



GRAFO / DIGRAFO

ESTRUCTURAS DE DATOS
y ALGORITMOS
LCC – LSI - TUPW

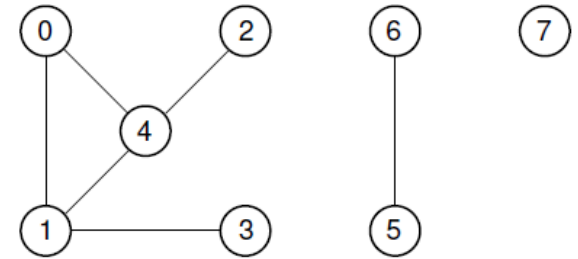
Objetivos

Conocer y evaluar los algoritmos tradicionales de manipulación de Grafo/Digrafo.

Evaluar las distintas alternativas de representación de Grafo/Digrafo.

Construir y usar, en problemáticas particulares, los TAD Grafo / TAD Digrafo.

GRAFO



Un **grafo no dirigido** $G=(V,E)$ se compone de dos conjuntos finitos:

- el conjunto $V=\{v_1, v_2, \dots\}$, que es el conjunto de **nodos** de G y
- el conjunto de **aristas** $E=\{e_1, e_2, \dots\}$ que es un conjunto de pares **no ordenados** de nodos diferentes de G .

Cada elemento del conjunto E , $e=(u,v)$ se conoce como una **arista** y se dice que une los nodos u y v .

Si $e=(u,v)$ es una arista de G , entonces los nodos u y v son **adyacentes**. Dado que estamos tratando con pares no ordenados, (u,v) y (v,u) representan la misma arista.

GRAFO

- El **grado** de un nodo en un grafo no dirigido está dado por el número de aristas en las que participa dicho nodo.
- Un **camino P de longitud n** , desde un nodo u a un nodo v , se define como la secuencia de $n+1$ nodos : $P(u,v) = (v_0, v_1, \dots, v_n)$ donde: $u=v_0$, $v=v_n$, y v_i es adyacente a v_{i-1} , $1 \leq i \leq n$.
- Un **camino P es cerrado** si $v_0=v_n$.
- Un **camino P es simple** si todos los nodos son distintos, a excepción de v_0 que puede ser igual a v_n .
- Un **ciclo** es un camino simple cerrado de longitud 3 o mas. Un ciclo de longitud k se llama k -ciclo.
- Un **grafo acíclico**, también llamado **bosque**, es aquel grafo que no contiene ningún ciclo.

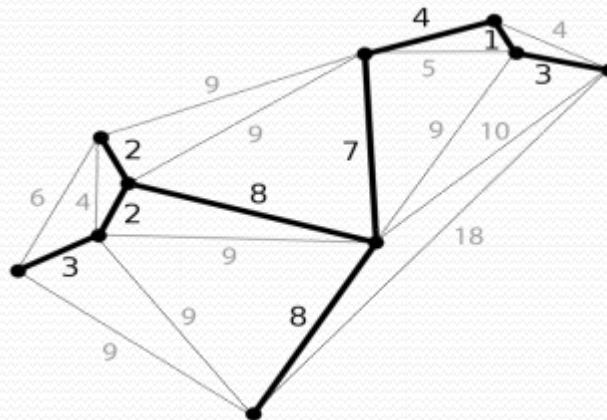
GRAFO

- Si hay un camino desde el nodo u al nodo v , entonces diremos que v es **accesible** desde u .
- Un **grafo** G es **conexo**, si al menos existe un camino entre cada par de nodos del grafo.
- **Árbol** es un grafo acíclico conexo.
- Un grafo G está **etiquetado**, si sus aristas tienen datos asignados.
- Un grafo G tiene peso- **grafo valorado** o **ponderado** o **con peso**-, si cada arista e de G tiene asignado un valor numérico, $w(e)$, llamado peso o longitud de e .
$$w: \{(u,v) \in E\} \rightarrow \mathbb{R}^+ \cup \{0\}$$
- En los grafos ponderados el peso de un camino P es la suma de los pesos de las aristas que unen los nodos que forman el camino. El camino de menor peso, entre dos nodos, se denomina **camino mínimo**. Si el grafo no es valorado, el camino mínimo entre dos nodos es aquel que contiene el menor número de aristas.

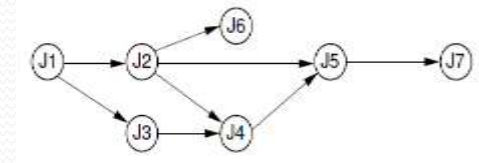
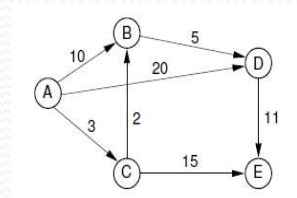
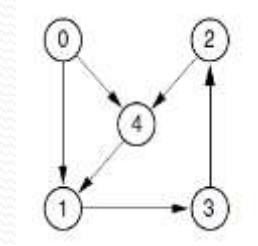
GRAFO

En los grafos ponderados el peso de un camino P es la suma de los pesos de las aristas que unen los nodos que forman el camino. El camino de menor peso, entre dos nodos, se denomina **camino mínimo**. Si el grafo no es valorado, el camino mínimo entre dos nodos es aquel que contiene el menor número de aristas.

Árbol de recubrimiento de un grafo G : subgrafo sin ciclos que contiene a todos los vértices de G . En caso de haber varios árboles de coste mínimo se elige el que posee menos arcos.



DIGRAFO



Un **grafo dirigido** o **digrafo**, $G=(V,E)$, es tal que cada arista e de G tiene una dirección asignada; es decir, cada arista e está identificada por un **par ordenado** (u,v) de nodos de G .

Consideremos el arco $e=(u,v)$ del digrafo G , entonces

- e empieza en u y termina en v .
- u es el origen o punto inicial de e , y v es el destino o punto terminal de e .
- u es un predecesor de v y v es sucesor de u .

