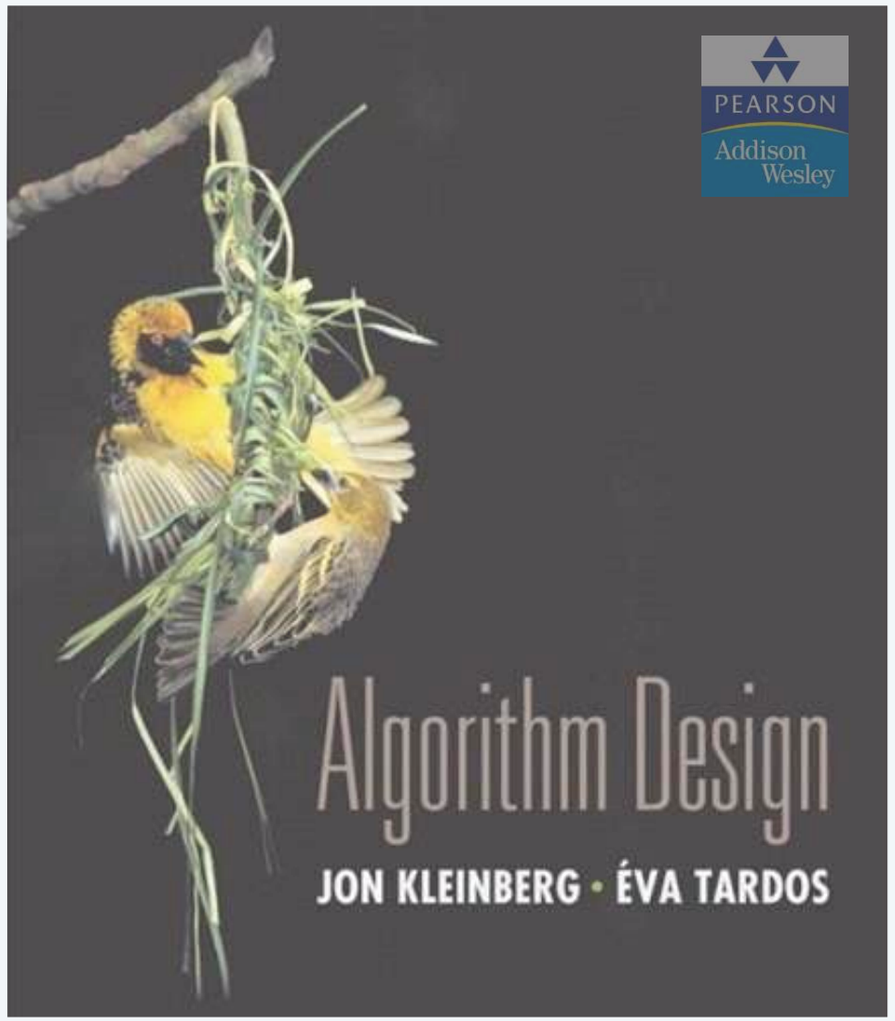


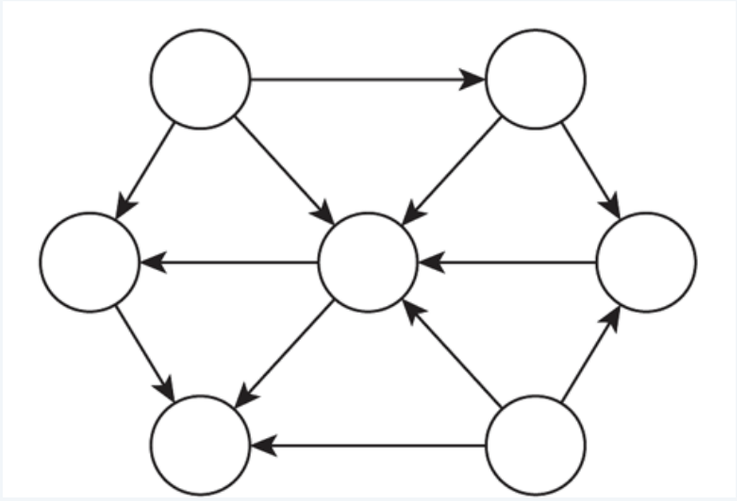
3. GRÁFICOS

- *definiciones básicas y aplicaciones*
- *conectividad y recorrido de grafos*
- *comprobación de la bipartición*
- *conectividad en grafos dirigidos*
- *DAGs y ordenación topológica*

Grafos dirigidos

Notación. $G = (V, E)$.

- La arista (u, v) sale del nodo u y entra en el nodo v.



