UNIVERSIDAD
DE GRANADA

Facultad de Ciencias
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Doble Grado en Ingeniería Informática y Matemáticas

TRABAJO DE FIN DE GRADO

Propiedades métricas de grafos y caminos de mínima longitud

Presentado por:
Pablo Cantón Ruiz

Tutor:
Manuel María Ritore Cortés
Departamento de Geometría y Topología

María Dolores Ruiz Jiménez
Departamento de Ciencias de la Computación e Inteligencia Artificial

Curso académico 2022-2023

Propiedades métricas de grafos y caminos de mínima longitud

Pablo Cantón Ruiz

Pablo Cantón Ruiz *Propiedades métricas de grafos y caminos de mínima longitud*.
Trabajo de fin de Grado. Curso académico 2022-2023.

Responsable de tutorización

Manuel María Ritore Cortés
Departamento de Geometría y Topología

María Dolores Ruiz Jiménez
Departamento de Ciencias de la Computación e Inteligencia Artificial

Doble Grado en Ingeniería Informática y Matemáticas

Facultad de Ciencias
Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

Universidad de Granada

Declaración de originalidad

D./Dña. Pablo Cantón Ruiz

Declaro explícitamente que el trabajo presentado como Trabajo de Fin de Grado (TFG), correspondiente al curso académico 2022-2023, es original, entendida esta, en el sentido de que no ha utilizado para la elaboración del trabajo fuentes sin citarlas debidamente.

En Granada a 1 de abril de 2023

Fdo: Pablo Cantón Ruiz

Índice general

Summary

An english summary of the project (around 800 and 1500 words are recommended).
File: `preliminares/summary.tex`

Introducción

De acuerdo con la comisión de grado, el TFG debe incluir una introducción en la que se describan claramente los objetivos previstos inicialmente en la propuesta de TFG, indicando si han sido o no alcanzados, los antecedentes importantes para el desarrollo, los resultados obtenidos, en su caso y las principales fuentes consultadas.

Ver archivo `preliminares/introduccion.tex`

1. Preliminares

1.1. Conceptos sobre espacios métricos

Las definiciones y conceptos de este capítulo han sido extraídos de libros [LC14].

El primer concepto que vamos a repasar y que nos hace falta es el de distancia sobre un conjunto, cuya definición formal es la siguiente:

Definición 1.1. Una *distancia* sobre un conjunto de elementos V es una aplicación $d : V \times V \to \mathbb{R}$ que cumple las siguientes propiedades:

- Anulación:
  $a, b \in X,\ d(a,b) = 0 \Leftrightarrow a = b$
- Simetría:
  $\forall a, b \in X : d(a,b) = d(b,a)$
- Desigualdad triangular:
  $\forall a, b, c \in X : d(a,b) \leq d(a,c) + d(c,b)$

En algunas definiciones se exige como propiedad lo siguiente:

$$\forall a, b \in X : d(a,b) \geq 0$$

Pero no es necesario, pues, supuestas ciertas las demás propiedades, se verifica:

$$0 = d(a,a) \leq d(a,b) + d(b,a) = 2d(a,b),\ \forall a, b \in V$$

Definición 1.2. Una *distancia asimétrica* sobre un conjunto de elementos V es una aplicación $d : V \times V \to \mathbb{R}$ que cumple las siguientes propiedades:

- No negatividad:
  $\forall a, b \in X : d(a,b) \geq 0$
  $a, b \in X,\ d(a,b) = 0 \Leftrightarrow a = b$
- Desigualdad triangular:
  $\forall a, b, c \in X : d(a,b) \leq d(a,c) + d(c,b)$

Definición 1.3. Un *espacio métrico* es un conjunto V con una distancia d asociada, denotaremos a estos espacios como (V, d) o simplemente V cuando la función distancia no sea importante.

Definición 1.4. Dado un espacio métrico, se define la *geodésica* entre dos puntos del espacio como la línea de mínima longitud que une los puntos.

Definición 1.5. Dado un espacio métrico (V, d) y $p \in V$, se define la *bola abierta* de centro p y radio r como el conjunto:

$$B_p(r) = \{x \in V : d(x, p) < r\}$$

Así mismo se define la *bola cerrada* de centro p y radio r como el conjunto:

$$\overline{B}_p(r) = \{x \in V : d(x, p) \leq r\}$$

1.2. Conceptos sobre estructuras de datos

En este apartado veremos algunas estructuras de datos que utilizaremos en la implementación de los algoritmos que trabajaremos además de las principales operaciones que realizaremos con dichas estructuras.

1.2.1. Cola

Una cola es una lista de tipo FIFO (First-in First-out), lo que significa que el primero objeto que entró a la cola de los objetos que contiene es el primero que saldrá cuando se necesite sacar un elemento de la cola.

- **push(*a*):** Inserta el elemento *a* en la cola.
- **pull():** Devuelve y elimina el primer elemento que entró en la cola.

1.2.2. Cola con prioridad

Sigue el mismo funcionamiento que las colas normales, con la salvedad de que a cada elemento se le asigna una *prioridad*, que dependerá de algún criterio en concreto como, por ejemplo, el valor de un atributo. Al extraer un elemento, se extraerá el de máxima prioridad, y, de coincidir varios elementos, se seguirá el orden de cola, es decir, se extraerá el primero que entró.