DIGRAFO

- El **grado de salida** de un nodo u , $\text{grad_sal}(u)$, es el número de aristas que empiezan en u .
- El **grado de entrada** de un nodo u , $\text{grad_ent}(u)$, es el número de aristas que terminan en u .
- Un nodo u es **nodo fuente**, si $\text{grad_sal}(u) > 0$ y $\text{grad_ent}(u) = 0$.
- Un nodo u es **nodo sumidero**, si $\text{grad_ent}(u) > 0$ y $\text{grad_sal}(u) = 0$.
- **Camino** (trayectoria) entre los nodos u y v , es una secuencia de nodos $P(u, v) = (v_0, v_1, \dots, v_n)$, $u = v_0$ y $v = v_n$ tal que $(v_0, v_1) \in E$, $(v_1, v_2) \in E, \dots, (v_{n-1}, v_n) \in E$.
- Un **digrafo G es fuertemente conexo**, si para cada par de nodos $u, v \in V$, existe un camino de u a v **y** un camino de v a u .
- Un **dígrafo G es simple conexo**, si para cada par de nodos $u, v \in V$, existe un camino de u a v **o** un camino de v a u .

GRAFO/DIGRAFO

Aplicaciones

Red de comunicaciones -computadoras, aeropuertos, ciudades, terminales, depósitos u otras entidades que pretendan comunicarse- bidireccional o no.

Capacidades conocidas -ancho de banda, distancias, tiempos-, de los canales de comunicación, correspondientes a una red de comunicaciones .

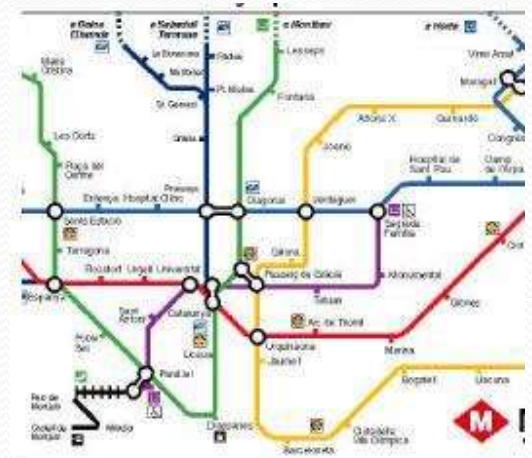
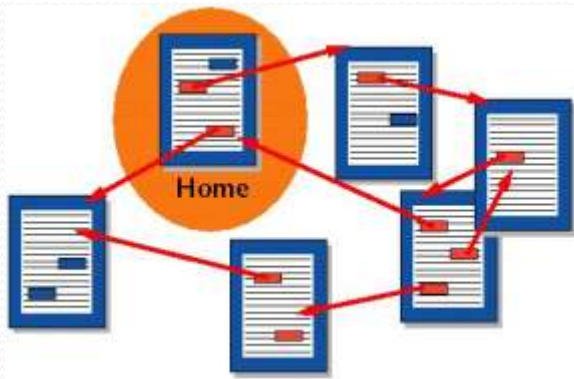
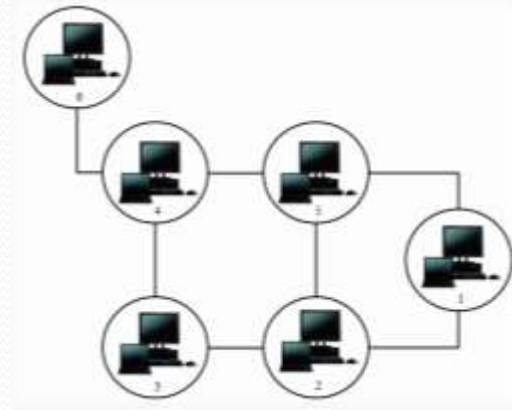
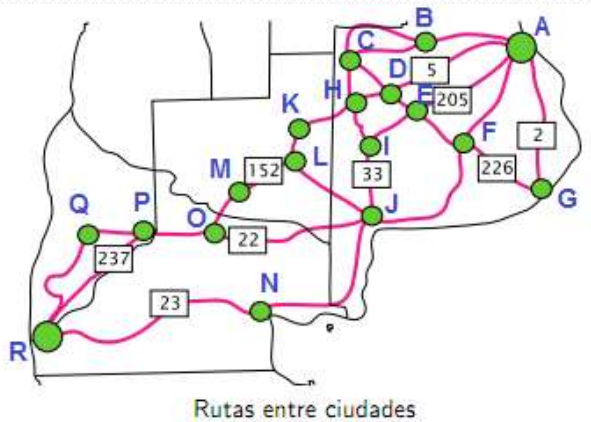
Hipertextos o multimedias.

Relaciones de precedencia que pueden existir entre las tareas que deben realizarse para completar un trabajo.

[Ver Video Teoria de Grafos-Adrian Paenza.mp4](#)

GRAFO/DIGRAFO

Aplicaciones



GRAFO/DIGRAFO

Requerimientos funcionales

- ¿Es posible desplazarse de un sitio a cualquier otro sitio?
- ¿Es posible desplazarse de un sitio a otro sitio particular?
- ¿Es posible transmitir información desde una computadora a cualquier otra?
- ¿Cuál es el camino con el menor número de conexiones?
- ¿Cuál es el camino mas corto?
- ¿Cuál sería una secuencia posible de realización de tareas?

...

T.A.D. GRAFO/DIGRAFO – Especificación

Operaciones Abstractas

Sean G : Grafo/Digrafo y u, v : nodos

<i>NOMBRE</i>	<i>ENCABEZADO</i>	<i>FUNCION</i>	<i>ENTRADA</i>	<i>SALIDA</i>
Adyacentes	Adyacentes(G, u)	Determina los nodos adyacentes de u	G y u	Reporta los nodos adyacentes a u .
Camino	Camino(G, u, v)	Determina el camino de u a v	G, u y v	Reporta el camino de u a v , si v es alcanzable desde u ; Error en caso contrario
Conexo	Conexo(G) – (Simple o Fuerte)	Evalúa si G es conexo (Simple o Fuerte)	G	V si G es conexo (Simple o Fuerte), F en caso contrario
Acíclico	Acíclico(G)	Evalúa si G es acíclico	G	V si G es acíclico, F en caso contrario
REA	REA(G)	Procesa todos los elementos de G en anchura	G	Está sujeta al proceso que se realice sobre los elementos de G
REP	REP(G)	Procesa todos los elementos de G en profundidad	G	Está sujeta al proceso que se realice sobre los elementos de G

