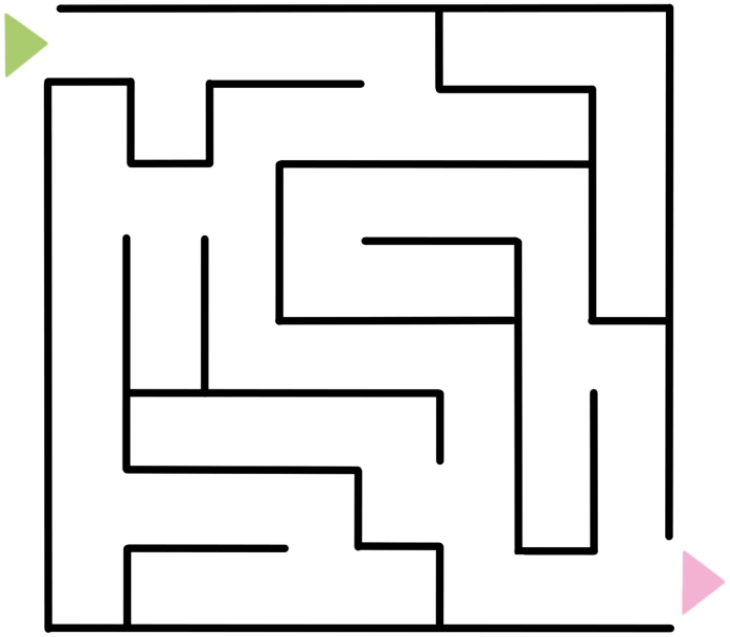
Os relacionamentos são as **arestas**

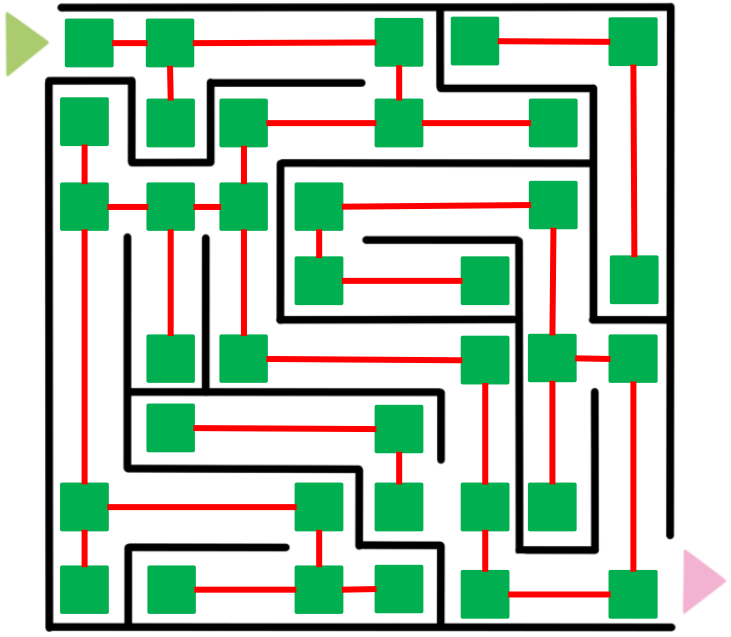


Grafos

Para que servem?

- Mapear contactos de covid19
- Modelar conexões nas redes sociais.
- Modelar Ruas de uma Cidade.
- Modelar labirinto.

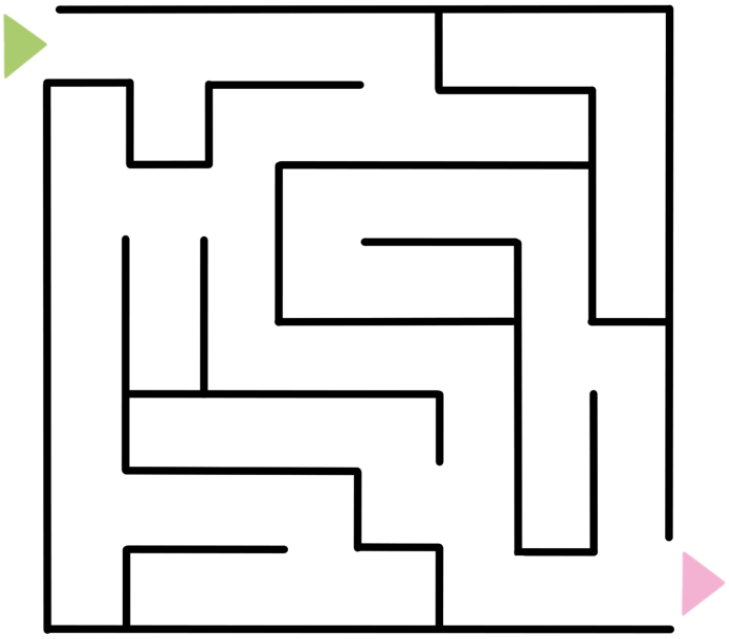


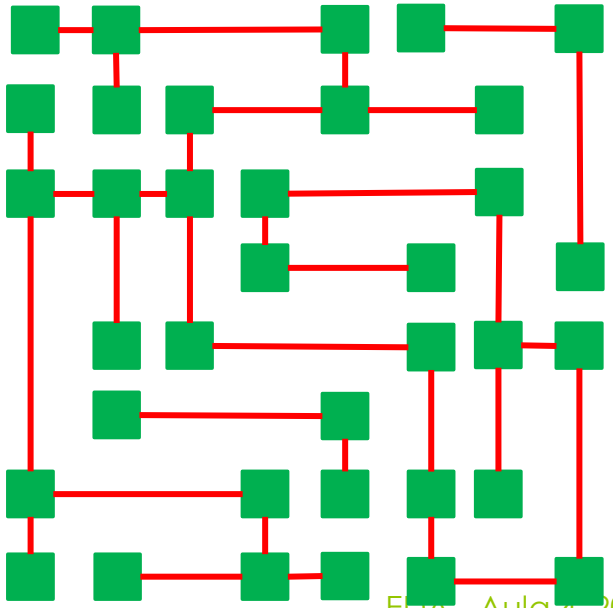


Grafos

Para que servem?

