```
deg = []

for i in 0, N-1:

    grad = int(extrac_random_num() ) + 1         % Extraer los
grados de numeros aleatorios de la distribucion input

    deg.append(grad)                              % Sumamos 1 (o
mejor 2) para estar seguros de que no hay nodos aislados

sorted(deg,cmp=reverse_numeric)                   % Ordenar los
grados de mayor a menor


deg_sat = []
% Creamos un deg_sat para ir restando los links que conectemos

for i in 0, N-1:

    deg_sat.append(deg[i])


red = []
% Creamos la lista de listas o diccionario de diccionarios donde van
las conexiones

list = []

for i in 0, N-1:

    red.append(list)


for i in 0, N-2:                                  % Vamos sobre todos
los nodos para conectarlos con aquellos con grado menor

    k = deg_sat[i]                                % Numero de links a
conectar

    n2 = i+1                                      % Primero nodo al
que podemos conectar

    for j in 0,k:

        while (n2 < N):                           % Vamos comprobando
los nodos a la derecha, si llegamos a n2 = N paramos

            if (deg_sat[n2] >0):                  % Si n2 tiene links
por conectar creamos la conexion

                red[i].append(n2)

                red[n2].append(i)
```

```
                deg_sat[i] = deg_sat[i]-1

                deg_sat[n2] = deg_sat[n2]-1

                n2 = n2 +1                      % En cualquier caso
pasamos al nodo siguiente hasta llegar a N

        if (n2 >= N) :                           % Si n2 es N ya no
podemos conectar mas links de i, cortamos el for de j
            break


cum_deg = []

Links_insat = 0

k = 0

for i in 0, N-1:

    Links_insat = Links_insat + deg_sat[i]      % Numero de links
sin conectar, a estudiar como funcion de N

    deg[i] = deg[i]-deg_sat[i]                  % Grado real de los
nodos, ojo: todos tienen que ser > 0 si no la red no sirve

    k = k + deg[i]

    cum_deg.append(k)                            % Vamos a crear un
array con el grado real acumulad a este punto, lo usamos luego.


for i in 0, N-1:

    for j in 0, deg[i]-1:                            % Vamos a
aleatorizar los links, esto hay que repetirlo varias veces

        n1 = i

        n2 = red[i][j]                               % Estos son los
nodos del primer link

        k = 0

        while (k < Num_max):                         % Maximo numero
de intentos para aleatorizar un link

            rand_link = int(cum_deg[N-1]*rand())    % Escogemos un
link al azar, rand() es un numero aleatorio en (0,1)

            k = k +1

            n3 = 0
```

```
            while (cum_deg[n3] <= rand_link):            % Hay
formas mas eficientes computacionalmente para encontrar n3 pero esta
funciona

                n3 = n3+1


            cum_deg_menor = 0

            if (n3 > 0):
                cum_deg_menor = cum_deg[n3-1]           %   Hay que
separar el caso n3 > 0 y n3 = 0

            n4 = red[n3][rand_link - cum_deg_menor]     % Aqui es
donde apunta el link seleccionado


            if ((n1 != n3) && (n1 != n4) && (n2 != n3) && (n2 !=
n4)):     % Hemos encontrado un posible link bueno, vamos a
comprobar que los nodos a conectar no tienen ya conexiones

                 key = 0
                foreach j in red[n1]:
                    if (j == n4):
                         key = 1

                foreach j in red[n2]:
                    if (j == n3):
                         key = 1

                if (key == 0):
% Ahora si, este es un link bueno para cambiar y vamos a hacer los
cambios

                 red[i][j] = n4                                    %
Conexion de n1 a n2 cambiada por n1 a n4


                 for m in 0,deg[n2]:
                                  if (red[n2][m] == n1):
                                  red[n2][m] = n3
% Conexion de n2 a n1 cambiada por n2 a n3



                 red[n3][rand_link - cum_deg_menor]] = n2        %
Conexion de n3 a n4 cambiado por n3 a n2


                 for m in 0,deg[n4]:
                                  if (red[n4][m] == n3):
                                  red[n4][m] = n1
% Conexion de n4 a n3 cambiada por n4 a n1
```

Esto ha construido ya la red aleatoria. Ahora solo queda comprobar si estar completamente conectada:

```
label = []

for i in 0, N-1:

    label.append(0)                          % Creamos una etiqueta para cada node, al principio todas son 0


Num_clus = 1                                 % Variable con el numero de clusters

label[0] = Num_clus

Nodos_sin_label = N-1

list_act = [0]                                % Vamos a crear dos listas con los nodos activos, los que acaban de cambiar de etiqueta

list_act2 = []


while (Nodos_sin_label > 0):                 % Mientras queden nodos sin etiqueta

    k = 0                                                 % contador de cambios

    for i in 0, len(list_act):

        for j in 0, deg[list_act[i]]:          % Vamos por los vecinos de los nodos activos

            if (label[red[list_act[i][j]] == 0):        % Si tienen aun un cero en la etiqueta los cambiamos y guardamos en list_act2

                k = k+1

                label[red[list_act[i][j]] = 1

                Nodos_sin_label = Nodos_sin_label - 1           % Esto baja los nodos sin etiqueta en 1

                list_act2.append(red[list_act[i][j])

                list_act = list_act2                            % Los nodos activos en esta "nueva generacion" pasan de list_act2 a list_act
```

```
            if ((k == 0) && (Nodos_sin_label > 0)):    % Si no hemos
hecho cambios puede ser que necesitemos buscar otro cluster

                for i in 0, N-1:

                    if (label[i] ==0):
% Aqui se comprueba si hay algun nodo con etiqueta 0 aun

                        Num_clus = Num_cluss+1                % Se
crea un nuevo cluster

                        label[i] =  Num_clus

                        list_act = [i]
% Se actualiza la lista de nodos activos y se para el for

                        break
```