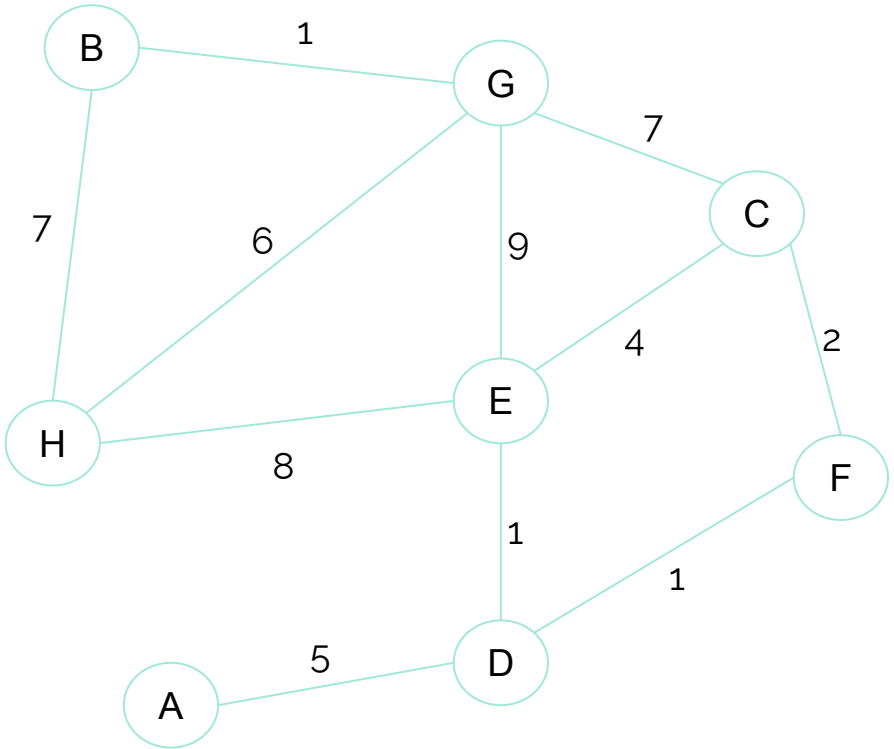
Dijkstra

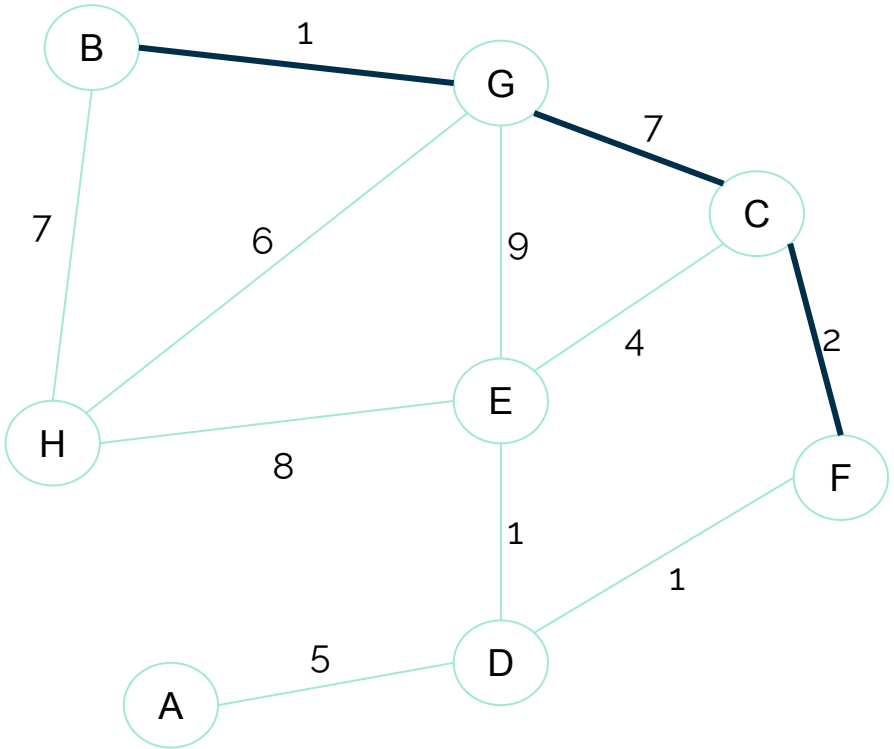
Isabella Cherubini (isabella.cherubini@uc.cl)

El camino más corto



Definimos el camino más corto entre dos nodos como la menor suma de las aristas que los conectan

El camino más corto



Por ejemplo, el camino más corto entre B y F

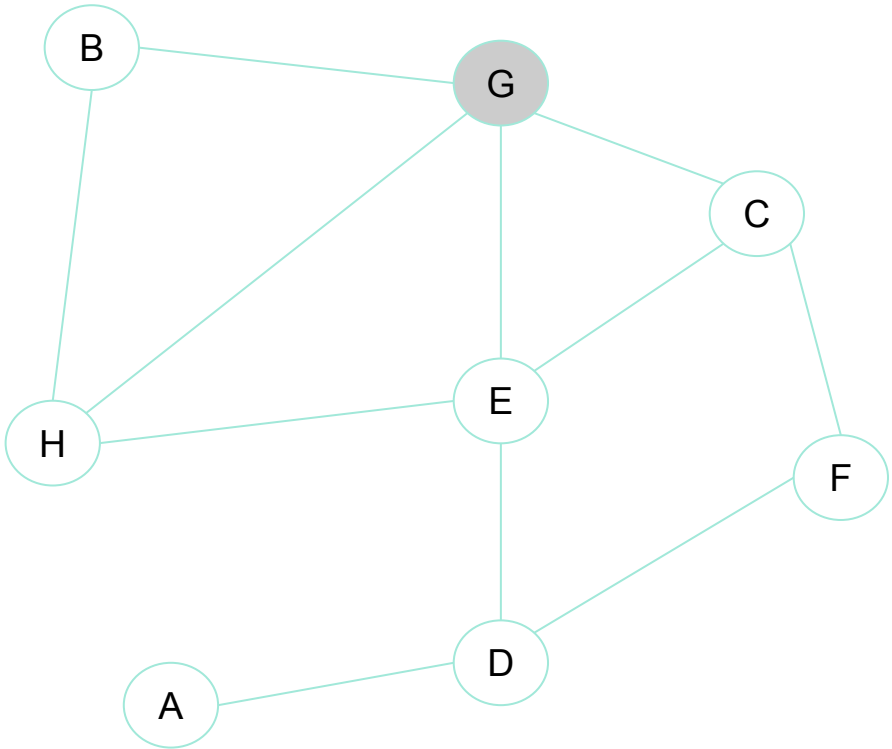
¿Cómo podemos encontrar este camino más corto?