T.A.D. DIGRAFO – Especificación

Operaciones Abstractas

Sean G : Digrafo y u : nodo

<i>NOMBRE</i>	<i>ENCABEZADO</i>	<i>FUNCION</i>	<i>ENTRADA</i>	<i>SALIDA</i>
GradEnt	$\text{GradEnt}(G,u)$	Determina cantidad de aristas que llegan a u	G y u	Reporta el grado de entrada de u .
GradSal	$\text{GradSal}(G,u)$	Determina cantidad de aristas que salen de u	G y u	Reporta el grado de salida de u
NodoF	$\text{NodoF}(G,u)$	Evalúa si u es nodo fuente de G	G y u	V si u es nodo fuente de G
NodoS	$\text{NodoS}(G,u)$	Evalúa si u es nodo sumidero de G	G y u	V si u es nodo sumidero de G

T.A.D. GRAFO/DIGRAFO

Representación

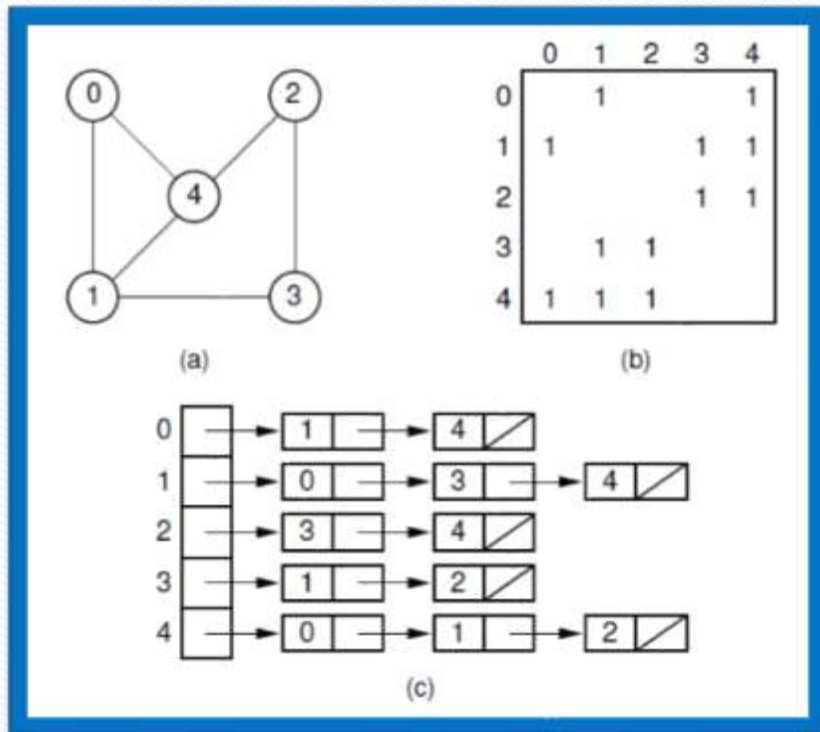


T.A.D. GRAFO/DIGRAFO

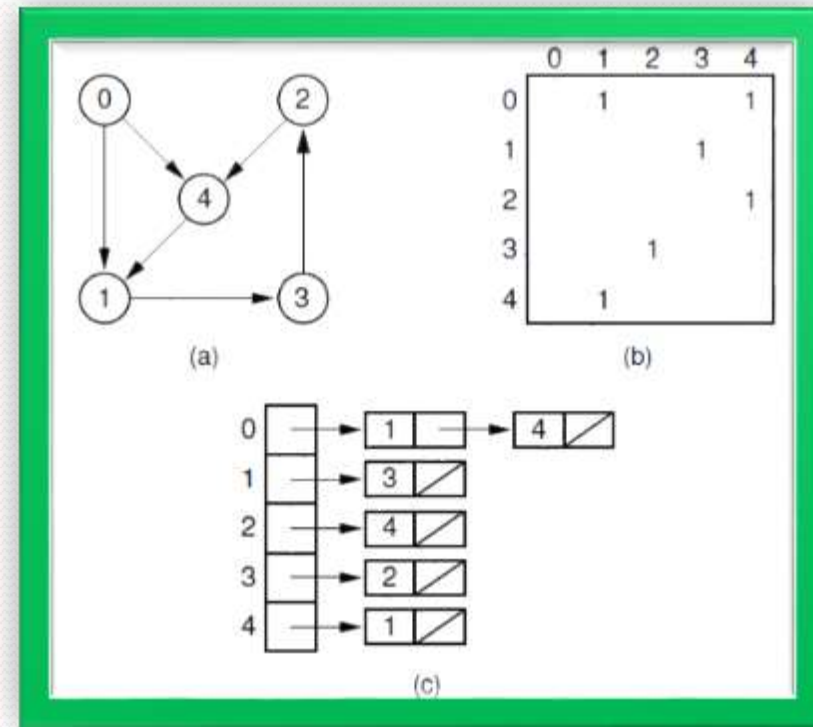
Representación

Requerimiento de memoria???

