Representación de Grafos

FCEyN UBA

Técnicas de Diseño de Algoritmos

FCEyN UBA

2C 2024

Esquema de hoy

1. Repaso
2. Modelado y Representación de grafos
   - Lista de aristas
   - Lista de adyacencia
   - Matriz de adyacencia
   - La fiesta de Santi
   - La CABArrera
3. Ejercicios Demostraciones sobre grafos
   - Buscaminas
   - Demo incorrecta
   - Demo inducción
4. Árboles

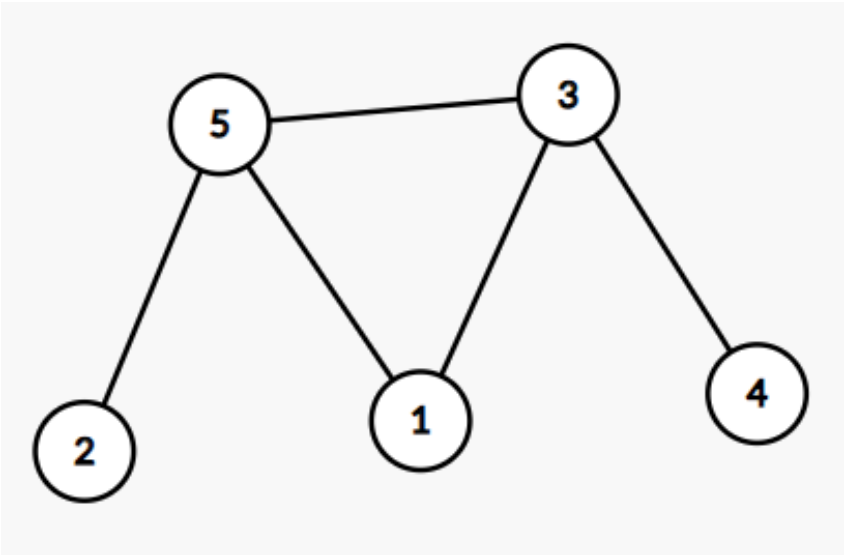
Esquema de hoy

Definiciones

Grafo

Un grafo es un par (V, E) con V un conjunto de nodos (o vértices) y E conjunto de aristas (o ejes) de la forma (u, v) con u, v $\in$ V.

Si no se aclara, se asume que los ejes no son dirigidos, es decir que (u, v)=(v, u).



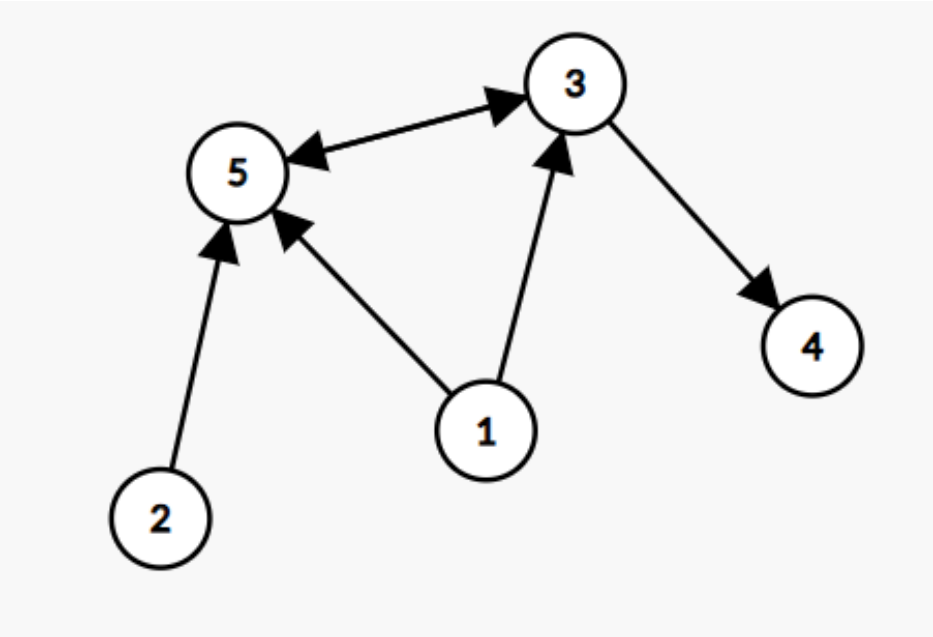
- En general vamos a notar *n* la cantidad de nodos y *m* a la cantidad de aristas.

Definiciones

Digrafo

Es un grafo, es decir, un par (V, E) como antes, pero cuyas aristas están orientadas. Es decir que en este caso $(u, v) \neq (v, u)$

Puedo tener dos ejes entre dos nodos, uno en cada sentido.