- Mapear contactos de covid19
- Modelar conexões nas redes sociais.
- Modelar Ruas de uma Cidade.
- Modelar labirinto.





Grafos

Para que servem?

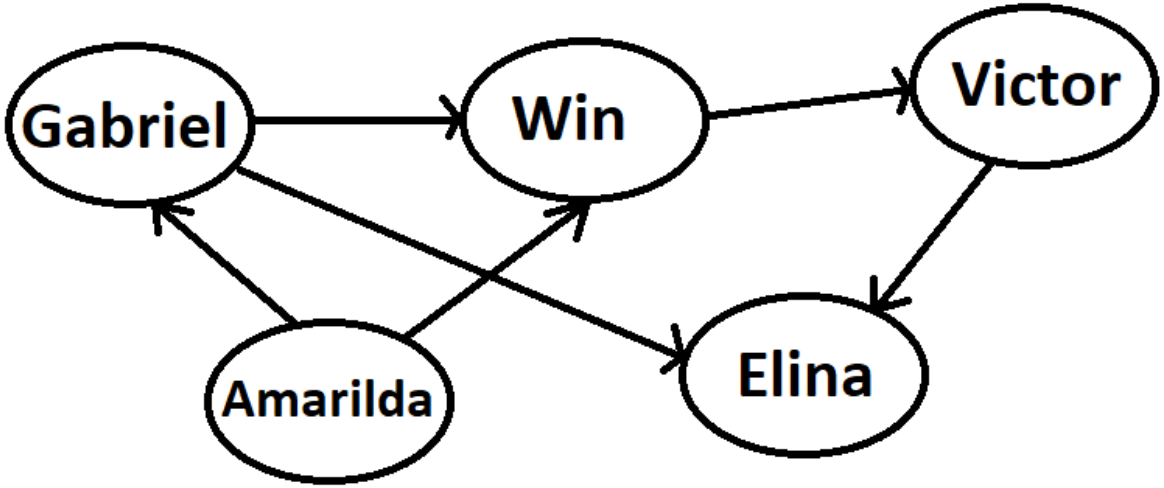
- Mapear contactos de covid19
- Modelar conexões nas redes sociais.
- Modelar Ruas de uma Cidade.
- Modelar labirinto.
- Rotas de Comboio

Grafos

Os grafos podem ser:

Dirigidos(ou **direcionados)**

São aqueles em que as relações das arestas têm sentido definido.

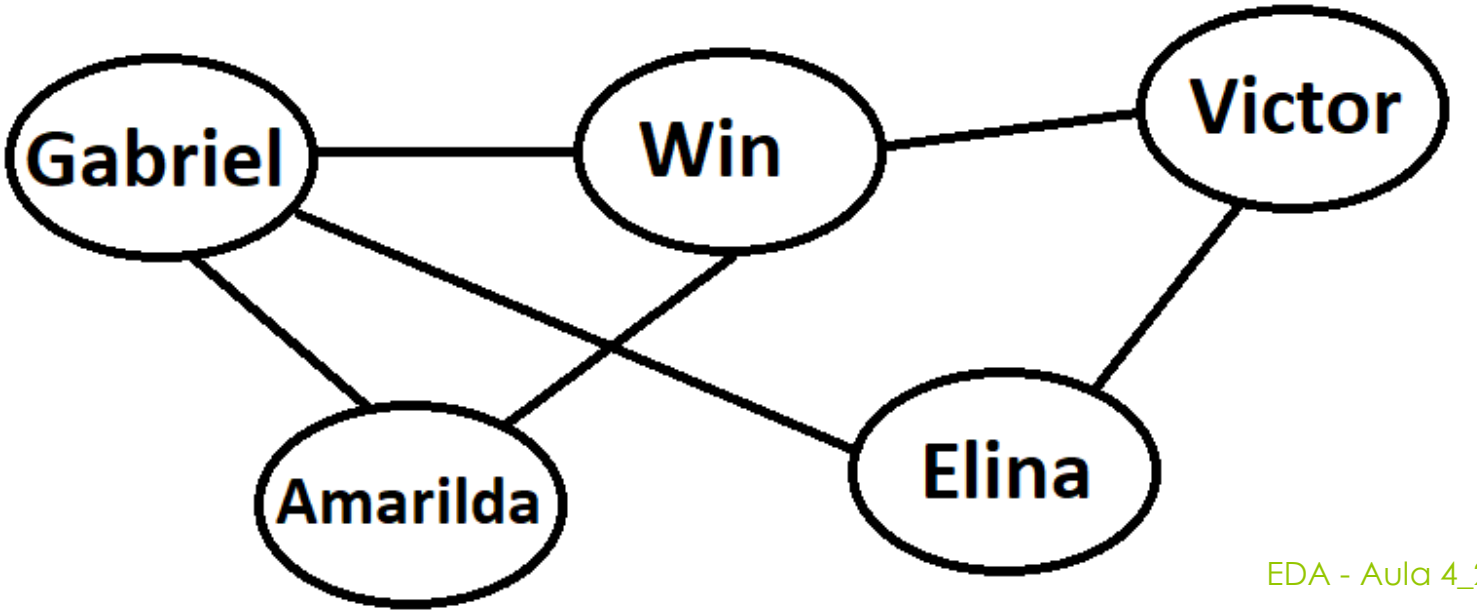


Grafos

Os grafos podem ser:

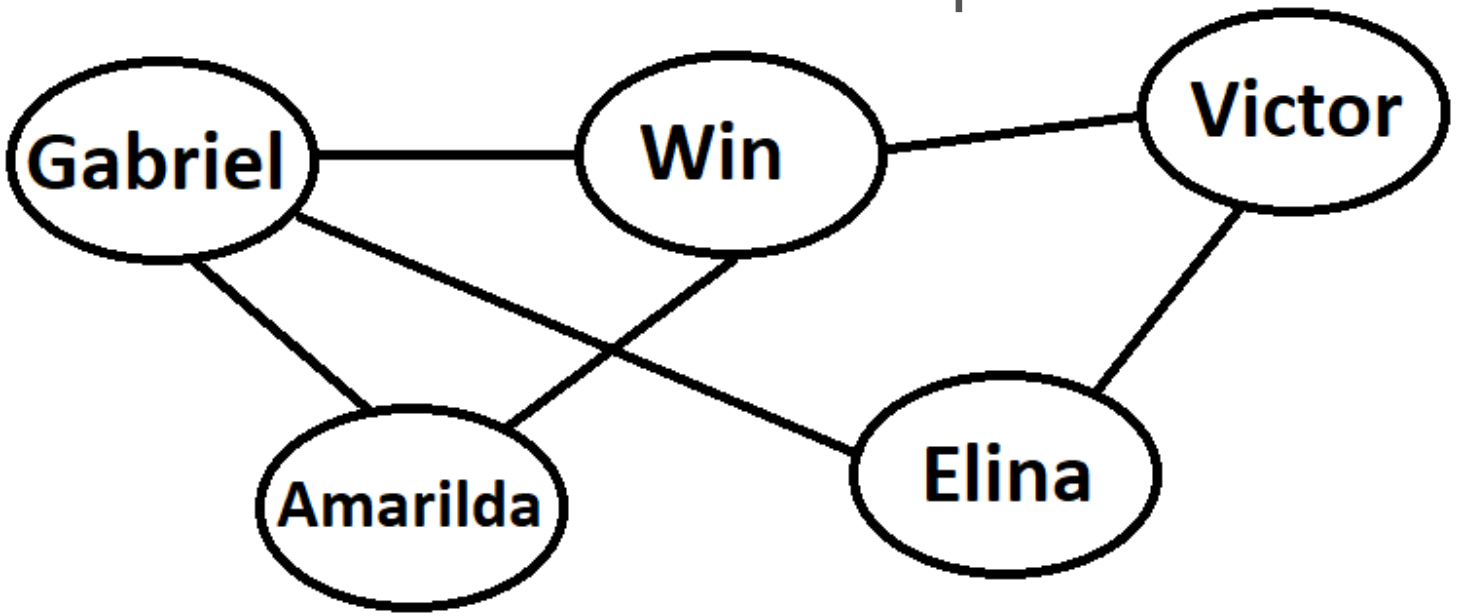
Não-dirigidos(ou **não direcionados)**

São aqueles em que as relações das arestas não têm sentido definido.



Grafos

Em grafos Não direcionados, **o grau** de um vértice é o número de arestas que incidem nele



gr(Gabriel)=3

gr(win)=3

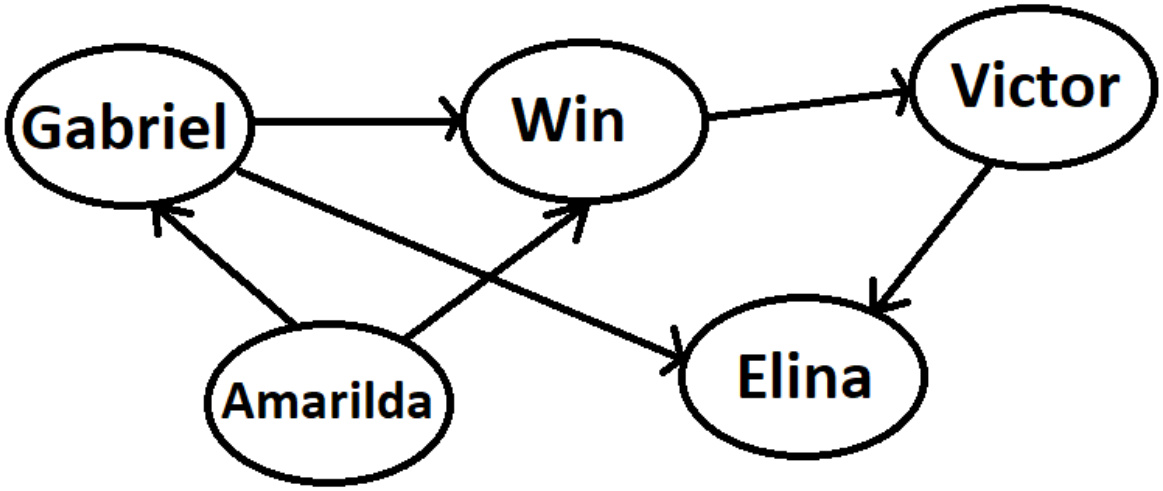
gr(Victor)=2

gr(Amarilda)=2

gr(Elina)=2

Grafos

Em grafos direcionados, **o grau** de um vértice é o número de arestas que saem do vértice mais o número de arestas que entram nele.



gr(Gabriel)=2+1 gr(Amarilda)=2+0

gr(win)=1+2 gr(Elina)=0+2

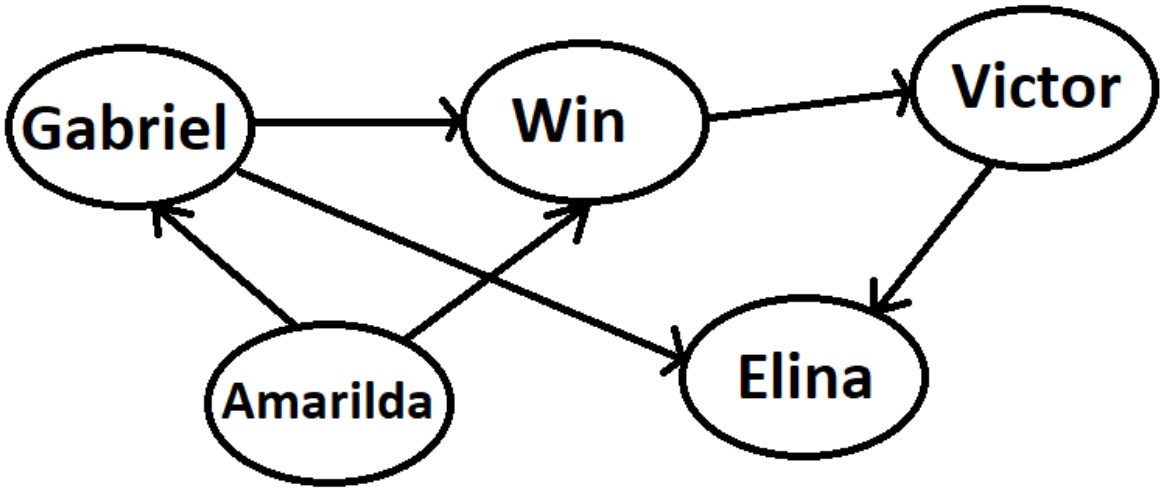
gr(Victor)=1+1

Grafos

Em grafos direcionados existe:

Grau de saída-Número de arestas que saem do vértice.

Grau de Entrada-Número de arestas que entram do vértice

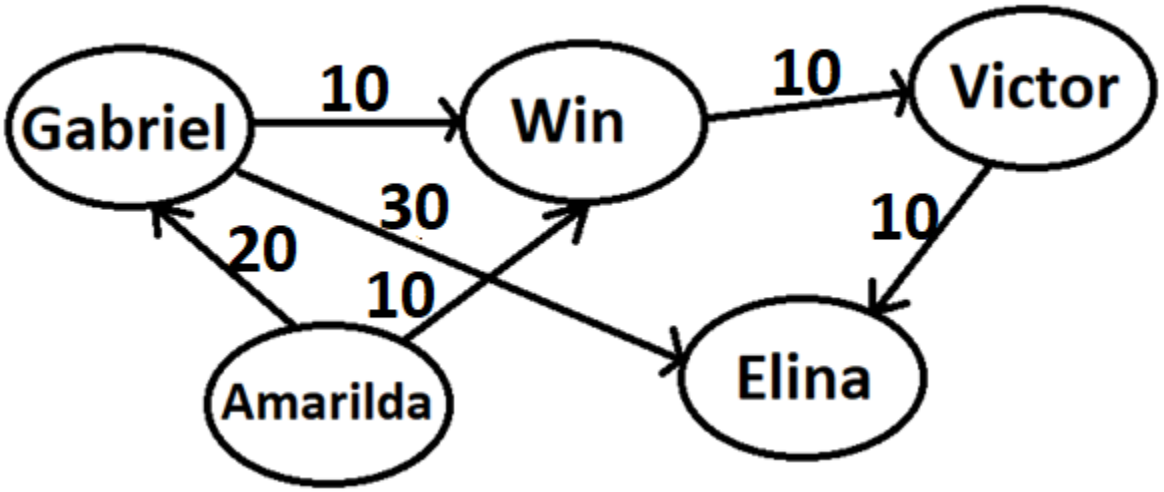


Grafos

Grafos **ponderados** são aqueles que as suas arestas possuem **pesos**.

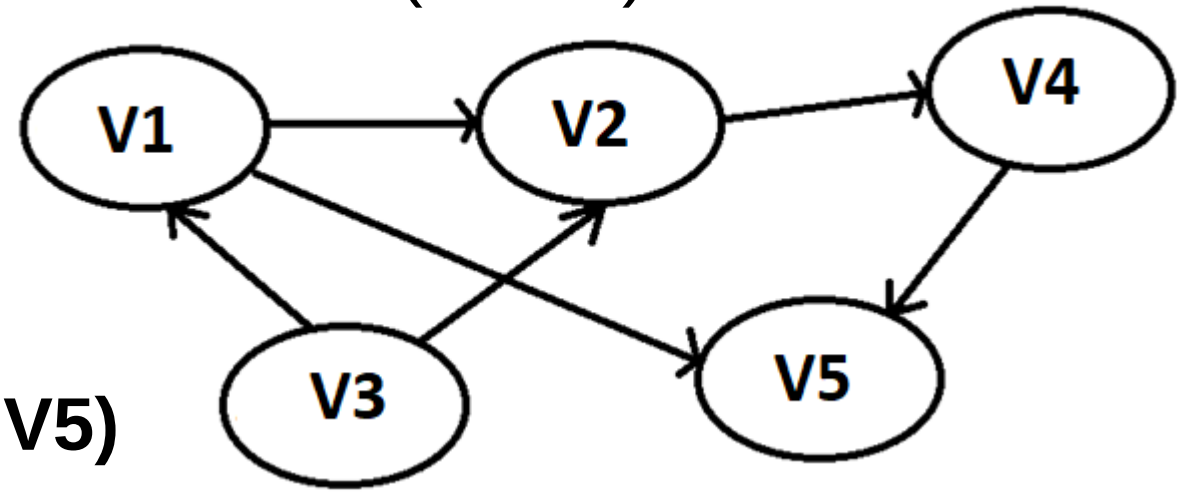
O **peso** de uma aresta pode representar:

- Custo
- Distância
- etc



Grafos

O **caminho** de um vértice V a um vértice U é uma sequência de vértice ligando o vértice V (primeiro) ao vértice U (último).



