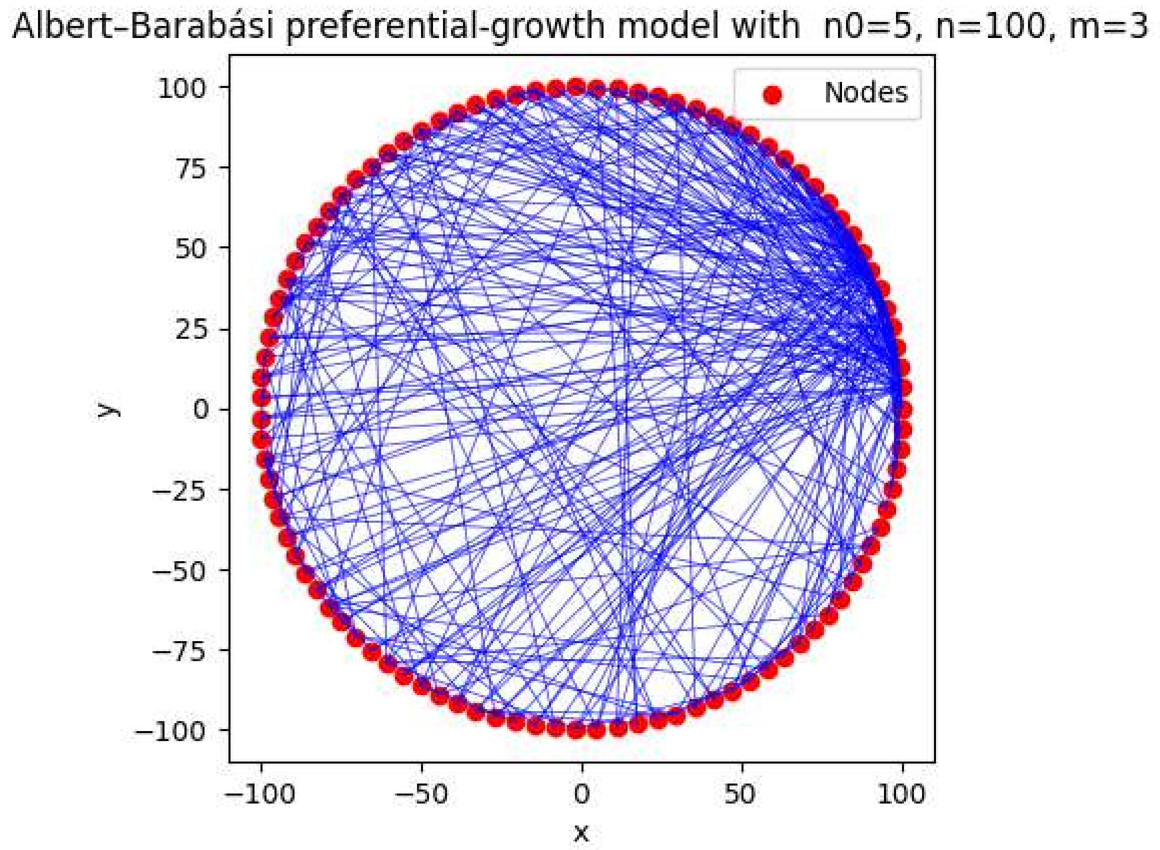
12.3a

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import random

n0=5
n=100
m=3
r=100

amatrix= np.ones((n0,n0))
np.fill_diagonal(amatrix, 0)

def find_degree (matrix):
    degree= np.zeros((len(matrix),1))
    for i in range(len(matrix)):
        degree[i,:] = np.sum(matrix[i,:])
    total_degree = np.sum(degree)
    cumulative_sum = np.cumsum(degree)
    prob = cumulative_sum/total_degree
    return (prob)

for i in range(1,n-n0+1):
    new_matrix = np.zeros((n0+i, n0+i))
    new_matrix[:-1,:-1] = amatrix
    new_edge =0
    previous_list=[]
    while new_edge<m:
        prob = find_degree(amatrix)
        random_number = np.random.uniform(0,1)
        for a in range(len(prob)):
            if prob[a]>=random_number and a not in previous_list:
                # toss = np.random.uniform(0,1)
                # if toss >=0.5:
                new_matrix[n0+i-1][a]=1
                # else:
                new_matrix[a][n0+i-1]=1
                new_edge+=1
                previous_list.append(a)
                break
                # print(new_edge)
    amatrix = new_matrix



theta = np.linspace(0, 2*np.pi, len(amatrix))
x = r * np.cos(theta)
y = r * np.sin(theta)




plt.figure(figsize=(10, 6))

