Grafos (Métodos de búsqueda)

VARGAS BERMÚDEZ PABLO

¿Qué es?

Es un tipo abstracto de datos (TAD), que consiste en un conjunto de nodos (también llamados vértices) y un conjunto de arcos (aristas) que establecen relaciones entre los nodos.

Normalmente, un grafo se define como G=(V,E), siendo V un conjunto cuyos elementos son los vértices del grafo y, E uno cuyos elementos son las aristas (edges en inglés), las cuales son pares (ordenados si el grafo es dirigido) de elementos en V

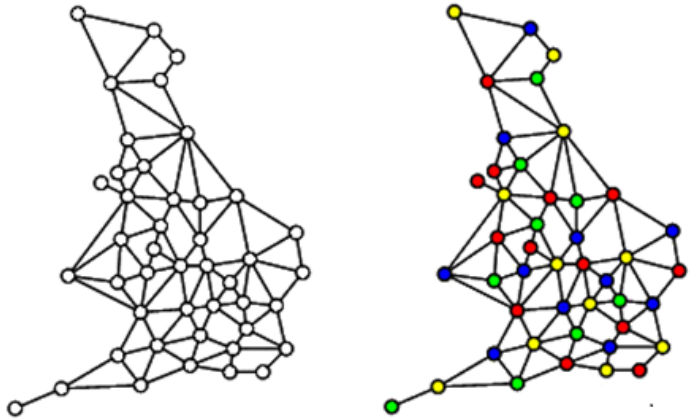
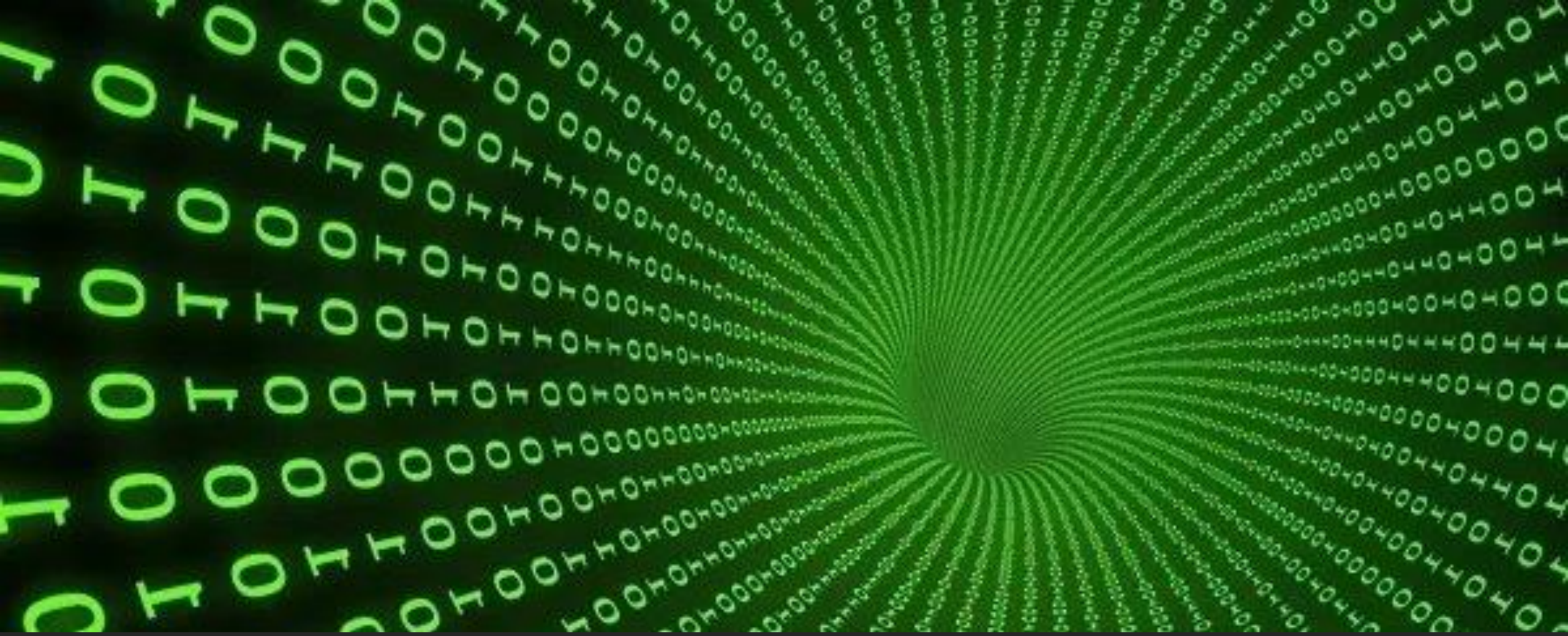
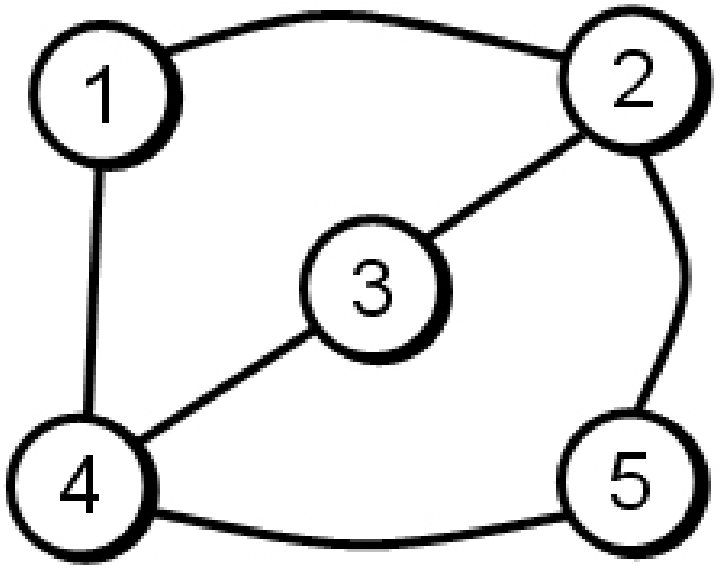


