

# Técnicas para Big Data

Clase 08 - Graph Analytics

# Outline

## **¿Por qué Graph Analytics?**

Camino más corto

Pagerank

Contar triángulos

Otros algoritmos

# Graph Analytics

Actualmente, sabemos cómo responder consultas sobre grafos gracias a los lenguajes de consultas como SPARQL y Neo4J

¿Es esta la única información que yo puedo obtener desde un grafo?

# Graph Analytics

Además de lo que sabemos, podemos aprovechar la estructura y forma del grafo para obtener aún más información

Por ejemplo, podríamos preguntar:

- ¿Cuál es el nodo más importante de la red?
- ¿Cuáles son las comunidades en una red?
- ¿Qué tan disperso es el grafo?
- ...

# Graph Analytics

Todo esto lo podemos lograr con algoritmos de **Graph Analytics**, que nos permiten responder preguntas como esta

# Outline

¿Por qué Graph Analytics?

**Camino más corto**

Pagerank

Contar triángulos

Otros algoritmos

# Camino más corto

Algoritmo de Dijkstra

# Camino más corto

Algoritmo de Dijkstra

Explora el camino más corto desde un nodo de origen hasta todos los otros nodos del grafo

# Camino más corto

Algoritmo de Dijkstra

Explora el camino más corto desde un nodo de origen hasta todos los otros nodos del grafo

# Camino más corto

Algoritmo de Dijkstra

Explora el camino más corto desde un nodo de origen hasta todos los otros nodos del grafo

Falla con aristas de costos negativos

1. Elegir los nodos de origen y destino, y agregar el nodo de origen al conjunto de nodos *resueltos*.
2. Del nodo de origen, buscar en amplitud los vecinos más cercanos y guardar el largo del camino hacia cada nodo.
3. Tomar el nodo con el camino más corto y marcarlo como resuelto, porque ya sabemos cuál es el camino más corto hasta él.
4. Visitar vecinos y guardar el largo acumulado del camino más corto hasta el nodo. No visitar ningún nodo que ya esté resuelto.
5. Repetir los pasos 3 y 4 hasta que el nodo de destino esté resuelto.

1. Sea  $D$  un arreglo de distancias.  $D_s = 0$ ,  $D_i$  es infinito para todo  $i$  distinto de  $s$ .
2. Creamos una cola de prioridad  $PQ$ , donde cada nodo  $i$  tiene prioridad  $D_i$ , y un conjunto de nodos vistos  $S$ .
3. Sea  $a = s$ .
4. Para cada vecino  $a$  ( $v_i$ ) que no está en  $S$ :
  1. Calculamos la distancia a  $v_i$  como:

$$dt(v_i) = Da + d(a, v_i).$$

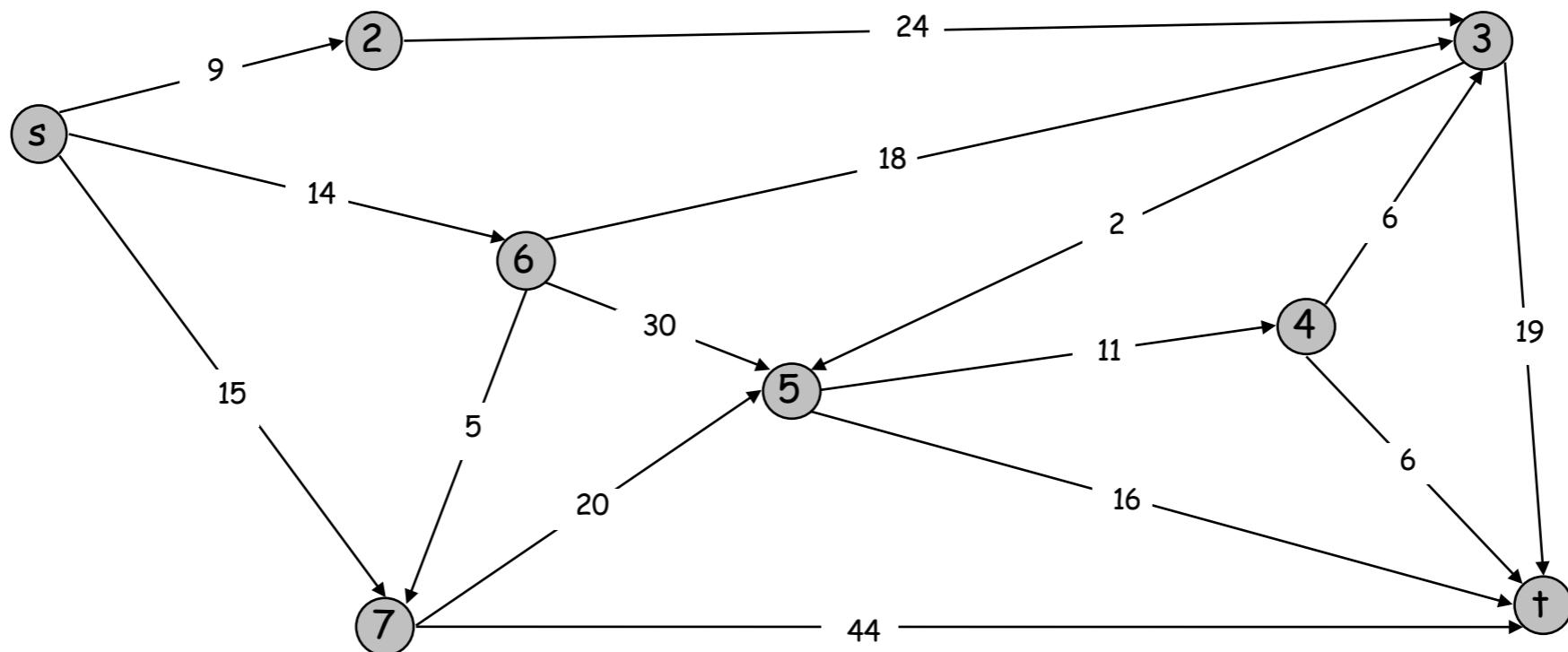
- Si  $dt(v_i) < Dv_i$ , entonces  $Dv_i = dt(v_i)$ .
2. Agregamos  $v_i$  a  $PQ$
  5. Agregamos  $a$  a  $S$ .
  6. Tomamos como próximo nodo actual el de menor valor en  $D$ , volvemos al paso 4 mientras existan nodos en  $PQ$ .

# Invariante

"Para todo nodo visitado  $v$ ,  $Dv$  es la distancia más corta desde el origen hacia él"

# Camino más corto

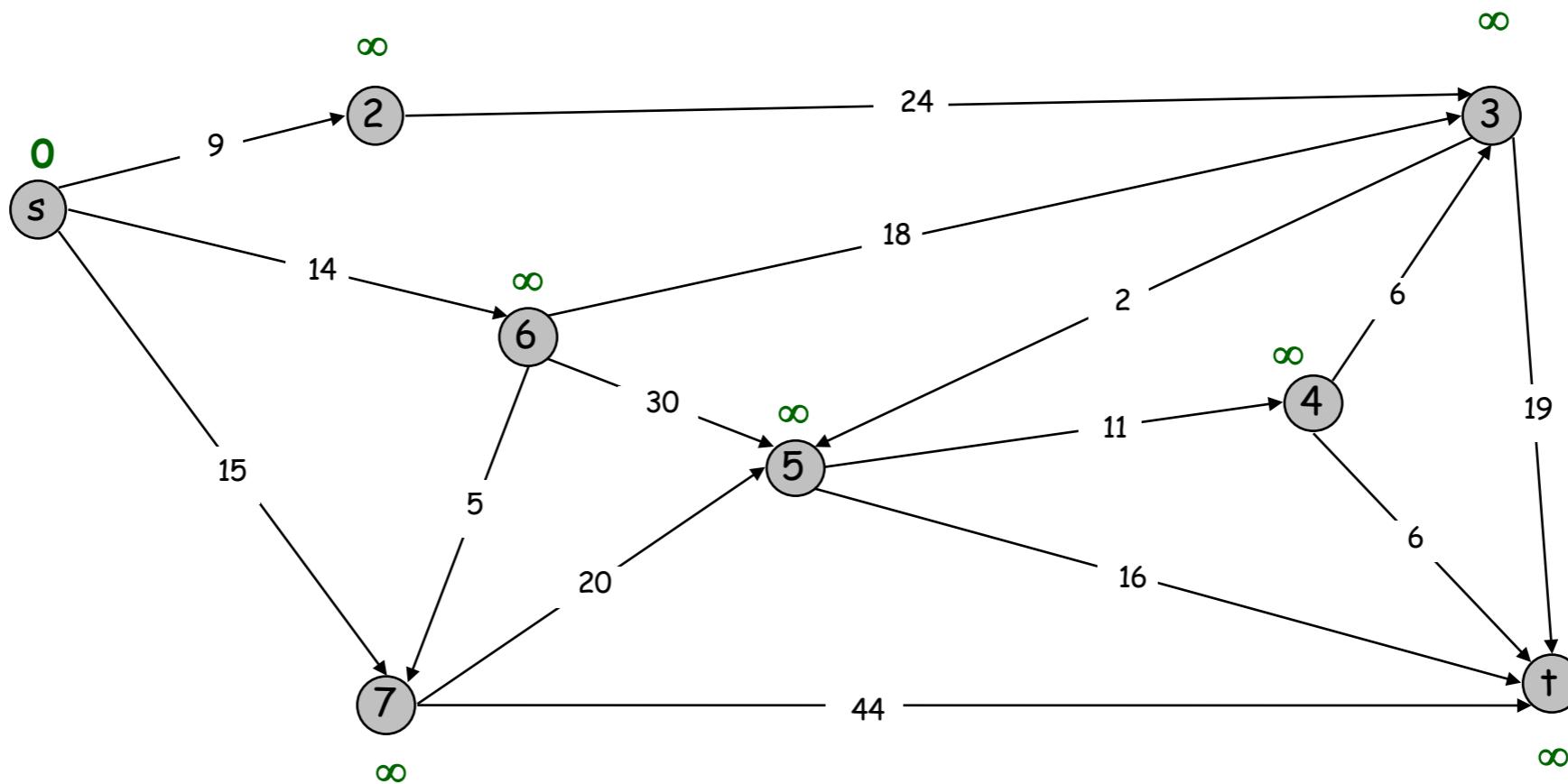
Encontrar la forma más barata de llegar desde  $s$  hasta  $t$