Definiciones

Otros:

- **Multigrafo:** grafo que permite tener múltiples aristas entre dos nodos.
- **Pseudografo:** grafo que permiten tener loops.
- **Grafo pesado:** grafo G=(V, E, w) con w(e) una "función de pesos"que asigna a cada arista e=(u,v) un peso. (Observacion: se puede usar para simplificar multigrafos).

Por el momento nos vamos a concentrar en grafos y digrafos.

Más definiciones...

- **Recorrido:** una sucesión de vértices y aristas del grafo
- **Camino:** un recorrido que **no** pasa dos veces por el mismo vértice.
- **Circuito:** un recorrido que empieza y termina en el mismo nodo
- **Ciclo o circuito simple:** un circuito que **no** repite vértices. (Nota: para grafos, no consideramos como válido al ciclo de longitud 2)
- **Longitud:** la longitud de un recorrido se nota $l(P)$ y es la cantidad de *aristas* del mismo.
- **Distancia entre dos vértices:** longitud del camino más corto entre los vértices (si no existe se dice ∞).
- Un nodo es **adyacente** a otro si están conectados.
- Una arista es **incidente** a un nodo si conecta dicho nodo con algún otro.

Esquema de hoy

1. Repaso
2. Modelado y Representación de grafos
   - Lista de aristas
   - Lista de adyacencia
   - Matriz de adyacencia
   - La fiesta de Santi
   - La CABArrera
3. Ejercicios Demostraciones sobre grafos
   - Buscaminas
   - Demo incorrecta
   - Demo inducción
4. Árboles

Modelando con grafos

- Una ciudad?
- Una red social?
- Amistades?
- Operaciones aritméticas?
- Redes de información?
- Tareas?
- El cerebro humano?

Representando grafos

Por conveniencia, vamos a suponer que todos los nodos del grafo son números de [1, n].

Principalmente vamos a utilizar las siguientes estructuras de representación:

- Lista de aristas
- Lista de adyacencia
- Matriz de adyacencia

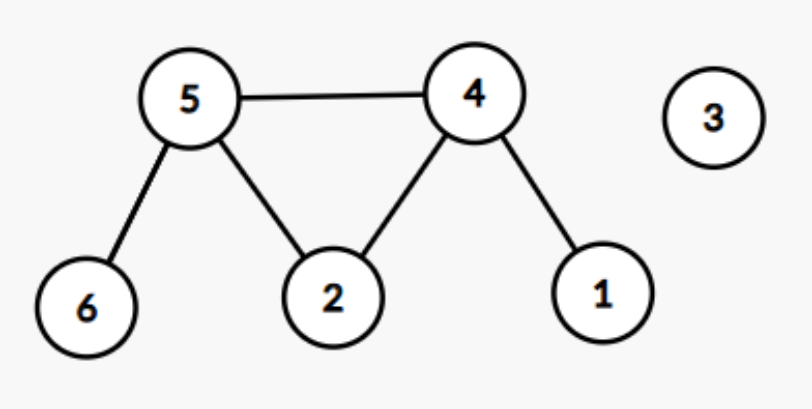
Veamos cada una de estas representaciones y sus diferencias.

Ejemplo: lista de aristas

lista de aristas

El conjunto de aristas como una secuencia (lista).

Tomemos el siguiente grafo como ejemplo:



Lista de aristas:

$\{(6,5),(5,2),(2,4),(5,4),(4,1)\}$

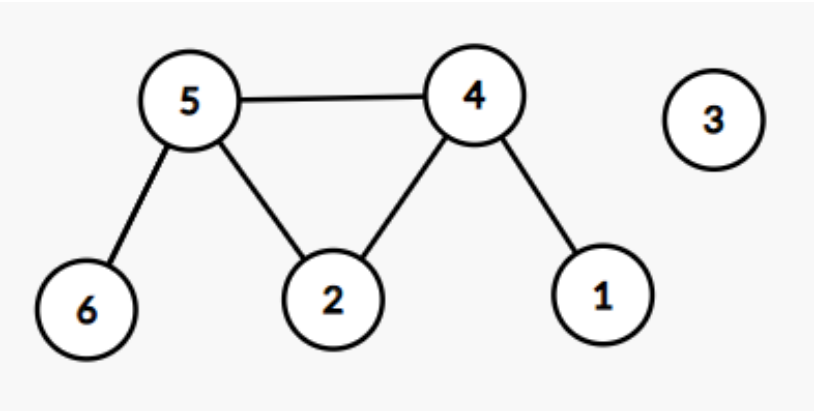
Nota 1: normalmente esta va a ser la única representación con la que se van a expresar los inputs de grafos.
Nota 2: si junto con la lista se pasa el tamaño del grafo, como por convención las etiquetas de los vértices están numeradas 1...n podemos deducir qué vértices no tienen vecinos.

