Árbol generador de menor costo

Invitado

October 3, 2019

1 Problema

Supongamos que a cada arista de la gráfica completa K_n se le asigna un valor ("peso"). Si a cada subgráfica le asignamos un peso igual a la suma de los pesos de sus aristas, consideraremos el problema de encontrar el árbol generador de menor peso.

2 Algoritmo de Kruskal

El algoritmo de Kruskal consiste en escoger sucesivamente la arista más barata de forma que no forme ciclo con la gráfica determinada por las aristas escogidas previamente; si en algún momento existen varias aristas que tienen el menor costo y no forman ciclos podemos escoger cualquiera.

3 Implementación en Python

Primeramente vamos a importar las bibliotecas que vamos a utilizar.

```
import networkx as nx
import matplotlib.pyplot as plt
from random import random as random
from scipy.spatial.distance import euclidean
```

A continuación definiremos una gráfica aleatoria con 10 vértices.

```
g=nx.gnp_random_graph(10,0.2)
```

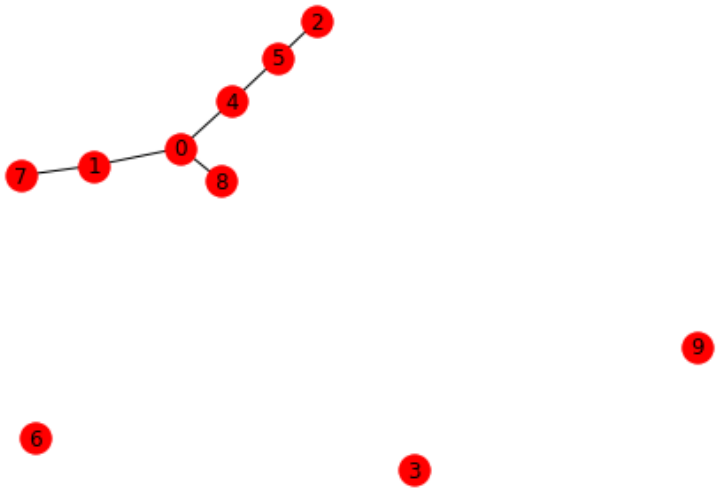
Veremos si nuestra gráfica aleatoria es un bosque.

```
nx.is_forest(g), nx.is_connected(g)
```

```
(True, False)
```

A continuación dibujaremos esta gráfica

```
nx.draw(g, with_labels=True)
```



Calcularemos las componentes conexas de esta gráfica:

```
list(nx.connected_components(g))
```

```
[{0, 1, 2, 4, 5, 7, 8}, {3}, {6}, {9}]
```

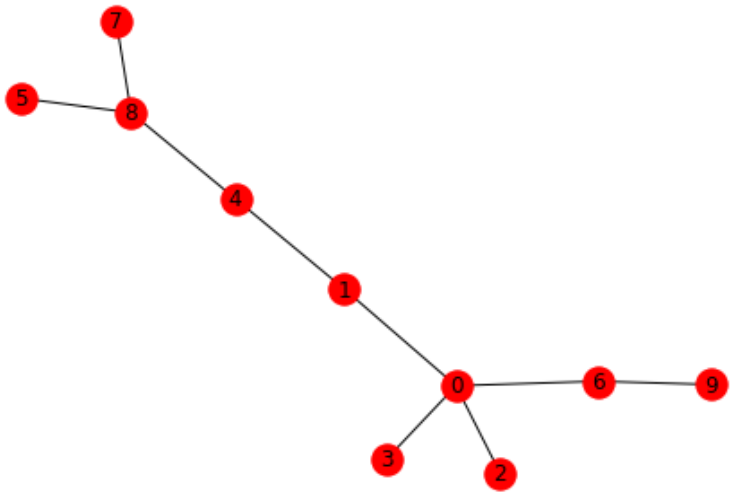
Veamos la componente que contiene al vértice 2.

```
nx.node_connected_component(g,2)
```

```
{0, 1, 2, 4, 5, 7, 8}
```

A continuación dibujaremos un árbol escogido aleatoriamente.

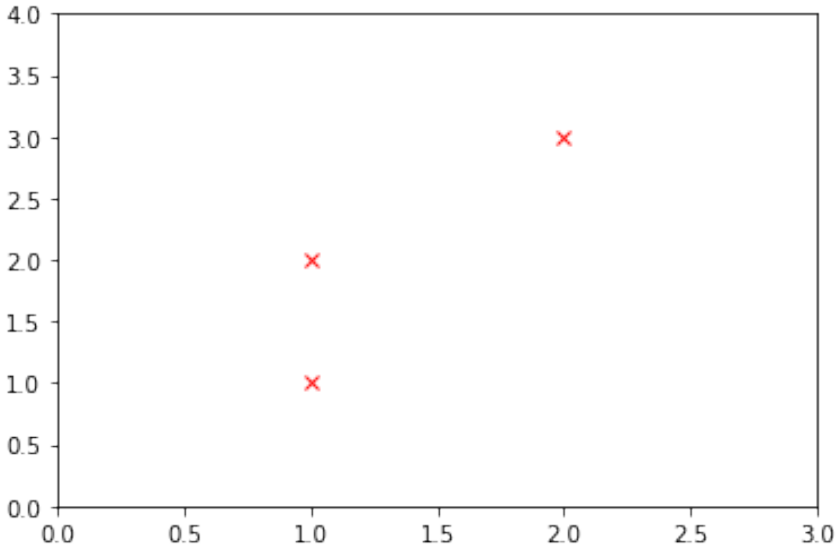
```
t=nx.random_tree(10)
nx.draw(t, with_labels=True)
```



