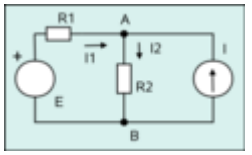
Grafos:
Fundamentos
Representaciones, etc.

Jose Aguilar

Introducción

Las **estructura de datos no lineales** se caracterizan por tener una **relación de adyacencia genérica** entre sus elementos, es decir, un elemento puede estar relacionado con cero, uno o más elementos.

- Hay **múltiples aplicaciones** de estas estructuras
- **Los grafos** son un ejemplo de esas estructuras, que se utilizan para **modelar diversas situaciones** tales como: sistemas de comunicación, redes de amigos, etc.
- Los grafos también son **muy utilizados en computación:** modelar programas paralelos, planificar tareas en el computador, planificar operaciones de E/S, etc.

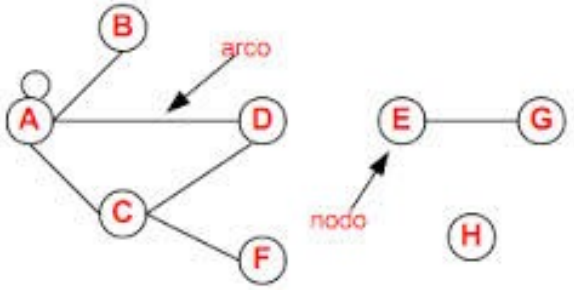


Introducción

La estructura no lineal de datos más general es el grafo,

Un grafo **G(N, A, f)** es un conjunto no vacío de:

- **$N=\{n_1, n_2, \ldots, n_M\}$** nodos o vértices,
- **$A=\{a_1, a_2, \ldots, a_K\}$** aristas o arcos
- La función **$f : R \rightarrow N \times N$** que indica cuales pares de nodos están relacionados.

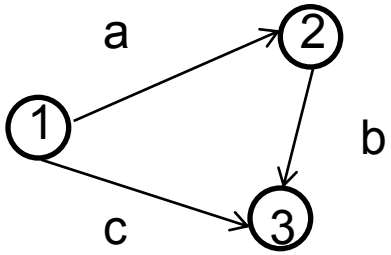


Introducción

Hay dos tipos de grafos según si sus relaciones (arcos) tienen dirección o no,

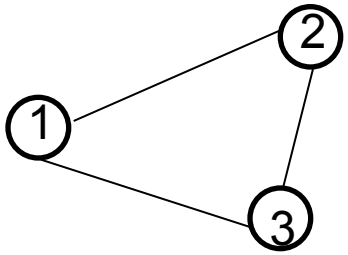
Grafos dirigidos o digrafos

D = {N, A, f} y tiene
N = {1, 2, 3},
A = {a, b, c, d} y
f(a) = (1, 2), f(b) = (2, 3),
f(c) = (1, 3),



f : A → N x N pares ordenados de nodos que indican de que nodo sale el arco y a que nodo llega ese arco.

Grafos no dirigidos



f : L → N x N de pares no ordenados que indican que esos nodos están unidos en ambas direcciones

Conceptos

Para un $a_l \in A$ asociado al par de nodos (n_i, n_j), donde $n_i \; n_j \in N$, si es dirigido y comienza en n_i y termina en n_j, n_i es el **nodo inicial** y n_j es el **nodo terminal**.

- **Arco incidente:** aj es un arco incidente sobre n_k , si n_k es el **nodo terminal** de a_j (digrafos).

.

- **Arcos adyacentes:** a_i y a_j son arcos adyacentes, si a_i y a_j son incidentes en el mismo nodo (digrafos).

- **Arcos paralelos**: Dos arcos son paralelos si ellos comienzan y terminan en los mismos nodos.

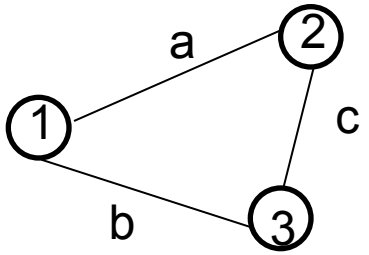
Conceptos

Bucle o lazo: Si un arco a_l comienza en n_i y termina en n_j e i=j, entonces n_{ij} es un bucle.

Ejemplo: Si f(w) = (3, 3) entonces w es un lazo.

Nodos adyacentes: Dado un par de nodos (n_i , n_j) que están unidos por el arco a_l, se dice que n_i es adyacente a n_j por a_l.

