Distancia y Caminos mínimos

Gregorio Hernández Peñalver

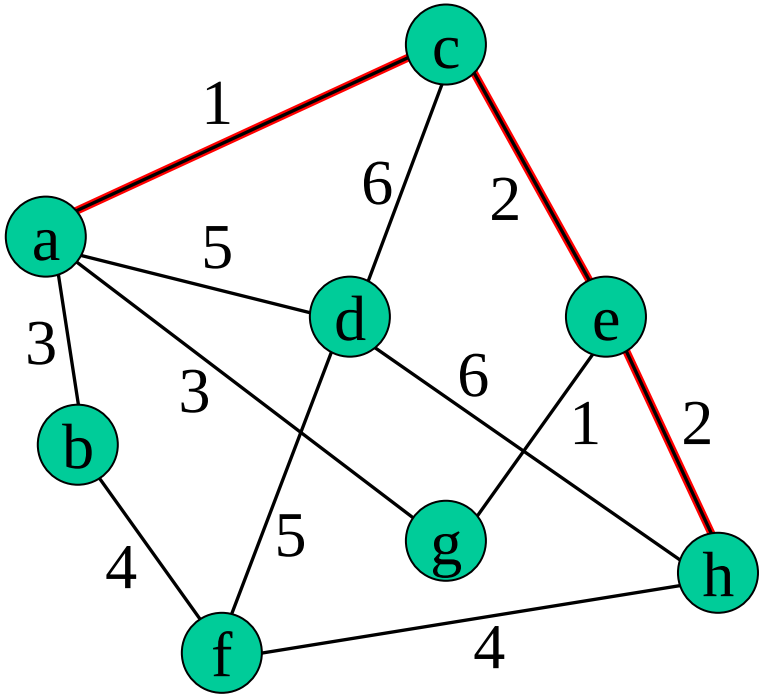
UPM

Matemática Discreta II

(MI)

Grafos ponderados

(G,w) $\quad w : A \longrightarrow R^+$



Peso de un subgrafo $H=(V_1, A_1)$

$$w(H) = \sum_{e \in A_1} w(e)$$

Distancia en un grafo ponderado

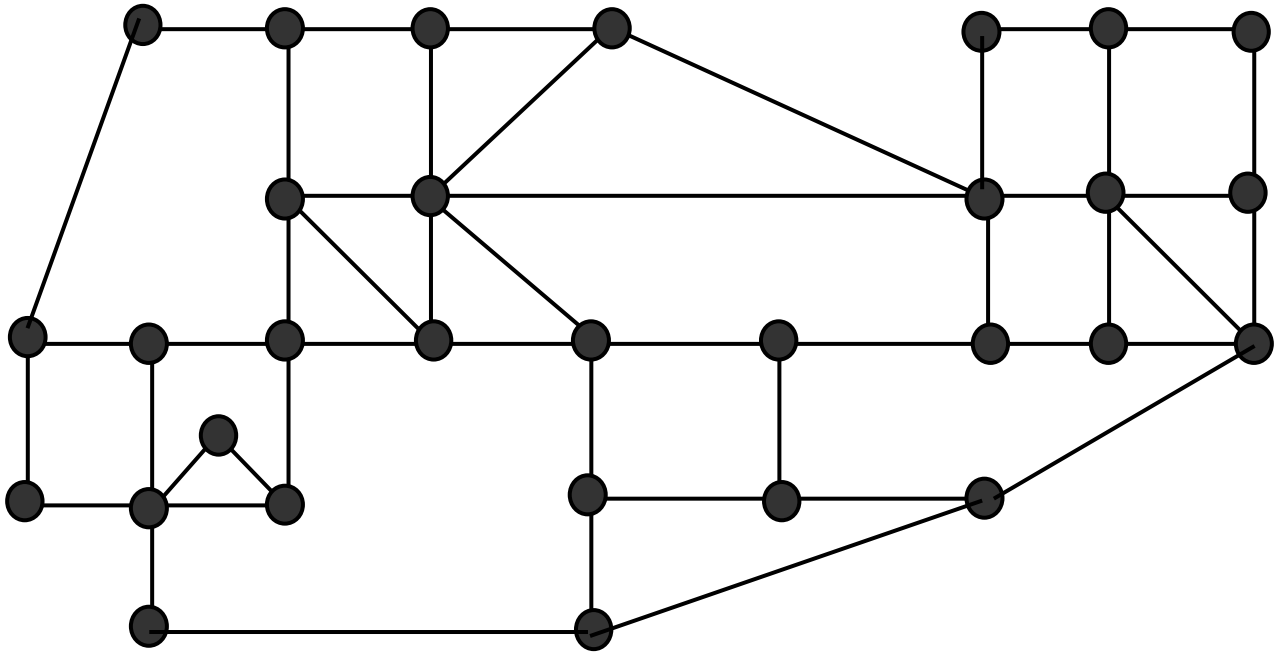
$d: V \times V \longrightarrow R^+$

$w(G) = 42$

$d(a,h) = 5$

$d(u,v)=\min\{w(C) \mid C \text{ camino de u a v}\}$

UBICACIÓN DE SERVICIOS

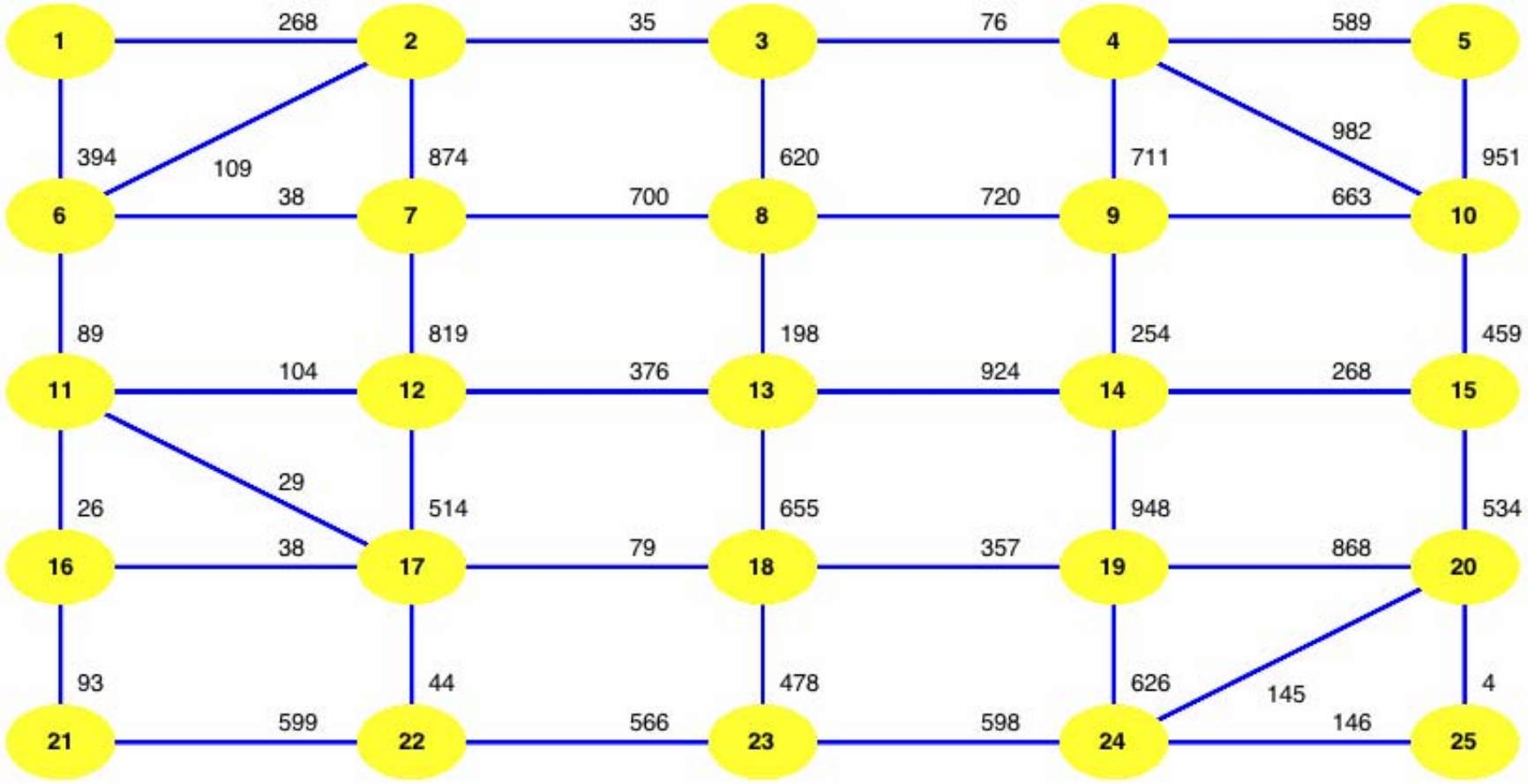


Parque de bomberos

Colegio

Centro comercial

UBICACIÓN DE SERVICIOS



UBICACIÓN DE SERVICIOS

Servicios de emergencia (Hospital, bomberos, policía, …)

Objetivo: **MINMAX**
Minimizar el coste máximo (tiempo, distancia, …)

Elegir un vértice v que minimice $\max\{dist(v,x) / x \in V(G)\}$

Servicios sociales (Colegio, Centro Comercial, salas de cine …)

Objetivo: **MINSUM**
Minimizar el coste promedio (tiempo, distancia, …)