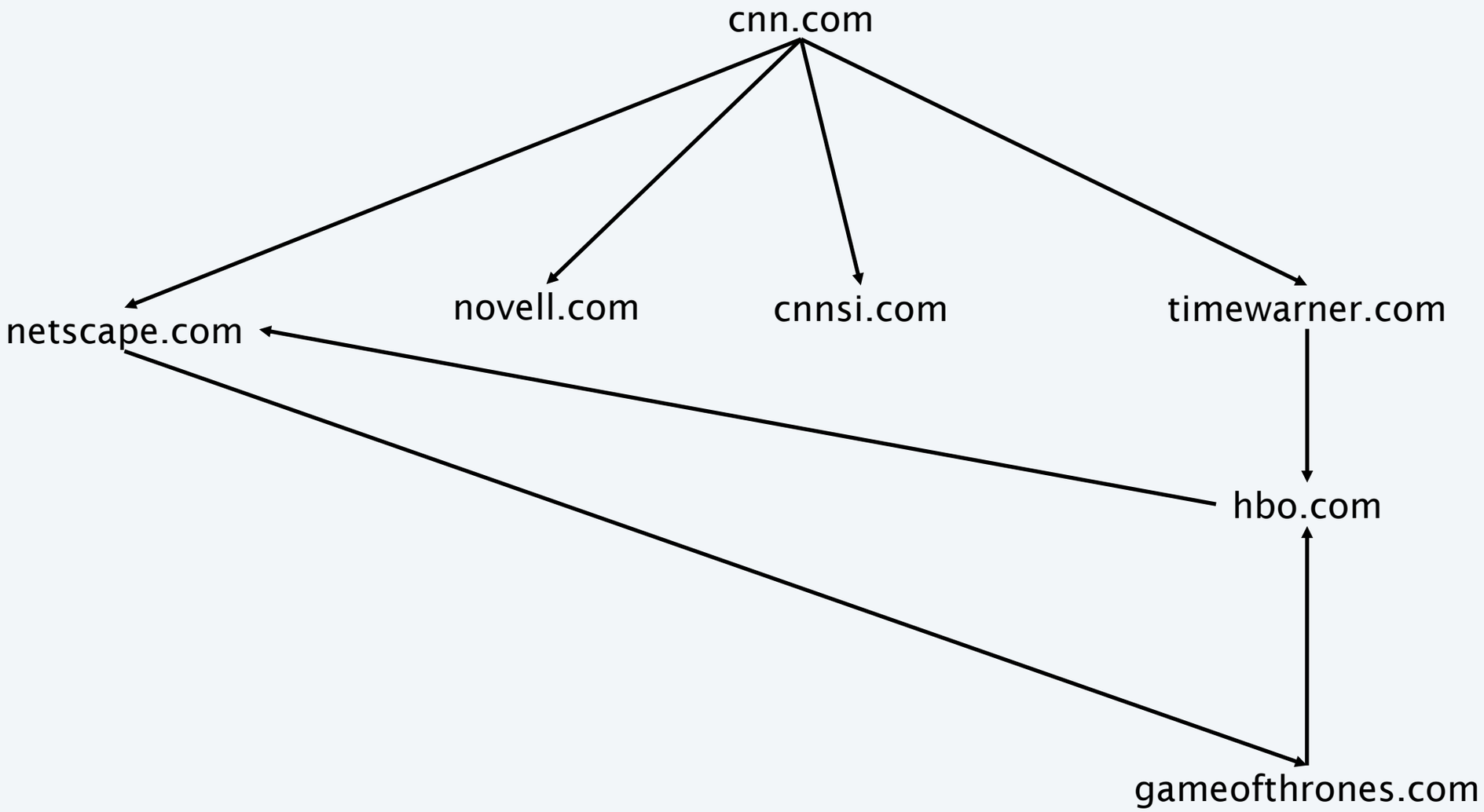
Ej. Gráfico web: puntos de hipervínculo de una página web a otra.

- La orientación de los bordes es crucial.
- Los buscadores web modernos aprovechan la estructura de hipervínculos para clasificar las páginas web según su importancia.

Red mundial

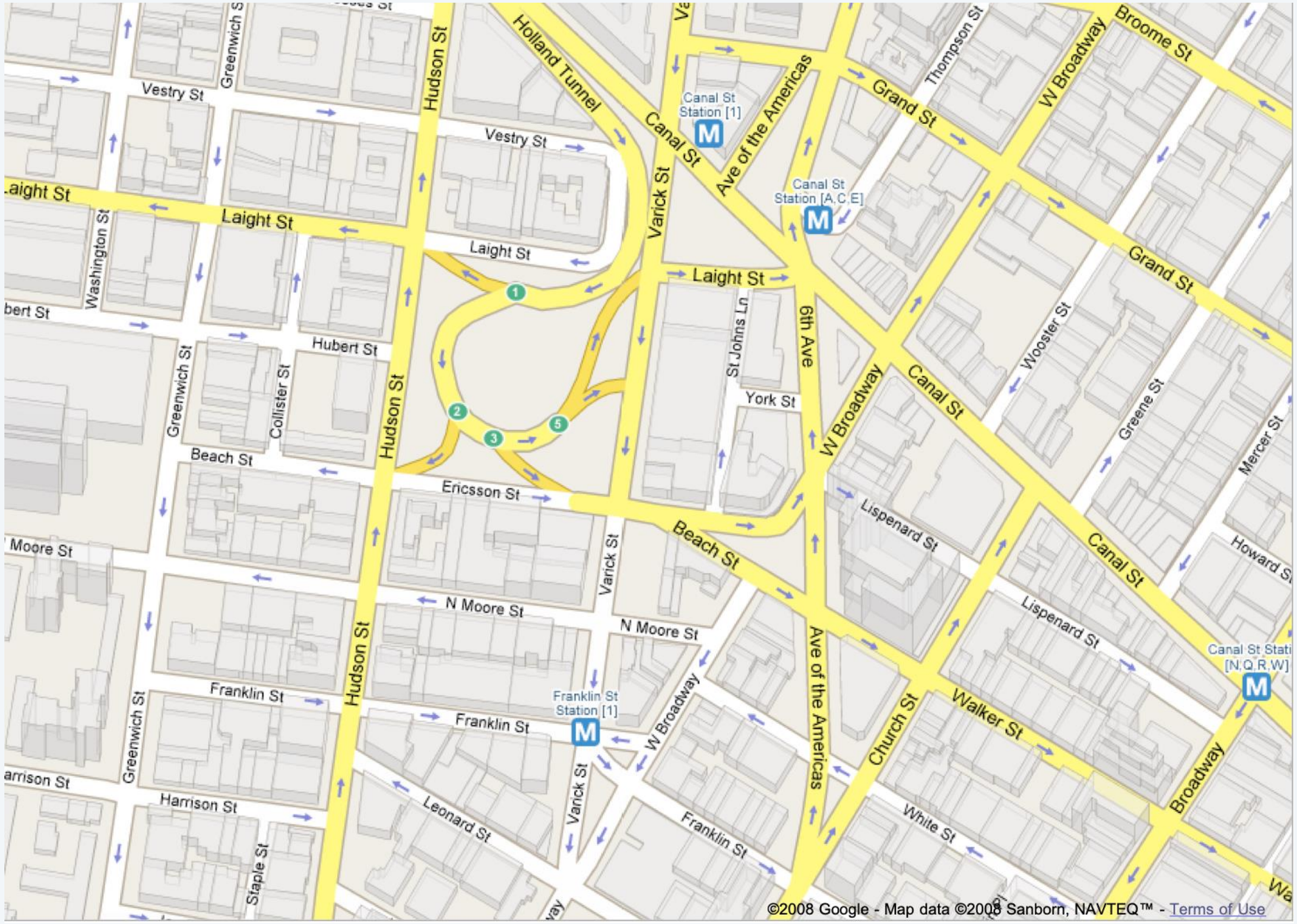
Gráfico web.