Figura 1



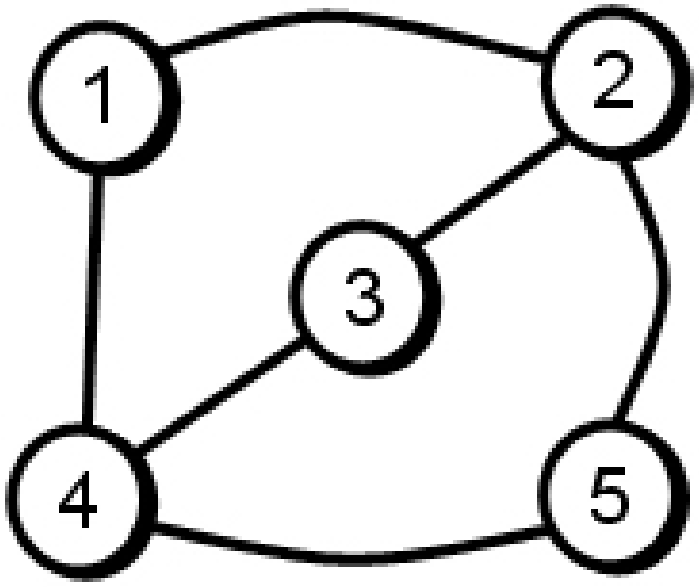
¿Cómo se representa?

