Bloque 1: Introducción

Pablo Rodríguez-Bocca
Departamento de IO, Instituto de Computación
prboccab@fing.edu.uy

4 de agosto de 2021

Bloque 1: Introducción

(I) Introducción ($\sim$ 3 clases):

- Utilidad del análisis de redes
  - análisis de datos, aprendizaje automático, datos en formato de redes, visualización, uso de las redes en distintas disciplinas
- La representación de grafos
  - nodos, enlace, matriz de adyacencia, grado de un nodo, redes de varios modos, conectividad ...
- [opcional] Introducción a la Inferencia estadística
- Prácticos 0-A, y 0-B (opcional)

Teoría de grafos (revisión)

Pablo Rodríguez-Bocca
Departamento de IO, Instituto de Computación
prboccab@fing.edu.uy

4 de agosto de 2021

Definiciones y conceptos básicos

Definiciones y conceptos básicos

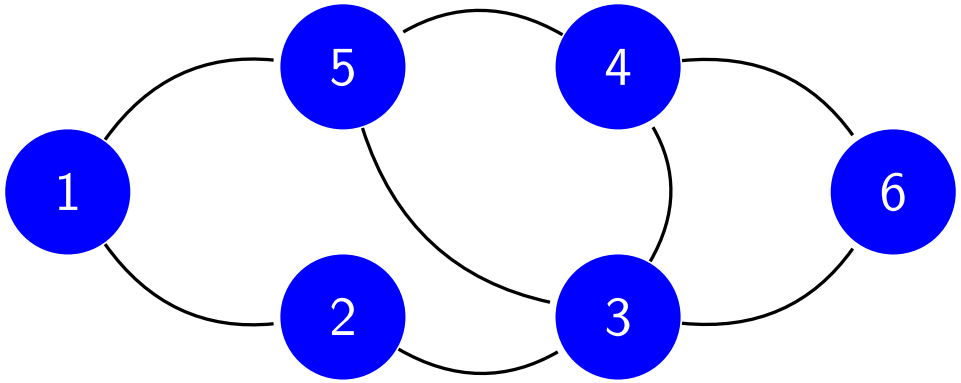
Movimientos en un grafo y conectividad

Familias de grafos

Álgebra de grafos

Grafos: estructuras de datos y algoritmos (opcional)

Grafos



- Grafo $G(V, E)$ $\Rightarrow$ Un conjunto V de vértices
  $\Rightarrow$ Conectados por un conjunto E de aristas
  $\Rightarrow$ Elementos de E son pares no-ordenados (u, v), $u, v \in V$
- En la Figura $\Rightarrow$ Vértices son $V = \{1, 2, 3, 4, 5, 6\}$
  $\Rightarrow$ Aristas $E = \{(1, 2), (1, 5), (2, 3), (3, 4), ...$
  $(3, 5), (3, 6), (4, 5), (4, 6)\}$
- Diremos que el grafo G tiene orden $N_v := |V|$, y tamaño $N_e := |E|$

De las redes a los grafos

- Redes son sistemas complejos de componentes interconectados
- Grafos son una representación matemática de esos sistemas

  $\Rightarrow$ Lenguaje formal al hablar de redes

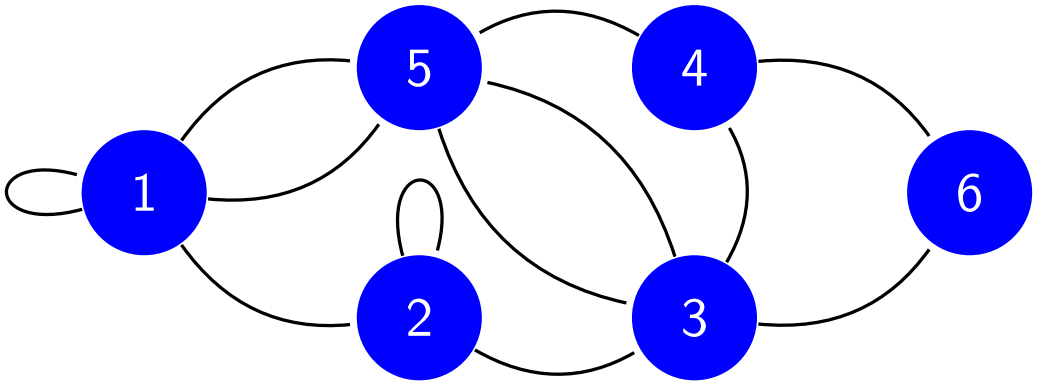
Sistemas	**Componentes**	**Interconexiones**
graphs $G(V, E)$	vertices V	edges E
computer science	nodes	links
physics	sites	bonds
sociology	actors	ties
$\vdots$	$\vdots$	$\vdots$

Vértices y aristas en redes

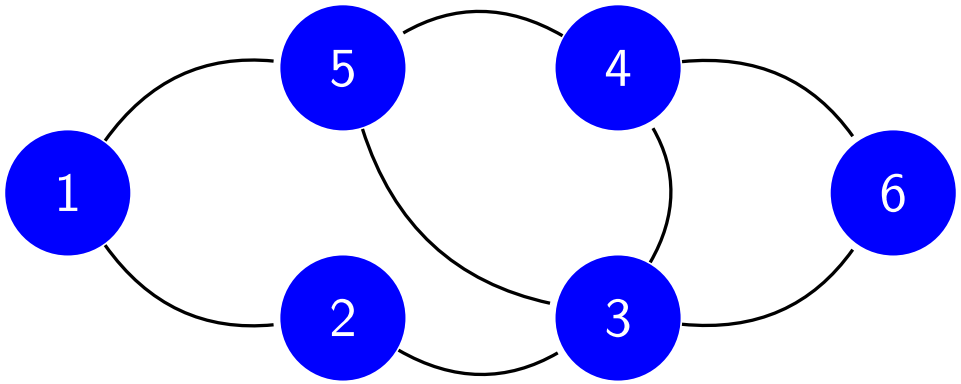
Network	**Vertex**	**Edge**
Internet	Computer/router	Cable or wireless link
Metabolic network	Metabolite	Metabolic reaction
WWW	Web page	Hyperlink
Food web	Species	Predation
Gene-regulatory network	Gene	Regulation of expression
Friendship network	Person	Friendship or acquaintance
Power grid	Substation	Transmission line
Affiliation network	Person and club	Membership
Protein interaction	Protein	Physical interaction
Citation network	Article/patent	Citation
Neural network	Neuron	Synapse
⋮	⋮	⋮

Grafos simple y multi-grafos

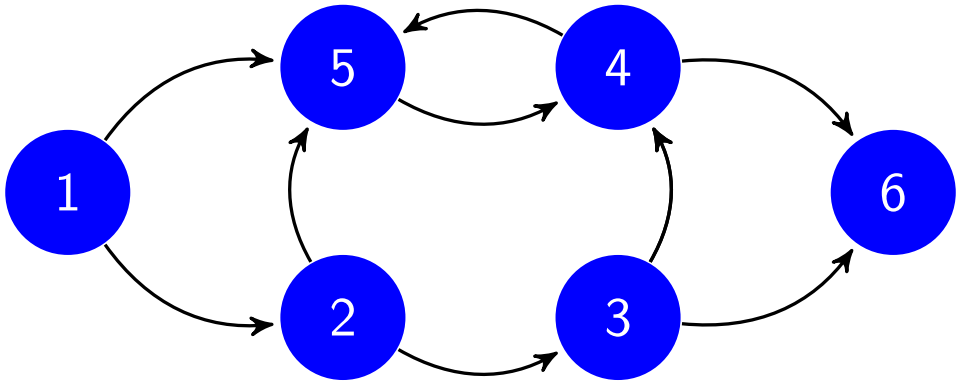
- En general, los grafos pueden tener bucles (*self-loops*) y multi-aristas (*multi-edges*)
  - $\Rightarrow$ Un grafo con alguno de ellos se llama multi-grafo



- Mayormente, trabajaremos con grafos simples, sin bucles ni multi-aristas



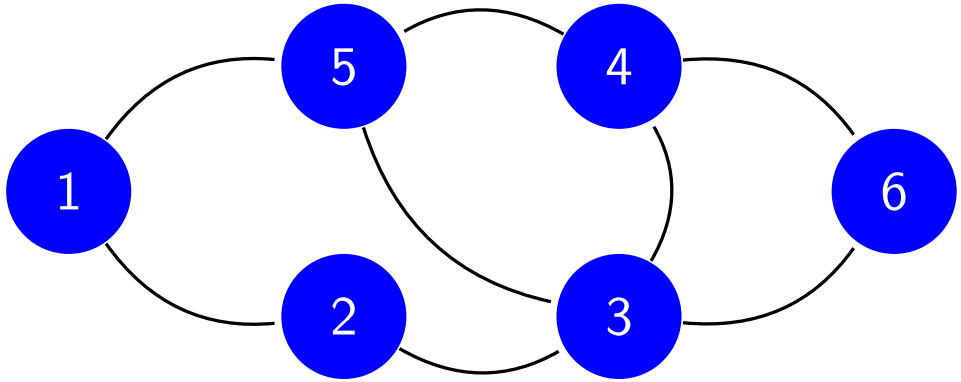
Grafos dirigidos



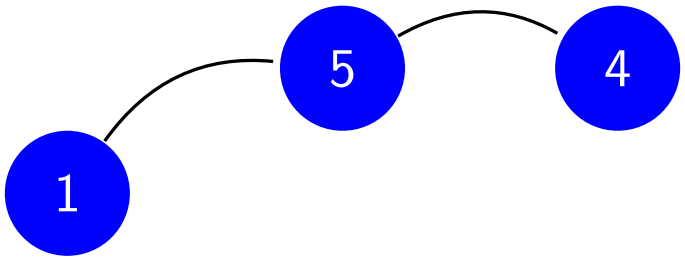
- En los grafos dirigidos, los elementos de E son pares ordenados (u, v), $u, v \in V$
  - $\Rightarrow$ Significa (u, v) distinto de (v, u)
  - $\Rightarrow$ Aristas dirigidas son llamadas arcos (*arcs*)
- Grafos dirigidos son llamados digrafos (*digraphs*)
  - $\Rightarrow$ Por convención, el arco (u, v) apunta a v
  - $\Rightarrow$ Si $\{(u, v), (v, u)\} \subseteq E$, los arcos se llaman mutuos (*mutual*)
- Ej: quién llama a quién, seguidores de Twitter

Subgrafos

- Considere el grafo $G(V, E)$



- **Def:** Grafo $G'(V', E')$ es un subgrafo inducido de G si $V' \subseteq V$ y $E' \subseteq E$ es la colección de aristas en G entre ese subconjunto de vértices
- Ej: Grafo inducido por $V' = \{1, 4, 5\}$