# Camino más corto

$$S = \{\}$$

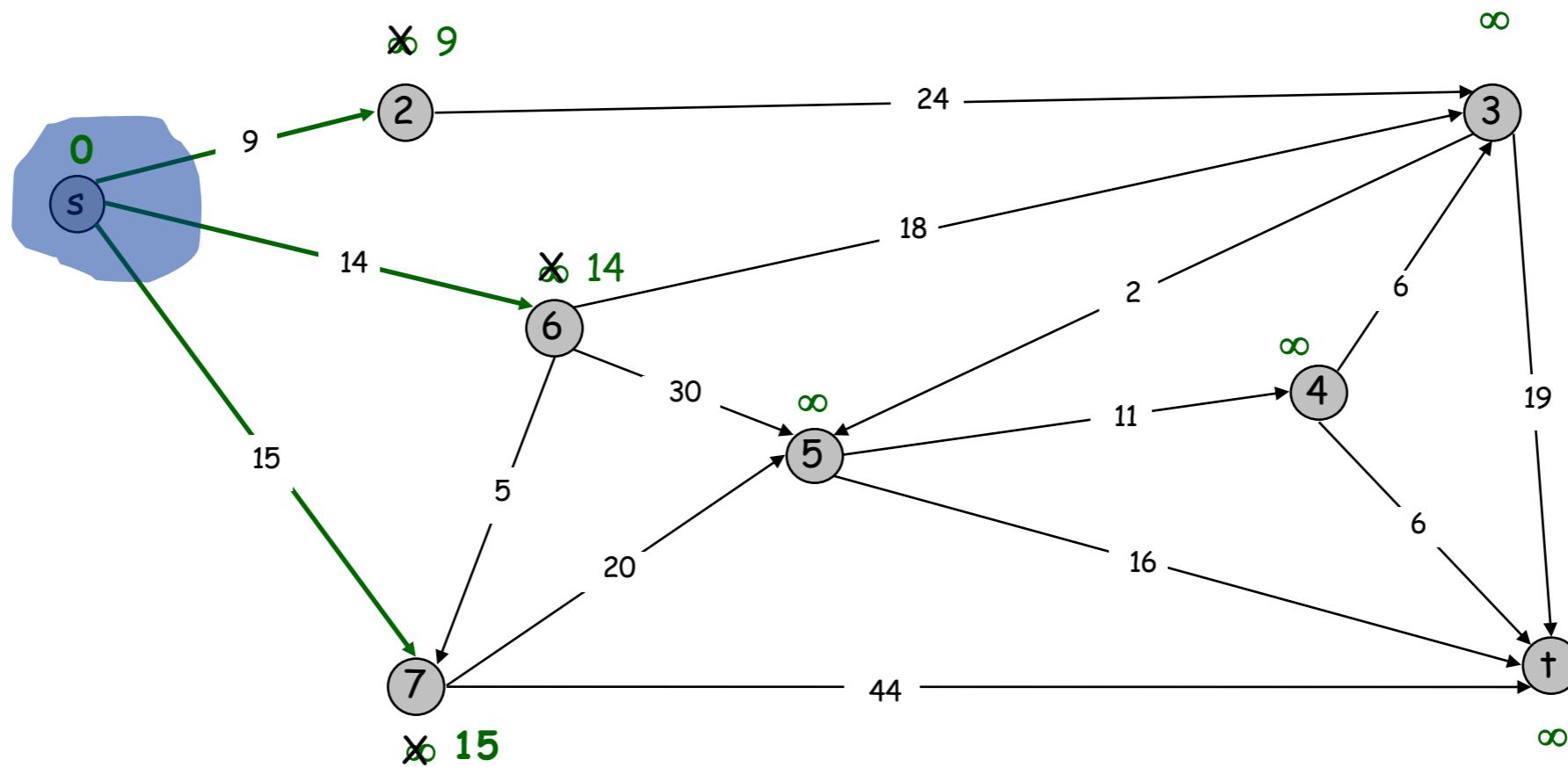
$$PQ = \{S\}$$



# Camino más corto

$$S = \{ s \}$$

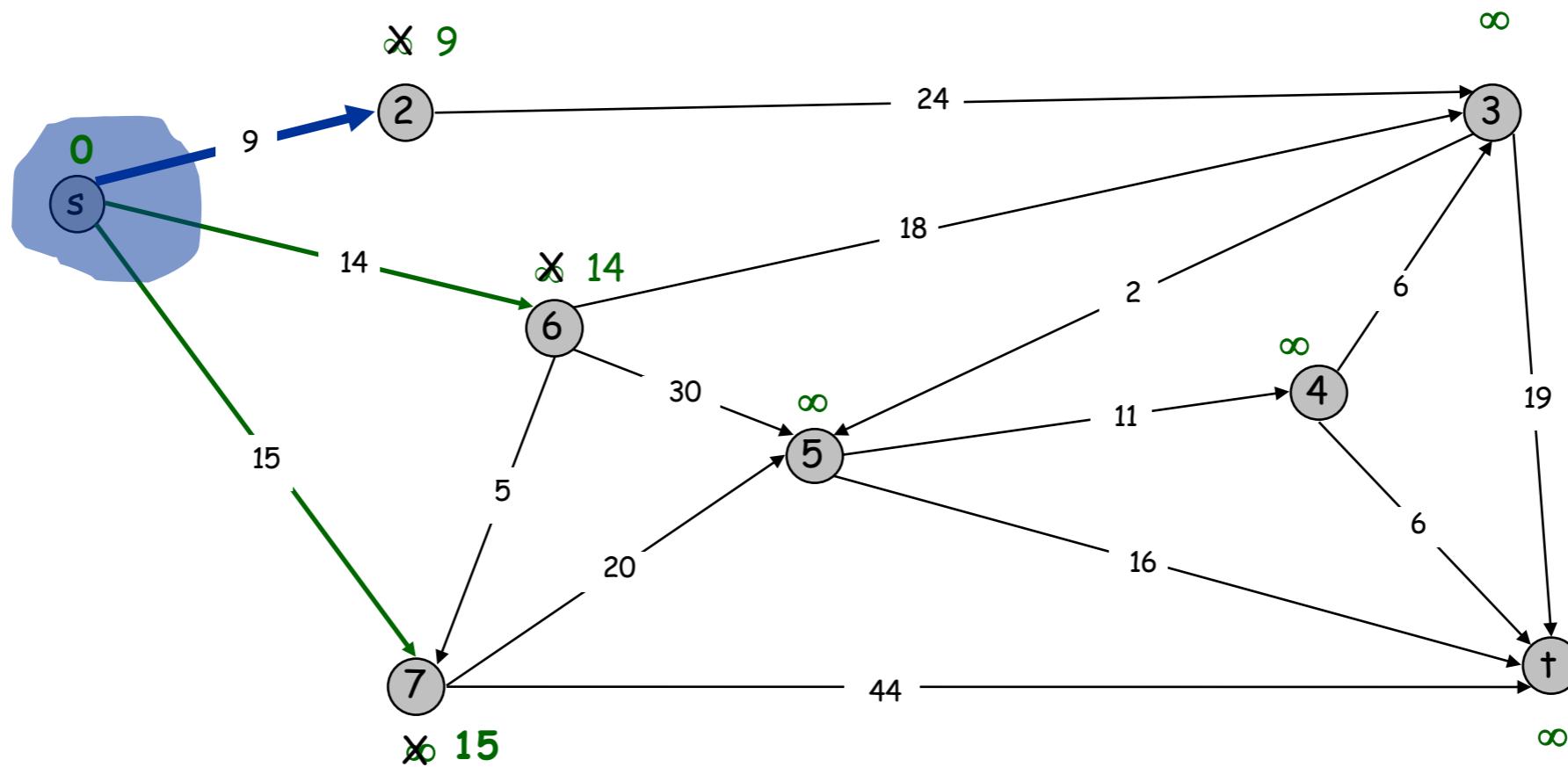
$$PQ = \{ 2, 6, 7 \}$$



# Camino más corto

$$S = \{ s \}$$

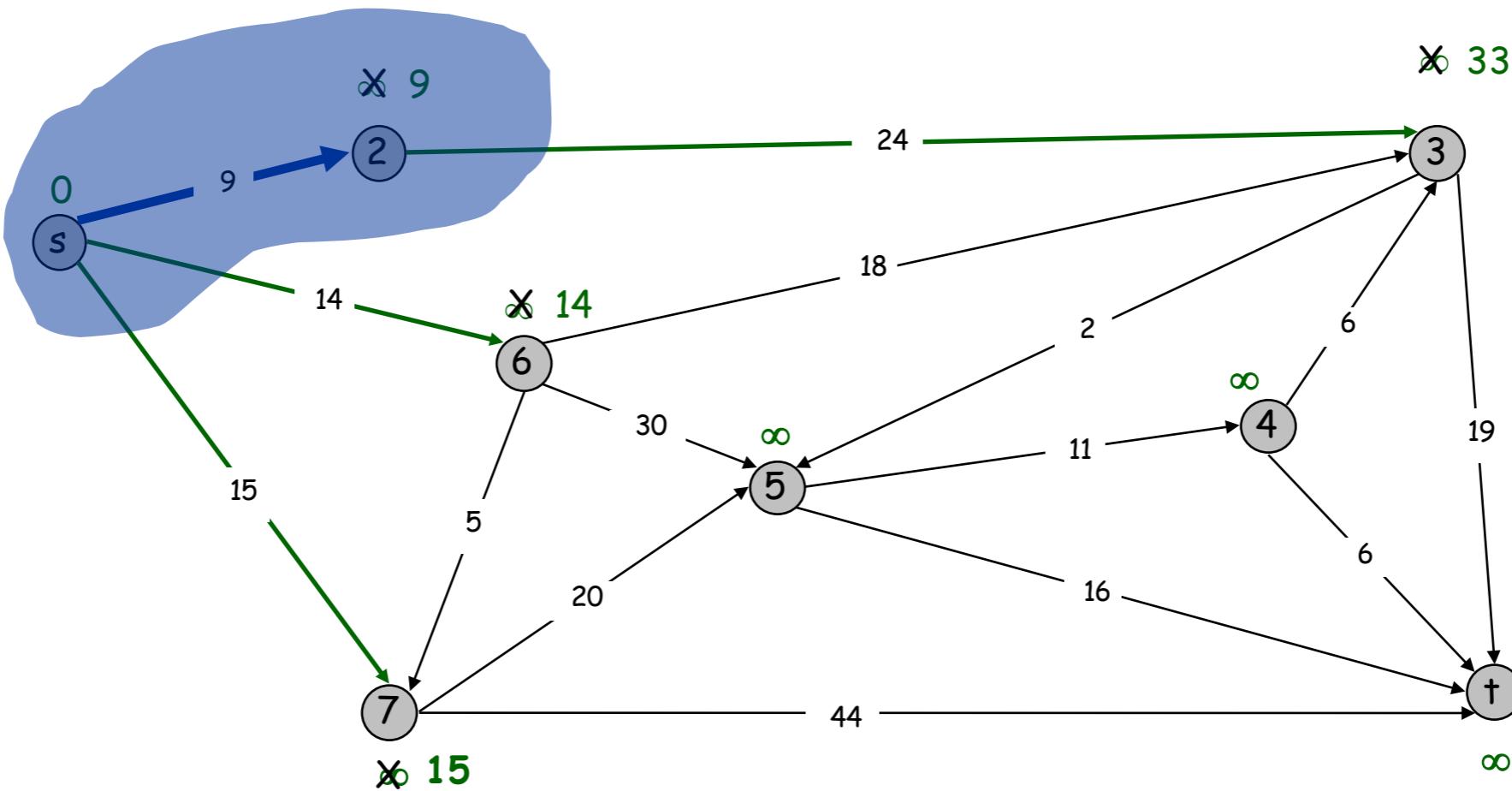
$$PQ = \{ 2, 6, 7 \}$$



# Camino más corto

$$S = \{ s, 2 \}$$

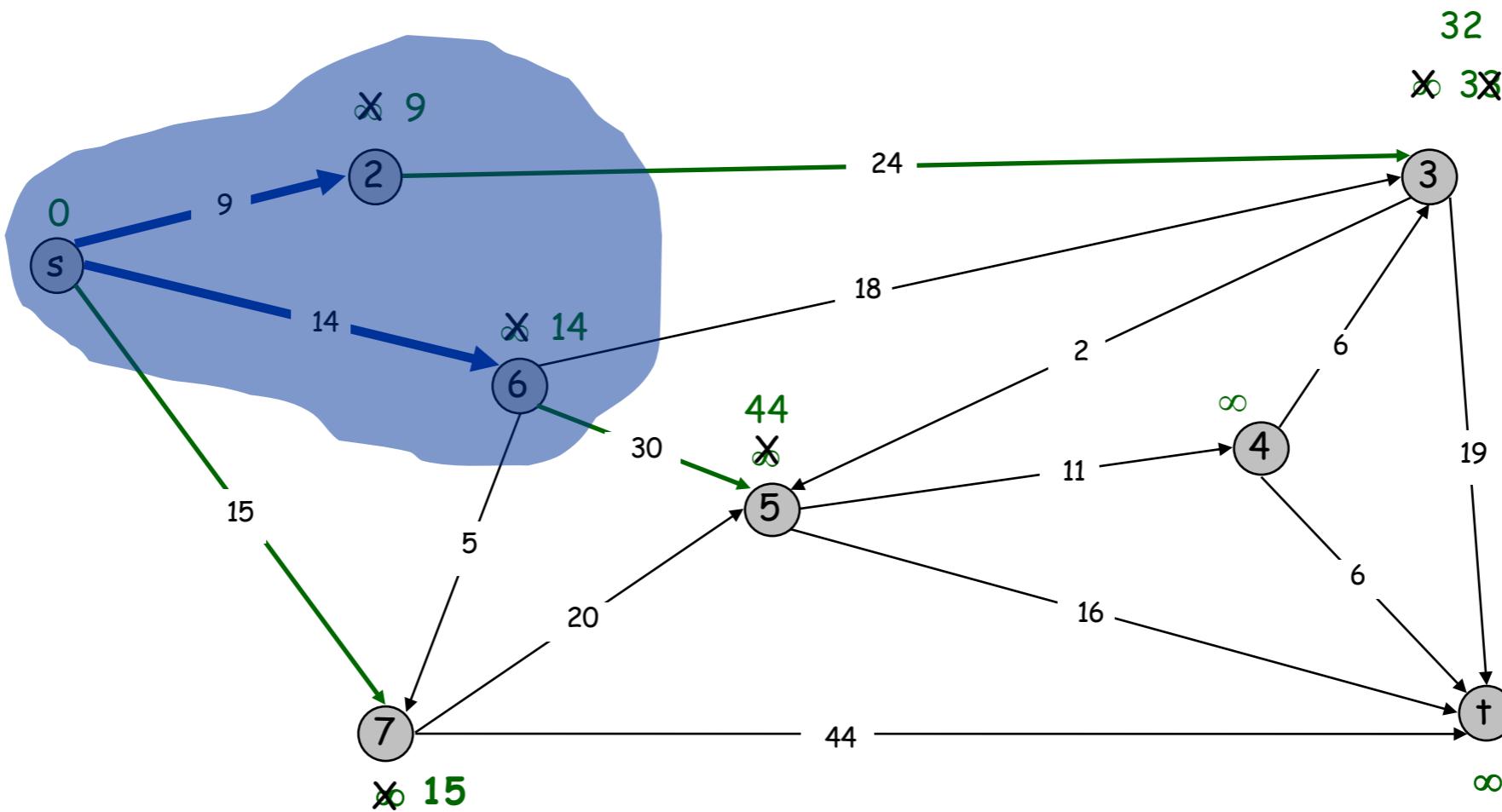
$$PQ = \{ 6, 7, 3 \}$$



# Camino más corto

$$S = \{ s, 2, 6 \}$$

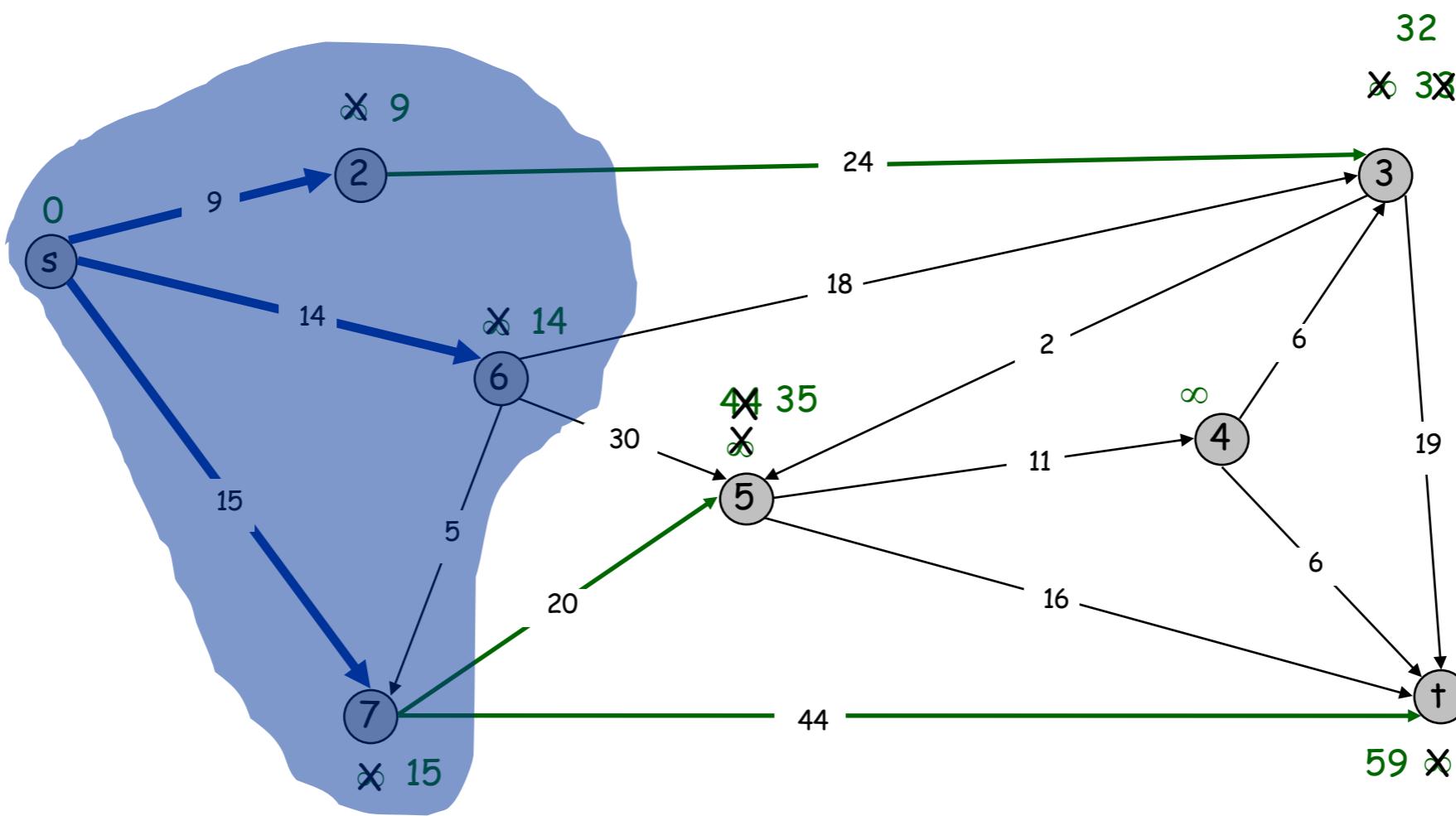
$$PQ = \{ 7, 3, 5 \}$$



# Camino más corto

$$S = \{ s, 2, 6, 7 \}$$

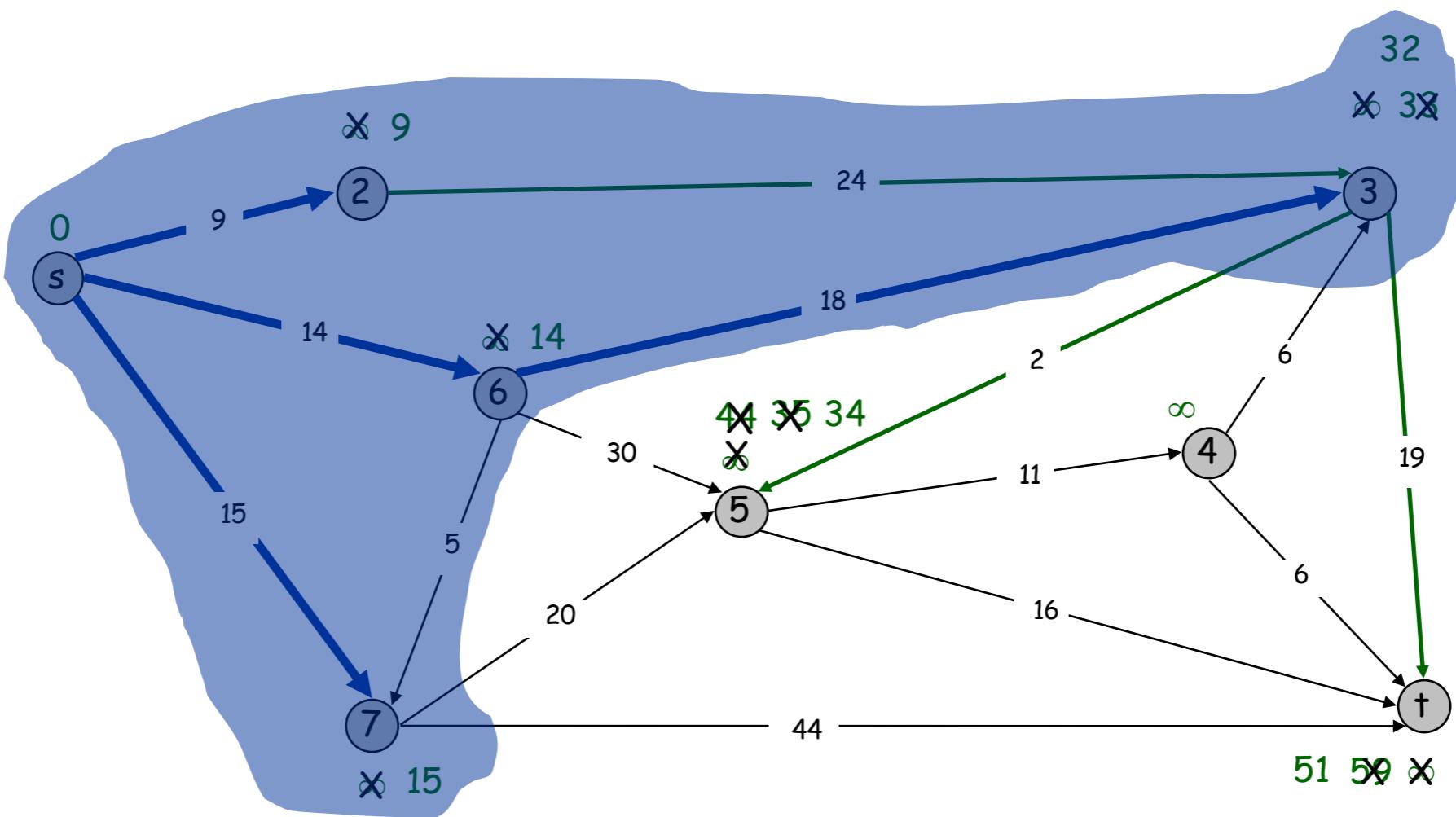
$$PQ = \{ 3, 5, t \}$$



# Camino más corto

$$S = \{ s, 2, 6, 7, 3 \}$$

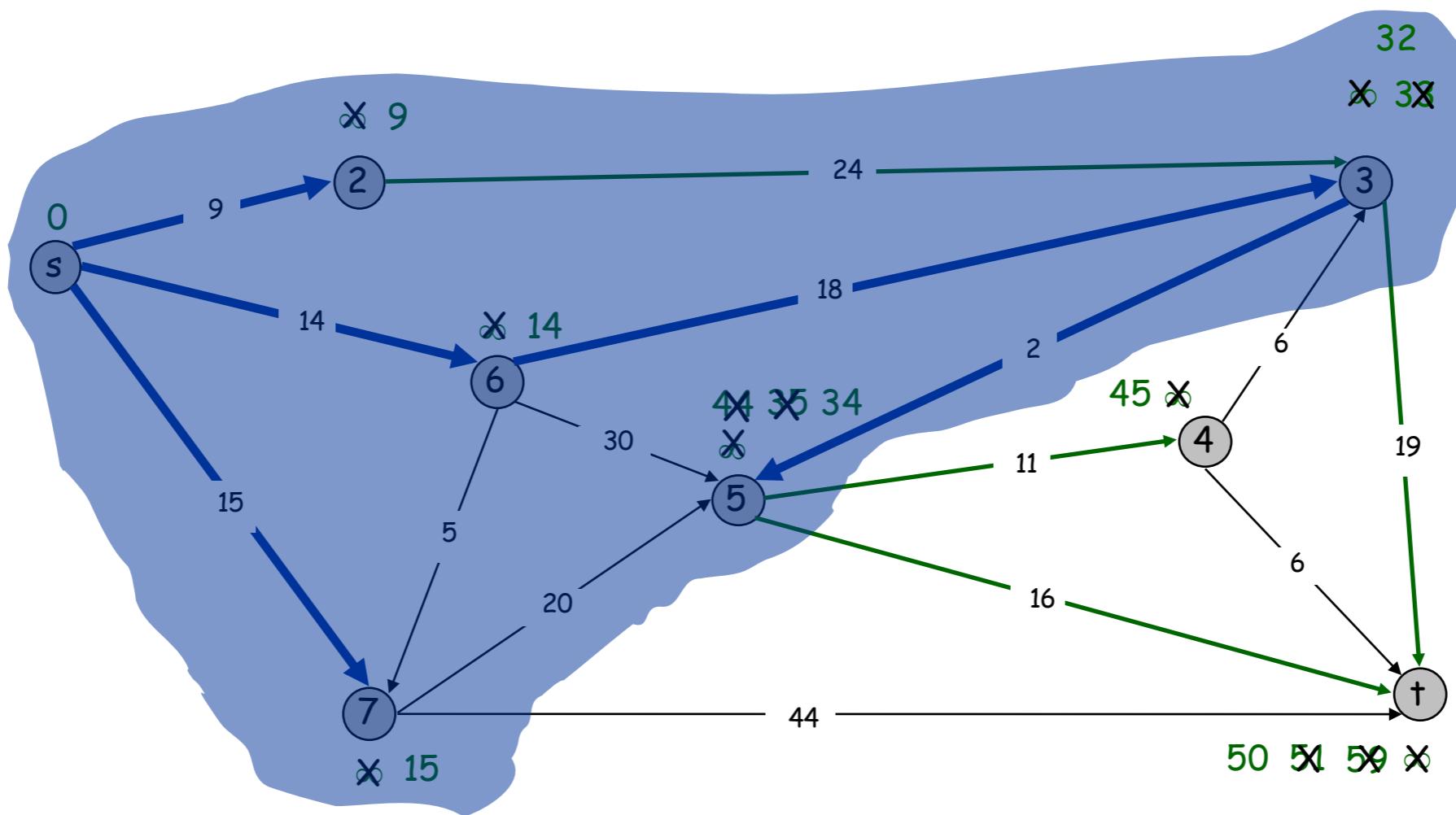
$$PQ = \{ 5, t \}$$



# Camino más corto

$$S = \{ s, 2, 6, 7, 3, 5 \}$$

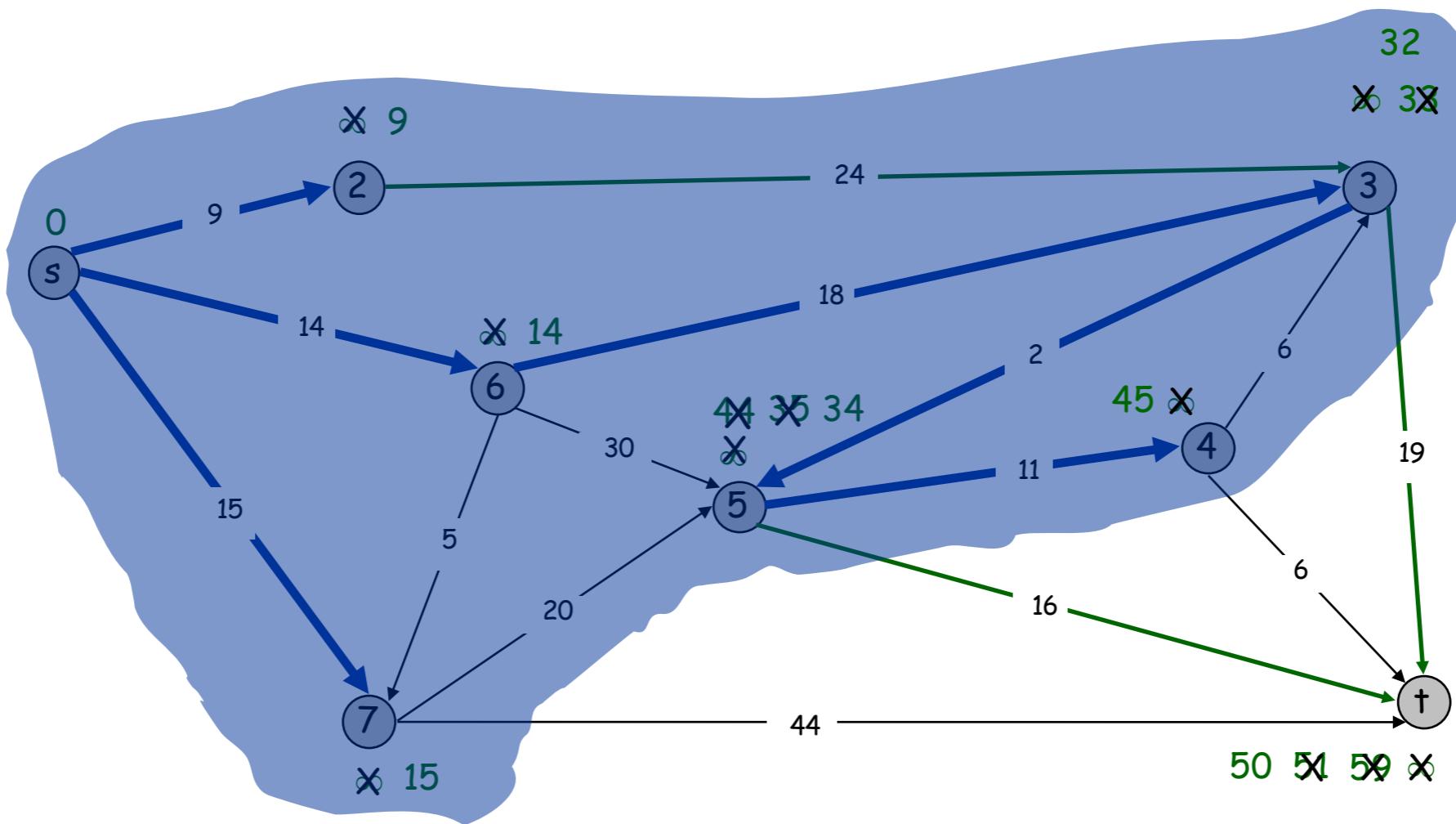
$$PQ = \{ 4, t \}$$



# Camino más corto

$$S = \{ s, 2, 6, 7, 3, 5, 4 \}$$

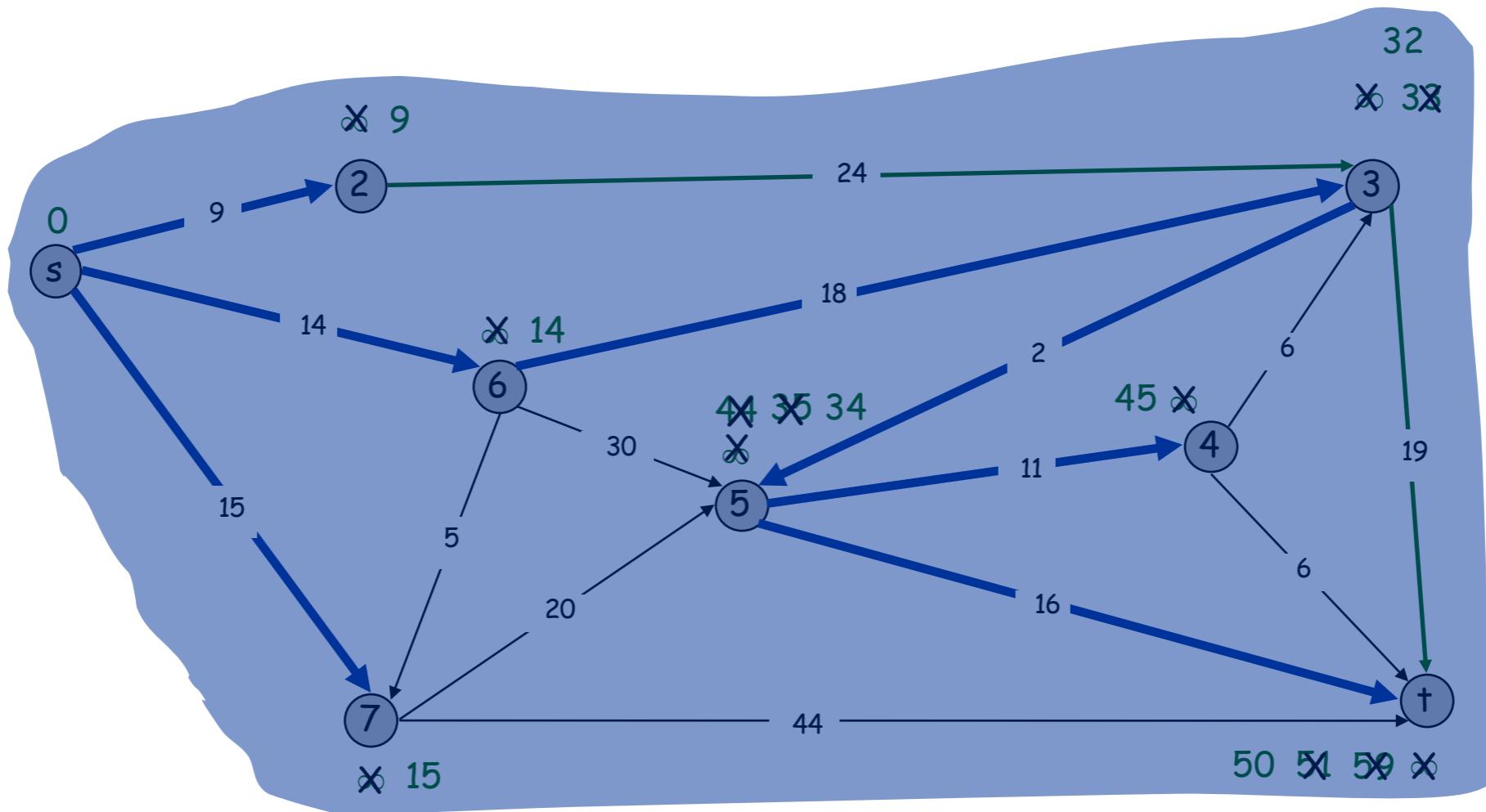
$$PQ = \{ t \}$$



# Camino más corto

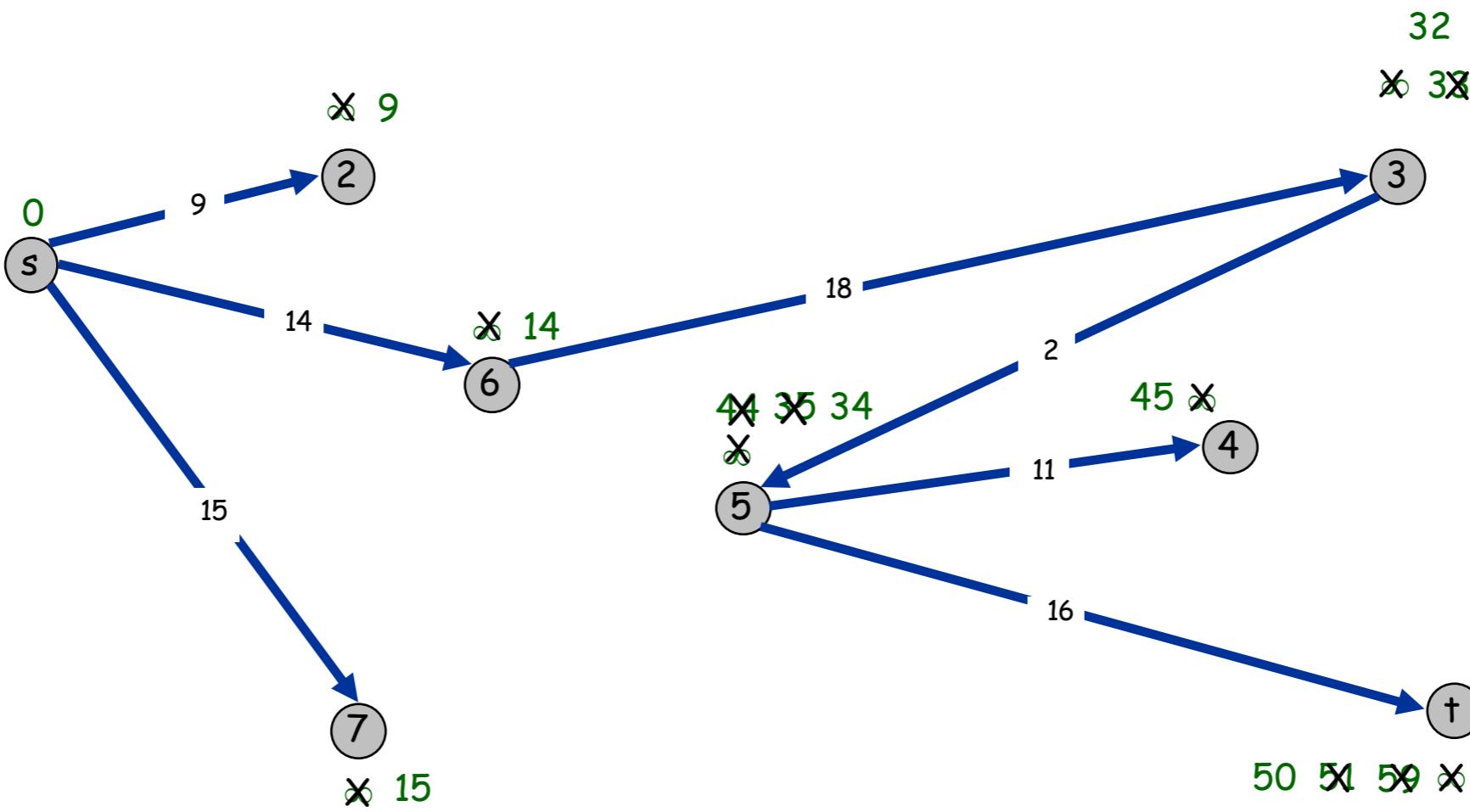
$$S = \{ s, 2, 6, 7, 3, 5, 4, t \}$$

$$PQ = \{ \}$$



Distancia de un nodo a todos los demás

# Distancia de un nodo a todos los demás



# Outline

¿Por qué Graph Analytics?

Camino más corto