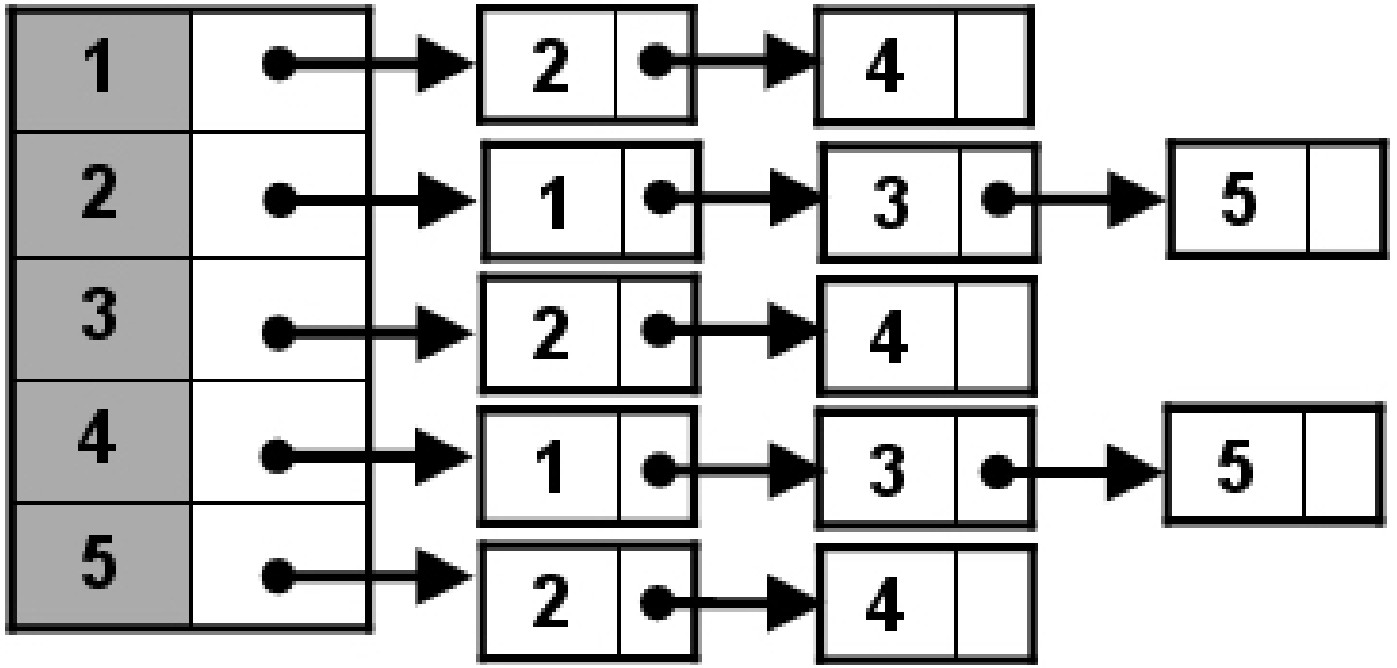


M	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	0	1
3	0	1	0	1	0
4	1	0	1	0	1
5	0	1	0	1	0

¿Cómo se representa?

Matriz de adyacencias: se asocia cada fila y cada columna a cada nodo del grafo, siendo los elementos de la matriz la relación entre los mismos, tomando los valores de 1 si existe la arista y 0 en caso contrario.





Lista de adyacencias

Lista de adyacencias: se asocia a cada nodo del grafo una lista que contenga todos aquellos nodos que sean adyacentes a él.

Algoritmos de búsqueda

Depth First Search (DFS)

Pseudocódigo:

Temporal: O(|V| + |E|)
Espacial: O(|V|)

procedure DFS(*G*,*v*):