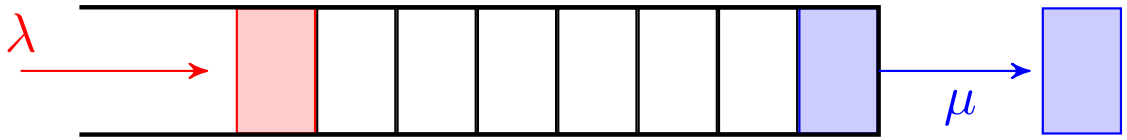


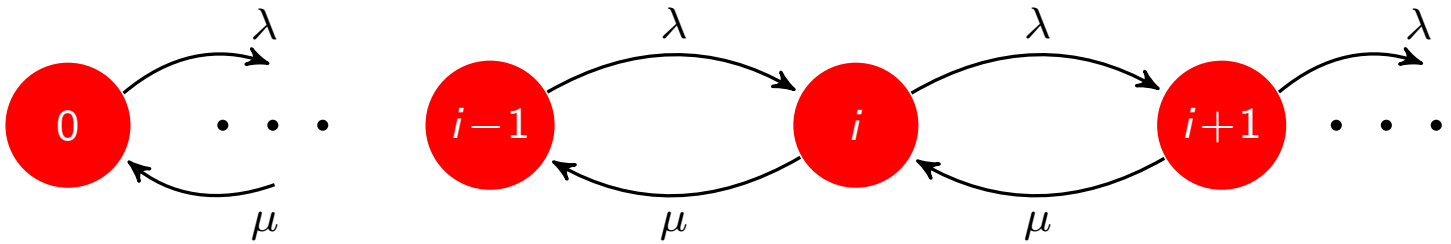
Grafos ponderados

- A veces pueden etiquetarse los vértices, las aristas, o ambos con valores numéricos

  $\Rightarrow$ Estos grafos tienen peso, *weighted graphs*

- Útil en el modelado
- Ej: diagramas de transición de Markov, sistema de filas de espera de un servidor (M/M/1)





- Las etiquetas pueden corresponder a medidas de procesos en la red

  Ej: Nodo infectado o no con influenza, tráfico IP de un enlace

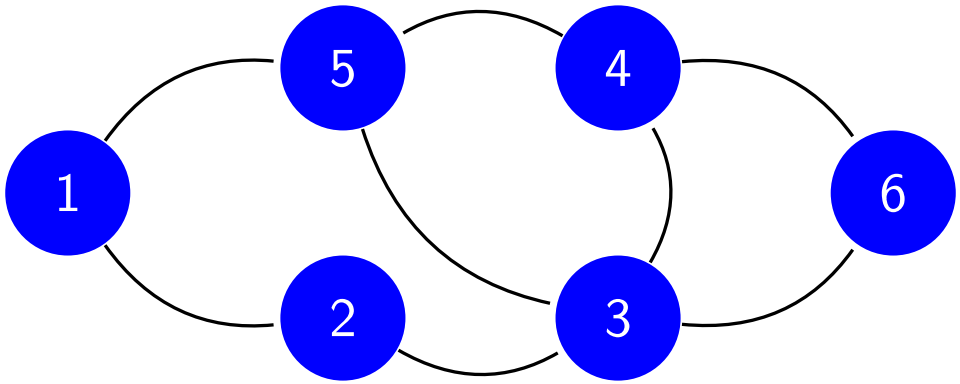
Representaciones típicas de redes

Network	**Graph representation**
WWW	Directed multi-graph (with loops), unweighted
Facebook friendships	Undirected, unweighted
Citation network	Directed, unweighted, acyclic
Collaboration network	Undirected, unweighted
Mobile phone calls	Directed, weighted
Protein interaction	Undirected multi-graph (with loops), unweighted
⋮	⋮

- Notar que las multi-aristas son usualmente representadas por pesos en las aristas (cantidad)

Adyacencia

- Adyacencia: noción simple y local de la conectividad que tiene un grafo
  - Vértices $u, v \in V$ son adyacentes $\Leftrightarrow$ enlazados por una arista en E
  - Aristas $e_1, e_2 \in E$ son adyacentes $\Leftrightarrow$ comparten un extremo en V

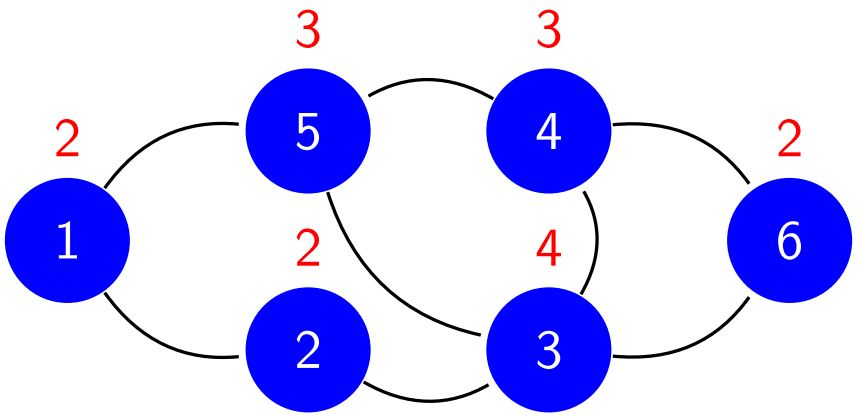


- En la Figura $\Rightarrow$ Vértices 1 y 5 son adyacentes; 2 y 4 no lo son
  $\Rightarrow$ La arista $(1, 2)$ es adyacente a $(1, 5)$ pero no a $(4, 6)$

Grado

- La arista (u, v) es incidente a los vértices u y v (no dirigido)
- **Def:** El grado d_v del vértice v es el número de aristas incidentes

  $\Rightarrow$ Secuencia de grado: vector de grados del grafo en orden creciente



- En la Figura $\Rightarrow$ El grado de los vértices en rojo, ej., $d_1 = 2$ y $d_5 = 3$

  $\Rightarrow$ Secuencia de grado del grafo: 2,2,2,3,3,4
- Vértices con alto grado son centrales, influyentes, prominentes.

  Próximamente más . . .

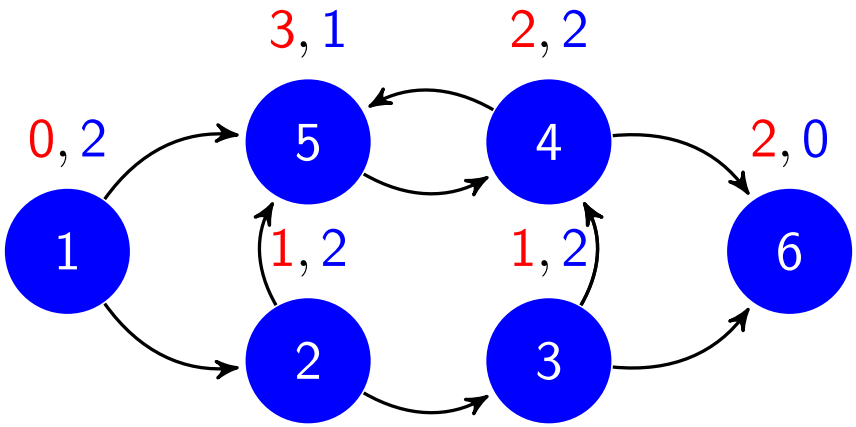
Grado: propiedades y observaciones

- Valores de grados entre 0 y $N_v - 1$
- La suma de todos los grados es el doble del tamaño del grafo

$$\sum_{v=1}^{N_v} d_v = 2|E| = 2N_e$$

$\Rightarrow$ La cantidad de vértices con grado impar es par

- En digrafos, se define el grado entrante d_v^{in} y saliente d_v^{out}



- en la Figura $\Rightarrow$ Grado de los vértices entrante en rojo y saliente en azul
  $\Rightarrow$ Ej., $d_1^{in} = 0, d_1^{out} = 2$ y $d_5^{in} = 3, d_5^{out} = 1$

Grado: medidas de dispersión

- **Def:** Grado medio, $\bar{d} := \frac{1}{N_v} \sum_{v=1}^{N_v} d_v$

  $\Rightarrow \bar{d} = \frac{2N_e}{N_v}$

- **Def:** Densidad del grafo, (*graph density*) $\rho := \frac{N_e}{\binom{N_v}{2}} = \frac{\bar{d}}{N_v - 1}$

- La mayoría de las redes reales son dispersas, es decir

$$N_e \ll \frac{N_v(N_v-1)}{2} \Leftrightarrow \bar{d} \ll N_v - 1 \Leftrightarrow \rho \ll 1$$

Ejemplo de redes reales (todas dispersas), por Leskovec et al '09

Network dataset	**Order** N_v	**Avg. degree** $\bar{d}$	**density** ρ
WWW (Stanford-Berkeley)	319,717	9,65	$3{,}0 \times 10^{-5}$
Social network (LinkedIn)	6,946,668	8,87	$1{,}3 \times 10^{-6}$
Communication (MSN IM)	242,720,596	11,1	$4{,}6 \times 10^{-8}$
Collaboration (DBLP)	317,080	6,62	$2{,}1 \times 10^{-5}$
Roads (California)	1,957,027	2,82	$1{,}4 \times 10^{-6}$
Proteins (S. Cerevisiae)	1,870	2,39	$1{,}3 \times 10^{-3}$

Movimientos en un grafo y conectividad

Definiciones y conceptos básicos

Movimientos en un grafo y conectividad

Familias de grafos

Álgebra de grafos

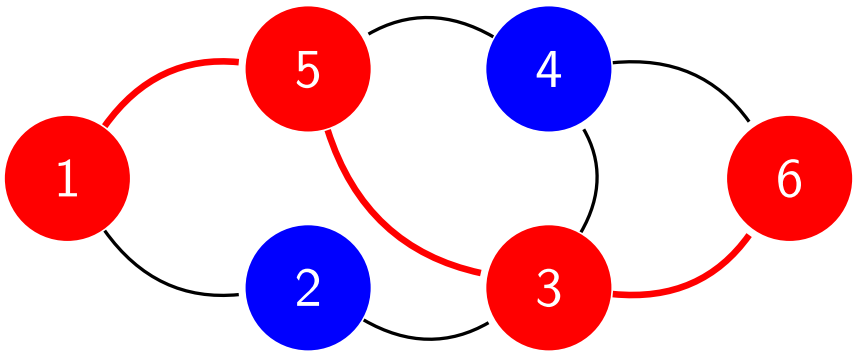
Grafos: estructuras de datos y algoritmos (opcional)

Movimientos en un grafo

- **Def:** Un camino (*walk*) de v_0 a v_l es una secuencia alternada:

$$\{v_0, e_1, v_1, \ldots, v_{l-1}, e_l, v_l\}, \text{ donde } e_i \text{ es incidente a } v_{i-1} \text{ y } v_i$$

- Un sendero (*trail*) es un camino que no repite aristas
- Un camino simple (*path*) es un camino que no repite vértices (entonces, también un sendero)



- Un camino o sendero es cerrado cuando $v_0 = v_l$. Un sendero cerrado es un circuito (*circuit*)
- Un ciclo (*cycle*) es un camino cerrado que no repite vértices excepto $v_0 = v_l$
- Estos conceptos se generalizan naturalmente a digrafos

Movimientos en un grafo: caminos particulares

- **Def:** Camino euleriano: camino que usa cada arista exactamente una vez

  $\Rightarrow$ puede no existir o haber varios

  $\Rightarrow$ no tiene que ser un camino simple

  $\Rightarrow$ si existe $\Rightarrow$ la cantidad de vértices con grado impar debe ser cero o dos (para un ciclo debe ser cero)

  $\Rightarrow$ encontrarlo *formaliza el problema de los siete puentes de Königsberg (Euler)*

- **Def:** Camino hamiltoniano: camino que visita cada vértice exactamente una vez

  $\Rightarrow$ puede no existir o haber varios

  $\Rightarrow$ es un camino simple

Distancias en un grafo

- **Def:** El largo del camino $\{v_0, e_1, v_1, \ldots, v_{l-1}, e_l, v_l\}$, es la cantidad de aristas