Ejemplo: lista de adyacencia

Lista de adyacencia

El diccionario es un vector y los vecindarios son listas de tamaño d(v) conteniendo a los nodos vecinos.

Tomemos el siguiente grafo como ejemplo:



Lista de adyacencia:

Nodo : lista de vecinos

1 : 4

2 : 4 $\longrightarrow$ 5

3 :

4 : 5 $\longrightarrow$ 1 $\longrightarrow$ 2

5 : 2 $\longrightarrow$ 6 $\longrightarrow$ 4

6 : 5

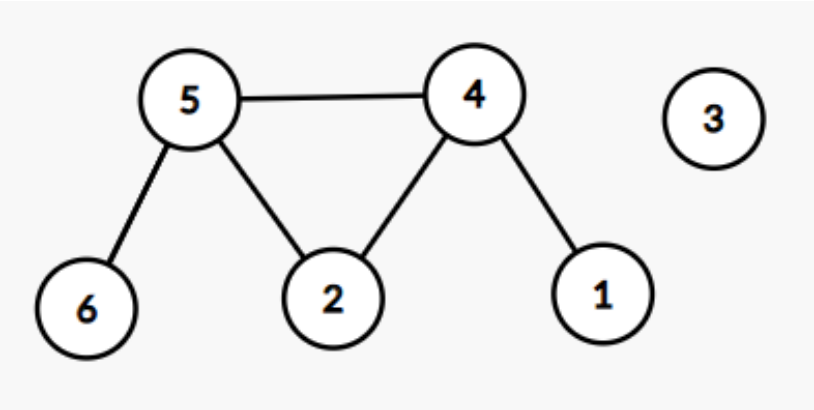
Nota: Se pueden hacer cosas como ordenar los vecindarios, pero es caro mantenerlo si el grafo cambia.

Ejemplo: matriz de adyacencia

Matriz de adyacencia

El diccionario y los vecindarios son vectores de tamaño n. Resultando así en una matriz de nxn donde $M_{ij} = 1$ si los vértices i y j son adyacentes y $M_{ij} = 0$ si no.

Tomemos el siguiente grafo como ejemplo:



Matriz de adyacencia:

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

Operaciones

¿Qué operaciones podríamos llegar a querer hacer sobre un grafo?

- Inicializar un grafo.
- Agregar o sacar un nodo del grafo.
- Agregar o sacar una arista del grafo
- Obtener el vecindario de un nodo v.
- Evaluar si dos aristas u y v son adyacentes.

Complejidades

Las complejidades para las representaciones vistas quedarían:

	lista de aristas	matriz de ady.	listas de ady.
construcción	$O(m)$	$O(n^2)$	$O(n+m)$
adyacentes	$O(m)$	$O(1)$	$O(d(v))$
vecinos	$O(m)$	$O(n)$	$O(d(v))$
agregarArista	$O(m)$	$O(1)$	$O(d(u)+d(v))$
removerArista	$O(m)$	$O(1)$	$O(d(u)+d(v))$
agregarVértice	$O(1)$	$O(n^2)$	$O(n)$
removerVértice	$O(m)$	$O(n^2)$	$O(n+m)$

Nota: como el tamaño del grafo está dado por su cantidad de nodos y aristas, una complejidad O(n+m) es lineal respecto al tamaño del grafo.

Eligiendo una representación

Qué representación conviene usar va a depender de:

- Las características del grafo (por ejemplo si es ralo o denso).
- Para qué lo vamos a querer usar y qué complejidades queremos para sus operaciones.

La fiesta de Santi

Enunciado

Santi quiere festejar su cumpleaños con otros k-1 amigos en su monoambiente. Para estar cómodos preferirían que no haya demasiada gente en la fiesta, entonces se pusieron de acuerdo en invitar solamente a los amigos que tienen en común.
Como Santi está muy cansado con tantos trabajos, nos pidió que, dada la lista de todos los amigos que tiene cada uno de los k cumpleañeros, le armemos la lista de invitados.