Elegir un vértice v que minimice $\sum_{x \in V(G)} dist(v, x)$

UBICACIÓN DE SERVICIOS

La **excentricidad** de un vértice v es

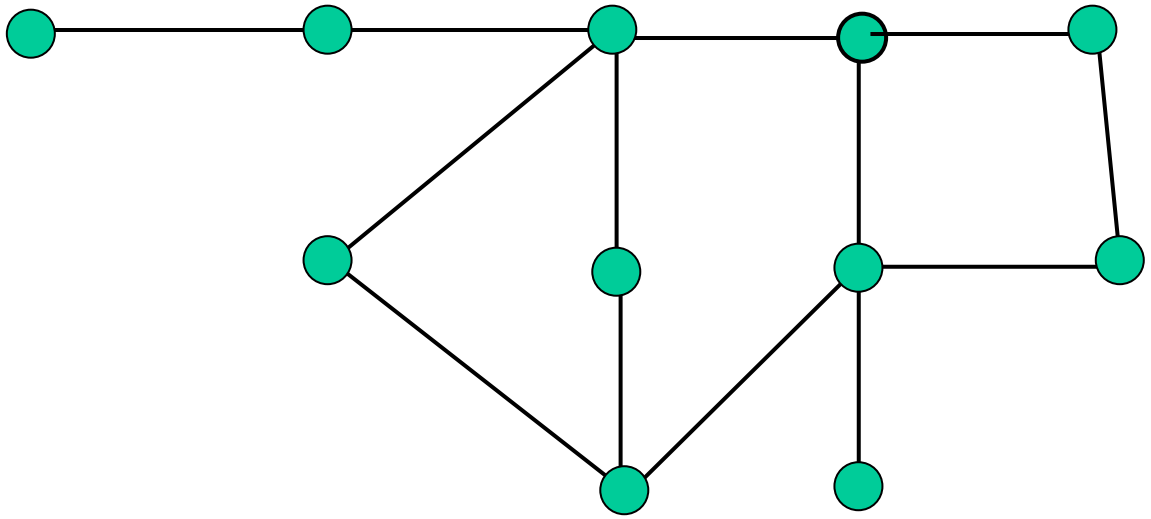
e(v) = máx{dist(v,z)/ z∈ V(G)}

El **radio** de un grafo es

rad(G) = mín{e(v)/ v∈ V(G)}

El **diámetro** de un grafo es

diam(G) = máx{e(v)/ v∈ V(G)}



e(u)=3

e(v)=4

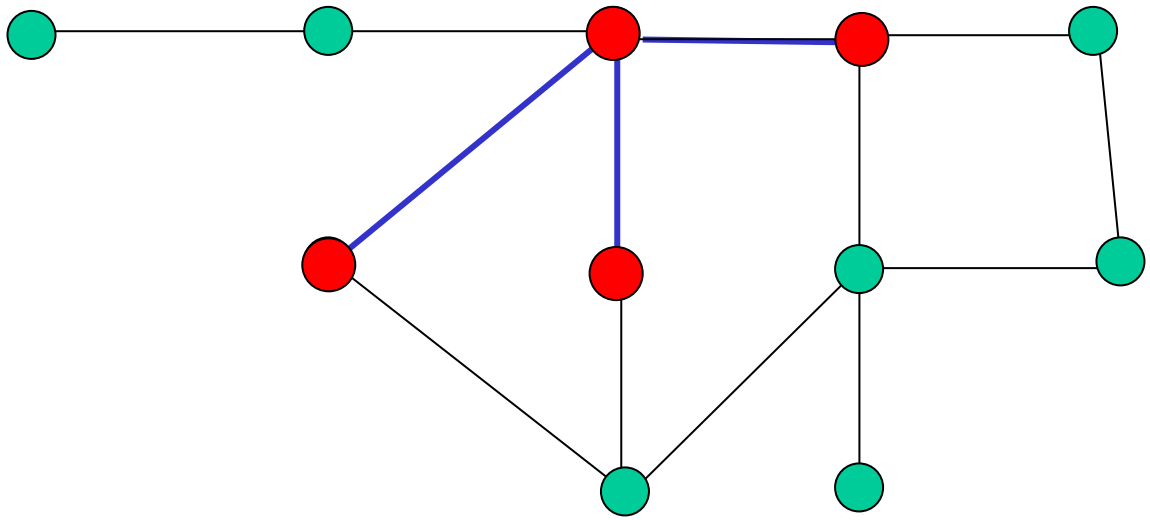
e(w)=5

rad(G)=3

diam(G)=5

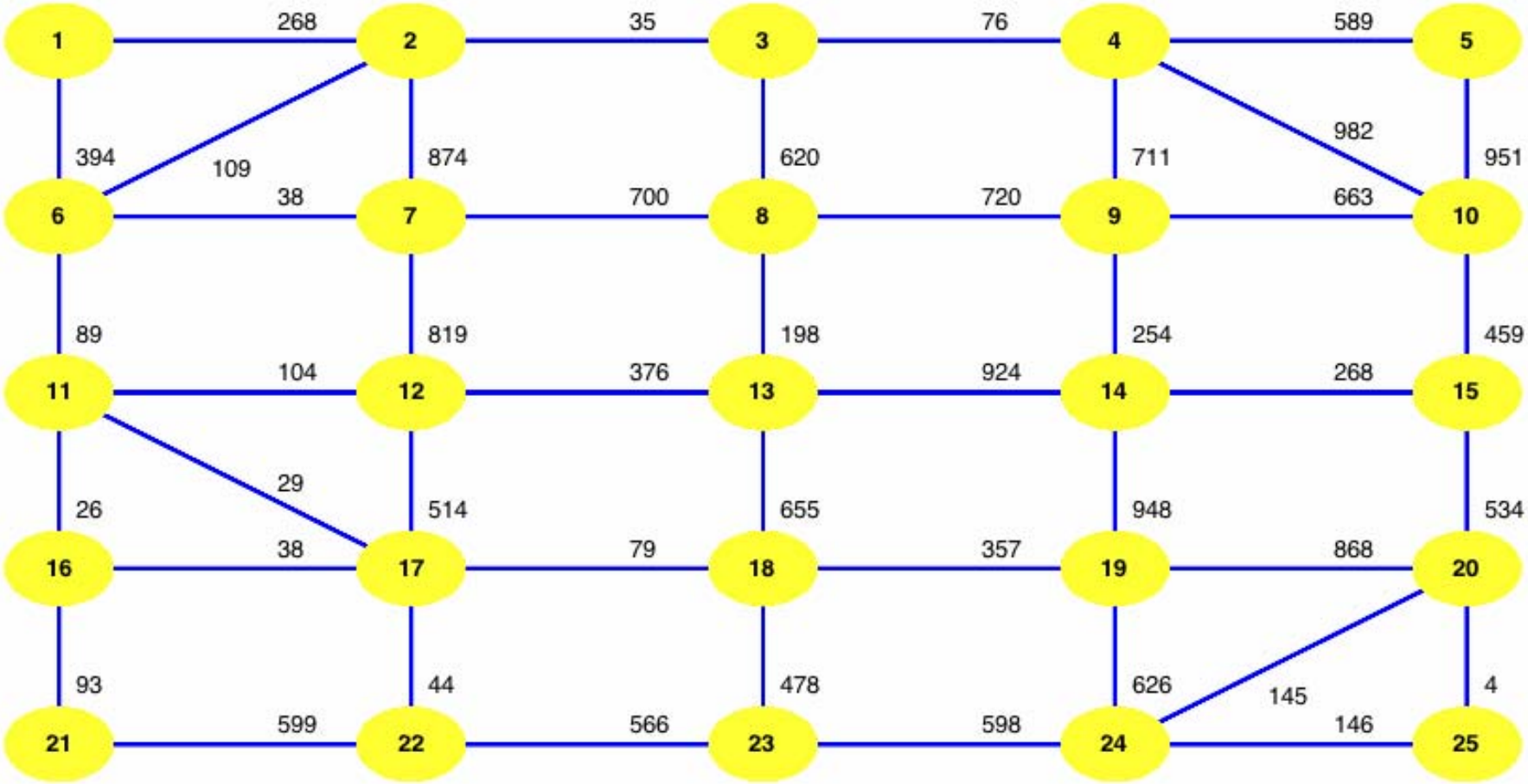
UBICACIÓN DE SERVICIOS

El **centro** de un grafo G es el subgrafo inducido por el conjunto de vértices de excentricidad mínima.



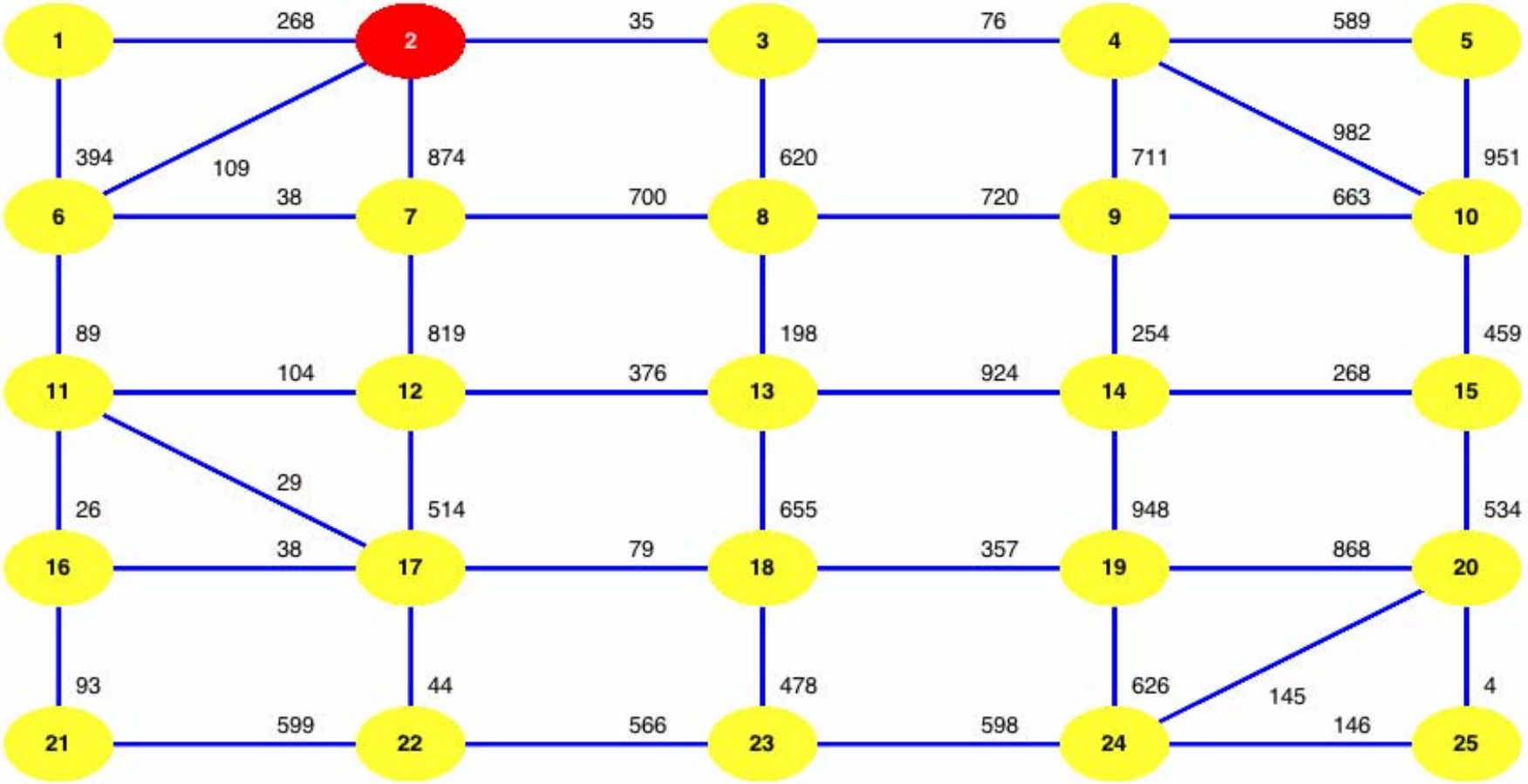
El parque de bomberos se debe instalar en un vértice del centro

