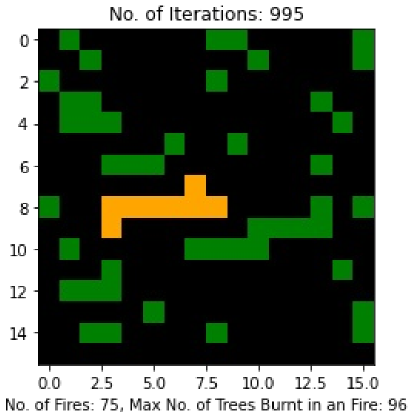
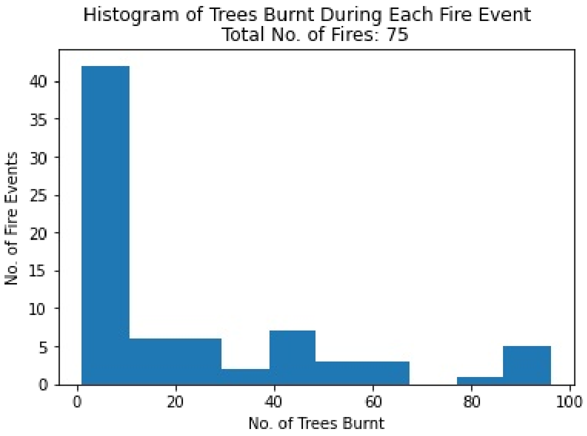
Exercise 3.1

In [110…

```
# Using matplotlib graphics to plot

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import time
from IPython.display import clear_output

# Parameters for simulation

N = l = 16

p_tree = 0.01
p_fire = 0.2
forest = np.zeros((N,N))
forestImage = np.zeros((N,N,3))
fire_count = 0
empty, tree, burning, burnt  = 0,1,2,3
# 1 = empty_forest
# 2 = growing trees
# 3 = burning forest(trees) and expanding fire
# 4 = burnt trees
size_of_fire = []
max_size_of_fire = 0

def PlotForest(forest, i, fire_count, max_size_of_fire):
    clear_output(wait=True)
    colors = ['Black','Green','Red','Orange']
    start = (int)(np.amin(forest))
    end = (int)(np.amax(forest))
    colormap = ListedColormap(colors[start:end+1])
    plt.imshow(forest,colormap)
    plt.title('No. of Iterations: '+str(iteration))
    plt.xlabel('No. of Fires: '+str(fire_count)+', Max No. of Trees Burnt in an Fire: '+str(max_size_of_fire))
    plt.show()
    plt.clf()

for iteration in range(1,1001):
    forest[(np.random.rand(N,N) < p_tree) & (forest == 0)] = 1   # Growing trees
    lightning_coordinates = (np.random.rand(2)*N).astype(int)
    if(forest[lightning_coordinates[0], lightning_coordinates[1]] == 1) and (np.random.rand() < p_fire):
        forest[lightning_coordinates[0], lightning_coordinates[1]] = 2 # burning the trees at these coordinates
        fire_count += 1

        while sum(sum(forest == 2)) > 0:
            for i,j in zip(np.where(forest == 2)[0], np.where(forest == 2)[1]):

                if (i > l-1 or i == l-1):
                    if forest[abs(i-(l-1)),j] == 1:
                        forest[abs(i-(l-1)),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3

                elif (i < l-l or i == l-l):
                    if forest[abs(i-(l-1)),j] == 1:
                        forest[abs(i-(l-1)),j] = 2
                    if forest[max(i+1,0),j] == 1:
                        forest[max(i+1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3


                elif (j > l-1 or j == l-1):
                    if forest[i, abs(j-(l-1))] == 1:
                        forest[i, abs(j-(l-1))] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    forest[i,j] = 3

                elif (j < l-l or j == l-l):
                    if forest[i, abs(j-(l-1))] == 1:
                        forest[i, abs(j-(l-1))] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 3
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    forest[i,j] = 3

                else:
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3

            if sum(sum(forest == 3)) > max_size_of_fire:
                max_size_of_fire = sum(sum(forest == 3))

        size_of_fire.append(sum(sum(forest == 3)))
        PlotForest(forest, iteration, fire_count, max_size_of_fire)

    forest[forest == 3] = 0

plt.hist(size_of_fire, bins = 10)
plt.xlabel('No. of Trees Burnt')
plt.ylabel('No. of Fire Events')
plt.suptitle('Histogram of Trees Burnt During Each Fire Event')
plt.title('Total No. of Fires: ' + str(fire_count))
plt.show()
```



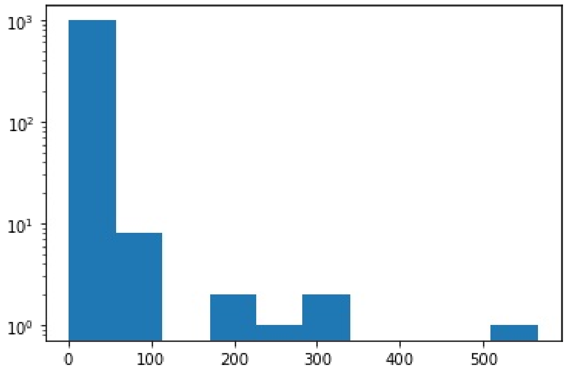


Exercise 3.3

In [111…

```
import numpy as np
import matplotlib.pyplot as plt

alpha = 2
exponent = 1 / (1 - alpha)
sequence_min = 1
sequence_length = 1000
sequence = np.zeros(sequence_length)
```

