Guía de Estudio: Grafos, DFS, BFS y Ordenación Topológica

Objetivo: entender desde cero cómo modelar grafos, representarlos (matriz y lista de adyacencia), recorrerlos (DFS/BFS), evitar ciclos con visitados, analizar complejidad y resolver ordenación topológica (3 enfoques). Incluye ejemplos que podés replicar en papel para un examen.

Pensado para el TP de Grafos y para rendir: contiene definiciones, pseudocódigo y diagramas.

Versión de estudio • Preparado para práctica y examen

1) Conceptos básicos de grafos

Grafo G = (V, E): V vértices (nodos) y E aristas (enlaces).

Dirigido: las aristas tienen sentido (u → v). No dirigido: aristas sin sentido (u—v).

Grado de entrada (in-degree): cantidad de aristas que llegan a un vértice. Grado de salida: las que salen.

Camino: secuencia de vértices conectados. Ciclo: camino que empieza y termina en el mismo vértice.

Árbol vs. Grafo: los árboles son casos particulares (sin ciclos y con raíz). Un grafo es más general.

2) Representaciones: lista vs. matriz de adyacencia

Lista de adyacencia: para cada vértice se guarda la lista de vecinos. Pros: eficiente en grafos ralos; recorrer vecinos es O(grado). Contras: consultar si (u,v) existe puede ser O(grado(u)).

Matriz de adyacencia: matriz $n \times n$ con 1 si hay arista (u,v), 0 si no. Pros: consulta de existencia O(1). Contras: ocupa $O(n^2)$, incluso si hay pocas aristas.

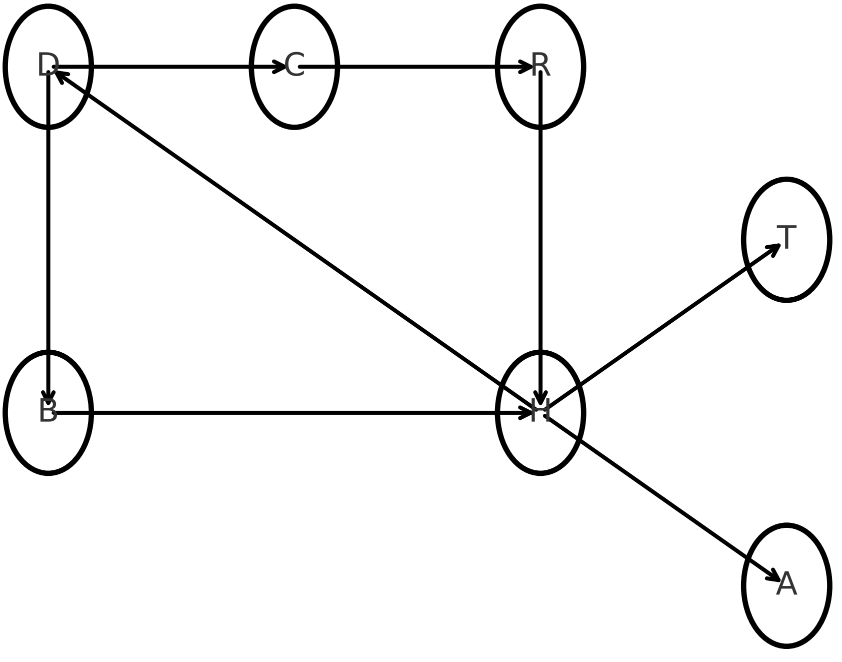
Recomendación práctica: lista para la mayoría de los TP, matriz si necesitás muchas consultas de existencia.

Matriz de adyacencia (ejemplo dirigido)

	D	C	R	H	T	A	B
D	0	1	0	0	0	0	1
C	0	0	1	0	0	0	0
R	0	0	0	1	0	0	0
H	1	0	0	0	1	1	0
T	0	0	0	0	0	0	0
A	0	0	0	0	0	0	0
B	0	0	0	1	0	0	0

Fila = origen, Columna = destino. 1 indica arista presente.

Grafo dirigido de ejemplo



El mismo grafo se usará para ilustrar DFS (profundidad) y BFS (amplitud).