UBICACIÓN DE SERVICIOS



[seq(Exc(Red,i), i=1..25)];
[1703, 1435, 1470, 1546, 1989, 1453, 1491, 1780, 1606, 1736, 1542, 1568, 1717, 1554, 1661, 1568, 1571, 1573, 1675, 1944, 1661, 1615, 1736, 1989, 1948]

UBICACIÓN DE SERVICIOS



El parque de bomberos se debe instalar en el centro del grafo, C(Red) = {2}

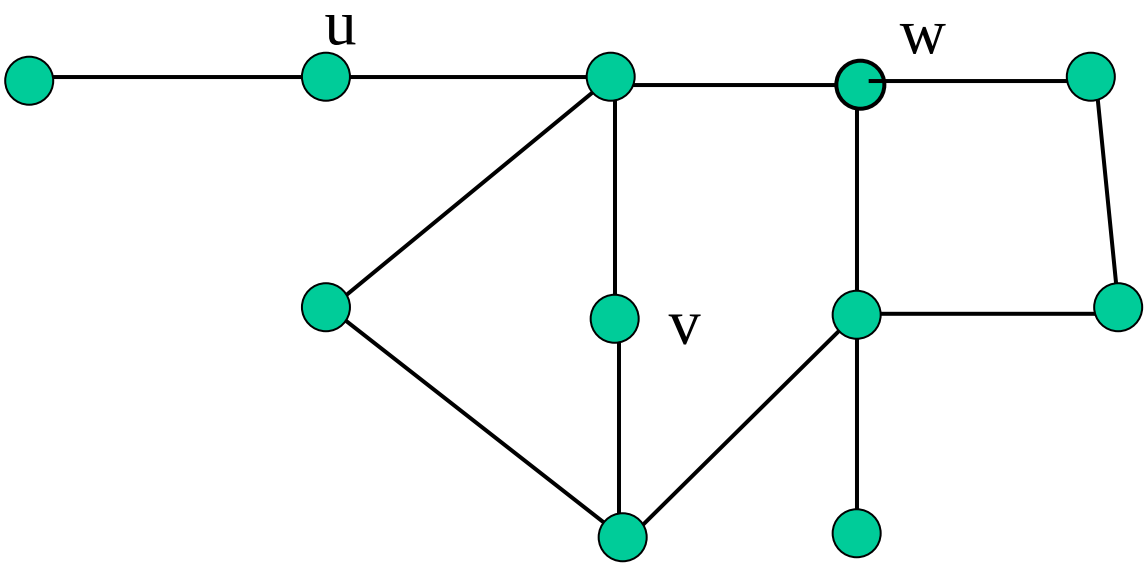
UBICACIÓN DE SERVICIOS

La **distancia total** de un vértice v es

$$dt(v) = \sum_{z \in V(G)} dist(v, z)$$

La **mediana** de un grafo G es el subgrafo inducido por el conjunto de vértices de distancia total mínima

El colegio se instala en un vértice de la mediana

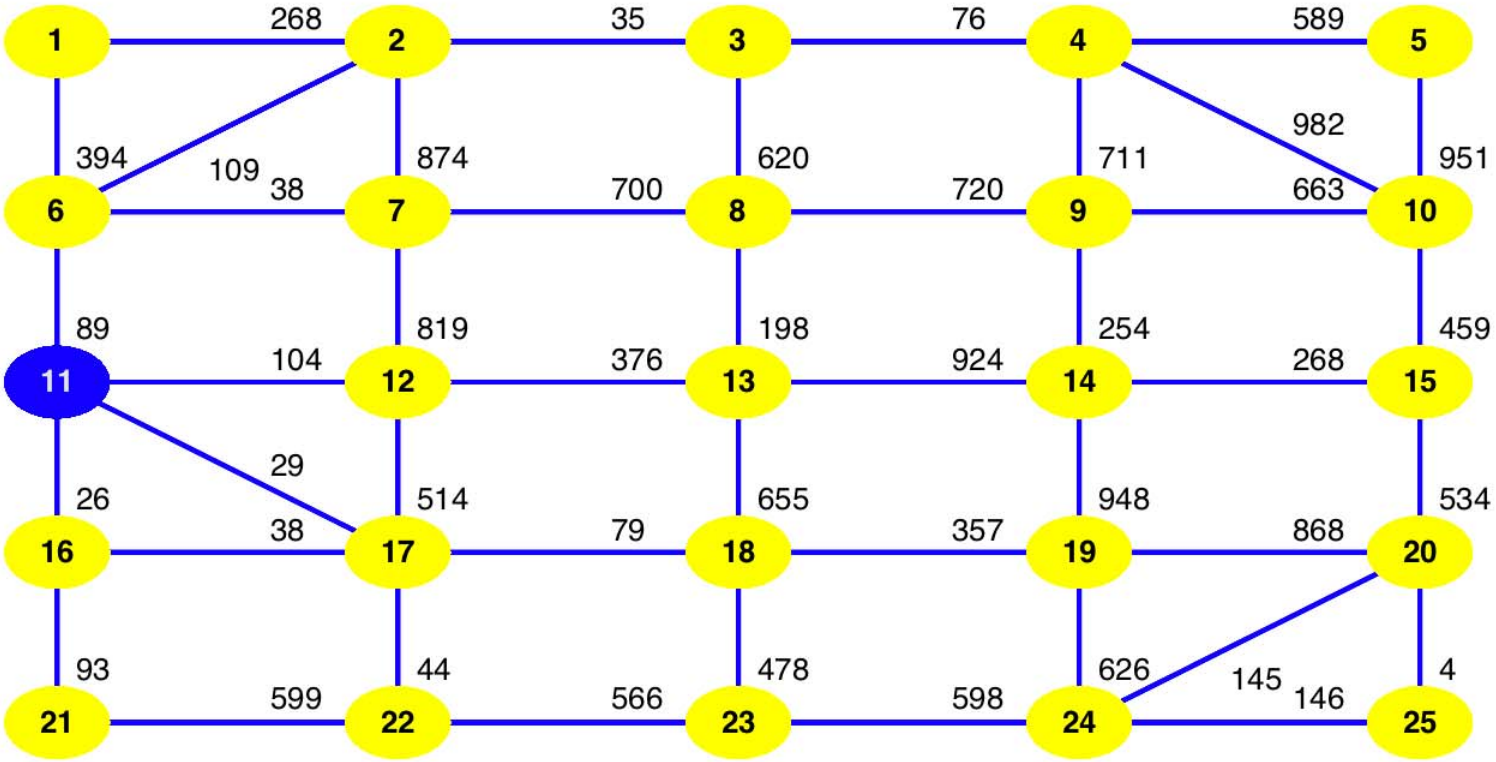


dt(u)=25

dt(v)=21

dt(w)=18

UBICACIÓN DE SERVICIOS



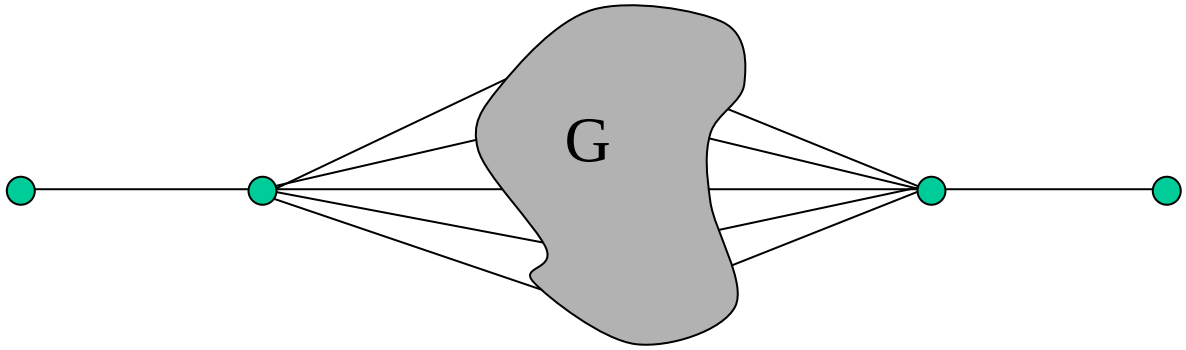
> [seq(DistTotal(Red,i), i=1..25)];
[20652, 14488, 14803, 15791, 27929, 13917, 14715, 22863, 23411, 28735, 13679, 15499, 21119, 25805, 28877, 14089, 13906, 14779, 19056, 28067, 16228, 14883, 21996, 26507, 28108]

El colegio se debe instalar en la mediana del grafo, Mediana(Red) = {11}

¿Cómo se halla el centro de un grafo? ¿Y la mediana?
¿Cómo se calculan las distancias en un grafo?
¿Cómo se halla el camino mínimo entre dos vértices de un grafo?