Ejemplo: 1 es adyacente a 2 por a.



Conceptos

Grado de incidencia positivo: El grado de incidencia positivo de un nodo n_i es el número de arcos que tienen como **nodo inicial** a n_i (digrafos).

Grado de incidencia negativo: El grado de incidencia negativo de un nodo n_i es el número de arcos que **terminan** en n_i (digrafos).

.

Grado de un nodo:

- **Para digrafos** es el grado de incidencia positivo menos el grado de incidencia negativo del nodo.
- **Para grafos no dirigidos** son los arcos conectados a él.

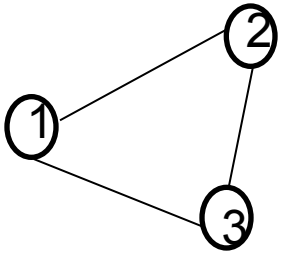
.

Conceptos

Multiplicidad de un par ordenado (n_i , n_j): Dado el digrafo D. Si para un par ordenado de nodos (n_i, n_j) ∈ N x N, hay k arcos distintos en A para los que f(a_l) = (n_i, n_j), es decir, hay k arcos que unen n_i con n_j, se dice que el par (n_i, n_j) tiene multiplicidad k donde k ≥ 0.

La función de multiplicidad m(n_i , n_j) = k.

Ejemplo: m(1, 1) = 0, m(1, 2) = 1, m(1, 3) = 1, etc.



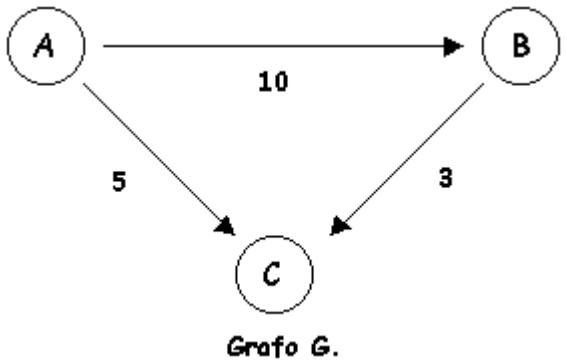
Grafo simple: Un digrafo D es simple si cada par ordenado (n_i n_j) ∈ N x N tiene como máximo de multiplicidad el valor 1.

Conceptos

Matriz de conexión: Todo grafo G tiene asociado una matriz M de dos dimensiones de orden |N| x |N|, cuyos elementos representan la multiplicidad del par (n_i , n_j).

Ejemplo: La matriz de conexión del digrafo G es

$$M = \begin{matrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{matrix}$$



Grafo G.

Conceptos digrafos

Camino: Un camino C de un **digrafo D** es una secuencia de arcos (a_1, a_2, ..., a_n) tal que para todo par (a_i, a_j) de arcos consecutivos en C, **el fin de a_i coincide con el inicio de a_j.**

Camino simple: Es el camino que no recorre el mismo arco dos veces.

Camino elemental: Es el camino que no visita un mismo nodo más de una vez.

Conceptos digrafos

Circuito: Es un camino finito en el que **el origen del primer arco coincide con el fin del último**.

Circuito simple: Un circuito es simple si a su vez es un camino simple.

Orden de un camino: Es el número de arcos que conforman el camino.

Conceptos digrafos

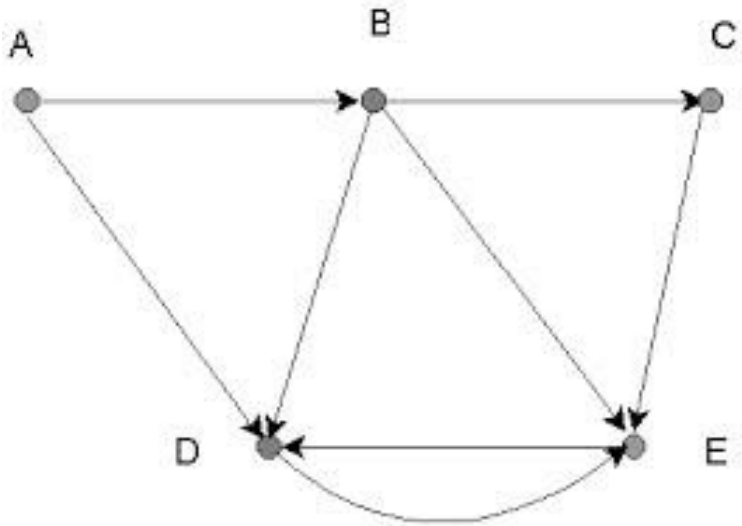
Ejemplos de caminos

(A, B, D, E)

(A, B, C, E), …

Ejemplos de circuitos

(D, E, D)



Conceptos

Nodo accesible: Un nodo n_i es accesible desde otro nodo n_j, si

- $n_i = n_j$ o
- existe un camino desde n_i hasta n_j.