BFS

Antes de ver cómo encontramos el camino más corto, debemos entender como funciona BFS:

- Es un algoritmo de búsqueda en amplitud
- Partiendo de un nodo del grafo, recorremos primero los nodos que están a una arista de distancia, luego a dos aristas de distancia, y así hasta llegar al último nodo
- Marcamos los nodos que ya visitamos con el fin de no revisar infinitamente
- Para esto usamos una cola FIFO, cada vez que encontramos un nodo nuevo lo agregamos al final de la cola, y para revisar nuevos sacamos el primer nodo

BFS

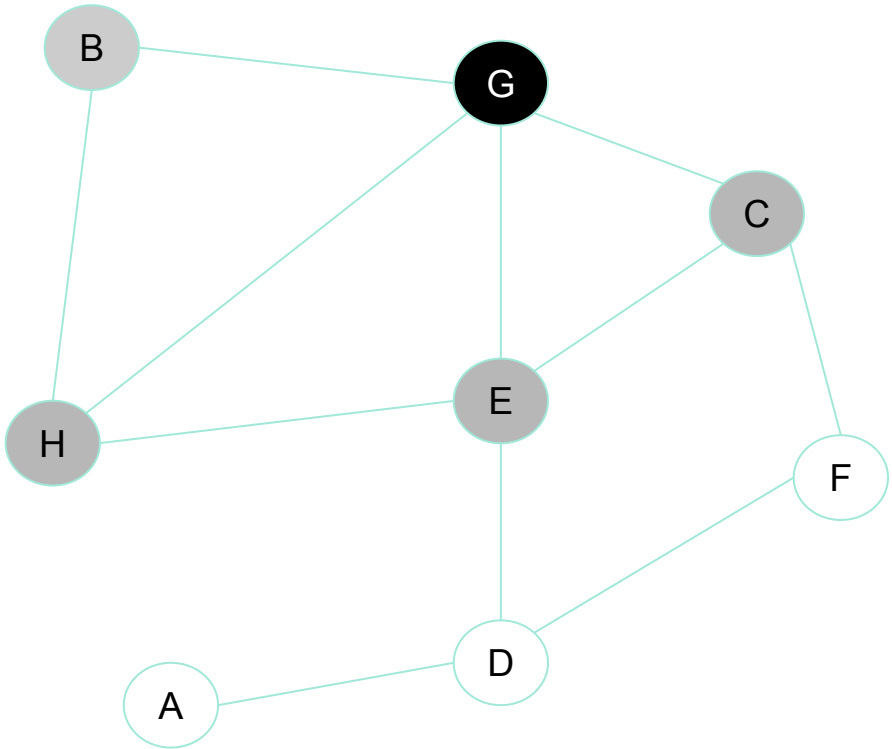


Partimos seleccionando un nodo (por ejemplo G)

Cada vez que agregamos un nodo a la cola lo marcamos de gris, cada vez que sacamos un nodo lo marcamos de negro

Q = {G}

BFS

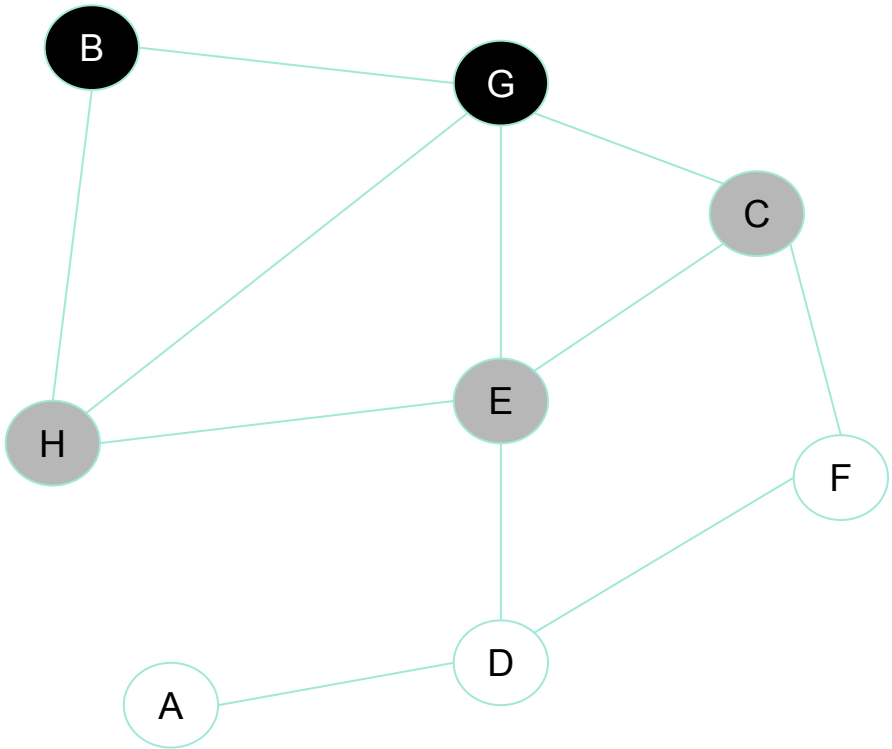


Si un nodo es gris o negro, no lo volvemos a agregar

Cada vez que agregamos un nodo a la cola lo marcamos de gris, cada vez que sacamos un nodo lo marcamos de negro