- Nodo: página web.
- Borde: hipervínculo de una página a otra (la orientación es crucial).
- Los motores de búsqueda modernos aprovechan la estructura de hipervínculos para clasificar las páginas web según su importancia.



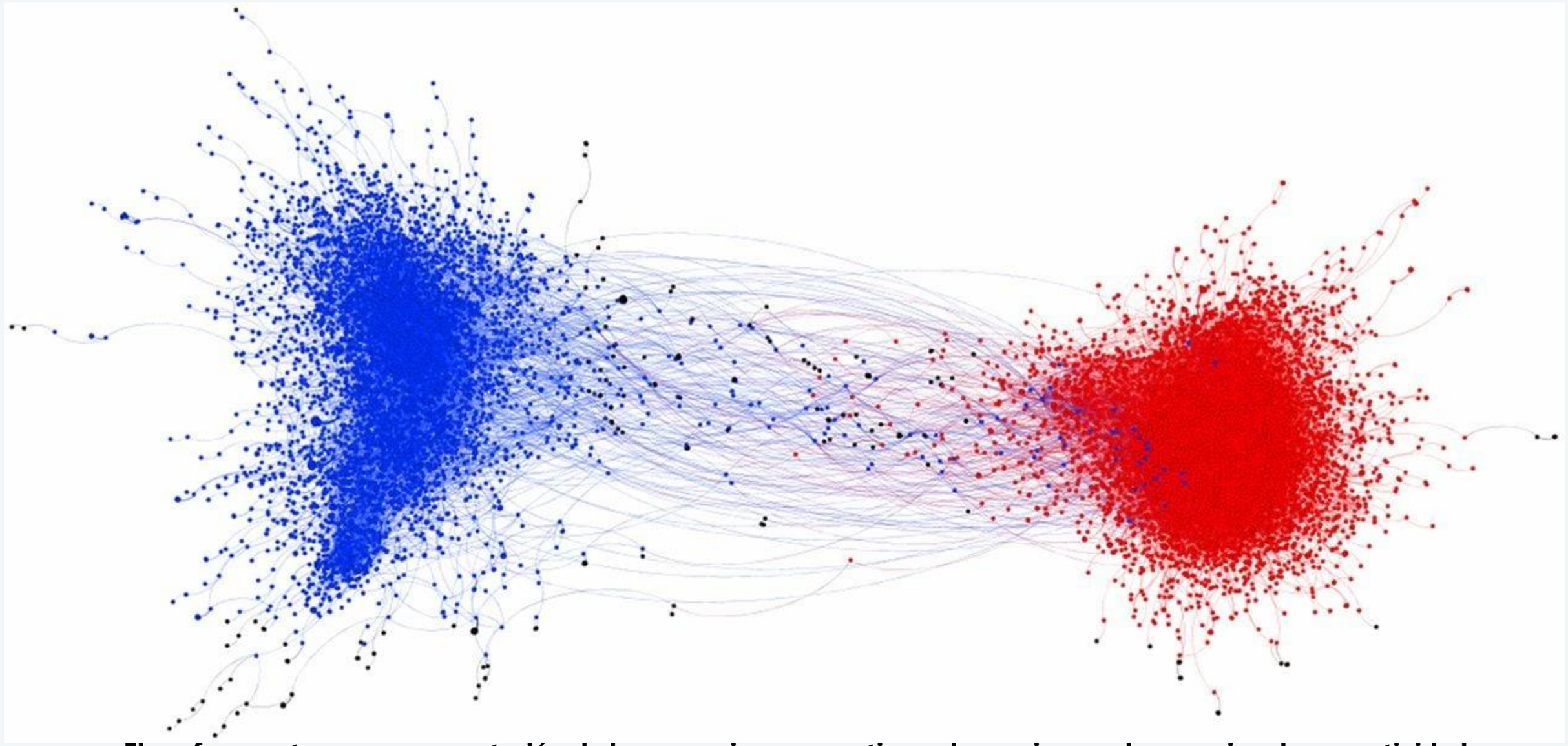
Red de carreteras

Nodo = intersección; arista = calle de sentido único.



Grafo de la blogosfera política

Nodo = usuario que tuitea; arista = retuit.

