Estructura de Datos: Grafo

Dr. Jaime Osorio Ubaldo

Introducción

Grafo

Un grafo es una representación abstracta de las relaciones

Grafo

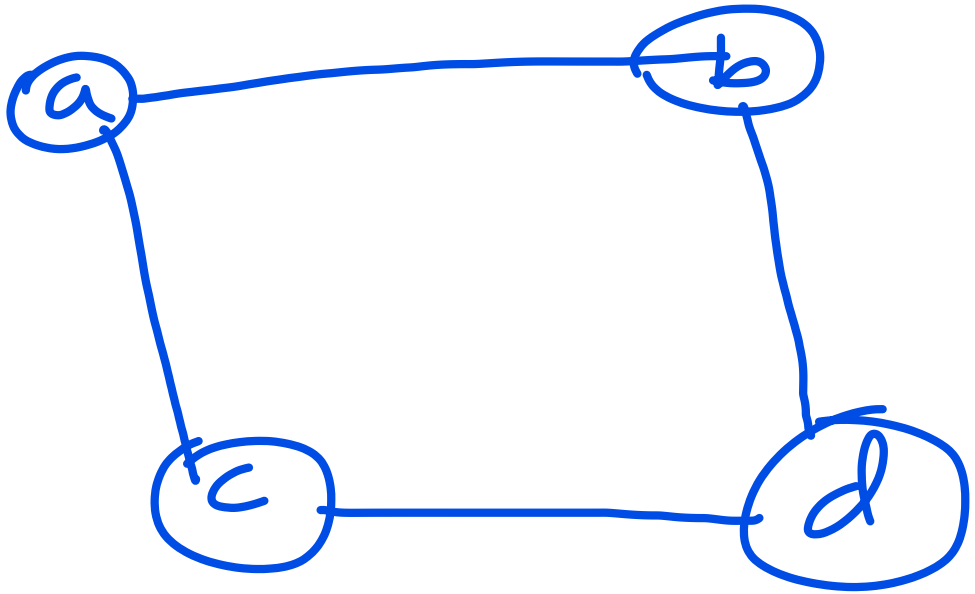
Un grafo es una representación abstracta de las relaciones(denominadas aristas o arcos)

Grafo

Un grafo es una representación abstracta de las relaciones(denominadas aristas o arcos)entre una serie de entidades

Grafo

Un grafo es una representación abstracta de las relaciones(denominadas aristas o arcos)entre una serie de entidades(denominadas vértices o nodos).



Grafo

Un grafo es una representación abstracta de las relaciones(denominadas aristas o arcos)entre una serie de entidades(denominadas vértices o nodos).Por ejemplo las redes sociales,

Grafo

Un grafo es una representación abstracta de las relaciones(denominadas aristas o arcos)entre una serie de entidades(denominadas vértices o nodos).Por ejemplo las redes sociales,redes de transporte como carreteras o trenes,

Grafo

Un grafo es una representación abstracta de las relaciones(denominadas aristas o arcos)entre una serie de entidades(denominadas vértices o nodos).Por ejemplo las redes sociales,redes de transporte como carreteras o trenes,redes de ordenadores,

Grafo

Un grafo es una representación abstracta de las relaciones(denominadas aristas o arcos)entre una serie de entidades(denominadas vértices o nodos).Por ejemplo las redes sociales,redes de transporte como carreteras o trenes,redes de ordenadores,organigramas empresariales,

Grafo

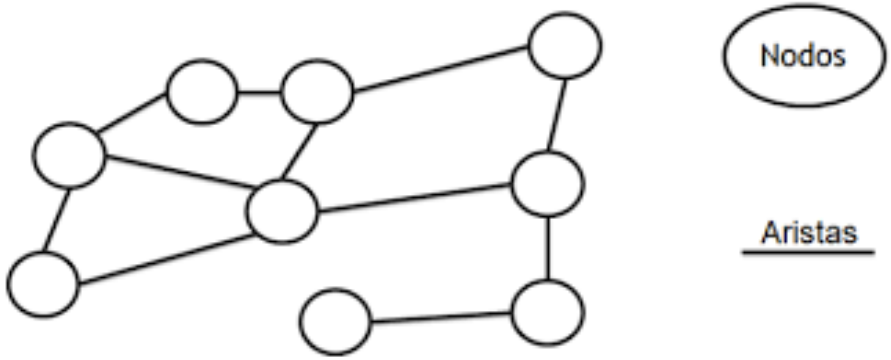
Un grafo es una representación abstracta de las relaciones(denominadas aristas o arcos)entre una serie de entidades(denominadas vértices o nodos).Por ejemplo las redes sociales,redes de transporte como carreteras o trenes,redes de ordenadores,organigramas empresariales,redes eléctricas,

Grafo

Un grafo es una representación abstracta de las relaciones(denominadas aristas o arcos)entre una serie de entidades(denominadas vértices o nodos).Por ejemplo las redes sociales,redes de transporte como carreteras o trenes,redes de ordenadores,organigramas empresariales,redes eléctricas,nuestro propio sistema nervioso es una red de neuronas conectadas entre sí.

Grafo

Un grafo es una representación abstracta de las relaciones(denominadas aristas o arcos)entre una serie de entidades(denominadas vértices o nodos).Por ejemplo las redes sociales,redes de transporte como carreteras o trenes,redes de ordenadores,organigramas empresariales,redes eléctricas,nuestro propio sistema nervioso es una red de neuronas conectadas entre sí.