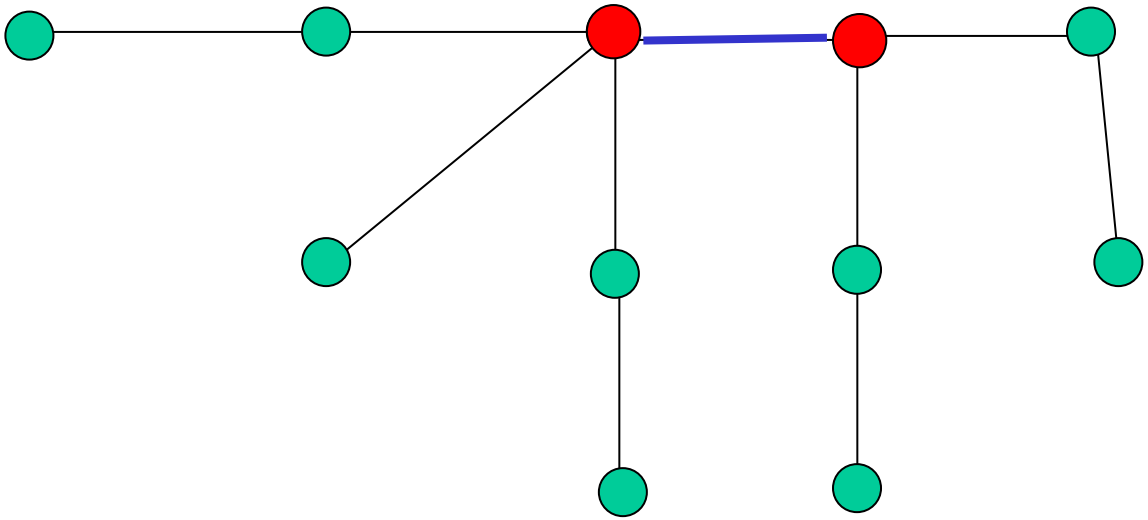
Propiedades

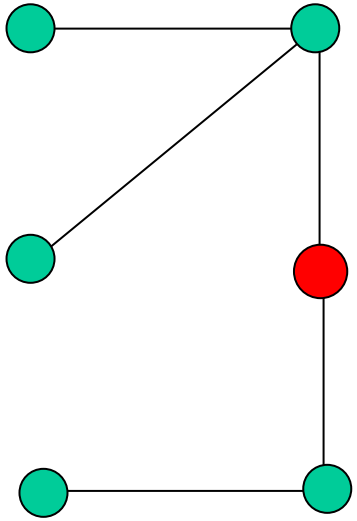
1. Si G es un grafo conexo entonces
   $rad(G) \leq diam(G) \leq 2rad(G)$
2. Todo grafo G es el centro de un grafo conexo



ÁRBOLES (sin pesos)

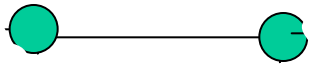
El centro de un árbol T está formado por uno o dos vértices de T





Algoritmo para obtener el centro de un árbol

Entrada: Un árbol T
Salida: El centro del árbol C(T)
Paso 1. Hacer H=T
Paso 2. Si H= K_1 ó K_2, entonces C(T)=H.
Paso 3. En caso contrario borrar todas las hojas de H para obtener un árbol H'. Hacer H'=H y volver al paso 2



Teorema

El algoritmo anterior obtiene C(T)

Si T es árbol, V(T)>2, T'= T –{hojas}, entonces C(T)=C(T')

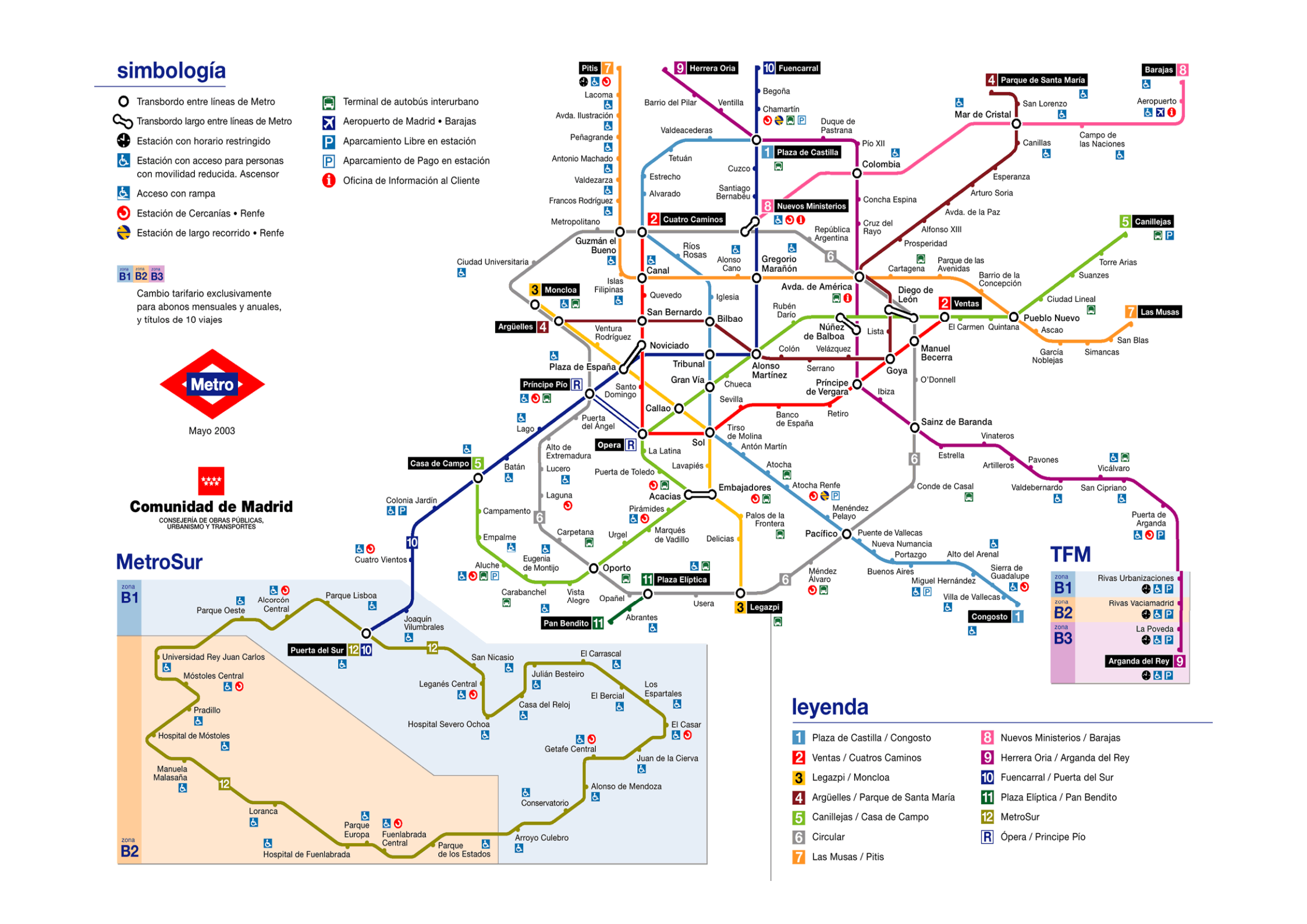
1. Si u es hoja de T, u' es su adyacente entonces e(u') < e(u)

   Por tanto, u no está en C(T)

2. Si z no es hoja de T, entonces e(z,T') = e(z,T) – 1

   Por tanto, rad(T') = rad(T) – 1 y C(T)=C(T')

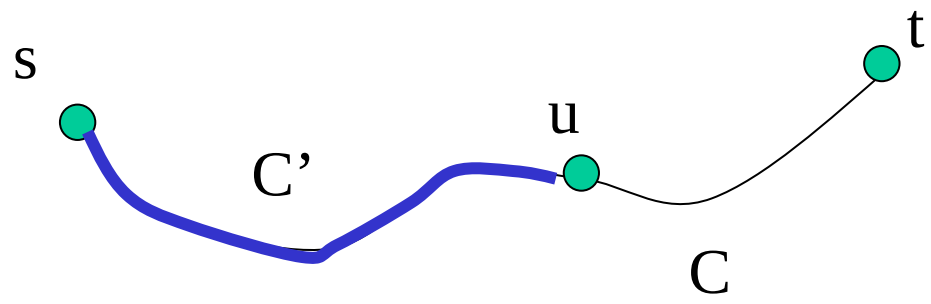
¿Y si las aristas de T tienen peso?


simbología
Transbordo entre líneas de Metro
Transbordo largo entre líneas de Metro
Estación con horario restringido
Estación con acceso para personas con movilidad reducida. Ascensor
Acceso con rampa
Estación de Cercanías • Renfe
Estación de largo recorrido • Renfe
Terminal de autobús interurbano
Aeropuerto de Madrid • Barajas
Aparcamiento Libre en estación
Aparcamiento de Pago en estación
Oficina de Información al Cliente
zona B1 zona B2 zona B3
Cambio tarifario exclusivamente para abonos mensuales y anuales, y títulos de 10 viajes
Metro
Mayo 2003
Comunidad de Madrid
CONSEJERÍA DE OBRAS PÚBLICAS, URBANISMO Y TRANSPORTES
MetroSur
zona B1
zona B2
Parque Oeste
Alcorcón Central
Parque Lisboa
Joaquín Vilumbrales
Puerta del Sur
Universidad Rey Juan Carlos
Móstoles Central
Pradillo
Hospital de Móstoles
Manuela Malasaña
Loranca
Hospital de Fuenlabrada
Parque Europa
Fuenlabrada Central
Parque de los Estados
Arroyo Culebro
Conservatorio
Alonso de Mendoza
Juan de la Cierva
Getafe Central
El Casar
Hospital Severo Ochoa
Leganés Central
San Nicasio
Julián Besteiro
El Carrascal
Casa del Reloj
El Bercial
Los Espartales
Pitis 7
Lacoma
Avda. Ilustración
Peñagrande
Antonio Machado
Valdezarza
Francos Rodríguez
Metropolitano
Ciudad Universitaria
Guzmán el Bueno
Islas Filipinas
Moncloa 3
Argüelles 4
Ventura Rodríguez
Plaza de España
Príncipe Pío R
Lago
Casa de Campo 5
Batán
Colonia Jardín
Cuatro Vientos
Campamento
Empalme
Aluche
Eugenia de Montijo
Carabanchel
Vista Alegre
Pan Bendito 11
Abrantes
Opañel
Oporto
Carpetana
Urgel
Laguna
Alto de Extremadura
Lucero
Puerta del Ángel
Noviciado
San Bernardo
Quevedo
Canal
Cuatro Caminos 2
Metropolitano
Barrio del Pilar
Ventilla
Valdeacederas
Tetuán
Estrecho
Alvarado
Ríos Rosas
Alonso Cano
Iglesia
Bilbao
Tribunal
Gran Vía
Callao
Santo Domingo
Opera R
Sol
La Latina
Lavapiés
Puerta de Toledo
Acacias
Pirámides
Marqués de Vadillo
Delicias
Embajadores
Palos de la Frontera
Atocha Renfe
Atocha
Tirso de Molina
Antón Martín
Sevilla
Chueca
Banco de España
Retiro
Príncipe de Vergara
Ibiza
Alonso Martínez
Colón
Serrano
Velázquez
Goya
Núñez de Balboa
Lista
Rubén Darío
Avda. de América
Gregorio Marañón
República Argentina
Nuevos Ministerios 8
Cruz del Rayo
Concha Espina
Santiago Bernabéu
Cuzco
Plaza de Castilla 1
Begoña
Chamartín
Duque de Pastrana
Pío XII
Colombia
Fuencarral 10
Herrera Oria 9
Plaza Elíptica 11
Usera
Legazpi 3
Pacífico
Méndez Álvaro
Palos de la Frontera
Menéndez Pelayo
Puente de Vallecas
Nueva Numancia
Portazgo
Buenos Aires
Miguel Hernández
Villa de Vallecas
Congosto 1
Alto del Arenal
Sierra de Guadalupe
Mar de Cristal
Parque de Santa María 4
San Lorenzo
Canillas
Esperanza
Arturo Soria
Avda. de la Paz
Alfonso XIII
Prosperidad
Barajas 8
Aeropuerto
Campo de las Naciones
Cartagena
Parque de las Avenidas
Barrio de la Concepción
Diego de León
Ventas 2
El Carmen
Quintana
Pueblo Nuevo
Ciudad Lineal
Torre Arias
Suanzes
Canillejas 5
Las Musas 7
Ascao
García Noblejas
San Blas
Simancas
Manuel Becerra
O'Donnell
Sainz de Baranda
Estrella
Conde de Casal
Vinateros
Artilleros
Pavones
Valdebernardo
Vicálvaro
San Cipriano
Puerta de Arganda
TFM
zona B1
Rivas Urbanizaciones
zona B2
Rivas Vaciamadrid
zona B3
La Poveda
Arganda del Rey 9
leyenda
1 Plaza de Castilla / Congosto
2 Ventas / Cuatros Caminos
3 Legazpi / Moncloa
4 Argüelles / Parque de Santa María
5 Canillejas / Casa de Campo
6 Circular
7 Las Musas / Pitis
8 Nuevos Ministerios / Barajas
9 Herrera Oria / Arganda del Rey
10 Fuencarral / Puerta del Sur
11 Plaza Elíptica / Pan Bendito
12 MetroSur
R Ópera / Principe Pío

