**Grafos:**

**Representação:**

- Orienta a objetos

- Lista de Adjacências

- Matriz de Adjacências

- Matriz de incidências

1:{1,2}, {1,3}, {1,4}

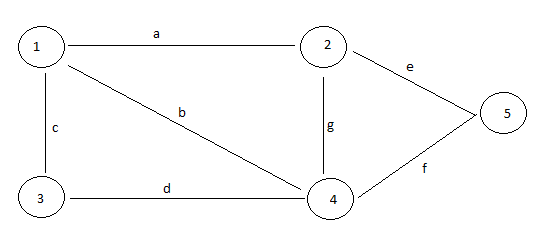
2:{2,1}, {2,4}, {2,5}

3:{3,1}, {3,4}

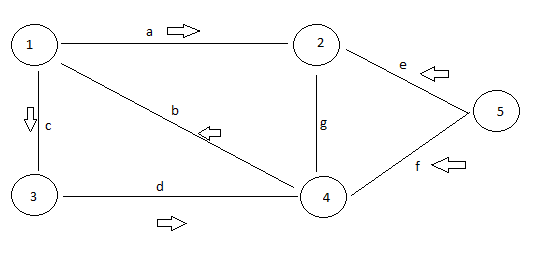
4:{4,3}, {4,2},{4,5},{4,1}

5:{5,2}, {5,4}

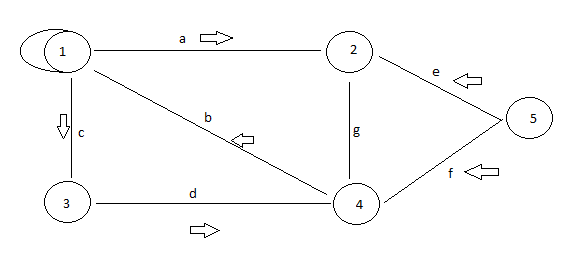
**Grafo simples**



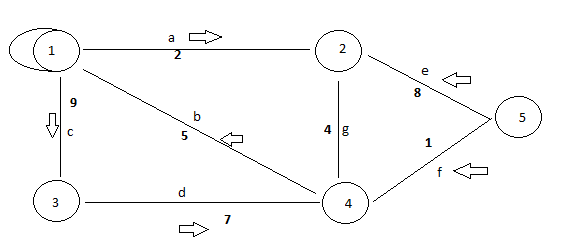
**Grafo que aponta para um próximo**



**Grafo pode apontar para ele mesmo**



**Matrizes:**



**Matriz de adjacências**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Colunas ->  Linhas \/ | 1 | 2 | 3 | 4 | 5 |
| 1 | 0 | 1 | 1 | 1 | 0 |
| 2 | 1 | 0 | 0 | 1 | 1 |
| 3 | 1 | 0 | 0 | 1 | 0 |
| 4 | 1 | 1 | 1 | 0 | 1 |
| 5 | 0 | 1 | 0 | 1 | 0 |

Se o vértice temligação com arestas, se tem, marca 1, se não marca 0.

**Matriz de incidências**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Colunas ->  Linhas \/ | A | B | C | D | E | F | G |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 4 | 0 | 1 | 0 | 1 | 0 | 1 | 1 |
| 5 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |

3 - 0011000

1 - 1110000

And -001

Vértices por arestas, e marca quais arestas um vértice tem.

**Conceitos de Grafos:**

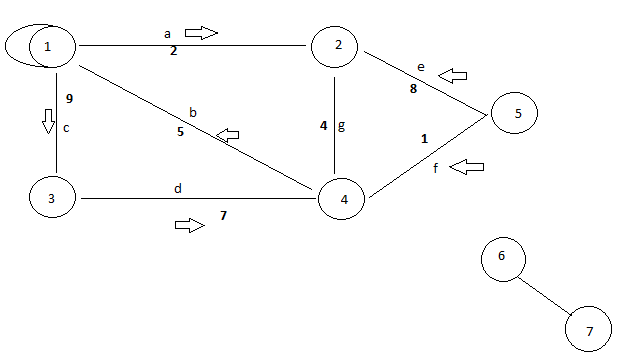
**Direcionados V.S. não direcionados**

O exemplo mais clássico de grafo direcionado é o GPS.