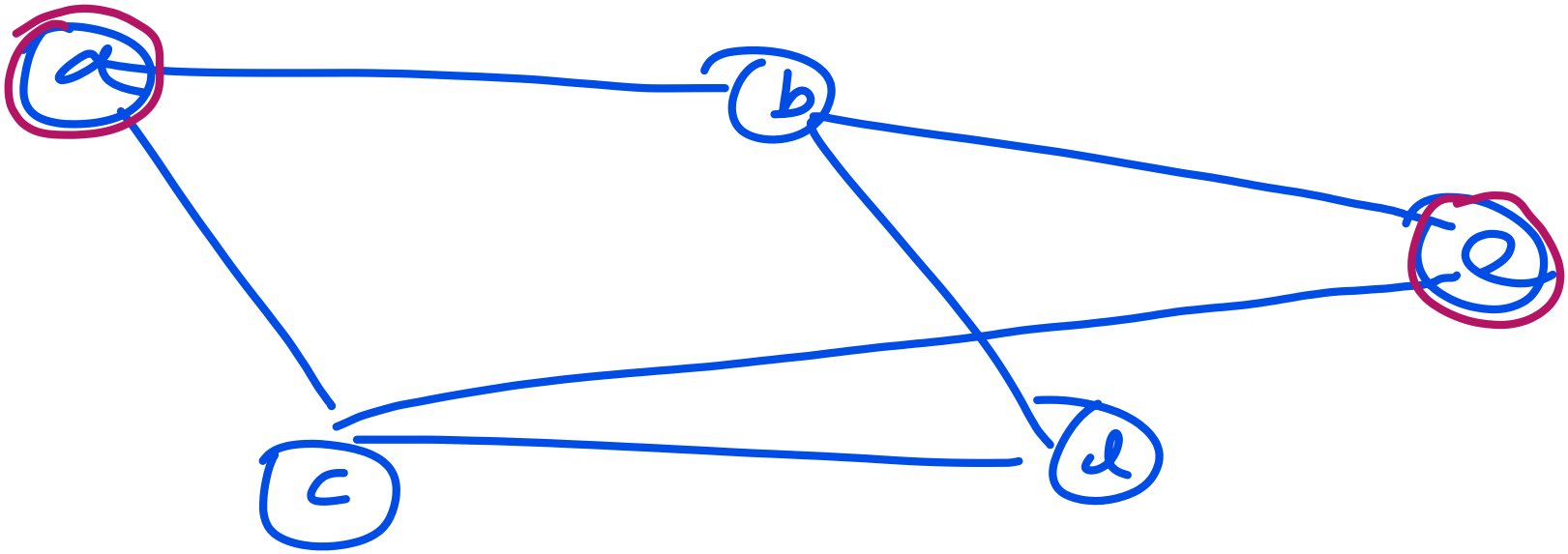


Aplicaciones

La teoría de grafos intenta resolver situaciones muy cotidianas,

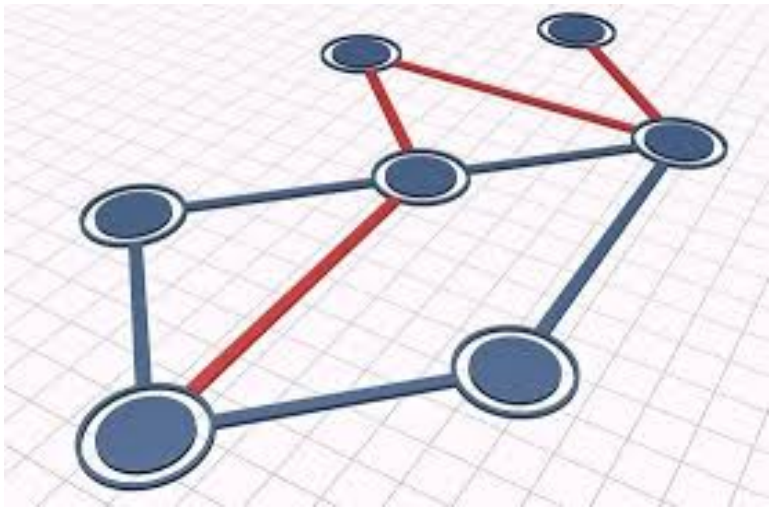
Aplicaciones

La teoría de grafos intenta resolver situaciones muy cotidianas,como puede ser la búsqueda del camino más corto entre dos vértices.



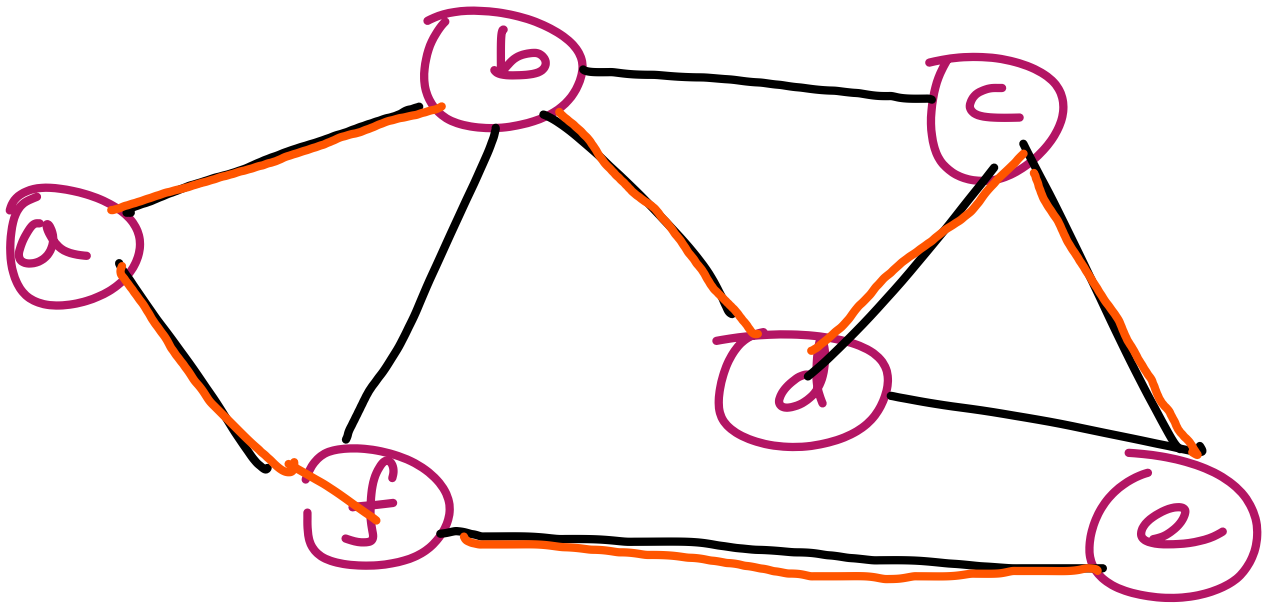
Aplicaciones

La teoría de grafos intenta resolver situaciones muy cotidianas,como puede ser la búsqueda del camino más corto entre dos vértices.Este problema, aparentemente tan simple, es una parte importante del cálculo de la ruta que hacen los navegadores GPS.