La fiesta de Santi: Pseudocódigo

Sea $G = (V, E)$ y $S = \{s_1, \ldots, s_k\}$ donde $s_i \in V$

Algorithm 1 Invitados(G,S)

```
1: invitados ← ∅
2: for v_i ∈ V do                                          ▷ O(n)
3:     if ∀s_j ∈ S : v_i ∈ N(s_j) then    ▷ matriz:O(k) / lista:O(d_m k)
4:         agregar(v_i, invitados)                          ▷ O(1)
5:     end if
6: end for
7: return invitados
```

$d_m = max_{v \in V}(d(v))$

Complejidad matriz de adyacencia: $O(nk)$
Complejidad lista de adyacencia: $O(nd_m k)$

La fiesta de Santi: Pseudocódigo 2

Sea $G = (V, E)$ y $S = \{s_1, \ldots, s_k\}$ donde $s_i \in V$

Algorithm 2 Invitados(G,S)

```
1: times ← [0, ..., 0]                                   ▷ O(n)
2: cumpleañero ← bitmap(S)                                ▷ O(n)
3: for (v, u) ∈ E do                                      ▷ O(m)
4:     if ¬cumpleañero[v] ∧ cumpleañero[u] then           ▷ O(1)
5:         times[v] + +                                   ▷ O(1)
6:     end if
7: end for
8: return {v ∈ V : times[v] = |S|}                        ▷ O(n)
```

Complejidad lista de adyacencia: $O(n + m)$
Complejidad lista de aristas: $O(n + m)$

La CABArrera

Enunciado

Dadas las hazañas de Franco Colapinto en F1, la Ciudad de Buenos Aires quiere hacer un circuito para que el corredor pueda hacer una carrera por la ciudad. Por las altas velocidades que se manejan en estas carreras, la ciudad quiere que desde todo punto del circuito se tenga acceso a un punto seguro en caso de una falla inesperada. Una vez en este punto ya no se podrán reincorporar a la carrera. Como la ciudad está muy ocupada por la organización nos pidieron que verifiquemos que el circuito propuesto tenga un punto seguro.

- ¿Cuál es la solución más directa? ¿Qué complejidad tiene?
- Buscar la fila en la matriz que tenga todos 0
- Complejidad: $\mathcal{O}(n^2)$

La CABArrera

¿Nos podemos aprovechar de la matriz?
¿Que pasa si $M_{ij} = 1$? ¿Y si $M_{ij} = 0$?

- Nos aprovechamos de que todos los nodos deben ser incidentes con lo cual, podemos ir descartando de a filas y columnas
- Si $M_{ij} = 1$ entonces sabemos que existe el arco ij con lo cual podemos descartar el nodo i.
- Si $M_{ij} = 0$ entonces sabemos que no existe el arco ij con lo cual podemos descartar el nodo j.
- La siguiente coordenada que visitaremos será el $max(i, j) + 1$ para asegurarnos que en toda iteración $i \neq j$ ya que $M_{ii} = 0$ para todos los nodos.

La CABArrera

Veamos algunas iteraciones, donde (i, j) es la coordenada y $\{v_1, \ldots, v_n\}$ los posibles candidatos.

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

$(1, 2)$ $\{1, 2, 3, 4\}$

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

$(3, 2)$ $\{2, 3, 4\}$

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

$(3, 4)$ $\{3, 4\}$

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

$(5, 4)$ $\{4\}$

Como 4 está fuera de rango nos quedamos con la fila 3 que representa el nodo 4.

La CABArrera

Sea M la matriz de adyacencia de G

Algorithm 3 Encontrar Nodo (M)

```
1:  i ← 0
2:  j ← 1
3:  while i < n ∧ j < n do
4:      if M_ij = 1 then
5:          i = max(i, j) + 1
6:      end if
7:      if M_ij = 0 then
8:          j = max(i, j) + 1
9:      end if
10: end while
11: v = min(i, j)
12: return ChequeoCandidato(v, M)
```

La CABArrera

¿Con una lista de adyacencia será mejor?

- Buscar el nodo con lista vacía en $\mathcal{O}(n)$
- Verificar que no sea un nodo aislado ($d_{in}(v) > 0$) en $\mathcal{O}(n + m)$

¿Con lo cuál nos conviene?

- Con matriz de adyacencia: $\mathcal{O}(n)$
- Con lista de adyacencia: $\mathcal{O}(n + n + m)$

¿Que pasa si el grafo es denso?

Esquema de hoy

1. Repaso
2. Modelado y Representación de grafos
   - Lista de aristas
   - Lista de adyacencia
   - Matriz de adyacencia
   - La fiesta de Santi
   - La CABArrera
3. Ejercicios Demostraciones sobre grafos
   - Buscaminas
   - Demo incorrecta
   - Demo inducción
4. Árboles

Buscaminas

Enunciado

Vicky se aburría así que se puso a jugar al buscaminas. Empieza con un tablero T formado por n casillas, cada una de las cuales contiene o bien una bomba, o bien un número. Luego de resolverlo, como seguía aburrida, se puso a armar, otro tablero T' con casillas dispuestas de la misma manera que en T. La diferencia es que cada casilla de T' contiene un número si la correspondiente casilla de T contiene una bomba, y viceversa. Tanto en T como en T', cuando una casilla contiene un numero, esa es la cantidad de bombas en las casillas del mismo tablero que son vecinas a la dada.

