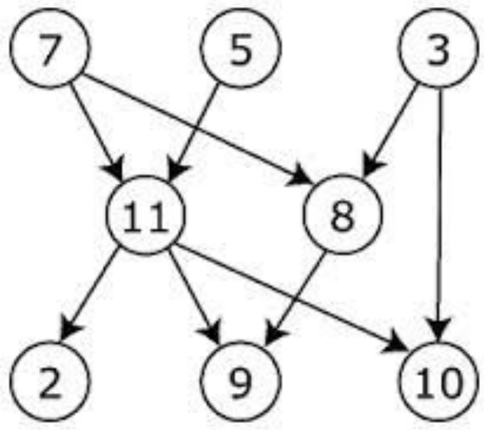
BFS + Dijkstra

Ejercicios de Repaso I3 2024-1

Paula Grune

Ejercicio 1: I3 2022-2

Considere un grafo dirigido G = (V, E) que en lugar de aristas con pesos tiene nodos con pesos. El problema de rutas más cortas desde una fuente se define análogamente, donde el costo de un camino es la suma de los costos de sus nodos. Proponga cómo modificar el grafo para poder utilizar el algoritmo de Dijkstra para resolver dicho problema con la misma complejidad vista en clase.



Podemos modificar el grafo incluyendo pesos en sus aristas de acuerdo al nodo de partida:

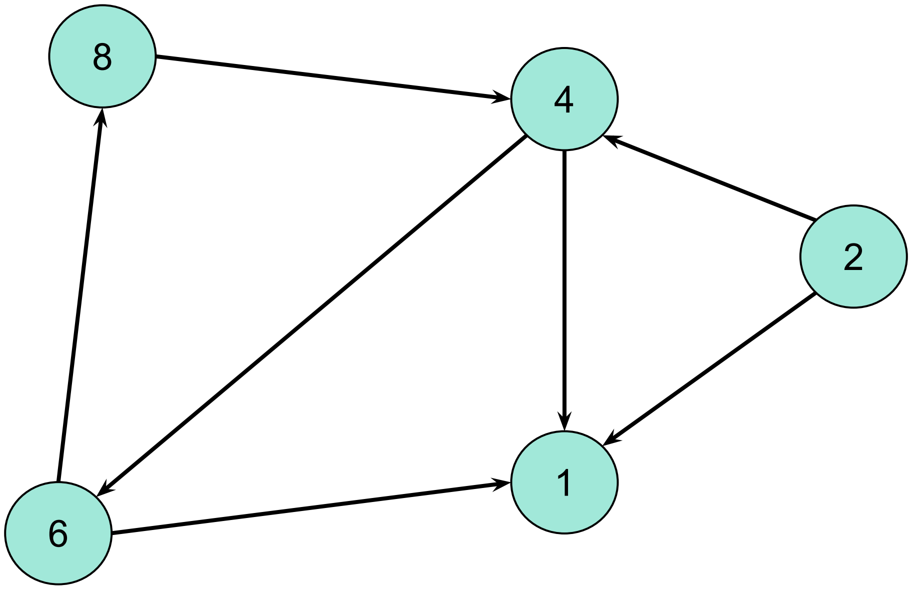
$\texttt{CreateWeights}(V, E)$:

    for $v \in V$ **:**

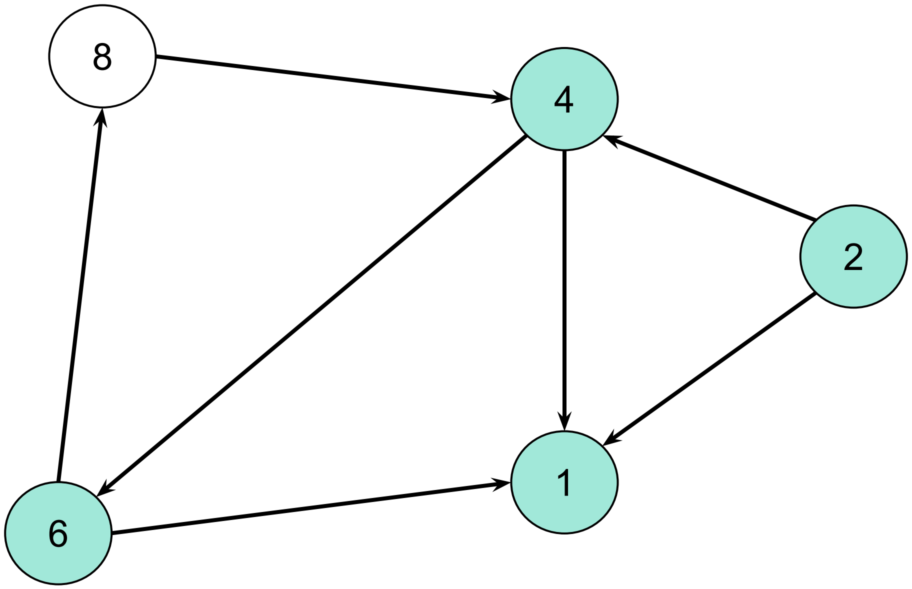
        for $(x, y) \in E$ **:**

            if $v = x$ **:**

                $cost(x, y) \leftarrow cost(v)$




```
CreateWeights(V, E):
    for v ∈ V :
        for (x, y) ∈ E :
            if v = x :
                cost(x, y) ← cost(v)
```