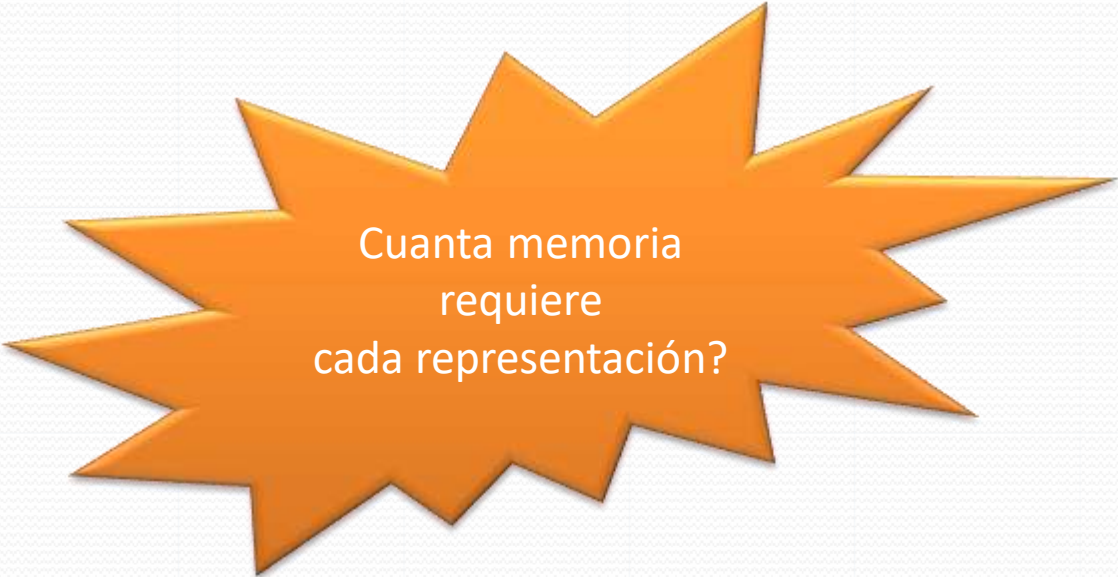
Digrafo



Grafo



T.A.D. GRAFO - DIGRAFO

A large, orange, multi-pointed starburst shape with a slight 3D effect, containing the text.

Cuanta memoria
requiere
cada representación?

T.A.D. GRAFO/DIGRAFO

Construcción de las Operaciones Abstractas (1)

Las mas importantes

Conexo : ¿es posible desplazarse de un sitio a cualquier otro sitio? ¿es posible transmitir información?.

Accesibilidad: ¿es posible desplazarse de un sitio a otro sitio particular?, $G=(V,E)$, nodo de origen s nodo destino d , $s,d \in V$, existe camino desde s hasta d ???

Caminos mínimos: camino con el menor número de aristas, o el camino con el menor peso total.

Ordenación Topológica: grafo dirigido acíclico planificación posible de tareas. orden lineal de V , tal que si $(u,v) \in E$, entonces u aparece antes que v .

T.A.D. GRAFO/DIGRAFO

Construcción de las Operaciones Abstractas (2)

Propiedades de un grafo $G \rightarrow$ analizar los nodos y las aristas

Dos criterios de análisis a partir de un nodo $s \in V$

Algoritmo **BEA**, todos los nodos adyacentes a s serán examinados primero, después serán examinados los nodos adyacentes a estos y así sucesivamente. Para ello el algoritmo hace uso de una cola.

Algoritmo **BEP**, hace uso de una pila, por lo que los nodos de V serán examinados desde s en profundidad.

Búsqueda en Anchura (**BEA**): Utiliza un arreglo $d[1..|V|]$, para almacenar los nodos marcados y los nodos no marcados de V . Al terminar el algoritmo, para cada nodo marcado $u \in V$, $d[u]$ contendrá el mínimo número de aristas de cualquier camino desde el origen s hasta u . Los nodos no marcados no son accesibles, (accesibilidad, camino mínimo con origen único para grafos no valorados, análisis de conexidad, etc).

T.A.D. GRAFO/DIGRAFO

Construcción de las Operaciones Abstractas (3)

Recorrido (búsqueda) en Anchura

REA (G, s)

s es el origen de G- Grafo/Digrafo

Para cada $v \in V$ hacer

$d[v] \leftarrow \infty$ todos los nodos están no marcados

$d[s] \leftarrow 0$ marcar el origen

Insertar (Cola , s)

Mientras no (Vacía (Cola)) hacer

 Suprimir (Cola , v)

 Para Cada u Adyacente a v hacer

 Si $d[u] = \infty$

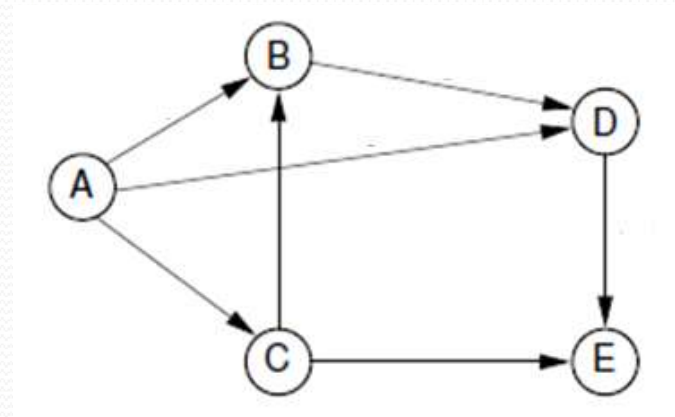
 entonces $d[u] \leftarrow d[v] + 1$

 Insertar (Cola, u)

 FinSi

 FinPara

FinMientras



marcar u

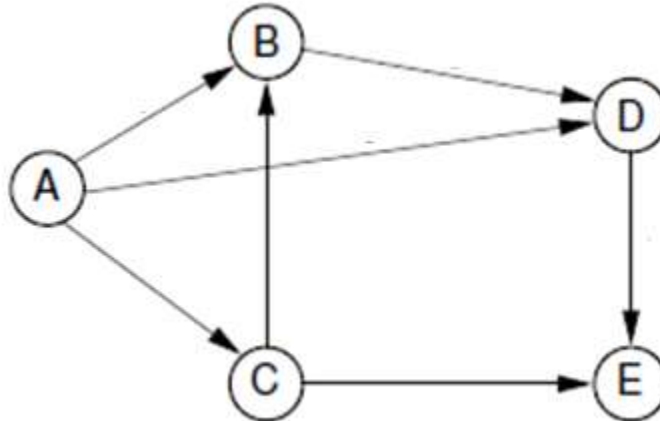
Tiempo de
ejecución???