$T =$

1	💣	1	2	💣
	2	3	2	💣
2	💣	💣	3	1
	💣	6	💣	1
2	💣	💣	2	0

$T' =$

💣	4	💣	💣	1
	💣	💣	💣	4
💣	4	4	💣	💣
	3	💣	4	💣
💣	2	2	💣	💣

Buscaminas

Enunciado

Como Vicky es muy buena con las matemáticas, cuando terminó de armar el tablero se dio cuenta instantáneamente que la suma de todos los números de T daba lo mismo que la suma de todos los números de T'. Esto le llamó mucho la atención, y nos pidió que la ayudemos a demostrar, usando grafos, que siembre la suma de los números en un tablero cualquiera T es igual a la suma de los números en un tablero T' armado como hizo ella.

- SUGERENCIA: modelar T con un grafo o digrafo G que tiene un vértice por cada casilla del tablero.
- Modelo: G=(V,E) donde V={Casilla del tablero}

  E=$\{(u, v) \in V \mid u$ es un número y v una bomba o viceversa$\}$

Buscaminas

Demostración:

- Sea G el grafo que modela el tablero T. Notar que el grafo es bipartito: una parte, llamémosla V_1, representa las casillas con bombas; y otra parte, V_2 representa las casillas con números.
- La suma de los números en T, es igual a la suma de los grados de los vértices en V_1, que al ser G bipartito es la cantidad de ejes de G.
- Sea G' el grafo que modela el tablero T'. Este grafo es isomorfo a G, donde ahora V_2 representa las casillas con bombas; y V_1 las casillas que tienen números. G' también es bipartito.
- Ahora, la suma de los números es T' va a ser igual a la suma de los grados de los vértices en V_2, que al ser G bipartito es la cantidad de ejes de G'.
- Ahora, como dijimos que G y G' son isomorfos, esto quiere decir que tienen la misma cantidad de aristas, y por lo tanto, la suma en T y T' son iguales.

Inducción en grafos

En grafos muchas veces vamos a tener que hacer inducción en la cantidad de nodos o aristas de un grafo.
Tengan en cuenta que:

- **NUNCA:** tomo un grafo G_k de k vértices (o k aristas) y digo que si una propiedad P se prueba para este grafo, agregándole un v vértice (o arista) sigue valiendo P para G_{k+1}. (*)
- **SIEMPRE:** tomo un grafo G_{k+1} de k + 1 vértices (o k + 1 aristas) con ciertas características y le saco un vértice (o arista) con algún tipo de estrategia particular y veo que cumple P. Luego, agrego el vértice o arista y veo que sigue cumpliendo P.

(*) el problema con esto es que estamos queriendo probar la propiedad para todo grafo G_{k+1} de k+1 nodos (o aristas). Sin embargo al tomar un grafo G_k y agregarle un nodo o arista estariamos solo considerando los grafos G_{k+1} que tenían a G_k como subgrafo, y eso no necesariamente eso abarca a todo G_{k+1} posible.

Demostrando

Enunciado

Si todos los vértices de un grafo tienen grado mayor a cero, entonces el grafo es conexo.

- Recordemos que un grafo es conexo si existe un camino entre cualquier par de vértices.

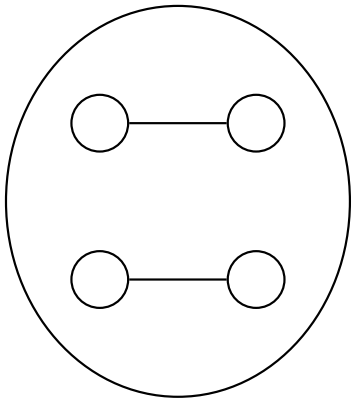
Demostrando **algo falso**

Enunciado

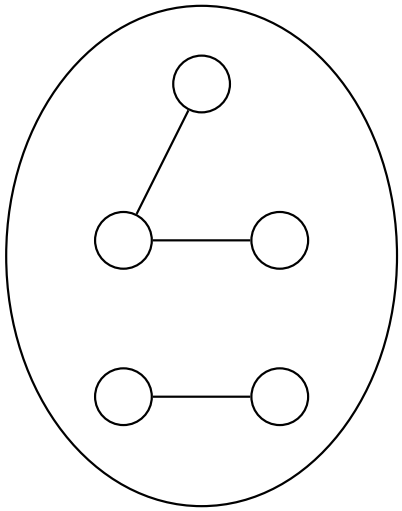
Si todos los vértices de un grafo tienen grado mayor a cero, entonces el grafo es conexo.