(V1,V2,V4,V5)

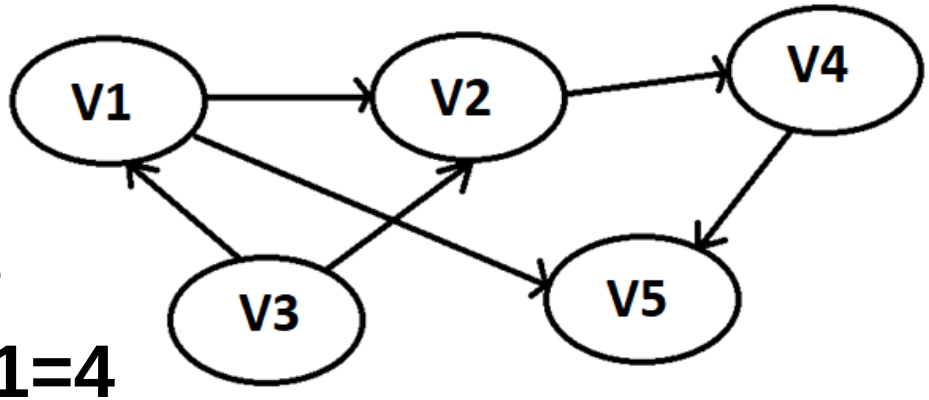
(V3,V1,V2,V4,V5)

(V1,V5)

Grafos

O **Comprimento** de um caminho é igual ao número de vértices(n) menos 1 desse caminho

L=n-1 (grafos sem pesos)



(V1,V2,V4,V5)=4-1=3

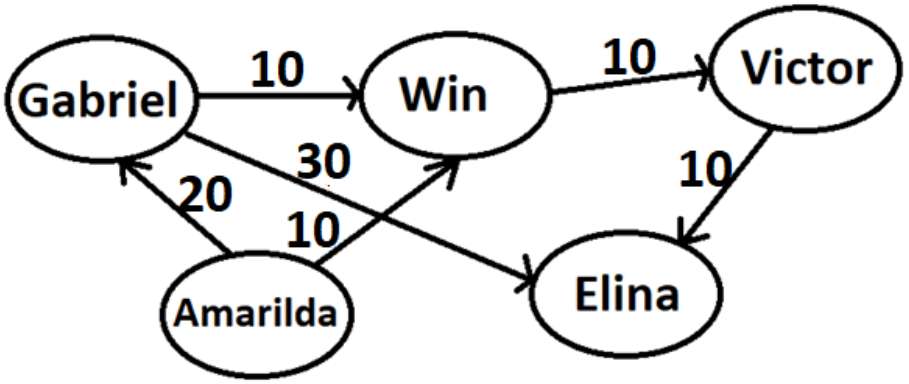
(V3,V1,V2,V4,V5)=5-1=4

(V1,V5)=2-1=1

Se existirem mais de um caminho de u a v, então o comprimento do caminho de u a v será igual ao menor comprimento dentre todos os caminhos de u a v.

Grafos

Para grafos com pesos o **Comprimento** de um caminho é igual a soma dos pesos das arestas desse caminho.



(G-W,W-V,V-E)=10+10+10=30

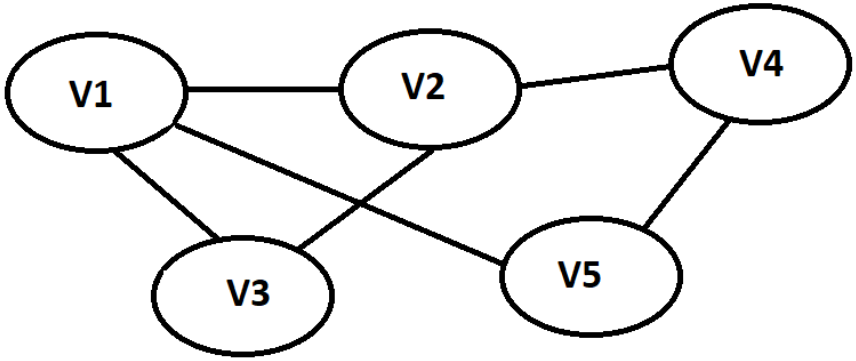
(G-E)=30

Se existirem mais de um caminho de u a v, então o comprimento do caminho de u a v será igual ao menor comprimento dentre todos os caminhos de u a v.

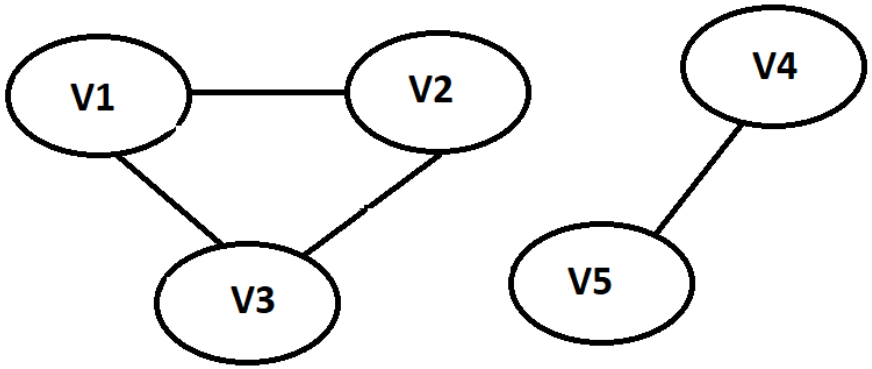
Grafos

Um grafo não direcionado é **conexo** (**conectado**) se cada par de vértices nele estiver conectado por um caminho.

Conexo



Desconexo



Grafos

Um grafo direcionado pode ser:

- **Fortemente conexo:**

  Se existe um caminho entre qualquer par de vértices no grafo.

- **Conexo:**

  Se possui um caminho de V para U ou um caminho de U para V para cada par de vértices.