Q = {B, H, E, C}

BFS

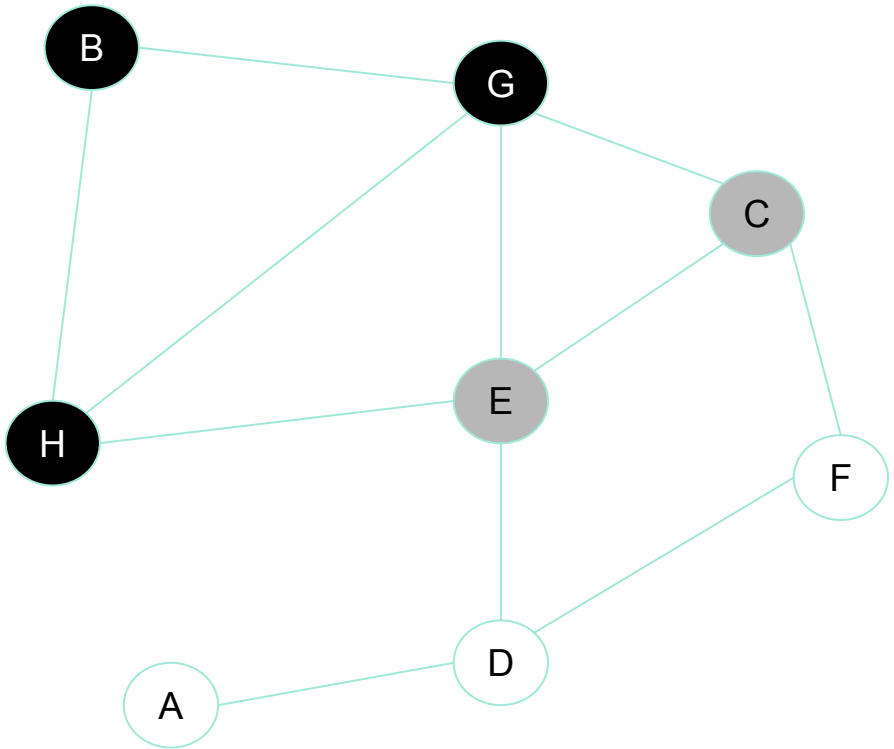


Si un nodo es gris o negro, no lo volvemos a agregar

Cada vez que agregamos un nodo a la cola lo marcamos de gris, cada vez que sacamos un nodo lo marcamos de negro

Q = {H, E, C}

BFS

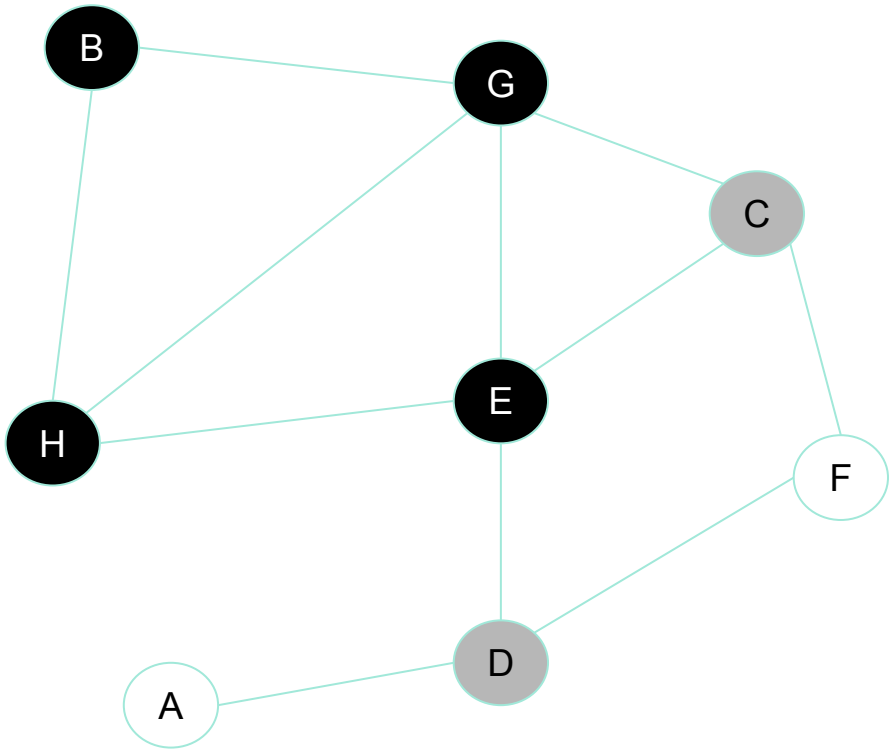


Si un nodo es gris o negro, no lo volvemos a agregar

Cada vez que agregamos un nodo a la cola lo marcamos de gris, cada vez que sacamos un nodo lo marcamos de negro

Q = {E, C}

BFS

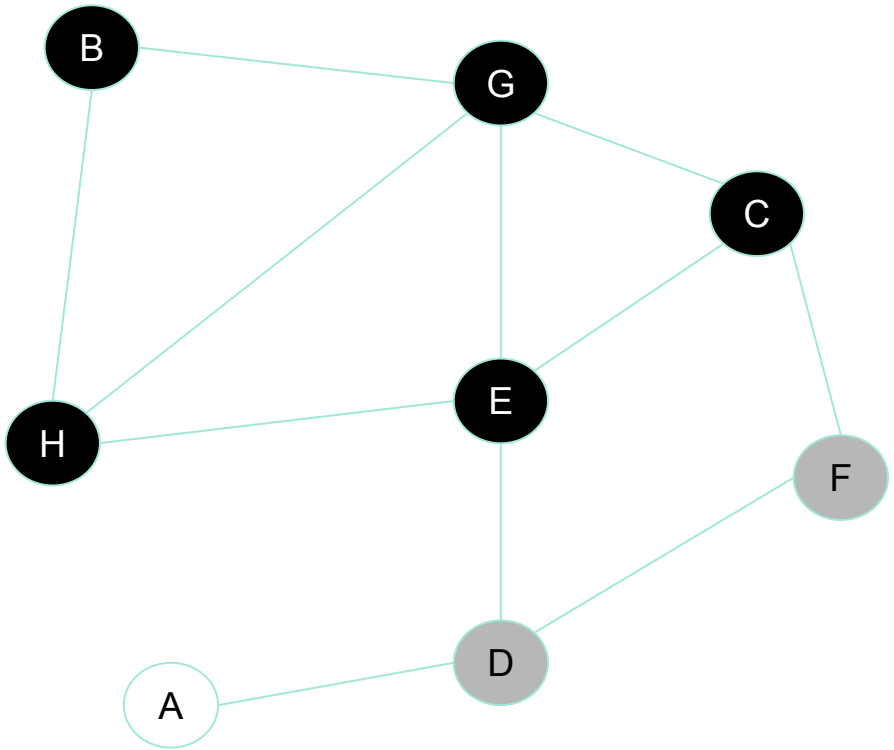


Si un nodo es gris o negro, no lo volvemos a agregar

Cada vez que agregamos un nodo a la cola lo marcamos de gris, cada vez que sacamos un nodo lo marcamos de negro