  $\Rightarrow$ Si la aristas estan ponderadas $\{w_e\}$, habitualmente se considera el largo del camino como $w_{e_1} + \ldots + w_{e_l}$

- **Def:** La distancia entre los vértices u y v es el largo del camino $u - v$ más corto. Llamada distancia geodésica (*geodesic distance*)

  $\Rightarrow$ Si no hay camino $u - v$, la distancia es ∞

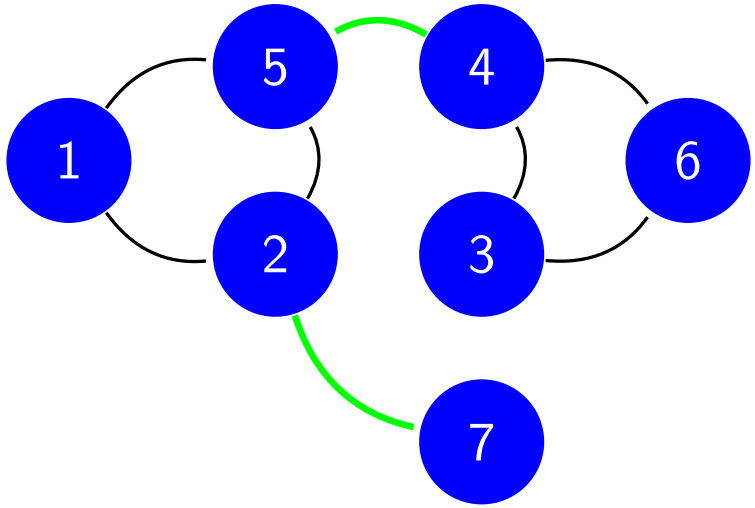
  $\Rightarrow$ Es el menor r que cumple $[A^r]_{uv} > 0$ (ver álgebra de grafos...)

  $\Rightarrow$ Camino más corto
  - no es necesariamente único
  - es un camino simple (no se intercepta a si mismo)

- **Def:** El diámetro de un grafo es la distancia más larga entre todo par de nodos conectados

Conectividad

- el vértice v es alcanzable desde $u \Leftrightarrow$ existe un camino $u - v$
- **Def:** Un grafo es conexo $\Leftrightarrow$ todo vértice es alcanzable desde cualquier otro

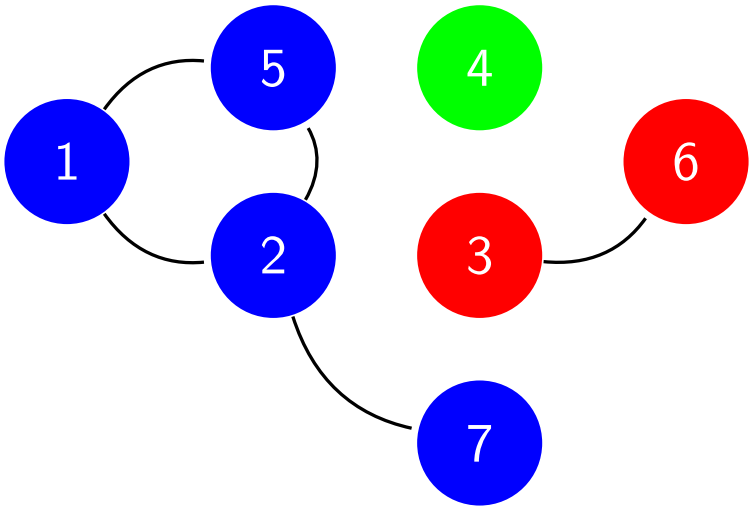


- Si las aristas puentes son removidas, el grafo se transforma en disconexo

Componentes conexas

- **Def:** Una componente es un subgrafo conexo maximal

  $\Rightarrow$ Maximal significa que si agregamos un vértice deja de ser conexo

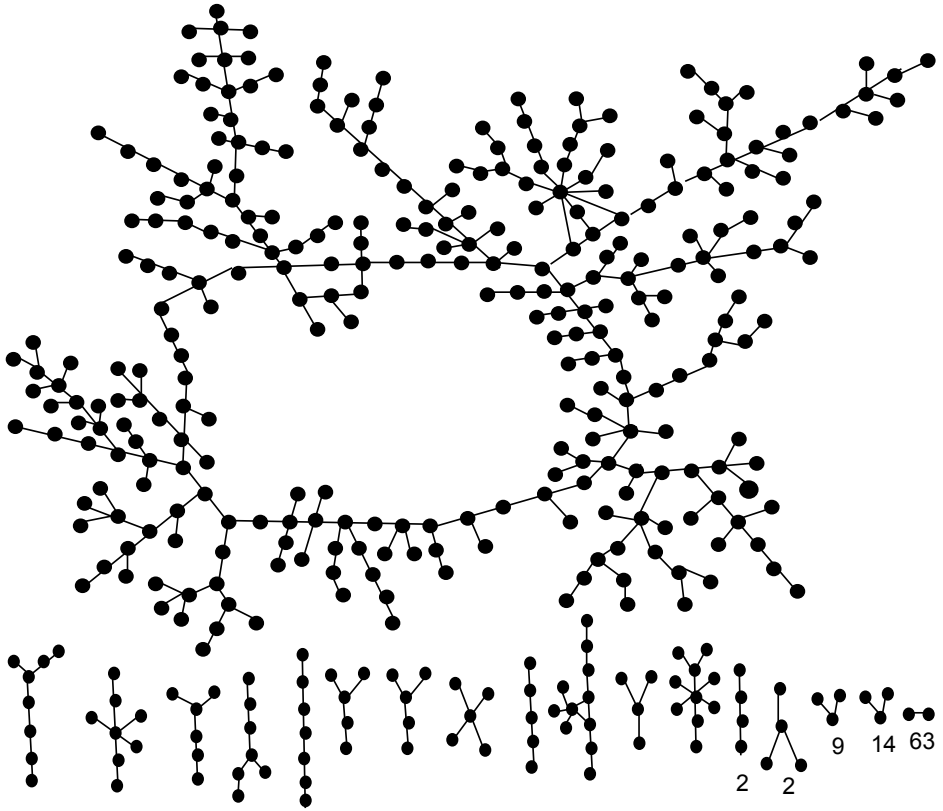


- En la Figura $\Rightarrow$ Las componentes son $\{1, 2, 5, 7\}$, $\{3, 6\}$ y $\{4\}$
  $\Rightarrow$ El subgrafo $\{3, 4, 6\}$ es disconexo,
  $\Rightarrow$ $\{1, 2, 5\}$ no es maximal
- Grafos disconexos tienen al menos dos componentes

  $\Rightarrow$ La componente más grande usualmente se llama componente gigante

Componente conexa gigante

- Las grandes redes reales generalmente tienen una componente gigante
- Ej: relaciones románticas en colegio de USA [Bearman et al'04]

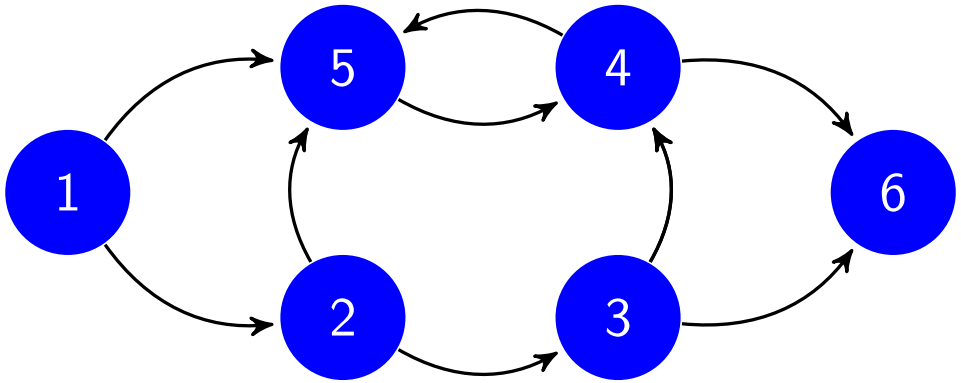


- Q: ¿Por qué esperamos encontrar una sola componente gigante?
- A: sólo se necesita una arista para unir dos componentes gigantes...

Conectividad en digrafos

- La conectividad es más sutil en grafos dirigidos. Dos notaciones
- **Def:** Digrafo fuertemente conexo $\Leftrightarrow$ para todo par $u, v \in V$, u es alcanzable desde v (via un camino dirigido) y viceversa

  $\Rightarrow$ todo nodo pertenece a un ciclo
- **Def:** Digrafo débilmente conexo $\Leftrightarrow$ conexo sin tener en cuenta las direcciones de los arcos (el grafo subyacente no dirigido es conexo)



- En la Figura, el grafo es débilmente conexo pero no fuertemente conexo

  $\Rightarrow$ Conectividad fuerte implica conectividad débil

Conectividad en digrafos: alcanzabilidad

- La componente saliente (*out-component*) del vértice v, son todos los vértices a los que puedo llegar desde v, inclusive v

  $\Rightarrow$ Las componentes salientes de los vértices de una componente fuertemente conexa coinciden

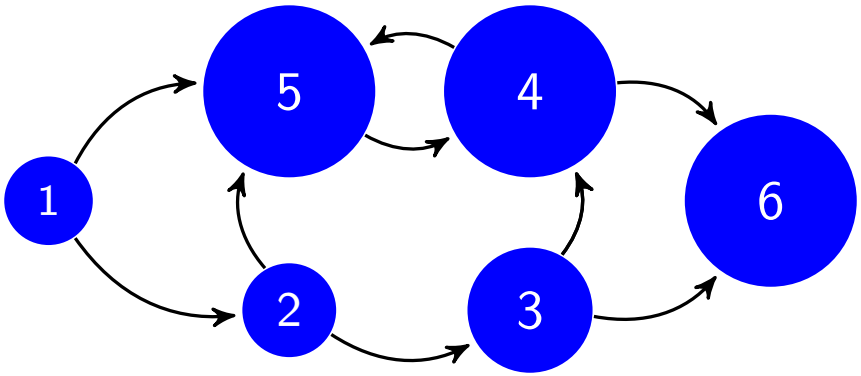
- La componente entrante (*in-component*) del vértice v, son todos los vértices que pueden llegar a v, inclusive v

  $\Rightarrow$ Las componentes entrantes de los vértices de una componente fuertemente conexa coinciden

- La intersección de las componentes entrante y saliente de v coincide con la componente fuertemente conexa de v

¿Qué tan bien conectados están los vértices?

- Q: ¿Qué vértice es el más conectado?
- A: Ranking de vértices para medir que tan central es
- Los indicadores de conectividad más importantes son
  - Cuántos enlaces apuntan a un vértice (los enlaces salientes son irrelevantes)
  - Qué tan importante son los enlaces que apuntan a un vértice



- Idea explotada por Google's PageRank© para medir la relevancia de las páginas Web

  ... por sociólogos para estudiar la confianza y reputación en redes sociales

  ... por la academia para medir impacto de trabajos científicos

  Próximamente más...

