



# 

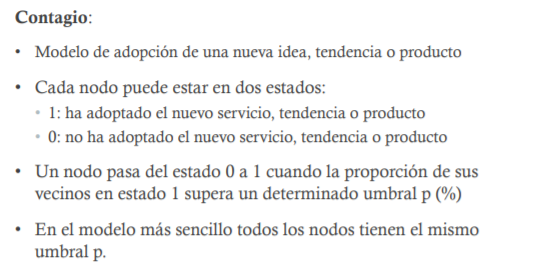
**Programación del Algoritmo de propagación por contagio en Python**

**Conceptos Avanzados de Internet**

*César Francisco San Nicolás Martínez*

*Jorge San Emeterio Villalaín*

1. **El algoritmo**



1. **Formulación del problema**

Tenemos los siguientes datos de entrada del problema:

* N nodos.
* A = matriz de adyacencia de nodos (N x N):
  + A(i,j) = 1 si Nodo i y Nodo j están conectados.
  + A(i,j) = 0 si Nodo i y Nodo j están conectados.
  + Autoenlaces no contemplados, por lo que A(i,j) = 0 si i = j.
* Dos nodos contagiados inicialmente.
* Umbral de contagio p.

El dato de salida del problema será un vector de N posiciones que tendrá dos posibles valores: 0 ó 1.

* La posición n de ese vector valdrá 1 si el Nodo n está contagiado\* al final del algoritmo.
* La posición n de ese vector valdrá 0 si el Nodo n no está contagiado\* al final del algoritmo.

\*\*Estar contagiado = adoptar el nuevo servicio, tendencia o producto.

1. **Ejecución del algoritmo**

En primer lugar, se puede ver una explicación del algoritmo programado en pseudocódigo:

While true

Foreach Nodo n

If Nodos\_contagiados.contains(Nodo n)

Continue

Nodos\_ = nodos conectados al Nodo n

Nodos\_contagiados = nodos conectados al Nodo n y contagiados

Umbral\_n = Nodos\_contagiados / Nodos\_

If p >= Umbral\_n

Nuevos\_nodos\_contagiados.add(Nodo n)

end

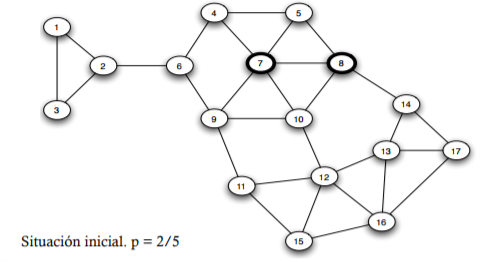
If Nuevos\_nodos\_contagiados.length = 0

Break

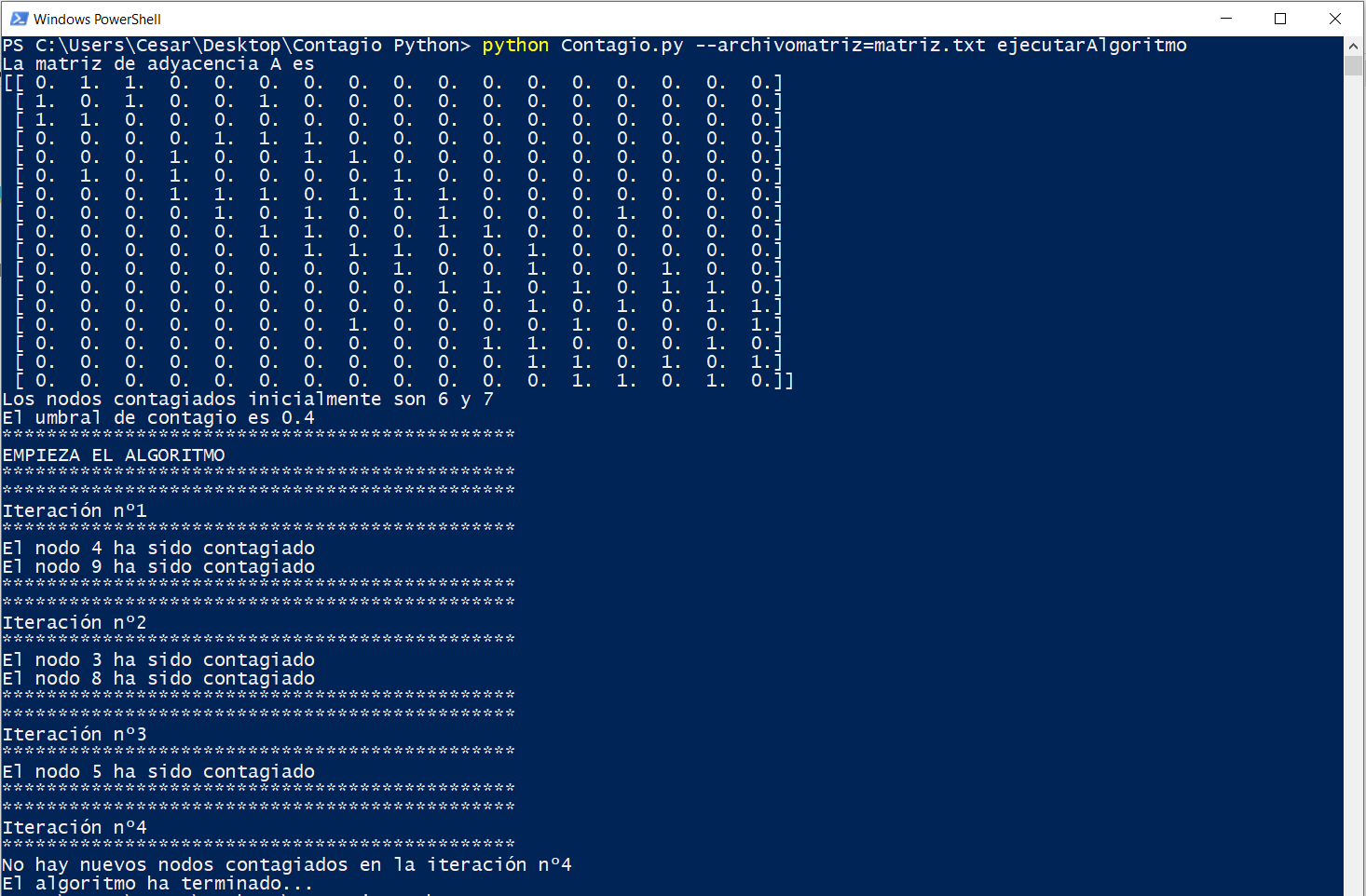
Nodos\_contagiados = nodos\_contagiados U Nuevos\_nodos\_contagiados