plt.subplot(1,2,1)
plt.scatter(x, y, color='red', label='Nodes')
for i in range(n):
    for j in range(n):
        if amatrix[i][j] ==1:
            plt.plot([x[i], x[j]], [y[i], y[j]], color='blue', alpha=0.5, linewidth= 0.5)
plt.gca().set_aspect('equal', adjustable='box')
plt.xlabel('x')
plt.ylabel('y')
plt.legend()
plt.title(f'Albert-Barabási preferential-growth model with  n0={n0}, n={n}, m={m}')
plt.show()
```

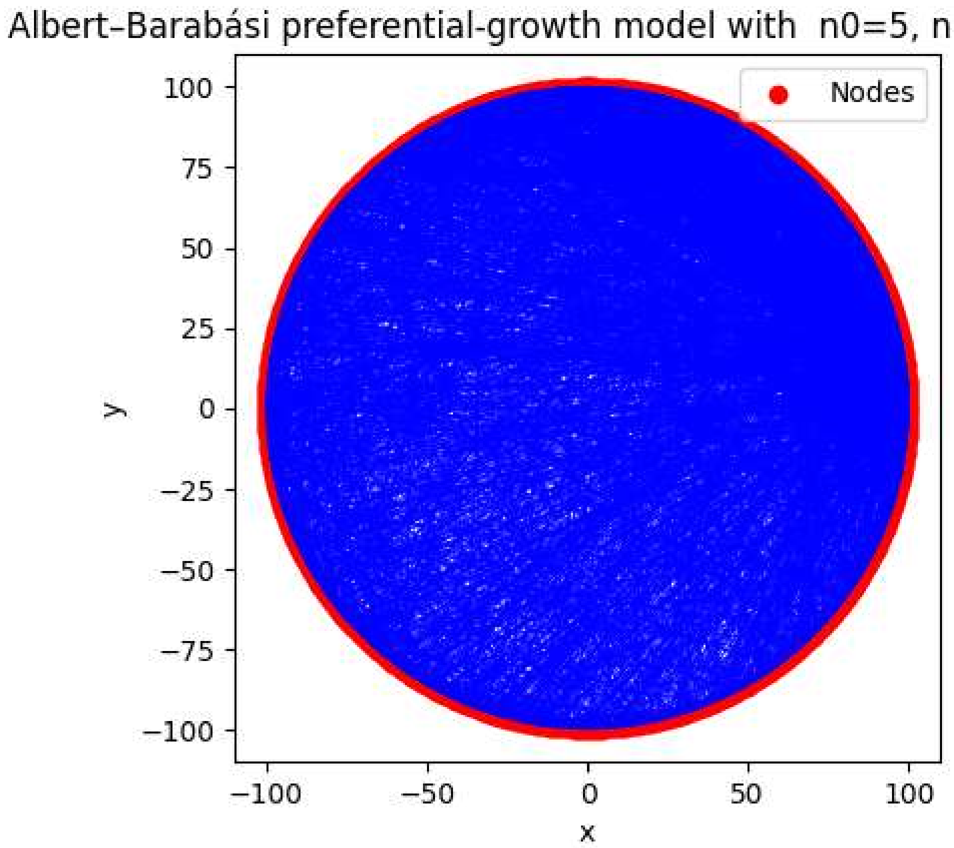


b

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import random

n0=5
n=1000
m=3
r=100

amatrix= np.ones((n0,n0))
np.fill_diagonal(amatrix, 0)

def find_degree (matrix):
    degree= np.zeros((len(matrix),1))
    for i in range(len(matrix)):
        degree[i,:] = np.sum(matrix[i,:])
    total_degree = np.sum(degree)
    cumulative_sum = np.cumsum(degree)
    prob = cumulative_sum/total_degree
    return (prob)

for i in range(1,n-n0+1):
    new_matrix = np.zeros((n0+i, n0+i))
    new_matrix[:-1,:-1] = amatrix
    new_edge =0
    previous_list=[]
    while new_edge<m:
        prob = find_degree(new_matrix)
        random_number = np.random.uniform(0,1)
        for a in range(len(prob)):
            if prob[a]>=random_number and a not in previous_list:
                # toss = np.random.uniform(0,1)
                # if toss >0.5:
                new_matrix[n0+i-1][a]=1
                # else:
                new_matrix[a][n0+i-1]=1
                new_edge+=1
                previous_list.append(a)
                break

    amatrix = new_matrix


print('new_edge', new_edge)
theta = np.linspace(0, 2*np.pi, len(amatrix))
x = r * np.cos(theta)
y = r * np.sin(theta)



plt.figure(figsize=(10, 6))

plt.subplot(1,2,1)
plt.scatter(x, y, color='red', label='Nodes')
for i in range(n):
    for j in range(n):
        if amatrix[i][j] ==1:
            plt.plot([x[i], x[j]], [y[i], y[j]], color='blue', alpha=0.5, linewidth= 0.5)
plt.gca().set_aspect('equal', adjustable='box')
plt.xlabel('x')
plt.ylabel('y')
```

```
plt.legend()
plt.title(f'Albert-Barabási preferential-growth model with  n0={n0}, n={n}, m={m}')
plt.show()
```

new_edge 3



In [ ]:
```
degree =[]
for i in range(len(amatrix)):
    x = np.sum(amatrix[i,:])
    degree.append(x)
ds = np.sort(degree)[::-1]


u= []
for i in range(1,len(amatrix)+1):
    u.append(i/n)
k_valus = x = np.logspace(0.5,2,100)
 #np.linspace(np.max(degree), np.min(degree))
def theoretical_degree_distribution(k, m):
    return (m**2) * (k**(-2))


degree_theortical = theoretical_degree_distribution(k_valus,m)
plt.scatter(ds,u)
plt.loglog(k_valus,degree_theortical)
plt.xscale('log')
plt.yscale('log')
plt.title('Inverse cumulative degree distribution for a preferential-growth graph')
plt.show()
```