Programación Avanzada y Estructuras de Datos

8. Grafos

Víctor M. Sánchez Cartagena

Grado en Ingeniería en Inteligencia Artificial
Dep. Lenguajes y Sistemas Informáticos
Universidad de Alicante

11 de diciembre de 2024

Índice

Ejemplo introductorio

¿Qué estructura de datos emplearías para representar los siguientes problemas?

- Obtener la combinación de vuelos más barata para ir de un aeropuerto a otro
- Obtener la ruta por carretera más corta entre dos ciudades
- Crear redes de comunicaciones con mínimo coste para que todas las ciudades deseadas estén conectadas

Ejemplo introductorio

¿Qué estructura de datos emplearías para representar los siguientes problemas?

- Obtener la combinación de vuelos más barata para ir de un aeropuerto a otro
- Obtener la ruta por carretera más corta entre dos ciudades
- Crear redes de comunicaciones con mínimo coste para que todas las ciudades deseadas estén conectadas

Un **grafo**

Definición

Definición de grafo

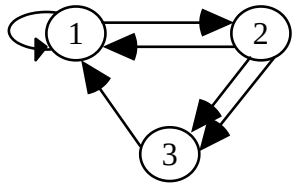
Un grafo G está formado por dos conjuntos V y A: $G = (V, A)$

- V es un conjunto finito no vacío de **vértices**
- A es un conjunto de **aristas** o **arcos**, tal que cada arista a_i es un par de vértices $a_i = (v_j, v_k)$

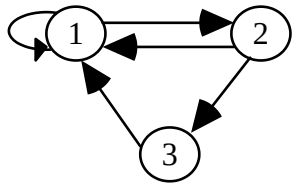
Tipos de grafo

Dependiendo de las restricciones sobre $a_i = (v_j, v_k)$:

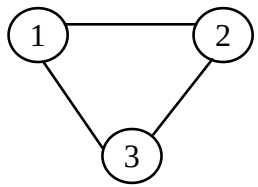
- **Multigrafo**: sin restricciones, existen arcos reflexivos y múltiples ocurrencias del mismo arco



- **Digrafo**: no hay múltiples ocurrencias del mismo arco



- **Grafo no dirigido**: las **aristas** indican que los vértices están conectados en ambos sentidos. No hay aristas reflexivas



Tipos de grafos

Pregunta

¿Cuál es el máximo número de aristas en un grafo no dirigido? ¿Cuál es el máximo número de arcos en un digrafo con n vértices?

Tipos de grafos

Pregunta

¿Cuál es el máximo número de aristas en un grafo no dirigido? ¿Cuál es el máximo número de arcos en un digrafo con n vértices?

No dirigido: $\frac{n \cdot (n-1)}{2}$ aristas

Tipos de grafos

Pregunta

¿Cuál es el máximo número de aristas en un grafo no dirigido? ¿Cuál es el máximo número de arcos en un digrafo con n vértices?

No dirigido: $\frac{n \cdot (n-1)}{2}$ aristas

Dirigido: n^2 arcos

Conceptos

- (Grafos no dirigidos) Los vértices v_1 y v_2 son adyacentes si $(v_1, v_2) \in A \vee (v_2, v_1) \in A$. La arista (v_1, v_2) (o (v_2, v_1)) es incidente a ambos vértices
- (Grafos dirigidos) El vértice v_1 es adyacente hacia v_2 y el vértice v_2 es adyacente desde v_1 si $(v_1, v_2) \in A$. El arco (v_1, v_2) es incidente a v_1 y v_2
- (Grafos no dirigidos) Adyacencia de un vértice: conjunto de vértices tal que hay una arista que los relaciona:

$$\mathrm{Ay}(x) = \{v_i : v_i \in V \wedge (x, v_i) \in A\}$$

Conceptos

- (Grafos dirigidos) Adyacencia de entrada:

$$\mathrm{Ay_E}(x) = \{v_i : v_i \in V \wedge (v_i, x) \in A\}$$

- (Grafos dirigidos) Adyacencia de salida:

$$\mathrm{Ay_S}(x) = \{v_i : v_i \in V \wedge (x, v_i) \in A\}$$

- (Grafos dirigidos) Grado de entrada: $|\mathrm{Ay_E}(x)|$
- (Grafos dirigidos) Grado de salida: $|\mathrm{Ay_S}(x)|$
- (Grafos dirigidos) Grado: $|\mathrm{Ay_E}(x)| + |\mathrm{Ay_S}(x)|$
- (Grafos no dirigidos) Grado: $|\mathrm{Ay}(x)|$