$\texttt{CreateWeights}(V, E)$:

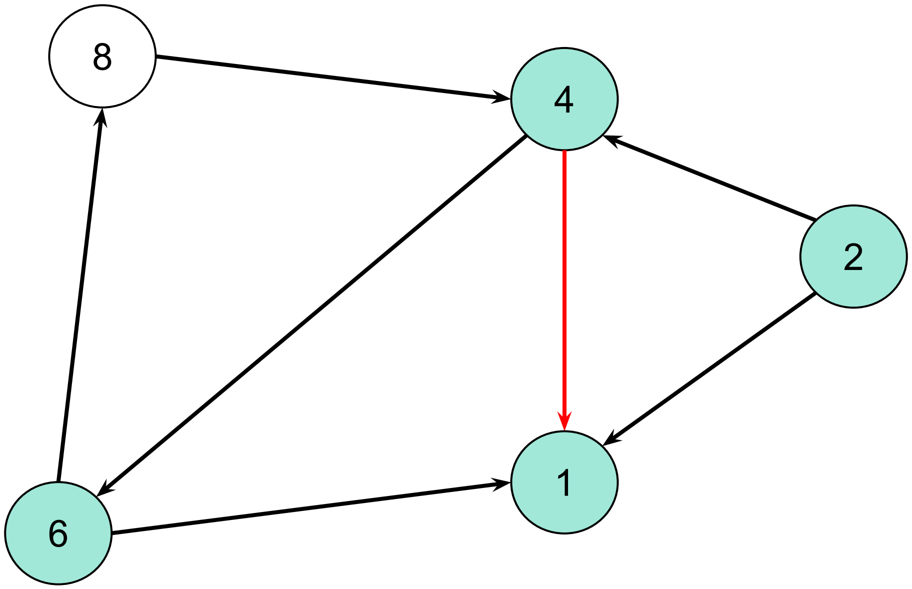
$\quad$ **for** $v \in V$ **:**

$\quad\quad$ **for** $(x, y) \in E$ **:**

$\quad\quad\quad$ **if** $v = x$ **:**

$\quad\quad\quad\quad cost(x, y) \leftarrow cost(v)$

v = 8



$\texttt{CreateWeights}(V, E)$:

  for $v \in V$ **:**

    for $(x, y) \in E$ **:**

      if $v = x$ **:**

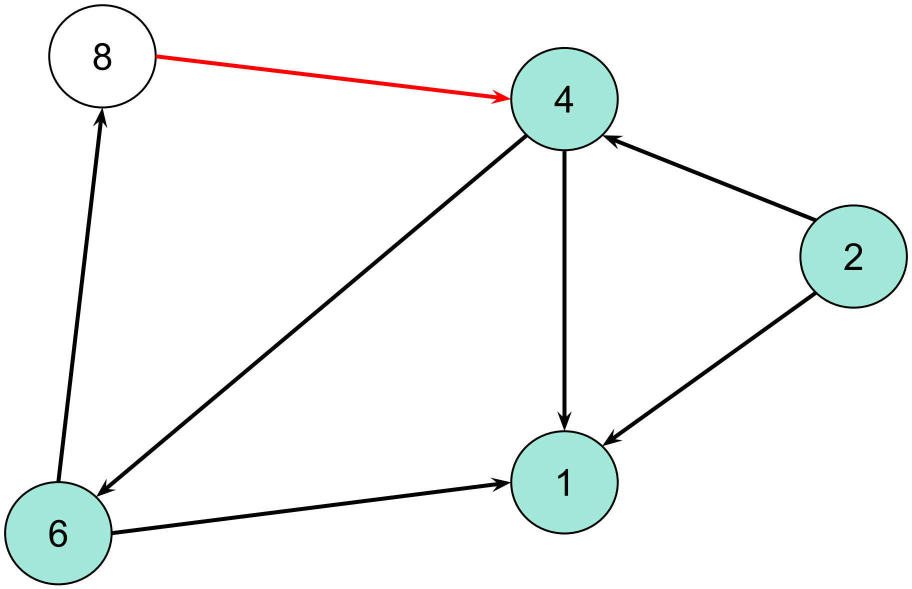
        $cost(x, y) \leftarrow cost(v)$

v = 8

(x, y) = (4, 1)

x = 4

y = 1



`CreateWeights`(V, E):