3) Visitados: por qué son clave

Arreglo/Mapa de visitados (boolean): evita ciclos y asegura terminar. Inicialmente todo en false.

Al visitar un vértice u: marcar visitado[u] = true y realizar la acción (agregar a lista, imprimir, etc.).

Sin visitados, en grafos con ciclos podés quedar en un loop infinito (por ejemplo, H → D → C → R → H...).

4) DFS (Depth-First Search) — teoría y pseudocódigo

Idea: desde un vértice u, vas lo más profundo posible por una rama antes de retroceder (backtracking).

Suele implementarse de forma recursiva. Analogia: similar al preorden en árboles (procesa nodo y baja).

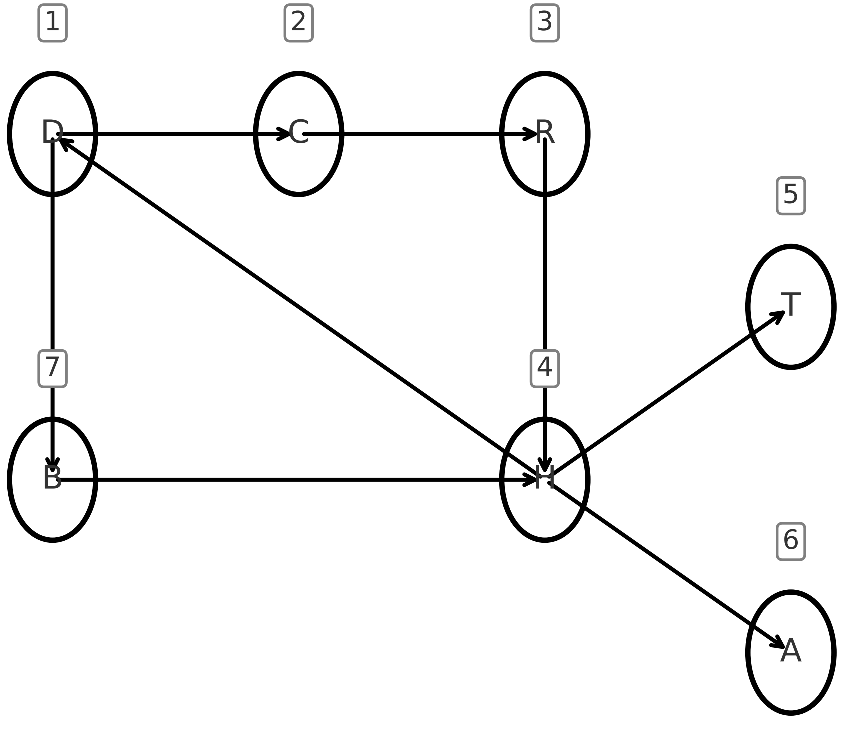
Complejidad: O(|V| + |E|) con lista de adyacencia.

Pseudocódigo (recursivo):

```
DFS(u):
  visitado[u] = true
  procesar(u)  // aquí hacés lo que necesites con el vértice
  para cada v en adyacentes(u):
    si !visitado[v]: DFS(v)
```

Recorrido total: inicializá visitados en false, y para cada vértice u no visitado hacé DFS(u).

DFS: un orden posible desde D



El orden concreto depende del orden de carga/iteración de adyacentes.

5) BFS (Breadth-First Search) — teoría y pseudocódigo

Idea: procesa primero los vecinos a distancia 1 del origen, luego los de distancia 2, etc. Usa una COLA.

Analogia: similar al recorrido por niveles en árboles.