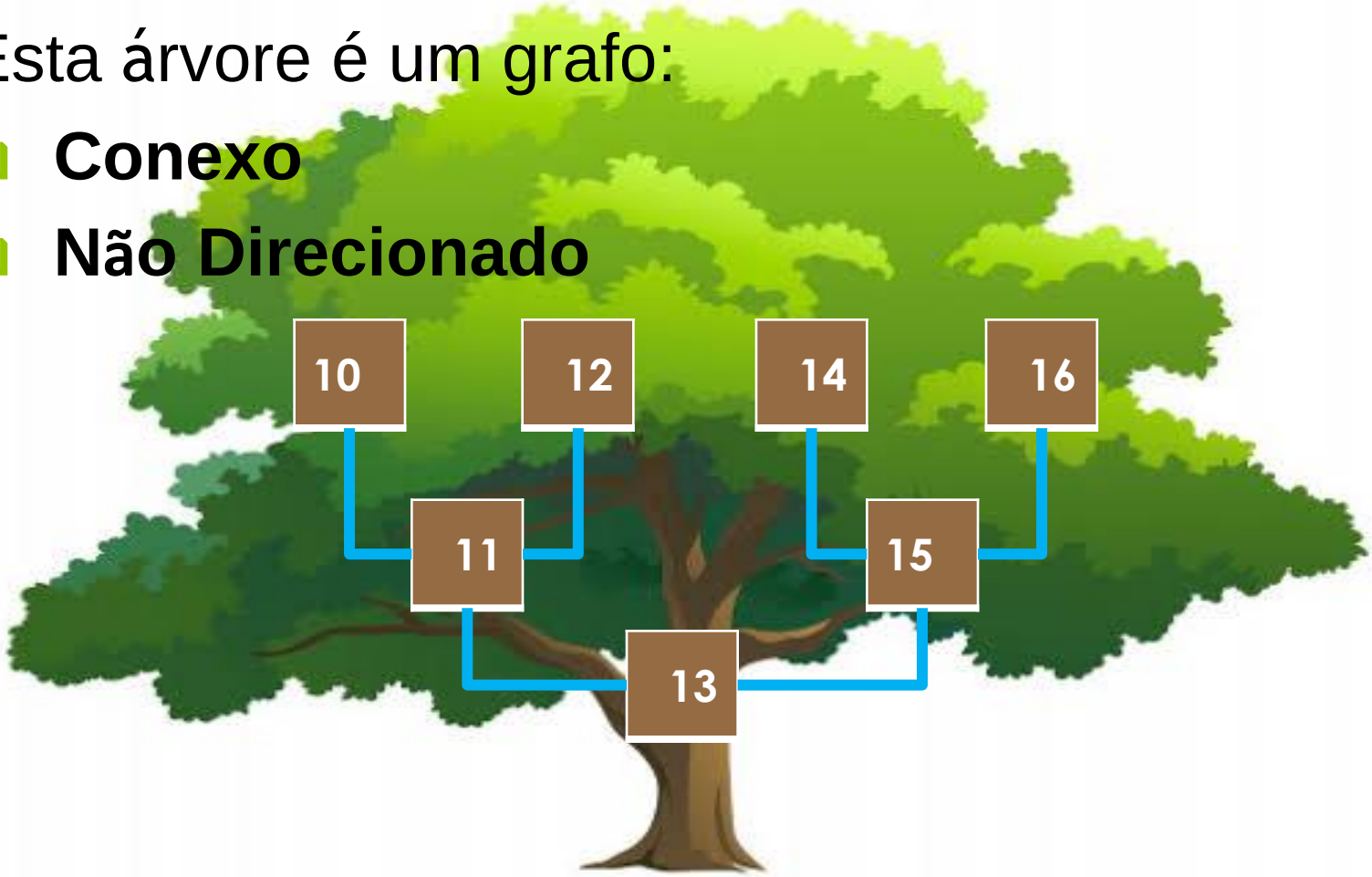
- **Fracamente conexo:**

  Se a substituição de todas as suas arestas não-direcionadas produz grafo conexo

Grafos

Esta árvore é um grafo:

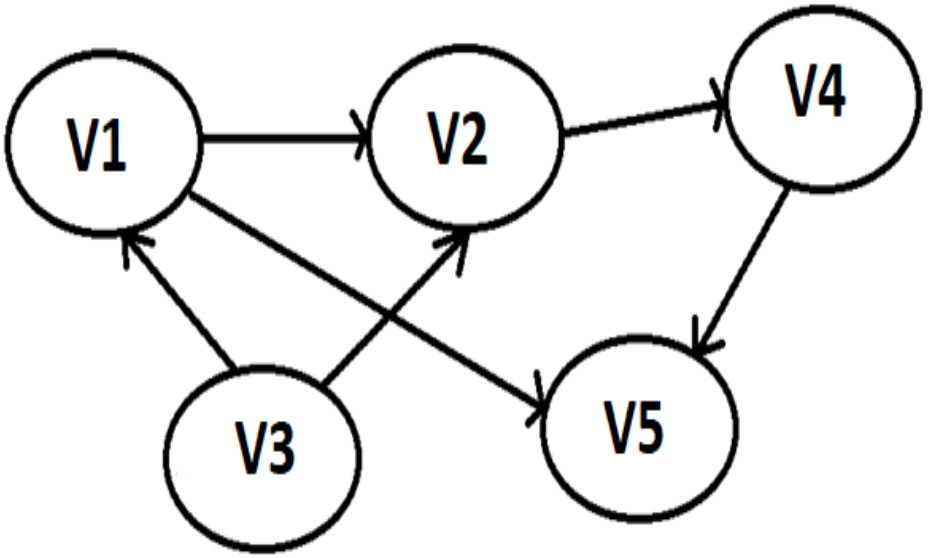
- **Conexo**
- **Não Direcionado**



Grafos

Como representar grafos?

- ✓ Como um mapeamento de cada nó à lista de nós aos quais ele está conectado



Nó	Conectado a
V1	V2, V5
V2	V4
V3	V1, V2
V4	V5
V5	----

Grafos

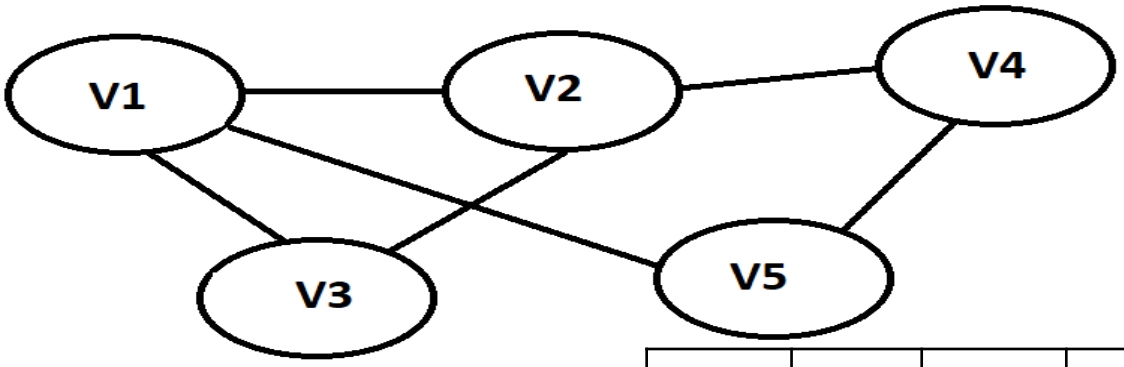
Como representar grafos no computador?