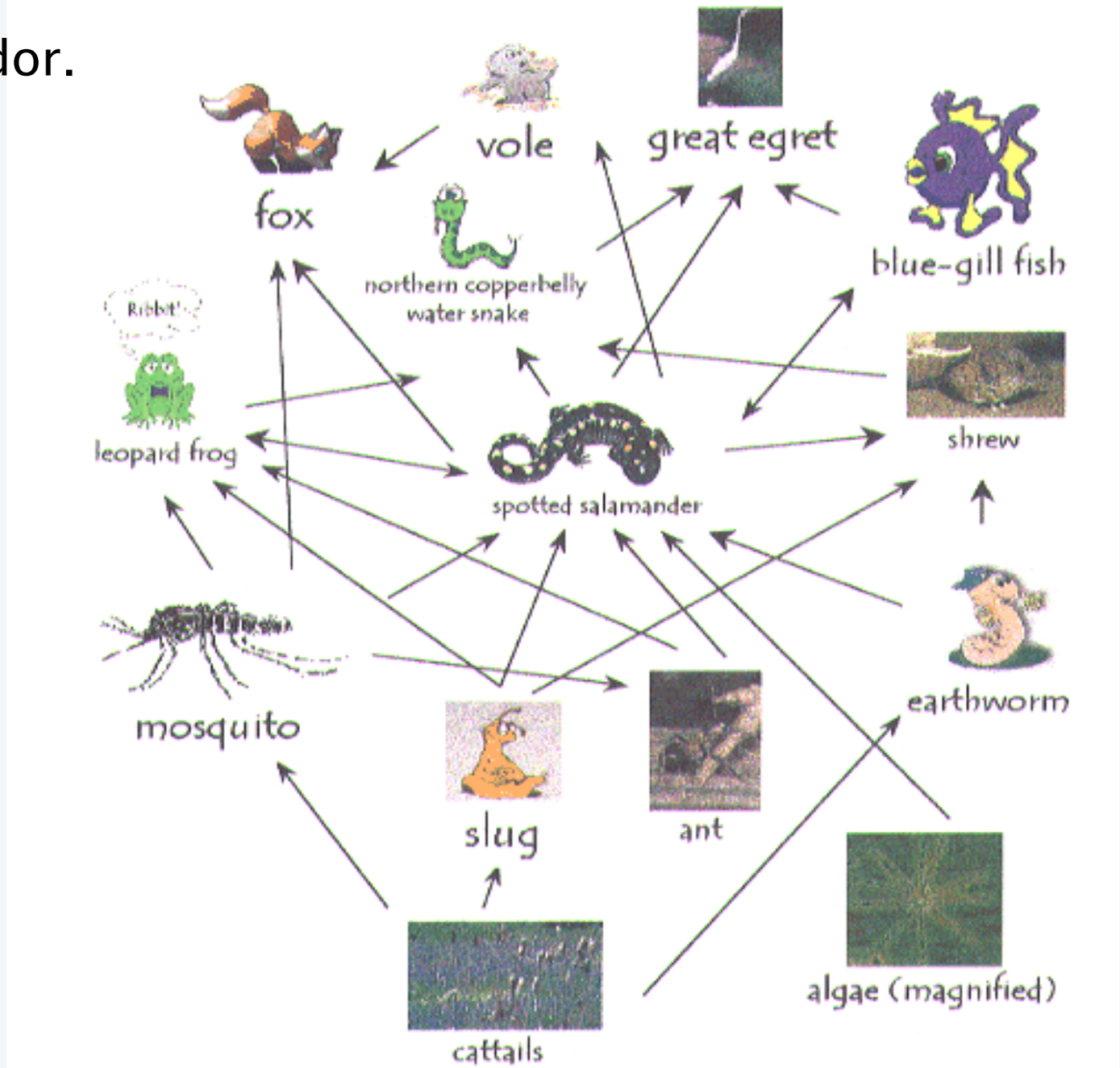


El grafo muestra una representación de los mensajes que contienen lenguaje moral y emocional, y su actividad de retuiteo, en relación con temas políticos (control de armas, matrimonio entre personas del mismo sexo, cambio climático). Los nodos representan a un usuario que envía un mensaje, y las aristas (líneas) representan a un usuario que retuitea a otro. Las dos grandes comunidades se sombrearon en función de la ideología media de cada una de ellas (el azul representa una media progresista y el rojo una media conservadora).

Red trófica ecológica

Grafo de la red alimentaria.

- Nodo = especie.
- Borde = de presa a depredador.



Referencia: http://www.twingroves.district96.k12.il.us/Wetlands/Salamander/SalGraphics/salfoodweb.giff

Algunas aplicaciones de los grafos dirigidos

grafo dirigido	nodo	borde dirigido
transporte	intersección de calles	calle de sentido único
web	página web	hipervínculo
red alimentaria	especies	relación depredador-presa
WordNet	sinónimo	hiperónimo
programación	tarea	restricción de precedencia
financiero	banco	transacción
teléfono móvil	persona	llamada realizada
enfermedades infecciosas	persona	infección
juego	posición en el consejo	movimiento legal
cita	artículo de revista	cita
grafo de objetos	objeto	puntero
jerarquía de herencia	clase	hereda de
flujo de control	bloque de código	saltar

Búsqueda en grafos

Alcanzabilidad dirigida. Dado un nodo s, encontrar todos los nodos alcanzables desde s.

Problema del camino más corto dirigido s-t. Dados dos nodos s y t ¿cuál es la longitud del camino más corto de s a t?

Búsqueda en grafos. BFS se extiende de forma natural a grafos dirigidos.

Rastreador (crawler) web. Empieza en la página web s. Encuentra todas las páginas web enlazadas desde s, directa o indirectamente.

Fuerte conectividad

Def. Los nodos u y v son mutuamente alcanzables si existe un camino de u a v y también un camino de v a u.

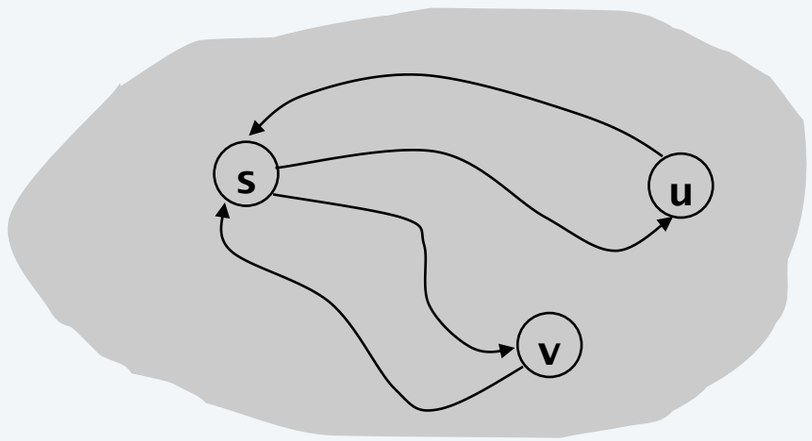
Def. Un grafo es fuertemente conexo si cada par de nodos es mutuamente alcanzable.

