

# Algoritmos y estructuras de datos

## Grafos

CEIS

Escuela Colombiana de Ingeniería

2024-1

# Agenda

## ① Grafos

Conceptos

Representaciones

Recorridos

## ② Aspectos finales

Ejercicios

# Agenda

## ① Grafos

Conceptos

Representaciones

Recorridos

## ② Aspectos finales

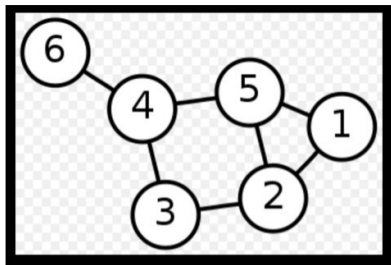
Ejercicios

## Definición

Es una pareja ordenada  $G = (V, E)$  donde:

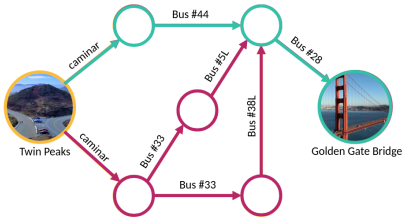
- $V$  es un conjunto de vértices o nodos
- $E$  es un conjunto de aristas o arcos que relacionan estos nodos

El orden del grafo  $G$  es su número de vértices  $|V|$

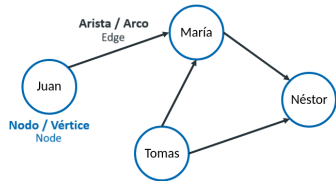


# Ejemplos

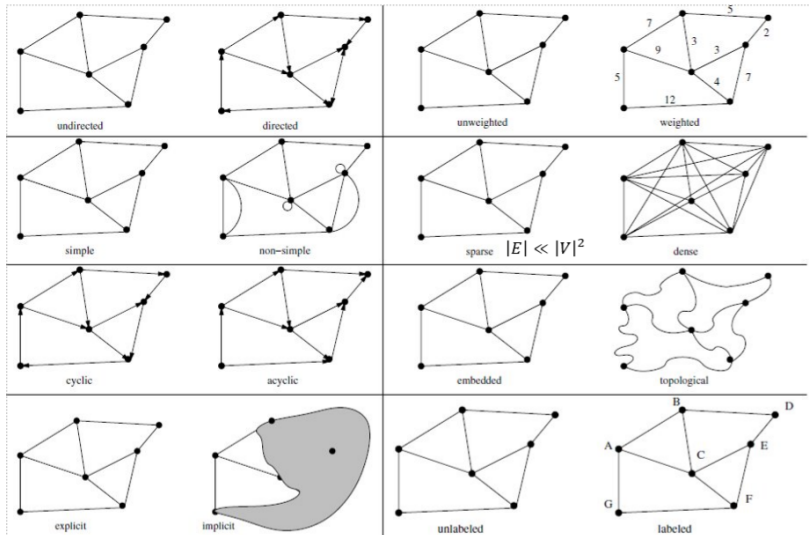
## Rutas



## Relaciones

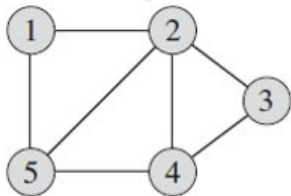


# Tipos

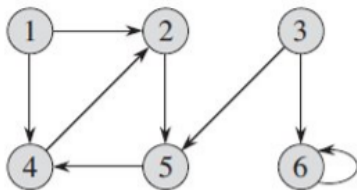


# Tipos

No dirigido

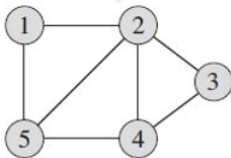


Dirigido

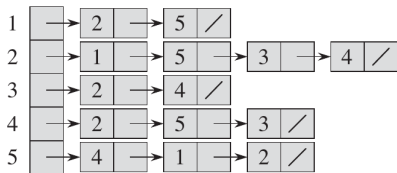


# Representación

No dirigido



Lista de Adyacencia



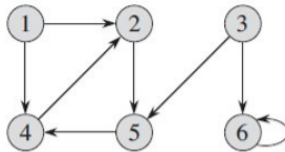
Matriz de Adyacencia

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

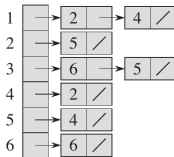


# Representación

## Dirigido



## Lista de Adyacencia



## Matriz de Adyacencia

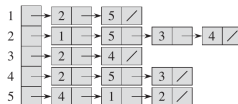
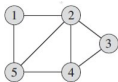
	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

# Lista de adyacencia

Consiste en un arreglo,  $adj$ , de  $|V|$  listas

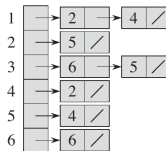
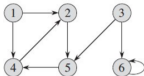
Para cada  $u \in V$ ,  $adj(u)$  contine todos los vertices  $v$  tales que existe un arco  $(u, v) \in E$