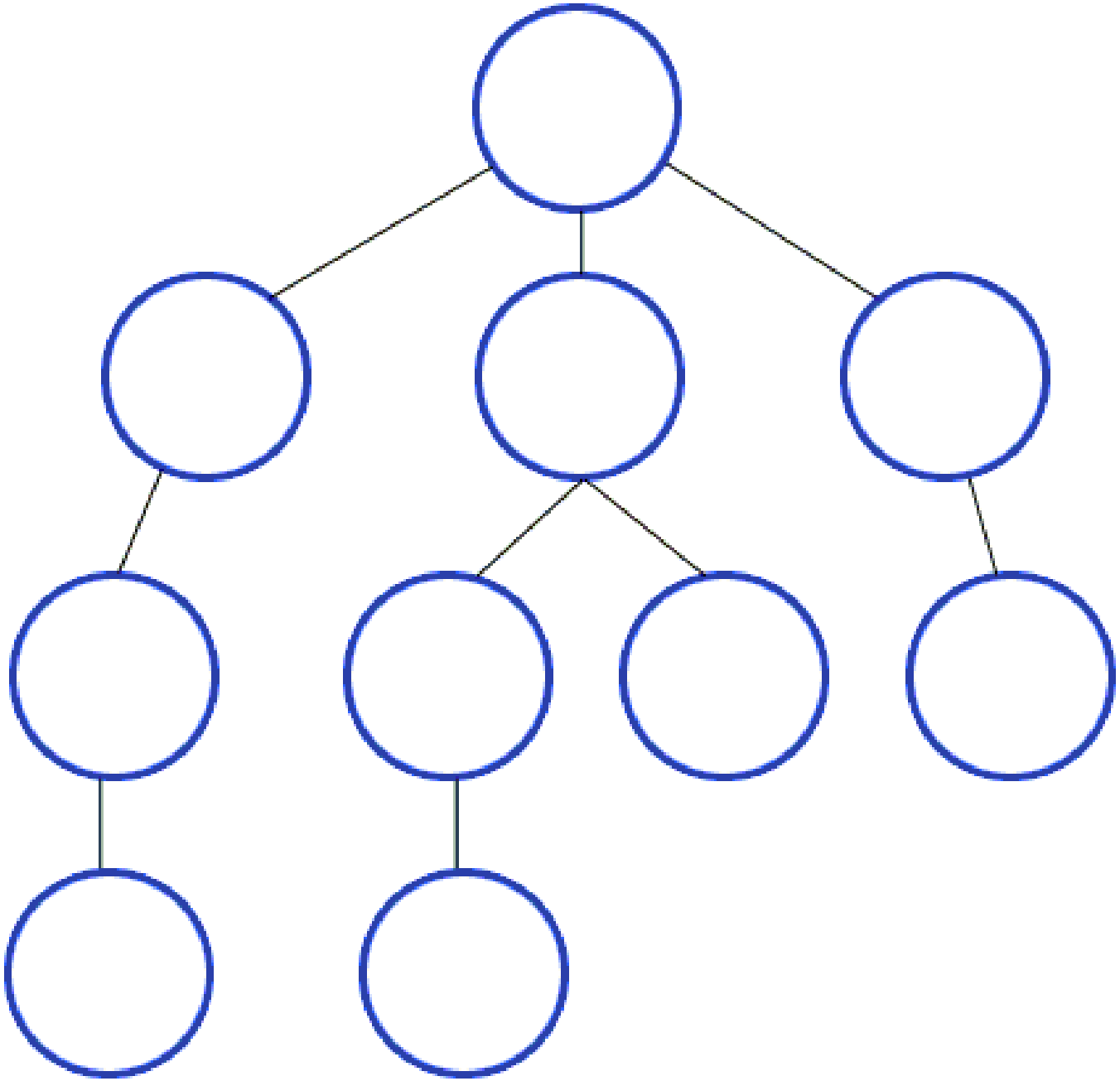
label *v* as discovered

for all directed edges from *v* to *w that are* **in** *G*.adjacentEdges(*v*) **do**

if vertex *w* is not labeled as discovered **then**

recursively call DFS(*G*,*w*)