Familias de grafos

Definiciones y conceptos básicos

Movimientos en un grafo y conectividad

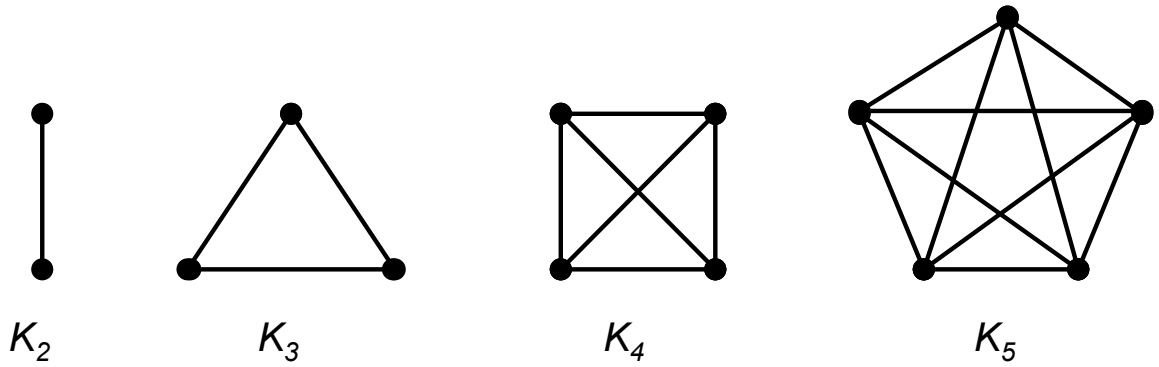
Familias de grafos

Álgebra de grafos

Grafos: estructuras de datos y algoritmos (opcional)

Grafos completos y *cliques*

- Un grafo completo K_n de orden n tiene todas las posibles aristas



- Q: ¿Cúal es el tamaño de K_n?

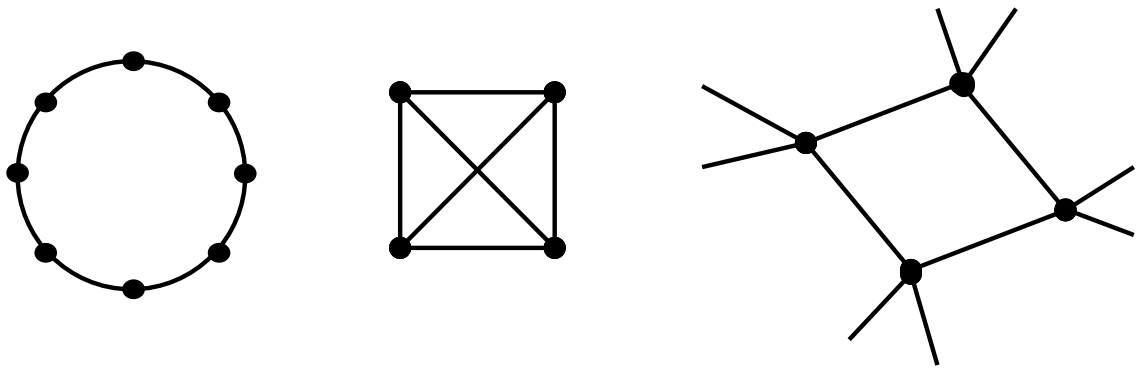
  A: Número de aristas en K_n = Número de pares de vértices = $\binom{n}{2} = \frac{n(n-1)}{2}$

- Los cliques son subgrafos completos

  $\Rightarrow$ Noción extrema de subgrupos cohesivos/comunidades

Grafos regulares

- Un grafo d-regular tiene todos sus vértices con grado d



- Obs., el grafo completo K_n es $(n-1)$-regular

  $\Rightarrow$ Los ciclos son (sub)grafos 2-regular

- Los grafos regulares son frecuentes al estudiar, por ejemplo:
  - la estructura de cristales (física y química)
  - configuraciones geo-espaciales como los modelos de adyacencia de píxeles (procesamiento de imágenes)
  - formación de la opinión, y ciclos de la información (sociología)

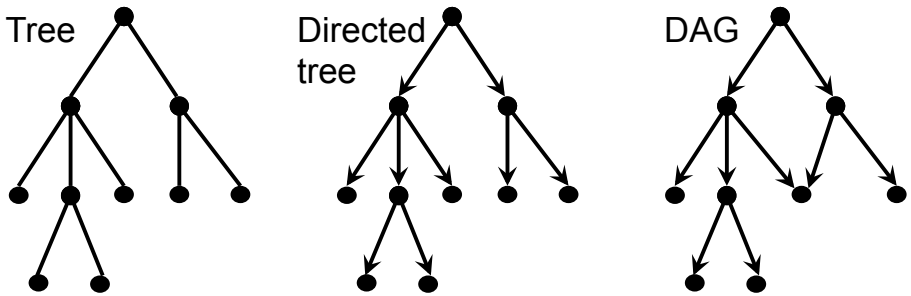
Árboles y grafos acíclicos dirigidos

- Un árbol es un grafo acíclico conexo. Un grafo acíclico es un bosque (un grafo sin ciclos)

  Ej: entramado de un río, cascadas de información en Twitter, redes de citaciones

  - Propiedades: "no tiene raiz", camino único entre todo par de vértices, $N_e = N_v - 1$
- Un árbol dirigido es un digrafo donde el grafo no dirigido subyacente es un árbol

  La raiz es el único vértice con camino a todos los otros
- Terminología de vértices: padre, hijo, ancestro, descendiente, hoja

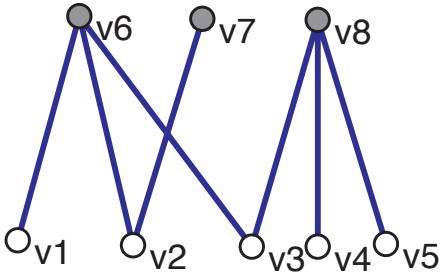


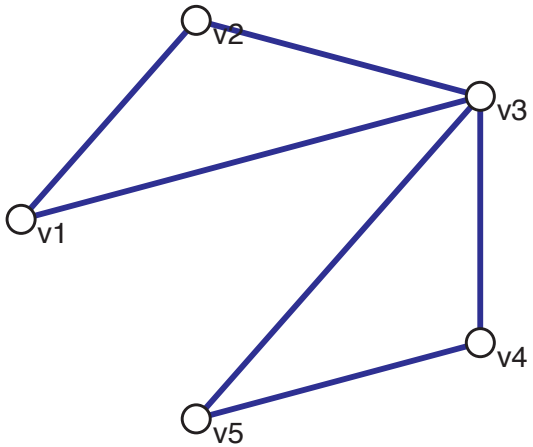
- El grafo subyacente de un grafo acíclico dirigido (DAG) no es un árbol

  DAGs tiene una estructura similar a un árbol, útil para los algoritmos

Grafos bipartitos

- Un grafo $G(V, E)$ es bipartito cuando
  - V puede ser particionado en dos conjuntos disjuntos, V_1 y V_2; y
  - Cada arista en E tiene un extremo en V_1 y el otro en V_2





- Útil para representar redes de pertenencia o afiliación

  $\Rightarrow$ Vértices en V_1 pueden ser personas, y vértices en V_2 son clubes

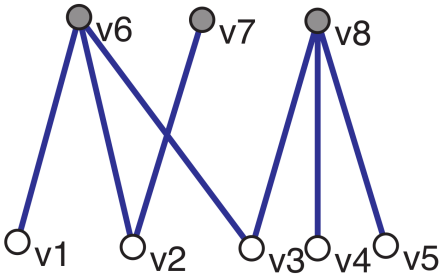
  $\Rightarrow$ Grafo inducido $G(V_1, E_1)$ junta miembros del mismo club.

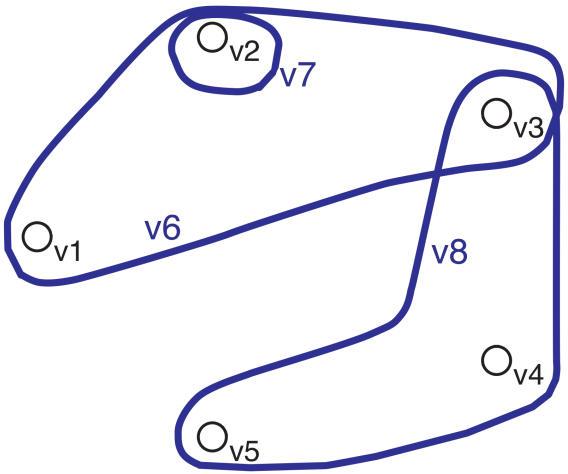
  Próximamente más

Hipergrafos como grafos bipartitos

- Una hiper-arista es una generalización de arista que junta a más de dos vértices
- Un hiper-grafo es un grafo con alguna hiper-arista

  Ej. pertenencia a grupo
- Todo hiper-grafo puede representarse con un grafo bipartito





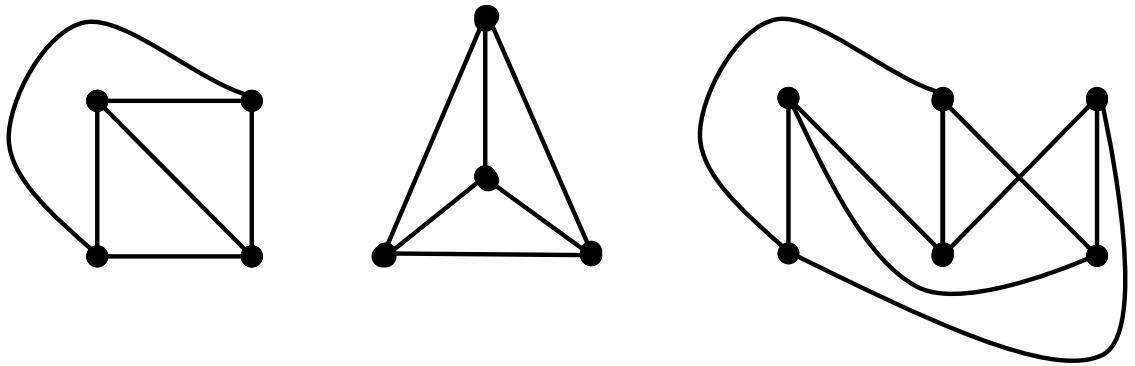
- En la Figura, *v6*, *v7*, y *v8* son hiper-aristas (se dibujan encerrando los vértices comunmente)

Hipergrafos como grafos bipartitos (ejemplos de redes reales)

Network	**Vertex**	**Group**
Film actors	Actors	Cast of a film
Coauthorship	Author	Authors of an article
Boards of directors	Director	Board of a company
Social events	People	Participants at social event
Recommender system	People	Item to recommend
Keyword index	Keywords	Pages where words appear
Rail connections	Stations	Train routes
Metabolic reactions	Metabolities	Participants in a reaction
⋮	⋮	⋮

Grafos Planares

- Un grafo $G(V, E)$ es planar si puede ser dibujado en un plano sin que sus aristas se crucen



- Grafos planares pueden ser dibujados usando solo lineas rectas
- **Def:** número cromático de un grafo: cantidad de colores necesarios para colorear los vértices sin que dos vértices adyacentes compartan color)
  - Grafos planares pueden ser coloreados con a lo sumo 4 colores
- Útil para representar datos espaciales (datos sobre mapas)

  $\Rightarrow$ Ej.: ríos, fronteras geográficas

  $\Rightarrow$ Grafos planares son raros

  $\Rightarrow$ Existen algoritmos de visualización que minimizan las aristas que se cruzan

Álgebra de grafos

Definiciones y conceptos básicos

Movimientos en un grafo y conectividad

Familias de grafos

Álgebra de grafos

Grafos: estructuras de datos y algoritmos (opcional)