Algoritmos de caminos mínimos

Queremos calcular d(s,t) y el camino correspondiente

Si u se encuentra en un camino C de longitud mínima entre s y t

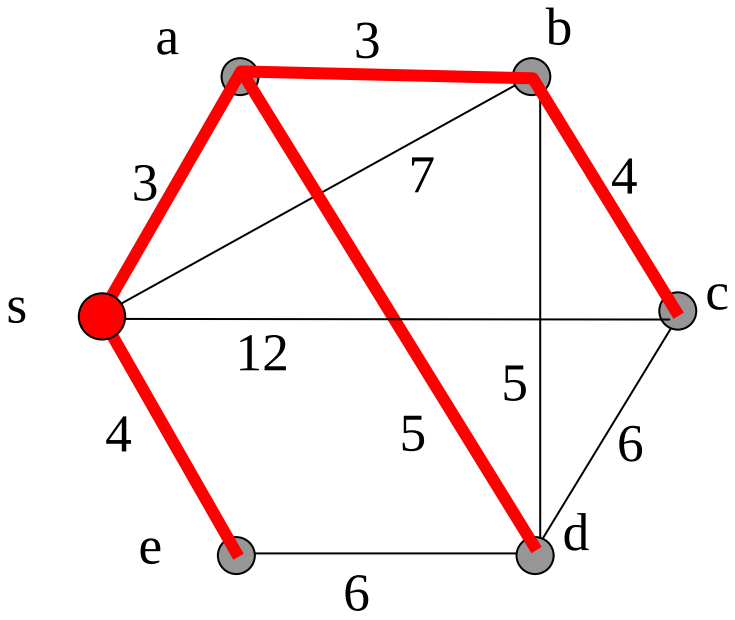


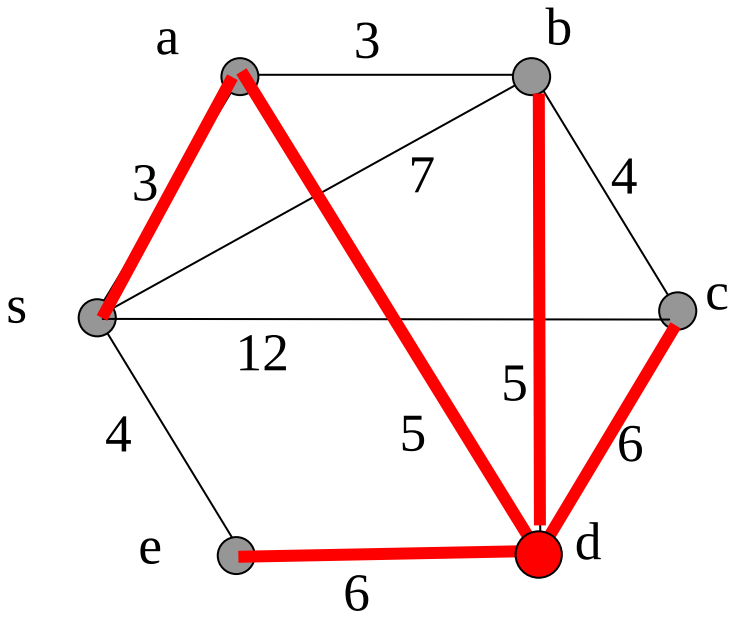
Un camino de longitud mínima de s a u es C'

Actualización de etiquetas (edge relaxation)