- Recordemos que un grafo es conexo si existe un camino entre cualquier par de vértices.

Contraejemplos:



G



H

Demostrando algo falso

Demostremos esta propiedad (falsa) por inducción (incorrecta) en el tamaño del grafo.

- P(n): Si todos los vértices de un grafo con n vértices tiene grado mayor a cero, entonces el grafo es conexo.
- Caso Base ($n \leq 2$):
  1. P(1) : no puede tener grado positivo. Cumple.
  2. P(2) : solo hay un grafo que cumple tener grados positivos, K2. Es un grafo conexo. Cumple.

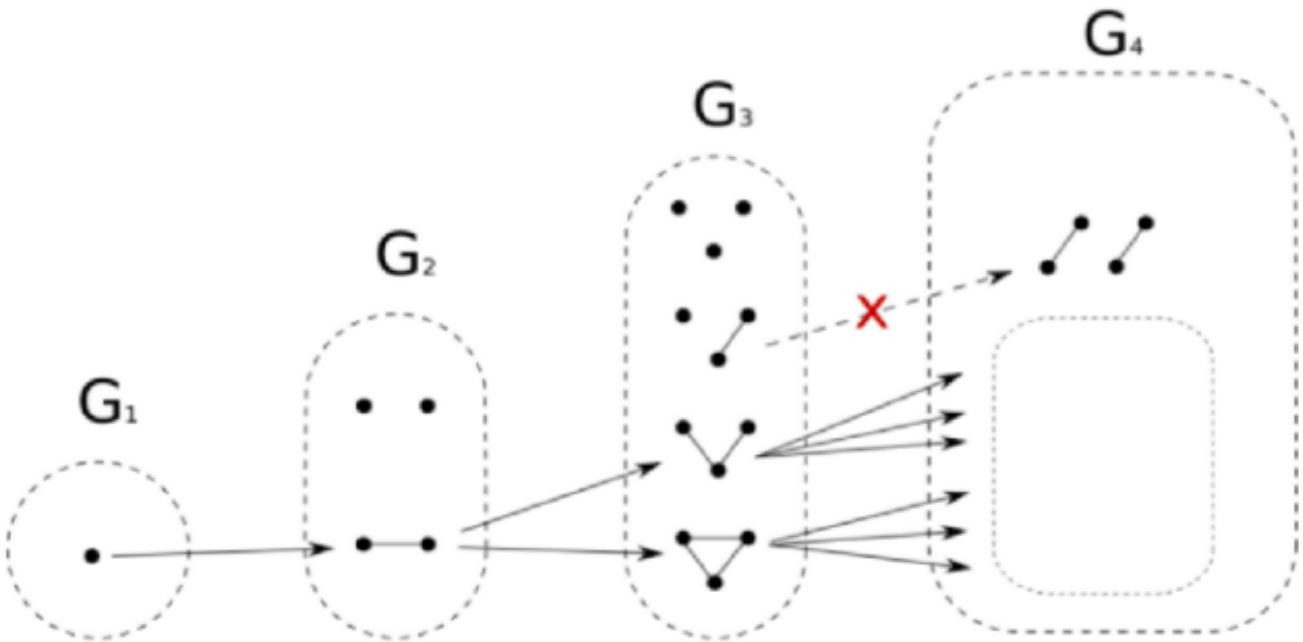
Demostrando algo falso: Paso Inductivo

Paso inductivo: debemos mostrar que P(n) $\Rightarrow$ P(n + 1) para todo n $\geq$ 2.

- Considerar G_n tal que $\forall v \in V(G_n)$, d(v) > 0.
- Por H.I. G_n es conexo. Agregamos el vértice x para obtener G_{n+1}.
- Para ver que G_{n+1} es conexo debemos ver que existe camino entre x y cualquier otro vértice z.
- Como x tiene grado positivo, existe una arista (x, y).
- Para llegar de x a z podemos usar la arista (x, y) y el camino [y, z]. Dicho camino existe pues G_n es conexo.
- Por lo tanto vale P(n+1).