T.A.D. GRAFO/DIGRAFO

Construcción de las Operaciones Abstractas (4)

Algoritmo REA

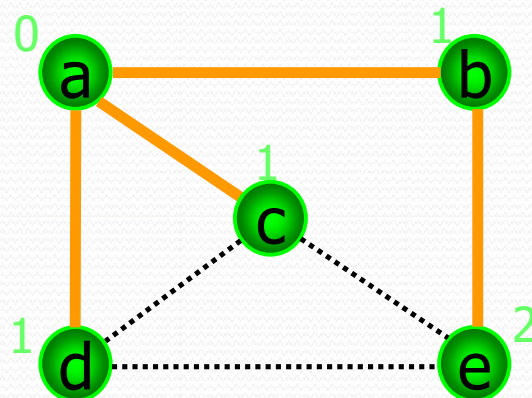
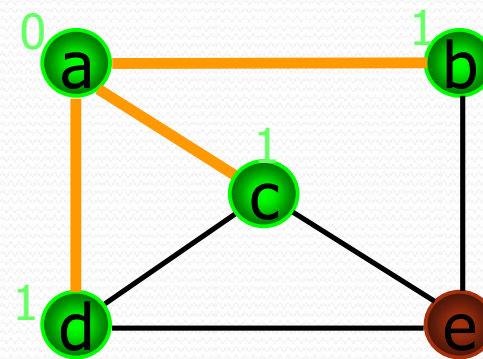
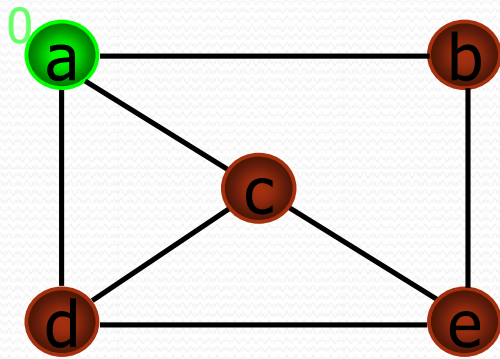
d	<table><tr><td>∞</td><td>∞</td><td>∞</td><td>∞</td><td>∞</td></tr><tr><td>A</td><td>B</td><td>C</td><td>D</td><td>E</td></tr></table>	∞	∞	∞	∞	∞	A	B	C	D	E
∞	∞	∞	∞	∞							
A	B	C	D	E							
Cola	<table><tr><td></td></tr></table>										



T.A.D. GRAFO/DIGRAFO

Búsqueda primero en anchura

Animación



T.A.D. GRAFO/DIGRAFO

Construcción de las Operaciones Abstractas (4)

Recorrido (búsqueda) en Profundidad

REP (G)

Para cada $v \in V$ hacer

$d[v] \leftarrow 0$

tiempo $\leftarrow 0$

Para cada $s \in V$ hacer

Si $d[s] = 0$

entonces

REP-Visita (G,s)

Tiempo de
ejecución???

REP- Visita (G, s)

tiempo \leftarrow tiempo + 1

$d[s] \leftarrow$ tiempo

Para Cada u Adyacente a s hacer

Si $d[u] = 0$

entonces

REP- Visita (G,u)

tiempo \leftarrow tiempo + 1

$f[s] \leftarrow$ tiempo

T.A.D. GRAFO/DIGRAFO

Construcción de las Operaciones Abstractas (5)

BEP propiedades de un grafo – grafo acíclico o no- a partir de la clasificación de sus aristas. (Heileman)

Aristas de árbol: cualquier colección de aristas de un grafo que formen un bosque-grafo acíclico. Todo nodo del grafo es un árbol con un único nodo con respecto a esta colección, o es parte de algún árbol mas grande por medio de su conexión a otro nodo vía una arista de árbol. Esta colección no es única.

Aristas hacia atrás: Dada una colección de aristas de árbol, las aristas hacia atrás de un grafo son aquellas aristas que conectan algún nodo descendiente de un árbol con un nodo antepasado del mismo árbol.

grafo no dirigido, BEP solo producirá aristas de árbol y aristas hacia atrás.

grafo no dirigido es **acíclico** si no tiene aristas hacia atrás (grafos dirigidos).

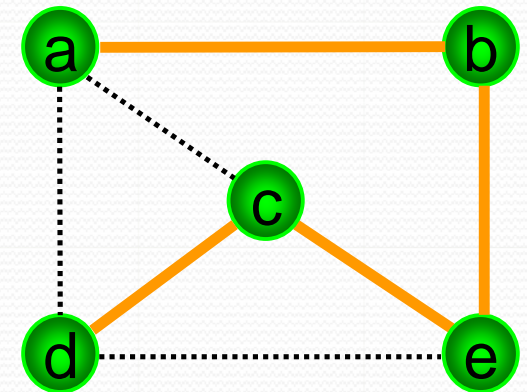
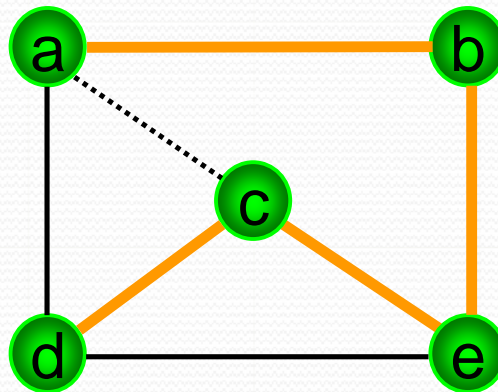
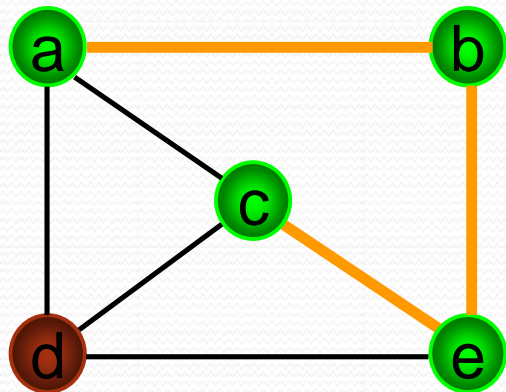
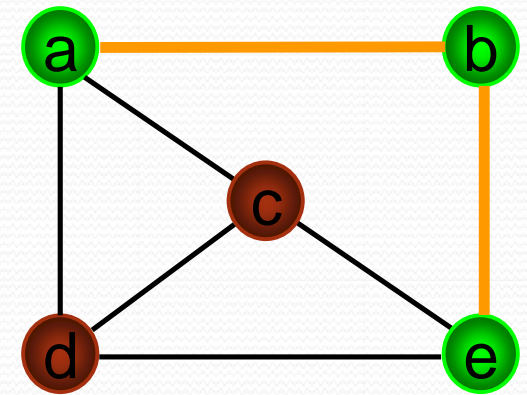
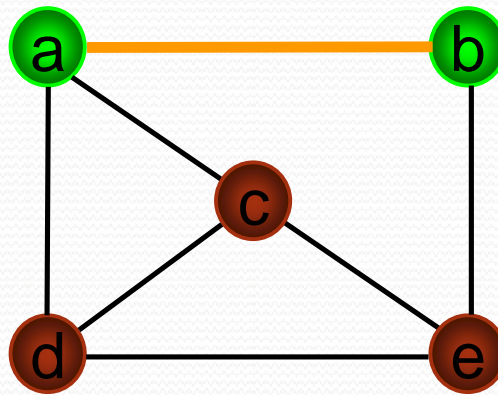
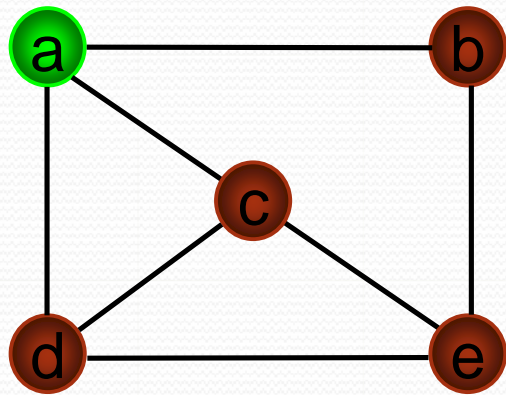
Reglas que propone Heileman para clasificar las aristas de un grafo usando BEP:

nodo descubierto encuentra un nodo no descubierto arista de árbol.

nodo descubierto encuentra nodo descubierto pero no terminado arista hacia atrás

T.A.D. GRAFO/DIGRAFO

Búsqueda primero en profundidad



T.A.D. GRAFO/DIGRAFO

Construcción de las Operaciones Abstractas (6)

Ordenación Topológica

El algoritmo **REP** puede ser utilizado para realizar una ordenación topológica sobre un **grafo dirigido acíclico** recibido como entrada.

Ordenación - Topológica (G)

Ejecutar **REP** (G) insertando nodos a la cabeza de la lista L conforme son terminados

Retornar L

L contiene la **Ordenación Topológica de G**

T.A.D. GRAFO/DIGRAFO

Construcción de las Operaciones Abstractas (7)

Ordenación Topológica

Ejemplo: las tareas de un proyecto de construcción.

Algoritmo: usar una versión modificada de REP

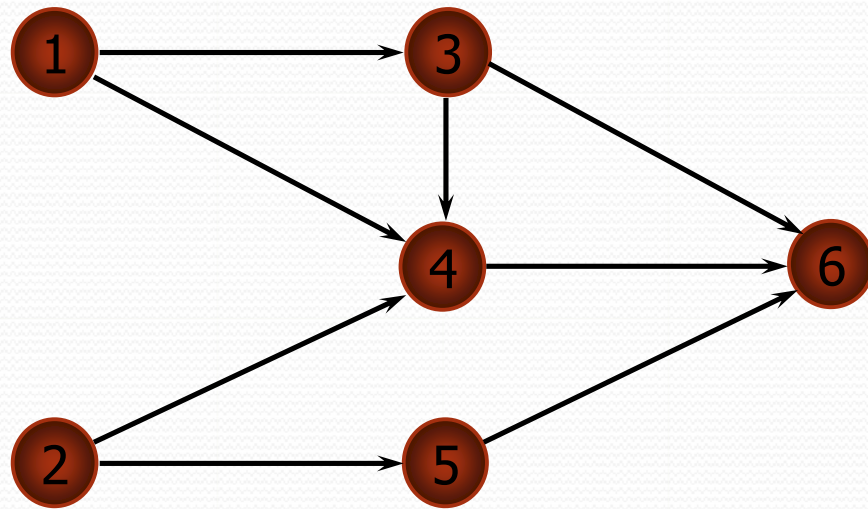
```
orden_topologico(v)      /* orden inverso */
{
    marca[v]=VISITADO;
    for cada vértice w en lista_adyacencia(v)
        if (marca[w]==SINVISITAR)
            orden_topologico(w);
    imprime(v);
}
```

Tiempo de
ejecución???

T.A.D. GRAFO/DIGRAFO

Orden topológico

Ejemplo

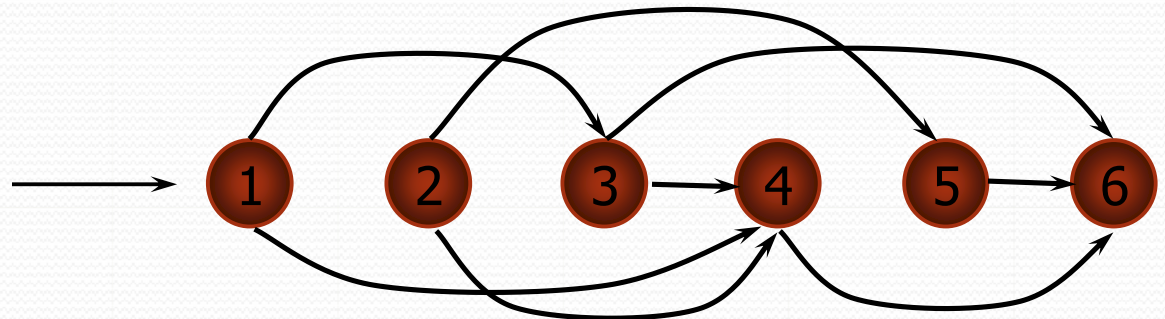


• *Orden topológico:*

1 2 3 4 5 6

1 3 2 4 5 6

2 1 5 3 4 6



T.A.D. GRAFO/DIGRAFO

Usos de los Recorridos

- **Ambos recorridos** se pueden usar para resolver los siguientes problemas:
 - Probar que G es conectado.
 - Obtener un árbol de expansión de G .
 - Obtener los componentes conectados de G .
 - Obtener un camino entre dos vértices dados de G , o indicar que no existe tal camino.
- El recorrido **REP** se usa para:
 - Obtener un ciclo en G , o indicar que G no tiene ciclos.
- El recorrido **REA** se usa para:
 - Obtener para cada vértice v de G , el número mínimo de aristas de cualquier camino entre s y v .