Q = {C, D}

BFS

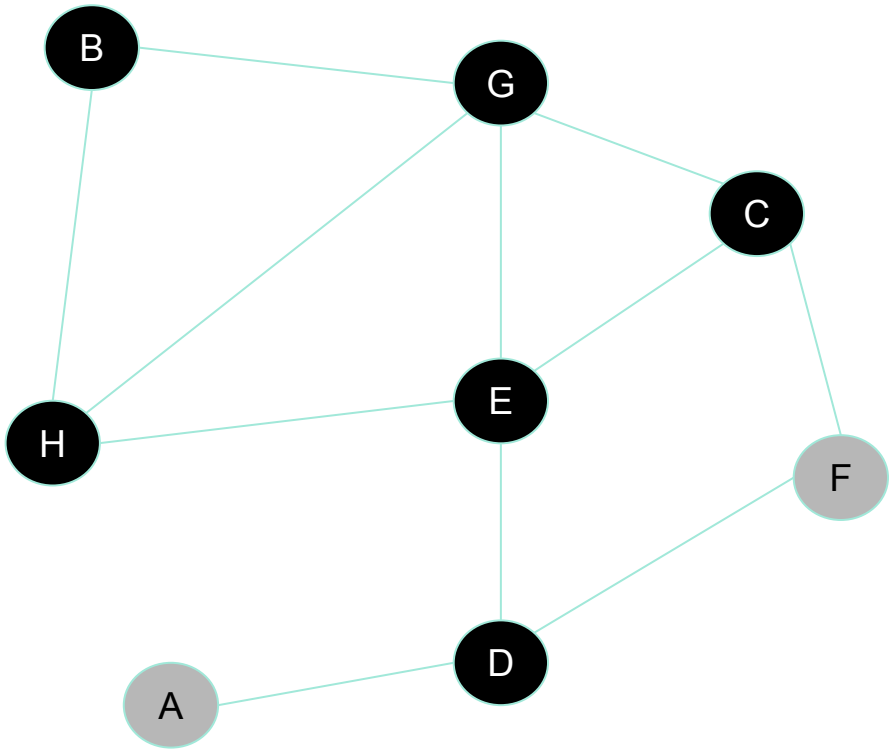


Si un nodo es gris o negro, no lo volvemos a agregar

Cada vez que agregamos un nodo a la cola lo marcamos de gris, cada vez que sacamos un nodo lo marcamos de negro

Q = {D, F}

BFS

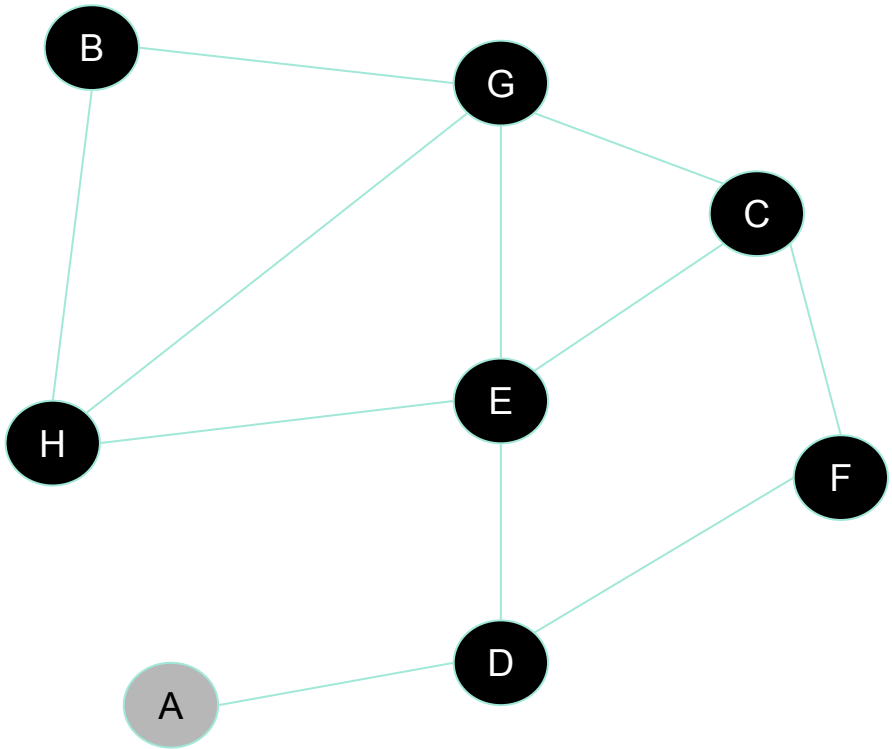


Si un nodo es gris o negro, no lo volvemos a agregar

Cada vez que agregamos un nodo a la cola lo marcamos de gris, cada vez que sacamos un nodo lo marcamos de negro