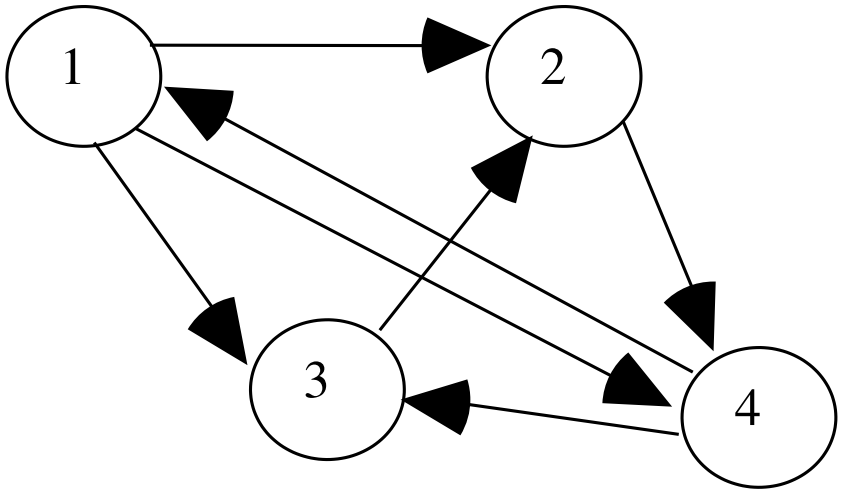
Conceptos

- Un camino desde v_p hasta v_q es una secuencia de vértices $(v_p, v_1, v_2, ..., v_n, v_q)$, tal que (v_p, v_1), (v_1, v_2), ..., (v_n, v_q) son arcos/aristas en A
- La longitud de un camino es el número de arcos/aristas que contiene
- Un camino simple no tiene vértices repetidos (excepto el primer y último)
- Un ciclo es un camino simple en el que el primer y último vértice coinciden

Conceptos

Pregunta

- ¿$(3, 2, 4, 3)$ es un ciclo? ¿De qué longitud?
- ¿$(3, 2, 4, 1, 4, 3)$ es un ciclo? ¿De qué longitud?



Conceptos

- Un subgrafo de un grafo $G = (V, A)$ es un grafo $G' = (V', A')$ tal que $V' \subseteq V$ y $A' \subseteq A$
- Un árbol extendido de un grafo $G = (V, A)$ es un subgrafo $T = (V', A')$ de G tal que T es un árbol y $V' = V$

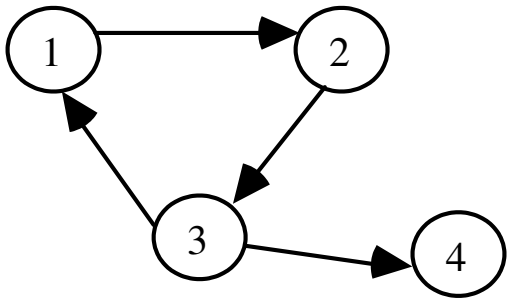
Conceptos

- Un grafo es conexo si $\forall v_i, v_j \in V$, existe un camino de v_i a v_j
- Las componentes fuertemente conexas de un grafo son el conjunto maximal de subgrafos conexos
  - Un grafo conexo tiene una única componente fuertemente conexa
- Un grafo acíclico es aquel que no tiene ciclos