T.A.D. GRAFO/DIGRAFO

Construcción de las Operaciones Abstractas (8)

Dijkstra(T: Tabla);

Para i desde 1 hasta $|V|$ hacer

$v \leftarrow$ vertice con la distancia mas corta y desconocido

$T[v].conocido \leftarrow \text{True}$

 Para cada w adyacente a v hacer

 Si $T[w].conocido = \text{False}$

 entonces

 Si $T[v].distancia + w(v,w) < T[w].distancia$

 entonces

 Reducir ($T[w].distancia$ a $T[v].distancia + w(v,w)$)

$T[w].camino \leftarrow v$

 FinSi

 FinSi

FinPara

FinPara

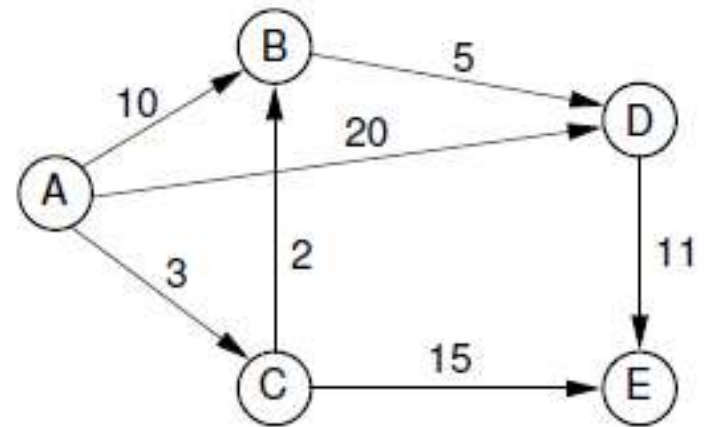
Tiempo de
ejecución???

T.A.D. GRAFO/DIGRAFO

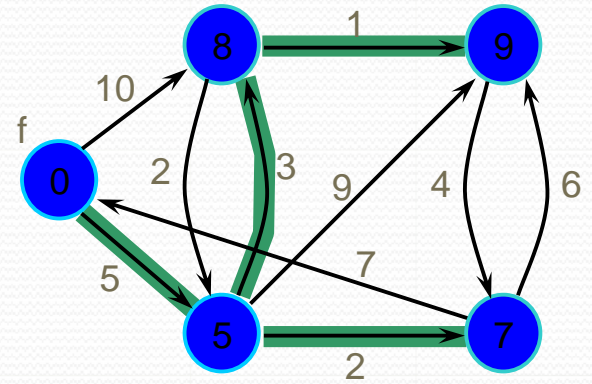
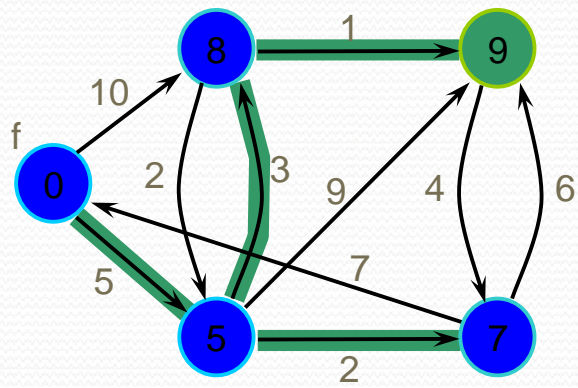
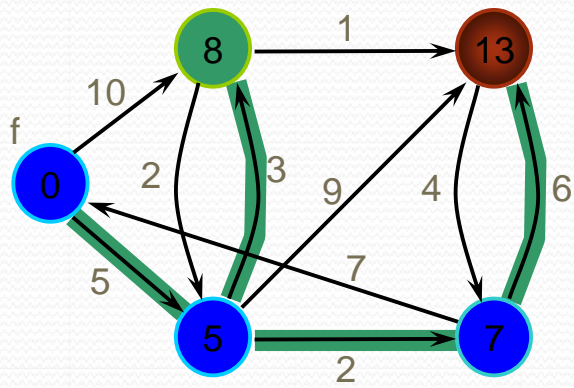
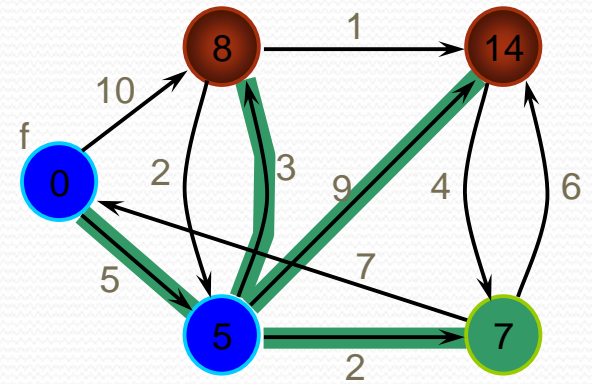
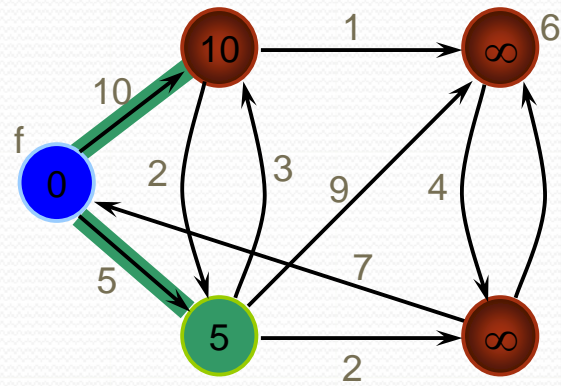
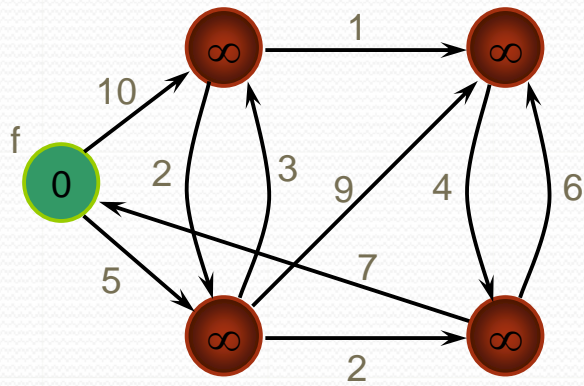
Construcción de las Operaciones Abstractas (9)