- **push(*a*):** Inserta el elemento *a* en la cola.
- **pull():** Devuelve y elimina el elemento de máxima prioridad.

2. Teoría de Grafos

Las definiciones y conceptos de este capítulo han sido extraídos de libros [Coro9].

2.1. Conceptos básicos y propiedades asociadas a grafos

Vamos a comenzar viendo algunas definiciones y conceptos básicos.

Definición 2.1. Un *grafo no dirigido* es un par (**V**, **E**), donde **V** es un conjunto no vacío, a cuyos elementos denominaremos *vértices* o *nodos*, y **E** es un conjunto finito de pares de elementos de **V**, a los que llamaremos *aristas* o *lados*.

- A los nodos v_1 y v_2 que forman una arista $e = \{v_1, v_2\}$ se les llama *extremos* de e. Cuando esto ocurra, se dirá que los vértices v_1 y v_2 son *adyacentes* o *vecinos*.
- Se dirá que una arista e es *incidente* con un vértice v cuando v sea uno de sus extremos. En dicho caso se dirá que e *incide* en v.

Definición 2.2. Se denominará *grado de incidencia* de un vértice $v \in V$ (que denotaremos por $g(v)$), al número de aristas incidentes con v.

Por conveniencia, no consideraremos lados que conecten un vértice consigo mismo, es decir, lados del tipo $\{v, v\}$.

En este tipo de grafos no se tiene en cuenta el orden en el que aparecen los vértices en una arista, es decir, $e = \{v_1, v_2\} = \{v_2, v_1\}$. El siguiente tipo de grafo sí que diferencia entre ambas aristas, incluyendo una orientación sobre las mismas.

Definición 2.3. Un *grafo dirigido* es un grafo G cuyas aristas cuentan con una orientación, en este caso pasarán a ser llamadas *arcos*. En este caso denotaremos los arcos como $e = (v_1, v_2)$, siendo e un arco orientado de v_1 hacia v_2.

Cuando planteamos un grafo asociado a un problema, las aristas suelen ser caminos entre nodos, con un coste asociado, para introducir estos costes definimos el siguiente tipo de grafo.

Definición 2.4. Un *grafo ponderado* (o grafo con pesos), es una terna (V, E, ω) donde el par (V, E) representa un grafo (dirigido o no dirigido), y $\omega : E \to \mathbb{R}$ es una aplicación que asigna a cada arista o arco el peso asociado.

En la Figura 2.1 podemos observar ejemplos de las definiciones anteriores, por ejemplo, los vértices 0 y 2 son adyacentes, y el grado de incidencia del vértice a es 4.

A continuación plantearemos algunas definiciones sobre distintos tipos de grafos.

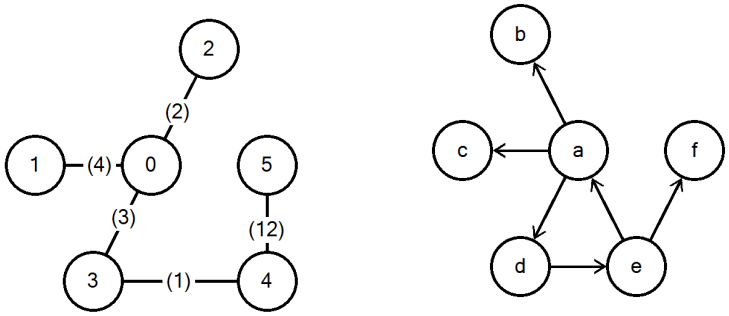


Figura 2.1.: Ejemplo de grafo no dirigido ponderado, izquierda, y grafo dirigido no ponderado, derecha.

Definición 2.5. Un *grafo finito* es un grafo G (dirigido o no dirigido), con un número finito de vértices, es decir, el conjunto V es finito.

Definición 2.6. Un *grafo de incidencia finita* es un grafo G donde el grado de incidencia de cada vértice es finito, es decir, $g(v)$ es finito para cada $v \in V$.

Definición 2.7. Dado un grafo $G = (V, E)$:

- Se dirá que $G' = (V', E')$ es un *subgrafo* de G si $V' \subseteq V$ y $E' \subseteq E$.
- Se dirá que $G' = (V, E')$ es un *subgrafo parcial* de G si $E' \subseteq E$.

Definición 2.8. Dado un grafo $G = (V, E)$ y $B \subseteq V$, se dirá *subgrafo de G inducido por B* a $G_B = (B, E_B)$ con

$$E_B = \{(i,j) \in E : \ i,j \in B\} \mid E_B = \{\{i,j\} \in E : \ i,j \in B\}$$

Un concepto natural que surge al hablar de grafos es el de camino entre dos nodos, definimos a continuación formalmente el concepto tanto del camino como tal como de la longitud del mismo.

Definición 2.9. Dado un grafo G, un *camino* es una sucesión de vértices $v_1v_2...v_{n+1}$ y aristas $e_1e_2...e_n$ tal que $e_i = \{v_i, v_{i+1}\}$ si es un grafo no dirigido, o bien $e_i = (v_i, v_{i+1})$ si es un grafo dirigido, $\forall i \in \{1, ..., n\}$. Para referirnos a un camino de ahora en adelante, utilizaremos la sucesión de vértices $v_1...v_{n+1}$ si no es necesario conocer las aristas en concreto, o $e_1...e_n$ si necesitamos conocerlas.