Matriz de Adyacencia

- Teoría algebraica de grafos trabaja con representaciones matriciales de grafos
- Q: ¿Cómo capturamos la conectividad de $G(V, E)$ en una matriz?
- A: Matriz de adyacencia, binaria y simétrica, $\mathrm{A} \in \{0,1\}^{N_v \times N_v}$, con elementos

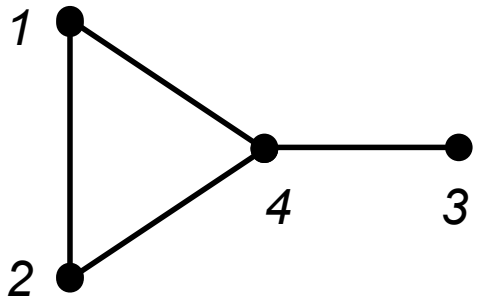
$$A_{ij} = \begin{cases} 1, & \text{si } (i,j) \in E \\ 0, & \text{otro caso} \end{cases} .$$

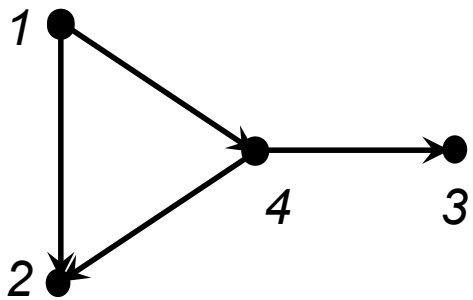
  $\Rightarrow$ Notar que los vértices son indexados con enteros $1, \ldots, N_v$

- En palabras, A es "uno" para los elementos cuyos índices fila-columna representan vértices en V unidos por una arista en E, y "cero" en otro caso
- Atención, [NE] define distinto la matriz de adyacencia, $\mathrm{A}_{[NE]} = \mathrm{A}^T$ (difiere en digrafos)

Matriz de Adyacencia: ejemplos

- Ejemplos para grafos no dirigidos y digrafos





$$A_u = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}, \quad A_d = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

- En digrafos, la matriz de adyacencia no es simétrica
- Recordar que la mayoría de las redes reales tiene matriz de adyacencia dispersa

Matriz de Adyacencia: extensiones

- Multi-grafos
  - los elementos de A cuentan la multiplicidad de las aristas
  - Un bucle en el vértice i, en un multi-grafo no dirigido, se representa por $A_{ii} = 2$ (en un multi-grafo dirigido por $A_{ii} = 1$)

  $\Rightarrow$ Para los grafos simples, $A_{ii} = 0 \; \forall \; i \in V$
- Grafos ponderados
  - ponderación en las aristas: puede guardarse el peso en (i, j) en lugar de 1
  - ponderación en los vértices: puede guardarse el peso en (i, i)

Matriz de Adyacencia: propiedades

- La matriz de adyacencia es útil para guardar la estructura del grafo. Próximamente más

  $\Rightarrow$ También, operaciones sobre A dan información útil sobre G

- Grados: sumas por filas dan los grados de los vértices, $\sum_{j=1}^{N_v} A_{ij} = d_i$

  Para digrafos, A no es simétrica y las sumas por fila y columna difieren:

$$\sum_{j=1}^{N_v} A_{ij} = d_i^{out}, \qquad \sum_{i=1}^{N_v} A_{ij} = d_j^{in}$$

- Caminos: Sea A^r la r-sima potencia de A, con elementos A_{ij}^r

  $\Rightarrow$ A_{ij}^r es la cantidad de caminos $i - j$ de largo r en G

- Corolarios:

  $\mathrm{tr}(\mathrm{A}^2)/2 = N_e$
  $\mathrm{tr}(\mathrm{A}^3)/6 = \#\triangle$ en G
  Cantidad de ciclos de largo r en G, $L_r = \mathrm{tr}(\mathrm{A}^r)$ (contando repetidos)

recordar, traza: $\mathrm{tr}(\mathrm{M}) = \sum_{i=1}^{N_v} M_{ii}$

Matriz de Adyacencia: propiedades (cont)

- En grafos no dirigidos:
  - La matriz de adyacencia es simétrica $\Rightarrow$ tiene N_v valores propios reales no negativos, k_i, y vectores propios con elementos reales
  - Siempre se puede descomponer $A = UKU^T$, donde U es una matriz ortogonal de los vectores propios, y K es una matriz diagonal de los valores propios
  - Se cumple que la cantidad de ciclos de largo r, $L_r = \sum_{i=1}^{N_v} k_i^r$
- En digrafos:
  - La matriz de adyacencia no tiene porque ser simétrica, es una matriz real
  - Siempre se puede descomponer $A = QTQ^T$ (descomposición de Schur), donde Q es una matriz ortogonal, y T es una matriz triangular. Los valores propios de T, k_i, son sus elementos diagonales, y coinciden con los valores propios de A (pueden ser complejos conjugados)
  - También se cumple, $L_r = \sum_{i=1}^{N_v} k_i^r$
- Espectro: G es d-regular $\Leftrightarrow$ 1 es un vector propio de A (con valor propio d):

$$A1 = d1$$

Matriz de Adyacencia: ejemplo de cálculos

- En las redes de citaciones, hay una arista desde el artículo i al artículo j, si i cita a j.
- La naturaleza de la red determina propiedades en A:

  $\Rightarrow$ Es un grafo acíclico dirigido (no se retrocede en el tiempo), A es nilpotente (todos los valores propios nulos)
- También interesante: se puede transformar en dos grafos no dirigidos, simples (ponderados):
  - Grafo de cocitación
  - Grafo de acoplamiento bibliográfico
- Grafo de cocitación, no dirigido, simple: hay una arista entre i y j si hay otro artículo que cita a ambos:
  - matriz de cocitación $\mathrm{C} = \mathrm{A}^T\mathrm{A}$, donde C_{ij} son la cantidad de artículos que citan a ambos i y j, y C_{ii} son la cantidad de artículos que citan a i
  - Matriz de adyacencia,

$$A^{cc}_{ij} = \begin{cases} 1, & \text{si } i \neq j \text{ y } C_{ij} > 0 \\ 0, & \text{otro caso} \end{cases} .$$

  - ponderación de vértices C_{ii}, ponderación de aristas C_{ij}

Matriz de Adyacencia: ejemplo de cálculos (cont)

- Grafo de acoplamiento bibliográfico, no dirigido, simple: hay una arista entre i y j si ambos artículos citan a otro en común
  - matriz de acoplamiento bibliográfico $\mathrm{B} = \mathrm{A}\mathrm{A}^T$, donde B_{ij} son la cantidad de artículos que i y j citan en común, y B_{ii} son la cantidad de artículos que i cita
  - Matriz de adyacencia,

$$A_{ij}^{\mathbf{ab}} = \begin{cases} 1, & \text{si } i \neq j \text{ y } B_{ij} > 0 \\ 0, & \text{otro caso} \end{cases} .$$

  - ponderación de vértices B_{ii}, ponderación de aristas B_{ij}
- Para analizar una red dirigida es útil transformarla en un grafo no dirigido. Dos opciones:
  - Eliminar la dirección de los arcos (se pierde mucha información!)
  - Transformarlo en uno/ambos de los grafos no dirigidos anteriores

Matriz de Incidencia

- Un grafo también puede representarse por su $N_v \times N_e$ matriz de incidencia B

  $\Rightarrow$ B en general no es cuadrada, a menos que $N_v = N_e$

- Para Grafos no dirigidos, los elementos de B son

$$B_{ij} = \begin{cases} 1, & \text{si vértice } i \text{ incide en la arista } j \\ 0, & \text{otro caso} \end{cases} .$$

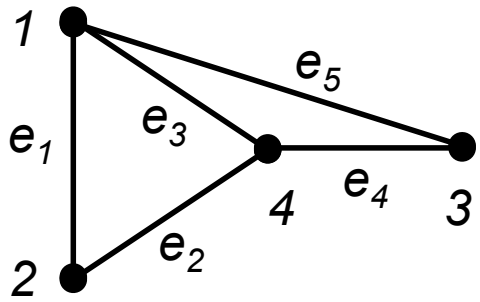
- Para Digrafos, se incluye la dirección de los arcos

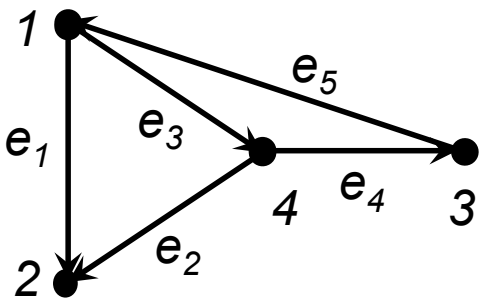
$$B_{ij} = \begin{cases} 1, & \text{si la arista } j \text{ es } (k,i) \\ -1, & \text{si la arista } j \text{ es } (i,k) \\ 0, & \text{otro caso} \end{cases} .$$

- Atención, [NE] define distinto la matriz de incidencia, $\mathrm{B}_{[NE]} = \mathrm{B}^T$

Matriz de Incidencia: ejemplos

- Ejemplos para grafos no dirigidos y digrafos



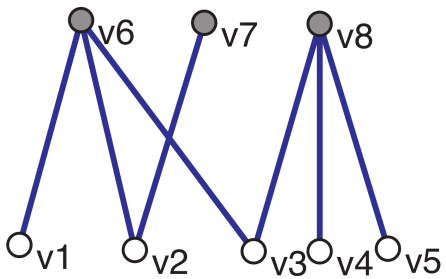


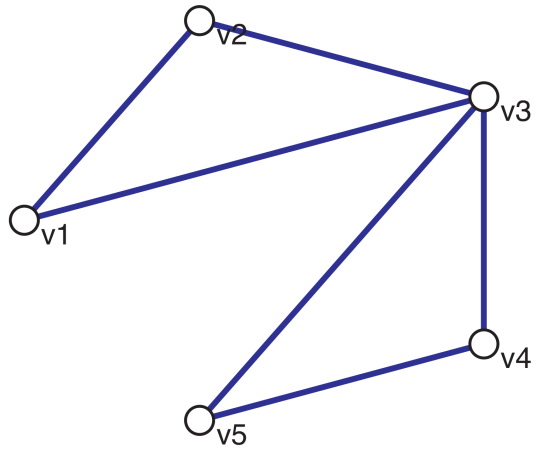
$$B_u = \begin{pmatrix} 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}, \quad B_d = \begin{pmatrix} -1 & 0 & -1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -1 \\ 0 & -1 & 1 & -1 & 0 \end{pmatrix}$$

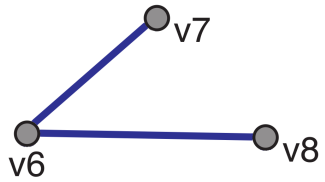
- En grafos ponderados: modificar los elementos 1 y -1 por el peso de la arista

Matriz de Incidencia: ejemplo de cálculos

- En los grafos bipartitos, el elemento B_{ij} de la matriz de incidencia puede interpretarse como si el vértice $i \in V_1$ pertenece al grupo $j \in V_2$
- En Sociología un grafo bipartito se llama red de dos modos (*two-mode network*)
- Puede proyectarse en dos redes de un modo (*one-mode network*):
  - Proyección a vértices
  - Proyección a grupos
    (Similar a lo realizado con las redes de citaciones)







Matriz de Incidencia: ejemplo de cálculos (cont)