Demostrando algo falso: Errores

Cada paso es correcto, el problema es que esto no prueba P(n + 1).
Necesitamos probar que **TODO** grafo de n + 1 vértices con grados positivos es conexo.
El error esta en suponer que todos los grafos de n + 1 vértices pueden ser construidos usando todos los grafos de n vértices que cumplen la propiedad P(n). (Ver contraejemplos)



Demo correcta por inducción

Enunciado

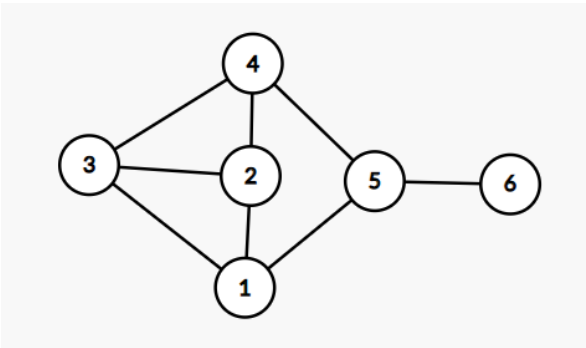
Probar que si G es un grafo de n nodos y tiene al menos n ejes, entonces tiene un ciclo.

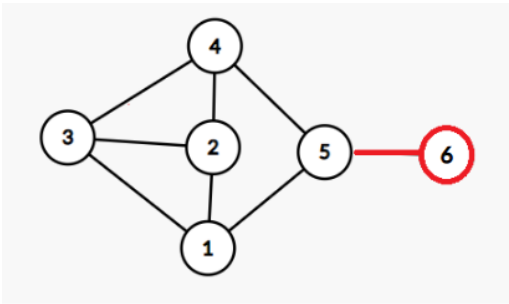
Demo correcta por inducción

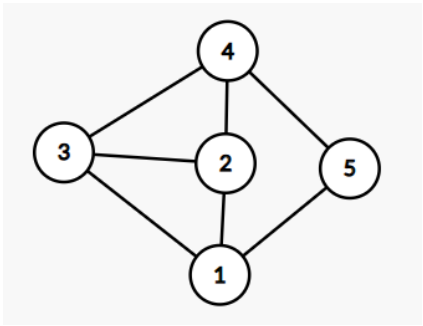
- **Hipótesis inductiva:** P(n)= para todo grafo de n nodos vale que si tiene al menos n ejes entonces tiene un ciclo.
- ¿Vale P(1)? El primer caso interesante es P(3), que también vale.
- Para el caso inductivo, queremos probar que $P(n) \Rightarrow P(n+1)$. Es decir, tenemos que tomar un grafo cualquiera de n+1 nodos, y demostrar (aprovechando que vale P(n)) que si tiene n+1 o más ejes entonces tiene un ciclo.

Demo correcta por inducción

- Sea G un grafo con n+1 nodos, y al menos n+1 ejes. Vamos a separar en dos casos:
- Hay un nodo v de grado a lo sumo 1: entonces $G \backslash \{v\}$ es un grafo de n nodos, con al menos n ejes. Luego, tiene un ciclo (por la hipótesis inductiva). Ese ciclo también está en G.
  A modo de ejemplo:







Demo correcta por inducción

- Caso contrario, todos los nodos tiene al menos grado 2. Se puede demostrar fácilmente entonces tiene al menos un ciclo:
  La idea es, podemos tomar un vértice v_1, movernos a otro vértice v_2, y luego siempre vamos a poder movernos a otro vértice v_i por una arista distinta a la que usamos para llegar.
  Pero como la cantidad de nodos es finita, si podemos ir de nodo en nodo de manera infinita, eventualmente vamos a tener que repetir un nodo. Por lo tanto existe un ciclo.