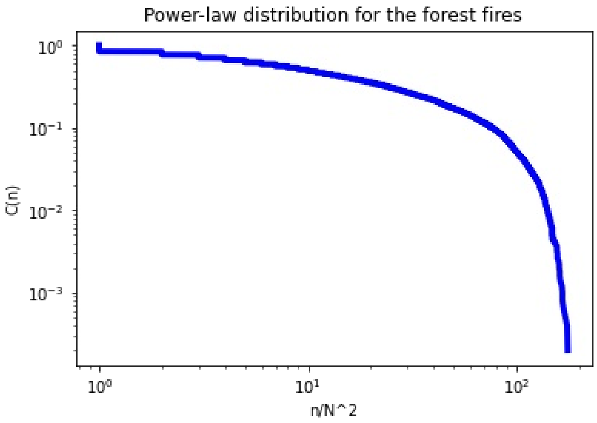
```
for i in range(sequence_length):
    sequence[i] = sequence_min * (np.random.random() ** exponent)

plt.hist(sequence, bins=10)
plt.yscale("log")
```



Exercise 3.4 (N = 16)

In [27]:

```
# Using matplotlib graphics to plot
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import time
from IPython.display import clear_output

# Parameters for simulation

N = l = 16

p_tree = 0.01
p_fire = 0.2
forest = np.zeros((N,N))
forestImage = np.zeros((N,N,3))
fire_count = 0
empty, tree, burning, burnt  = 0,1,2,3
# 1 = empty_forest
# 2 = growing trees
# 3 = burning forest(trees) and expanding fire
# 4 = burnt trees
size_of_fire = []
max_size_of_fire = 0

while fire_count < 5000:
    forest[(np.random.rand(N,N) < p_tree) & (forest == 0)] = 1   # Growing trees
    lightning_coordinates = (np.random.rand(2)*N).astype(int)
    if(forest[lightning_coordinates[0], lightning_coordinates[1]] == 1) and (np.random.rand() < p_fire):
        forest[lightning_coordinates[0], lightning_coordinates[1]] = 2 # burning the trees at these coordinates
        fire_count += 1

        while sum(sum(forest == 2)) > 0:
            for i,j in zip(np.where(forest == 2)[0], np.where(forest == 2)[1]):

                if (i > l-1 or i == l-1):
                    if forest[abs(i-(l-1)),j] == 1:
                        forest[abs(i-(l-1)),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3

                elif (i < l-l or i == l-l):
                    if forest[abs(i-(l-1)),j] == 1:
                        forest[abs(i-(l-1)),j] = 2
                    if forest[max(i+1,0),j] == 1:
                        forest[max(i+1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3


                elif (j > l-1 or j == l-1):
                    if forest[i, abs(j-(l-1))] == 1:
                        forest[i, abs(j-(l-1))] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    forest[i,j] = 3

                elif (j < l-l or j == l-l):
                    if forest[i, abs(j-(l-1))] == 1:
                        forest[i, abs(j-(l-1))] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 3
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    forest[i,j] = 3

                else:
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3

            if sum(sum(forest == 3)) > max_size_of_fire:
                max_size_of_fire = sum(sum(forest == 3))

        size_of_fire.append(sum(sum(forest == 3)))
    forest[forest == 3] = 0

size_of_fire.sort()
k = 5000
C = []
for i in range(0,k):
    c = (k-i)/k
    C.append(c)

ratio = [a/N*N for a in size_of_fire]

plt.plot(ratio,C,"b",linewidth= 4)
plt.yscale("log")
plt.xscale("log")
plt.xlabel("n/N^2")
plt.ylabel("C(n)")
plt.title("Power-law distribution for the forest fires")
plt.show()
```



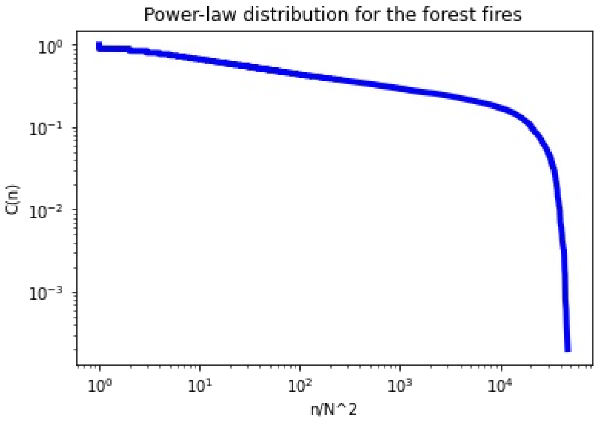
Exercise 3.4 (N = 256)