**Pagerank**

Contar triángulos

Otros algoritmos

# Pagerank

*PageRank works by counting the number and quality of links to a page to determine a rough estimate of how important the website is. The underlying assumption is that more important websites are likely to receive more links from other websites*

# Pagerank

La idea es simular un navegante de la web aleatorio, que hace clicks y va cambiando de páginas

La idea es calcular en qué páginas es más probable que termine

El algoritmo incluye un *damping factor* **d**, que corresponde a la probabilidad que deje de hacer links y salte a una página aleatoriamente

# Pagerank

- **Entrada:** grafo
- **Salida:** distribución de probabilidad relativa de que un usuario haciendo clicks random termine en una cierta página web
- Loops (links de una página a sí misma) son ignorados. Múltiples links de una página a otra son considerados como un mismo arco

# Pagerank

- Inicialmente, todas las páginas tienen la misma probabilidad
- En cada iteración, cada página le entrega parte de su probabilidad a todas aquellas a las que tiene un link
- En cada iteración, la probabilidad de un nuevo salto se atenúa según el factor  $d$

# Pagerank

Algoritmo

En la iteración 0 definimos el PageRank para cada nodo como:

$$PR_0(n_i) = 1/N$$

Y para las siguientes iteraciones como:

$$PR_t(n_i) = \frac{1 - d}{N} + d \sum_{n_j \in In(n_i)} \frac{PR_{t-1}(n_j)}{Out(n_j)}$$

# Pagerank

Algoritmo

$$PR_t(n_i) = \frac{1 - d}{N} + d \sum_{n_j \in In(n_i)} \frac{PR_{t-1}(n_j)}{Out(n_j)}$$

PageRank del nodo i en la iteración t

Número de nodos en el grafo

Damping factor, usualmente 0.85

Número de aristas saliendo del nodo j

Nodos que tienen una arista entrando al nodo i

# Pagerank

## Algoritmo

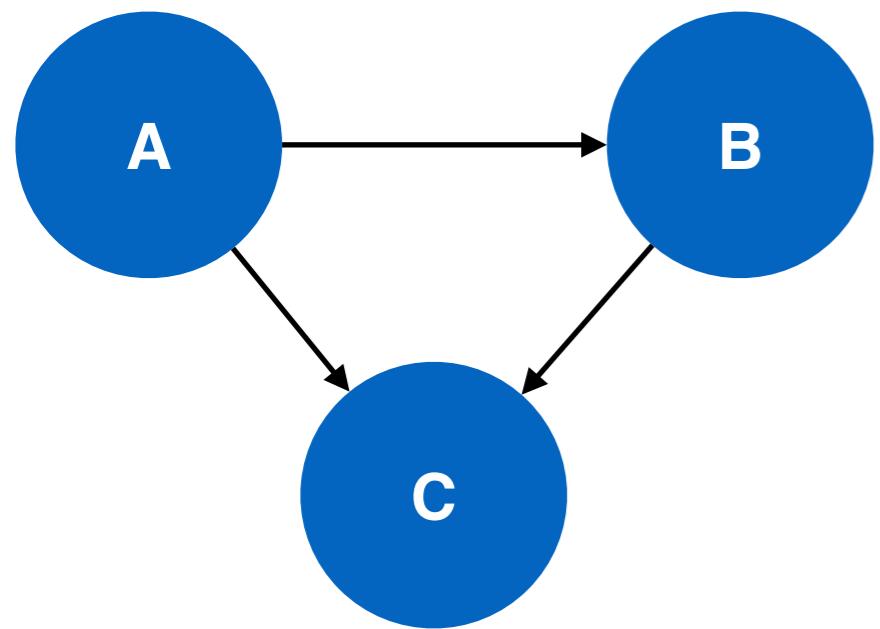
En cada iteración calculamos el siguiente valor:

$$\sqrt{(PR_t(nodo_1) - PR_{t-1}(nodo_1))^2 + \dots + (PR_t(nodo_N) - PR_{t-1}(nodo_N))^2}$$

Hasta que tome un valor "suficientemente pequeño"

# Pagerank

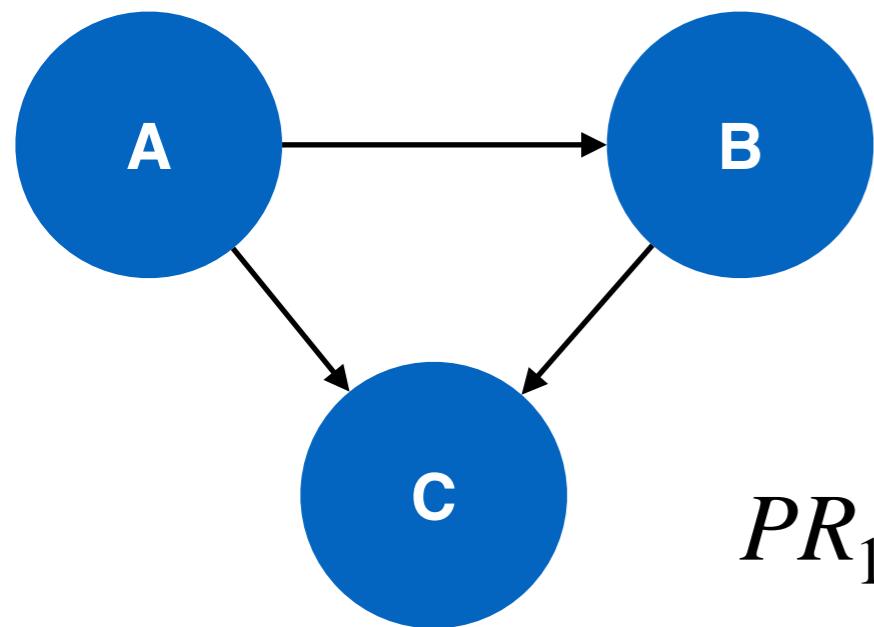
## Ejemplo



$$PR_0(A) = PR_0(B) = PR_0(C) = \frac{1}{3}$$

# Pagerank

## Ejemplo



$$PR_0(A) = PR_0(B) = PR_0(C) = \frac{1}{3}$$

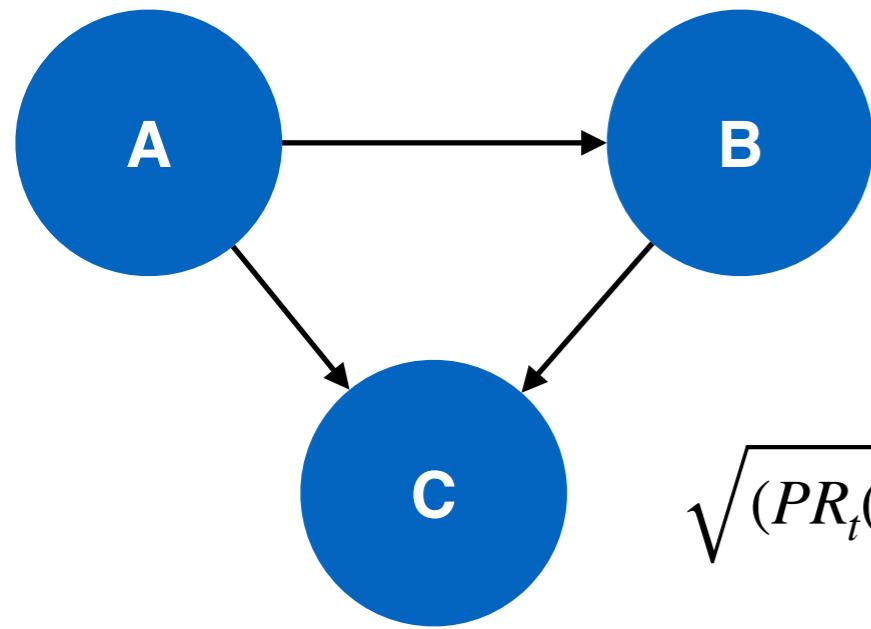
$$PR_1(A) = \frac{1 - 0.85}{3} + 0.85 \cdot 0$$

$$PR_1(B) = \frac{1 - 0.85}{3} + 0.85 \cdot \left( \frac{PR_0(A)}{2} \right)$$

$$PR_1(C) = \frac{1 - 0.85}{3} + 0.85 \cdot \left( \frac{PR_0(A)}{2} + \frac{PR_0(B)}{1} \right)$$