Índice

Representación

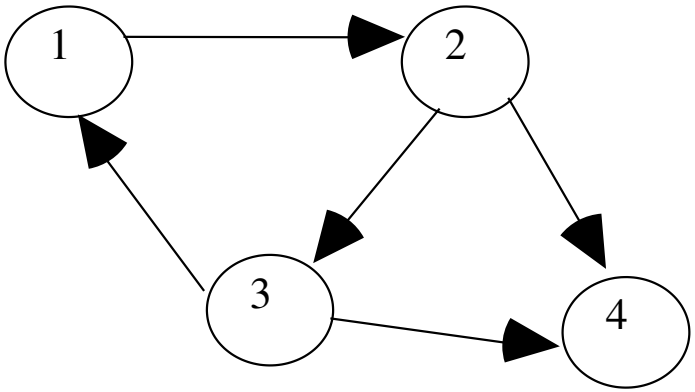
- Dado un grafo $G = (V, A)$ con $n = |V|$ (n vértices), la **matriz de adyacencia** es una matriz M cuadrada n x n tal que:
  - $M[i][j] = 1$ si $(v_i, v_j) \in A$
  - $M[i][j] = 0$ si $(v_i, v_j) \notin A$
- Para grafos no dirigidos, la matriz es simétrica respecto a la diagonal principal, que tiene todos los valores a 0.

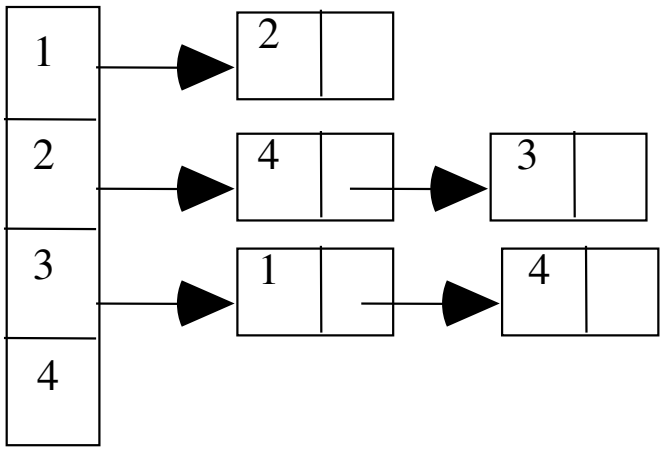


$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Representación

- **Lista de adyacencia**: Lista enlazada con la adyacencia de salida para cada vértice





Representación

Pregunta

¿Cuál es la complejidad temporal en el peor caso, para un digrafo con n vértices, de las siguientes operaciones en cada representación?

- Búsqueda (comprobar si hay un arco de v_1 a v_2)
- Inserción de un arco
- Cálculo de la adyacencia de salida de un vértice
- Cálculo de la adyacencia de entrada de un vértice

Operación	Matriz	Lista
Búsqueda		
Inserción		
$\mathrm{Ay_E}(x)$		
$\mathrm{Ay_S}(x)$		

Representación

Pregunta

¿Cuál es la complejidad temporal en el peor caso, para un digrafo con n vértices, de las siguientes operaciones en cada representación?

- Búsqueda (comprobar si hay un arco de v_1 a v_2)
- Inserción de un arco
- Cálculo de la adyacencia de salida de un vértice
- Cálculo de la adyacencia de entrada de un vértice

Operación	Matriz	Lista
Búsqueda	$O(1)$	$O(n)$
Inserción	$O(1)$	$O(n)$
$\mathrm{Ay_E}(x)$	$O(n)$	$O(n^2)$
$\mathrm{Ay_S}(x)$	$O(n)$	$O(n)$

Representación

Pregunta

¿Hay algún motivo para emplear lista de adyacencia en vez de matriz de adyacencia?

Representación

Pregunta

¿Hay algún motivo para emplear lista de adyacencia en vez de matriz de adyacencia?

Sí, la complejidad espacial. En la matriz siempre es $\Theta(n^2)$, y en la lista es proporcional al número de arcos. Preferiremos la representación en lista para grafos dispersos.