In [28]:

```
# Using matplotlib graphics to plot
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import time
from IPython.display import clear_output

# Parameters for simulation
```

```
N = l = 256

p_tree = 0.01
p_fire = 0.2
forest = np.zeros((N,N))
forestImage = np.zeros((N,N,3))
fire_count = 0
empty, tree, burning, burnt  = 0,1,2,3
# 1 = empty_forest
# 2 = growing trees
# 3 = burning forest(trees) and expanding fire
# 4 = burnt trees
size_of_fire = []
max_size_of_fire = 0

while fire_count < 5000:
    forest[(np.random.rand(N,N) < p_tree) & (forest == 0)] = 1   # Growing trees
    lightning_coordinates = (np.random.rand(2)*N).astype(int)
    if(forest[lightning_coordinates[0], lightning_coordinates[1]] == 1) and (np.random.rand() < p_fire):
        forest[lightning_coordinates[0], lightning_coordinates[1]] = 2 # burning the trees at these coordinates
        fire_count += 1

        while sum(sum(forest == 2)) > 0:
            for i,j in zip(np.where(forest == 2)[0], np.where(forest == 2)[1]):

                if (i > l-1 or i == l-1):
                    if forest[abs(i-(l-1)),j] == 1:
                        forest[abs(i-(l-1)),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3

                elif (i < l-l or i == l-l):
                    if forest[abs(i-(l-1)),j] == 1:
                        forest[abs(i-(l-1)),j] = 2
                    if forest[max(i+1,0),j] == 1:
                        forest[max(i+1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3


                elif (j > l-1 or j == l-1):
                    if forest[i, abs(j-(l-1))] == 1:
                        forest[i, abs(j-(l-1))] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    forest[i,j] = 3

                elif (j < l-l or j == l-l):
                    if forest[i, abs(j-(l-1))] == 1:
                        forest[i, abs(j-(l-1))] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 3
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    forest[i,j] = 3

                else:
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3

            if sum(sum(forest == 3)) > max_size_of_fire:
                max_size_of_fire = sum(sum(forest == 3))

        size_of_fire.append(sum(sum(forest == 3)))
    forest[forest == 3] = 0

size_of_fire.sort()
```

```
k = 5000
C = []
for i in range(0,k):
    c = (k-i)/k
    C.append(c)

ratio = [a/N*N for a in size_of_fire]

plt.plot(ratio,C,"b",linewidth= 4)
plt.yscale("log")
plt.xscale("log")
plt.xlabel("n/N^2")
plt.ylabel("C(n)")
plt.title("Power-law distribution for the forest fires")
plt.show()
```



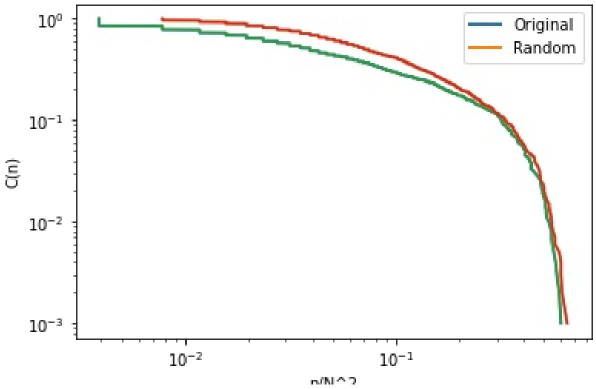
Exercise 3.5 (N = 16)

In [80]:

```
import numpy as np
import random
import numpy.random
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap
import time
from IPython.display import clear_output
from sklearn.utils import shuffle
```

In [81]:

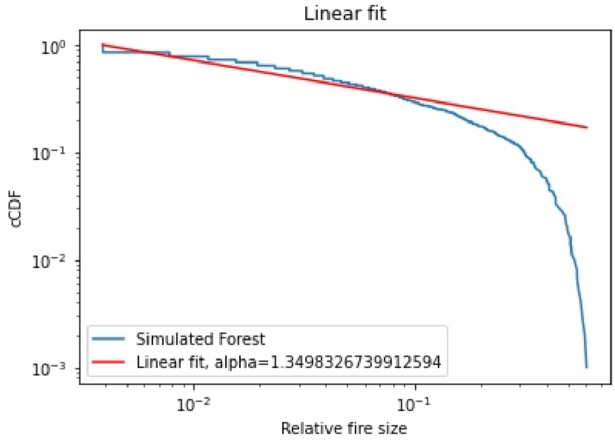
```
# N = l = 16

p_tree = 0.01
p_fire = 0.2
forest = np.zeros((N,N))
forestImage = np.zeros((N,N,3))
fire_count = 0
empty, tree, burning, burnt  = 0,1,2,3
random_fire_count = 0
number_of_burnt_trees_original_forest = 0
number_of_burnt_trees_random_forest = 0
grown_trees_original_forest = []
burnt_trees_original_forest = []
burnt_trees_random_forest = []

def random_forest_generate(forest):

    temp_forest = forest
    temp_forest = shuffle(temp_forest)
    return temp_forest

def random_fire_location(temp_forest, N, random_fire_count, burnt_trees_random_forest):
    tree_coordinates = list(zip(np.where(temp_forest==1)[0],np.where(temp_forest==1)[1]))
    random_tree_coordinate_for_fire = random.choice(tree_coordinates) #(np.random.rand(2)*N).astype(int)
    random_fire_count =+ 1
    temp_forest[random_tree_coordinate_for_fire[0],random_tree_coordinate_for_fire[1]] = 2

#    Fire spreading to all neighbouring trees
    while sum(sum(temp_forest == 2)) > 0:
            for i,j in zip(np.where(temp_forest == 2)[0], np.where(temp_forest == 2)[1]):

                if (i > l-1 or i == l-1):
                    if temp_forest[abs(i-(l-1)),j] == 1:
                        temp_forest[abs(i-(l-1)),j] = 2
                    if temp_forest[max(i-1,0),j] == 1:
                        temp_forest[max(i-1,0),j] = 2
                    if temp_forest[i,min(j+1,l-1)] == 1:
                        temp_forest[i,min(j+1,l-1)] = 2
                    if temp_forest[i,max(j-1,0)] == 1:
                        temp_forest[i,max(j-1,0)] = 2
                    temp_forest[i,j] = 3

                elif (i < l-l or i == l-l):
                    if temp_forest[abs(i-(l-1)),j] == 1:
                        temp_forest[abs(i-(l-1)),j] = 2
                    if temp_forest[max(i+1,0),j] == 1:
                        temp_forest[max(i+1,0),j] = 2
                    if temp_forest[i,min(j+1,l-1)] == 1:
                        temp_forest[i,min(j+1,l-1)] = 2
                    if temp_forest[i,max(j-1,0)] == 1:
                        temp_forest[i,max(j-1,0)] = 2
                    temp_forest[i,j] = 3

                elif (j > l-1 or j == l-1):
                    if temp_forest[i, abs(j-(l-1))] == 1:
                        temp_forest[i, abs(j-(l-1))] = 2
                    if temp_forest[i,max(j-1,0)] == 1:
                        temp_forest[i,max(j-1,0)] = 2
                    if temp_forest[min(i+1,l-1),j] == 1:
                        temp_forest[min(i+1,l-1),j] = 2
                    if temp_forest[max(i-1,0),j] == 1:
                        temp_forest[max(i-1,0),j] = 2
                    temp_forest[i,j] = 3

                elif (j < l-l or j == l-l):
                    if temp_forest[i, abs(j-(l-1))] == 1:
                        temp_forest[i, abs(j-(l-1))] = 2
                    if temp_forest[i,min(j+1,l-1)] == 1:
                        temp_forest[i,min(j+1,l-1)] = 3
                    if temp_forest[min(i+1,l-1),j] == 1:
                        temp_forest[min(i+1,l-1),j] = 2
                    if temp_forest[max(i-1,0),j] == 1:
                        temp_forest[max(i-1,0),j] = 2
                    temp_forest[i,j] = 3

                else:
                    if temp_forest[min(i+1,l-1),j] == 1:
                        temp_forest[min(i+1,l-1),j] = 2
                    if temp_forest[max(i-1,0),j] == 1:
                        temp_forest[max(i-1,0),j] = 2
                    if temp_forest[i,min(j+1,l-1)] == 1:
                        temp_forest[i,min(j+1,l-1)] = 2
                    if temp_forest[i,max(j-1,0)] == 1:
                        temp_forest[i,max(j-1,0)] = 2
                    temp_forest[i,j] = 3


    number_of_burnt_trees_random_forest = sum(sum(temp_forest == 3))
    burnt_trees_random_forest.append(number_of_burnt_trees_random_forest)
    return burnt_trees_random_forest
    temp_forest[temp_forest == 3] = 0

while fire_count < 1000:
    forest[(np.random.rand(N,N) < p_tree) & (forest == 0)] = 1   # Growing trees
#     print('forest', forest)
    number_of_trees_grown = sum(sum(forest == 1))
    grown_trees_original_forest.append(number_of_trees_grown)
    lightning_coordinates = (np.random.rand(2)*N).astype(int)
    if(forest[lightning_coordinates[0], lightning_coordinates[1]] == 1) and (np.random.rand() < p_fire):
        forest[lightning_coordinates[0], lightning_coordinates[1]] = 2 # burning the trees at these coordinates
        fire_count += 1
        random_forest_generation = random_forest_generate(forest)
        fire_generation = random_fire_location(random_forest_generation, N, random_fire_count, burnt_trees_random

        while sum(sum(forest == 2)) > 0:
            for i,j in zip(np.where(forest == 2)[0], np.where(forest == 2)[1]):

                if (i > l-1 or i == l-1):
                    if forest[abs(i-(l-1)),j] == 1:
                        forest[abs(i-(l-1)),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3

                elif (i < l-l or i == l-l):
                    if forest[abs(i-(l-1)),j] == 1:
                        forest[abs(i-(l-1)),j] = 2
                    if forest[max(i+1,0),j] == 1:
                        forest[max(i+1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3


                elif (j > l-1 or j == l-1):
                    if forest[i, abs(j-(l-1))] == 1:
                        forest[i, abs(j-(l-1))] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    forest[i,j] = 3

                elif (j < l-l or j == l-l):
                    if forest[i, abs(j-(l-1))] == 1:
                        forest[i, abs(j-(l-1))] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 3
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    forest[i,j] = 3

                else:
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3

        number_of_burnt_trees_original_forest = sum(sum(forest == 3))
        burnt_trees_original_forest.append(number_of_burnt_trees_original_forest)
        forest[forest == 3] = 0


# print('fire count', fire_count)
# print('trees burnt original forest' ,burnt_trees_original_forest)
# print(len(burnt_trees_original_forest))
# print('trees burnt random forest' ,burnt_trees_random_forest)
# print(len(burnt_trees_random_forest))


burnt_trees_original_forest.sort()
burnt_trees_random_forest.sort()


k = fire_count
C =[]
for i in range(0,k):
    c = (k-i)/k
    C.append(c)

burnt_trees_original_forest = np.array(burnt_trees_original_forest)
ratio1 = burnt_trees_original_forest/(N*N)
burnt_trees_random_forest = np.array(burnt_trees_random_forest)
ratio2 = burnt_trees_random_forest/(N*N)
line1, = plt.plot(ratio1, C, linewidth = 2)
line2, = plt.plot(ratio2, C, linewidth = 2)
plt.loglog(ratio1, C)
plt.loglog(ratio2, C)
plt.xlabel("n/N^2")
plt.ylabel("C(n)")
plt.legend(['Original','Random'])
plt.show()
```