Árbol de caminos mínimos desde s

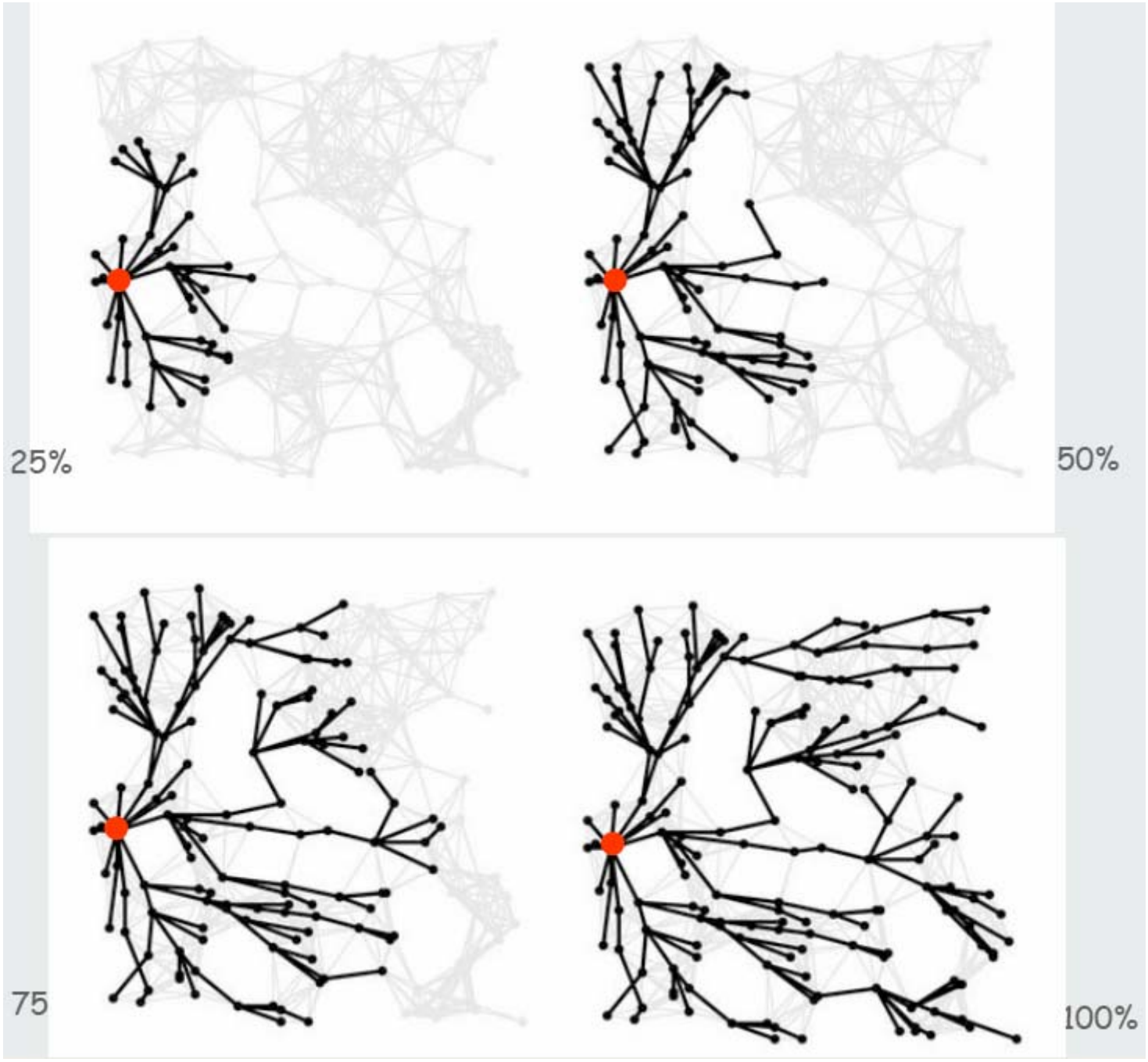
ÁRBOL DE CAMINOS MÍNIMOS desde s



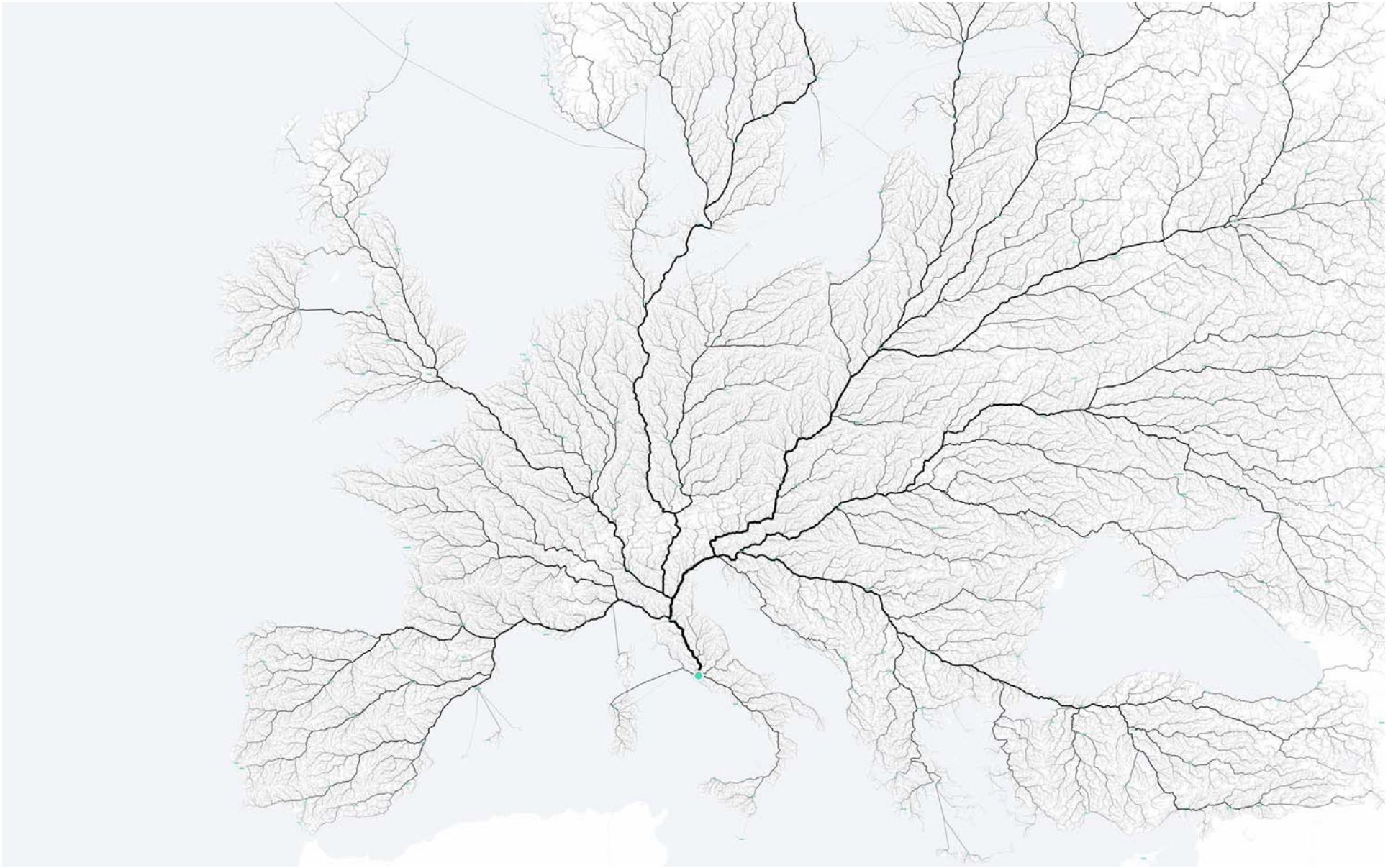


Árbol de caminos mínimos desde d

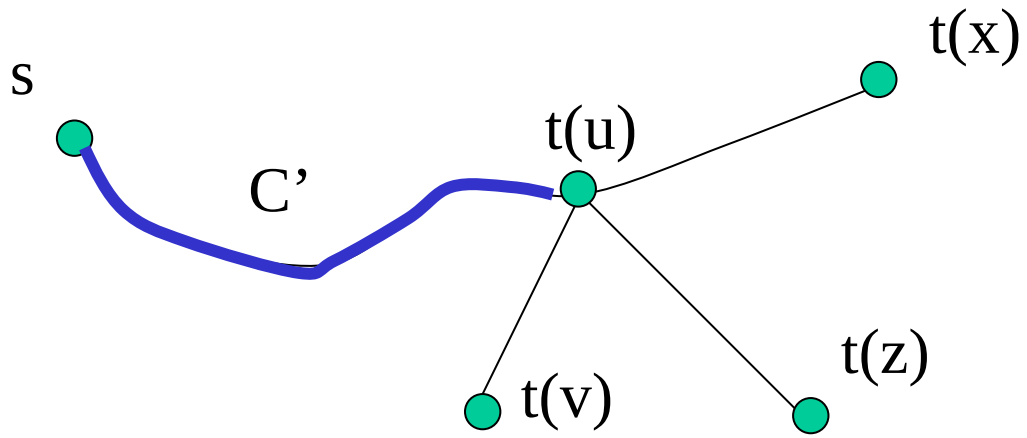
Este árbol NO es el MinimumSpanningTree


25%
50%
75
100%

Todos los caminos conducen a Roma (desde 486713 lugares)



Algoritmos de caminos mínimos



t(u) longitud del camino mínimo de s a u

Nuevas etiquetas en los vecinos de u

$$t(x) := \min\{t(x), t(u) + w(ux)\}$$

Algoritmo de Dijkstra (1959)

Entrada: Un grafo (o digrafo) ponderado, un vértice $s \in V$. El peso de la arista uv se indica por w(uv), poniendo $w(uv)=\infty$ si uv no es arista. (Las aristas tienen pesos no negativos)

Clave: Mantener el conjunto T de vértices para el que se conoce el camino más corto y ampliar T hasta que T=V. Para ello etiquetamos cada vértice z con t(z) que es la longitud del camino más corto ya encontrado.

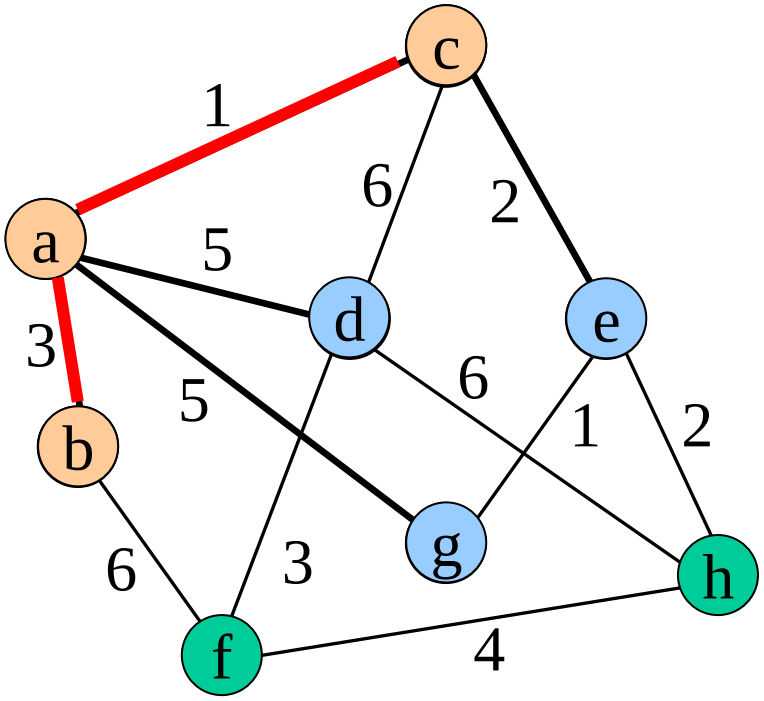
Inicialización: Sea $T=\{s\}$, $t(s)=d(s,s)=0$, $t(z)=w(sz)$ para $z \neq s$.

Iteración: Elegir el vértice $v \notin T$ con etiqueta mínima. Añadir v a T

Analizar cada arista vz con $z \notin T$ y actualizar la etiqueta de z a $\min\{t(z), t(v)+w(vz)\}$

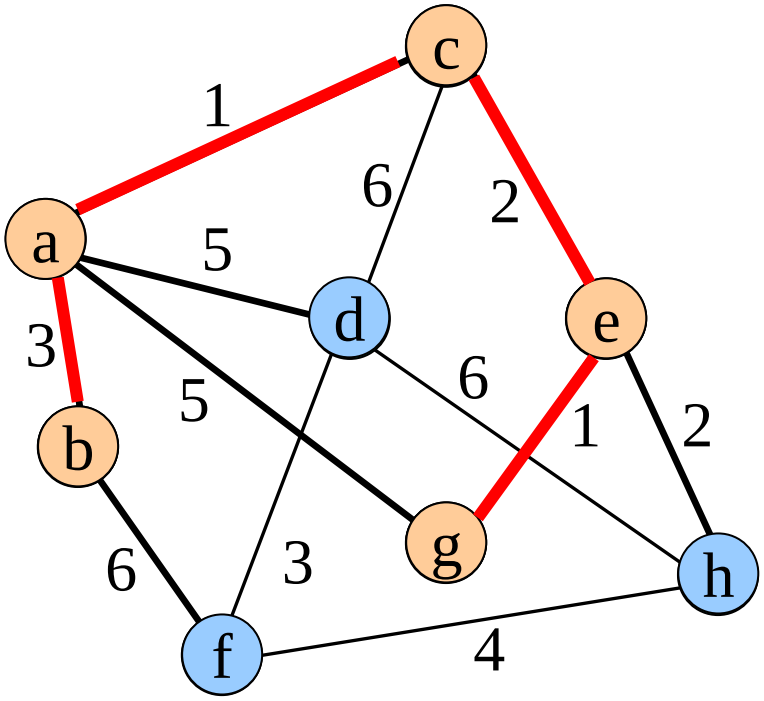
La iteración continua hasta que T=V(G) o hasta que $t(z)=\infty$ para cada vértice $z \notin T$