Aplicaciones

Otro problema famoso es el del viajante de comercio:

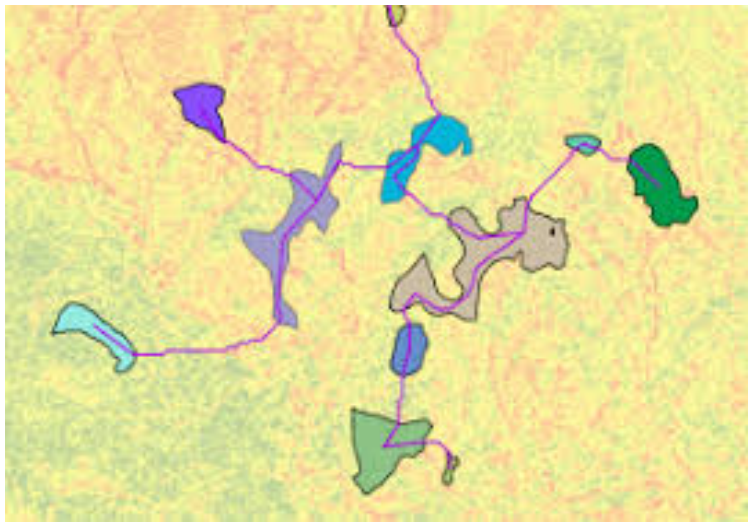


Aplicaciones

Otro problema famoso es el del viajante de comercio:encontrar la ruta más corta que pasa por un conjunto de ciudades y regresa a la ciudad de origen.

Aplicaciones

Otro problema famoso es el del viajante de comercio:encontrar la ruta más corta que pasa por un conjunto de ciudades y regresa a la ciudad de origen.En este caso, la fama viene de la complejidad de calcular la ruta óptima.



Grafo

Un grafo no dirigido

Grafo

Un grafo no dirigido es un conjunto de vértices V

Grafo

Un grafo no dirigido es un conjunto de vértices V y un conjunto de Aristas A,

Grafo

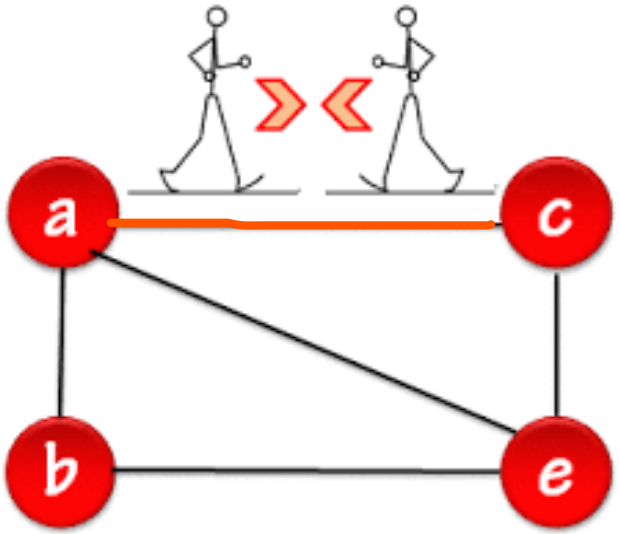
Un grafo no dirigido es un conjunto de vértices V y un conjunto de Aristas A,denotado por $G(V, A)$.

Grafo

Un grafo no dirigido es un conjunto de vértices V y un conjunto de Aristas A,denotado por $G(V, A)$. Una arista es denotada por {v,w} donde v y w son denominados vértices adyacentes.

Grafo

Un grafo no dirigido es un conjunto de vértices V y un conjunto de Aristas A,denotado por $G(V, A)$. Una arista es denotada por {v,w} donde v y w son denominados vértices adyacentes.



{a, c}

{c, e}

{a, e}

⋮

Grafo Dirigido

Un grafo dirigido

Grafo Dirigido

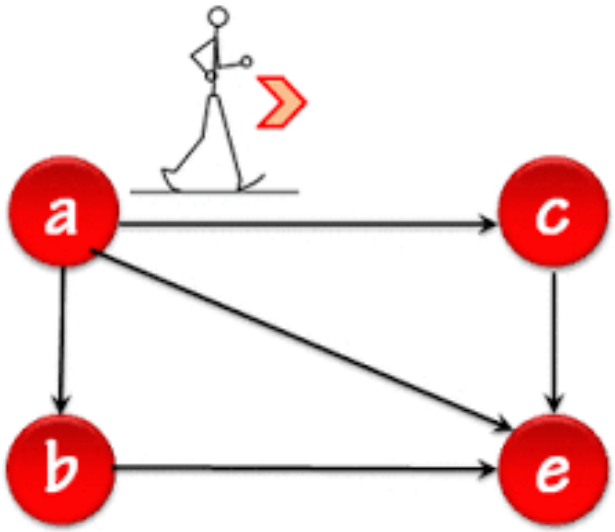
Un grafo dirigido es un conjunto de vértices V y un conjunto de Aristas (dirigidas) A, denotado por $G(V, A)$

Grafo Dirigido

Un grafo dirigido es un conjunto de vértices V y un conjunto de Aristas (dirigidas) A, denotado por $G(V, A)$Una arista es el par ordenado (v,w) donde v es el vértice cola y w es el vértice cabeza.

Grafo Dirigido

Un grafo dirigido es un conjunto de vértices V y un conjunto de Aristas (dirigidas) A, denotado por $G(V, A)$Una arista es el par ordenado (v,w) donde v es el vértice cola y w es el vértice cabeza.



(a, c)

(c, e)

Representación de un Grafo

Matriz de adyacencia.

	a	b	c	e
a	0	1	1	1
b	0	0	0	1
c	0	0	0	1
e	0	0	0	0

Representación de un Grafo

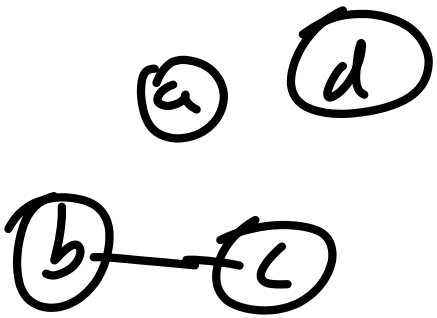
Matriz de adyacencia. Se puede aprovechar la simetría, usa bits(0 y 1) en lugar de enteros.

