

---

# Tema 19

## Enrutamiento de amplia difusión (*broadcast*)

Eytan Modiano

# Enrutamiento de amplia difusión

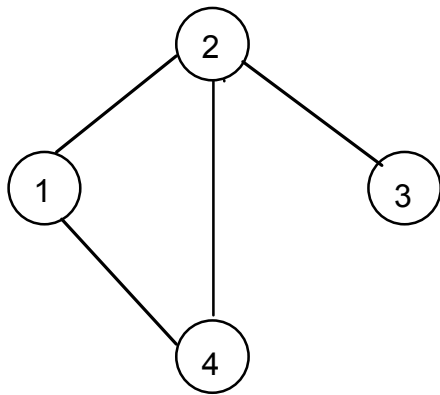
---

- **Se dirige un paquete desde un origen hacia todos los nodos de la red**
- **Posibles soluciones:**
  - **Inundación (*flooding*): cada nodo envía el paquete a todos los links de salida:  
Se descartan los paquetes recibidos por segunda vez**
  - **Enrutamiento de árbol de expansión: se envía el paquete a lo largo de un árbol que incluye todos los nodos de la red**

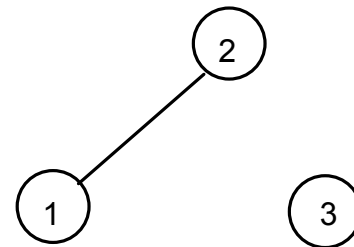
# Grafos

---

- Un grafo  $G = (N,A)$  consta de un conjunto finito no vacío de nodos  $N$  y un conjunto de pares de nodos  $A$  llamados arcos (enlaces o aristas)



$$N = \{1,2,3,4\}$$
$$A = \{(1,2),(2,3),(1,4),(2,4)\}$$

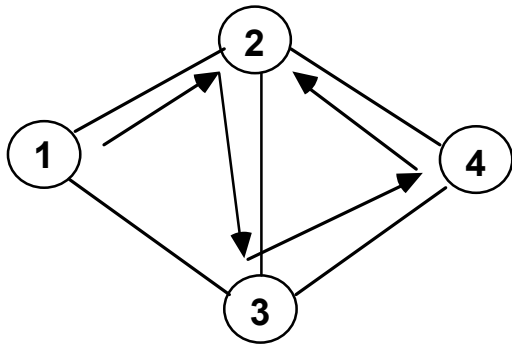


$$N = \{1,2,3\}$$
$$A = \{(1,2)\}$$

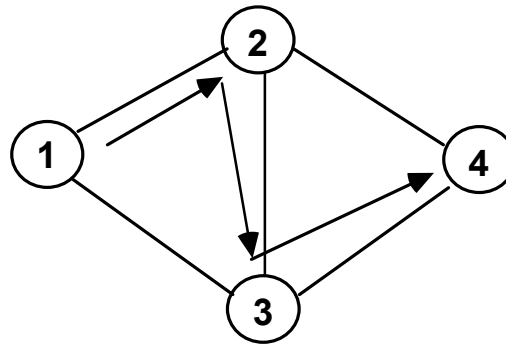
# Caminos y rutas

---

- Un camino (*walk*) es una secuencia de nodos ( $n_1, n_2, \dots, n_k$ ) en la que cada par de nodos adyacentes es un arco.
- Una ruta (*path*) es un camino sin nodos repetidos



**Camino (1,2,3,4,2)**

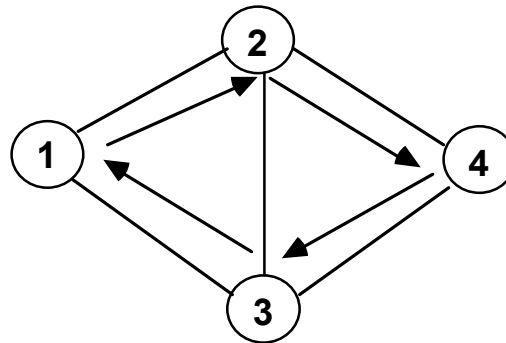


**Ruta (1,2,3,4)**

# Ciclos

---

- Un ciclo es un camino  $(n_1, n_2, \dots, n_k)$  con  $n_1 = n_k$ ,  $k > 3$ , y sin nodos repetidos excepto  $n_1 = n_k$