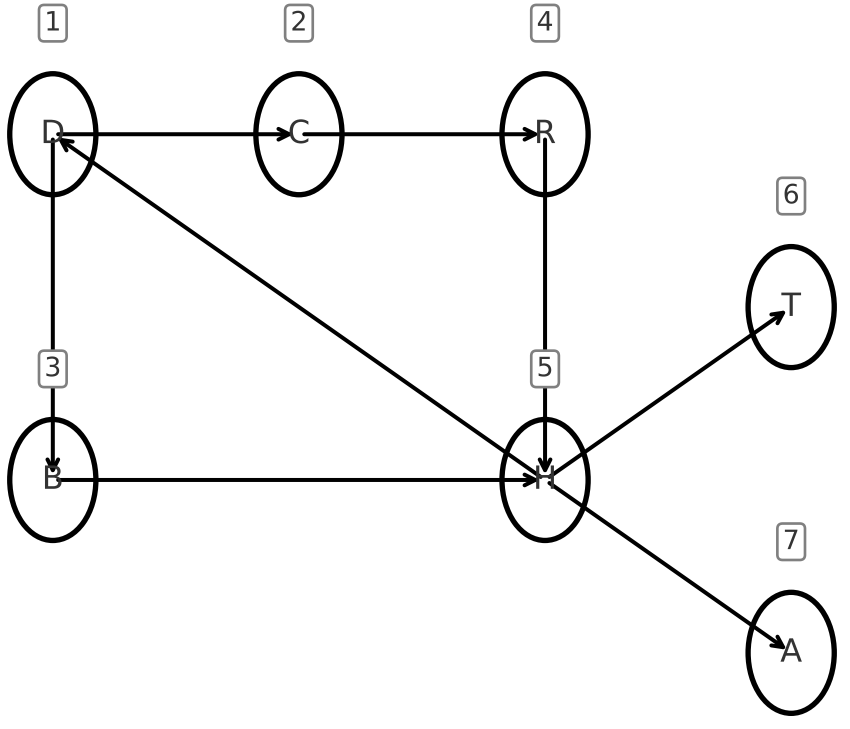
Complejidad: O(|V| + |E|) con lista de adyacencia.

Pseudocódigo (con cola):

```
BFS(origen):
  crear cola Q; visitado[origen] = true; encolar(origen)
  mientras Q no vacía:
    u = desencolar(Q); procesar(u)
    para cada v en adyacentes(u):
      si !visitado[v]: visitado[v] = true; encolar(v)
```

Recorrido total: si el grafo no es conexo, repetí BFS desde otros vértices no visitados.

BFS: un orden posible desde D



Primero D (distancia 0), luego C y B (distancia 1), luego R y H (2), y así.

6) DFS vs. BFS — similitudes y diferencias

Similitudes: ambos visitan cada vértice a lo sumo una vez (con visitados) y corren en $O(|V|+|E|)$.

Diferencias: DFS usa recursión/pila implícita y explora en profundidad; BFS usa cola y explora por capas.

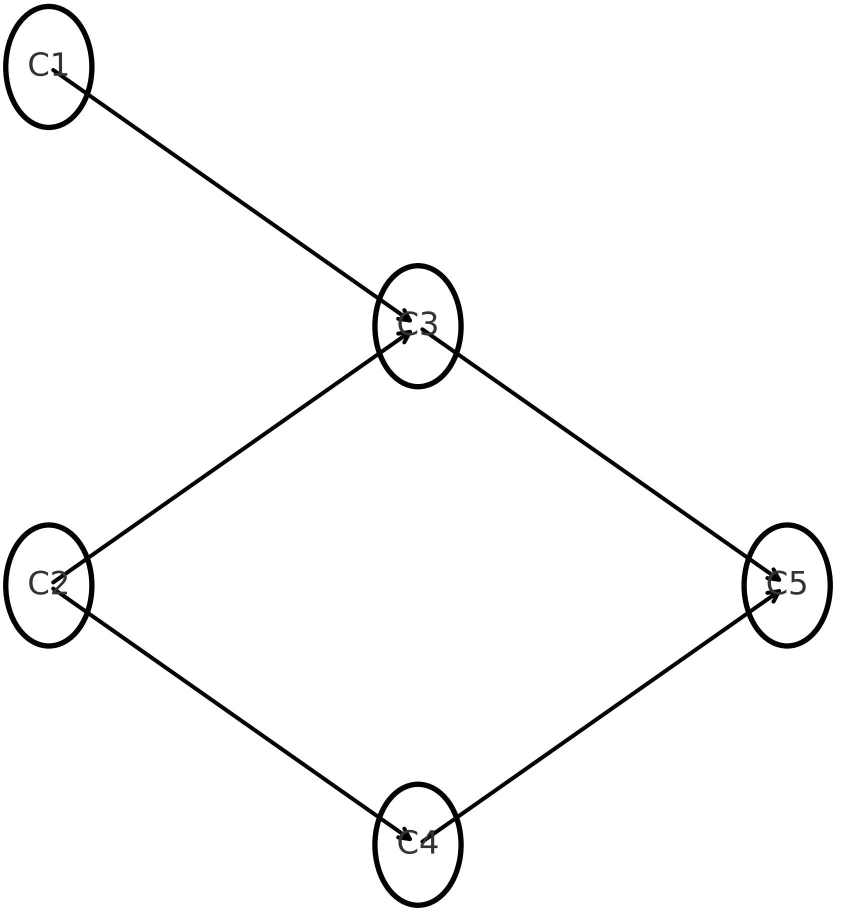
Elección: depende del problema (por ejemplo, distancias mínimas en grafos no ponderados → BFS).