Q = {F, A}

BFS

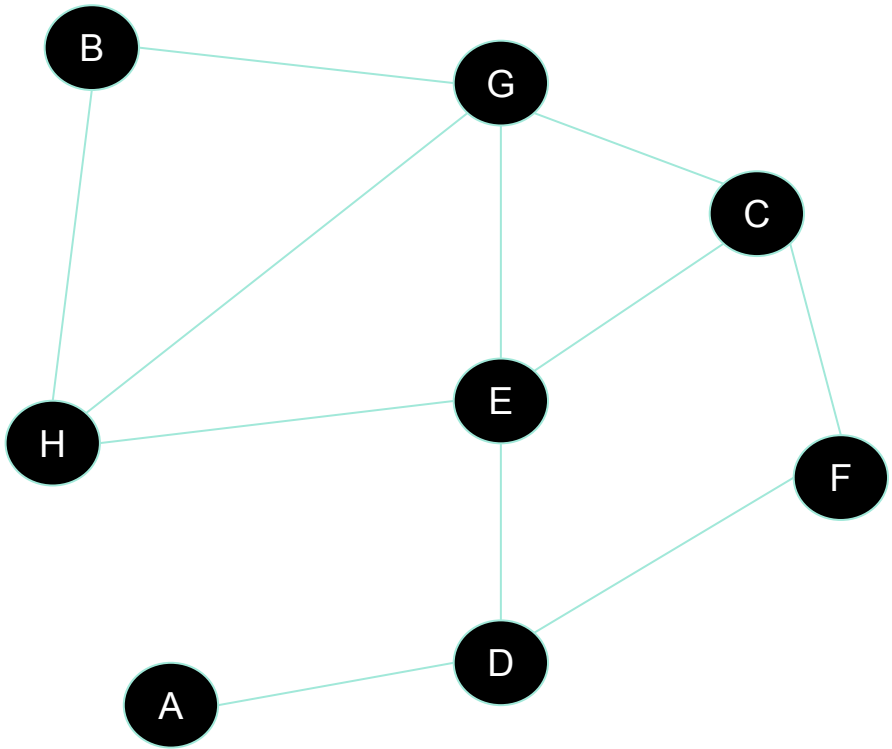


Si un nodo es gris o negro, no lo volvemos a agregar

Cada vez que agregamos un nodo a la cola lo marcamos de gris, cada vez que sacamos un nodo lo marcamos de negro

Q = {A}

BFS



Si un nodo es gris o negro, no lo volvemos a agregar

Cada vez que agregamos un nodo a la cola lo marcamos de gris, cada vez que sacamos un nodo lo marcamos de negro

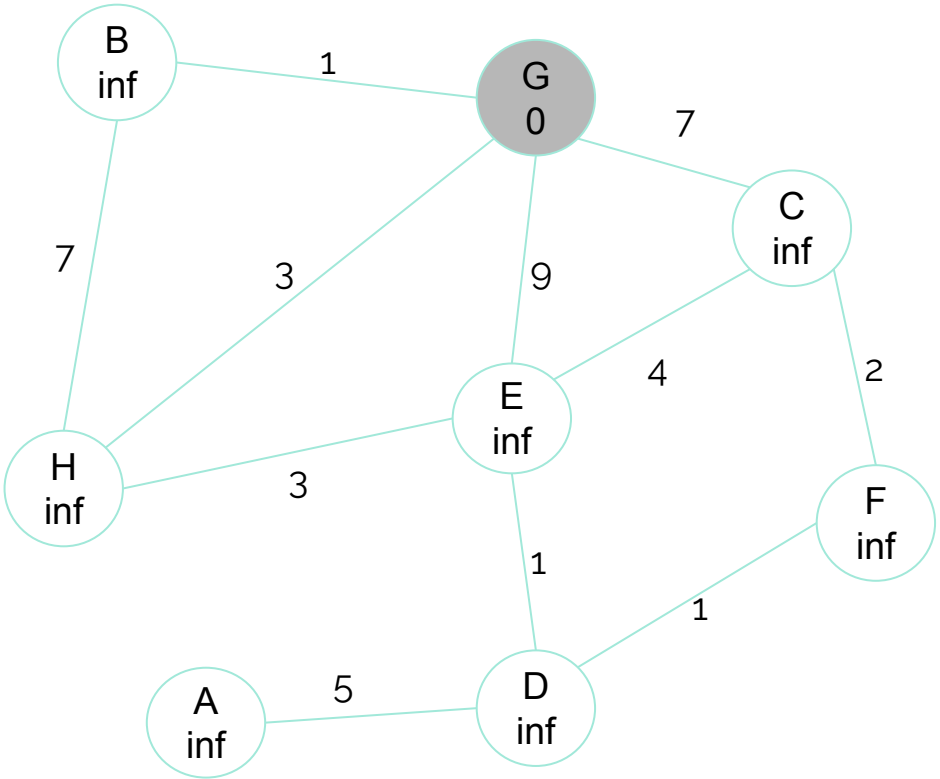
Q = {}

Una vez que la cola está vacía termina el algoritmo

¿Cómo podemos mejorar BFS para encontrar la ruta más corta?

- Cambiamos la cola por una cola de prioridad, que ordene las aristas según peso de menor a mayor, de tal forma que siempre revisemos primero posibles caminos más cortos
- Iniciamos todos los nodos con distancia infinita a nuestro nodo inicial, de tal forma que cada vez que lo visitamos revisamos si encontramos una distancia menor

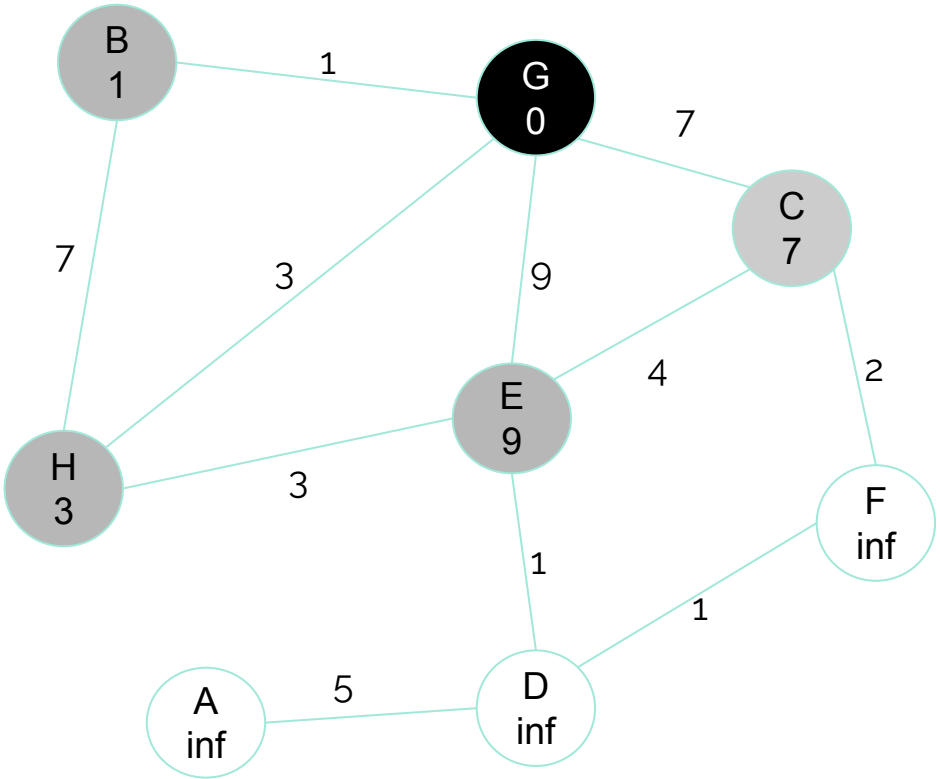
Dijkstra



Partimos desde el nodo G, similar a BFS marcamos los nodos con los mismos códigos de color