# Pagerank

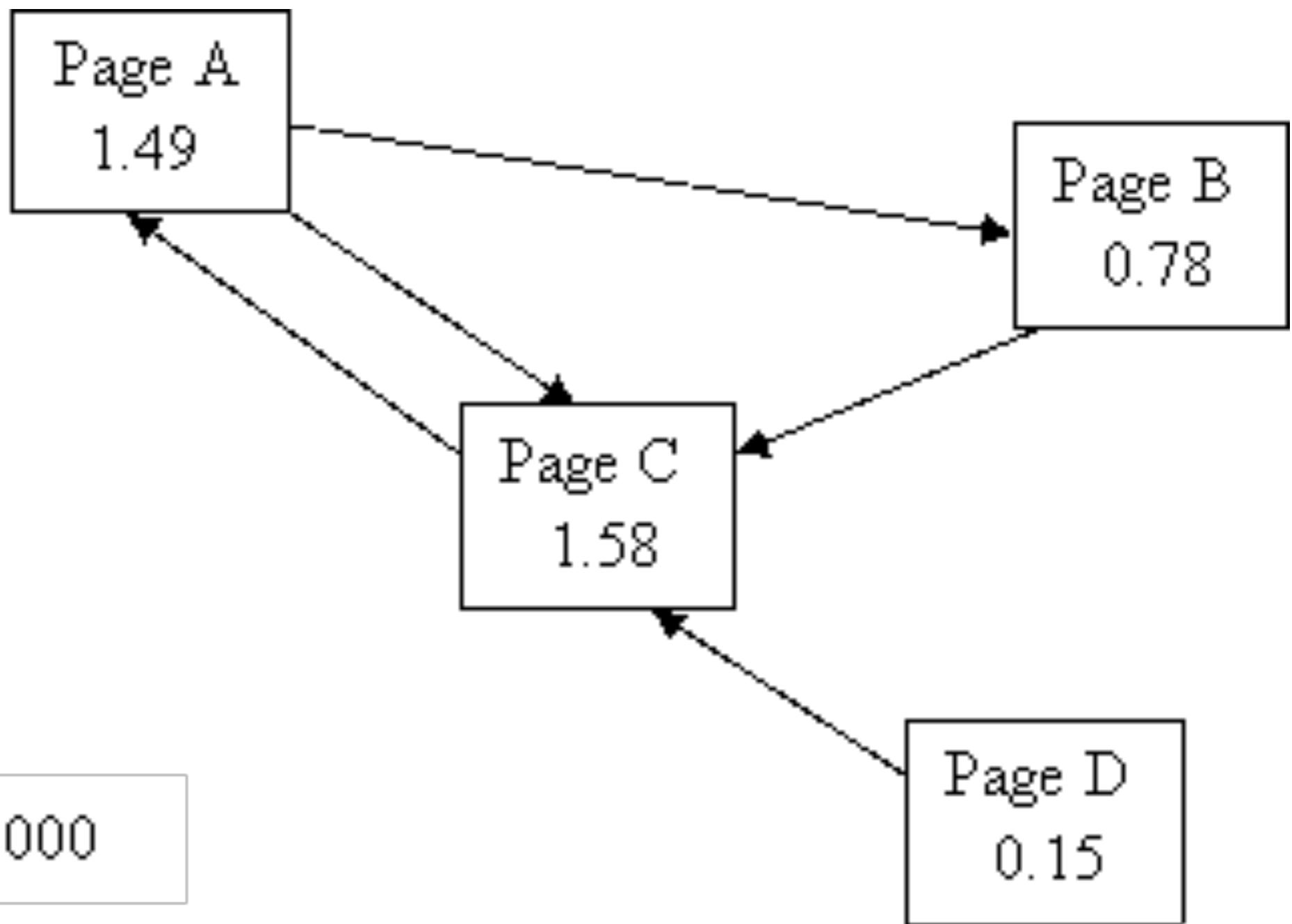
Ejemplo



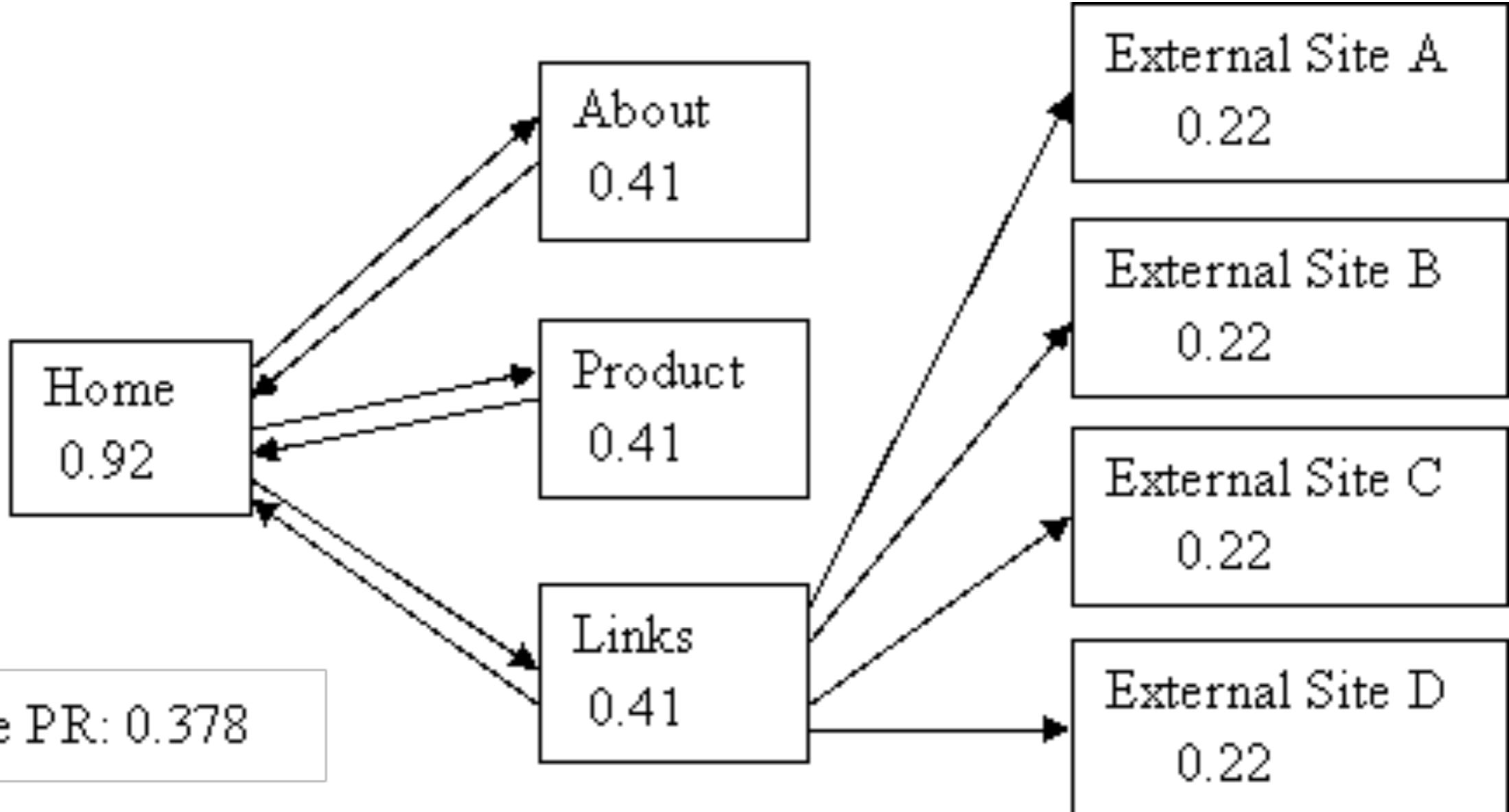
Iteramos hasta que el valor de:

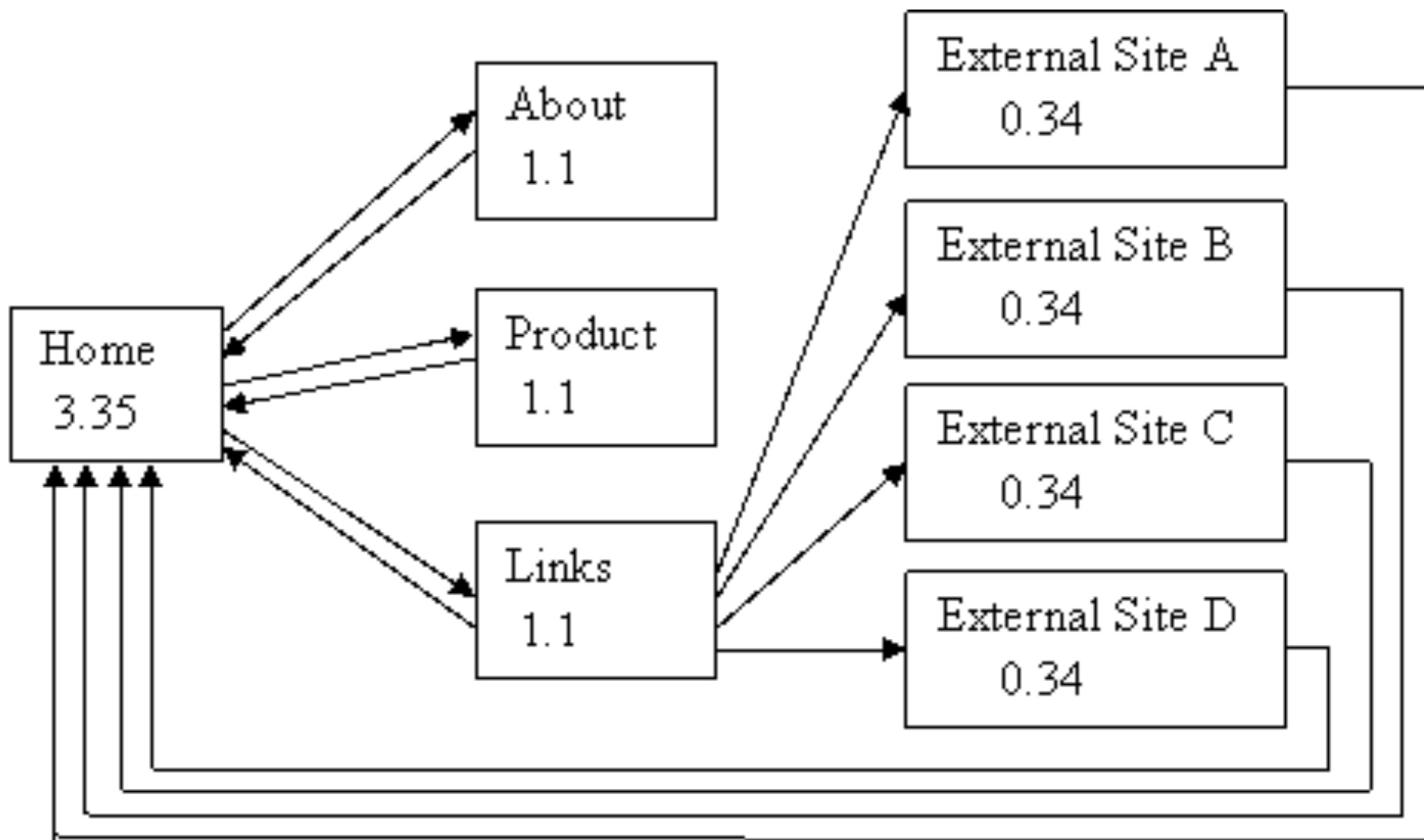
$$\sqrt{(PR_t(A) - PR_{t-1}(A))^2 + (PR_t(B) - PR_{t-1}(B))^2 + (PR_t(C) - PR_{t-1}(C))^2}$$

Sea menor a un número muy bajo  
(por ejemplo, 0.0001)



Average PR: 1.000





Average PR: 1.000

# ¿Qué pasa con la web?

# ¿Qué pasa con la web?

- Si  $d$  es muy grande, necesitamos muchas iteraciones para converger

# ¿Qué pasa con la web?

- Si  $d$  es muy grande, necesitamos muchas iteraciones para converger
- Si  $d$  es muy chico, todo converge a 1

# ¿Qué pasa con la web?

- Si  $d$  es muy grande, necesitamos muchas iteraciones para converger
- Si  $d$  es muy chico, todo converge a 1
- En el paper original, el grafo de la web converge aceptablemente en 52 iteraciones con  $d = 0.85$

# Outline

¿Por qué Graph Analytics?