- Proyección a vértices, V_1, no dirigido, simple: hay una arista entre i y j si comparten un grupo:
  - Sea $\mathrm{P}^1 = \mathrm{B}\mathrm{B}^T$, donde P^1_{ij} son la cantidad de grupos que comparten i y j, y P^1_{ii} son la cantidad de grupos a los que pertenece i
  - Matriz de adyacencia,

$$A^1_{ij} = \begin{cases} 1, & \text{si } i \neq j \text{ y } P^1_{ij} > 0 \\ 0, & \text{otro caso} \end{cases} .$$

  - ponderación de vértices P^1_{ii}, ponderación de aristas P^1_{ij}
- Proyección a grupos, V_2, no dirigido, simple: hay una arista entre los grupos i y j si comparten vértices:
  - Sea $\mathrm{P}^2 = \mathrm{B}^T\mathrm{B}$, donde P^2_{ij} son la cantidad de vértices que pertencen a ambos grupos i y j, y P^2_{ii} son la cantidad de vértices en el grupo i
  - Matriz de adyacencia,

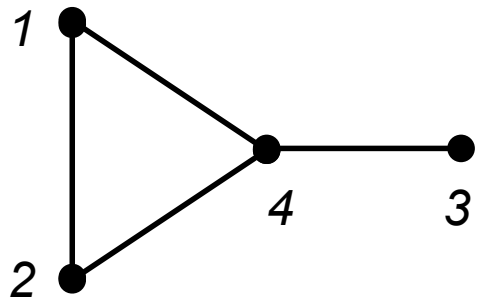
$$A^2_{ij} = \begin{cases} 1, & \text{si } i \neq j \text{ y } P^2_{ij} > 0 \\ 0, & \text{otro caso} \end{cases} .$$

  - ponderación de vértices P^2_{ii}, ponderación de aristas P^2_{ij}

Matriz Laplaciana

- Sea D la matriz diagonal, donde $D_{ii} = d_i$ es el grado del vértice i

$$\mathrm{D} = \begin{pmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$



- La matriz simétrica $N_v \times N_v$, $\mathrm{L} := \mathrm{D} - \mathrm{A}$ se llama Matriz Laplaciana (*graph Laplacian*)

$$L_{ij} = \begin{cases} d_i, & \text{si } i = j \\ -1, & \text{si } (i,j) \in E \\ 0, & \text{otro caso} \end{cases}, \quad \mathrm{L} = \begin{pmatrix} 2 & -1 & 0 & -1 \\ -1 & 2 & 0 & -1 \\ 0 & 0 & 1 & -1 \\ -1 & -1 & -1 & 3 \end{pmatrix}$$

Matriz Laplaciana: difusión y origen del nombre

- Difusión de un fluido en un grafo:
  - El vértice i tiene $\psi_i(t)$ del fluido en el tiempo t
  - El fuido fluye de i a su adyacente j a una tasa $C(\psi_i - \psi_j)$, donde C es una constante positiva
  - La ecuación que rige la difusión es:

$$\frac{\partial \psi}{\partial t} + C\mathrm{L}\psi = 0$$

- Difusión de gas en espacio continuo:

$$\frac{\partial \psi}{\partial t} + C\nabla^2\psi = 0$$

(donde el operador ∇^2 se llama laplaciano)

- el nombre de matriz laplaciana surge de esta similitud

Matriz Laplaciana: propiedades

- Suavidad: Para cualquier vector $x \in \mathbb{R}^{N_v}$ de "valores de vértices", se cumple

$$x^\top Lx = \sum_{(i,j)\in E} (x_i - x_j)^2$$

que puede ser minimizado para reforzar la suavidad de las funciones sobre G

$\Rightarrow$ Semi-definida positiva: porque $x^\top Lx \geq 0$ para todo $x \in \mathbb{R}^{N_v}$, y los valores propios son no negativos: $0 \leq \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$

- Singular: como $L1 = 0$,
  - El valor propio más pequeño λ_1 de L es 0
  - L no tiene inversa (es deficiente en rango)
- Espectro y conectividad:
  - El segundo valor propio más pequeño $\lambda_2 \neq 0 \Leftrightarrow G$ es conexo
  - L tiene n valores propios nulos $\Leftrightarrow G$ tiene n componentes conexas

Lectura guiada

[SAND] Capítulo 2, Sección 2.1.

[SANDR] Capítulo 2

Algoritmos e implementación

- En el curso no cubriremos la Parte III de [NE], CH9-11 $\approx$ 120 pags.
- Trata de algoritmos e implementación
- En su lugar incluiremos la siguiente sección resumida

Lectura opcional

[NE] CH6 $\approx$ 60 pags.

- 6 - MATHEMATICS OF NETWORKS
  6.1 NETWORKS AND THEIR REPRESENTATION
  6.2 THE ADJACENCY MATRIX
  6.3 WEIGHTED NETWORKS
  6.4 DIRECTED NETWORKS
  6.4.1 COCITATION AND BIBLIOGRAPHIC COUPLING
  6.4.2 ACYCLIC DIRECTED NETWORKS
  6.5 HYPERGRAPHS
  6.6 BIPARTITE NETWORKS
  6.7 TREES
  6.8 PLANAR NETWORKS
  6.9 DEGREE
  6.10 PATHS
  6.10.1 GEODESIC PATHS
  6.10.2 EULERIAN AND HAMILTONIAN PATHS
  6.11 COMPONENTS
  6.11.1 COMPONENTS IN DIRECTED NETWORKS
  6.12 INDEPENDENT PATHS, CONNECTIVITY, AND CUT SETS
  6.12.1 MAXIMUM FLOWS AND CUT SETS ON WEIGHTED NETWORKS
  6.13 THE GRAPH LAPLACIAN
  6.13.1 DIFFUSION
  6.13.2 EIGENVALUES OF THE GRAPH LAPLACIAN
  6.13.3 COMPONENTS AND THE ALGEBRAIC CONNECTIVITY
  6.14 RANDOM WALKS
  6.14.1 RESISTOR NETWORKS

mandatorio opcional fuera de alcance

Grafos: estructuras de datos y algoritmos (opcional)

Definiciones y conceptos básicos

Movimientos en un grafo y conectividad

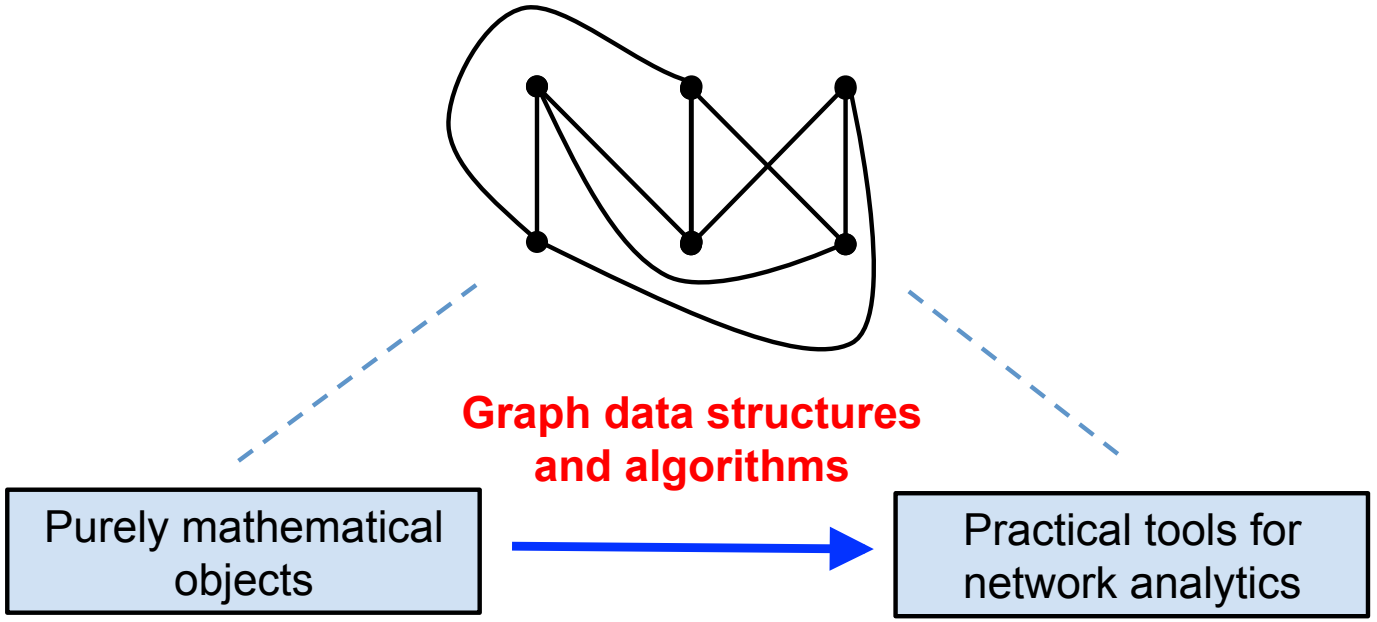
Familias de grafos

Álgebra de grafos

Grafos: estructuras de datos y algoritmos (opcional)

Grafos: estructuras de datos y algoritmos

- Q: ¿Cómo guardar y analizar un grafo G usando una computadora?



- Estructura de datos: guardado y manejo eficiente de un grafo
- Algoritmos: métodos computacionales escalables para analizar un grafo

  $\Rightarrow$ mayores contribuciones son de la Ciencia de la Computación

Estructuras de datos para grafos

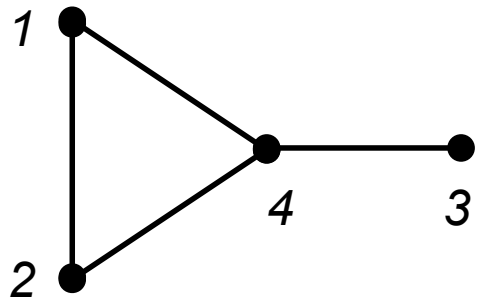
- Q: ¿Cómo representar y guardar un grafo G en una computadora?
- A: Estructuras de datos frecuentemente utilizadas:
  1) Matriz de adyacencia
  2) Lista de adyacencia
     - una variante es la Lista de aristas

Estructura de datos: matriz de adyacencia

- La matriz de adyacencia, $\mathrm{A} \in \{0,1\}^{N_v \times N_v}$

$$A_{ij} = \begin{cases} 1, & \text{si } (i,j) \in E \\ 0, & \text{otro caso} \end{cases} .$$

$$\mathrm{A} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix}$$



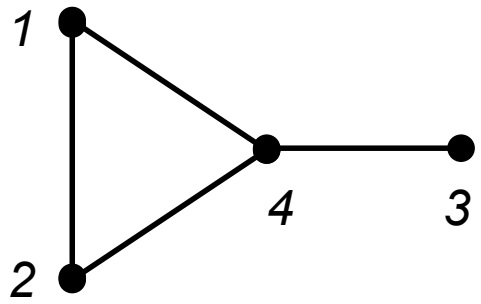
- Las matrices (arrays) son objetos de datos básicos en los entornos de software
  - Implementación básica: cota superior de uso de memoria es $O(N_v^2)$
    $\Rightarrow$ Representación no deseada para grafos grandes y dispersos (situación usual)
  - Software de cálculo incluyen matrices dispersas eficientes

Estructura de datos: lista de adyacencia y lista de aristas

- La lista de adyacencia representa el grafo G en un *array* de tamaño N_v

  El elemento i-simo del *array* es una lista de vértices adyacentes a i

$$
\begin{aligned}
L_a[1] &= \{2,4\} \\
L_a[2] &= \{1,4\} \\
L_a[3] &= \{4\} \\
L_a[4] &= \{1,2,3\}
\end{aligned}
$$



- Similarmente, la lista de aristas guarda la pareja de vértices incidentes a cada arista

$$
\begin{aligned}
L_e[1] &= \{1,2\} \\
L_e[2] &= \{1,4\} \\
L_e[3] &= \{2,4\} \\
L_e[4] &= \{3,4\}
\end{aligned}
$$

- Ambos casos, requieren memoria $O(N_v + N_e)$. Existen implementaciones con optimizaciones de acceso (*hash* y uso de I/O)

Algoritmos y complejidad

Pueden plantearse muchas preguntas interesantes sobre un grafo dado...