PQ = {(G, 0)}

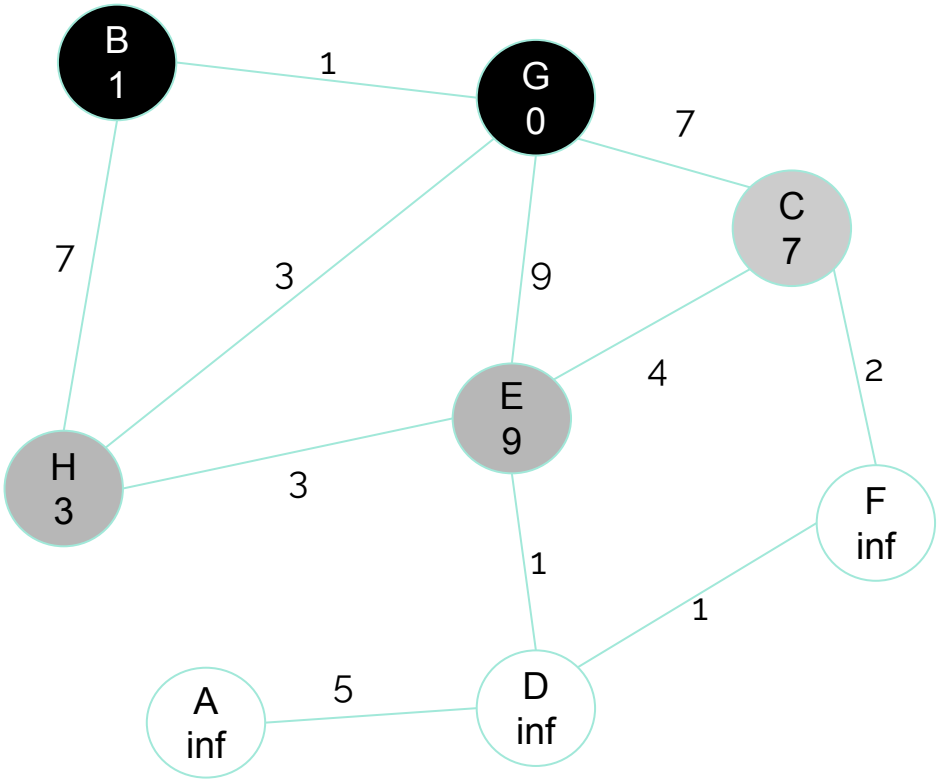
Dijkstra



Para calcular las distancias, sumamos el peso de la arista con la distancia que lleva acumulada el nodo

PQ = {(B, 1), (H, 3), (C, 7), (E, 9)}

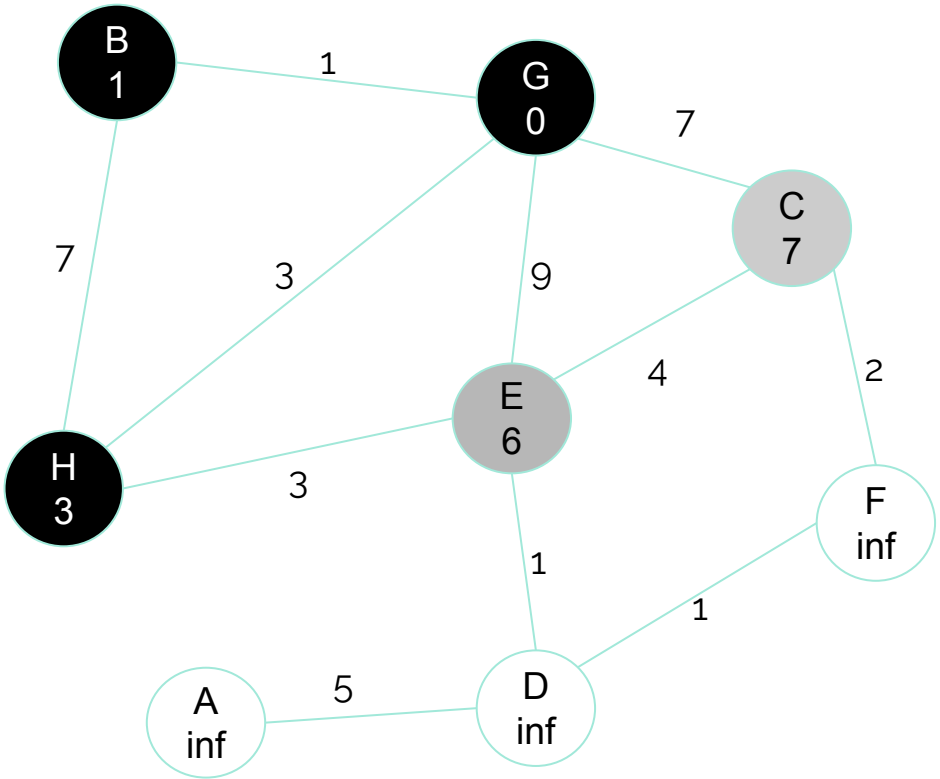
Dijkstra



Para este caso tenemos que la distancia de G a H pasando por B es mayor que la que teníamos