Exercise 3.6 (N=16)

In [82]:

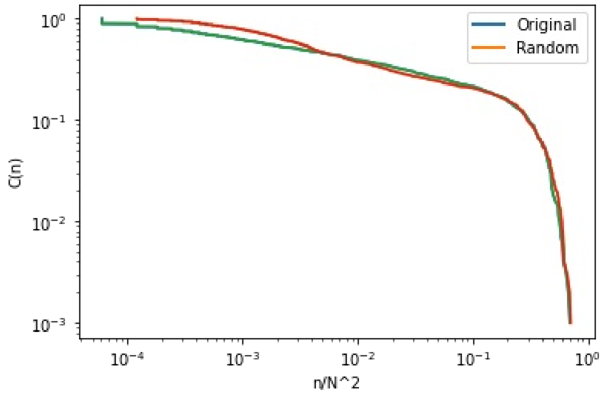
```
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
# scipy.optimize.curve_fit(f, xdata, ydata, p0=None, sigma=None, absolute_sigma=False, check_finite=True, **kw)
forest = 'Simulated Forest'
x_values= np.sort(burnt_trees_original_forest)/(N*N)
y_values= C

def func(x,slope,b):
    return x**slope*np.exp(b)

x = np.linspace(10e-5,1,len(x_values))
y = func(x, -0.15, 1)


# Optimal values for the parameters so that the sum of the squared error of f(xdata, *popt) - ydata is minimized
# The estimated covariance of popt. The diagonals provide the variance of the parameter estimate.
popt, pcov = curve_fit(func, x_values, y_values)
popt
figur = plt.figure()
ax = figur.add_axes([0.1,0.1,.8,.8])

plt.loglog(x_values, y_values,label = forest)
plt.loglog(x_values, func(x_values, *popt), 'r-',
         label=f'Linear fit, alpha={1-popt[0]}')

popt, pcov = curve_fit(func, x, y)
popt

plt.xlabel('Relative fire size')
plt.ylabel('cCDF')
plt.title(f'Linear fit')
plt.legend()
plt.show()
```



Exercise 3.5 (N = 128)

In [83]:

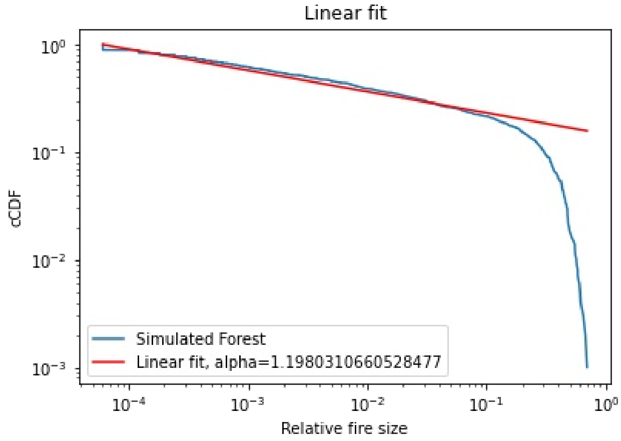
```
N = l = 128

p_tree = 0.01
p_fire = 0.2
forest = np.zeros((N,N))
forestImage = np.zeros((N,N,3))
fire_count = 0
empty, tree, burning, burnt  = 0,1,2,3
random_fire_count = 0
number_of_burnt_trees_original_forest = 0
number_of_burnt_trees_random_forest = 0
grown_trees_original_forest = []
burnt_trees_original_forest = []
burnt_trees_random_forest = []

def random_forest_generate(forest):

    temp_forest = forest
    temp_forest = shuffle(temp_forest)
    return temp_forest

def random_fire_location(temp_forest, N, random_fire_count, burnt_trees_random_forest):
    tree_coordinates = list(zip(np.where(temp_forest==1)[0],np.where(temp_forest==1)[1]))
    random_tree_coordinate_for_fire = random.choice(tree_coordinates) #(np.random.rand(2)*N).astype(int)
    random_fire_count =+ 1
    temp_forest[random_tree_coordinate_for_fire[0],random_tree_coordinate_for_fire[1]] = 2

#    Fire spreading to all neighbouring trees
    while sum(sum(temp_forest == 2)) > 0:
            for i,j in zip(np.where(temp_forest == 2)[0], np.where(temp_forest == 2)[1]):

                if (i > l-1 or i == l-1):
                    if temp_forest[abs(i-(l-1)),j] == 1:
                        temp_forest[abs(i-(l-1)),j] = 2
                    if temp_forest[max(i-1,0),j] == 1:
                        temp_forest[max(i-1,0),j] = 2
                    if temp_forest[i,min(j+1,l-1)] == 1:
                        temp_forest[i,min(j+1,l-1)] = 2
                    if temp_forest[i,max(j-1,0)] == 1:
                        temp_forest[i,max(j-1,0)] = 2
                    temp_forest[i,j] = 3

                elif (i < l-l or i == l-l):
                    if temp_forest[abs(i-(l-1)),j] == 1:
                        temp_forest[abs(i-(l-1)),j] = 2
                    if temp_forest[max(i+1,0),j] == 1:
                        temp_forest[max(i+1,0),j] = 2
                    if temp_forest[i,min(j+1,l-1)] == 1:
                        temp_forest[i,min(j+1,l-1)] = 2
                    if temp_forest[i,max(j-1,0)] == 1:
                        temp_forest[i,max(j-1,0)] = 2
                    temp_forest[i,j] = 3

                elif (j > l-1 or j == l-1):
                    if temp_forest[i, abs(j-(l-1))] == 1:
                        temp_forest[i, abs(j-(l-1))] = 2
                    if temp_forest[i,max(j-1,0)] == 1:
                        temp_forest[i,max(j-1,0)] = 2
                    if temp_forest[min(i+1,l-1),j] == 1:
                        temp_forest[min(i+1,l-1),j] = 2
                    if temp_forest[max(i-1,0),j] == 1:
                        temp_forest[max(i-1,0),j] = 2
                    temp_forest[i,j] = 3

                elif (j < l-l or j == l-l):
                    if temp_forest[i, abs(j-(l-1))] == 1:
                        temp_forest[i, abs(j-(l-1))] = 2
                    if temp_forest[i,min(j+1,l-1)] == 1:
                        temp_forest[i,min(j+1,l-1)] = 3
                    if temp_forest[min(i+1,l-1),j] == 1:
                        temp_forest[min(i+1,l-1),j] = 2
                    if temp_forest[max(i-1,0),j] == 1:
                        temp_forest[max(i-1,0),j] = 2
                    temp_forest[i,j] = 3

                else:
                    if temp_forest[min(i+1,l-1),j] == 1:
                        temp_forest[min(i+1,l-1),j] = 2
                    if temp_forest[max(i-1,0),j] == 1:
                        temp_forest[max(i-1,0),j] = 2
                    if temp_forest[i,min(j+1,l-1)] == 1:
                        temp_forest[i,min(j+1,l-1)] = 2
                    if temp_forest[i,max(j-1,0)] == 1:
                        temp_forest[i,max(j-1,0)] = 2
                    temp_forest[i,j] = 3


    number_of_burnt_trees_random_forest = sum(sum(temp_forest == 3))
    burnt_trees_random_forest.append(number_of_burnt_trees_random_forest)
    return burnt_trees_random_forest
    temp_forest[temp_forest == 3] = 0

while fire_count < 1000:
    forest[(np.random.rand(N,N) < p_tree) & (forest == 0)] = 1   # Growing trees
#     print('forest', forest)
    number_of_trees_grown = sum(sum(forest == 1))
    grown_trees_original_forest.append(number_of_trees_grown)
    lightning_coordinates = (np.random.rand(2)*N).astype(int)
    if(forest[lightning_coordinates[0], lightning_coordinates[1]] == 1) and (np.random.rand() < p_fire):
        forest[lightning_coordinates[0], lightning_coordinates[1]] = 2 # burning the trees at these coordinates
        fire_count += 1
        random_forest_generation = random_forest_generate(forest)
        fire_generation = random_fire_location(random_forest_generation, N, random_fire_count, burnt_trees_random

        while sum(sum(forest == 2)) > 0:
            for i,j in zip(np.where(forest == 2)[0], np.where(forest == 2)[1]):

                if (i > l-1 or i == l-1):
                    if forest[abs(i-(l-1)),j] == 1:
                        forest[abs(i-(l-1)),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3

                elif (i < l-l or i == l-l):
                    if forest[abs(i-(l-1)),j] == 1:
                        forest[abs(i-(l-1)),j] = 2
                    if forest[max(i+1,0),j] == 1:
                        forest[max(i+1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3


                elif (j > l-1 or j == l-1):
                    if forest[i, abs(j-(l-1))] == 1:
                        forest[i, abs(j-(l-1))] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    forest[i,j] = 3

                elif (j < l-l or j == l-l):
                    if forest[i, abs(j-(l-1))] == 1:
                        forest[i, abs(j-(l-1))] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 3
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    forest[i,j] = 3

                else:
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3

        number_of_burnt_trees_original_forest = sum(sum(forest == 3))
        burnt_trees_original_forest.append(number_of_burnt_trees_original_forest)
        forest[forest == 3] = 0


# print('fire count', fire_count)
# print('trees burnt original forest' ,burnt_trees_original_forest)
# print(len(burnt_trees_original_forest))
# print('trees burnt random forest' ,burnt_trees_random_forest)
# print(len(burnt_trees_random_forest))


burnt_trees_original_forest.sort()
burnt_trees_random_forest.sort()


k = fire_count
C =[]
for i in range(0,k):
    c = (k-i)/k
    C.append(c)

burnt_trees_original_forest = np.array(burnt_trees_original_forest)
ratio1 = burnt_trees_original_forest/(N*N)
burnt_trees_random_forest = np.array(burnt_trees_random_forest)
ratio2 = burnt_trees_random_forest/(N*N)
line1, = plt.plot(ratio1, C, linewidth = 2)
line2, = plt.plot(ratio2, C, linewidth = 2)
plt.loglog(ratio1, C)
plt.loglog(ratio2, C)
plt.xlabel("n/N^2")
plt.ylabel("C(n)")
plt.legend(['Original','Random'])
plt.show()
```



Exercise 3.6 (N = 128)