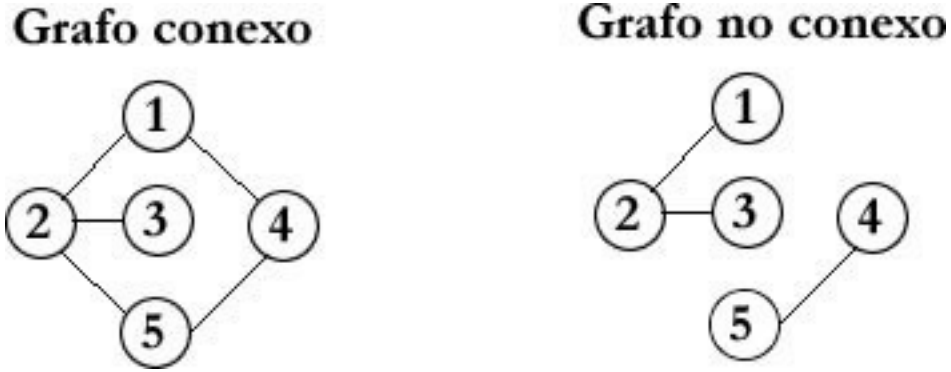
Nodo sucesor: Un nodo n_j es sucesor de un nodo n_i si existe un camino que va desde n_i hasta n_j.

Nodo predecesor: Un nodo n_i es predecesor de otro nodo n_j si existe un camino que va desde n_i hasta n_j.

Conceptos

Grafo conexo: Un grafo G es conexo si y sólo si **sus nodos no pueden ser particionados** en dos conjuntos no vacíos N1 y N2, tal que todas sus **trayectoria (**caminos/cadenas) desde un punto cualquiera a otro estén en el mismo conjunto.

Un grafo es conexo si, **para cualquier par de nodos *a* y *b* en *G*, existe al menos una trayectoria** (una sucesión de vértices adyacentes que no repita vértices) de *a* a *b*

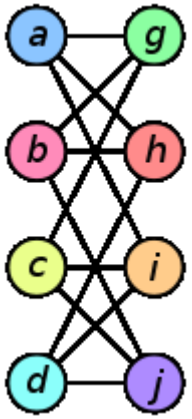


Conceptos

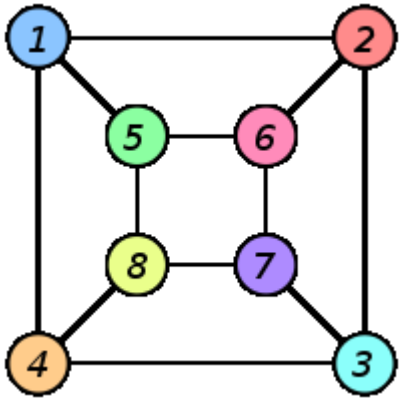
Subgrafo: Un subgrafo SG de G es un grafo constituido por un subconjunto SN de N y un subconjunto SA de A que conectan los nodos de SN.

Isomorfismo: Dos grafos G y G' son isomórficos si existe una biyección f: N→N' tal que $(n_i , n_j) \in A \Leftrightarrow (f(n_i), f(n_j)) \in A'$.

G



G'



Isoformismo entre G y G'

F(a)=1
F(b)=6
F(c)= 8
F(d)=3
F(g)=5
F(h)= 2
…

Vecino: En un grafo dirigido G, el vecino de un nodo n es cualquier nodo adyacente a n en la versión no dirigida de G.