Índice

Recorrido en profundidad

- **Recorrer** un grafo consiste en listar todos sus vértices, una vez cada uno
- El recorrido en profundidad es una generalización del preorden aplicado a grafos

```
visitados ← {}
function DFS(v,G)
    visitados ← visitados ∪ {v}
    visitar(v)
    for w ∈ Ay_S(v) do
        if w ∉ visitados then
            DFS(w, G)
```

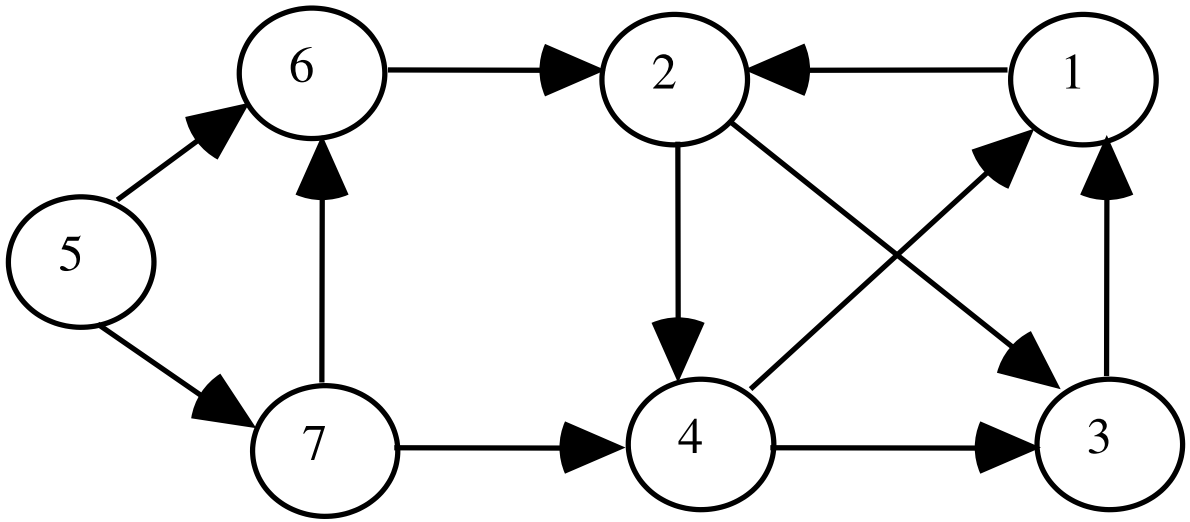
Recorrido en profundidad

- Los arcos (v, w) permiten construir un **árbol extendido en profundidad**
- Una vez construido el árbol, se pueden clasificar los arcos del grafo:
  - De árbol: los que forman parte del árbol extendido en profundidad
  - De retroceso: van de un vértice a un ascendente en el árbol
  - De avance: van de un vértice a un descendiente en el árbol
  - De cruce: ninguno de los anteriores
- **Bosque extendido en profundidad**: al terminar $DFS(v)$, se vuelve a iniciar el algoritmo si todavía no se han visitado todos los vértices
- El grafo es acíclico $\Leftrightarrow$ el bosque extendido en profundidad no tiene arcos de retroceso
- El árbol extendido en profundidad no contiene todos los vértices $\rightarrow$ el grafo no es conexo