  for $v \in V$ **:**

    for $(x, y) \in E$ **:**

      if $v = x$ **:**

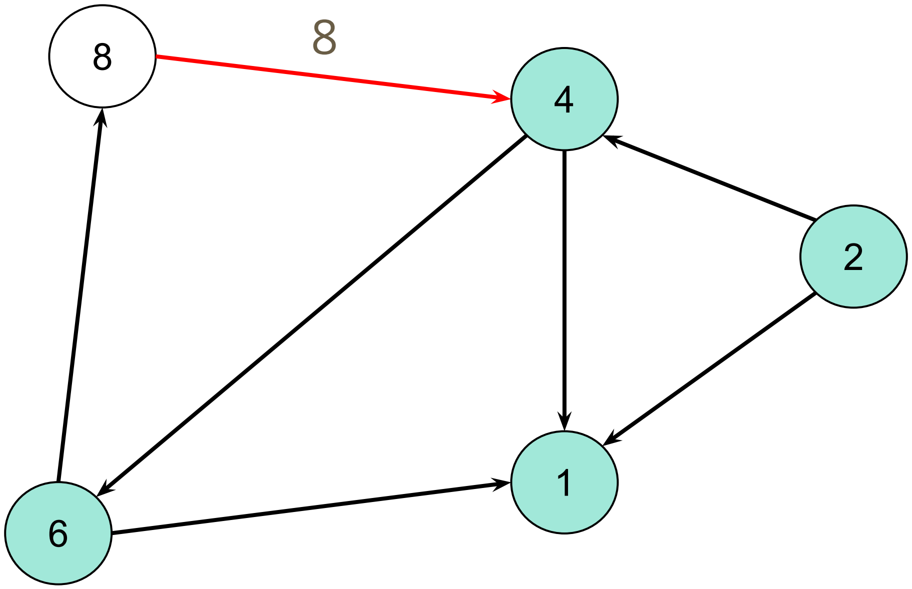
        $cost(x, y) \leftarrow cost(v)$

v = 8

(x, y) = (8, 4)

x = 8

y = 4



$\texttt{CreateWeights}(V, E)$:

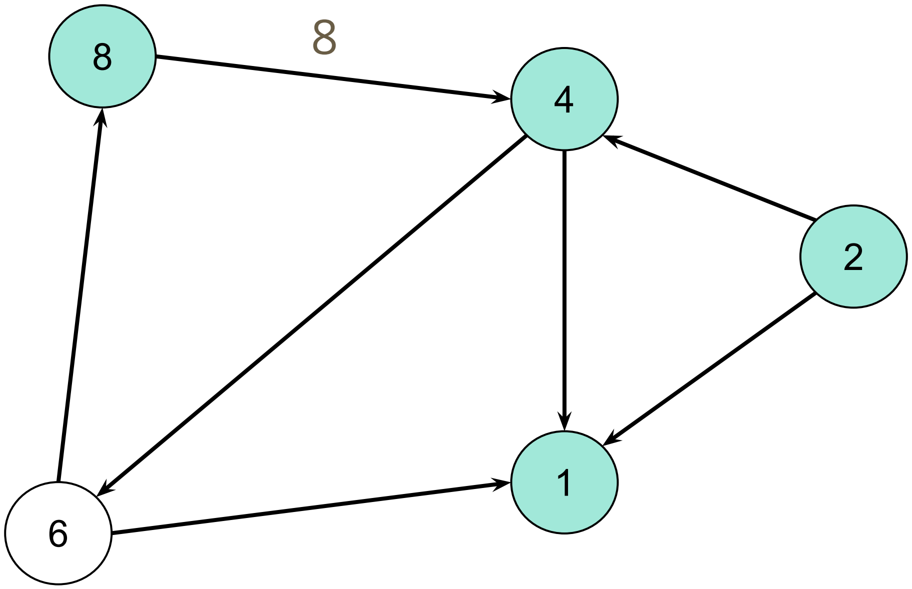
$$
\begin{array}{l}
\textbf{for } v \in V \textbf{ :} \\
\quad \textbf{for } (x, y) \in E \textbf{ :} \\
\quad\quad \textbf{if } v = x \textbf{ :} \\
\quad\quad\quad cost(x, y) \leftarrow cost(v)
\end{array}
$$

$v = 8$

$(x, y) = (8, 4)$

$x = 8$

$y = 4$



$\texttt{CreateWeights}(V, E)$:

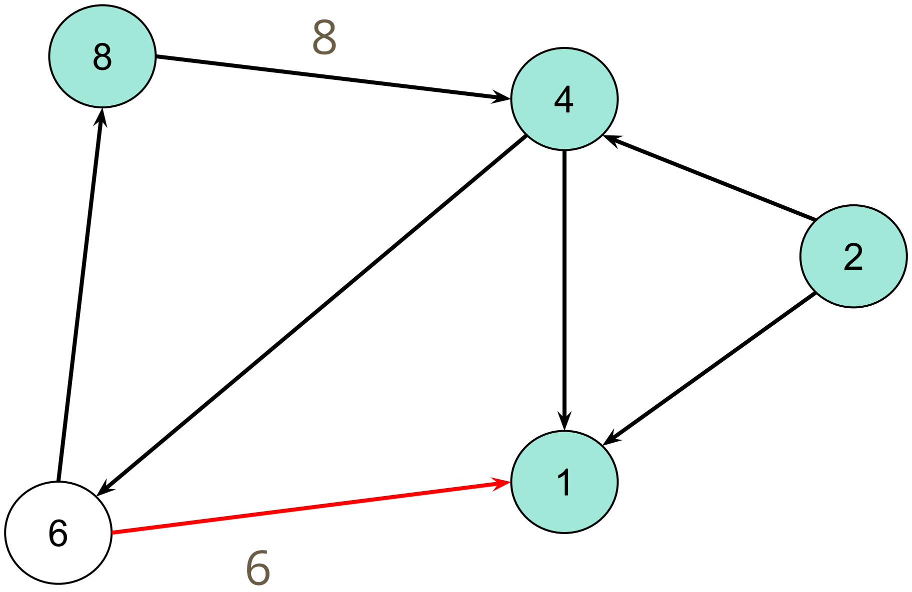
for $v \in V$ **:**

  for $(x, y) \in E$ **:**

    if $v = x$ **:**

      $cost(x, y) \leftarrow cost(v)$

v = 6



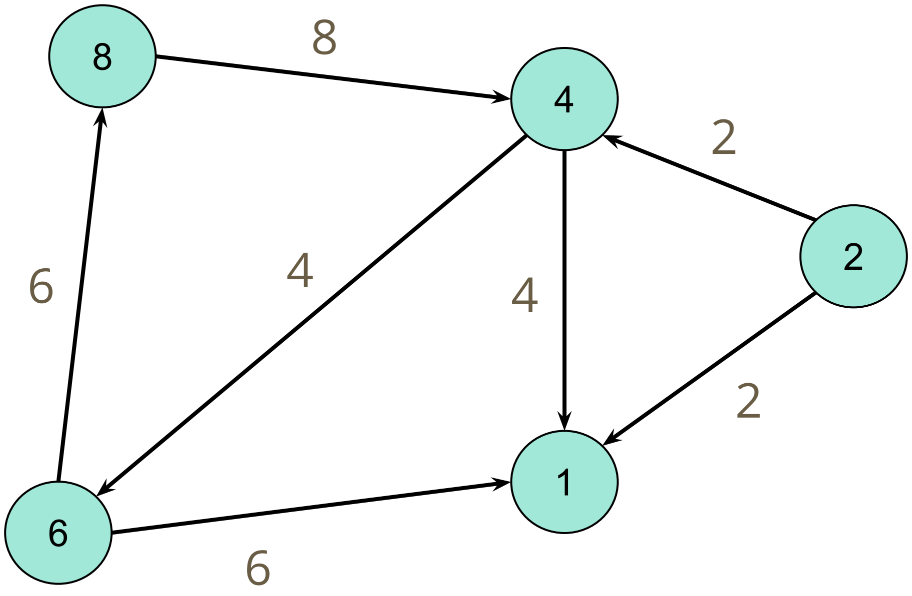

```
CreateWeights(V, E):
    for v ∈ V :
        for (x, y) ∈ E :
            if v = x :
                cost(x, y) ← cost(v)
```

$v = 6$

$(x, y) = (6, 1)$

$x = 6$

$y = 1$



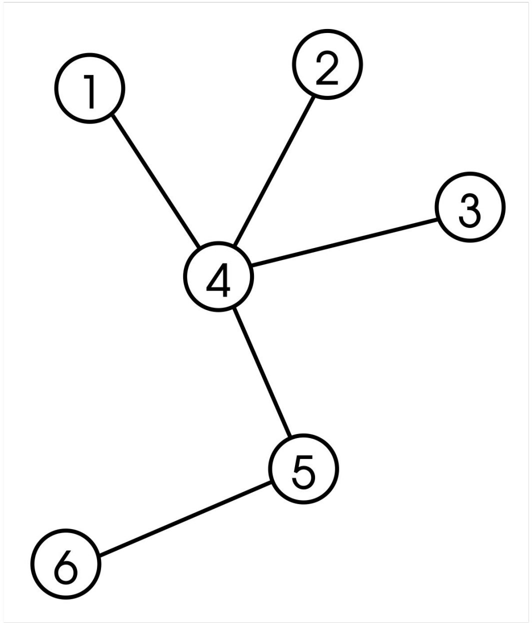

```
CreateWeights(V, E):
    for v ∈ V :
        for (x, y) ∈ E :
            if v = x :
                cost(x, y) ← cost(v)
```