Conceptos grafo no dirigido

Cadena: Para un grafo no dirigido GND es la secuencia de líneas (l_1 , l_2, ..., l_n) tal que **el fin l_i coincide con el origen de l_{i+1}.**

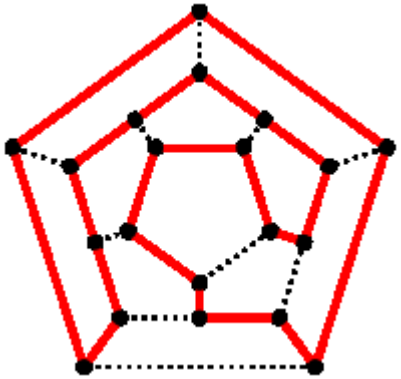
Cadena simple: Es la cadena donde ninguna línea se repite.

Cadena elemental: Es la cadena donde ningún nodo se repite.

Conceptos grafo no dirigido

Ciclo: Es una cadena finita donde **el nodo inicial de la cadena coincide con el nodo terminal de la misma**.

Ciclo simple: Es el ciclo que a su vez es una cadena simple.



Ciclo hamiltoniano.

Conceptos

Grafo múltiple: Es un grafo que contiene alguna arista paralela.

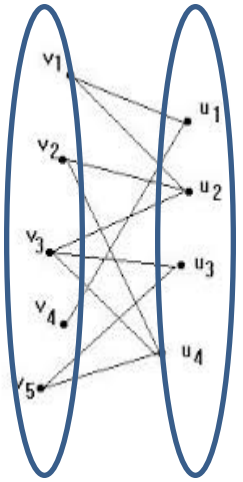
Digrafo acíclico: Es un digrafo que no contiene circuitos. Se le conoce con las siglas **dag**.

Grafo o digrafo con peso: Es un grafo o digrafo que tiene un valor entero o real asignado a cada arista.

Grafo completo: Es un grafo no dirigido donde cada par de nodos es adyacente

Conceptos

Grafo bipartito: Es un grafo no dirigido que puede ser dividido en dos subgrafos sin conexión entre ellos. Es un grafo $G=(N,E)$ cuyos vértices se pueden separar en dos conjuntos disjuntos U y V,

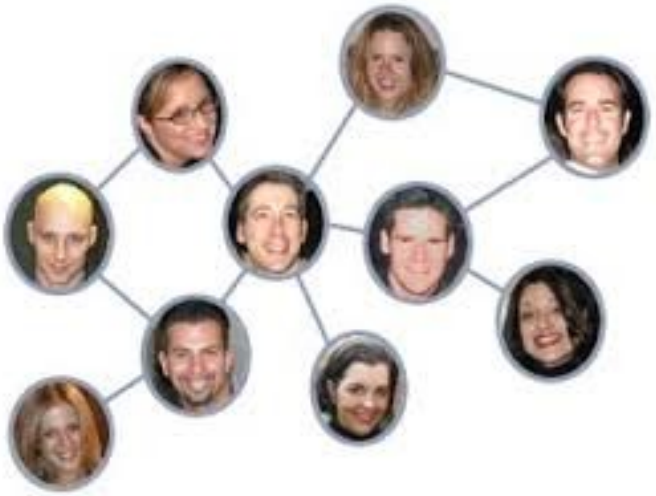


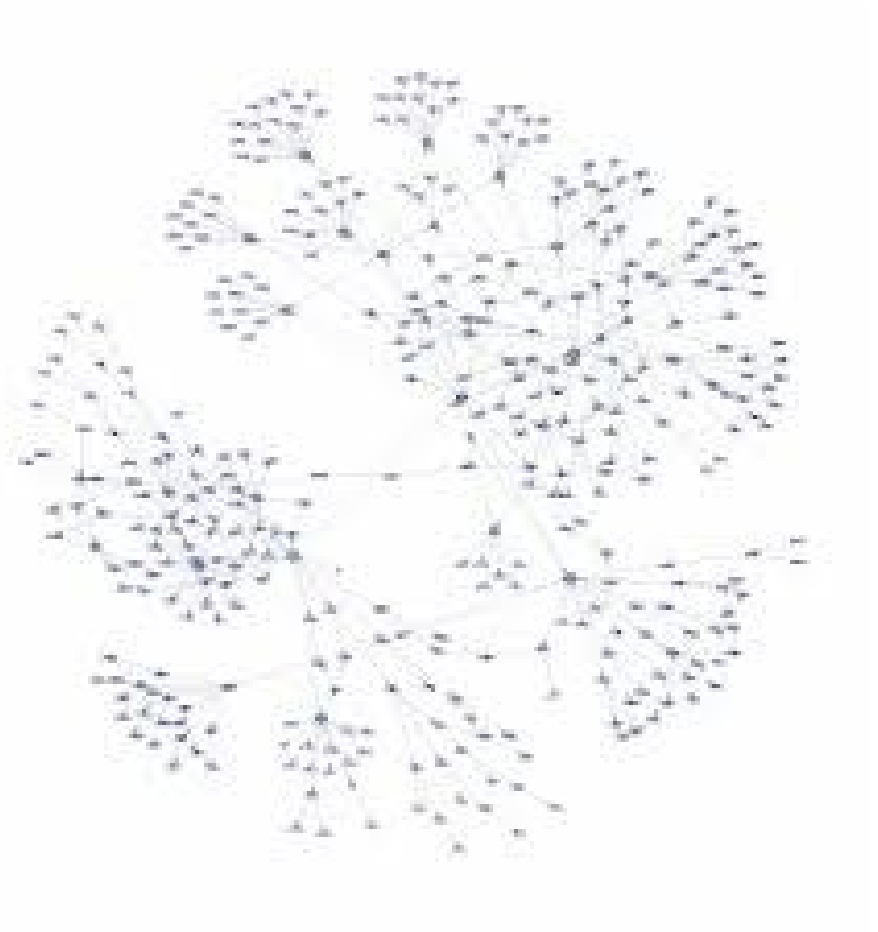
Multigrafo: Es un grafo no dirigido que puede tener varias aristas y lazos entre sus nodos.