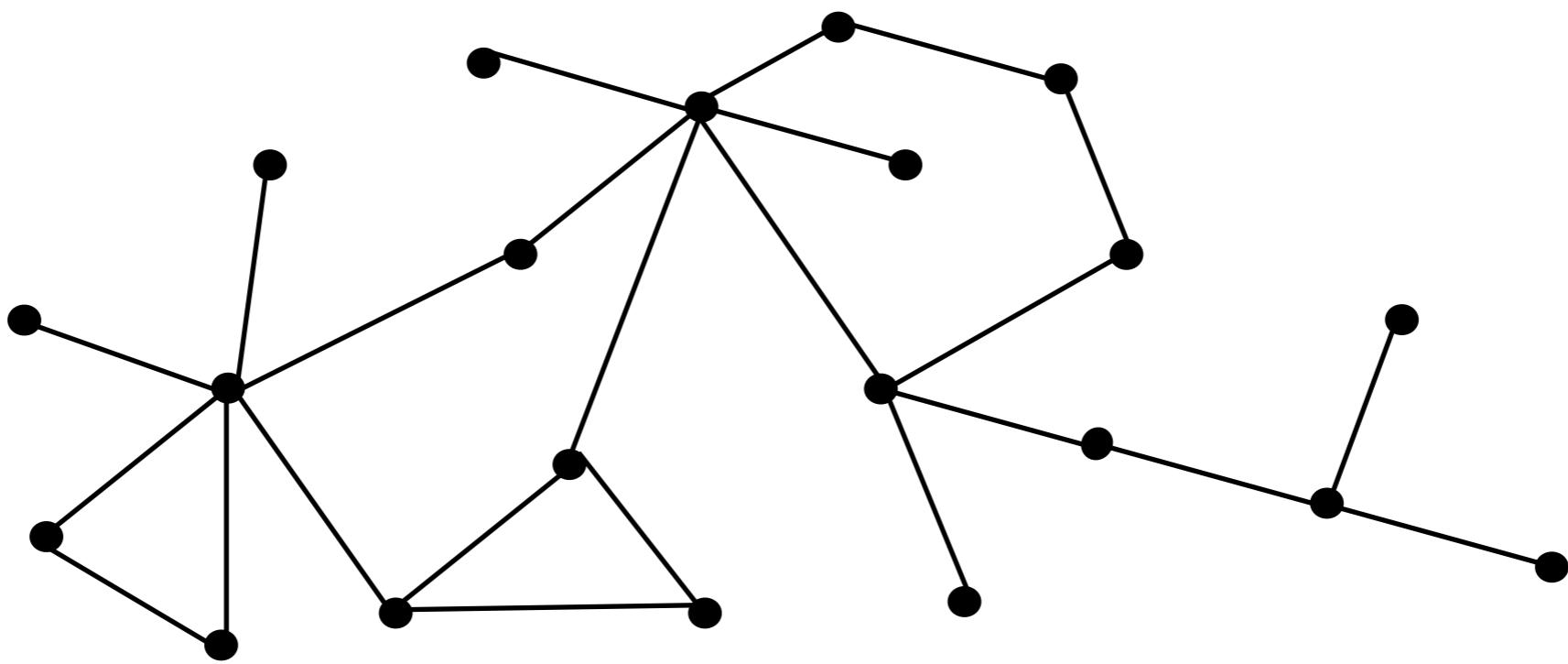
Camino más corto

Pagerank

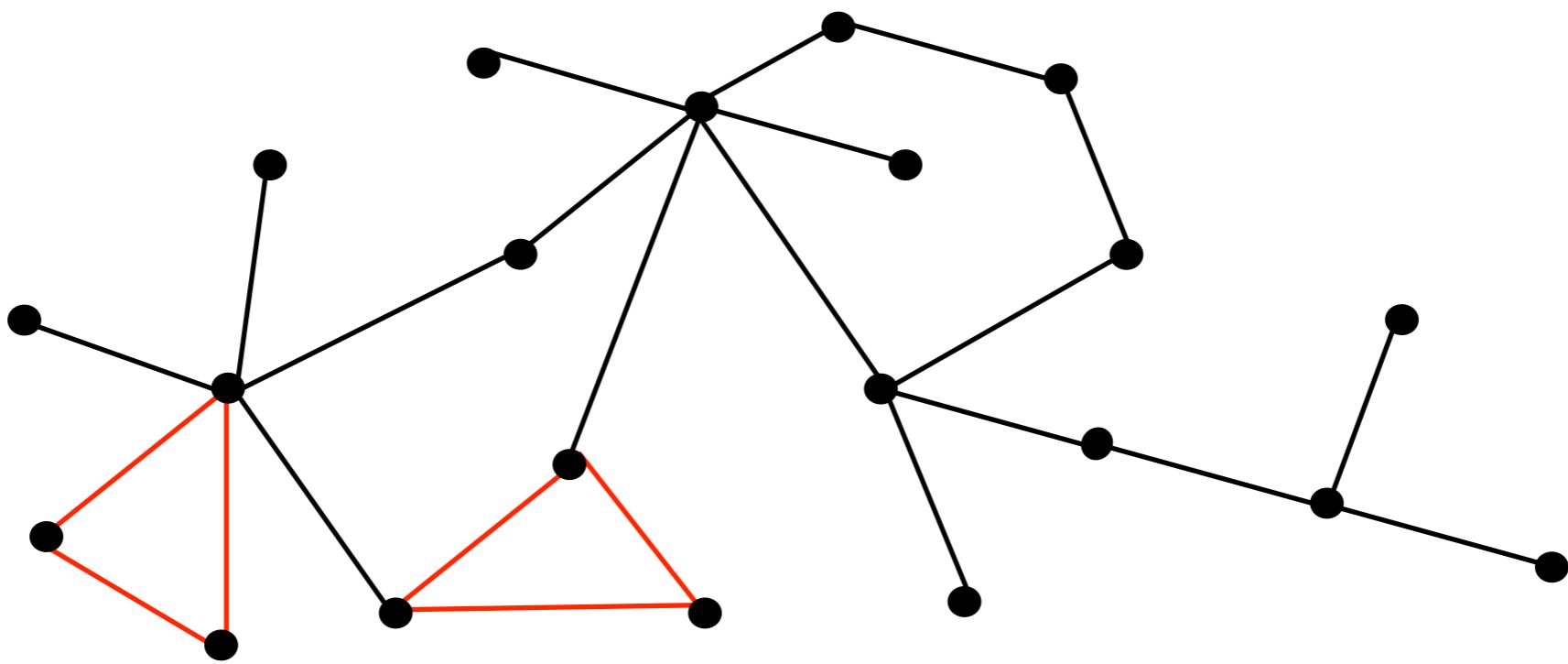
**Contar triángulos**

Otros algoritmos

# Contar Triángulos



# Contar Triángulos



# Contar Triángulos

Algoritmo obvio

```
triangulos = 0
for i in V:
    for j in V:
        for k in V:
            if (i, j) in A and
                (j, k) in A and
                (k, i) in A:
                    triangulos+=1
```

# Contar Triángulos

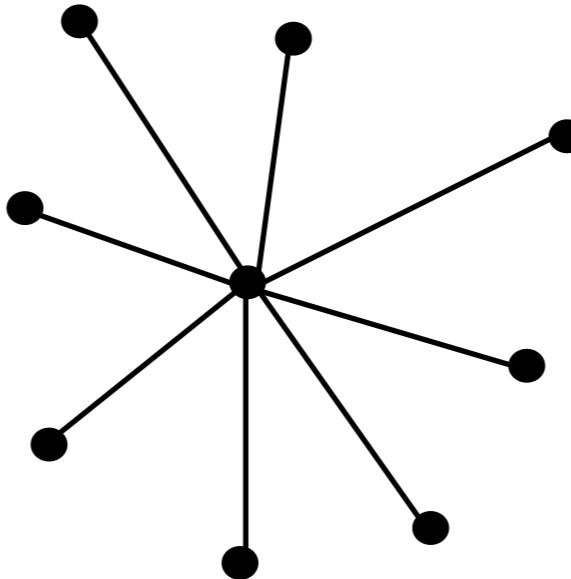
Algoritmo obvio

El algoritmo obvio tiene una muy mala *performance* en la práctica, ¿qué pasa en el siguiente grafo?

# Contar Triángulos

Algoritmo obvio

El algoritmo obvio tiene una muy mala *performance* en la práctica, ¿qué pasa en el siguiente grafo?



# Un mejor algoritmo

Enumerar pares de nodos adyacentes

Para cada nodo  $v$  en  $V$ :

Para cada par  $u, w$  tales que  $u$  y  $w$  son vecinos y distintos de  $v$ :

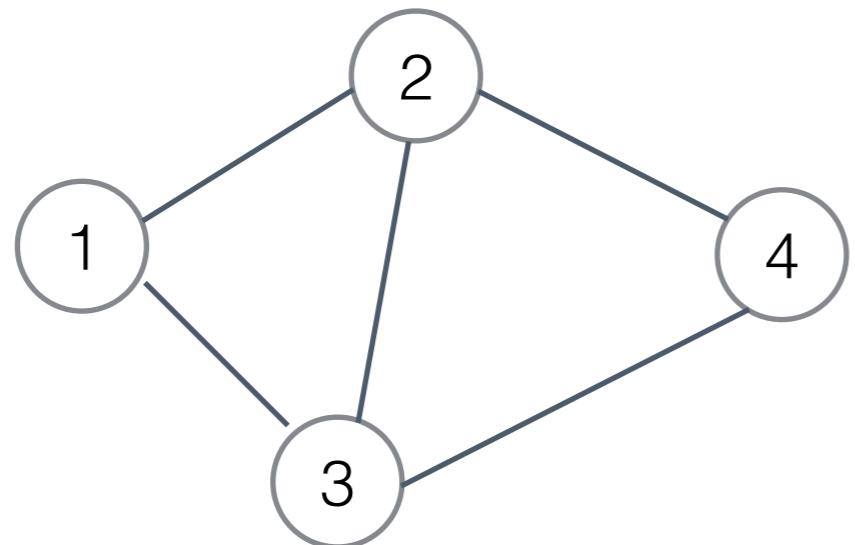
Si hay un arco entre  $u$  y  $w$ , entonces aumento en 1 la cuenta de triángulos que incluyen  $v$

# Un mejor algoritmo

```
triangulos = 0
for v in V:
    for u, w in vecinos(v):
        if u==w: continue
        if (u, w) in A:
            triangulos+=1
```

# Matrices de adyacencia

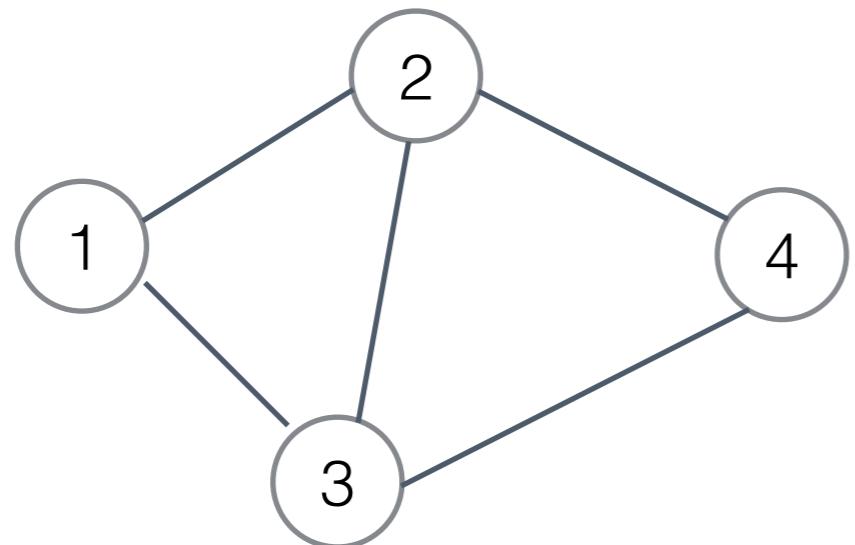
Si hay un arco de  $u$  a  $w$ , entonces  $M_{u,w} = 1$



$$M = \begin{array}{|c|c|c|c|} \hline 0 & 1 & 1 & 0 \\ \hline 1 & 0 & 1 & 1 \\ \hline 1 & 1 & 0 & 1 \\ \hline 0 & 1 & 1 & 0 \\ \hline \end{array}$$

# Matrices de adyacencia

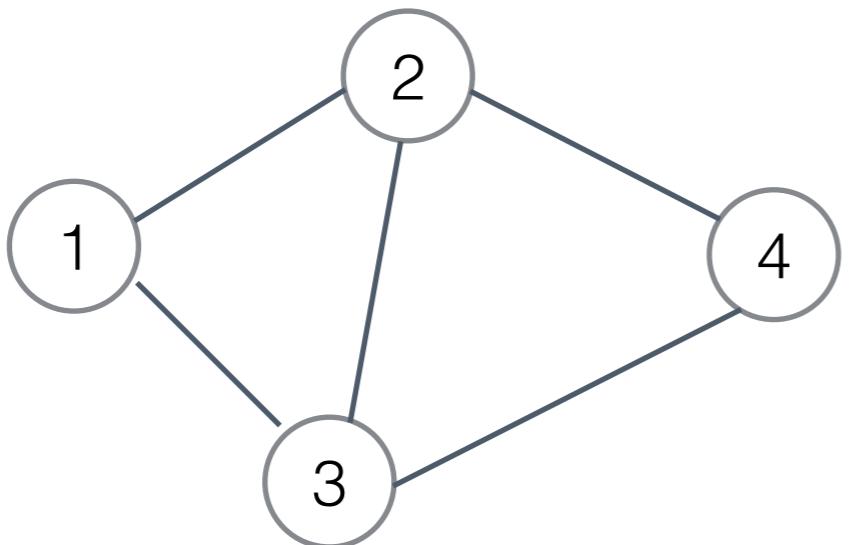
Si hay un arco de  $u$  a  $w$ , entonces  $M_{u,w} = 1$



$$M^2 = \begin{array}{|c|c|c|c|} \hline 2 & 1 & 1 & 2 \\ \hline 1 & 3 & 2 & 1 \\ \hline 1 & 2 & 3 & 1 \\ \hline 2 & 1 & 1 & 2 \\ \hline \end{array}$$

# Matrices de adyacencia

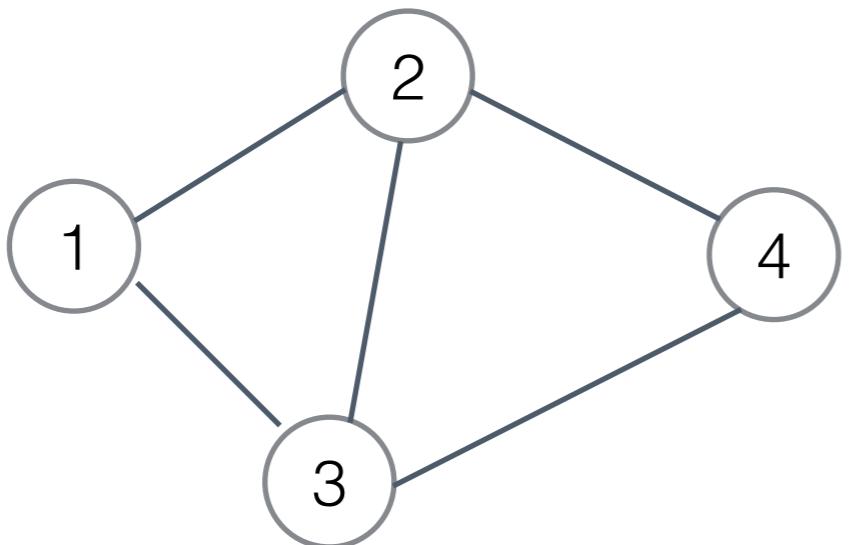
Si hay un arco de  $u$  a  $w$ , entonces  $M_{u,w} = 1$



$$M^3 = \begin{array}{|c|c|c|c|} \hline 2 & 5 & 5 & 2 \\ \hline 5 & 4 & 5 & 5 \\ \hline 5 & 5 & 4 & 5 \\ \hline 2 & 5 & 5 & 2 \\ \hline \end{array}$$