Recorrido en profundidad

Ejercicio

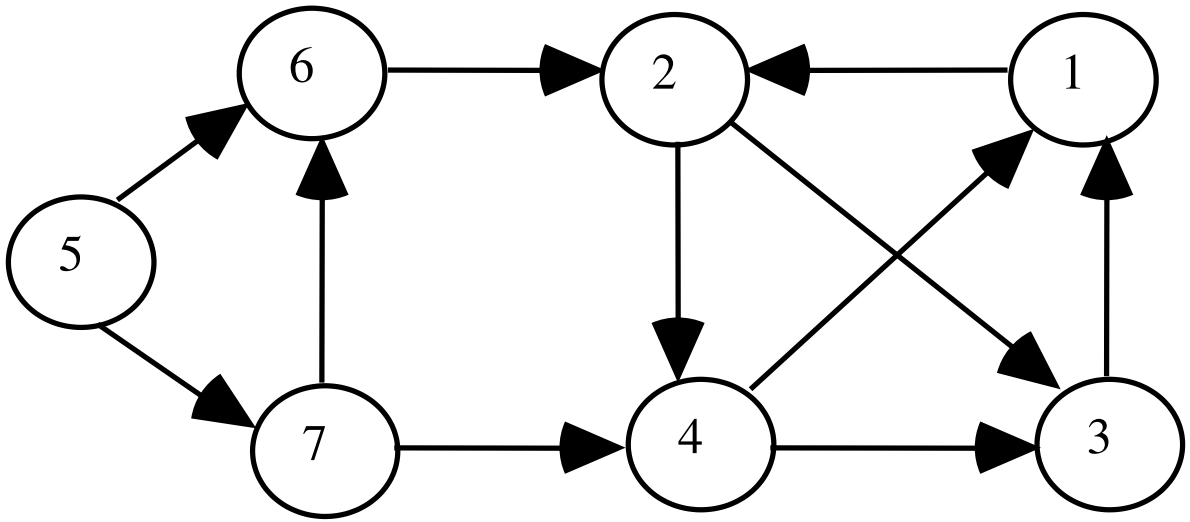
Calcula el recorrido en profundidad (DFS) del siguiente grafo, partiendo de los vértices 1,2,3,4 y 5. Recorre el conjunto Ay_S en orden de menor a mayor.



Recorrido en profundidad

Ejercicio

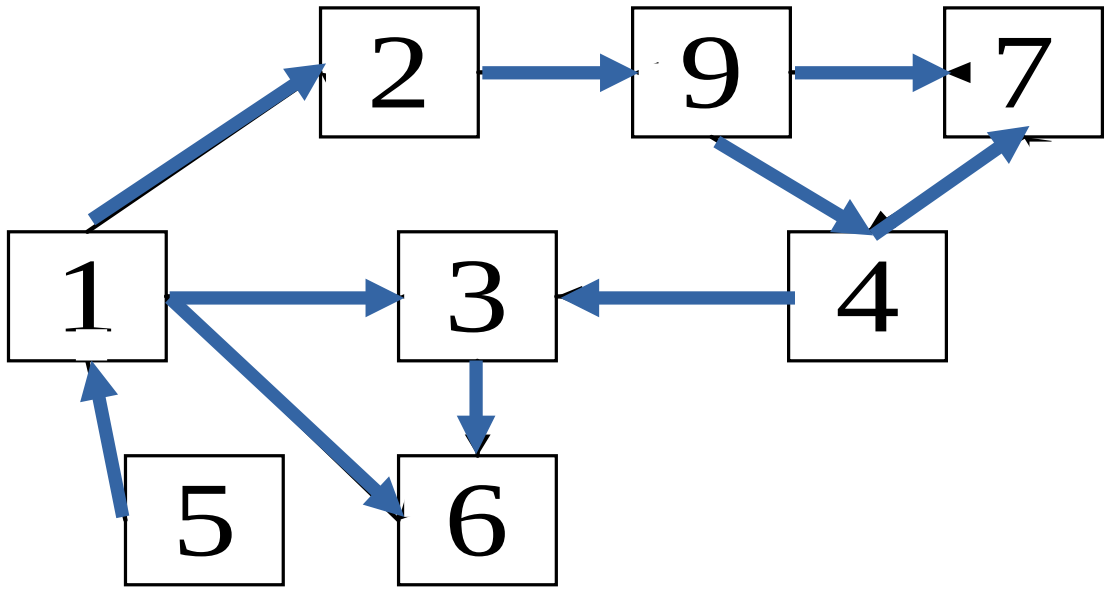
Calcula el bosque extendido en profundidad del siguiente grafo partiendo del vértice 5. Recorre el conjunto Ay_S en orden de mayor a menor. Clasifica los arcos.