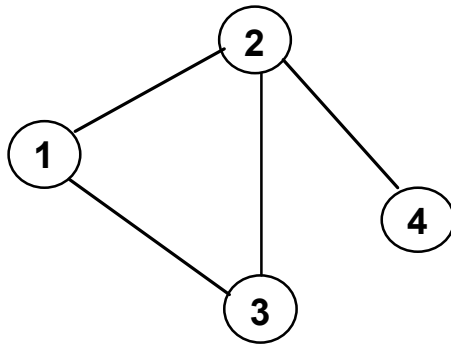


**Ciclo (1,2,4,3,1)**

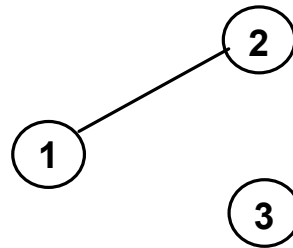
# Grafo conexo

---

- Un grafo es conexo si existe una ruta entre cada par de nodos



Conexo



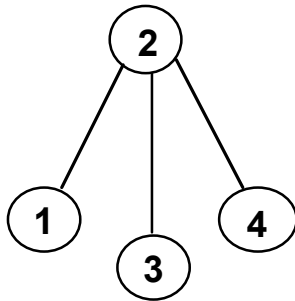
Inconexo

- Un grafo inconexo se puede separar en dos o más componentes conexos

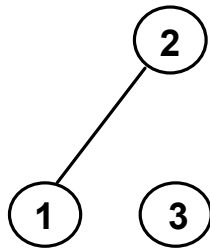
# Árboles y grafos acíclicos

---

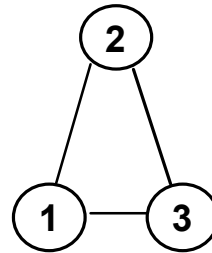
- Un grafo acíclico es un grafo sin ciclos
- Un árbol es un grafo acíclico conexo



**Acíclico,  
conexo**



**Inconexo,  
no árbol**



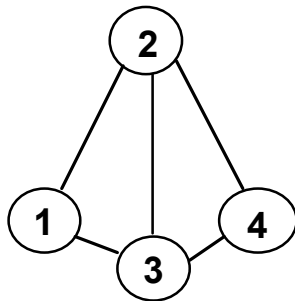
**Cíclico,  
no árbol**

- El número de arcos en un árbol es siempre uno menos que el número de nodos:
  - Demostración: empezar con un nodo arbitrario y añadir un nodo cada vez que se añade un arco => N nodos y N-1 enlaces. Si se añade un arco sin añadir un nodo, el arco deberá ir a un nodo que ya esté en el árbol formando, así, un ciclo

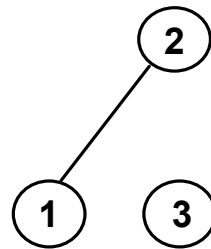
# Subgrafos

---

- $G' = (N', A')$  es un subgrafo de  $G = (N, A)$  si:
  - 1)  $G'$  es un grafo
  - 2)  $N'$  es un subconjunto de  $N$
  - 3)  $A'$  es un subconjunto de  $A$
- Un subgrafo se obtiene eliminando nodos y arcos de un grafo:
  - Nota: los arcos adyacentes a un nodo eliminado también deben ser eliminados



– Grafo  $G$

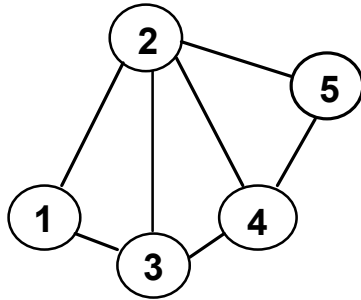


Subgrafo  $G'$  de  $G$

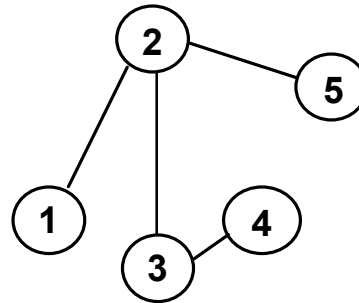
# Árboles de expansión (*Spanning trees*)

---

- $T = (N', A')$  es un árbol de expansión de  $G = (N, A)$  si:
  - $T$  es un subgrafo de  $G$  con  $N' = N$  y  $T$  es un árbol