In [92]:

```
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
# scipy.optimize.curve_fit(f, xdata, ydata, p0=None, sigma=None, absolute_sigma=False, check_finite=True, **kw)
forest = 'Simulated Forest'
x_values= np.sort(burnt_trees_original_forest)/(N*N)
y_values= C

def func(x,slope,b):
    return x**slope*np.exp(b)

x = np.linspace(10e-5,1,len(x_values))
y = func(x, -0.15, 1)


# Optimal values for the parameters so that the sum of the squared error of f(xdata, *popt) - ydata is minimized
# The estimated covariance of popt. The diagonals provide the variance of the parameter estimate.
popt, pcov = curve_fit(func, x_values, y_values)
popt
figur = plt.figure()
ax = figur.add_axes([0.1,0.1,.8,.8])

plt.loglog(x_values, y_values,label = forest)
plt.loglog(x_values, func(x_values, *popt), 'r-',label=f'Linear fit, alpha={1-popt[0]}')

popt, pcov = curve_fit(func, x, y)
popt

plt.xlabel('Relative fire size')
plt.ylabel('cCDF')
plt.title(f'Linear fit')
plt.legend()
plt.show()
```

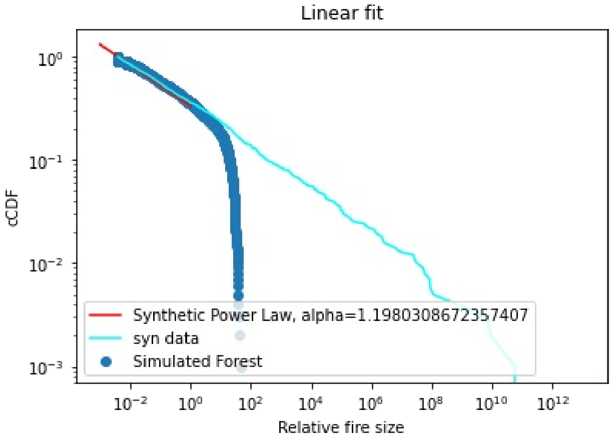


Exercise 3.7

In [120…

```
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
```

```
forest = 'Simulated Forest'
x_values= np.sort(burnt_trees_original_forest)/(N*N)
y_values= C
total_area = N*N

def func(x,slope,b):
    return x**slope*np.exp(b)

x = np.linspace(1/len(x_values),1,len(x_values))
y = func(x, -0.15, 1)


popt, pcov = curve_fit(func, x_values, y_values)
popt


alpha = 1 - popt[0]
rand_vector = np.sort(np.random.random(1000))
x_min = 1 # min(burnt_trees_random_forest)
y_syn = np.linspace(0,1,len(rand_vector))

cCDF_syn =  x_min*(1-rand_vector)**(-1/(alpha-1))
cCDF_syn = np.flip(cCDF_syn)/(total_area)


figur = plt.figure()
ax = figur.add_axes([0.1,0.1,.8,.8])
plt.scatter(x_values, y_values, label = forest)
plt.loglog(x, func(x, *popt), 'r-', label=f'Synthetic Power Law, alpha={1-popt[0]}')
plt.loglog(cCDF_syn, y_syn, c='cyan' ,label='syn data')
plt.xlabel('Relative fire size')
plt.ylabel('cCDF')
plt.title(f'Linear fit')
plt.legend()
# ax.set_xlim([None, 1.5])
plt.show()
alpha = 1 - popt[0]
```