**Ponderados V.S. não Ponderados**

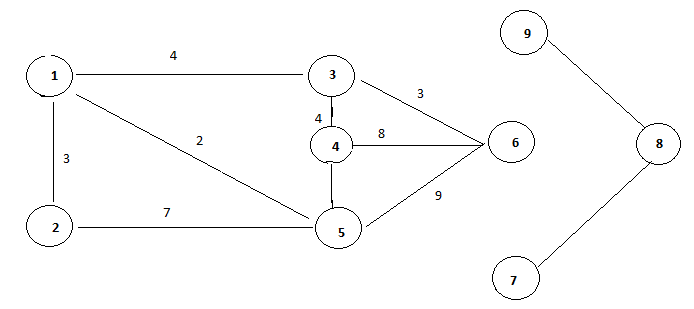
Distancia de um ponto ao outro, no caso do GPS, define o melhor caminho.

**Totalmente Conectados V.S. Não Totalmente conectados**

****

**Problemas em Grafos:**

**Conectividade:**



S = {3,4}

L = {1,2}

X = 2

Enquanto ! S.empty()

X = S.pop()

L.push(x)

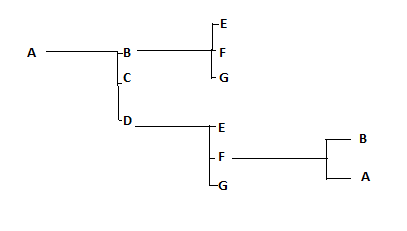
Para cada vértice V adjacente a X e não pertence a L

S.push(x)

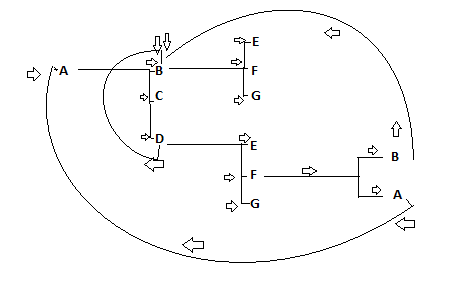
Se a lista gerada não estiver de acordo com o grafo, a lista está errada.

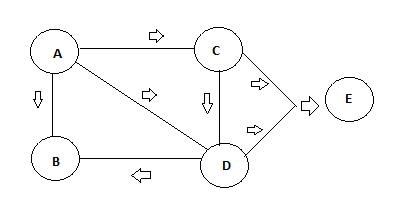
**Verificação em ciclos:**

Problema de loop



Sem problema de loop, todos os diretórios têm chegada e saída definidas.





|  |  |  |
| --- | --- | --- |
|  | IN | OUT |
| A | 0 | 3 |
| B | 2 | 0 |
| C | 1 | 2 |
| D | 2 | 2 |
| E | 2 | 0 |

S = { A}

Enquanto !S.Empty()

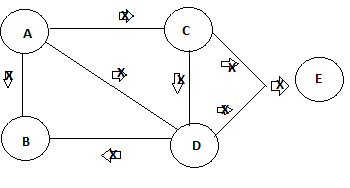
Para cada vértice com chegada de S, remove a aresta. (S,V)

Adiciona a S todos V com in == 0

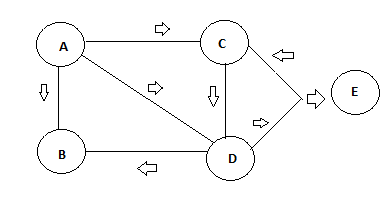
Tabela IN/OUT após algoritmo

|  |  |  |
| --- | --- | --- |
|  | IN | OUT |
| A | 0 | 3 |
| B | 0 | 0 |
| C | 0 | 2 |
| D | 0 | 2 |
| E | 0 | 0 |

**Grafo sem ciclo:**

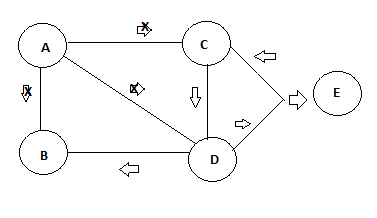


**Proximografo:**



|  |  |  |
| --- | --- | --- |
|  | IN | OUT |
| A | 0 | 3 |
| B | 2 | 0 |
| C | 2 | 1 |
| D | 2 | 2 |
| E | 1 | 1 |

**Grafo com ciclo:**



|  |  |  |
| --- | --- | --- |
|  | IN | OUT |
| A | 0 | 3 |
| B | 1 | 0 |
| C | 1 | 1 |
| D | 1 | 2 |
| E | 1 | 1 |

G é conjunto V e A

Representação:

- Lista de adjacências (v,v)

- Matriz de adjacência (v,v)

- Matriz de incidência(v,a)

Classificação:

- Direcionadas

- Ponderadas

-Cíclicas

-Conectadas

Algoritmos:

- Buscas:

- Em largura

- Em profundidade

- Verificações:

- de conectividade

- de DAG ( Directlyaciclycgraph ( Grafos acíclicos direto))