PQ = {(H, 3), (C, 7), (E, 9)}

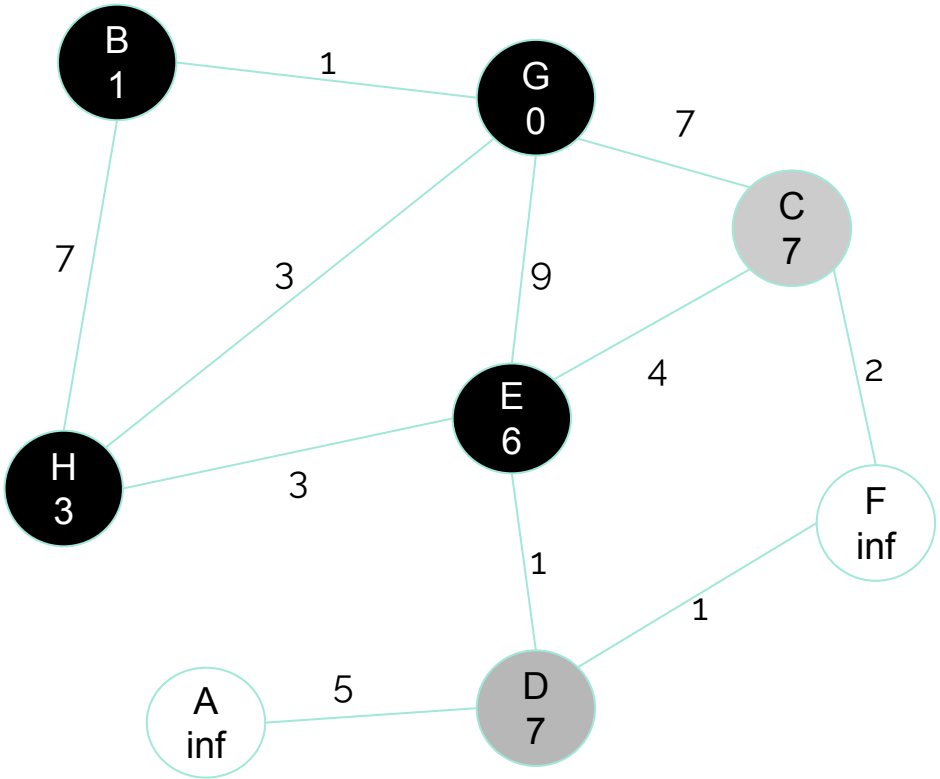
Dijkstra



Para este caso tenemos que la distancia de G a E pasando por H es menor que la que teníamos