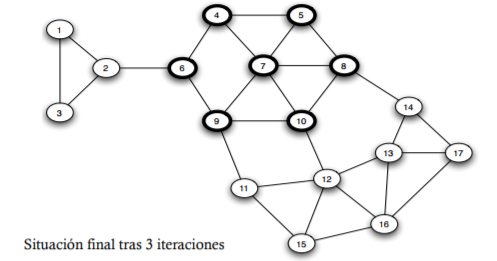
end

Para comprobar la correcta implementación, hemos utilizado el ejemplo visto en teoría:



A continuación, podemos ver una imagen en la que se ve la ejecución del algoritmo con los parámetros de entrada del caso de teoría:





Vemos que el resultado del algoritmo programado y del algoritmo de teoría coinciden en su resultado. Tras 3 iteraciones, hay 7 nodos contagiados.

\*\* En teoría, los índices van de 1 a 17, mientras que en el algoritmo programado van de 0 a 16.

1. **Bibliografía y recursos**

* Teoría de la asignatura Conceptos Avanzados de Internet.
* <https://www.pybonacci.org>
* Paquetes Numpy y Click de Python.

**ANEXO. Código del algoritmo en Python**

import click  
import numpy as np  
  
@click.group()  
@click.option('--archivomatriz',  
 prompt=True,  
 help='Ruta del archivo donde se encuentra definida la matriz de adyacencia')  
@click.pass\_context  
def cli(ctx,archivomatriz):  
 ctx.obj = archivomatriz  
  
@cli.command(name='ejecutarAlgoritmo')  
@click.option('--semillauno',  
 default=6,  
 help='Primer nodo contagiado inicialmente')  
@click.option('--semillados',  
 default=7,  
 help='Segundo nodo contagiado inicialmente')  
@click.option('--umbral',  
 default=0.4,  
 help='Umbral para establecer contagio')  
@click.pass\_context  
def ejec\_algoritmo(ctx,semillauno,semillados,umbral):  
  
 A = extraer\_matriz(ctx.obj)  
 nodos\_contagiados = algoritmo(A,semillauno,semillados,umbral)  
  
  
def algoritmo(A,semillauno,semillados,umbral):  
 numero\_nodos = len(A)  
 print('La matriz de adyacencia A es ')  
 print(A)  
 print('Los nodos contagiados inicialmente son ' + str(semillauno) + ' y ' + str(semillados))  
 print('El umbral de contagio es ' + str(umbral))  
  
 nodos\_contagiados = np.zeros((1, numero\_nodos))  
 nodos\_contagiados[0][semillauno] = 1  
 nodos\_contagiados[0][semillados] = 1  
  
 print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')  
 print('EMPIEZA EL ALGORITMO')  
  
 iter = 1  
 while (True):  
 nuevos\_nodos\_contagiados = []  
 print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')  
 print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')  
 print('Iteración nº' + str(iter))  
 print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')  
 for nodo in range(numero\_nodos):  
 if (nodos\_contagiados[0][nodo] == 1):  
 continue  
 fila = A[nodo, :]  
 nodos\_totales\_nodo = sum(fila)  
 nodos\_contagiados\_nodo = sum(np.multiply(fila, nodos\_contagiados[0, :]))  
 umbral\_nodo = nodos\_contagiados\_nodo / nodos\_totales\_nodo  
 if (umbral\_nodo >= umbral):  
 nuevos\_nodos\_contagiados.append(nodo)  
 print('El nodo ' + str(nodo) + ' ha sido contagiado')  
  
 if (len(nuevos\_nodos\_contagiados) == 0):  
 print('No hay nuevos nodos contagiados en la iteración nº' + str(iter))  
 print('El algoritmo ha terminado...')  
 print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*')  
 break  
  
 iter = iter + 1  
 nodos\_contagiados = actualizar\_nodos\_contagiados(nodos\_contagiados, nuevos\_nodos\_contagiados)  
  
 return nodos\_contagiados  
  
def extraer\_matriz(path):  
  
 file = open(path,'r')  
 filas = file.readlines()  
 num\_nodos = len(filas)  
 matriz = np.zeros((num\_nodos, num\_nodos))  
 cont\_fila = 0  
 for fila in filas:  
 elementos = fila.split()  
 cont\_col = 0  
 for elem in elementos:  
 matriz[cont\_fila][cont\_col] = elem  
 cont\_col = cont\_col + 1  
 cont\_fila = cont\_fila + 1  
  
 return matriz  
  
def actualizar\_nodos\_contagiados(nodos\_contagiados\_actuales, nuevos\_nodos\_contagiados):  
  
 nodos\_contagiados = nodos\_contagiados\_actuales  
  
 for index in nuevos\_nodos\_contagiados:  
 nodos\_contagiados[0][index] = 1  
  
 return nodos\_contagiados  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 cli()