7) Ordenación topológica (DAG) — idea general

Aplica solo a grafos dirigidos acíclicos (DAG). Devuelve un orden lineal de vértices tal que para toda arista u→v, u aparece antes que v.

No es única: puede haber múltiples órdenes válidos.

DAG de ejemplo para sort topológico



8) Sort topológico — Método 1: Grados de entrada (búsqueda secuencial)

Estrategia:

1) Calcular grado de entrada (in-degree) de todos los vértices.

2) Repetir: elegir algún vértice con in-degree = 0 que NO esté visitado; marcarlo y 'remover' sus aristas salientes (decrementar in-degree de sus vecinos).

3) Continuar hasta visitar todos los vértices.

Complejidad: O(|V|^2 + |E|) si cada vez buscás cero en arreglo secuencialmente.

Ejemplo (DAG): un posible orden: C2, C1, C4, C3, C5 (podrían existir otros válidos).

9) Sort topológico — Método 2: Usando Pila/Cola de ceros (optimizado)

Mejora: mantener una estructura (pila o cola) con todos los vértices que tienen in-degree = 0.

Proceso: extraer uno, agregarlo al orden, y cuando algún vecino quede en 0, encolarlo/apilarlo.

Complejidad: O(|V| + |E|).

La diferencia entre usar pila o cola solo afecta al orden resultante (sigue siendo válido).

10) Sort topológico — Método 3: DFS en posorden

Estrategia: hacer DFS y, cuando terminás de explorar un vértice (posorden), lo agregás a una pila o lo numerás.

Al final, invertir ese orden produce un sort topológico válido.

Ventaja: también O(|V| + |E|) y simple si ya tenés DFS implementado.

11) Consejos para el TP (interfaces y recorridos)

Interfaces típicas: Grafo<T>, Vertice<T>, Arista<T>. Métodos comunes: agregarVertice, eliminarVertice, conectar(u,v), desconectar(u,v), esVacio, listaDeVertices, adyacentes(u), peso(u,v) (si es pesado), etc.

Dónde insertar la lógica de 'procesar(v)': en el momento exacto en que marcás visitado[v] = true.

Para pruebas: imprimí el orden de visita, o devolvé una Lista con el orden recorrido.

Cuidado con grafos no conexos: recorré desde todos los vértices no visitados.

12) Checklist de examen + ejercicios para practicar

✔ Definir grafo, dirigido/no, grado in/out, ciclo, camino.

✔ Diferenciar matriz vs. lista de adyacencia (costos y memoria).

✔ Escribir y explicar DFS y BFS, con visitados; complejidad O(|V|+|E|).

✔ Elegir cuándo usar DFS vs. BFS según el problema.

✔ Explicar sort topológico, condiciones (DAG) y tres métodos.

Ejercicios sugeridos:

(1) Dado el grafo de la guía, listar un posible orden DFS desde D y otro distinto cambiando el orden de adyacentes.

(2) Hacer BFS desde D e indicar niveles (distancias mínimas en aristas no pesadas).

(3) Para el DAG C1..C5, obtener al menos dos órdenes topológicos válidos y marcar qué vértices arrancan con grado 0.

¡Éxitos! Repasá con estos esquemas y practicá a mano 2 o 3 variantes de cada ejercicio.