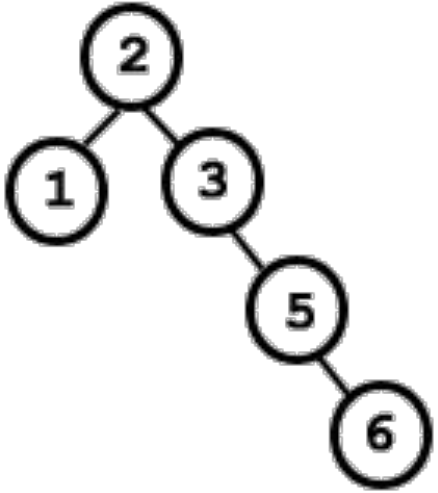
Luego de ejecutar este algoritmo, las aristas de salida de los nodos tienen como costo el de su nodo inicial. Luego, usamos Dijkstra como siempre para encontrar rutas más cortas

Ejercicio 2: I3 2022-2

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.

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.





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.

$\texttt{Diameter}(V, E)$:

$x \leftarrow$ cualquier nodo de V

$d \leftarrow \texttt{BFS}(x)$

$u \leftarrow$ nodo que tiene máximo $d[\cdot]$

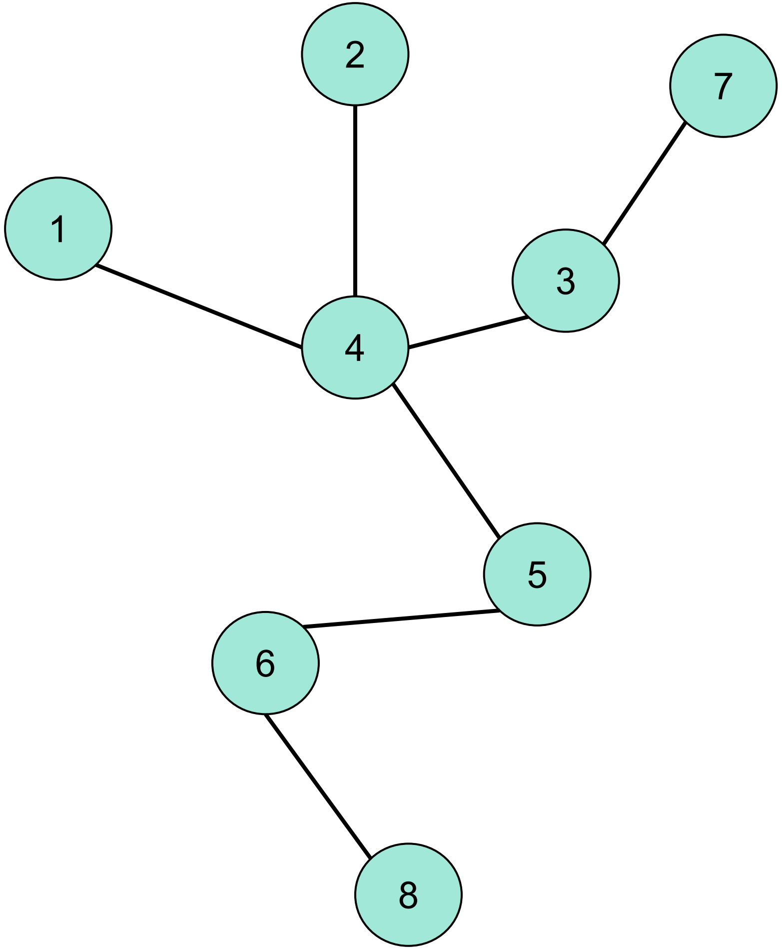
$d \leftarrow \texttt{BFS}(u)$

$D \leftarrow \max\{d[v] \mid v \in V\}$

return D

Donde se usa una versión modificada de BFS para que retorne el **arreglo con distancias desde la fuente**, así como el nodo asociado a cada distancia.

d = [(v_1, 3), (v_2, 9), (v_3, 16), ...]



$\texttt{Diameter}(V, E)$:

$x \leftarrow$ cualquier nodo de V

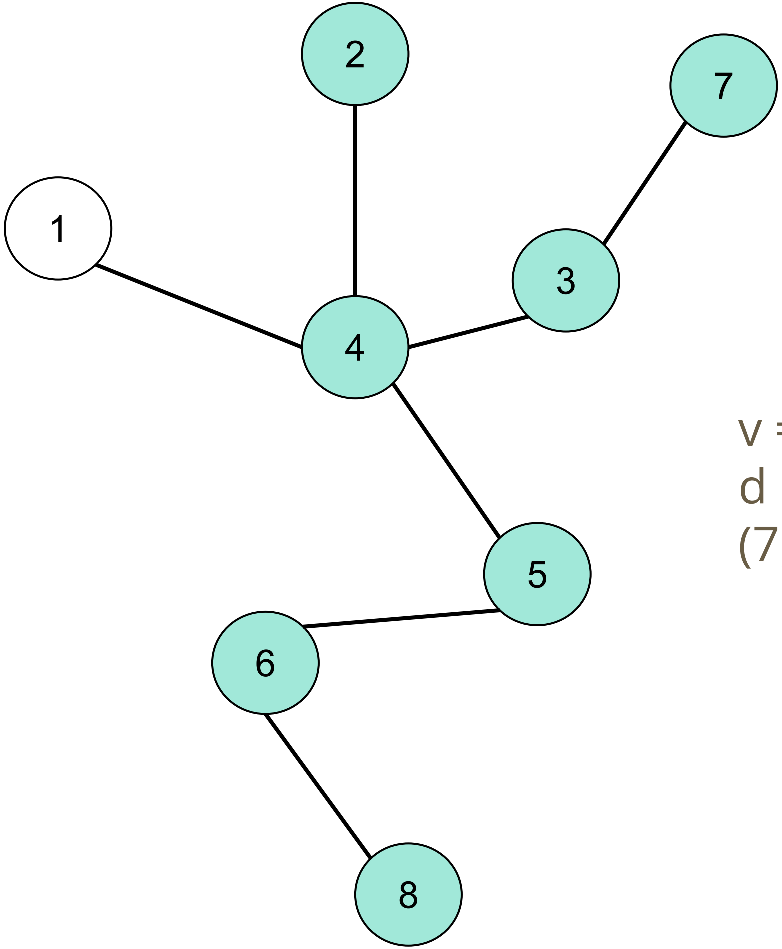
$d \leftarrow \texttt{BFS}(x)$

$u \leftarrow$ nodo que tiene máximo $d[\cdot]$

$d \leftarrow \texttt{BFS}(u)$

$D \leftarrow \text{máx}\{d[v] \mid v \in V\}$

return D



$\texttt{Diameter}(V, E)$:

$x \leftarrow$ cualquier nodo de V

$d \leftarrow \texttt{BFS}(x)$

$u \leftarrow$ nodo que tiene máximo $d[\cdot]$

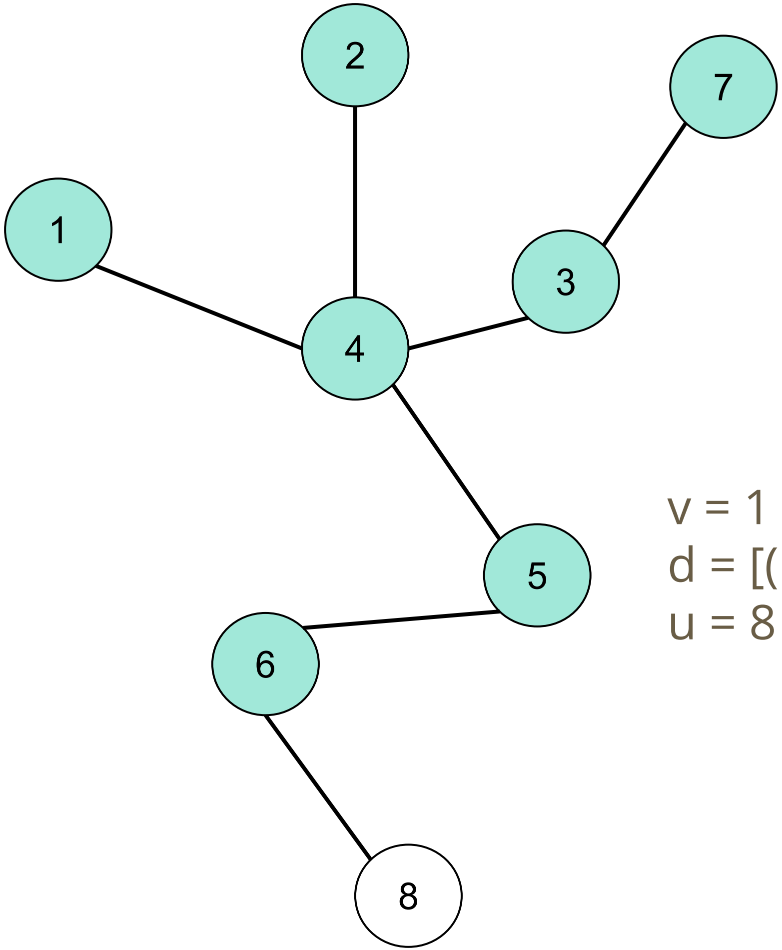
$d \leftarrow \texttt{BFS}(u)$

$D \leftarrow \text{máx}\{d[v] \mid v \in V\}$

return D

v = 1

d = [(1, 0), (2, 2), (3, 2), (4, 1), (5, 2), (6, 3), (7, 3), (8, 4)]



Diameter(V, E):