# Matrices de adyacencia

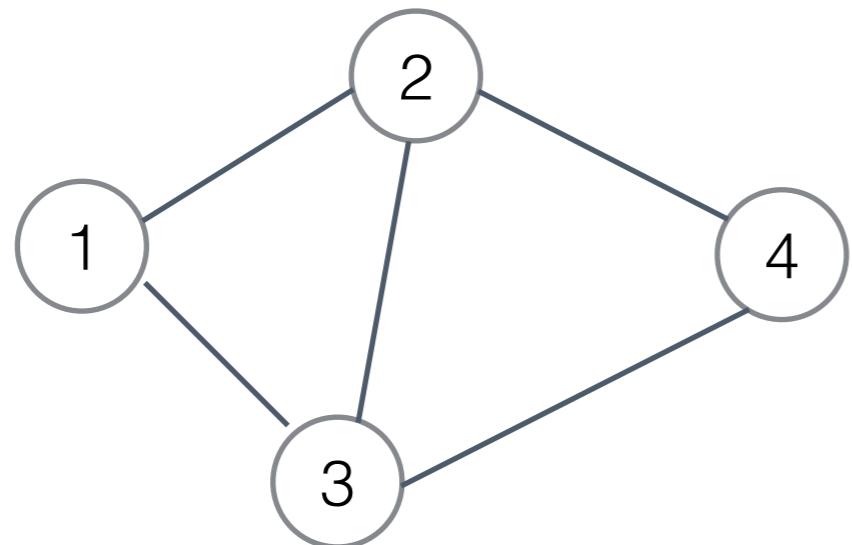
Si hay un arco de u a w, entonces  $M_{u,w} = 1$



$$M^3 = \begin{array}{|c|c|c|c|} \hline & 2 & 5 & 5 & 2 \\ \hline 2 & 5 & 4 & 5 & 5 \\ \hline 5 & 5 & 4 & 5 & \\ \hline 2 & 5 & 5 & 2 & \\ \hline \end{array}$$

# Matrices de adyacencia

Si hay un arco de u a w, entonces  $M_{u,w} = 1$

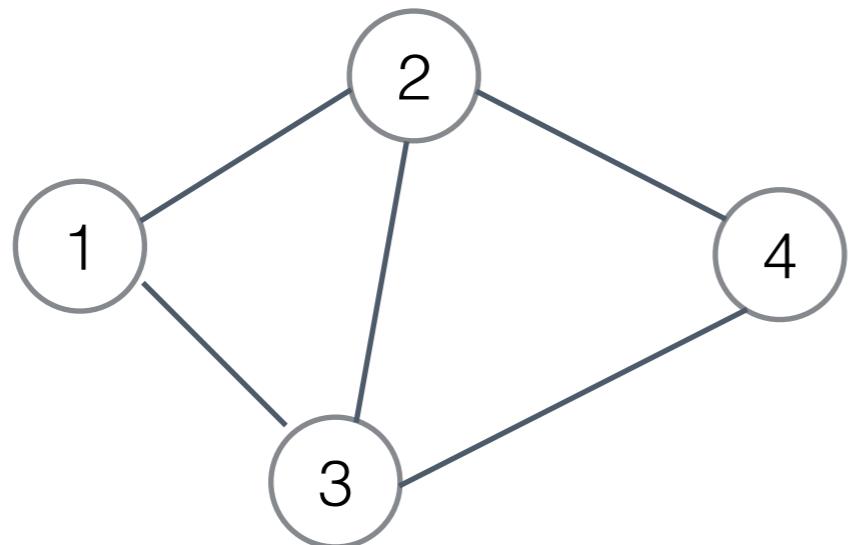


$$M^3 = \begin{array}{|c|c|c|c|} \hline & 2 & 5 & 5 & 2 \\ \hline 2 & 5 & 4 & 5 & 5 \\ \hline 5 & 5 & 4 & 5 & \\ \hline 2 & 5 & 5 & 2 & \\ \hline \end{array}$$

$$2+4+4+2 = 12$$

# Matrices de adyacencia

Si hay un arco de u a w, entonces  $M_{u,w} = 1$



$$M^3 = \begin{array}{|c|c|c|c|} \hline & 2 & 5 & 5 & 2 \\ \hline 2 & 5 & 4 & 5 & 5 \\ \hline 5 & 5 & 4 & 5 & \\ \hline 2 & 5 & 5 & 2 & \\ \hline \end{array}$$

$$2+4+4+2 = 12$$

$$12/6 = 2$$

# Más razones para contar triángulos

- Homofilia y transitividad (redes sociales)
- Descubrir estructuras temáticas subyacentes (web)
- Descubrir servidores con spam (web)
- Detección de comunidades
- Aplicaciones en CAD

# Outline

¿Por qué Graph Analytics?

Camino más corto

Pagerank

Contar triángulos

**Otros algoritmos**

# Clustering

La idea es bien simple: encontrar grupos de nodos en un grafo que se **parezcan** entre si

Dependiendo de que entendemos por parezcan, vamos a encontrar distintos tipos de clustering

# Clustering

Dos tipos principales:

- Encontrar conjuntos pequeños muy conectados (cliques)
- Grupos grandes de nodos de acuerdo a distancia

# Clustering

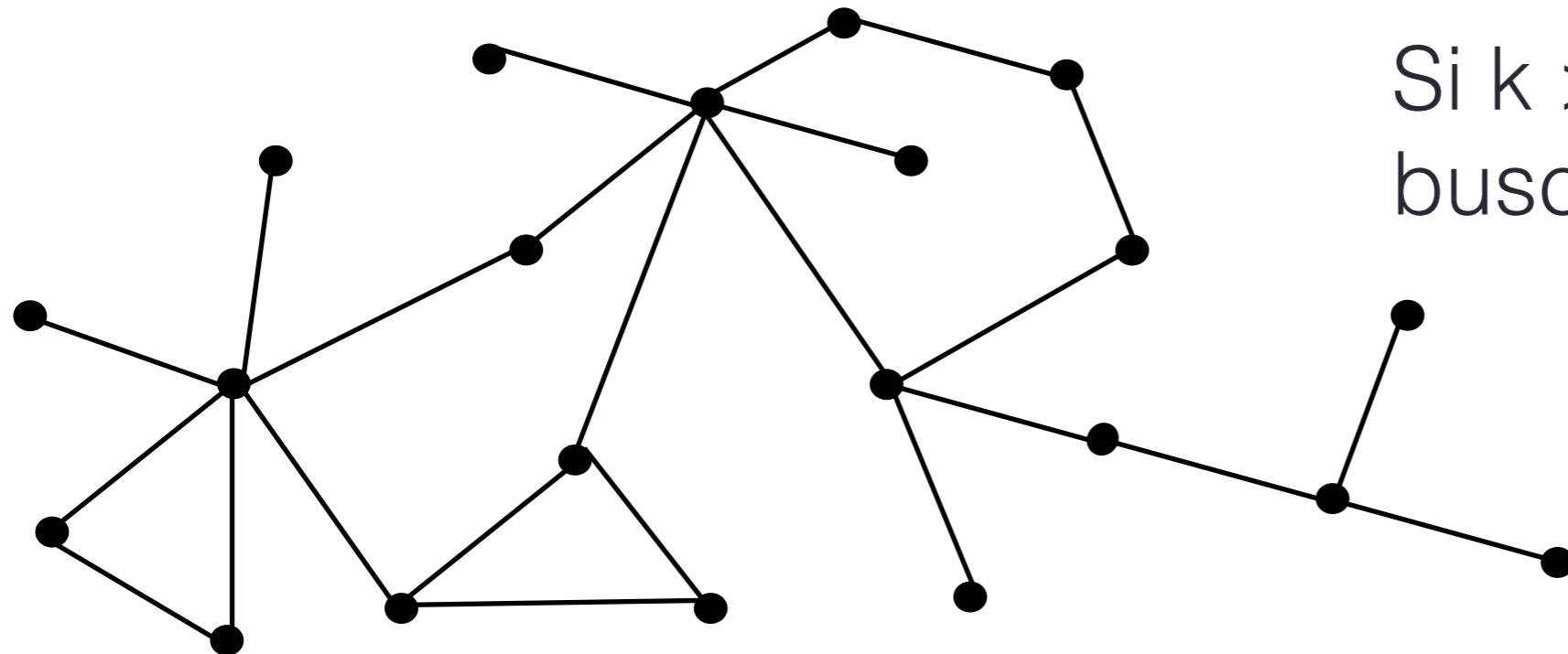
Encontrar conjuntos pequeños muy conectados (cliques)

- Cliques de tamaño k
- Todos los cliques maximales
- Casi cliques

# K-Cliques

El input es un grafo y un número  $k$ . Queremos encontrar todos los cliques de tamaño  $k$  en el grafo

Si  $k = 3$ ,  
buscamos los triángulos!



Si  $k > 3$ ?,  
buscamos los triángulos!

# K-Cliques

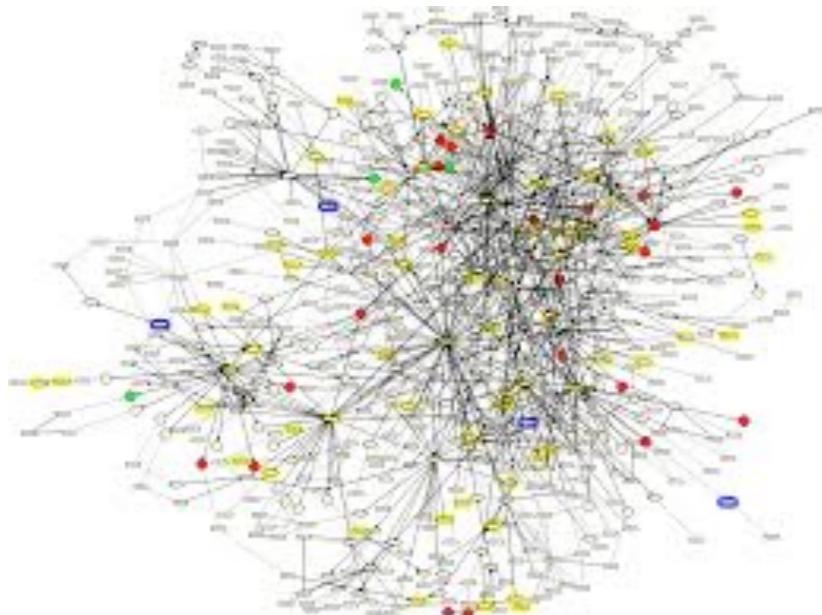
Algoritmo (fuerza bruta):

- Buscar todas las combinaciones de  $k$  nodos en un grafo
- Ver si esos nodos forman un clique

¿Podemos hacerlo mejor?

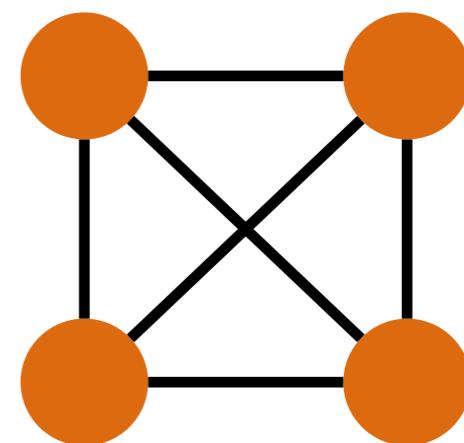
Es un problema abierto!

# K-Cliques y patrones



Listar los  $k$ -cliques de este grafo equivale a ver la forma de hacer match de el patron que corresponde al  $k$ -clique

Por ejemplo, para ver los 4-cliques hacemos match de este patrón:



# K-Cliques y patrones

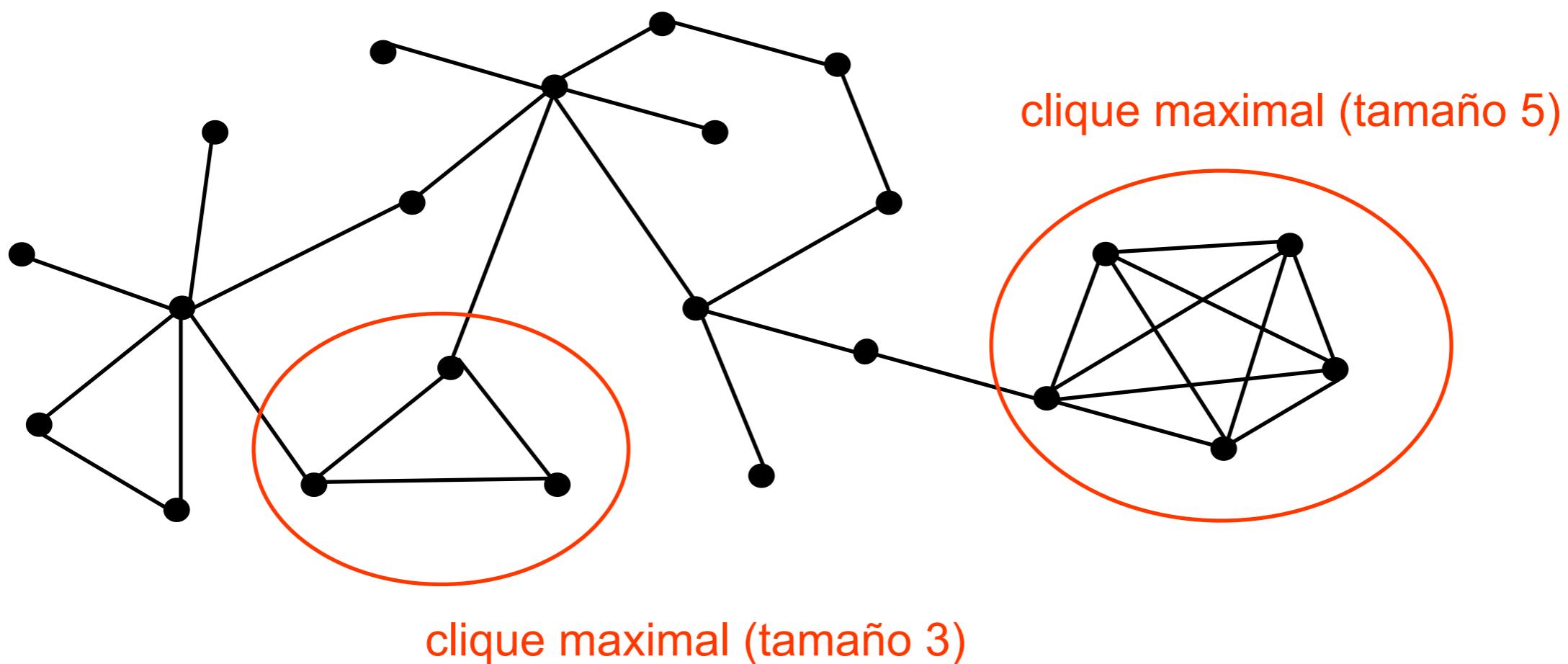
Por ahora, lo mejor es aprovechar las utilidades en una base de datos de grafos (vamos a ver cómo funcionan en un par de semanas)