Exercise 3.8

In [ ]:

```
N = l = 128

p_tree = 0.01
p_fire = 0.2
forest = np.zeros((N,N))
forestImage = np.zeros((N,N,3))
fire_count = 0
empty, tree, burning, burnt  = 0,1,2,3
random_fire_count = 0
number_of_burnt_trees_original_forest = 0
number_of_burnt_trees_random_forest = 0
grown_trees_original_forest = []
burnt_trees_original_forest = []
burnt_trees_random_forest = []

def random_forest_generate(forest):

    temp_forest = forest
    temp_forest = shuffle(temp_forest)
    return temp_forest

def random_fire_location(temp_forest, N, random_fire_count, burnt_trees_random_forest):
    tree_coordinates = list(zip(np.where(temp_forest==1)[0],np.where(temp_forest==1)[1]))
    random_tree_coordinate_for_fire = random.choice(tree_coordinates) #(np.random.rand(2)*N).astype(int)
    random_fire_count =+ 1
    temp_forest[random_tree_coordinate_for_fire[0],random_tree_coordinate_for_fire[1]] = 2

#    Fire spreading to all neighbouring trees
    while sum(sum(temp_forest == 2)) > 0:
            for i,j in zip(np.where(temp_forest == 2)[0], np.where(temp_forest == 2)[1]):

                if (i > l-1 or i == l-1):
                    if temp_forest[abs(i-(l-1)),j] == 1:
                        temp_forest[abs(i-(l-1)),j] = 2
                    if temp_forest[max(i-1,0),j] == 1:
                        temp_forest[max(i-1,0),j] = 2
                    if temp_forest[i,min(j+1,l-1)] == 1:
                        temp_forest[i,min(j+1,l-1)] = 2
                    if temp_forest[i,max(j-1,0)] == 1:
                        temp_forest[i,max(j-1,0)] = 2
                    temp_forest[i,j] = 3

                elif (i < l-l or i == l-l):
                    if temp_forest[abs(i-(l-1)),j] == 1:
                        temp_forest[abs(i-(l-1)),j] = 2
                    if temp_forest[max(i+1,0),j] == 1:
                        temp_forest[max(i+1,0),j] = 2
                    if temp_forest[i,min(j+1,l-1)] == 1:
                        temp_forest[i,min(j+1,l-1)] = 2
                    if temp_forest[i,max(j-1,0)] == 1:
                        temp_forest[i,max(j-1,0)] = 2
                    temp_forest[i,j] = 3

                elif (j > l-1 or j == l-1):
                    if temp_forest[i, abs(j-(l-1))] == 1:
                        temp_forest[i, abs(j-(l-1))] = 2
                    if temp_forest[i,max(j-1,0)] == 1:
                        temp_forest[i,max(j-1,0)] = 2
                    if temp_forest[min(i+1,l-1),j] == 1:
                        temp_forest[min(i+1,l-1),j] = 2
                    if temp_forest[max(i-1,0),j] == 1:
                        temp_forest[max(i-1,0),j] = 2
                    temp_forest[i,j] = 3

                elif (j < l-l or j == l-l):
                    if temp_forest[i, abs(j-(l-1))] == 1:
                        temp_forest[i, abs(j-(l-1))] = 2
                    if temp_forest[i,min(j+1,l-1)] == 1:
                        temp_forest[i,min(j+1,l-1)] = 3
                    if temp_forest[min(i+1,l-1),j] == 1:
                        temp_forest[min(i+1,l-1),j] = 2
                    if temp_forest[max(i-1,0),j] == 1:
                        temp_forest[max(i-1,0),j] = 2
                    temp_forest[i,j] = 3

                else:
                    if temp_forest[min(i+1,l-1),j] == 1:
                        temp_forest[min(i+1,l-1),j] = 2
                    if temp_forest[max(i-1,0),j] == 1:
                        temp_forest[max(i-1,0),j] = 2
                    if temp_forest[i,min(j+1,l-1)] == 1:
                        temp_forest[i,min(j+1,l-1)] = 2
                    if temp_forest[i,max(j-1,0)] == 1:
                        temp_forest[i,max(j-1,0)] = 2
                    temp_forest[i,j] = 3


    number_of_burnt_trees_random_forest = sum(sum(temp_forest == 3))
    burnt_trees_random_forest.append(number_of_burnt_trees_random_forest)
    return burnt_trees_random_forest
    temp_forest[temp_forest == 3] = 0

while fire_count < 1000:
    forest[(np.random.rand(N,N) < p_tree) & (forest == 0)] = 1   # Growing trees
#     print('forest', forest)
    number_of_trees_grown = sum(sum(forest == 1))
    grown_trees_original_forest.append(number_of_trees_grown)
    lightning_coordinates = (np.random.rand(2)*N).astype(int)
    if(forest[lightning_coordinates[0], lightning_coordinates[1]] == 1) and (np.random.rand() < p_fire):
        forest[lightning_coordinates[0], lightning_coordinates[1]] = 2 # burning the trees at these coordinates
        fire_count += 1
        random_forest_generation = random_forest_generate(forest)
        fire_generation = random_fire_location(random_forest_generation, N, random_fire_count, burnt_trees_random

        while sum(sum(forest == 2)) > 0:
            for i,j in zip(np.where(forest == 2)[0], np.where(forest == 2)[1]):

                if (i > l-1 or i == l-1):
                    if forest[abs(i-(l-1)),j] == 1:
                        forest[abs(i-(l-1)),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3

                elif (i < l-l or i == l-l):
                    if forest[abs(i-(l-1)),j] == 1:
                        forest[abs(i-(l-1)),j] = 2
                    if forest[max(i+1,0),j] == 1:
                        forest[max(i+1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3


                elif (j > l-1 or j == l-1):
                    if forest[i, abs(j-(l-1))] == 1:
                        forest[i, abs(j-(l-1))] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    forest[i,j] = 3

                elif (j < l-l or j == l-l):
                    if forest[i, abs(j-(l-1))] == 1:
                        forest[i, abs(j-(l-1))] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 3
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    forest[i,j] = 3

                else:
                    if forest[min(i+1,l-1),j] == 1:
                        forest[min(i+1,l-1),j] = 2
                    if forest[max(i-1,0),j] == 1:
                        forest[max(i-1,0),j] = 2
                    if forest[i,min(j+1,l-1)] == 1:
                        forest[i,min(j+1,l-1)] = 2
                    if forest[i,max(j-1,0)] == 1:
                        forest[i,max(j-1,0)] = 2
                    forest[i,j] = 3

        number_of_burnt_trees_original_forest = sum(sum(forest == 3))
        burnt_trees_original_forest.append(number_of_burnt_trees_original_forest)
        forest[forest == 3] = 0


# print('fire count', fire_count)
# print('trees burnt original forest' ,burnt_trees_original_forest)
# print(len(burnt_trees_original_forest))
# print('trees burnt random forest' ,burnt_trees_random_forest)
# print(len(burnt_trees_random_forest))


burnt_trees_original_forest.sort()
burnt_trees_random_forest.sort()


k = fire_count
C =[]
for i in range(0,k):
    c = (k-i)/k
    C.append(c)

burnt_trees_original_forest = np.array(burnt_trees_original_forest)
ratio1 = burnt_trees_original_forest/(N*N)
burnt_trees_random_forest = np.array(burnt_trees_random_forest)
ratio2 = burnt_trees_random_forest/(N*N)
line1, = plt.plot(ratio1, C, linewidth = 2)
line2, = plt.plot(ratio2, C, linewidth = 2)
plt.loglog(ratio1, C)
plt.loglog(ratio2, C)
plt.xlabel("n/N^2")
plt.ylabel("C(n)")
plt.legend(['Original','Random'])
plt.show()
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

N	Alpha
16	1.35294
32	1.27227
64	1.22548
128	1.20324
256	1.17201
512	1.15345