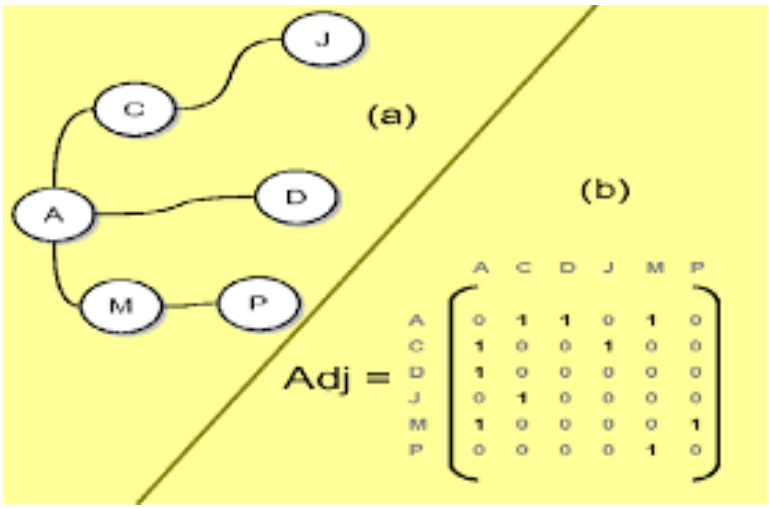
Representación de un Grafo

Matriz de adyacencia. Se puede aprovechar la simetría, usa bits(0 y 1) en lugar de enteros.Ideal para grafos densos.

Representación de un Grafo

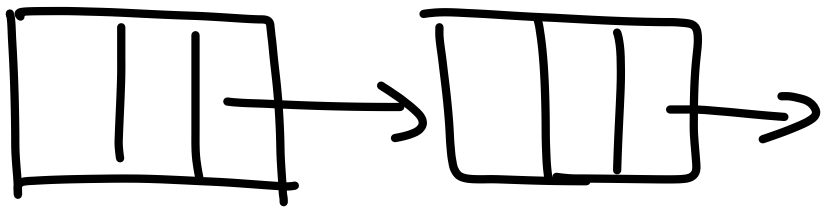
Matriz de adyacencia. Se puede aprovechar la simetría, usa bits(0 y 1) en lugar de enteros.Ideal para grafos densos.





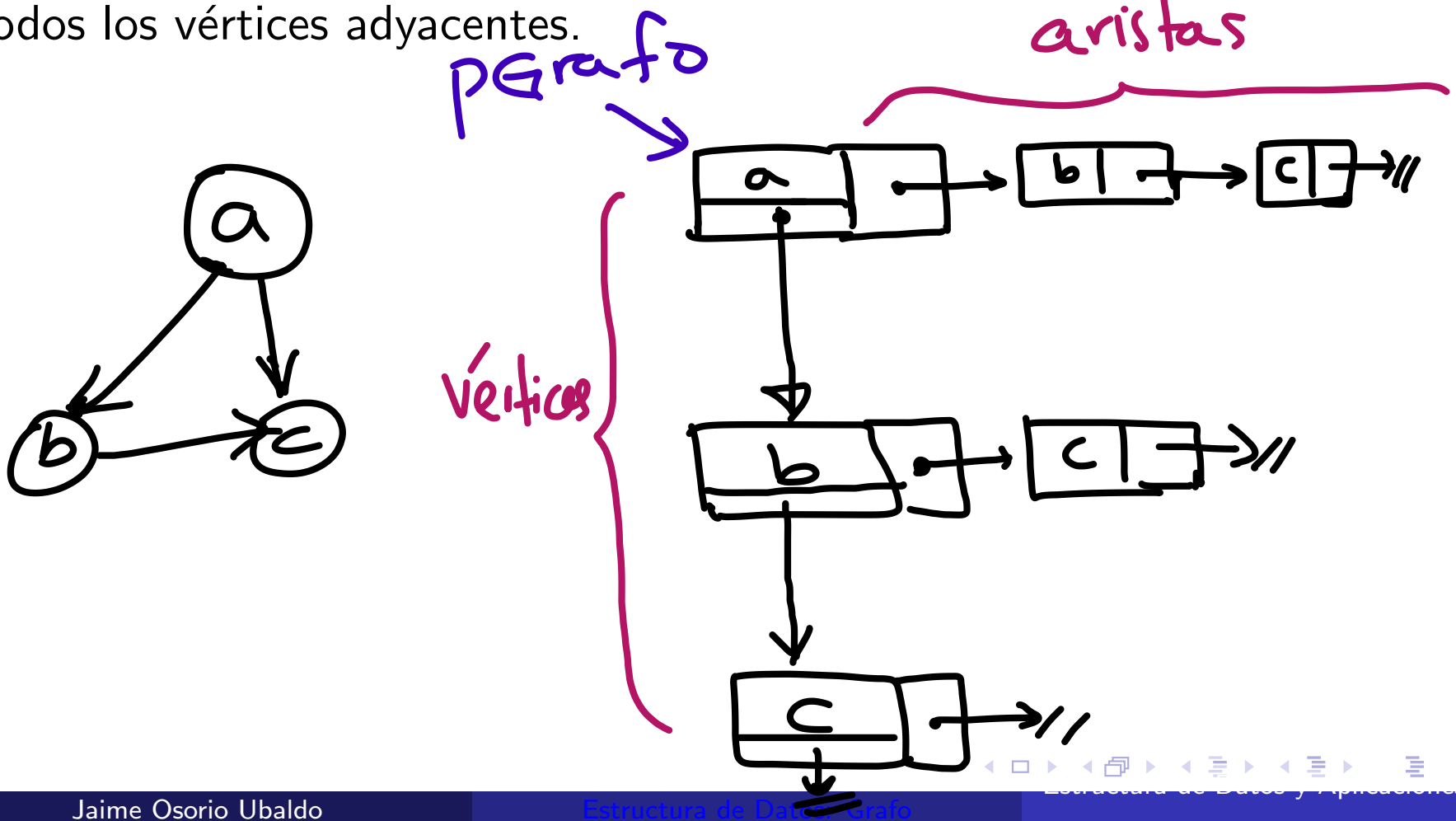
Representación de un Grafo

Lista de adyacencia.