# Cliques maximales

Un clique maximal en un grafo  $G$  es un conjunto de vértices tal que:

- Los vértices forman un clique
- No se puede agregar otro vértice y que siga existiendo un clique

# Cliques maximales



# Cliques maximales

Los cliques maximales representan comunidades

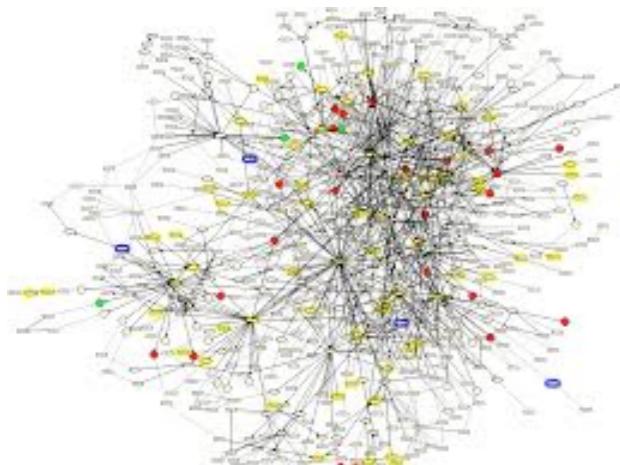
Pero ni siquiera es fácil diseñar un algoritmo de fuerza bruta... cuando paro? cuantas combinaciones busco?

Un algoritmo para hacer esto es el de Bron - Kerbosch

# Clustering

Grupos grandes de nodos de acuerdo a distancia

Aquí la idea es encontrar clusters, o conjuntos de nodos, tales que:



- Todos los nodos del conjunto estén **cerca**
- Los nodos estén **lejos** de otros conjuntos

# Clustering

Distancia entre nodos

Dados nodos  $n_1$  y  $n_2$ , la distancia entre  $n_1$  y  $n_2$  es el largo del camino más corto entre  $n_1$  y  $n_2$

Entonces tenemos:

- Set de nodos de un grafo
- Distancia entre cada par de nodos

Esto es exactamente igual que computar un cluster en cualquier set de datos!

# Clustering

Distancia entre nodos

Podemos usar cualquier algoritmo de clustering que conozcamos: K-vecinos, DBScan, Clustering jerárquico, etc...

# Clustering

## Problema

¿Cuánto cuesta computar?

- Set de nodos de un grafo
- Distancia entre cada par de nodos

Si tenemos 1.000.000 de nodos, aproximadamente haremos 1.000.000 de Dijkstra por nodo!

# Clustering

## Problema

¿Cuánto cuesta computar?

- Set de nodos de un grafo
- Distancia entre cada par de nodos

Si tenemos 1.000.000 de nodos, aproximadamente haremos 1.000.000 de Dijkstra por nodo!

Sin embargo, todavía podemos usar algunos algoritmos

# Clustering

## K-means

Si quiero K clusters de nodos:

- Elijo primero K nodos bien separados entre si
- Para cada uno de los nodos del grafo:
  - Veo la distancia al menor de los K
  - Lo asigno al cluster con menor distancia

# Clustering

## K-means

Si quiero K clusters de nodos:

- Elijo primero K nodos bien separados entre si
- Para cada uno de los nodos del grafo:
  - Veo la distancia al menor de los K
  - Lo asigno al cluster con menor distancia

Con esto, solo necesito la distancia de los nodos del grafo  
a los K nodos... solo necesito K Dijkstras!

# Clustering

¿Si no sé cuántos clusters quiero al principio?

- Puedo probar para distintos valores de K
- Puedo seleccionar algunos nodos del grafo, y solo computar las distancias con respecto a esos nodos (y después correr cualquier algoritmo de clustering)

Eso es una aproximación de la distancia real

# Clustering

Puedo probar también con otras nociones de distancia (por ejemplo, distancias que tomen en cuenta atributos de los nodos)

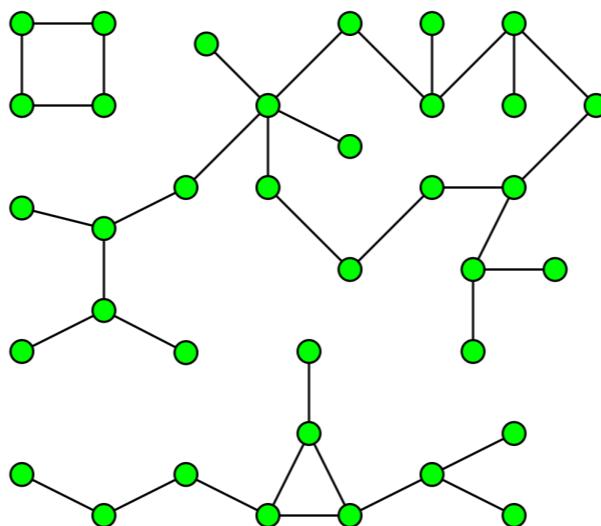
El fundamento siempre es el mismo:

- Usar K-means para evitar el computo de tanta distancia
- Usar algunos nodos intermedios para aproximarla

# Connected Components

Las *Connected Components* de un grafo son subgrafos del mismo

Se debe cumplir que para cada componente, todos los vértices deben estar conectados a los otros vértices de la componente por un camino



# Clustering Coefficient

El *clustering coefficient* es una medida del cuanto los nodos de un grafo tienden a formar clusters

En general, el coeficiente de *clustering* de un nodo es la proporción de números de aristas entre los vértices de su vecindad, dividido en el número de links que podrían existir entre sus vecinos (intuitivamente, requiere contar los triángulos en los que participa cada nodo)

# Clustering Coefficient

Sea  $v_i$  un vértice del grafo, su **local clustering coefficient**  $C_i$  se define como:

$$C_i = \frac{|\{ e_{jk} : v_j, v_k \in N_i, e_{jk} \in E \}|}{k_i(k_i - 1)}$$

Donde  $N_i$  es la vecindad del vértice,  $k_i$  es el número de vecinos del vértice y  $E$  son los vértices del grafo

# Caso: Facebook y Cambridge Analytica



## Personality Test

12 de septiembre de 2016 ·

...

I WILL HAVE 2 CHILDREN ❤️ ❤️

And YOU? Are you curious to know how many children will you have ?



Take the test and find out how many children you will have

subscribe



Take the test and find out how many children you will have



Me gusta



Comentar

0 de 0

[Take Another Test](#)

[Share on Facebook](#)

[Share on Twitter](#)

- With just 9 Facebook Likes, a computer can predict your personality as well as a colleague.
- With 65 Likes, as well as a friend.
- And with 125 Likes, your family.

<https://towardsdatascience.com/weapons-of-micro-destruction-how-our-likes-hijacked-democracy-c9ab6fcd3d02>

# Christopher Wylie



# Cambridge Analytica

# Cambridge Analytica

- Obtiene perfil psicológico de usuarios

# Cambridge Analytica

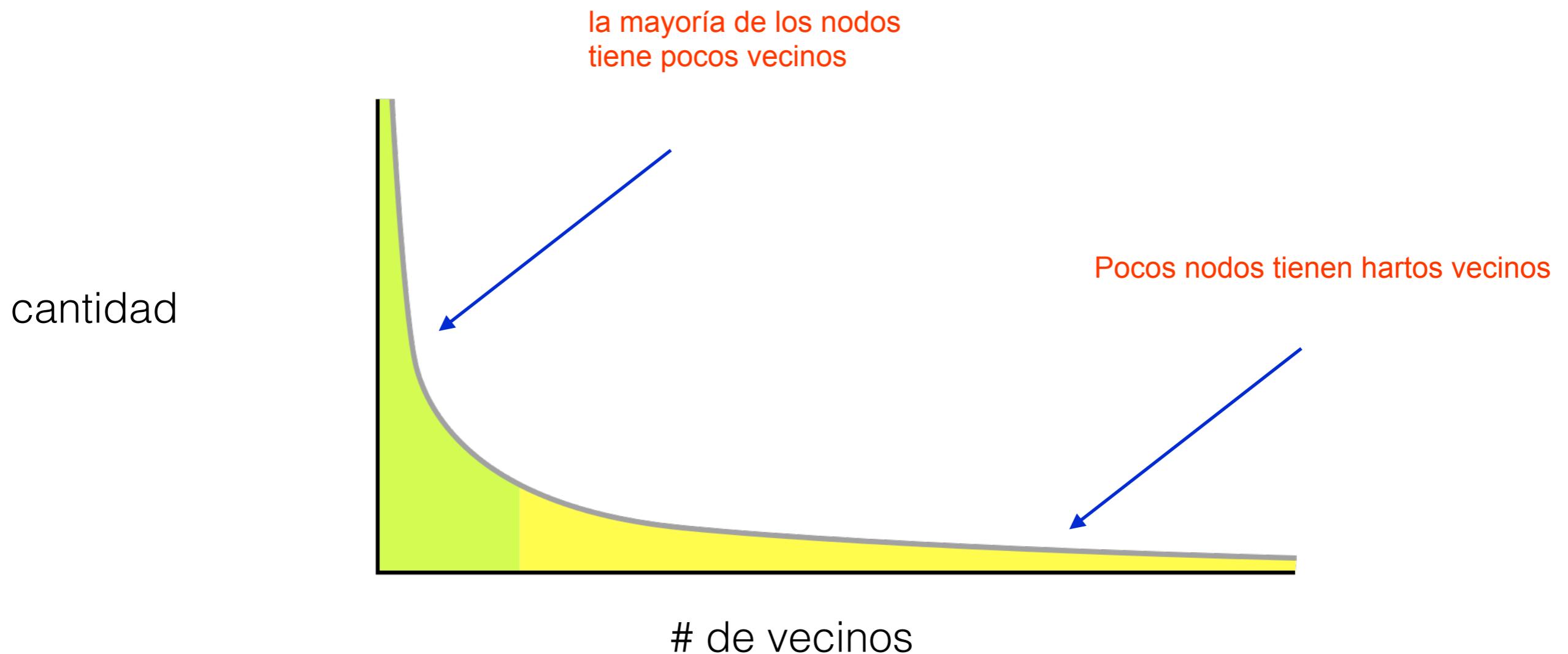
- Obtiene perfil psicológico de usuarios
- Obtiene información de FB de usuarios

# Cambridge Analytica

- Obtiene perfil psicológico de usuarios
- Obtiene información de FB de usuarios
- Obtiene información de sus amigos

# Cambridge Analytica

- Obtiene perfil psicológico de usuarios
- Obtiene información de FB de usuarios
- Obtiene información de sus amigos
- Genera noticias falsas según el perfil psicológico de los usuarios



# Estrategia

# Estrategia

- Hacemos *target* con noticias falsas a gente sensible a ellas en FB

# Estrategia

- Hacemos *target* con noticias falsas a gente sensible a ellas en FB
- ¿Por qué los medios no hablan de estas noticias que veo?

# Estrategia

- Hacemos *target* con noticias falsas a gente sensible a ellas en FB
- ¿Por qué los medios no hablan de estas noticias que veo?
- Los medios deben estar vendidos

# Estrategia

- Hacemos *target* con noticias falsas a gente sensible a ellas en FB
- ¿Por qué los medios no hablan de estas noticias que veo?
- Los medios deben estar vendidos
- No confío en los medios

# Estrategia

- Hacemos *target* con noticias falsas a gente sensible a ellas en FB
- ¿Por qué los medios no hablan de estas noticias que veo?
- Los medios deben estar vendidos
- No confío en los medios
- Confío en las noticias que veo en Facebook

# Estrategia

- Hacemos *target* con noticias falsas a gente sensible a ellas en FB
- ¿Por qué los medios no hablan de estas noticias que veo?
- Los medios deben estar vendidos
- No confío en los medios
- Confío en las noticias que veo en Facebook
- Comparto las noticias que veo





Cambridge  
Analytica



Cambridge  
Analytica





Cambridge  
Analytica



“I made Steve Bannon's psychological warfare tool”

– Christopher Wylie

<https://www.theguardian.com/news/2018/mar/17/data-war-whistleblower-christopher-wylie-facebook-nix-bannon-trump>

# Facebook Stock Continues To Plummet As FTC Reportedly Opens Investigation Into Company



<https://digg.com/2018/facebook-stock-price-drop-sell>

“UK data watchdog still seeking Cambridge Analytica warrant”

– <https://www.ft.com/content/b4fcfc80-2c51-11e8-9b4b-bc4b9f08f381>

“Facebook Is Under Investigation By The FTC”

– <https://digg.com/2018/facebook-stock-price-drop-sell>

# ¿Y qué tiene que ver con nosotros?

- Clustering
- Triángulos
- Pagerank

¿Cuándo fue la última vez que reenviaron una  
cadena por whatsapp?

¿O una noticia de un blog?

¿Cuándo fue la última vez le dieron permiso a una aplicación para ver sus datos en Facebook?

¿O su correo?

¿Le avisaron a sus  
contactos?

“Say goodbye to the information age: it’s all about reputation now”

<https://aeon.co/ideas/say-goodbye-to-the-information-age-its-all-about-reputation-now>

# Técnicas para Big Data

Clase 08 - Graph Analytics