$x \leftarrow$ cualquier nodo de V

$d \leftarrow \texttt{BFS}(x)$

$u \leftarrow$ nodo que tiene máximo $d[\cdot]$

$d \leftarrow \texttt{BFS}(u)$

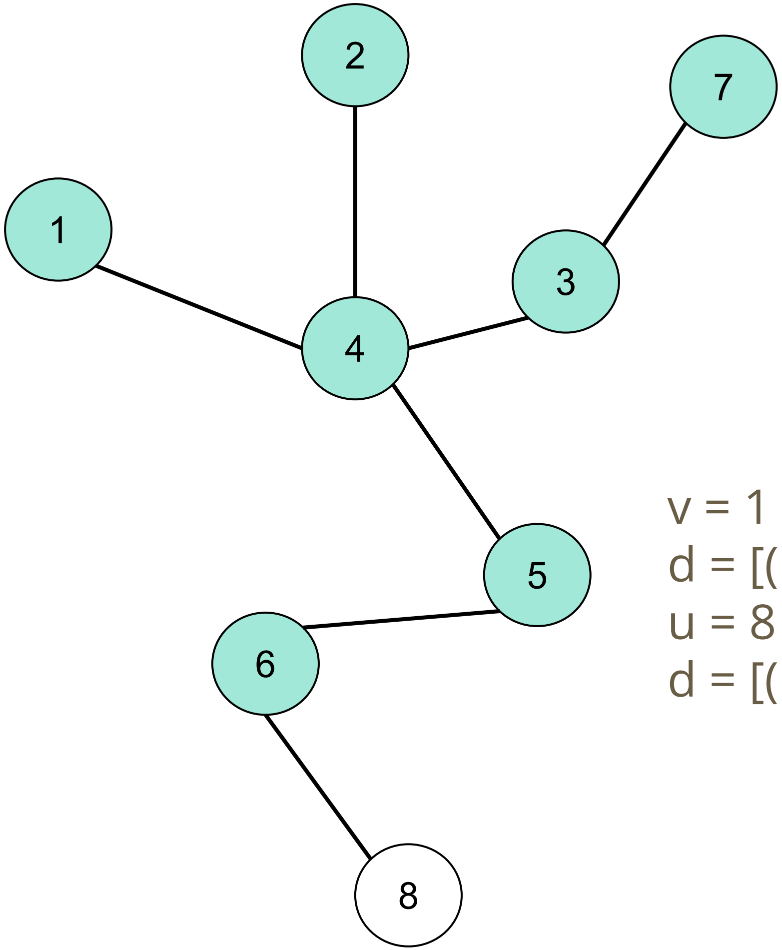
$D \leftarrow \max\{d[v] \mid v \in V\}$

return D

v = 1

d = [(1, 0), (2, 2), (3, 2), (4, 1), (5, 2), (6, 3), (7, 3), **(8, 4)**]

u = 8



$\texttt{Diameter}(V, E)$:

$x \leftarrow$ cualquier nodo de V

$d \leftarrow \texttt{BFS}(x)$

$u \leftarrow$ nodo que tiene máximo $d[\cdot]$

$d \leftarrow \texttt{BFS}(u)$

$D \leftarrow \max\{d[v] \mid v \in V\}$

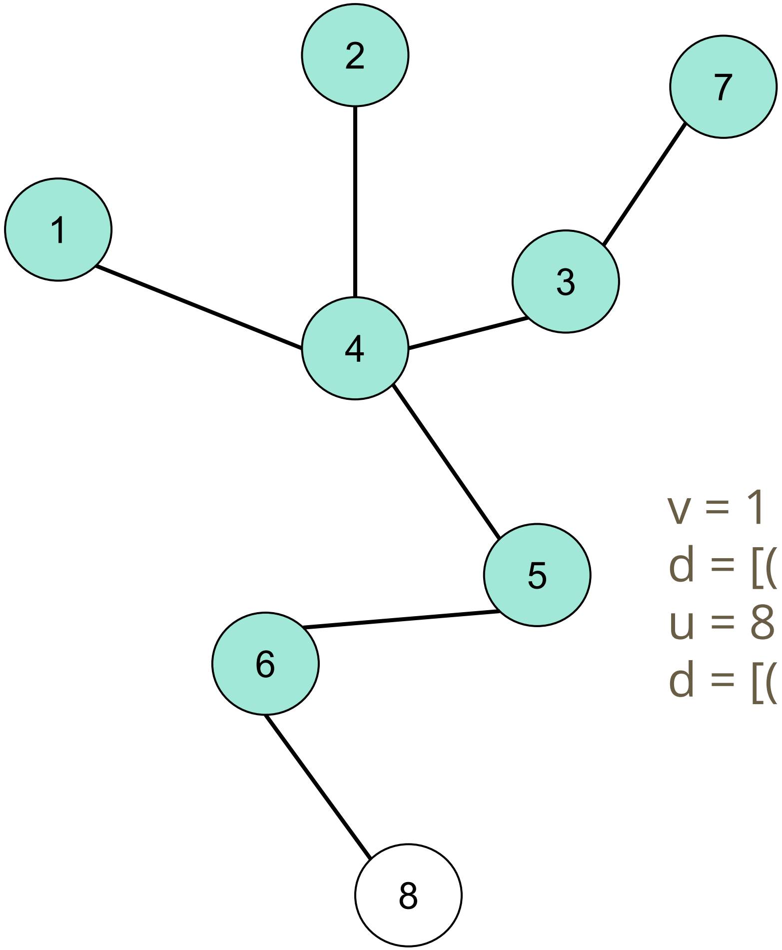
return D

v = 1

d = [(1, 0), (2, 2), (3, 2), (4, 1), (5, 2), (6, 3), (7, 3), (8, 4)]

u = 8

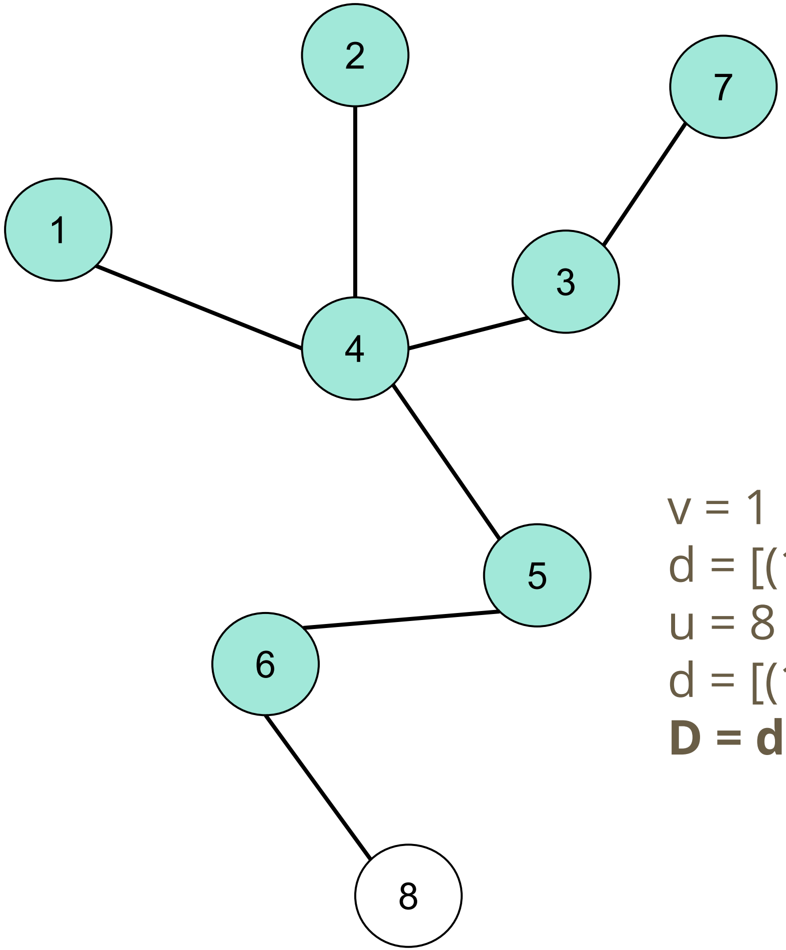
d = [(1, 4), (2, 4), (3, 4), (4, 3), (5, 2), (6, 1), (7, 5), (8, 0)]



$\texttt{Diameter}(V, E)$:

$x \leftarrow$ cualquier nodo de V
$d \leftarrow \texttt{BFS}(x)$
$u \leftarrow$ nodo que tiene máximo $d[\cdot]$
$d \leftarrow \texttt{BFS}(u)$
$D \leftarrow \text{máx}\{d[v] \mid v \in V\}$
return D

v = 1
d = [(1, 0), (2, 2), (3, 2), (4, 1), (5, 2), (6, 3), (7, 3), (8, 4)]
u = 8
d = [(1, 4), (2, 4), (3, 4), (4, 3), (5, 2), (6, 1), **(7, 5)**, (8, 0)]



$\texttt{Diameter}(V, E)$:

$x \leftarrow$ cualquier nodo de V

$d \leftarrow \texttt{BFS}(x)$

$u \leftarrow$ nodo que tiene máximo $d[\cdot]$

$d \leftarrow \texttt{BFS}(u)$

$D \leftarrow \text{máx}\{d[v] \mid v \in V\}$

return D

v = 1

d = [(1, 0), (2, 2), (3, 2), (4, 1), (5, 2), (6, 3), (7, 3), (8, 4)]

u = 8

d = [(1, 4), (2, 4), (3, 4), (4, 3), (5, 2), (6, 1), (7, 5), (8, 0)]

D = d[v = 7] = 5