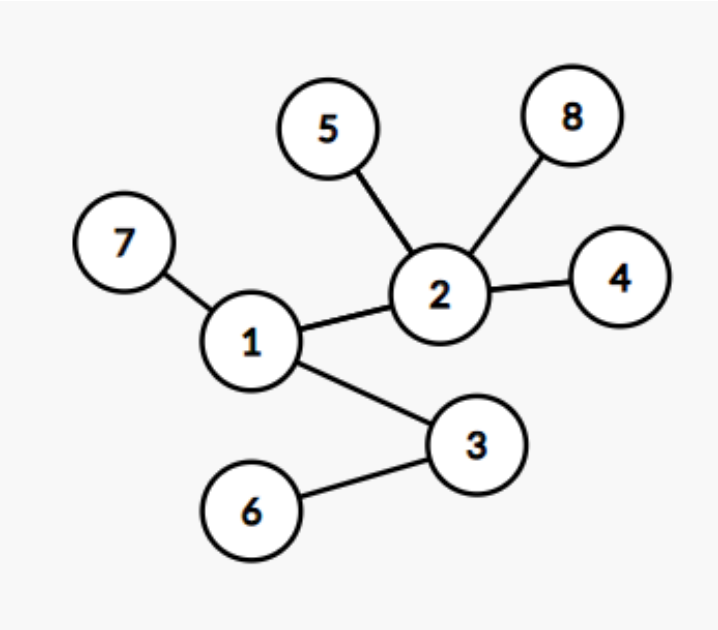
Esquema de hoy

Representación de árboles

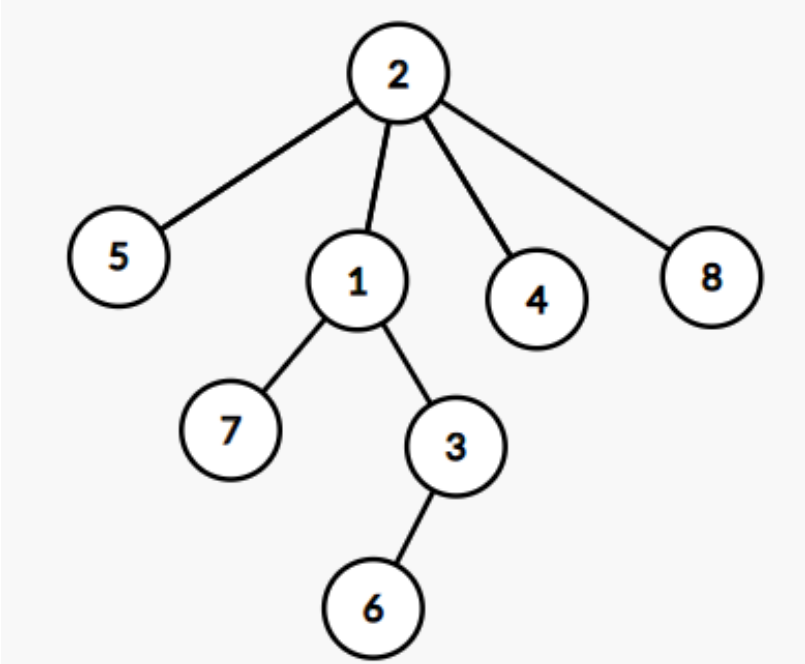
Árbol

Un árbol es un grafo conexo, acíclico y con n-1 aristas.
Nota: alcanza con saber que cumple con dos de dichas propiedades para afirmar que un grafo es un árbol.

Por ejemplo es un árbol:

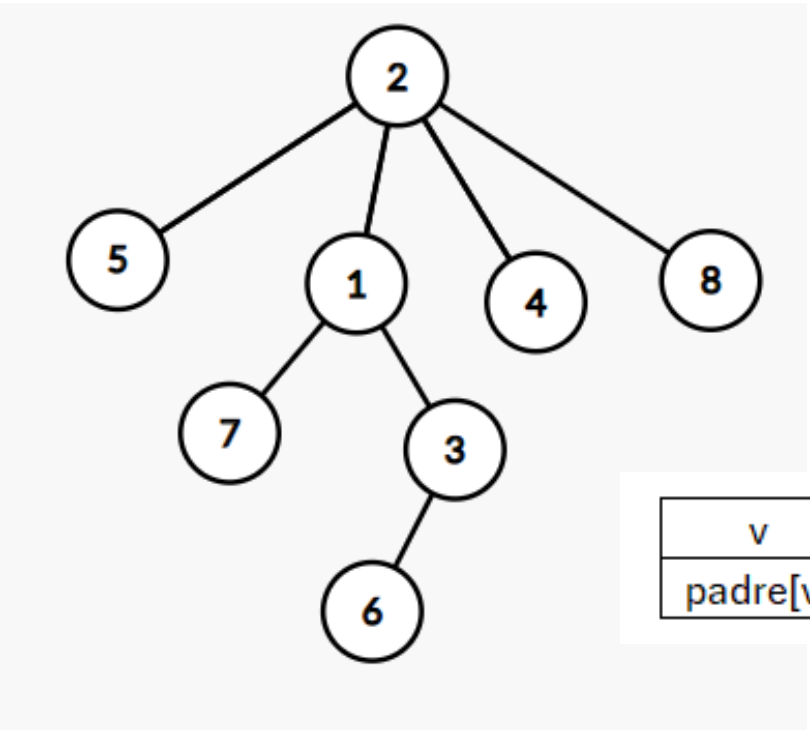


Si lo **enraizamos** por ejemplo en el nodo 2 nos quedaría:



Representación de árboles

¿Podemos aprovecharnos de esto para representarlos de una manera más compacta? ¡Sí!
Podemos tener una función *padre* tal que padre[v] es el (único) padre de v para todo v que no es la raíz. Para la raíz *r* podríamos poner padre[r] = r por ejemeplo.
Con el ejemplo anterior:



v	1	2	3	4	5	6	7	8
padre[v]	2	2	1	2	2	3	1	2