Definición 2.10. Se define la *longitud* de un camino $v_1v_2...v_{n+1}$ con aristas $e_1e_2...e_n$ como sigue:

$$Long(v_1v_2...v_{n+1}) = \sum_{i=1}^{n} \omega(e_i)$$

En caso de que no sea un grafo ponderado, supondremos coste 1 para todas las aristas, en cuyo caso, la longitud del camino será n.

Definimos a continuación una función que nos será útil más adelante.

Definición 2.11. Dado un grafo finito $G = (V, E)$ y $v_0, v_1 \in V$, Definimos $d : V \times V \to \mathbb{R} \cup \{\infty\}$ como sigue:

$$d(v_0, v_1) = \begin{cases} min\{Long(v_0...v_1) : v_0...v_1 \text{ camino desde } v_0 \text{ a } v_1\} & \text{si existe camino entre } v_0 \text{ y } v_1 \\ \infty & \text{en otro caso} \end{cases}$$

Sabemos que existe el mínimo, pues, como los conjuntos de vértices y aristas son finitos, el conjunto de caminos entre dos vértices es también finito.

Probaremos a continuación que, tal y como sugiere la definición, d es un distancia.

Proposición 2.1. *Sea G un grafo finito no dirigido, no ponderado o ponderado con pesos exclusivamente positivos, se verifica que $d : V \times V \to \mathbb{R}$ con la definición anterior es una distancia en G.*

Demostración. Dado que los únicos caminos de longitud 0 son aquellos en los que no hay aristas, es claro que $d(v_0, v_1) = 0 \Leftrightarrow v_0 = v_1$. La simetría es clara, pues, en un grafo no dirigido, un camino desde v_0 a v_1 es también un camino desde v_1 a v_0, dado que podemos recorrer las aristas en los dos sentidos. Para probar la desigualdad triangular, es decir, $\forall v_0, v_1, v_2 \in V,\ d(v_0, v_1) \leq d(v_0, v_2) + d(v_2, v_1)$, hay que distinguir dos casos, el primero, si no existe ningún camino que conecte v_0 con v_2 o ningún camino que conecte v_2 con v_1, en cuyo caso se tendría $d(v_0, v_1) \leq \infty$, por lo que siempre se verifica la desigualdad. En caso contrario, si existieran caminos c_1 y c_2 que conectan v_0 con v_2 y v_2 con v_1, respectivamente, $d(v_0, v_1) \leq Long(c) = Long(c_1) + Long(c_2)$, donde c es el camino que resulta de unir c_1 y c_2, esta última desigualdad es cierta por la definición de d, para cualquier par de caminos c_1, c_2, por lo que, tomando ínfimos sobre estos caminos, obtenemos que $d(v_0, v_1) \leq d(v_0, v_2) + d(v_2, v_1)$, como se quería. □

Proposición 2.2. *Sea G un grafo dirigido, no ponderado o ponderado con pesos exclusivamente positivos, se verifica que $d : V \times V \to \mathbb{R}$ con la definición anterior es una distancia asimétrica en G.*

Esta proposición no necesita demostración, pues los argumentos utilizados para probar la no negatividad y la desigualdad triangular en la Proposición 2.1 son válidos también para grafos dirigidos.

Definición 2.12. Dado un grafo $G = (V, E)$ y un camino $v_0...v_1$ entre los vértices v_0 y v_1, a dicho camino se le llama *camino de longitud mínima* desde v_0 a v_1 si y solo si es la geodésica entre v_0 y v_1, es decir:

$$Long(v_0...v_1) = d(v_0, v_1)$$

Nótese que en el grafo, las líneas entre puntos son caminos entre vértices. Veremos a continuación un par de propiedades muy interesantes asociadas a este tipo de caminos.

Proposición 2.3. *Dado un camino de longitud mínima $v_0...v_i...v_1$ desde v_0 a v_1, se verifica:*

- *$v_0...v_i$ es un camino de longitud mínima desde v_0 a v_i.*
- *$v_i...v_1$ es un camino de longitud mínima desde v_i a v_1.*

Demostración. Sea $e_1...e_{i-1}$ el conjunto de aristas asociadas al camino entre v_0 y v_i, entonces, por contradicción, supongamos que $v_0...v_i$ no es un camino de longitud mínima (el caso en que $v_i...v_1$ no es un camino de longitud mínima es análogo), al no ser camino de longitud mínima, debe existir otro camino entre v_0 y v_i de longitud menor, es decir, $\exists e'_1...e'_m$ camino entre v_0 y v_i con $Long(e'_1...e'_m) < Long(e_1...e_{i-1})$. Llamemos $e_i...e_n$ al conjunto de aristas del camino entre v_i y v_1, entonces, juntando el nuevo camino entre v_0 y v_i con éste último, obtenemos:

$$\begin{aligned} Long(e'_1...e'_m e_i...e_n) &= Long(e'_1...e'_m) + Long(e_i...e_n) \\ &< Long(e_1...e_{i-1}) + Long(e_i...e_n) \\ &= Long(e_1...e_n) \end{aligned}$$

Esto es una contradicción pues, por hipótesis, $Long(e_1...e_n) = d(v_0, v_1)$, al ser $e_1...e_n$ camino de longitud mínima, pero hemos encontrado entonces un camino entre v_0 y v_1, el cual es $e'_1...e'_m e_i...e_n$, de longitud menor, lo que no puede ser. □

Proposición 2.4. *Sea $G = (V, E, \omega)$ un grafo de incidencia finita ponderado con pesos positivos tal que:*

- $\omega(e) \geq a > 0 \quad \forall e \in E$.