**Grafo G**



**Árbol de expansión de G**

# Árboles de expansión

---

- **Los árboles de expansión son útiles para difundir y recoger información de control en las redes; a veces son útiles también para el enrutamiento**
- **Para difundir datos desde el nodo n:**
  - **El nodo n realiza una amplia difusión de los datos en todos los arcos del árbol adyacente**
  - **Otros nodos retardan los datos en los arcos de otros árboles adyacentes**
- **Para recoger datos en el nodo n:**
  - **Todas las ramas del árbol (todas salvo la n) envían datos**
  - **Los otros nodos (todos salvo el n) esperan la recepción de datos en todos sus arcos salvo en uno adyacente y, a continuación, envían los datos recibidos más los propios por el arco restante**

# Construcción general de un árbol de expansión

---

- **Algoritmo para construir un árbol de expansión para un grafo conexo  $G = (N,A)$ :**
  - 1) **Seleccionar cualquier nodo  $n$  en  $N$ ;  $N' = \{n\}$ ;  $A' = \{ \}$**
  - 2) **Si  $N' = N$ , entonces parar ( $T=(N',A')$  es un árbol de expansión)**
  - 3) **Elegir  $(i,j) \in A$ ,  $i \in N'$ ,  $j \notin N'$**   
 **$N' := N' \cup \{j\}$ ;  $A' := A' \cup \{(i,j)\}$ ; ir al paso 2**
- **La conectividad de  $G$  garantiza que se puede elegir un arco en el paso 3 siempre que  $N' \neq N$**
- **¿Es único el árbol de expansión?**

# Algoritmo de los árboles de expansión

---

- El algoritmo nunca forma un ciclo, puesto que cada nuevo arco va a un nuevo nodo
- $T = (N', A')$  es un árbol en cada paso del algoritmo, dado que  $T$  está siempre conectado y cada vez que añadimos un arco añadimos también un nodo
- Teorema: si  $G$  es un grafo conexo de  $n$  nodos, entonces:
  - 1)  $G$  contiene al menos  $n-1$  arcos
  - 2)  $G$  contiene un árbol de expansión
  - 3) Si  $G$  contiene exactamente  $n-1$  arcos,  $G$  es un árbol de expansión

# Algoritmos distribuidos para detectar árboles de expansión

---

- 1) Un nodo fijo envía un mensaje de "inicio" por cada arco adyacente del grafo
- 2) Cada uno de los otros nodos marca el primer arco en el que se ha recibido el mensaje de inicio como un arco del árbol de expansión y, a continuación, envía un mensaje de "inicio" por cada uno de los otros arcos:
  - Esto es una implementación distribuida del algoritmo general de los árboles de expansión
  - Presenta varios problemas que comparten muchos otros algoritmos de este tipo:
    - a) ¿Quién elige el nodo de inicio?
    - b) ¿Cuándo termina el algoritmo?
    - c) El árbol resultante es de algún modo aleatorio

# Árbol de expansión de peso mínimo

---

- Dado un grafo con pesos asignados a cada arco, encontrar un árbol de expansión de peso total mínimo (MST)
- Definir un "fragmento" como subárbol de un MST
- Teorema:
  - Dado un fragmento  $F$  de un MST, sea  $a(i,j)$  un arco de peso mínimo que sale desde  $F$ , donde  $j$  no se encuentra en  $F$
  - Entonces,  $F$  extendido con el arco  $a(i,j)$  y el nodo  $j$  es un fragmento
- Demostración:
  - Sea  $M$  el MST que no incluye  $a(i,j)$
  - Puesto que  $a(i,j)$  no forma parte de  $M$ , al añadirlo a  $M$  se crea un ciclo. Debe haber algún enlace en el ciclo  $b \neq a$  que sale desde  $F$
  - Al eliminar  $b$  y añadir  $a$  se crea un nuevo árbol de expansión. Dado que el peso de  $b$  no puede ser menor que el de  $a$ ,  $M'$  tiene que ser un MST:
    - Si el peso de  $a =$  peso de  $b$ , entonces ambos son MST; de lo contrario,  $M$  podría no ser un MST