Hipergrafo: Es un grafo no dirigido con hiper-aristas que conectan varios nodos (muchas).

Ejemplo: hipertexto, hipermedia, redes sociales.

Conceptos





Conceptos

Grafo fuertemente conexo: Un digrafo G con caminos para todo par fe vertivcecs en ambos sentidos

Grafos no dirigidos particulares

- **Euleriano** admite un circuito que contiene todas sus aristas
- **Hamiltoneado:** tiene un camino cerrado elemental que contiene todos los vertices
- **Completo**

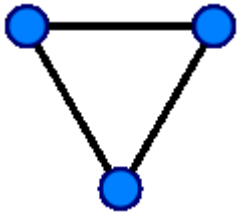
vértice de corte o punto de articulación

es un vértice de un grafo tal que si el grafo estaba conectado antes de retirar el vértice, entonces pasará a desconectarse.

A pesar de que estén bien definidos para grafos dirigidos, los vértices de corte se usan principalmente en los grafos no dirigidos.

Una arista de corte o puente, es una arista análoga a un vértice de corte;

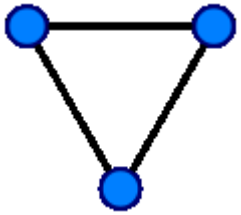
3 vertices de corte



No hay
vertices de corte

Grafo biconexo y bicoherente

Un grafo G es biconexo si es conexo y no tiene puntos de articulación



Ejemplo: si G representa una red de telecomunicaciones, entonces G asegura que la red puede seguir funcionando aunque falle uno de los equipos de los nodos.

Un grafo G es bicoherente si todo punto de articulación está unido mediante al menos 2 aristas con cada componente del subgrafo restante

Si además de biconexo, G es bicoherente, se tiene la seguridad que la red seguirá funcionando aunque falle una línea de transmisión

Tipo de Dato Abstracto del Grafo

Un grafo se especifica en base a sus funciones básicas, entre las que se pueden mencionar:

- **crearlo,**
- **insertar un nodo nuevo o una arista nueva,**
- **conocer si está vacío o no,**
- **consultar si un nodo existe,**
- **Verificar si tiene un circuito o ciclo**
- **etc.**

TDA del Grafo

Existen dos métodos para almacenar un grafo G:

- Matriz de adyacencia: se implementa utilizando el método secuencial o con arreglos y
- Listas de adyacencia implementada utilizando un método enlazado o listas enlazadas

Matriz de adyacencia

Es un arreglo de dos dimensiones que representa las conexiones entre pares de nodos o vértices.

Sea un grafo G con un conjunto de nodos N y un conjunto de aristas A. Suponga que el grafo es de orden n (número de nodos del grafo), donde $n > 0$.