Entonces, si existe un camino entre dos nodos, existe un camino de longitud mínima entre ambos (geodésica).

Demostración. Sean $v_0, v_1 \in V$ tales que existe un camino de longitud $L > 0$ entre ambos, si c es otro camino entre v_0 y v_1 de K aristas, entonces

$$Long(c) \geq Ka$$

por hipótesis.
Si $Ka > L$, c no puede ser una geodésica, consideramos entonces el subgrafo inducido por los vértices que se pueden unir con v_0 mediante caminos de $K \leq \frac{L}{a}$ lados. El vértice v_1 pertenece a dicho grafo por el camino inicial. Ahora, por ser G un grafo de incidencia finita, se tiene que dicho subgrafo es finito y, por tanto, existe una geodésica que une v_0 con v_1 en el subgrafo y, dado que el resto de caminos c entre v_0 y v_1 tienen longitud $Long(c) > L$, la geodésica en el subgrafo es geodésica también en G. □

Para trabajar con grafos, necesitamos representarlos de alguna manera, para ello se definen a continuación dos estructuras muy utilizadas a la hora de la representación de grafos:

Definición 2.13. Dado un grafo finito G, con $|V| = n$, llamaremos *matriz de adyacencia* de G a una matriz $A_{n\times n} = (a_{ij})$ de tal manera que:

$$a_{ij} = \begin{cases} 1 & si\ (v_i, v_j) \in E\ (\{v_i, v_j\}\ o\ \{v_j, v_i\} \in E) \\ 0 & en\ otro\ caso \end{cases}$$

Definición 2.14. Dado un grafo finito G, con $|V| = n$, se define la *lista de adyacencia* de $v \in V$ como:

$$Adj(v) = \{u \in V : (v, u) \in E(\{v, u\}\ o\ \{u, v\} \in E)\}$$

De tal manera que para representar el grafo, basta calcular la lista de adyacencia de cada vértice.

3. Algoritmos de búsqueda en grafos

En este capítulo abordaremos algunos algoritmos de búsqueda en grafos, con el objetivo de encontrar caminos entre puntos, geodésicas y componentes conexas asociadas a los distintos grafos. En esta sección nos referiremos a los vértices del grafo por nodos.

3.1. Algoritmo de búsqueda en anchura

El algoritmo de búsqueda en anchura (BFS-Breadth First Search en inglés) es un algoritmo de búsqueda no informada, es decir, que cuando se da un paso en el algoritmo, no se sabe si este es el mejor posible. Es uno de los algoritmos más simples de búsqueda en grafos y funciona como base para el desarrollo de otros algoritmos más complejos. La idea intuitiva del algoritmo es partir del nodo inicial e ir explorando todos los nodos vecinos (conectados mediante un camino), cuando se han explorado todos los vecinos, se repite el proceso para cada uno de estos, hasta recorrer todo el grafo.

El nombre de búsqueda en anchura viene del hecho de que el algoritmo explora la frontera entre nodos explorados y no explorados, de manera uniforme a lo largo del ancho de la misma, es decir, explora todos los nodos a distancia k antes de explorar un nodo a distancia $k+1$.

El algoritmo funciona tanto en grafos no dirigidos como grafos dirigidos, sin embargo, requiere que los pesos de las aristas sean 1, es decir, que sea un grafo no ponderado. El algoritmo se puede utilizar tanto para calcular las componentes conexas de un grafo como para encontrar los caminos de mínima longitud, geodésicas, entre dos nodos.

Desde un punto de vista matemático, sea p el nodo inicial, el algoritmo consiste en explorar la bola cerrada $\overline{B}_p(r)$ empezando en $r = 1$, marcando los elementos dentro de la bola, e ir aumentando en cada paso el radio en una unidad hasta que no se puedan alcanzar más elementos desde el nodo inicial, es decir, hasta que $\overline{B}_p(r) \cap \overline{B}_p(r+1) = \emptyset$. Para el cálculo de componentes conexas, si, tras terminar el proceso anterior, quedan nodos sin explorar, se selecciona un nodo no explorado y se repite el proceso, hasta que todos los nodos sean explorados. Para el cálculo de las geodésicas la exploración continúa hasta encontrar el nodo destino, si no se encuentra, no existe camino entre los nodos.