Lema. Sea s un nodo cualquiera. G es fuertemente conexo si todo nodo es alcanzable desde s, y s es alcanzable desde todo nodo.

Dem. $\Rightarrow$ Se deduce de la definición.

Dem. $\Leftarrow$ Camino de u a v: concatenar el camino $u \rightsquigarrow s$ con el camino $s \rightsquigarrow v$.

Camino de v a u: concatenar el camino $v \rightsquigarrow s$ con el camino $s \rightsquigarrow u$. ▪



ok si los caminos se solapan

Conectividad fuerte: algoritmo

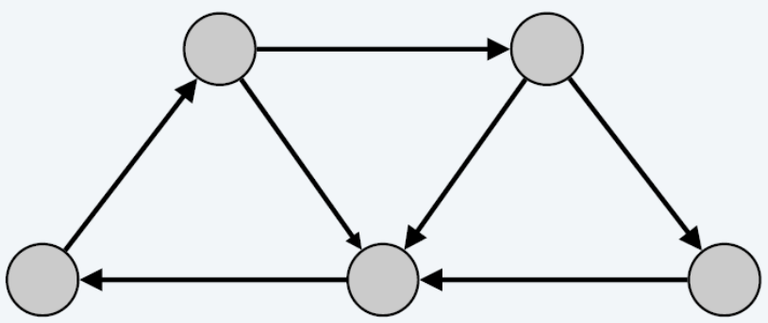
Teorema. Se puede determinar si G está fuertemente conectado en tiempo $O(m + n)$.

Dem.

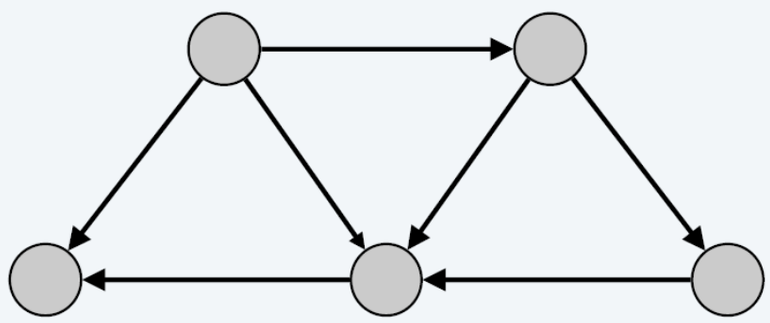
- Elige cualquier nodo s.
- Ejecuta BFS desde s en G.
- Ejecuta BFS desde s en $G^{reverse}$.
- Devuelve *true* si todos los nodos son alcanzados en ambos BFS.

orientación inversa de cada arista en G

- La corrección se deduce inmediatamente del lema anterior. ▪



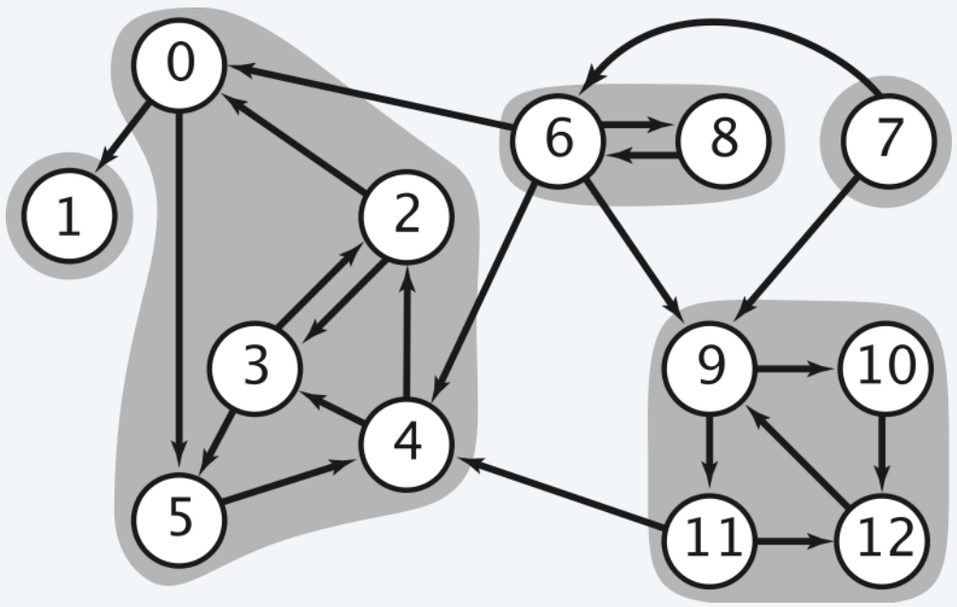
fuertemente conexo



no fuertemente conexo

Componentes fuertes

Def. Una componente fuertemente conexa es un subconjunto máximo de nodos mutuamente alcanzables.