- Las más simples, pueden responderse directamente de las estructuras de datos:

  Q1: ¿Los vértices u y v son enlazados por una arista?
  Q2: ¿Cuál es el grado del vértice u?

- Otras requieren más trabajo, pero pueden responderse eficientemente[1]:

  Q1: ¿Cuál es el camino más corto entre dos vértices u y v?
  Q2: ¿Cuántas componentes conexas tiene un grafo?
  Q3: ¿Es acíclico un digrafo?

- Desafortunamente, hay casos que no se conocen algoritmos eficientes

  Q1: ¿Cuál es el clique maximal de un grafo?

Teoría de complejidad es clave en el análisis de (grandes) redes modernas

[1] tiempo polinomial en N_v y/o N_e

Testeo de conectividad

- Goal: verificar la conectividad de un grafo usando la lista de adyacencia
- Idea: comenzar en el vértice s, explorar el grafo, marcar vértices visitados

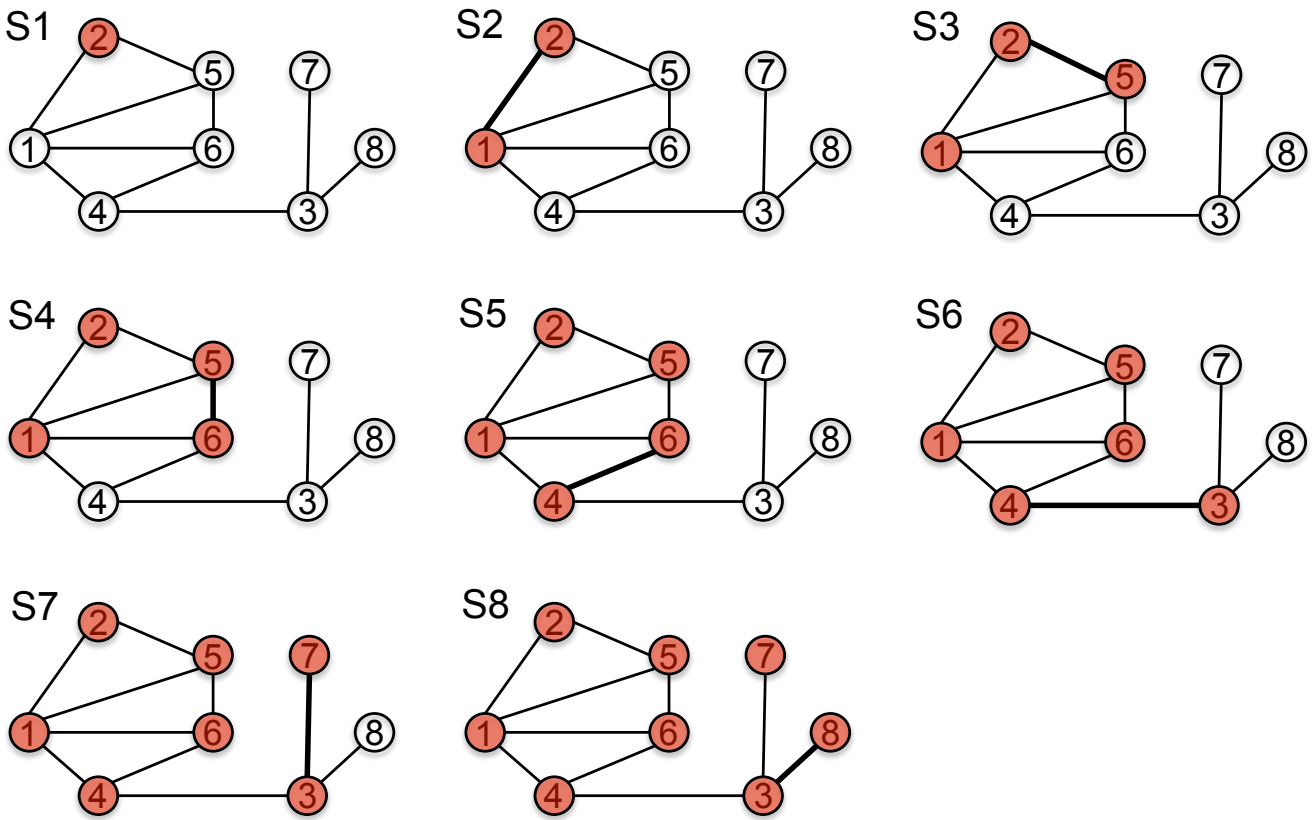
Output : List M of marked vértices in the component
Input : Graph G (e.g., adjacency list)
Input : Starting vertex s

$L := \{s\}$; $M := \{s\}$; % Initialize exploration and marking lists
% Repeat while there are still nodes to explore
while $L \neq \emptyset$ **do**
  choose $u \in L$; % Pick arbitrary vertex to explore
  if $\exists\,(u, v)$ *such that* $v \notin M$ **then**
    choose (u, v) with v of smallest index;
    $L := L \cup \{v\}$; $M := M \cup \{v\}$; % Mark and augment
  else
    $L := L \setminus \{u\}$; % Prune
  end
end

Testeo de conectividad: ejemplo de exploración

- En cada paso indicamos los vértices elegidos y marcados. Iniciando en $s = 2$

L	Mark
{2}	2
{2,1}	1
{2,1,5}	5
{2,1,5,6}	6
{1,5,6}	
{1,5,6,4}	4
{5,6,4}	
{5,4}	
{5,4,3}	3
{5,3}	
{5,3,7}	7
{5,3}	
{3}	
{3,8}	8
{3}	
{}	



- La exploración lleva $2N_v$ pasos y es en orden arbitrario. Cada vértice se agrega y quita una vez

Breadth-first search (BFS): exploración en amplitud

- Breadth-first search (BFS): elige u como el primer elemento de L

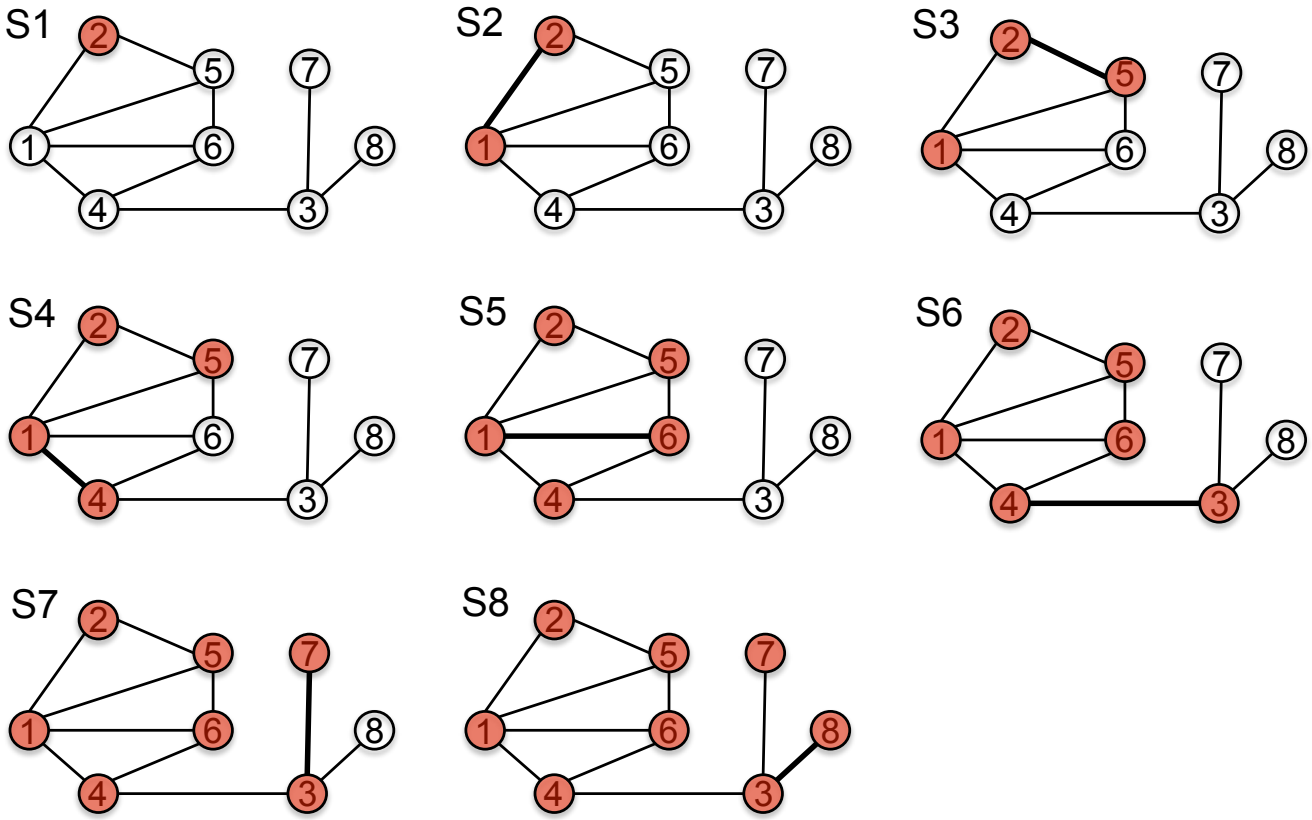
Output : List M of marked vértices in the component
Input : Graph G (e.g., adjacency list)
Input : Starting vertex s

$L := \{s\}$; $M := \{s\}$; % Initialize exploration and marking lists
% Repeat while there are still nodes to explore
while $L \neq \emptyset$ **do**
    $u :=$ first(L); % Breadth first
    if $\exists\,(u, v)$ *such that* $v \notin M$ **then**
        choose (u, v) with v of smallest index;
        $L := L \cup \{v\}$; $M := M \cup \{v\}$; % Mark and augment
    else
        $L := L \setminus \{u\}$; % Prune
    end
end

BFS: ejemplo

- En cada paso indicamos los vértices elegidos y marcados. Iniciando en $s = 2$

L	Mark
{2}	2
{2,1}	1
{2,1,5}	5
{1,5}	
{1,5,4}	4
{1,5,4,6}	6
{5,4,6}	
{4,6}	
{4,6,3}	3
{6,3}	
{3}	
{3,7}	7
{3,7,8}	8
{7,8}	
{8}	
{}	



- El algortimo construye un árbol "ancho" (primero amplitud)