Proposición 3.1. *Sea $v \in V$ el nodo inicial y $p \in V$ el nodo final, el camino encontrado por el algoritmo de búsqueda en anchura entre estos nodos es una geodésica.*

Demostración. Sea l la longitud del camino encontrado, por construcción del algoritmo, esto supone dos cosas, por un lado, $p \in \overline{B}_v(l)$, y, por otro lado, $p \notin \overline{B}_v(r)\ \forall r < l$ pues, de lo contrario, el algoritmo hubiera parado antes de explorar $\overline{B}_v(l)$ y la longitud sería menor. Pero entonces esto supone que no existe ningún camino entre v y p de longitud menor que l, es decir, que el camino encontrado de longitud l es una geodésica, como se quería. □

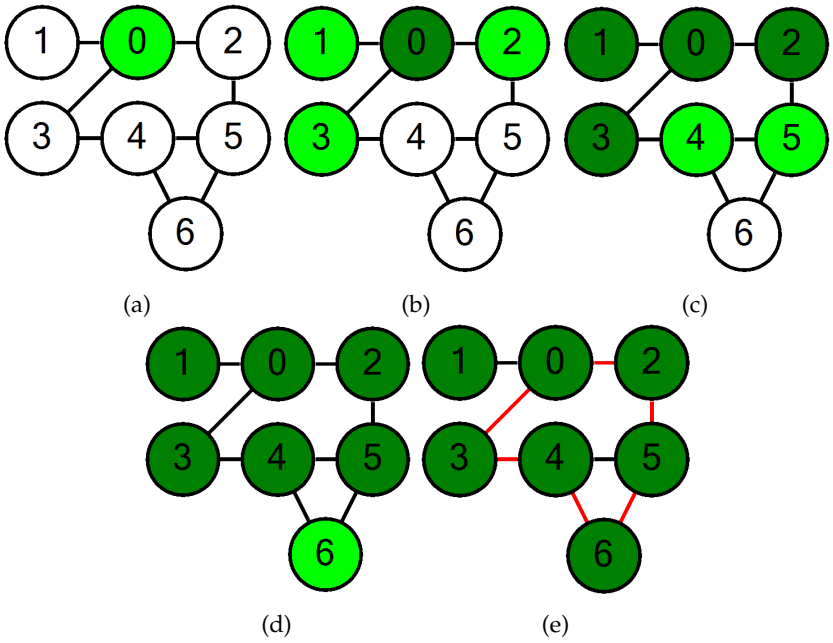


Figura 3.1.: Cálculo de las geodésicas entre el nodo 0 y el nodo 6.

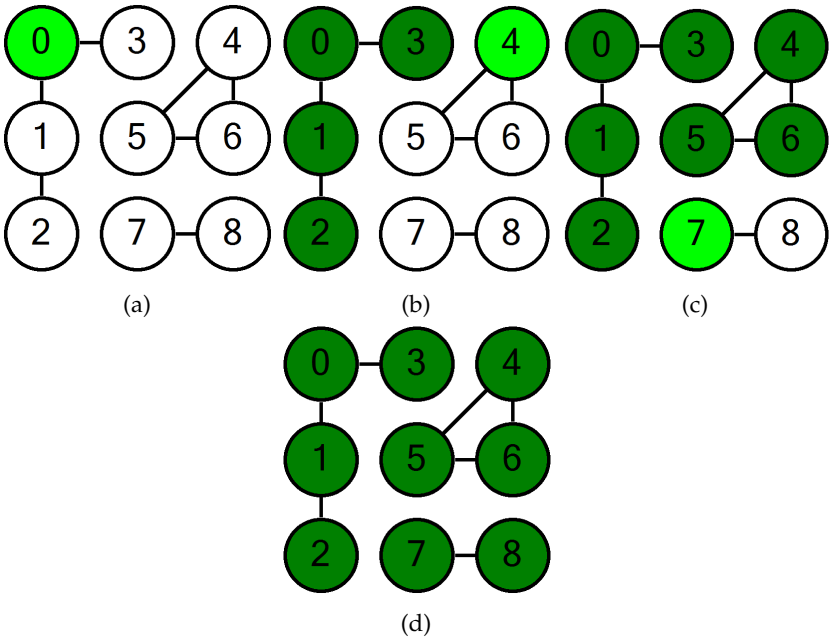


Figura 3.2.: Cálculo de las componentes conexas del grafo, en este caso, 3.

En la Figura 3.1 y la Figura 3.2 podemos observar un ejemplo del comportamiento del algoritmo, tanto para encontrar las geodésicas entre dos nodos como para calcular el número de componentes conexas. En el primer caso, se muestra en cada figura el estado de los nodos

tras la exploración completa de cada bola, marcando en verde oscuro los nodos explorados y en verde pistacho los nodos visitados que serán explorados en la siguiente bola, por ejemplo, en el apartado (c) acaba de finalizar la exploración de la bola $\overline{B}_0(1)$. En la segundo figura se muestra el estado del grafo tras finalizar el algoritmo base sobre cada componente, y haber seleccionado el siguiente nodo a explorar.

3.1.1. Implementación del algoritmo de búsqueda en anchura

A continuación veremos la implementación mediante pseudocódigo de las dos versiones del algoritmo que trabajaremos, la primera, cuya finalidad será la de encontrar todas las geodésicas entre dos nodos, y la segunda, que calculará el número de componentes conexas del grafo.

Para la construcción del algoritmo, utilizaremos la representación mediante listas de adyacencia y, para representar los atributos de los nodos, como por ejemplo el predecesor, se utilizará la notación $v.d$, siendo v el nodo y d el atributo.

Empezamos por el algoritmo de cálculo de geodésicas entre dos nodos.