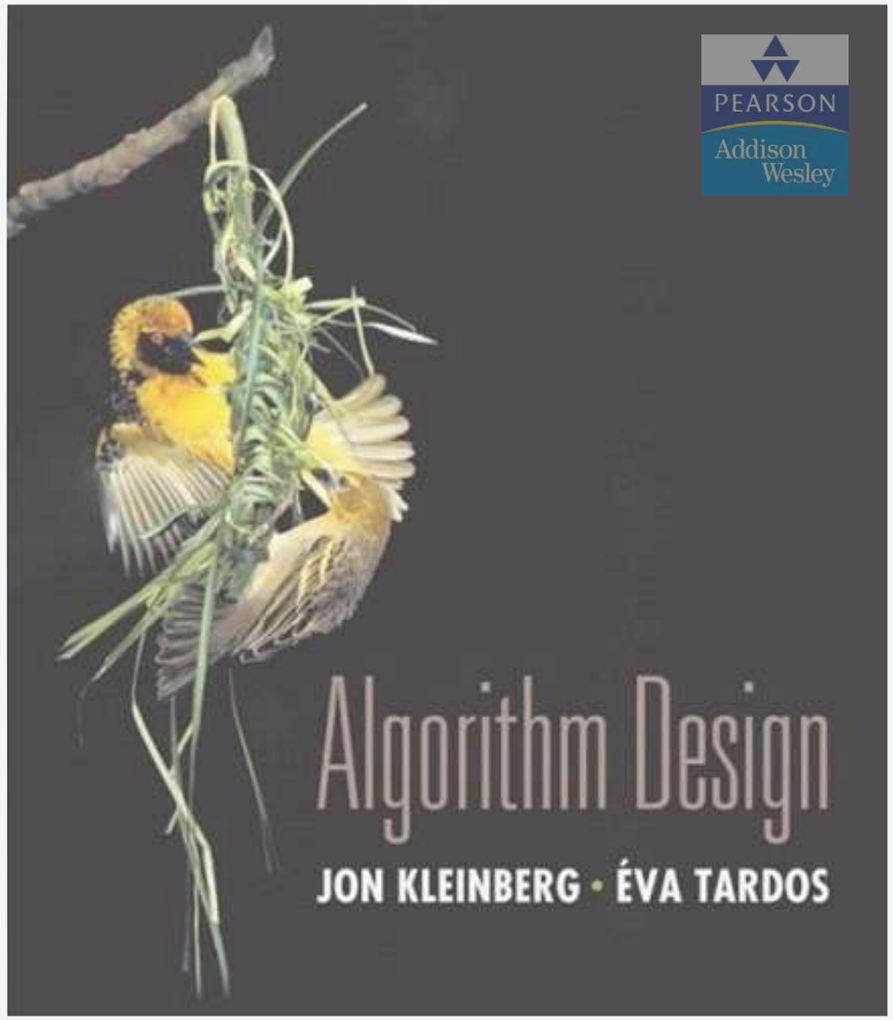
Teorema. [Tarjan 1972] Puede encontrar todos los componentes fuertes en tiempo $O(m + n)$.

SIAM J. COMPUT.
Vol. 1, No. 2, June 1972

DEPTH-FIRST SEARCH AND LINEAR GRAPH ALGORITHMS*

ROBERT TARJAN†

Abstract. The value of depth-first search or "backtracking" as a technique for solving problems is illustrated by two examples. An improved version of an algorithm for finding the strongly connected components of a directed graph and an algorithm for finding the biconnected components of an undirect graph are presented. The space and time requirements of both algorithms are bounded by $k_1V + k_2E + k_3$ for some constants k_1, k_2, and k_3, where V is the number of vertices and E is the number of edges of the graph being examined.



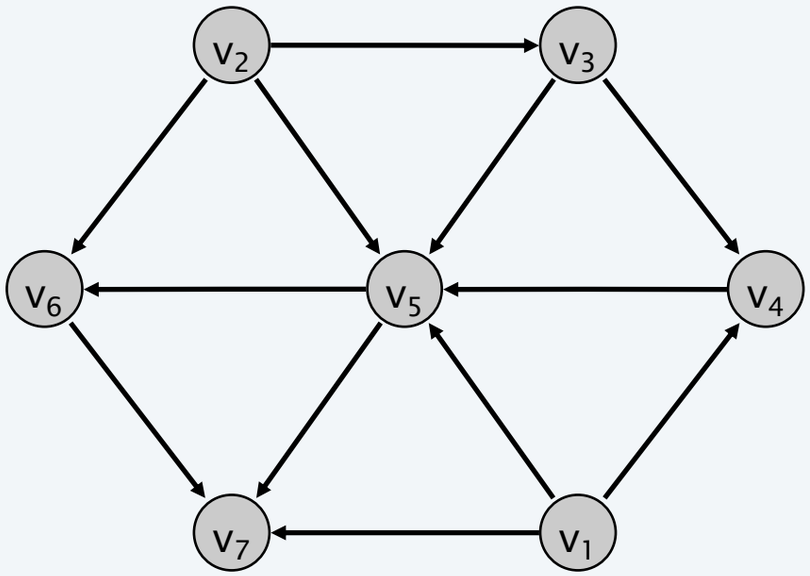
3. GRÁFICOS

- *definiciones básicas y aplicaciones*
- *conectividad de grafos y recorrido de grafos*
- *comprobación de la bipartición*
- *conectividad en grafos dirigidos*
- *DAGs y ordenación topológica*