Algoritmo de Dijkstra



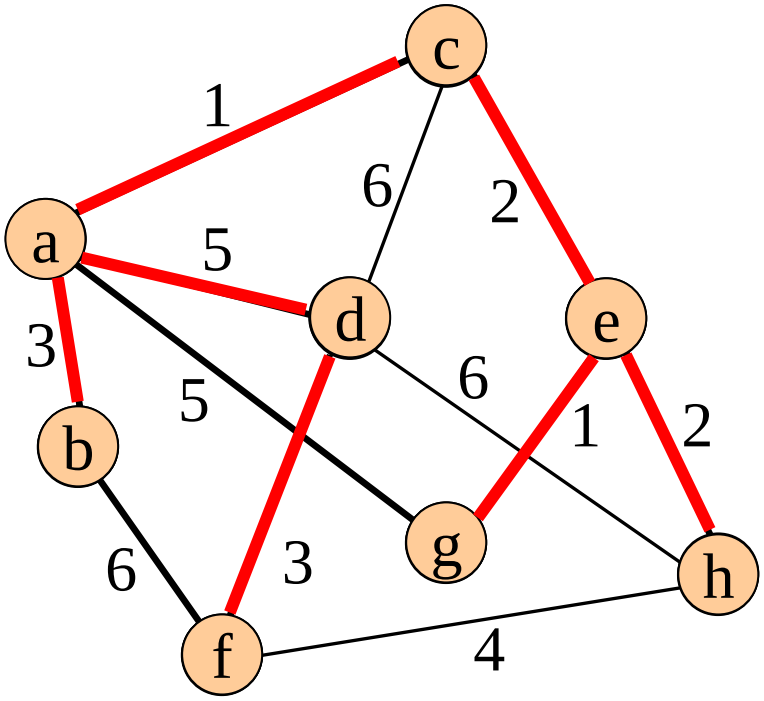
a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**T**	arista
0	**3**	**1**	**5**	∞	∞	**5**	∞	**c**	ac
	3		**5**	**3**	∞	**5**	∞	**b**	ab

Algoritmo de Dijkstra



a	b	c	d	e	f	g	h	T	arista
0	**3**	**1**	**5**	∞	∞	**5**	∞	**c**	ac
	3		**5**	**3**	∞	**5**	∞	**b**	ab
			5	**3**	**9**	**5**	∞	**e**	ce
			5		**9**	**4**	**5**	**g**	eg

Algoritmo de Dijkstra



a	b	c	d	e	f	g	h	T	arista
0	**3**	**1**	**5**	∞	∞	**5**	∞	**c**	ac
	3		**5**	**3**	∞	**5**	∞	**b**	ab
			5	**3**	**9**	**5**	∞	**e**	ce
			5		**9**	**4**	**5**	**g**	eg
			5		**9**		**5**	**d**	ad
					8		**5**	**h**	eh
					8			**f**	df

Análisis de la complejidad

El nº de iteraciones es n, en cada una se elige una etiqueta mínima, entre n – 1 , entre n – 2, …, coste total $O(n^2)$

Actualizaciones de etiqueta, cada arista da lugar a una, coste total O(q)

$$O(n^2+q)$$

Se puede implementar en $O(q\log n)$

Teorema (Validez del algoritmo)

El algoritmo de Dijkstra calcula d(s,z) para cada vértice z

Debemos probar que la etiqueta definitiva t(z) es d(s,z).
Sean $x_1, x_2, ..., x_n$ los vértices de G ordenados por su incorporación al conjunto T. Así x_1=s. Vamos a demostrar el resultado por inducción sobre i.

Primer paso) El resultado es cierto para i=1, pues x_1=s y sabemos que d(s,s)=0=t(s)

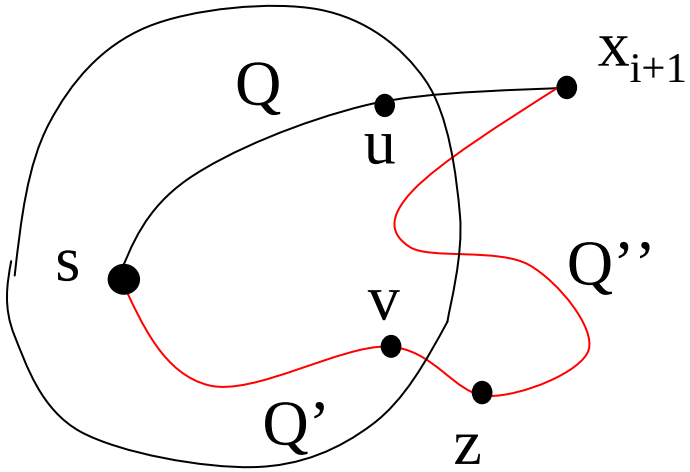
Paso de i a i+1) La hipótesis de inducción es que $t(x_1)=d(s,x_1)$, ..., $t(x_i)=d(s,x_i)$. Debemos probar que $t(x_{i+1})=d(s,x_{i+1})$

Teorema (Validez del algoritmo)

Llamemos S={x_1, x_2, ..., x_i}, La etiqueta t(x_{i+1}) es la longitud de un camino Q s,..., u, x_{i+1}, donde u es el último vértice en S. Si hay otro camino Q' de s a x_{i+1} debemos probar que

$$long(Q) \leq long(Q').$$

Sea z el primer vértice de Q' fuera de S, vz la primera arista y Q" el resto del camino de z a x_{i+1}



Teorema (Validez del algoritmo)

long(Q')=d(s,v)+w(vz) +long(Q") ≥ t(z)+long(Q")

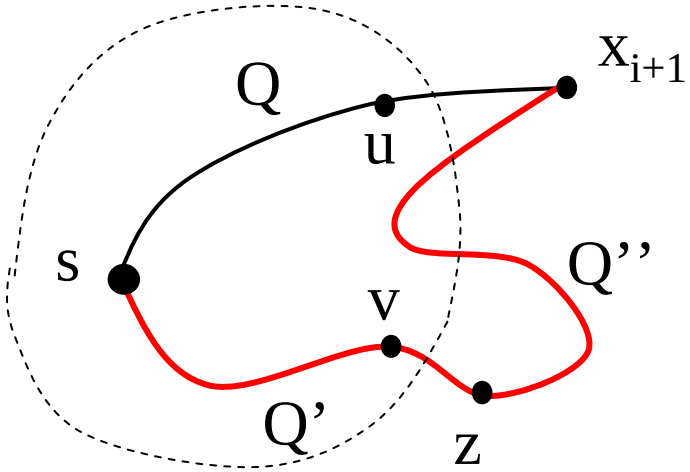
y como x_{i+1} se elige como vértice de menor etiqueta será

$$t(z) \geq t(x_{i+1})$$

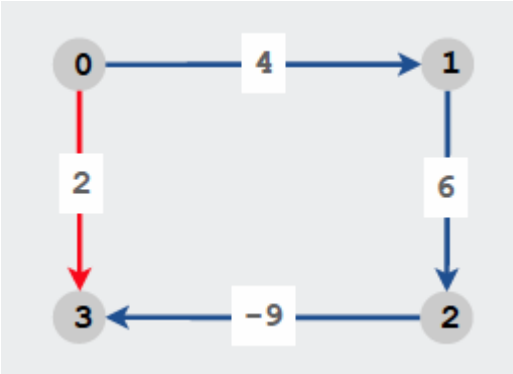
y así long(Q') ≥ $t(x_{i+1})$ + long(Q") ≥ $t(x_{i+1})$ = long(Q)

por ser todas las aristas de peso no negativo.

Por tanto, $t(x_{i+1})=long(Q)=d(s,x_{i+1})$

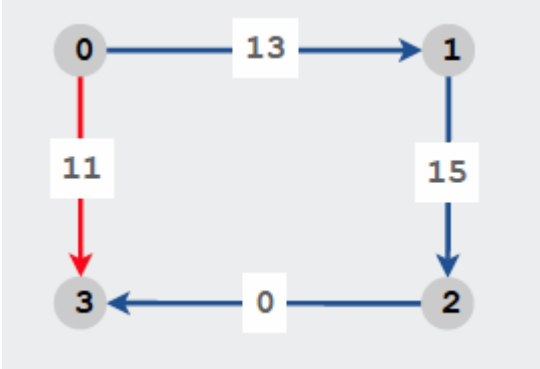


Digrafo con pesos negativos



Dijkstra: Camino mínimo 0 - 3 peso 2
Real: Camino mínimo 0-1-2-3 peso 1

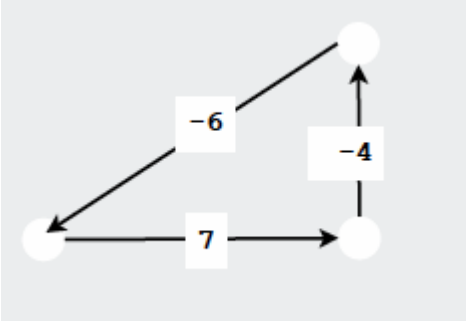
Primera idea: Añadir una constante al peso de cada arco



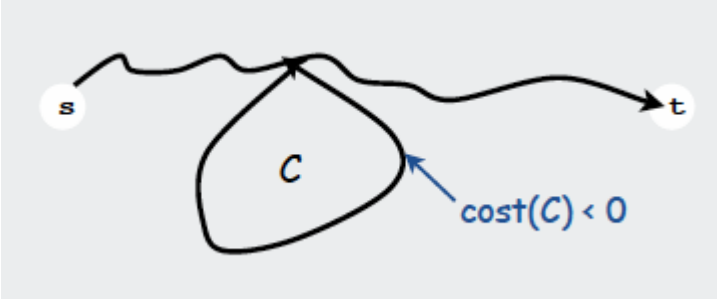
¡Cambia el camino mínimo!

NO VALE

Digrafo con pesos negativos



CICLOS NEGATIVOS



Si hay un ciclo negativo en un recorrido de s hasta t, entonces se puede rebajar indefinidamente el peso del camino mínimo

Algoritmo de Bellman-Ford

- detecta un ciclo negativo, si existe
- encuentra el camino mínimo si no existen ciclos negativos

Algoritmo de Bellman-Ford

Permite pesos negativos y detecta ciclos negativos

Descripción del algoritmo

Entrada: Un digrafo ponderado con pesos no negativos en los arcos, un vértice $s \in V$. El peso del arco uv se indica por w(uv), poniendo $w(uv)=\infty$ si uv no es arco.

Salida: La distancia desde s a cada vértice del grafo

Clave: Mejorar en cada paso las etiquetas de los vértices, t(u)

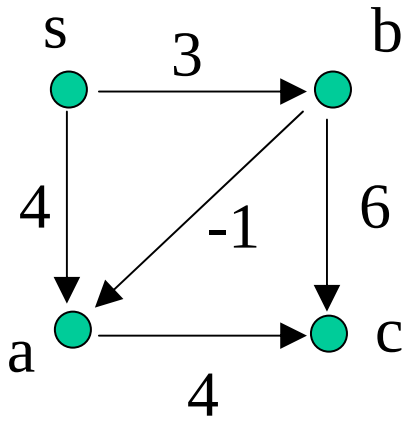
Inicialización: Sea $T=\{s\}$, $t(s)=d(s,s)=0$, $t(z)=w(sz)$ para $z \neq s$.