Algorithm 1 BFS_geodesicas(G, v, p)

```
1:  for u ∈ G.V do
2:      u.explorado = False
3:      u.distancia = ∞
4:      u.predecesor = {}
5:  v.distancia = 0
6:  Cola Q = ∅
7:  Q.push(v)
8:  while Q ≠ ∅ do
9:      u = Q.pull()
10:     if u == p then return True
11:     for n ∈ G.Adj[u] do
12:         if !n.explorado then
13:             if n.distancia >= u.distancia + 1 then
14:                 n.distancia = u.distancia + 1
15:                 n.predecesor.push(u)
16:                 Q.push(n)
17:     u.explorado = True
    return False
```

En el algoritmo anterior, G es el grafo, v el nodo inicial, p el nodo final y Q es una cola.

Esta versión del algoritmo es muy similar a la versión normal del algoritmo, con la excepción de que, para poder encontrar todas las geodésicas, debemos considerar nodos ya visitados, pues pueden ser explorados por varios nodos a la misma distancia. Para evitar que antes de explorar un nodo sea visitado y, por tanto, alterado, por otro nodo a distancia mayor que la primera vez que fue visitado, se ha incluido el condicional de la línea 13, nótese que, al iniciar las distancias a infinito, la primera vez que se visita un nodo, este condicional es siempre cierto.

Pasamos a continuación con la versión del algoritmo que calcula el número de componentes conexas del grafo.

Algorithm 2 BFS_conexas(G)

```
1:  N = ∅
2:  conexas = 0
3:  for u ∈ G.V do
4:      u.explorado = False
5:      N.push(u)
6:  while N ≠ ∅ do
7:      conexas = conexas + 1
8:      u = N.pull()
9:      u.explorado = True
10:     Cola Q = ∅
11:     Q.push(u)
12:     while Q ≠ ∅ do
13:         u = Q.pull()
14:         for n ∈ G.Adj[u] do
15:             if !n.explorado then
16:                 Q.push(n)
17:                 n.explorado = True
18:                 N.remove(n)
    return conexas
```

Para asegurarnos de que se exploran todos los nodos, los incluimos primero en un conjunto, del que los sacamos conforme los exploramos, y continuamos explorando hasta que dicho conjunto sea vacío. Puesto que, para el cálculo de las componentes conexas tan solo necesitamos conocer si podemos alcanzar un nodo desde el inicio, marcamos como explorado los nodos en la adyacencia del nodo que estamos explorando, de tal manera nos aseguramos que solo entra a la cola una vez.

3.2. Algoritmo de Dijkstra

El siguiente algoritmo que vamos a estudiar es el algoritmo de Dijkstra, al igual que la búsqueda en anchura, es un algoritmo de búsqueda del camino más corto entre nodos, con la salvedad de que funciona también para grafos ponderados. Su nombre proviene del científico Edsger Dijkstra, que lo ideó en 1956 y publicó por primera vez en 1959.

La idea del algoritmo es en esencia, la misma que la búsqueda en anchura, con la salvedad de que la exploración de los nodos se realiza de forma ordenada, es decir, en cada paso, se escoge el nodo que esté a distancia menor de todos los que han sido visitados, para explorar. Al igual que el algoritmo de búsqueda en anchura, este algoritmo no funciona si existen aristas con pesos negativos.

A nivel de espacios métricos, sea p el nodo inicial, el primer paso es calcular el mínimo de las distancias del resto de nodos a p, sea r dicha distancia, entonces procederemos a explorar la bola cerrada $\overline{B}_p(r)$, en dicha bola se encuentran el nodo inicial y, al menos, un

nodo más, que está a distancia r. El siguiente paso es calcular r', que va a ser el mínimo entre las distancias de los nodos no explorados a p, y las distancias de los nodos no explorados a los nodos del conjunto $\overline{B}_p(r)\backslash\{p\}$ más la distancia de dichos nodos a p, es decir:

$$r' = min\{d(p,v) :\ v \in V\backslash\overline{B}_p(r),\ d(p,v) + d(v,u) :\ v \in \overline{B}_p(r)\backslash\{p\},\ u \in V\backslash\overline{B}_p(r)\}$$

A continuación se explora la bola cerrada $\overline{B}_p(r')$ y el proceso se repite hasta encontrar el nodo destino.

Proposición 3.2. *Sea $v \in V$ el nodo inicial y $p \in V$ el nodo final, el camino encontrado por el algoritmo de Dijkstra entre estos nodos es una geodésica.*

Demostración. Puesto que $d(p,v) > r,\ v \in V\backslash\overline{B}_p(r)$, por elección de r, y $d(p,v) + d(v,u) > r,\ v \in \overline{B}_p(r)\backslash\{p\},\ u \in V\backslash\overline{B}_p(r)$, pues, de lo contrario, $d(p,u) \leq d(p,v) + d(v,u) \leq r$, pero esto significa que $u \in \overline{B}_p(r)$, lo que es una contradicción. Por tanto, se verifica que, en cada paso del algoritmo, $r' > r$, es decir, que la distancia de los nodos explorados es siempre creciente, por lo que la demostración de la Proposición 3.1 es cierta también para este algoritmo. □

3.2.1. Implementación del algoritmo de Dijkstra

A nivel de implementación el algoritmo difiere un poco de la definición matemática, pues es muy ineficiente tener que actualizar el radio en cada paso, teniendo en cuenta además todos los nodos del grafo. Por esto, se introduce el concepto de nodos *visitados*, que son los nodos vecinos de un nodo explorado, es decir, nodos en la frontera del conjunto de nodos explorados. Se introduce este concepto pues, tras la exploración de un nodo, el siguiente nodo que escoge el algoritmo es el de menor distancia y éste es siempre un nodo en la frontera, es decir, un nodo marcado como visitado, pues el resto de nodos del grafo están a mayor distancia de los nodos en la frontera, pues los pesos de las aristas son positivos.