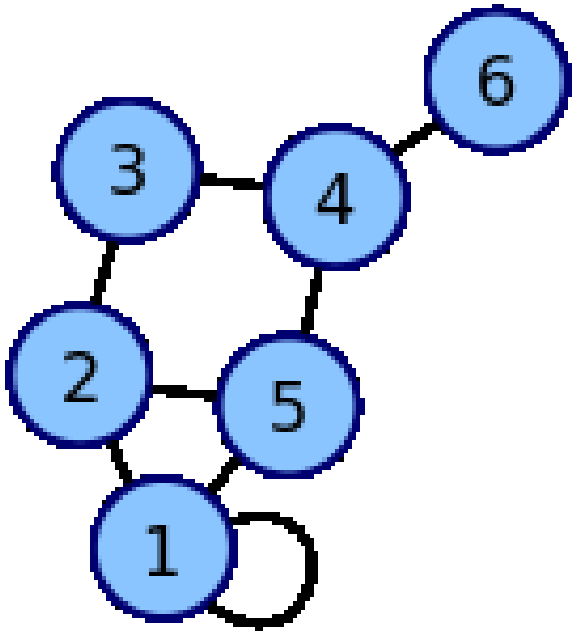
- La matriz de adyacencia se representa por un arreglo de tamaño n x n, donde:

$$M(i, j) = \begin{cases} 1 & \text{si existe un arco}(n_i\ n_j) \text{ en A,} \\ 0 & \text{en caso contrario} \end{cases}$$

- Las columnas y las filas de la matriz representan los nodos del grafo.
- Si el grafo es no dirigido, la **matriz es simétrica**

$$M(i,j) = M(j,i).$$

Matriz de adyacencia



$$\begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

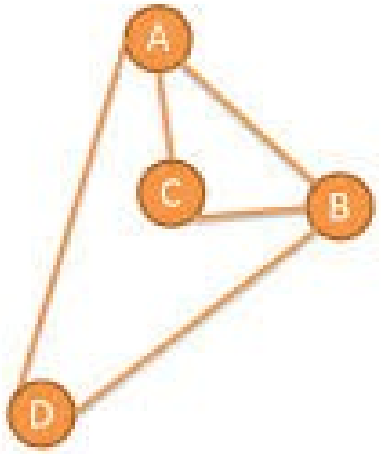
Listas de adyacencia

El segundo método utilizado para representar grafos es útil cuando un grafo posee muchos nodos y pocas aristas (grafos no densos).

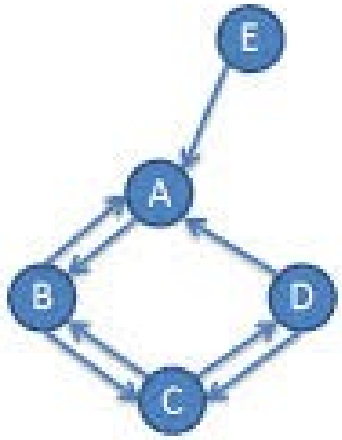
- Se utiliza una lista enlazada, llamada directorio, que almacena los nodos N del grafo, y asociado a cada nodo del grafo se encuentra una **lista enlazada con los nodos que son adyacentes a él.**
- En las listas están las aristas del grafo.
  - Un **grafo no dirigido** de orden N con A aristas requiere N entradas en el directorio y 2*A entradas de listas enlazadas, excepto si existen bucles, que reduce el número de entradas a la lista en 1
  - Un **grafo dirigido** de orden N con A arcos requiere N entradas en el directorio y A entradas de listas enlazadas

Listas de adyacencia

Grafo No Dirigido



Grafo Dirigido



A	->	B	->	C	->	D
B	->	D	->	C	->	A
C	->	A	->	B		
D	->	B	->	A		

A	->	B		
B	->	A	->	C
C	->	D	->	B
D	->	C	->	A
E	->	A		

UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

tipo de datos abstractos (TDAs)