Representación de un Grafo

Lista de adyacencia.Para cada vértice tenemos una lista enlazada con todos los vértices adyacentes.

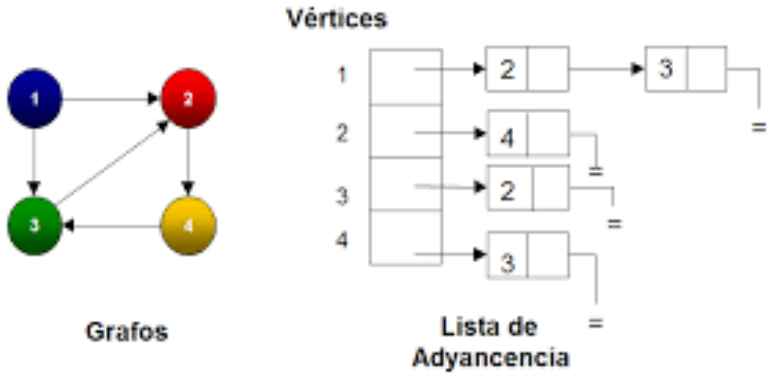


Representación de un Grafo

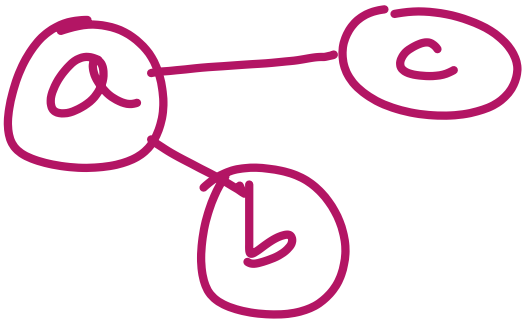
Lista de adyacencia.Para cada vértice tenemos una lista enlazada con todos los vértices adyacentes.Ideal para grafos no densos.

Representación de un Grafo

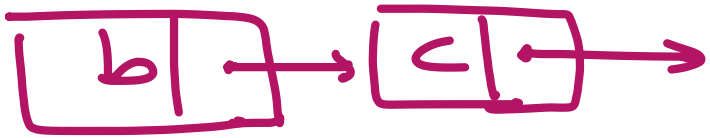
Lista de adyacencia.Para cada vértice tenemos una lista enlazada con todos los vértices adyacentes.Ideal para grafos no densos.



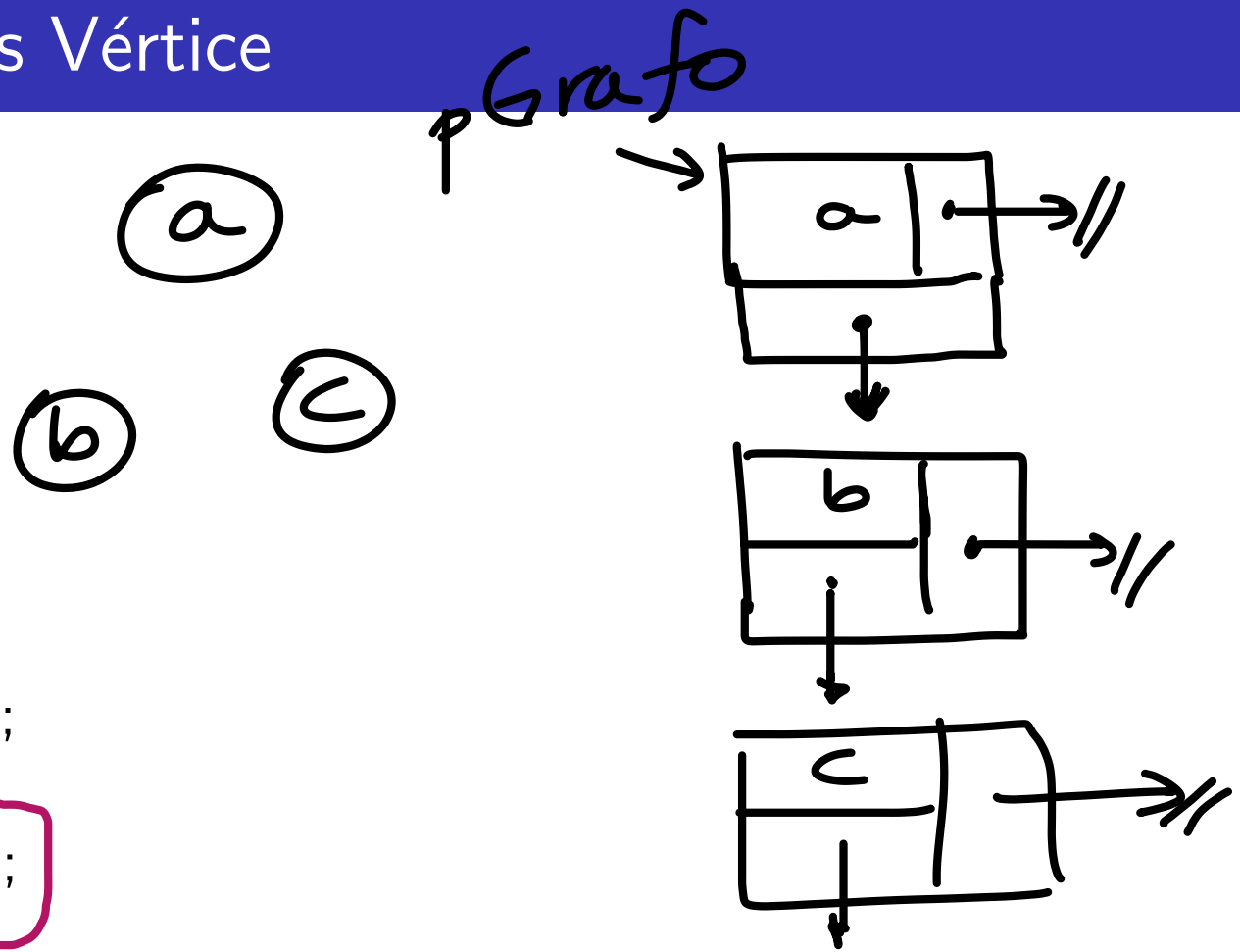
Estructura de Datos Arista




```
struct arista{
        int datoDestino;
        arista *sgteArista;
};
typedef arista *parista;
```