Descripción gráfica



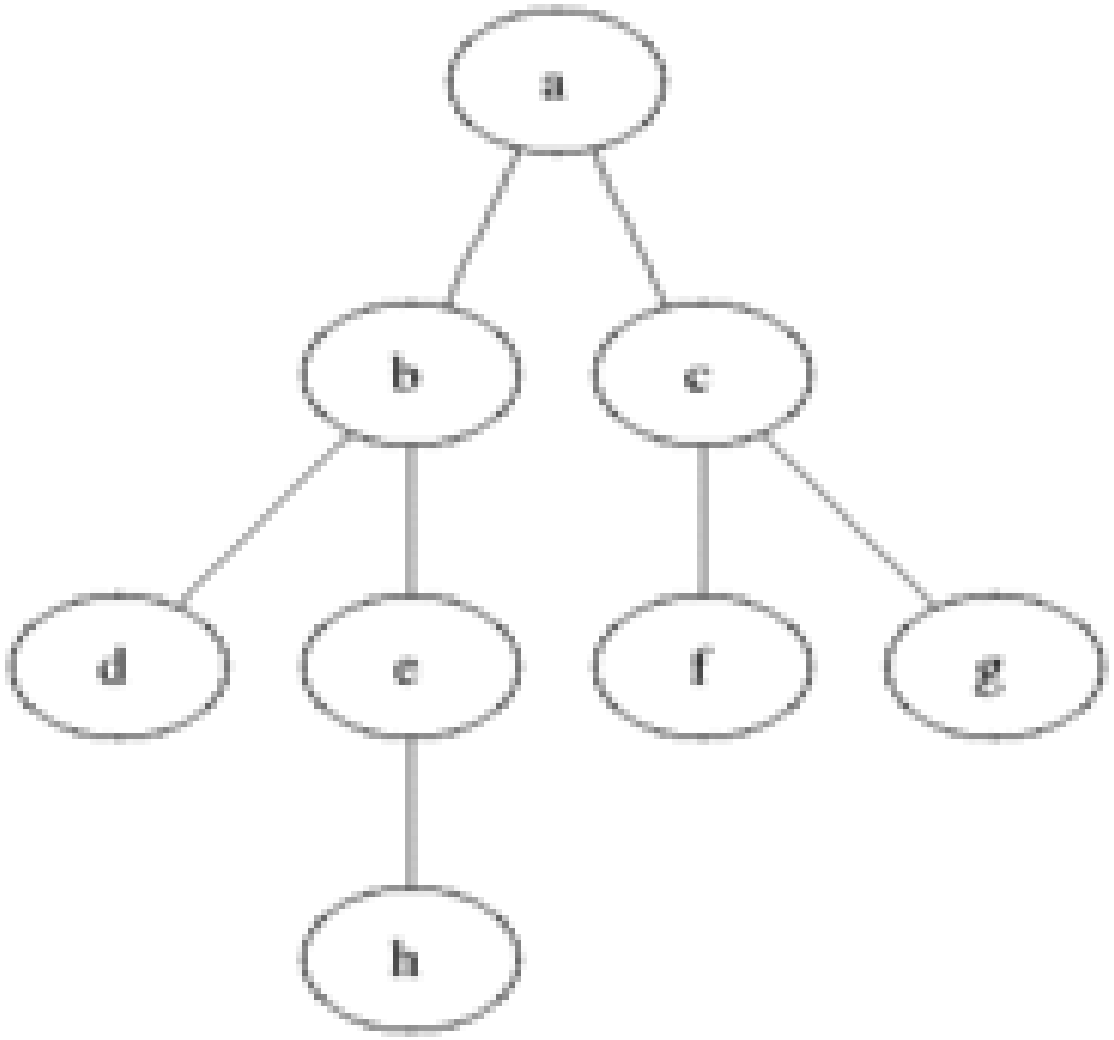
Aplicaciones

Clasificación topológica.
Encontrar los puentes de un gráfico.
Prueba de planaridad.
Resolver acertijos con una sola solución, como laberintos.
La generación de laberintos.
Encontrar biconnectividad en gráficos.

Breadth First Search (BFS)

```
procedure BFS(G,start_v):
    let Q be a queue
    label start_v as discovered
    Q.enqueue(start_v)
    while Q is not empty
        v = Q.dequeue()
        if v is the goal:
            return v
        for all edges from v to w in G.adjacentEdges(v) do
            if w is not labeled as discovered:
```

Temporal: O(|V| + |E|)
Espacial: O(|V|)