El procedimiento del algoritmo entonces consiste en explorar el nodo visitado a menor distancia del nodo inicial. La exploración del nodo consiste en marcar todos los nodos vecinos no explorados como visitados y actualizar su distancia al nodo inicial a partir del peso de la arista que los une y la distancia del propio nodo.

Este procedimiento mantiene invariante la propiedad principal del algoritmo, la distancia de los nodos explorados es siempre creciente a medida que avanza el algoritmo, lo que permite mantener la corrección del algoritmo probada en la Proposición 3.2, es decir, lo caminos encontrados de esta manera son geodésicas.

En la Figura 3.3 se puede observar un ejemplo del algoritmo, se empieza marcando el nodo inicial, a, como visitado, con distancia 0, y el resto de nodos con distancia -1, para señalar que no han sido visitados todavía. A continuación se escoge el nodo visitado (en verde pistacho) con menor distancia y se explora, es decir, se marcan todos sus vecinos no explorados y no visitados como visitados y se actualizan las distancias de dichos nodos como la distancia del propio nodo más el peso de la arista que los une. En caso de que un nodo vecino esté marcado como visitado, si la distancia desde el nodo actual es menor a la distancia guardada, se actualiza, como se puede observar en el apartado (g). El algoritmo

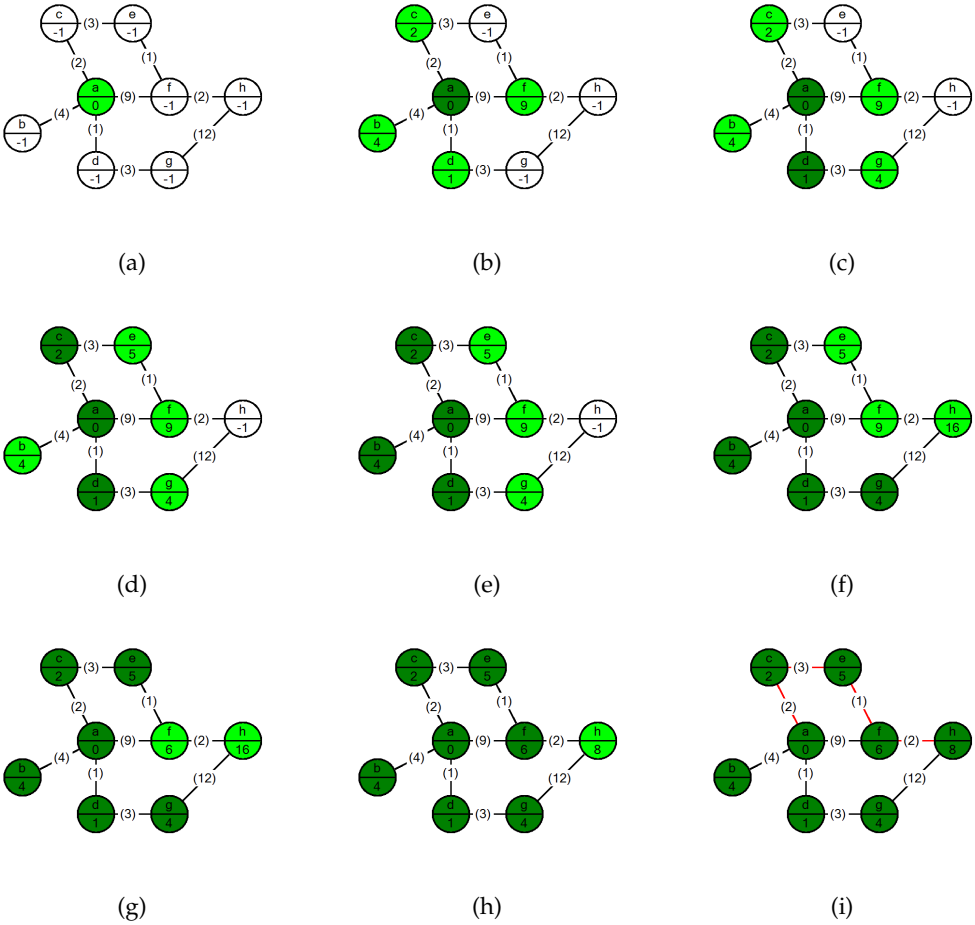


Figura 3.3.: Cálculo de las geodésicas entre el nodo a y el nodo h.

termina cuando se explora el nodo final o no quedan nodos visitados que explorar.

Este comportamiento se puede simular utilizando una cola con prioridad en el algoritmo BFS en vez de una cola normal, de tal manera que el nodo escogido en cada paso sea el de menor distancia.

Se muestra a continuación el pseudocódigo asociado a la implementación del algoritmo.