## No Dirigido



La suma de las longitudes de todas las listas es  $2 * |E|$

## Dirigido



La suma de las longitudes de todas las listas es  $|E|$

La cantidad de memoria es  $\Phi(V + e)$

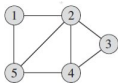
## Matriz de adyacencia

Consiste en una matriz  $a$ , de  $|V| \times |V|$  de  $[0, 1]$

$$a(u, v) = 1 \text{ si esiste un arco } (u, v) \in E$$

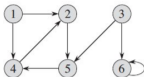
0 de lo contrario

No Dirigido



	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

## Dirigido



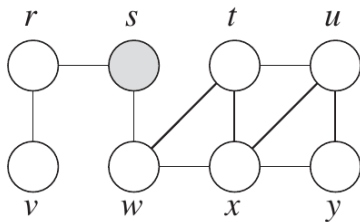
	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

La cantidad de memoria es  $\Phi(V * V)$

## Busqueda por anchura

Dado un grafo  $G = (V, E)$  y un vértice fuente denominado  $s$ , la **BFS** (Breadth-First Search) explora sistemáticamente los arcos de  $G$  para descubrir cada vértice que es alcanzable desde  $s$ .

Este algoritmo descubre primero todos los vértices a una distancia  $k$  desde  $s$  antes de descubrir los vértices a una distancia  $k + 1$ .



## Busqueda por anchura

BFS( $G, s$ )

```
1  for each vertex  $u \in G.V - \{s\}$ 
2       $u.color = \text{WHITE}$ 
3       $u.d = \infty$ 
4       $u.\pi = \text{NIL}$ 
5   $s.color = \text{GRAY}$ 
6   $s.d = 0$ 
7   $s.\pi = \text{NIL}$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11      $u = \text{DEQUEUE}(Q)$ 
12     for each  $v \in G.Adj[u]$ 
13         if  $v.color == \text{WHITE}$ 
14              $v.color = \text{GRAY}$ 
15              $v.d = u.d + 1$ 
16              $v.\pi = u$ 
17             ENQUEUE( $Q, v$ )
18      $u.color = \text{BLACK}$ 
```

- Se asume una representación usando la lista de adyacencia.
- El algoritmo usa una cola  $Q$  para manejar el conjunto de vértices.
- Si  $u$  no tiene predecesor o  $u$  no ha sido descubierto entonces  $u.\pi = \text{NIL}$ .

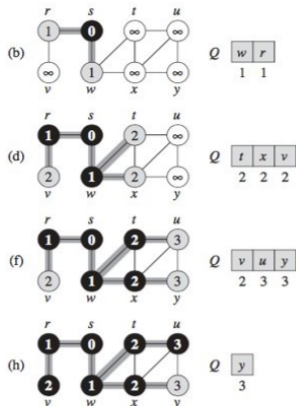
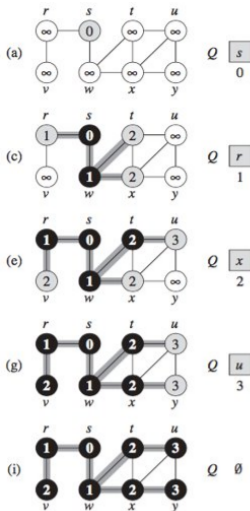
# Busqueda por anchura

BFS( $G, s$ )

```

1  for each vertex  $u \in G.V - \{s\}$ 
2     $u.color = WHITE$ 
3     $u.d = \infty$ 
4     $u.\pi = NIL$ 
5   $s.color = GRAY$ 
6   $s.d = 0$ 
7   $s.\pi = NIL$ 
8   $Q = \emptyset$ 
9  ENQUEUE( $Q, s$ )
10 while  $Q \neq \emptyset$ 
11    $u = DEQUEUE(Q)$ 
12   for each  $v \in G.Adj[u]$ 
13     if  $v.color == WHITE$ 
14        $v.color = GRAY$ 
15        $v.d = u.d + 1$ 
16        $v.\pi = u$ 
17       ENQUEUE( $Q, v$ )
18    $u.color = BLACK$ 

```



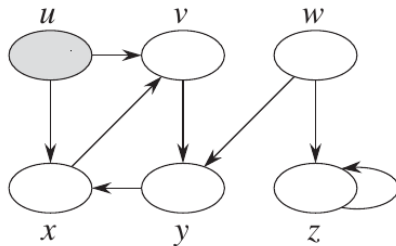
## Busqueda por profundidad

La estrategia es buscar cada vez más profundo en el grafo siempre y cuando sea posible.

Su funcionamiento consiste en ir expandiendo los vértices que va localizando, de forma recurrente, en un camino concreto.

Cuando ya no quedan más nodos que visitar en dicho camino, regresa (backtracking), y repite el mismo proceso con cada uno de los hermanos del vértice ya procesado.