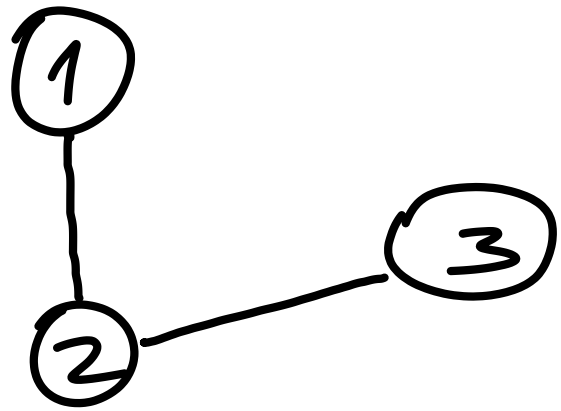
Estructura de Datos Vértice



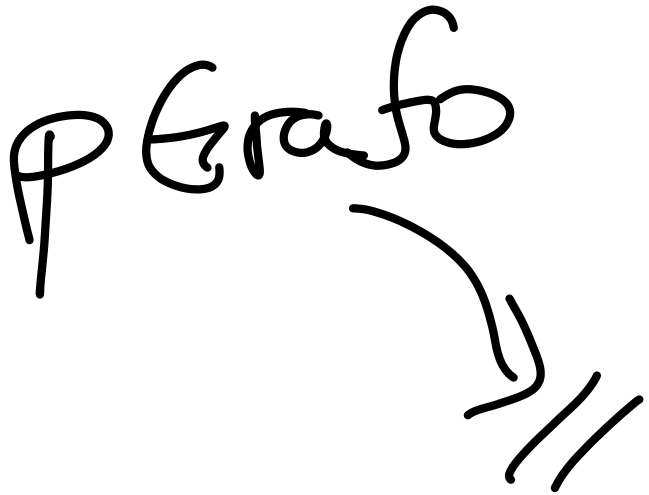

```
struct vertice{
        int datoOrigen;
        arista *adyacente;
        vertice *sgteVertice;
};
typedef vertice *pvertice;
```

Clase Grafo

```
class grafo{
private:
      pvertice pGrafo;
public:
      grafo();
    ~grafo();
      void insertarVertice(int);
      void insertarArista(int,int);
      void imprimirGrafo();
};
```



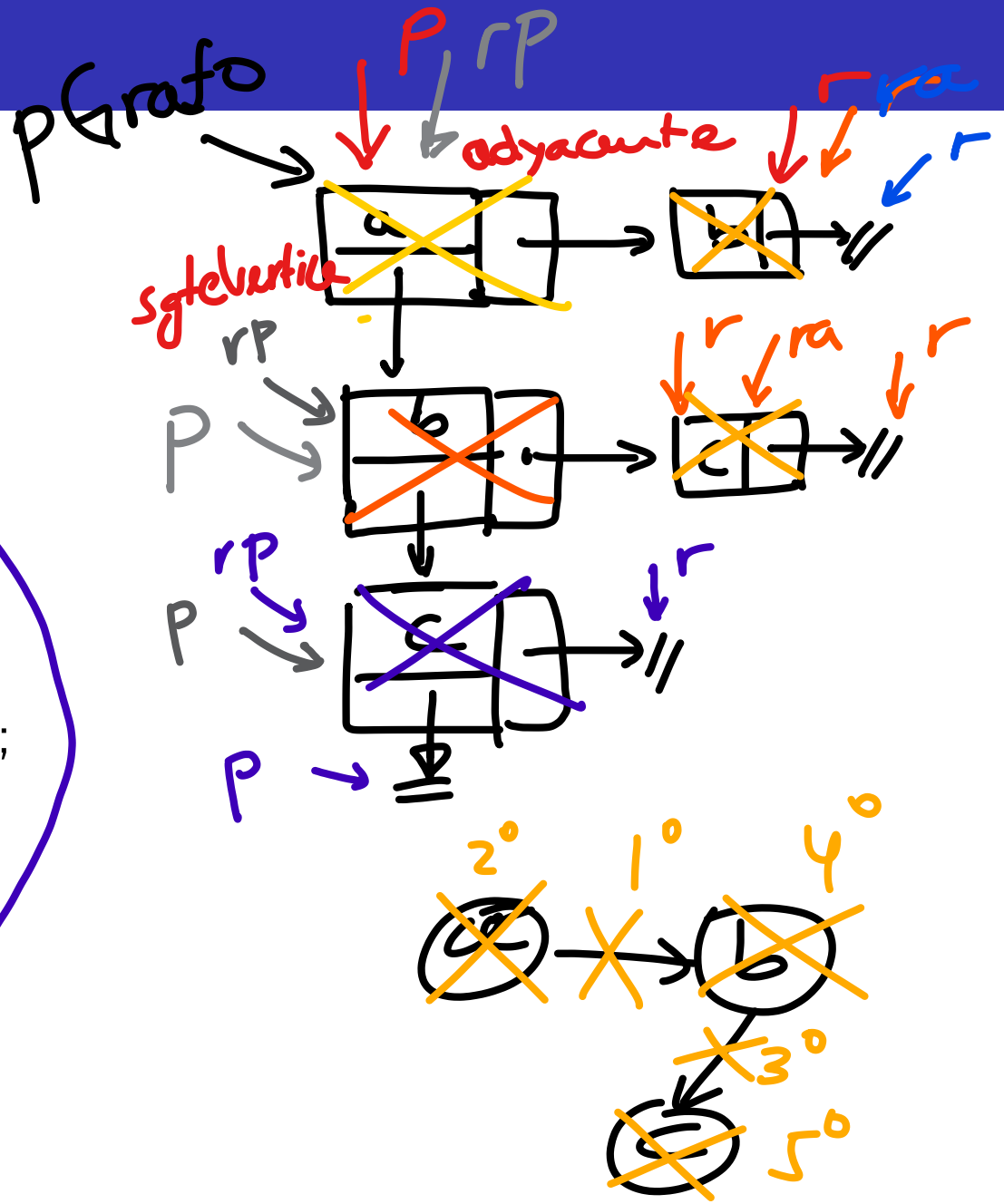
Constructor




```
grafo::grafo(){
      pGrafo=NULL;
}
```