Recorrido en profundidad

Ejercicio

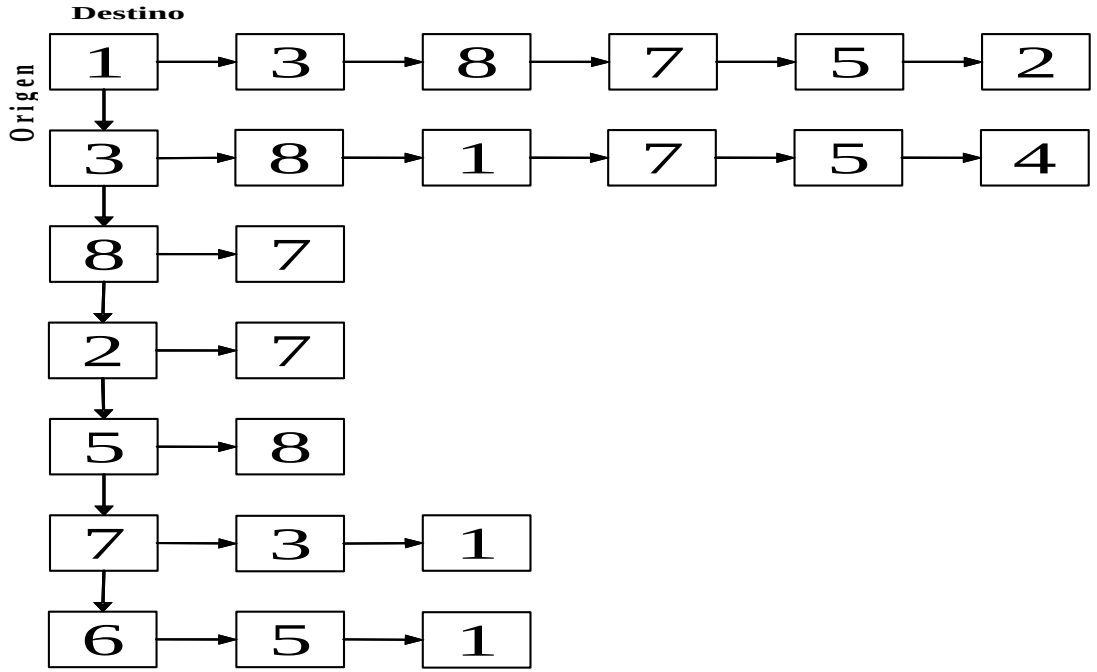
Calcula el bosque extendido en profundidad del siguiente grafo partiendo del vértice 1. Recorre el conjunto Ay_S en orden de menor a mayor. Clasifica los arcos. ¿Este grafo tiene ciclos? ¿Es conexo?



Recorrido en profundidad

Ejercicio

Calcula el bosque extendido en profundidad del siguiente grafo partiendo del vértice 1. Recorre el conjunto Ay_S en el mismo orden que aparece en las listas de adyacencia. Clasifica los arcos. ¿Este grafo tiene ciclos? ¿Es conexo?



Índice

Recorrido en anchura

function BFS(v,G)
    visitados $\leftarrow \{\}$
    cola $\leftarrow ()$
    visitados $\leftarrow$ visitados $\cup \{v\}$
    enqueue(*cola*, *v*)
    visitar(v)
    while cola $\neq ()$ **do**
        $w_1 \leftarrow$ front(*cola*)
        dequeue(*cola*)
        for $w_2 \in \text{Ay}_S(w_1)$ **do**
            if $w_2 \notin$ visitados **then**
                visitar(w_2)
                enqueue(*cola*, w_2)
                visitados $\leftarrow$ visitados $\cup \{w_2\}$

