In [ ]:
```
''' 1) b) Simulate one single realization of the chain, that is, starting from X0 =
the value of Xi using the probabilities defined by the process'''.
```

In [9]:
```
import numpy as np
import math
import matplotlib.pyplot as plt
```

In [22]:
```
trans_matrix = np.array([
    [0.2, 0.7, 0.1],
    [0.2, 0.5, 0.3],
    [0.2, 0.4, 0.4]
])

initial_state = 1
state_list = [initial_state]

for _ in range(20):
    initial_state = np.random.choice([1, 2, 3], p=trans_matrix[initial_state-1])
    state_list.append(initial_state)

state_list
```

Out[22]:
```
[1, 2, 2, 1, 2, 2, 3, 2, 2, 2, 3, 2, 3, 1, 2, 3, 2, 2, 3, 2, 3]
```

In [ ]:
```
```

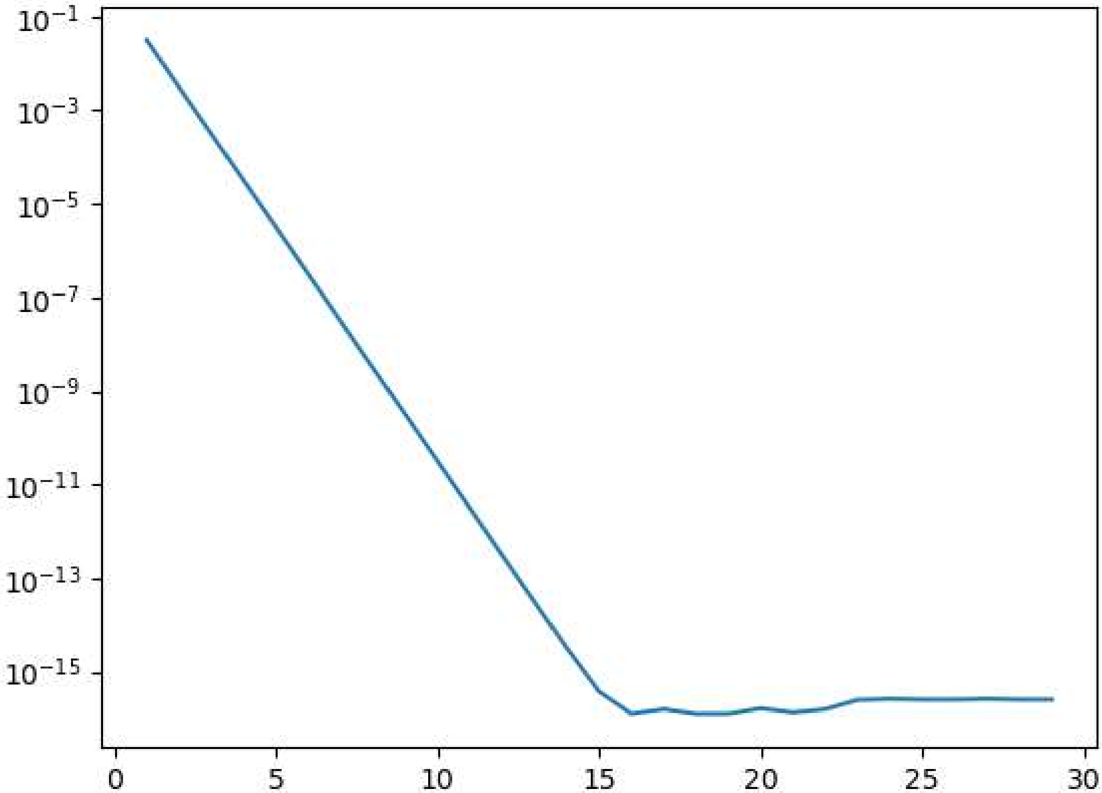
In [ ]:
```
'''2) b) Starting now from an initial probability distribution π0 on the nodes, com
0 P i the probability distribution at time i. Show that πi → π∞
1
and make plot of i vs ‖πi − π∞‖2
2. Generate this plot for at least two different
initial conditions π0 and compare.'''
```