Descripción gráfica

Aplicaciones

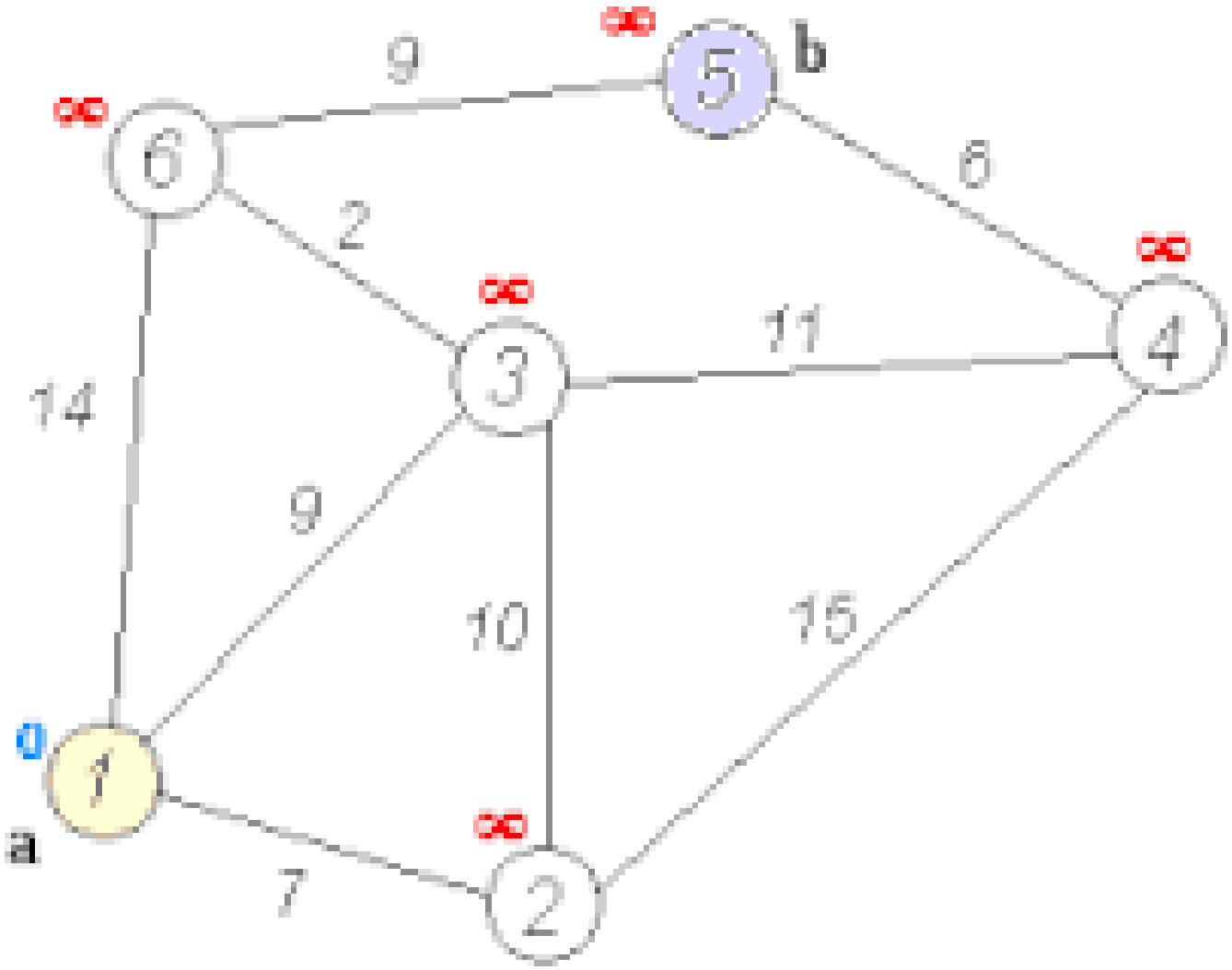
Encontrar la ruta más corta entre dos nodos u y v, con la longitud de la ruta medida por el número de bordes

La serialización / deserialización de un árbol binario frente a la serialización en orden ordenado permite que el árbol se reconstruya de manera eficiente.

Prueba de bipartidismo de un gráfico.

Dijkstra

Se trata de un algoritmo de búsqueda sobre grafos que soluciona el problema del camino mínimo (de ahí que también sea conocido como algoritmo de caminos mínimos) en grafos de aristas con pesos no negativos.