4 Puntos en el plano

Si tenemos dos listas de números de tamaño n, podemos dibujar n puntos en el plano, las coordenadas x de la primera lista y las coordenadas y de la segunda.

```
plt.plot([1,1,2],[1,2,3],'rx')
plt.axis([0,3,0,4])
plt.show()
```

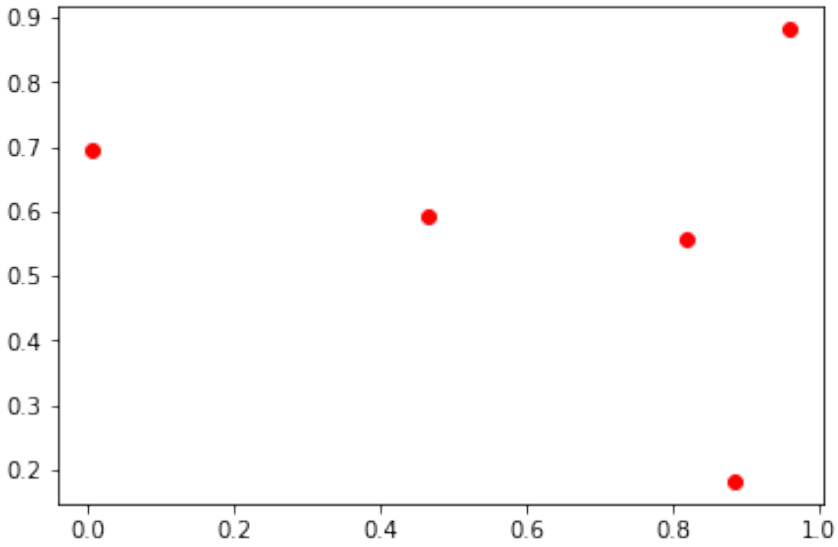


Vamos a definir una función que dibuje n puntos en e plano aleatoriamente.

```
def puntos_en_el_plano(n):
   listax=[]
   listay=[]
   for i in range(n):
       listax.append(random())
       listay.append(random())
   return listax, listay
```

```
puntos=puntos_en_el_plano(50)
```

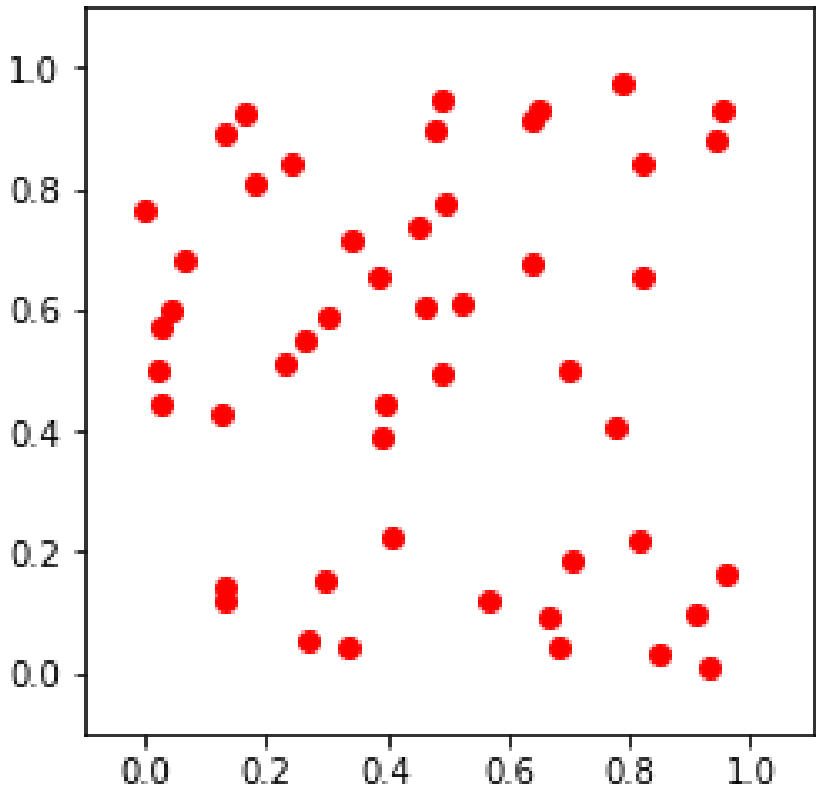
```
plt.plot(puntos[0], puntos[1], 'ro')
plt.show()
```



Hagamos una función tal que a partir de dos listas, produzca el dibujo:

```
def dibujo_puntos(listax, listay):
    plt.plot(listax, listay, 'ro')
    plt.axis([-0.1,1.1,-0.1,1.1])
    plt.gca().set_aspect('equal')
    plt.show()
```

```
dibujo_puntos(*puntos)
```


1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0