Algorithm 3 BFS_geodesicas(G, v, p)

1: **for** $u \in G.V$ **do**
2: $u.explorado = False$
3: $u.distancia = \infty$
4: $u.predecesor = \{\}$
5: $v.distancia = 0$
6: *Cola con prioridad* $Q = \emptyset$

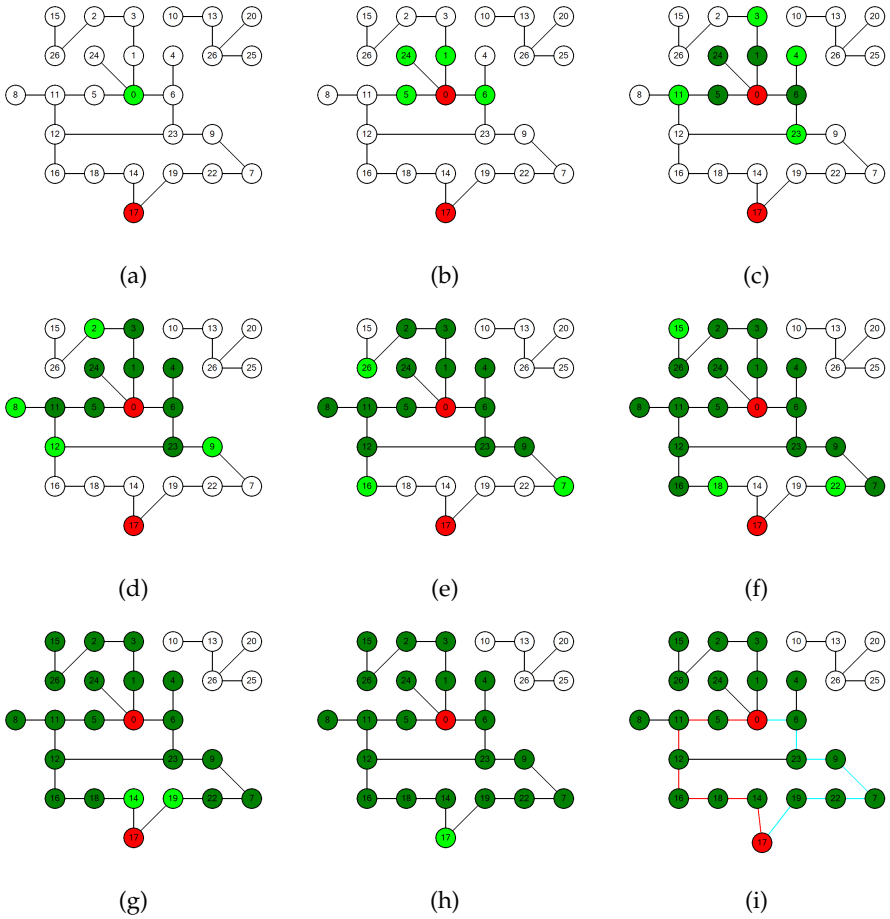
```
7: Q.push(v)
8: while Q ≠ ∅ do
9:     u = Q.pull()
10:    if u == p then return True
11:    for n ∈ G.Adj[u] do
12:        if !n.explorado then
13:            if n.distancia > u.distancia + d[u][n] then
14:                n.distancia = u.distancia + d[u][n]
15:                n.predecesor = {u}
16:                Q.push(n)
17:            else if n.distancia == u.distancia + d[u][n] then
18:                n.predecesor.push(u)
19:    u.explorado = True
   return False
```

En el algoritmo anterior, G es el grafo, v el nodo inicial, p el nodo final, Q una cola con prioridad y los pesos de las aristas vienen dados en una matriz, donde el valor $d[a][b]$ corresponde al peso de la arista que une a con b.

A. Primer apéndice

A.1. Ejemplo extendido de cálculo de geodésicas con BFS

Con afán de mostrar el procedimiento del algoritmo BFS sobre un grafo de mayor extensión, se presenta a continuación en la Figura A.1 el proceso de búsqueda de geodésicas, donde cada imagen representa el estado del grafo al finalizar una capa, es decir, tras explorar por completo cada bola, el nodo inicial es el nodo 0, y el nodo final está marcado en rojo para mejor visibilidad. Los nodos ya explorados se han marcado en verde oscuro, y los nodos visitados pero no explorados, es decir, los nodos en la cola, en verde pistacho. Los caminos marcados en rojo representan las geodésicas encontradas, que se pueden calcular a través de los predecesores de cada nodo.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

Figura A.1.: Cálculo de las geodésicas entre el nodo 0 y el nodo 17, marcado en rojo.

Glosario

La inclusión de un glosario es opcional.
Archivo: `glosario.tex`

$\mathbb{R}$ Conjunto de números reales.

$\mathbb{C}$ Conjunto de números complejos.

$\mathbb{Z}$ Conjunto de números enteros.

Bibliografía

Las referencias se listan por orden alfabético. Aquellas referencias con más de un autor están ordenadas de acuerdo con el primer autor.

[Cor09] T. H. Cormen. *Introduction to algorithms / Thomas H. Cormen... [et al]*. MIT Press, Cambridge, Mass, 3rd ed. edition, 2009.

[LC14] Rafael López Camino. *Topología / Rafael López Camino*. Manuales major ; 67. Ciencias experimentales y exactas. Universidad de Granada, Granada, 2014.