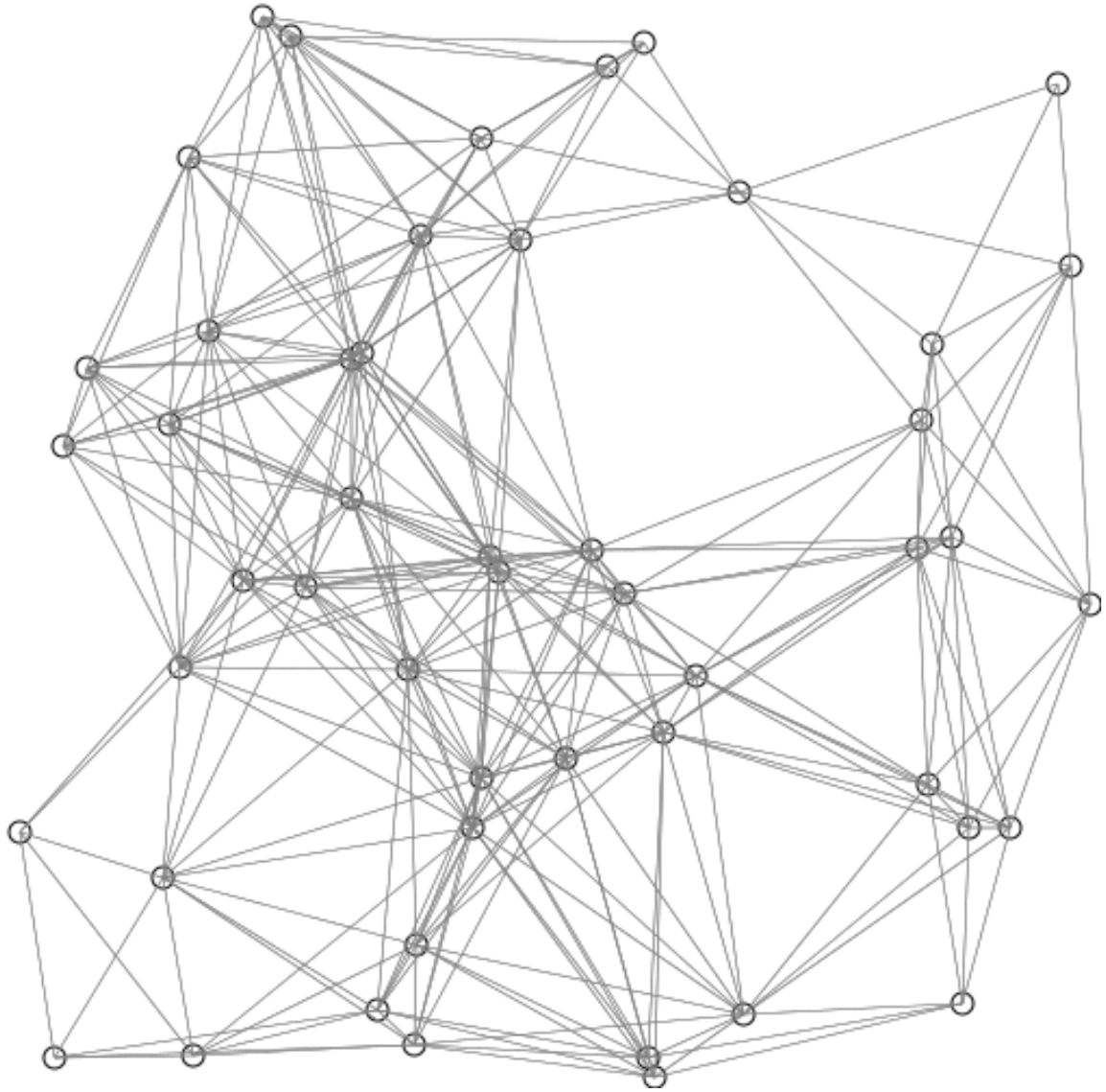


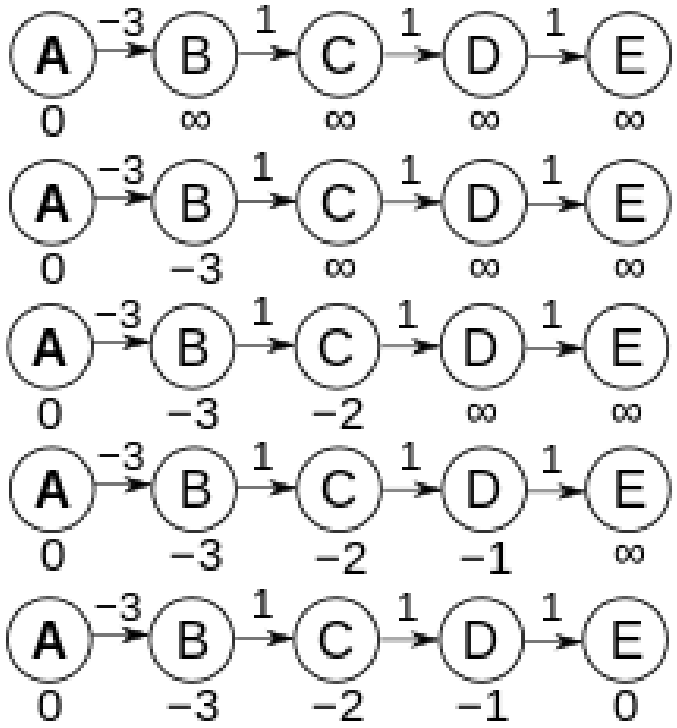
Pseudocódigo

```
function Dijkstra(Graph, source):

    create vertex set Q


    for each vertex v in Graph:

        dist[v] ← INFINITY

        prev[v] ← UNDEFINED

        add v to Q

        dist[source] ← 0


    while Q is not empty:

        u ← vertex in Q with min dist[u]
```

Temporal: O(|E| + |V|log|V|)

Descripción gráfica





Bellman Ford

El algoritmo de Bellman-Ford genera el camino más corto en un grafo dirigido ponderado (en el que el peso de alguna de las aristas puede ser negativo).