Depth-first search (DFS): exploración en profundidad

- Depth-first search (DFS): elige u como el último elemento de L

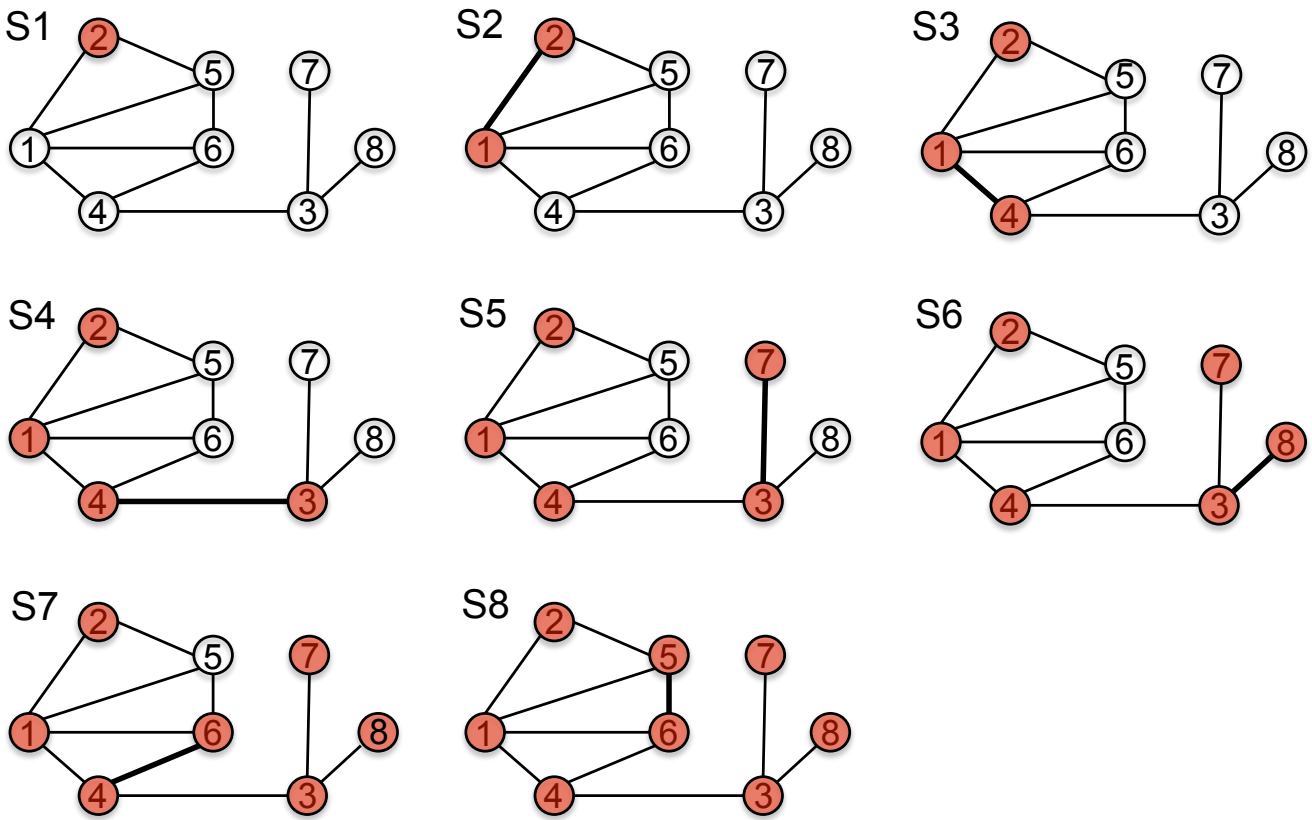
Output : List M of marked vértices in the component
Input : Graph G (e.g., adjacency list)
Input : Starting vertex s

$L := \{s\}$; $M := \{s\}$; % Initialize exploration and marking lists
% Repeat while there are still nodes to explore
while $L \neq \emptyset$ **do**
    $u :=$ last(L); % Depth first
    if $\exists\,(u, v)$ *such that* $v \notin M$ **then**
        choose (u, v) with v of smallest index;
        $L := L \cup \{v\}$; $M := M \cup \{v\}$; % Mark and augment
    else
        $L := L \setminus \{u\}$; % Prune
    end
end

DFS: ejemplo

- En cada paso indicamos los vértices elegidos y marcados. Iniciando en $s = 2$

L	Mark
{2}	2
{2,1}	1
{2,1,4}	4
{2,1,4,3}	3
{2,1,4,3,7}	7
{2,1,4,3}	
{2,1,4,3,8}	8
{2,1,4,3}	
{2,1,4}	
{2,1,4,6}	6
{2,1,4,6,5}	5
{2,1,4,6}	
{2,1,4}	
{2,1}	
{2}	
{}	



- El algoritmo construye los caminos más largos (primero profundidad)

Computar distancias con BFS

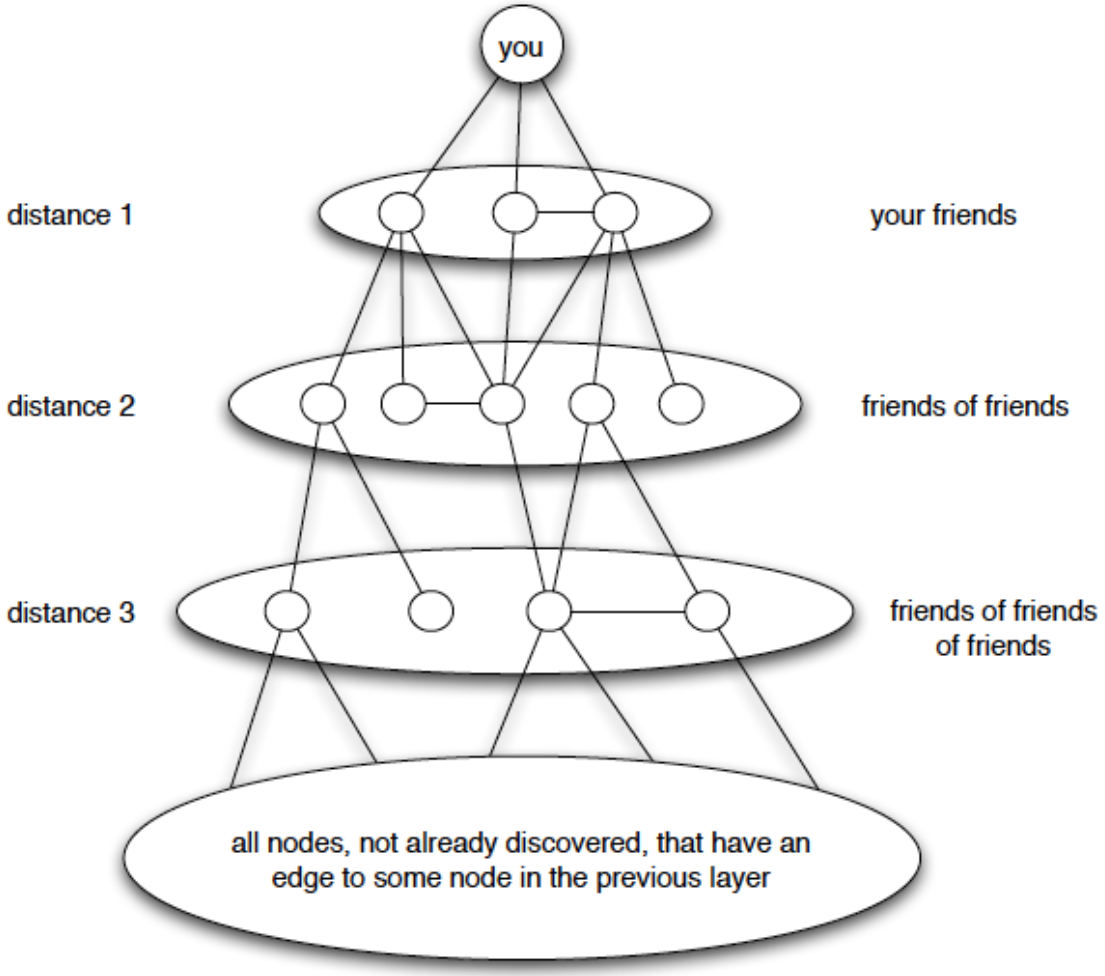
- Algoritmos conocidos: algoritmo de Dijkstra , BFS (pesos unitarios)
- Usar BFS y guardar traza de los largos de los caminos en la exploración. Incrementar las distancias en 1 cada vez que un vértice es marcado

Output : Vector d of distances from reference vertex
Input : Graph G (e.g., adjacency list)
Input : Reference vertex s

$L := \{s\}$; $M := \{s\}$; $d(s) = 0$; % Initialization
while $L \neq \emptyset$ *% Repeat while there are still nodes to explore*
do
  $u :=$ first(L); % Breadth first
  if $\exists\,(u, v)$ *such that* $v \notin M$ **then**
    choose (u, v) with v of smallest index;
    $L := L \cup \{v\}$; $M := M \cup \{v\}$; % Mark and augment
    $d(v) := d(u) + 1$ % Increment distance
  else
    $L := L \setminus \{u\}$; % Prune
  end
end

Ejemplo: distancias en redes sociales

- Árbol BFS para la red de amigos de una persona



Glosario

- (Di) Graph
- Arc
- (Induced) Subgraph
- Incidence
- Degree sequence
- Walk, trail and path
- Connected graph
- Giant connected component
- Strongly connected digraph
- Clique
- Tree
- Bipartite graph
- Directed acyclic graph (DAG)
- Adjacency matrix
- Graph Laplacian
- Adjacency and edge lists
- Sparse graph
- Graph density
- Breadth-first search
- Depth-first search (DFS)
- Geodesic distance (BFS)
- Diameter

Lectura guiada

[SAND] Capítulo 2, Sección 2.1.

[SANDR] Capítulo 2

Lectura opcional

[NE] CH9-11 $\approx$ 120 pags.

- 9 - BASIC CONCEPTS OF ALGORITHMS
  9.1 RUNNING TIME AND COMPUTATIONAL COMPLEXITY
  9.2 STORING NETWORK DATA
  9.3 THE ADJACENCY MATRIX
  9.4 THE ADJACENCY LIST
  9.5 TREES
  9.6 OTHER NETWORK REPRESENTATIONS
  9.7 HEAPS
- 10 - FUNDAMENTAL NETWORK ALGORITHMS
  10.1 ALGORITHMS FOR DEGREES AND DEGREE DISTRIBUTIONS
  10.2 CLUSTERING COEFFICIENTS
  10.3 SHORTEST PATHS AND BREADTH-FIRST SEARCH
  10.4 SHORTEST PATHS IN NETWORKS WITH VARYING EDGE LENGTHS
  10.5 MAXIMUM FLOWS AND MINIMUM CUTS
- 11 - MATRIX ALGORITHMS AND GRAPH PARTITIONING
  11.1 LEADING EIGENVECTORS AND EIGENVECTOR CENTRALITY
  11.2 DIVIDING NETWORKS INTO CLUSTERS
  11.3 GRAPH PARTITIONING
  11.4 THE KERNIGHAN-LIN ALGORITHM
  11.5 SPECTRAL PARTITIONING
  11.6 COMMUNITY DETECTION
  11.7 SIMPLE MODULARITY MAXIMIZATION
  11.8 SPECTRAL MODULARITY MAXIMIZATION
  11.9 DIVISION INTO MORE THAN TWO GROUPS
  11.10 OTHER MODULARITY MAXIMIZATION METHODS
  11.11 OTHER ALGORITHMS FOR COMMUNITY DETECTION

mandatorio opcional fuera de alcance