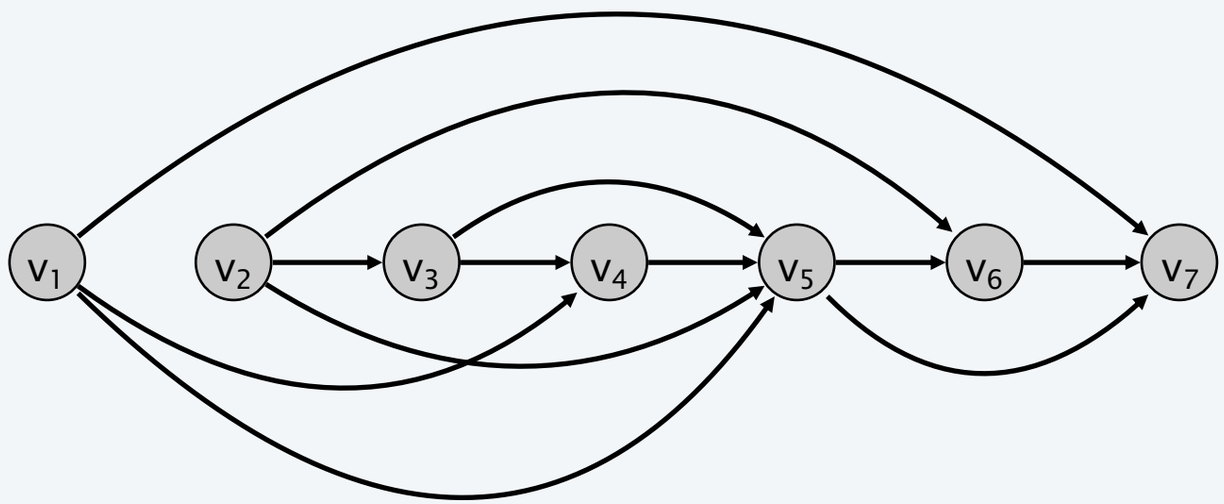
Grafos acíclicos dirigidos

Def. Un DAG es un grafo dirigido que no contiene ciclos dirigidos.

Def. Un orden topológico de un grafo dirigido $G = (V, E)$ es un ordenamiento de sus nodos como v_1 , v_2 , ..., v_n de manera que para cada arista (v_i , v_j) tenemos $i < j$.



a DAG



una ordenación topológica

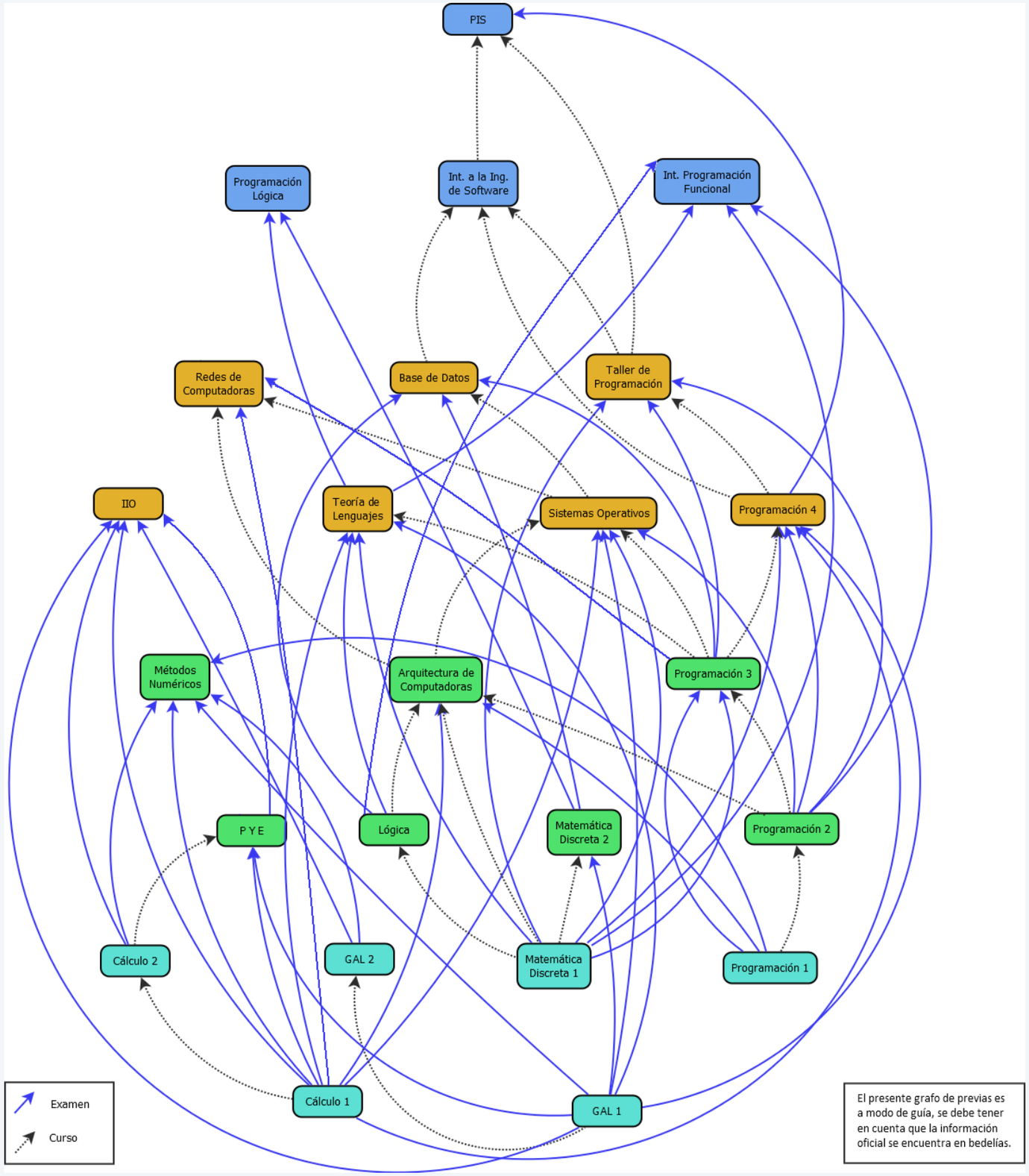
Restricciones de precedencia

Restricciones de precedencia.

La arista (v_i, v_j) significa que la tarea v_i debe producirse antes que v_j .

Aplicaciones.

- Grafo de previaturas: el curso v_i debe realizarse antes del v_j.
- Compilación: el módulo v_i debe compilarse antes que *el* v_j .
- Cadena de tareas de cálculo: se necesita la salida de la tarea v_i para determinar la entrada de la tarea v_j .