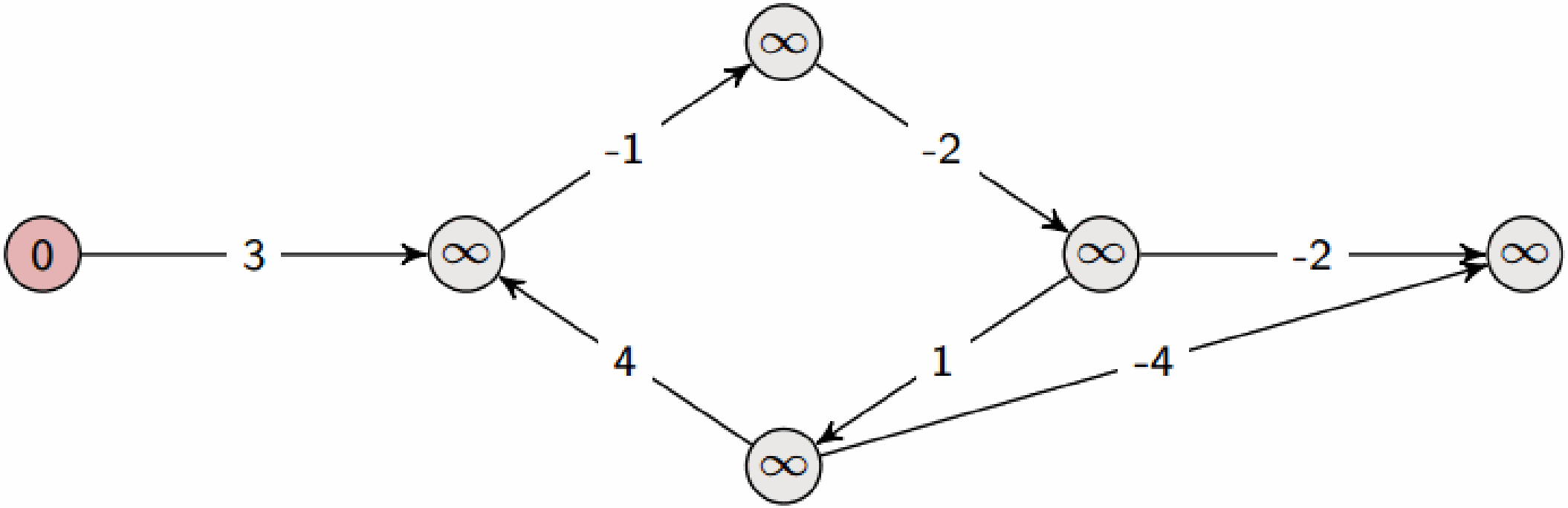
Pseudocódigo

Temporal: O(|V||E|)
Espacial: O(|V|)

```
function BellmanFord(list vertices, list edges, vertex source)
  ::distance[],predecessor[]

  // Step 1: initialize graph
  for each vertex v in vertices:
      distance[v] := inf             // Initialize the distance to all vertices to infinity
      predecessor[v] := null         // And having a null predecessor

  distance[source] := 0              // The distance from the source to itself is, of course, zero

  // Step 2: relax edges repeatedly
```

Pseudocódigo

```
// Step 3: check for negative-weight cycles
for each edge (u, v) with weight w in edges:
    if distance[u] + w < distance[v]:
        error "Graph contains a negative-weight cycle"

return distance[], predecessor[]
```



Descripción gráfica

Bibliografía

Tipo de dato y Representaciones:

https://es.wikipedia.org/wiki/Grafo_(tipo_de_dato_abstracto)

Algoritmos de búsqueda:

http://www.dma.fi.upm.es/personal/gregorio/grafos/web/iagraph/busqueda.html

DFS:

https://en.wikipedia.org/wiki/Depth-first_search#Pseudocode

BFS:

https://en.wikipedia.org/wiki/Breadth-first_search

Dijkstra:

https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm#Pseudocode