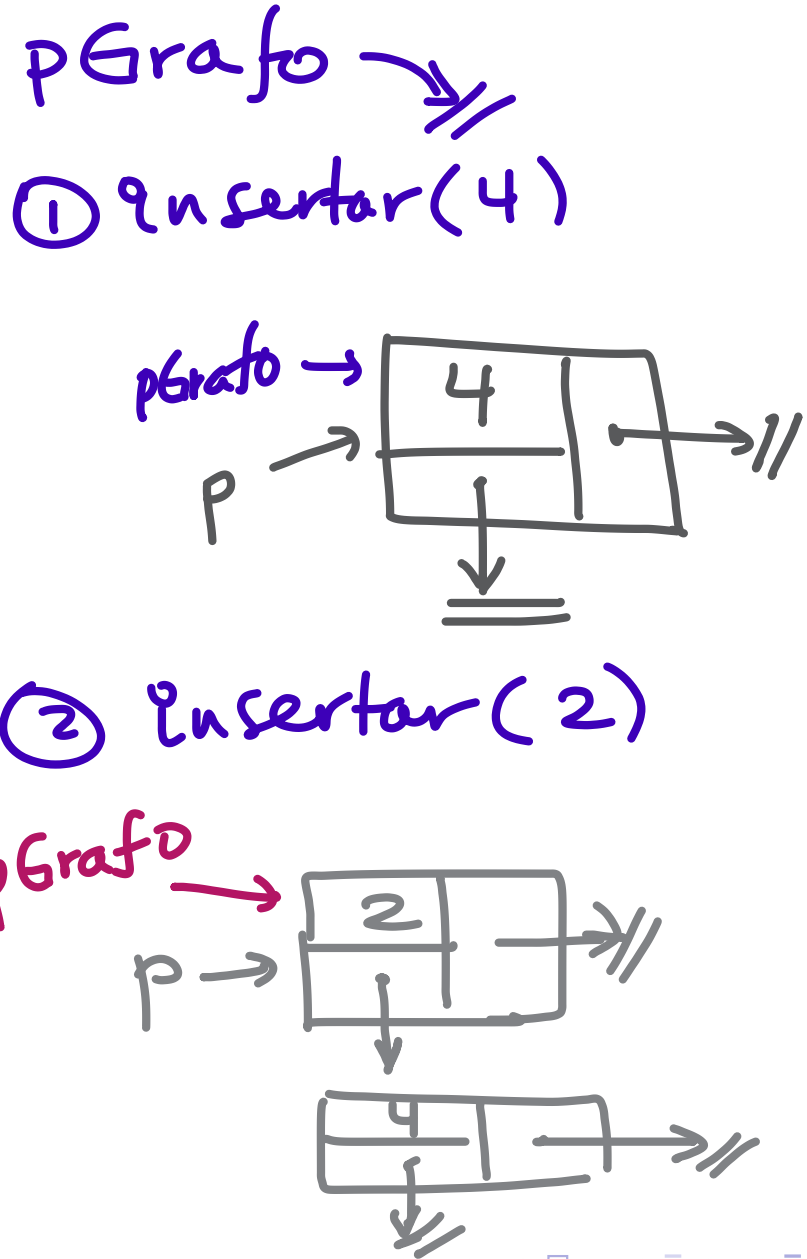
Destructor

```
grafo::~grafo(){
        pvertice p,rp;
        parista r,ra;
        p=pGrafo;
        while(p!=NULL){
                r=p->adyacente;
                while(r!=NULL){
                        ra=r;
                        r=r->sgteArista;
                        delete ra;
                }
                rp=p;
                p=p->sgteVertice;
                delete rp;
        }
}
```