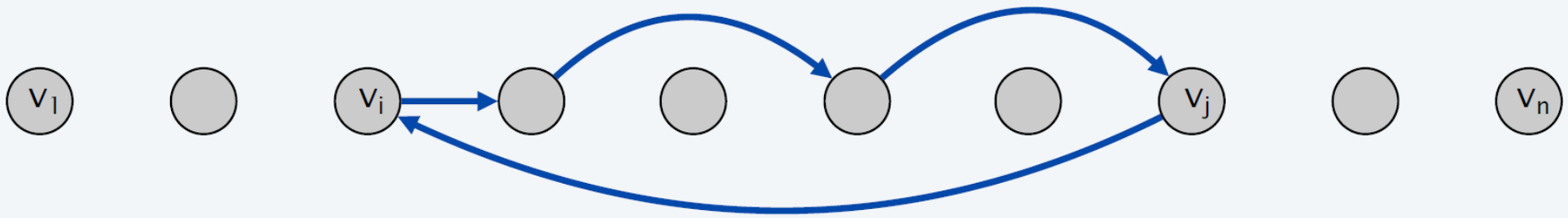
Grafos acíclicos dirigidos

Lemma. Si G tiene un orden topológico, entonces G es un DAG.

Dem. [por contradicción]

- Supongamos que G tiene un orden topológico v_1 , v_2 , ..., v_n y que G también tiene un ciclo dirigido C. Veamos qué ocurre.
- Sea v_i el nodo de menor índice en C, y sea v_j el nodo justo antes de v_i ; por tanto (v_j, v_i) es una arista.
- Por nuestra elección de i, tenemos $i < j$.
- Por otro lado, dado que (v_j, v_i) es una arista y v_1 , v_2 , ..., v_n es un orden topológico, debemos tener $j < i$, una contradicción. ▪

el ciclo dirigido C



el supuesto orden topológico: v_1 , ..., v_n

Grafos acíclicos dirigidos

Lemma. Si G tiene un orden topológico, entonces G es un DAG.

Q. ¿Tiene cada DAG un orden topológico?

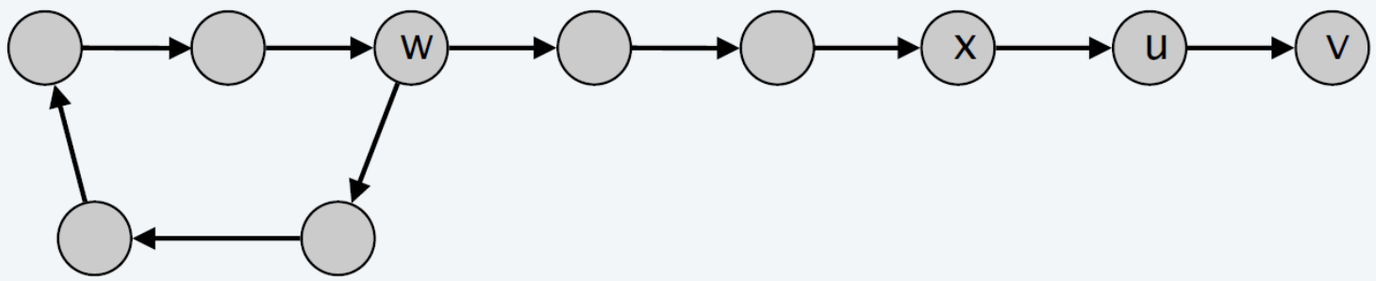
Q. En caso afirmativo, ¿cómo se calcula?

Grafos acíclicos dirigidos

Lemma. Si G es un DAG, entonces G tiene un nodo sin aristas entrantes.

Pf. [por contradicción]

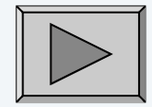
- Supongamos que G *es* un DAG y que cada nodo tiene al menos una arista entrante. Veamos qué ocurre.
- Escoja cualquier nodo v, y comience a seguir las aristas hacia atrás desde v. Dado que v tiene al menos una arista de entrada (u, v) podemos caminar hacia atrás hasta u.
- Entonces, como u tiene al menos una arista entrante (x, u), podemos caminar hacia atrás hasta x.
- Repítalo hasta que visitemos un nodo, digamos w, dos veces.
- Sea C la secuencia de nodos encontrados entre visitas sucesivas a w. C es un ciclo.



Grafos acíclicos dirigidos

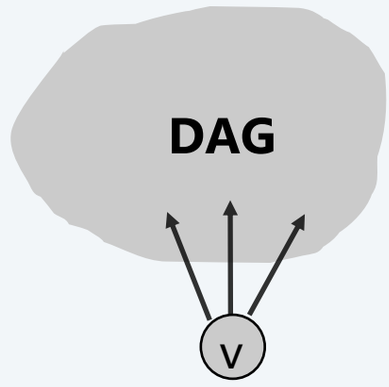
Lemma. Si G es un DAG, entonces G tiene un orden topológico.

Pf. [por inducción en n]



- Caso base: verdadero si $n = 1$.
- Dado un DAG sobre $n > 1$ nodos, encontrar un nodo v sin aristas entrantes.
- G - { v } es un DAG, ya que la eliminación de v no puede crear ciclos.
- Por hipótesis inductiva, G - { v } tiene un orden topológico.
- Coloca v primero en el orden topológico; luego añade nodos de G - { v }
- en orden topológico. Esto es válido ya que v no tiene aristas entrantes.

```
To compute a topological ordering of G:
Find a node v with no incoming edges and order it first
Delete v from G
Recursively compute a topological ordering of G−{v}
  and append this order after v
```



Algoritmo de ordenación topológica: tiempo de ejecución

Teorema. El algoritmo encuentra un orden topológico en tiempo $O(m + n)$.

Pf.

- Mantenga la siguiente información:
  - $count(w)$ = número restante de aristas entrantes
  - S = conjunto de nodos restantes sin aristas entrantes
- Inicialización: $O(m + n)$ mediante una sola exploración del gráfico.
- Actualización: para eliminar v
  - eliminar v de S
  - decrementa $count(w)$ para todas las aristas de v a w;
    y añade w a S si $count(w)$ llega a 0
  - esto es $O(1)$ por arista ▪