Exitem duas maneiras comuns para representar grafos computacionalmente:

- Matrizes de adjacências
- Listas de adjacência

Grafos

Matrizes de adjacências



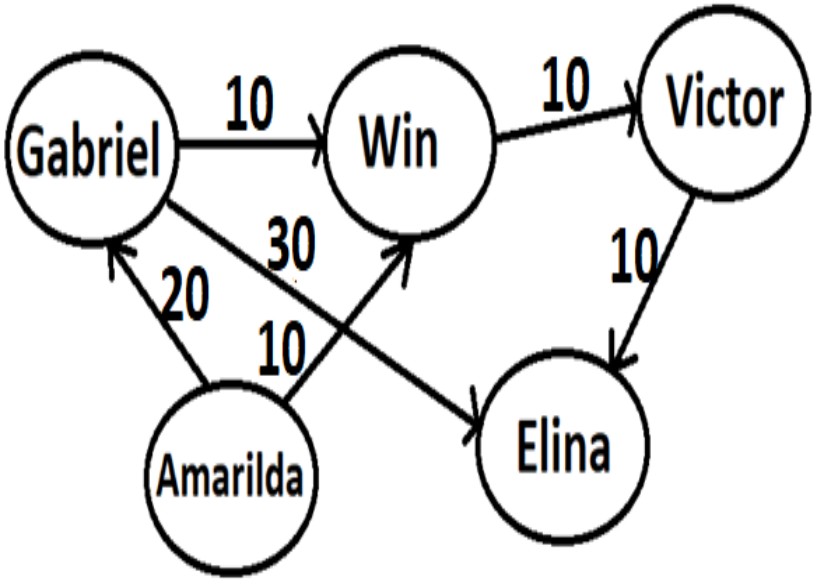
Grafo Não direcionado e sem peso

Nó	V1	V2	V3	V4	V5
V1	0	1	1	0	1
V2	1	0	1	1	0
V3	1	1	0	0	0
V4	0	1	0	0	1
V5	1	0	0	1	0

Grafos

Matrizes de adjacências

Grafo direcionado com peso



Nó	A	E	G	W	V
A	0	0	20	10	0
E	0	0	0	0	0
G	0	10	0	10	0
W	0	0	0	0	10
V	0	10	0	0	0

NB: O zero representa a falta de relacionamento, pode ser trocado por um outro valor dependendo do problema

Grafos

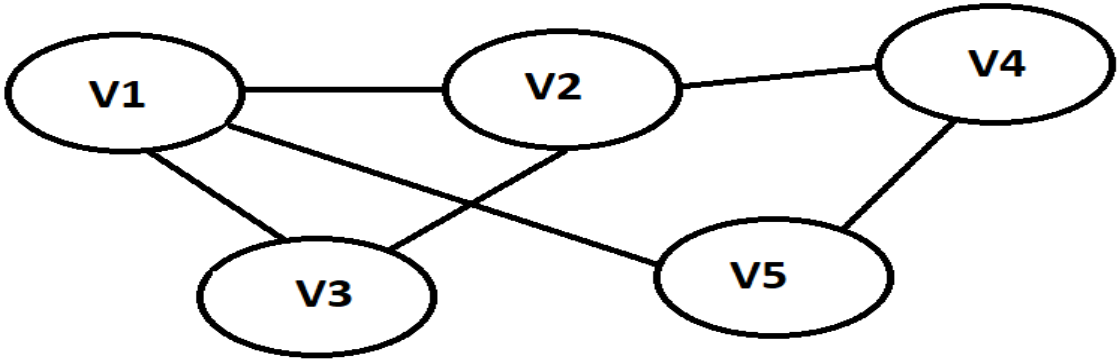
Listas de adjacências

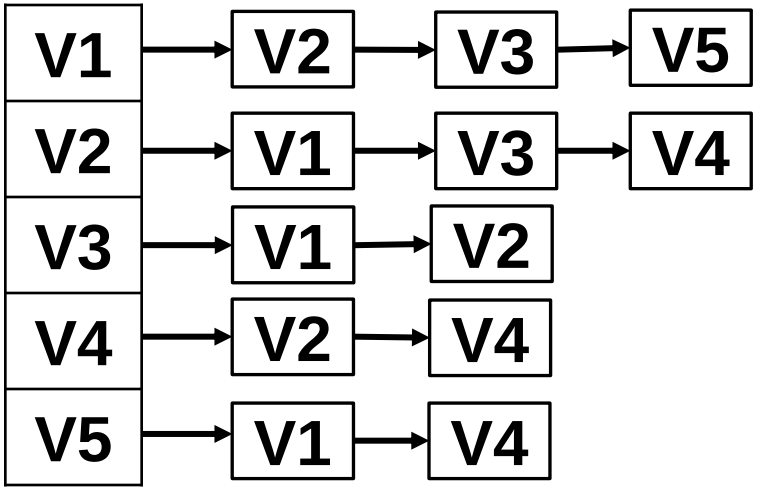
Uma lista de adjacências de um grafo com n vértices consiste de um array de *n* listas, uma para cada vértice no grafo.

Para cada vértice *u*, a lista contém todos os vizinhos de *u*.