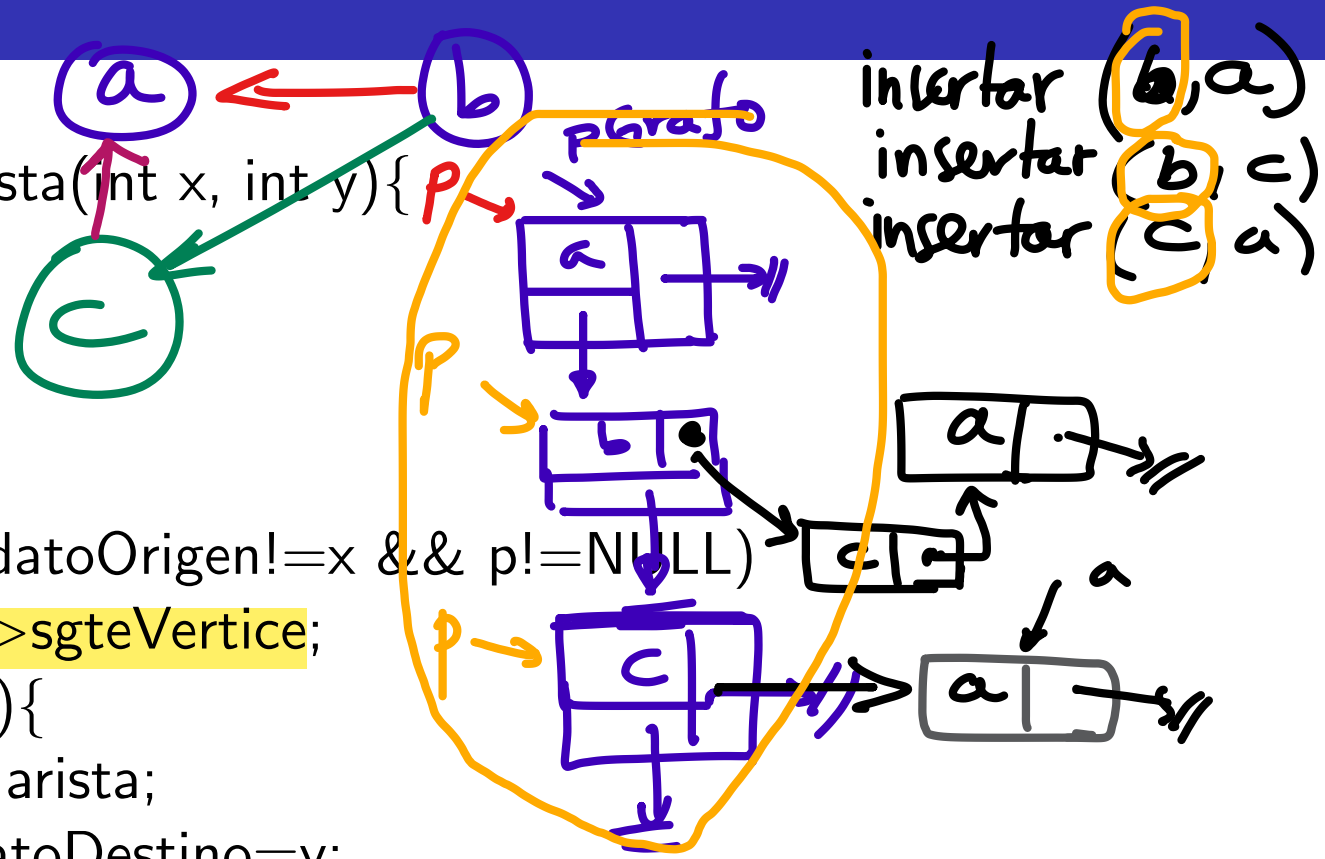


Insertar Vértice

```
void grafo::insertarVertice(int x){
        pvertice p;
        p=new vertice;
        p->datoOrigen=x;
        p->adyacente=NULL;
        p->sgteVertice=pGrafo;
        pGrafo=p;
}
```



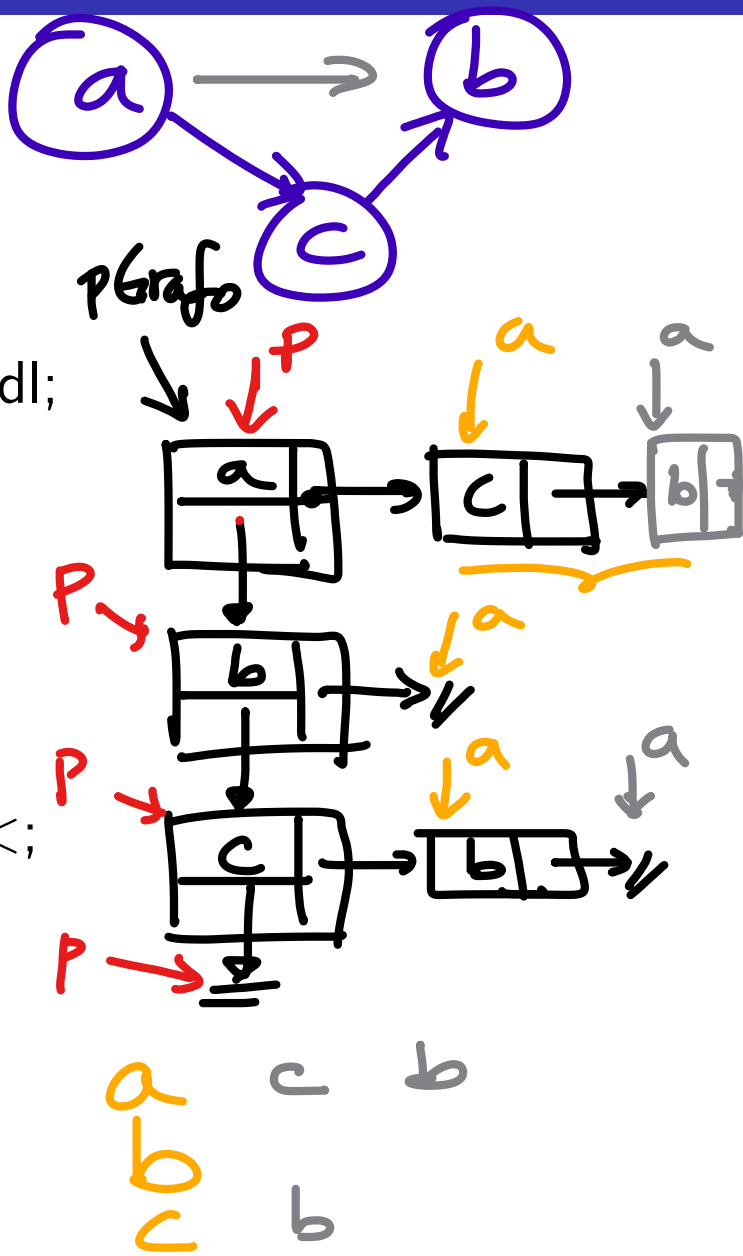
Insertar Arista




```
void grafo::insertarArista(int x, int y){
      pvertice p;
      parista a;
      p=pGrafo;
      if(p!=NULL){
            while(p->datoOrigen!=x && p!=NULL)
                  p=p->sgteVertice;
            if(p!=NULL){
                  a=new arista;
                  a->datoDestino=y;
                  a->sgteArista=p->adyacente;
                  p->adyacente=a;
                }
      }
}
```

Imprimir Grafo

```
void grafo::imprimirGrafo(){
        pvertice p;
        parista a;
        p=pGrafo;
        if(p==NULL) cout<<"Grafo vacio"<<endl;
        else
            while(p!=NULL){
                cout<<p- > datoOrigen<<;
                a=p- >adyacente;
                while(a!=NULL){
                    cout<<a- >datoDestino<<;
                    a=a- >sgteArista;
                }
                cout<<endl;
                p=p- >sgteVertice;
            }
}
```



Ejecutar Grafo

```
int main(){
    grafo g;
    int x,y;
    g.insertarVertice(4);
    g.insertarVertice(6);
    g.insertarVertice(3);
    g.insertarArista(4,6);
    g.insertarArista(3,6);
    g.insertarArista(3,4);
    cout<<"Vert||Aristas"<<endl;
    g.imprimirGrafo();
}
```

Tarea

En cada arista agregar un campo llamando peso y crear una función que sume todos los pesos de las aristas del grafo.