Iteración: Mientras existan arcos e=xz para los que

$$t(z) > t(x) + w(e)$$

actualizar la etiqueta de z a $\min\{t(z), t(x)+w(xz)\}$

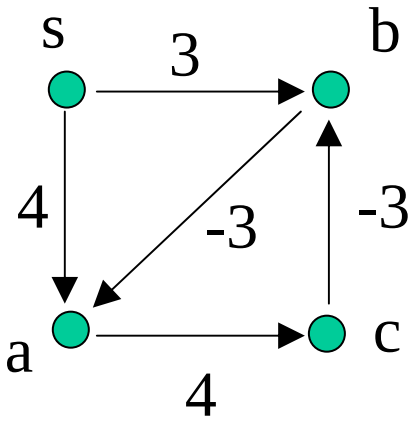
Algoritmo de Bellman-Ford



s	**a**	**b**	**c**
0	∞	∞	∞

sa	0	**4**	∞	∞
sb	0	**4**	**3**	∞
ac	0	**4**	**3**	**8**
ba	0	**2**	**3**	**8**
bc	0	**2**	**3**	**8**

sa	0	**2**	**3**	**8**
sb	0	**2**	**3**	**8**
ac	0	**2**	**3**	**6**
ba	0	**2**	**3**	**6**
bc	0	**2**	**3**	**6**



CICLO NEGATIVO

s	**a**	**b**	**c**
0	∞	∞	∞

sa	0	**4**	∞	∞
sb	0	**4**	**3**	∞
ac	0	**4**	**3**	**8**
ba	0	**0**	**3**	**8**
cb	0	**0**	**3**	**8**

sa	0	**0**	**3**	**8**
sb	0	**0**	**3**	**8**
ac	0	**0**	**3**	**4**
ba	0	**0**	**3**	**4**
cb	0	**0**	**1**	**4**

s	**a**	**b**	**c**
0	0	**1**	**4**

sa	0	**0**	**1**	**4**
sb	0	**0**	**1**	**4**
ac	0	**0**	**1**	**4**
ba	0	**-2**	**1**	**4**
cb	0	**-2**	**1**	**4**

sa	0	-2	**1**	**4**
sb	0	-2	**1**	**4**
ac	0	-2	**1**	**2**
ba	0	**-2**	**1**	**2**
cb	0	-2	-1	2

Análisis de la complejidad

- Cada arco puede considerarse varias veces.
- Se repite el proceso hasta que en una pasada completa (de todos los arcos) no se produzca ningún cambio de etiquetas.
- Si D no contiene ciclos negativos puede demostrarse que, si el camino mínimo $s \longrightarrow u$ contiene k arcos entonces, después de k pasadas se alcanza la etiqueta definitiva para u.
- Complejidad del algoritmo de Ford es O(qn).
- Se detecta un ciclo negativo si se produce una mejora en las etiquetas en la pasada número n.

Algoritmo de Floyd

- Calcula la distancia entre cada par de vértices
- *Idea básica*: Sucesión de matrices W^0, W^1, ..., W^n, donde el elemento ij de la matriz W^k nos indique la longitud del camino mínimo i $\longrightarrow$ j utilizando como vértices interiores del camino los del conjunto $\{v_1, v_2, ..., v_k\}$.
- La matriz W^0 es la matriz de pesos del digrafo,
  w^0_{ij}=w(ij) si existe el arco i $\rightarrow$ j,
  w^0_{ii}=0 y w^0_{ij}=∞ si no existe el arco i $\rightarrow$ j.
- *Clave*: Construimos la matriz W^k a partir de la matriz W^{k-1} así

$$w_{ij}^{k} = \min\{w_{ij}^{k-1}, w_{ik}^{k-1} + w_{kj}^{k-1}\}$$

- El elemento ij de la matriz W^n es la longitud de un camino mínimo entre los vértices i y j

Análisis de la complejidad

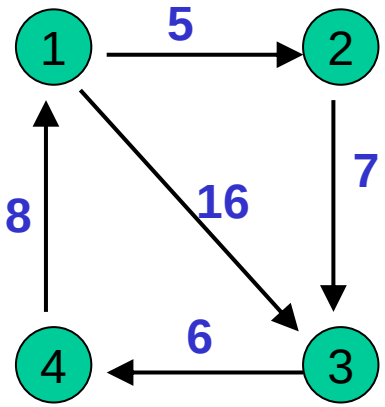
Se deben construir n matrices de tamaño nxn, cada elemento se halla en tiempo constante. La complejidad del algoritmo es $O(n^3)$

Observaciones

1. Si hay un ciclo negativo entonces algún elemento de la diagonal principal se hará negativo en alguna de las matrices W^k
2. Si se desean los caminos mínimos, se construye una sucesión de matrices P^0, P^1,..., P^n :
   - El elemento ij de la matriz P^0 es j (si $w_{ij} \neq \infty$) ó – (si $w_{ij} = \infty$)
   - Si en el elemento ij de la matriz W^k no se produce cambio entonces $p^k_{ij} = p^{k-1}_{ij}$. Si hay cambio entonces $p^k_{ij} = p^{k-1}_{ik}$

Así el elemento ij de la matriz P^n indica el primer vértice después de v_i en el camino de longitud mínima que conecta los vértices v_i y v_j.
Y con esto resulta fácil reconstruir todo el camino.

Algoritmo de Floyd



$$W = \begin{bmatrix} 0 & 5 & 16 & \infty \\ \infty & 0 & 7 & \infty \\ \infty & \infty & 0 & 6 \\ 8 & \infty & \infty & 0 \end{bmatrix} \qquad P = \begin{bmatrix} 1 & 2 & 3 & - \\ - & 2 & 3 & - \\ - & - & 3 & 4 \\ 1 & - & - & 4 \end{bmatrix}$$

$$W^1 = \begin{bmatrix} 0 & 5 & 16 & \infty \\ \infty & 0 & 7 & \infty \\ \infty & \infty & 0 & 6 \\ 8 & 13 & 24 & 0 \end{bmatrix} \quad W^2 = \begin{bmatrix} 0 & 5 & 12 & \infty \\ \infty & 0 & 7 & \infty \\ \infty & \infty & 0 & 6 \\ 8 & 13 & 20 & 0 \end{bmatrix} \quad W^3 = \begin{bmatrix} 0 & 5 & 12 & 18 \\ \infty & 0 & 7 & 13 \\ \infty & \infty & 0 & 6 \\ 8 & 13 & 20 & 0 \end{bmatrix} \quad W^4 = \begin{bmatrix} 0 & 5 & 12 & 18 \\ 21 & 0 & 7 & 13 \\ 14 & 19 & 0 & 6 \\ 8 & 13 & 20 & 0 \end{bmatrix}$$

$$P^1 = \begin{bmatrix} 1 & 2 & 3 & - \\ - & 2 & 3 & - \\ - & - & 3 & 4 \\ 1 & 1 & 1 & 4 \end{bmatrix} \quad P^2 = \begin{bmatrix} 1 & 2 & 2 & - \\ - & 2 & 3 & - \\ - & - & 3 & 4 \\ 1 & 1 & 1 & 4 \end{bmatrix} \quad P^3 = \begin{bmatrix} 1 & 2 & 2 & 2 \\ - & 2 & 3 & 3 \\ - & - & 3 & 4 \\ 1 & 1 & 1 & 4 \end{bmatrix} \quad P^4 = \begin{bmatrix} 1 & 2 & 2 & 2 \\ 3 & 2 & 3 & 3 \\ 4 & 4 & 3 & 4 \\ 1 & 1 & 1 & 4 \end{bmatrix}$$

Algoritmo de Floyd

W

	s	*a*	*b*	*c*	*d*	*e*	*f*	*g*
s	0	5	10	4	5			
a	5	0	4			5	10	
b	10	4	0					
c	4			0	4			
d	5			4	0			10
e		5				0	6	
f		10				6	0	
g					10			0

W_1

	s	*a*	*b*	*c*	*d*	*e*	*f*	*g*
s	0	5	10	4	5			
a	5	0	4	9	10	5	10	
b	10	4	0	14	15			
c	4	9	14	0	4			
d	5	10	15	4	0			10
e		5				0	6	
f		10				6	0	
g					10			0

W_2

	s	*a*	*b*	*c*	*d*	*e*	*f*	*g*
s	0	5	9	4	5	10	15	
a	5	0	4	9	10	5	10	
b	9	4	0	13	14	9	14	
c	4	9	13	0	4	14	19	
d	5	10	14	4	0	15	20	10
e	10	5	9	14	15	0	6	
f	15	10	14	19	20	6	0	
g					10			0

W_3

	s	*a*	*b*	*c*	*d*	*e*	*f*	*g*
s	0	5	9	4	5	10	15	
a	5	0	4	9	10	5	10	
b	9	4	0	13	14	9	14	
c	4	9	13	0	4	14	19	
d	5	10	14	4	0	15	20	10
e	10	5	9	14	15	0	6	
f	15	10	14	19	20	6	0	
g					10			0

W_4

	s	*a*	*b*	*c*	*d*	*e*	*f*	*g*
s	0	5	9	4	5	10	15	
a	5	0	4	9	10	5	10	
b	9	4	0	13	14	9	14	
c	4	9	13	0	4	14	19	
d	5	10	14	4	0	15	20	10
e	10	5	9	14	15	0	6	
f	15	10	14	19	20	6	0	
g					10			0

W_5

	s	*a*	*b*	*c*	*d*	*e*	*f*	*g*
s	0	5	9	4	5	10	15	15
a	5	0	4	9	10	5	10	20
b	9	4	0	13	14	9	14	24
c	4	9	13	0	4	14	19	14
d	5	10	14	4	0	15	20	10
e	10	5	9	14	15	0	6	25
f	15	10	14	19	20	6	0	30
g	15	20	24	14	10	25	30	0

…… **W_8**

…

Herramientas informáticas para visualizar algoritmos de caminos mínimos en grafos desarrolladas por estudiantes de la Facultad de Informática

Algoritmo de Dijkstra
IAGraph http://www.dma.fi.upm.es/personal/gregorio/grafos/web/iagraph/

Los tres algoritmos
http://www.dma.fi.upm.es/personal/gregorio/grafos/web/caminos_minimos/

Medidas de centralidad y excentricidad
http://www.dma.fi.upm.es/personal/gregorio/grafos/web/centralidad_excentricidad/index.html