Grafos

Listas de adjacências

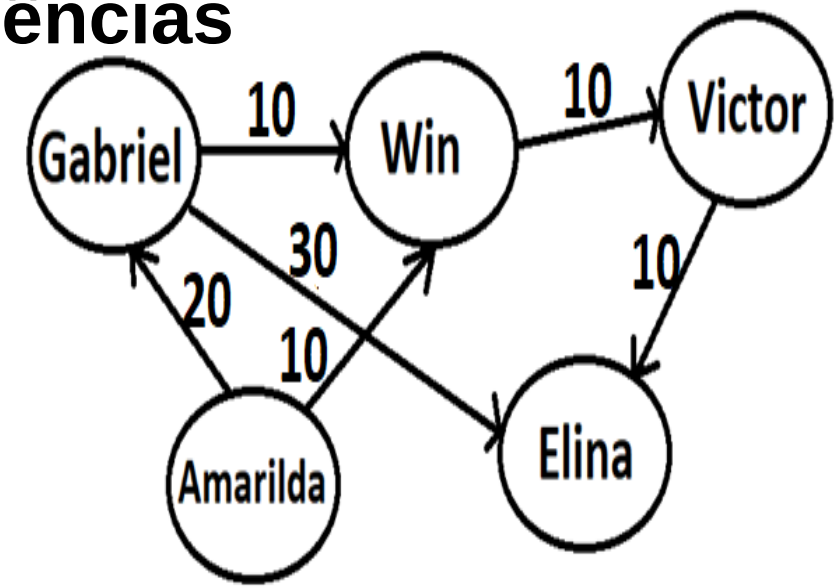


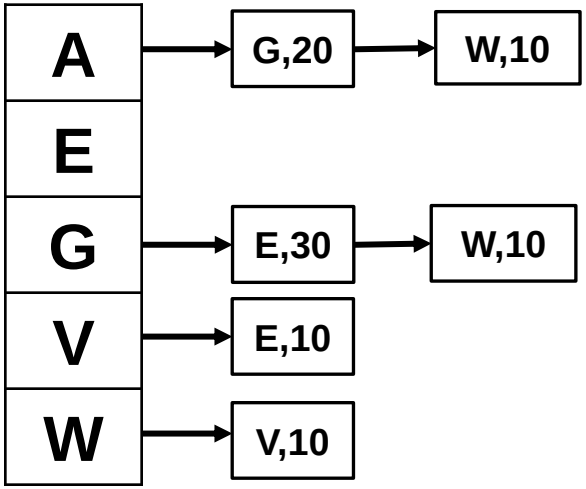


Grafo Não direcionado e sem peso

Grafos

Listas de adjacências





Grafo direcionado com peso

Grafos

Como implementar em JAVA?

```
package aula_pratica_grafos;

import java.util.ArrayList;
public class Grafo<TIPO> {
    private ArrayList<Vertice> vertices;
    private ArrayList<Aresta> arestas;

    public Grafo() {
        this.vertices = new ArrayList<Vertice>();
        this.arestas = new ArrayList<Aresta>();
    }
```

Grafos

Como implementar em JAVA?

```
public void adicionarVertice(TIPO dado) {
    Vertice<TIPO> novoVertice = new Vertice<TIPO>(dado);
    this.vertices.add(novoVertice);
}

public void adicionarAresta(Double peso, TIPO dadoInicio, TIPO dadoFim) {
    Vertice<TIPO> inicio = this.procurarVertice(dadoInicio);
    Vertice<TIPO> fim = this.procurarVertice(dadoFim);
    Aresta<TIPO> aresta = new Aresta<TIPO>(peso, inicio, fim);
    inicio.adicionarArestaSaida(aresta);
    fim.adicionarArestaEntrada(aresta);
    this.arestas.add(aresta);

}
```

Grafos

Como implementar em JAVA?

```
public Vertice<TIPO> procurarVertice(TIPO dado) {
    Vertice<TIPO> vertice = null;
    for (int i = 0; i < this.vertices.size(); i++) {
        if (this.vertices.get(i).getDado().equals(dado)) {
            vertice = this.vertices.get(i);
            break;
        }
    }
    return vertice;
}
```

Grafos

Como implementar em JAVA?

```
public class Aresta<TIPO>{
    private Double peso;
    private Vertice<TIPO> inicio;
    private Vertice<TIPO> fim;

    public Aresta(Double peso, Vertice<TIPO> inicio, Vertice<TIPO> fim){
        this.peso=peso;
        this.inicio=inicio;
        this.fim=fim;
    }
```

Grafos

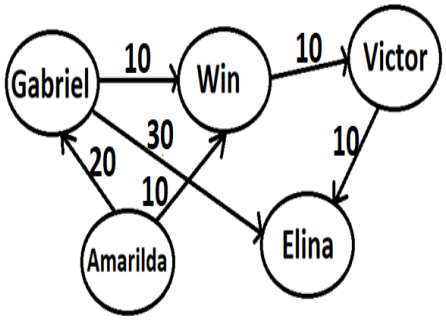
Como implementar em JAVA?

```
public class Vertice<TIPO>{
    private TIPO dado;
    private ArrayList<Aresta> arestasEntrada;
    private ArrayList<Aresta> arestasSaida;

    public Vertice(TIPO valor){
        this.dado=valor;
        this.arestasEntrada=new ArrayList<Aresta>();
        this.arestasSaida=new ArrayList<Aresta>();
    }
```

Grafos

Como implementar em JAVA?




```
public class Aula_Pratica_Grafos {
    public static void main(String[] args) {
        Grafo<String> grafo=new Grafo<String>()
        grafo.adicionarVertice("Gabriel");
        grafo.adicionarVertice("Win");
        grafo.adicionarVertice("Amarilda");
        grafo.adicionarVertice("Elina");
        grafo.adicionarVertice("Victor");
        grafo.adicionarAresta(10.0, "Gabriel", "Win");
        grafo.adicionarAresta(10.0, "Win", "Victor");
        grafo.adicionarAresta(10.0, "Victor", "Elina");
        grafo.adicionarAresta(10.0, "Amarilda", "Win");
        grafo.adicionarAresta(30.0, "Gabriel", "Elina");
        grafo.adicionarAresta(20.0, "Amarilda", "Gabriel");
    }

}
```

Por hoje é tudo