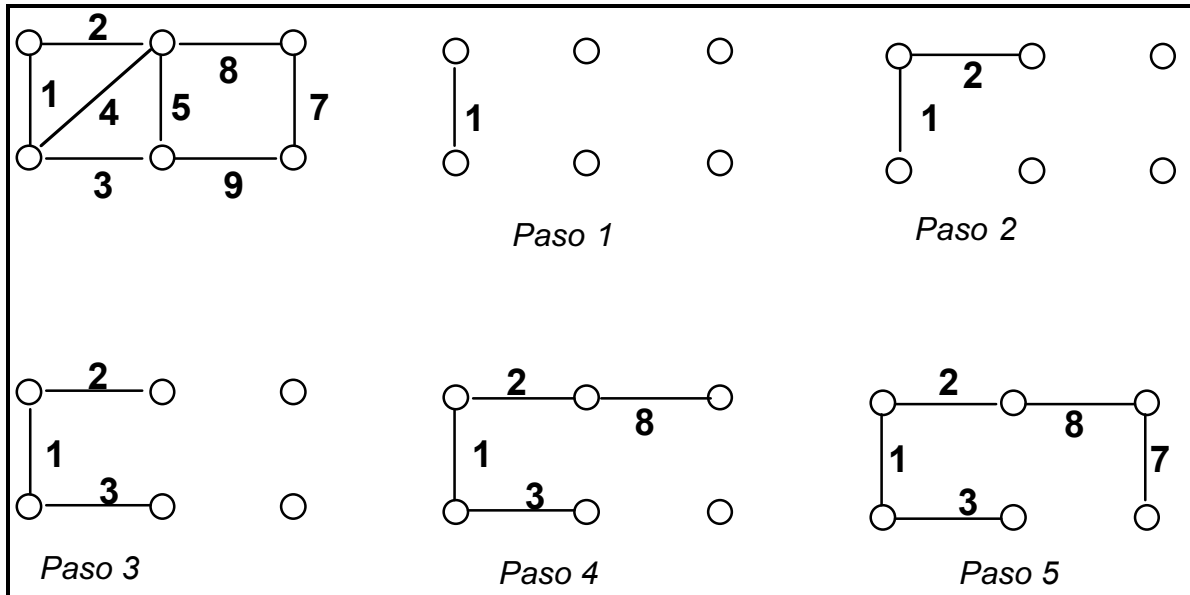
# Algoritmos MST

---

- **Pasos de un algoritmo MST genérico:**
  - Dado un conjunto de subárboles de un MST (llamados fragmentos) añadir un arco de salida de peso mínimo a algún fragmento
- **Prim-Dijkstra: empezar con un sólo nodo arbitrario como fragmento**
  - Añadir un arco de salida de peso mínimo
- **Kruskal: empezar con cada nodo como fragmento**
  - Añadir el arco de salida de peso mínimo, minimizado sobre todos los fragmentos

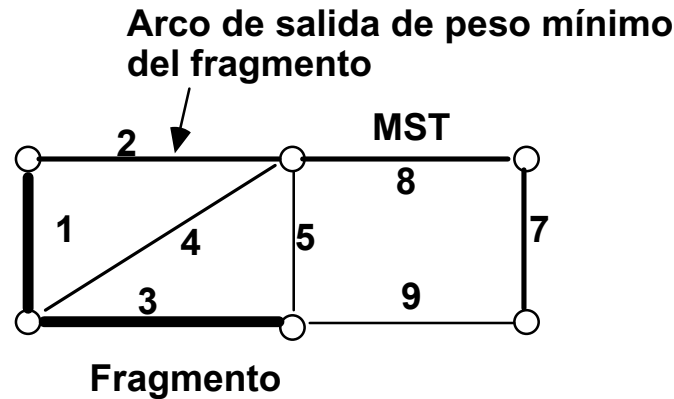
# Algoritmo de Prim-Dijkstra

---



# Algoritmo de Kruskal

---



- **Suponer que los arcos de peso 1 y 3 son un fragmento:**
  - Analizar cualquier árbol de expansión utilizando estos arcos y el arco de peso 4, que es un arco de salida del fragmento
  - Suponer que el árbol de expansión no utiliza el arco de peso 2
  - Eliminar el arco de peso 4 y añadir el arco de peso 2 da lugar a otro árbol de menor peso
  - De este modo, un arco de salida de peso mínimo del fragmento debe estar en el MST