20/11/07	Especificación Grafo[TipoEle]	
1	**Sintáctica** creaGrafo()→Grafo, nuevoNodo(Grafo,TipoEle) →Grafo, nuevaArista(Grafo,TipoEle,TipoEle) →Grafo, eliNodo(Grafo,TipoEle) →Grafo, eliArista(Grafo,TipoEle,TipoEle) →Grafo, conNodo(Grafo,TipoEle) →Lógico, conArista(Grafo,TipoEle,TipoEle) →Lógico, nodoAdyacente(Grafo,TipoEle) →Lista[TipoEle], vacíoGrafo(Grafo) →Lógico>	-creaGrafo(): Crea un grafo vacío. -nuevoNodo(): Ingresa un nuevo nodo al grafo -nuevaArista(): Ingresa una nueva arista entre los dos nodos especificados, si existen. -eliNodo(): Elimina un nodo del grafo. -eliArista(): Elimina una arista del grafo entre los nodos especificados. -conNodo(): Regresa verdadero sí existe en el grafo. -conArista(): Regresa verdadero si el arco existe. -nodoAdyacente(): Regresa la lista de los arcos adyacentes a él. -vacíoGrafo(): Regresa verdadero si es el grafo esta vacío -destruyeGrafo(): Destruye el grafo
2	**Declaraciones** TipoEle: no,{TipoNoDef}	
3	**Semántica** vacíoGrafo(creaGrafo())=Verdadero vacíoGrafo(nuevoNodo(creaGrafo(),no)=Falso conNodo(creaGrafo(), no)=Falso conNodo(nuevoNodo(creaGrafo(),no),no)=Verdadero conArista(creaGrafo(),no,no)=Falso conArista(nuevaArista(creaGrafo(),no,no),no,no)=Cierto eliNodo(creaGrafo(),no)=creaGrafo()	

UNIVERSIDAD DE LOS ANDES
MERIDA VENEZUELA

Versión 1.0

Grafo[TipoEle]

Clases: Arreglo de Entero, Entero, Tabla, TipoEleTab, Salida[TipoEle], TipoEle}

1	Estructura: privado:	-Grafo(). *Constructor. Crea un grafo vacío*
2	Operaciones: público:	-nuevoNodo(). *Transformador. Inserta un nuevo nodo al grafo.*
3	Grafo()	-nuevaArista(). *Transformador. Ingresa un nuevo arco al grafo.*
4	nuevoNodo(TipoEle: n)	-eliNodo(). *Transformador. Elimina un nodo del grafo*
5	nuevaArista(TipoEle: ni, TipoEle: nf):	-eliArista(). *Transformador. Elimina un arco del grafo.*
6	eliNodo(TipoEle: n)	-conNodo(). *Observador. Regresa verdadero, si existe.*
	eliArista(TipoEle: ni, TipoEle: nj)	-conArista(). *Observador. Regresa verdadero si existe el arco.*
	conNodo(TipoEle: n): Lógico	-nodoAdyacente(). *Observador. Regresa el arreglo de nodos* adyacentes y su tamaño.
	conArista(TipoEle: ni, TipoEle: nf):Lógico	-recorridoEnAmp(). *Observador. Recorre el grafo en amplitud.*
	nodoAdyacente(TipoEle: n, Entero: t): Arreglo[100] de TipoEle	-recorridoEnProf(). *Observador. Recorre el grafo en* profundidad
	recorridoEnAmp()	-rutaOptima(). *Observador. Regresa un arreglo con los nodos* que conforman el camino de menos arcos a partir de un nodo inicial
	recorridoEnProf()	-vacioGrafo(). *Observador. Regresa verdadero si el grafo está* vacío.
	rutaOptima(TipoEle: nj): Arreglo[100] de Salida[TipoEle]	-numNodos(), numAristas(). *Observadores. Regresa el número* de nodos y el de aristas del grafo, respectivamente.
	vacíoGrafo():Lógico	-=(). *Transformador. Asigna un grafo.*
	numNodos():Entero	~Grafo(). *Destructor. Destructor del grafo*
	numAristas():Entero	
	=(Grafo[TipoEle])	
	~Grafo()	

Listas de adyacencia

```
#include <vector>
#include <list>
vector< list<int> > grafo(MAX_VERT);
char visitado[MAX_VERT];
void inserta_arista(int i, int j){
    grafo[i].push_back(j);
    grafo[j].push_back(i);
}
void limpia_grafo(){
int i;
    for(i = 0; i < nvert; i++){
            grafo[i].clear();
            visitado[i] = 0;
    }
}
list<int>::iterator aux, fin;
aux = grafo[j].begin();
fin = grafo[j].end();
while(aux != fin){
    if(!visitado[*aux]){
            pila.push(*aux);
            visitado[*aux] = 1;
    }
    aux++;
```

Tarea de grafo

Tome una situación de su cotidiana (red de amigos, rutas para ir a su casa, recorrido en un viaje realizado, etc.) y modélela usando grafos.

Además, determine en el grafo los conceptos vistos en clase e interprételos (que podría significar el grado, para que sirve determinar que sea conexo, para que le sirve determinar una trayectoria, etc.