Algoritmo de Dijkstra

Vértices	Conocido	Distancia	Camino
A	F	0	
B	F	∞	
C	F	∞	
D	F	∞	
E	F	∞	



Algoritmo de Dijkstra



T.A.D. GRAFO/DIGRAFO

Construcción de las Operaciones Abstractas (10)

Algoritmo de Prim

El algoritmo de Prim encuentra un árbol de recubrimiento mínimo en un grafo conexo, no dirigido y cuyas aristas están ponderadas. En otras palabras, el algoritmo encuentra un subconjunto de aristas que forman un árbol con todos los vértices, donde el peso total de todas las aristas en el árbol es el mínimo posible. Si el grafo no es conexo, entonces el algoritmo encontrará el árbol de recubrimiento mínimo para uno de los componentes conexos que forman dicho grafo no conexo

T.A.D. GRAFO/DIGRAFO

Construcción de las Operaciones Abstractas (11)

Prim(T: Tabla);

Para i desde 1 hasta $|V|$ hacer

$v \leftarrow$ vertice con la distancia mas corta y desconocido

$T[v].conocido \leftarrow \text{True}$

Para cada w adyacente a v hacer

Si $T[w].conocido = \text{False}$

entonces

Si $w(v,w) < T[w].distancia$

entonces

$T[w].distancia \leftarrow w(v,w)$

$T[w].camino \leftarrow v$

FinSi

FinSi

FinPara

FinPara

T.A.D. GRAFO/DIGRAFO

Construcción de las Operaciones Abstractas (12)

Algoritmos de Warshall y de Floyd

El algoritmo de Warshall, es para caminos en grafos dirigidos y el algoritmo de Floyd es para caminos mínimos en grafos dirigidos ponderados, representados en matrices de adyacencia o de peso .

T.A.D. GRAFO/DIGRAFO

Construcción de las Operaciones Abstractas (13)

Algoritmo de Warshall, para encontrar la matriz de caminos P a partir de la matriz de adyacencia A de G.

$$P_k[i, j] = \begin{cases} 1 & \text{Si existe un camino simple de } v_i \text{ a } v_j \text{ que no} \\ & \text{usa otros nodos aparte de } v_1, v_2, \dots, v_k \\ 0 & \text{En otro caso} \end{cases}$$

WARSHALL (A, P)

P=A

Para 1 ≤ k ≤ n hacer

Para 1 ≤ i ≤ n hacer

Para 1 ≤ j ≤ n hacer

$P[i, j] = P[i, j] \vee (P[i, k] \wedge P[k, j])$

FinPara

FinPara

FinPara

Tiempo de
ejecución???

T.A.D. GRAFO/DIGRAFO

Construcción de las Operaciones Abstractas (14)

Algoritmo de Floy

grafo con peso  matriz de pesos $W = w_{ij}$

$$w_{ij} = \begin{cases} w(e) & \text{si hay una arista } e \text{ de } v_i \text{ a } v_j \\ 0 & \text{si no hay arista de } v_i \text{ a } v_j \end{cases}$$

matriz de pesos  matriz de caminos mínimos Q

donde q_{ij} contiene la longitud del camino mínimo de v_i a v_j .

las matrices P_0, P_1, \dots, P_k Q_0, Q_1, \dots, Q_k donde

$q_k[i, j] = \underset{\text{ó}}{\text{menor de las long de los anteriores caminos de } v_i \text{ a } v_j}$

suma de las longitudes de los anteriores caminos v_i a v_k y v_k a v_j

T.A.D. GRAFO/DIGRAFO

Construcción de las Operaciones Abstractas (15)

Es decir: $q_k [i, j] = \text{Mínimo} (q_{k-1} [i, j], q_{k-1} [i, k] + q_{k-1} [k, j])$

Tiempo de
ejecución???

CAMINO_MINIMOFloy (W, Q)

Q = W

Para $1 \leq k \leq n$ hacer

Para $1 \leq i \leq n$ hacer

Para $1 \leq j \leq n$ hacer

Q[i , j] = **Mínimo** (Q [i , j] , Q [i , k] + Q [k , j])

T.A.D. GRAFO/DIGRAFO

Construcción de las Operaciones Abstractas (16)

Matriz Potencia: matriz de adyacencia A  matrices potencia $A^2, A^3, A^4, \dots, A^k$. $a_k(i, j)$ - entrada i, j matriz A^k – números de **caminos de longitud k desde v_i a v_j** .

Ejemplo En una red de transporte hay 5 almacenes que suministran mercancías. Los almacenes 1, 2 y 4 están relacionados por una autopista que une al almacén 1 con el 2 y al 2 con el 4. El almacén 3 está unido con el 5 por una carretera local.

Representa en forma de grafo y de matriz, y comprueba mediante matrices que no existe comunicación total entre ellos.

Matriz de adyacencia: $M = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$

Las sucesivas potencias de M son: $M^2 = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$ $M^3 = \begin{pmatrix} 0 & 2 & 0 & 0 & 0 \\ 2 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{pmatrix}$ $M^4 = \begin{pmatrix} 2 & 0 & 0 & 2 & 0 \\ 0 & 4 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 2 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$

y su suma $B = M + M^2 + M^3 + M^4 = \begin{pmatrix} 3 & 3 & 0 & 3 & 0 \\ 3 & 6 & 0 & 3 & 0 \\ 0 & 0 & 2 & 0 & 2 \\ 3 & 3 & 0 & 3 & 0 \\ 0 & 0 & 2 & 1 & 1 \end{pmatrix}$

Como no todos los elementos de B son distintos de cero, los almacenes no están todos conectados entre ellos (como se ve fácilmente si dibujáis el grafo), esto es, el grafo no es conexo.

Tiempo de ejecución???