Si quedan vertices por explorar, se selecciona uno de ellos y se repite la búsqueda desde esa fuente.



# Busqueda por profundidad

- Se asume una representación usando la lista de adyacencia.
- Los vértices se colorean para indicar su estado
- $v.\pi = u$  si  $v$  fue visitado al recorrer la lista de adyacencia del vértice visitado  $u$
- Si  $v$  no tiene predecesor o  $u$  no ha sido descubierto entonces  $u.\pi = NIL$ .

DFS( $G$ )

```
1  for each vertex  $u \in G.V$ 
2       $u.color = WHITE$ 
3       $u.\pi = NIL$ 
4   $time = 0$ 
5  for each vertex  $u \in G.V$ 
6      if  $u.color == WHITE$ 
7          DFS-VISIT( $G, u$ )
```

DFS-VISIT( $G, u$ )

```
1   $time = time + 1$ 
2   $u.d = time$ 
3   $u.color = GRAY$ 
4  for each  $v \in G.Adj[u]$ 
5      if  $v.color == WHITE$ 
6           $v.\pi = u$ 
7          DFS-VISIT( $G, v$ )
8   $u.color = BLACK$ 
9   $time = time + 1$ 
10  $u.f = time$ 
```



# Busqueda por profundidad

DFS( $G$ )

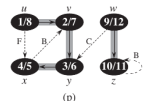
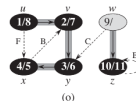
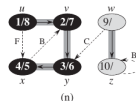
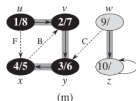
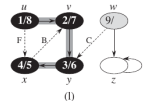
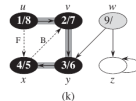
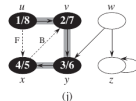
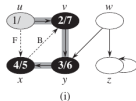
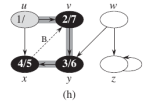
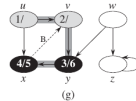
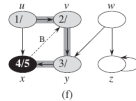
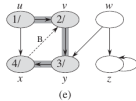
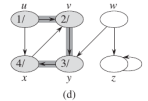
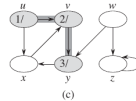
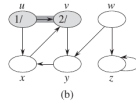
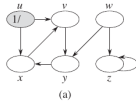
```

1  for each vertex  $u \in G.V$ 
2       $u.color = \text{WHITE}$ 
3       $u.\pi = \text{NIL}$ 
4   $time = 0$ 
5  for each vertex  $u \in G.V$ 
6      if  $u.color == \text{WHITE}$ 
7          DFS-VISIT( $G, u$ )
    
```

DFS-VISIT( $G, u$ )

```

1   $time = time + 1$ 
2   $u.d = time$ 
3   $u.color = \text{GRAY}$ 
4  for each  $v \in G.Adj[u]$ 
5      if  $v.color == \text{WHITE}$ 
6           $v.\pi = u$ 
7          DFS-VISIT( $G, v$ )
8   $u.color = \text{BLACK}$ 
9   $time = time + 1$ 
10  $u.f = time$ 
    
```



# Agenda

## ① Grafos

Conceptos

Representaciones

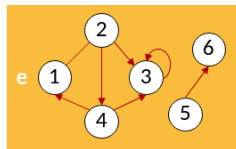
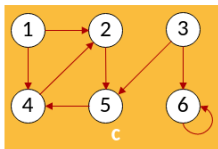
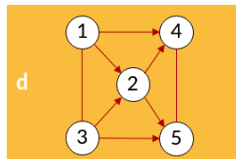
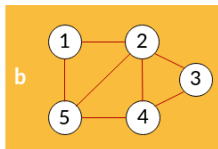
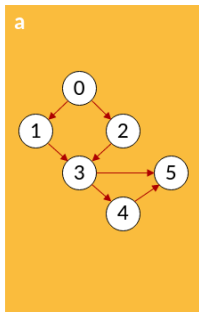
Recorridos

## ② Aspectos finales

Ejercicios

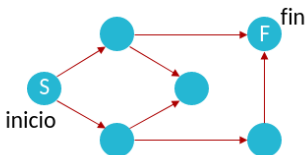
# Ejercicios

Represente los siguientes grafos con una lista y con una matriz de adyacencias:

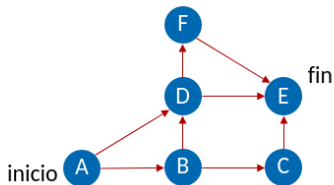


## Ejercicios

Ejecute el algoritmo BFS en los siguientes grafos para obtener la respuesta solicitada:



Encuentre la longitud de la ruta más corta de inicio a fin



Encuentre la longitud de la ruta más corta de A a E

Elabore una lista del orden en que las siguientes actividades pueden realizarse (una tarea es dependiente de otra si existe un arco entre ellas)