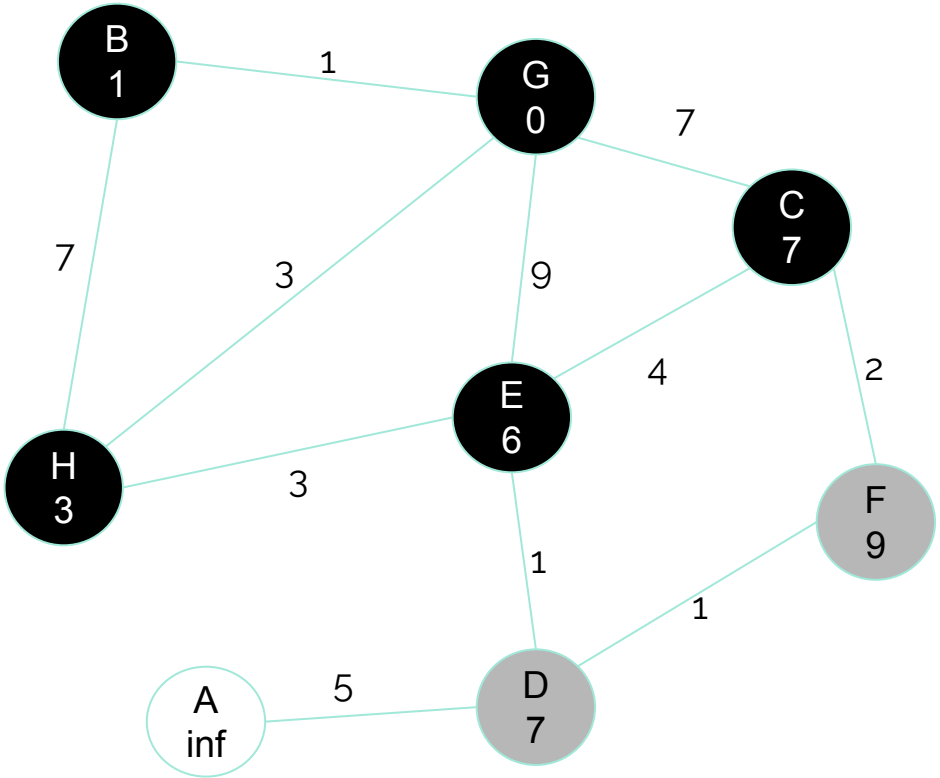
PQ = {(E, 6), (C, 7)}

Dijkstra



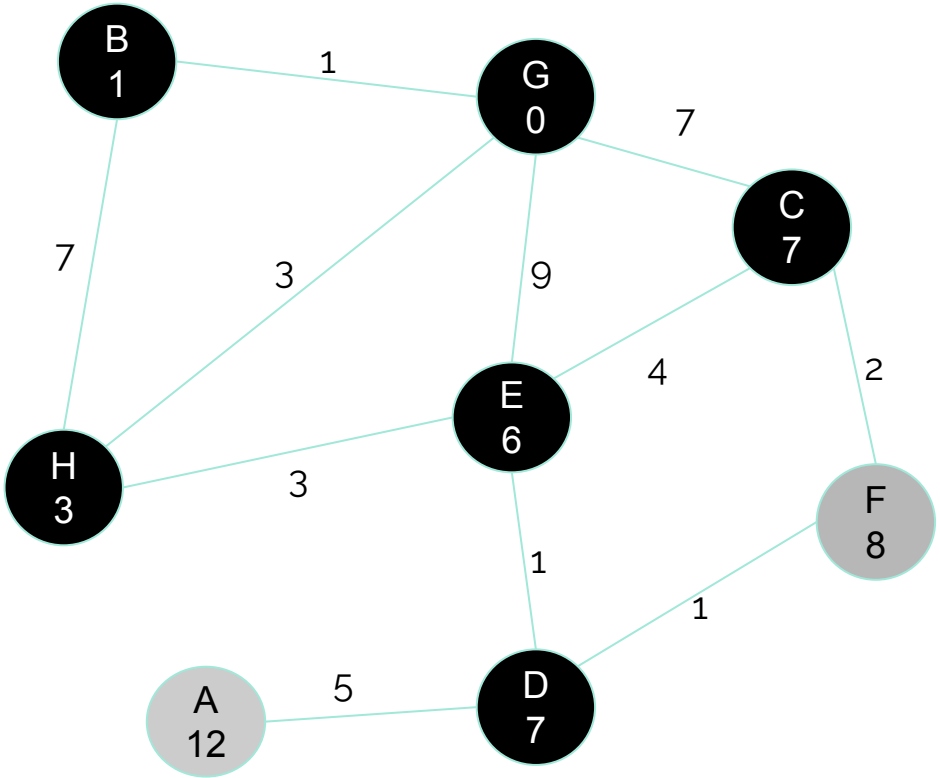
PQ = {(C, 7), (D, 7)}

Dijkstra



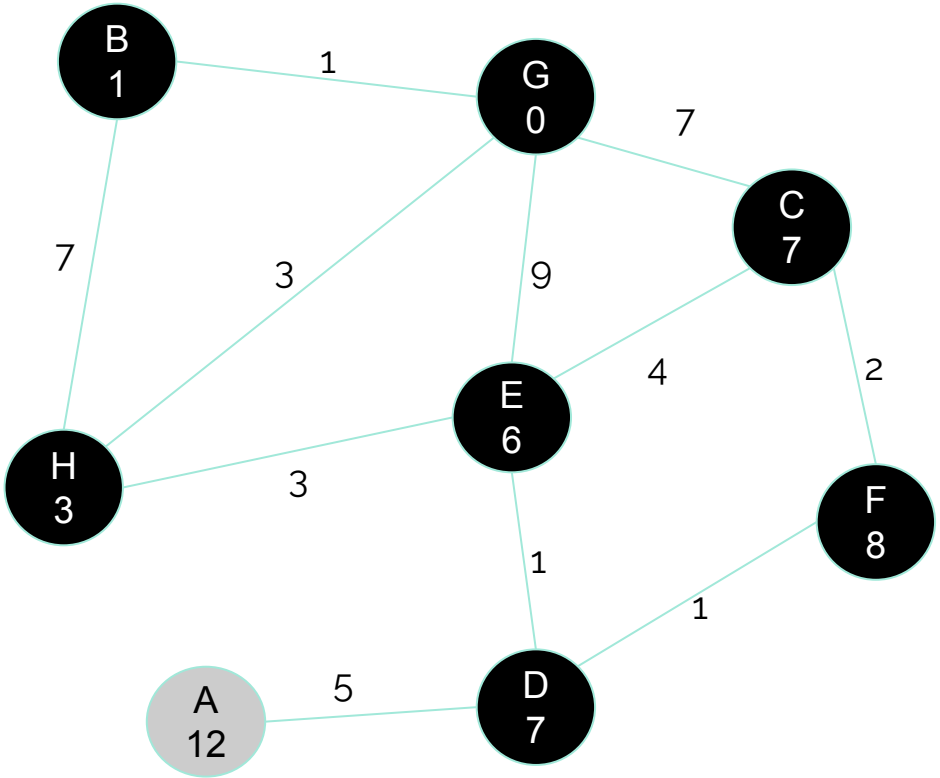
PQ = {(D, 7), (F, 9)}

Dijkstra



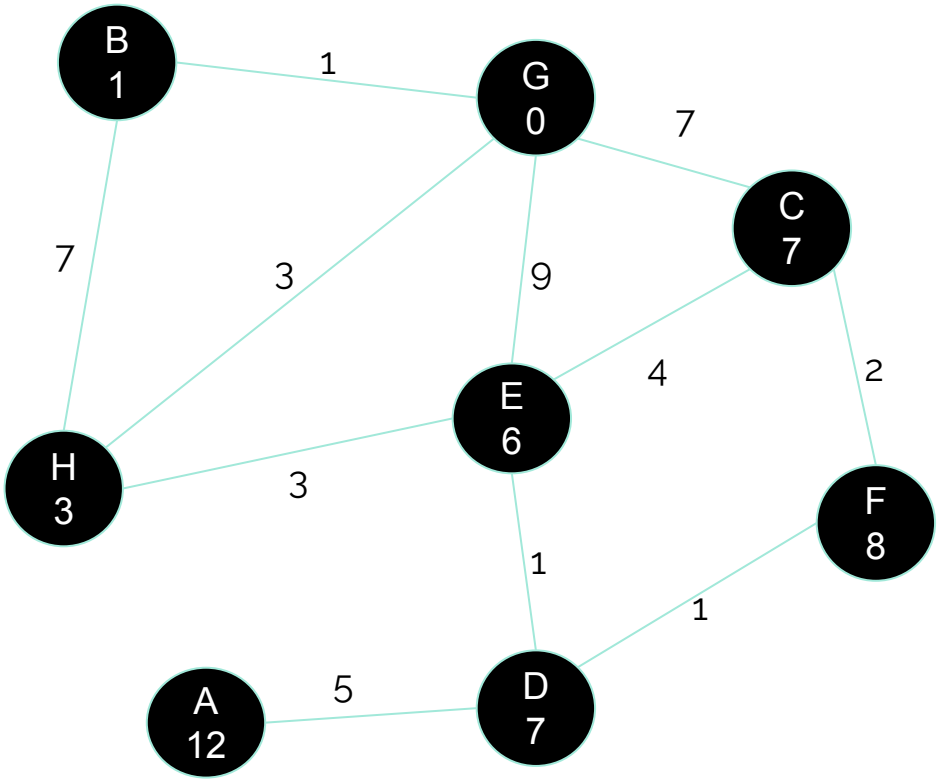
PQ = {(F, 8), (A, 12)}

Dijkstra



PQ = {(A, 12)}

Dijkstra



PQ = {}

I3 2022-2 P4

El diámetro de un árbol no dirigido conexo T se define como el largo del camino más largo en T. Suponga que existe un único camino de largo máximo en T con extremos u y v. Si x es un nodo cualquiera, se puede demostrar que el nodo más lejano a x es u o v. Usando este resultado, proponga un algoritmo que determine el diámetro de un árbol T.