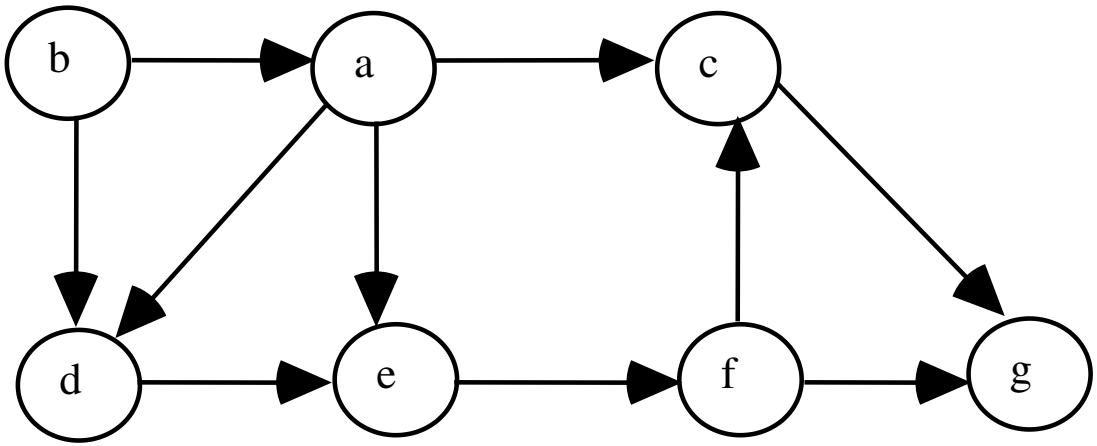
Recorrido en anchura

- Los arcos (w_1, w_2) permiten construir un **árbol extendido en anchura**
- Los arcos se pueden clasificar del mismo modo que en el recorrido en profundidad
- Las propiedades sobre ciclos y grafos conexos también se mantienen
- El bosque extendido en anchura contiene los **caminos más cortos** desde el vértice inicial a cada vértice

Recorrido en anchura

Ejercicio

Calcula el recorrido en anchura (BFS) del siguiente grafo, partiendo de los vértices a, b y d. Recorre el conjunto Ay_S en orden alfabético. Dibuja los correspondientes árboles extendidos en anchura.



Recorridos en grafos no dirigidos

- Se aplican los mismos algoritmos DFS y BFS presentados anteriormente
- Sólo hay dos tipos de aristas:
  - De árbol
  - De retroceso: forman ciclos
- El grafo es conexo $\Leftrightarrow$ el árbol extendido en profundidad contiene todos los vértices

Recorridos en grafos no dirigidos

Ejercicio

Calcula el bosque extendido en profundidad y el bosque extendido en anchura a partir del vértice 1. Clasifica los arcos. Recorre el conjunto Ay de menor a mayor. El grafo es **no dirigido** y la lista de adyacencia sólo contiene las aristas en uno de los dos sentidos.

$1 \to 3 \to 4 \to 6 \to 8 \to 9$
$2 \to 4 \to 5 \to 7 \to 10$
$3 \to 5 \to 6$
$4 \to 6 \to 8$
$5 \to 7 \to 8 \to 10$
$6 \to 7 \to 8 \to 9$
$7 \to 8 \to 9$
$8 \to 9 \to 10$
$9 \to 10$
10

Ejercicio adicional

Ejercicio

Calcula el bosque extendido en profundidad a partir del vértice 1 del siguiente grafo **dirigido**. Clasifica los arcos. Recorre el conjunto Ay_S de menor a mayor. Identifica al menos 4 ciclos en el grafo.

$1 \to 3 \to 8$
$2 \to 4 \to 5 \to 7 \to 9 \to 10$
$3 \to 5 \to 6 \to 10$
$4 \to 2 \to 6 \to 8$
$5 \to 7$
$6 \to 7 \to 8 \to 10$
7
$8 \to 1 \to 10$
$9 \to 4$
$10 \to 3 \to 5$
$11 \to 10 \to 12 \to 13$
$12 \to 2 \to 3 \to 11$

Índice

Grafos acíclicos dirigidos

- La detección de ciclos (arcos de retroceso) permite comprobar si un grafo es acíclico
- Los grafos acíclicos tienen múltiples aplicaciones:
  - Planificación de tareas: los vértices representan tareas y los arcos, requisitos: es necesario completar la tarea A antes de comenzar la tarea B
  - Dependencias entre asignaturas en un plan de estudios
  - Herencia entre clases de un programa en C++
  - etc.
- **Ordenamiento topológico**: ordenar los vértices de tal manera que para todo arco (v_i, v_j), v_i aparece antes que v_j en el ordenamiento
  - Para la planificación de tareas, implica ordenar las tareas de manera que todos los requisitos se cumplen