In [25]:
```
import numpy as np
import matplotlib.pyplot as plt

target_distribution = np.array([1/5, 23/45, 13/45])
initial_distribution = np.array([1/3, 1/3, 1/3])
iteration_axis = []
distance_values = []
transpose_initial_distribution = np.transpose(initial_distribution)

for iteration_step in range(1, 30):
    iteration_axis.append(iteration_step)
    iteration_matrix = np.linalg.matrix_power(transition_matrix, iteration_step)
    current_distribution = np.dot(initial_distribution, iteration_matrix)
    distance = math.dist(current_distribution, target_distribution)
    distance_values.append(distance)

plt.plot(iteration_axis, distance_values)
plt.yscale("log")
plt.show()
```



In [ ]:

```
In [ ]:
```

```
'''3 Absorbing state. Consider now that node 3 is an absorbing state and we want to
estimate the waiting time until the process arrives at Xi = 3 from any other node.
a) Starting from each one of X0 = 1 and X0 = 2, run multiple simulation of the
Markov chain (Problem 1, part b) until Xi = 3 and store the arrival time until
this happens. Make a histogram of the arrival time for both X0 = 1 and X0 = 2
and compute the mean.
```

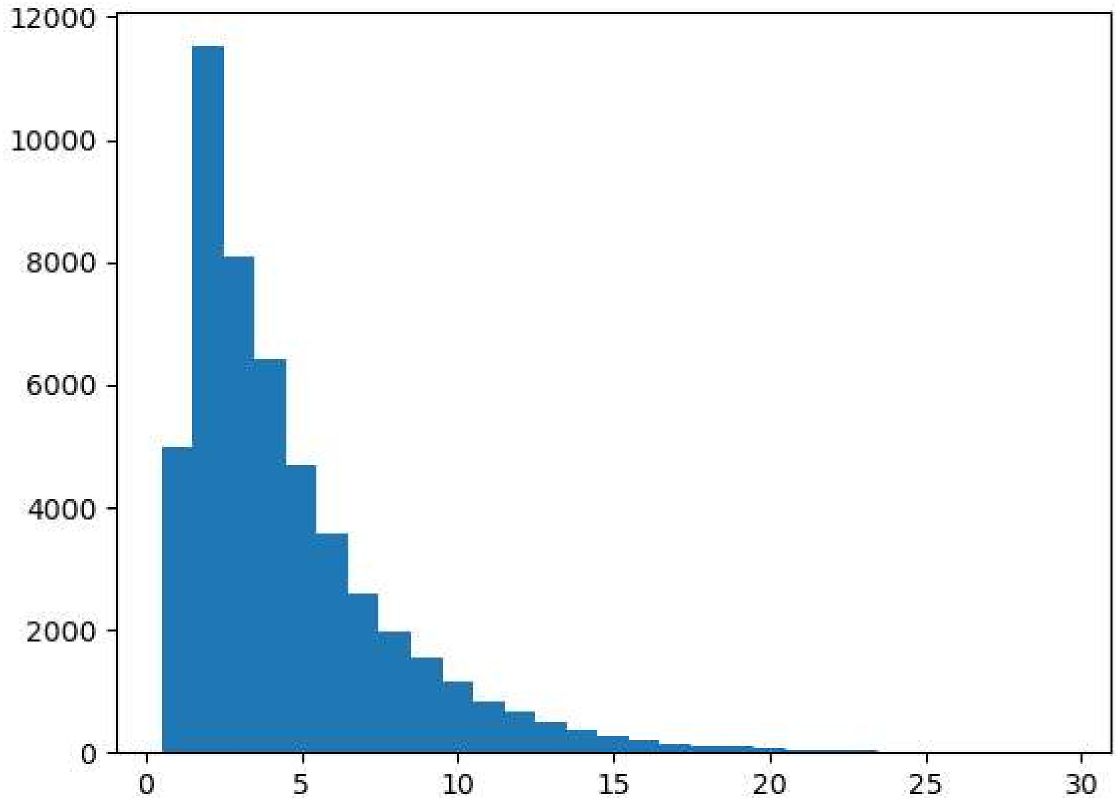
In [40]:

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

arrival_durations = []

for _ in range(1, 50000):
    current_position = 1
    position_history = []
    while current_position != 3:
        current_position = np.random.choice([1, 2, 3], p=transition_matrix[current_
        position_history.append(current_position)
    arrival_durations.append(len(position_history))

np.mean(arrival_durations)
plt.hist(arrival_durations, bins=np.arange(0.5, 30, 1))
plt.show()
np.mean(arrival_durations)
```



Out[40]: 4.617032340646813

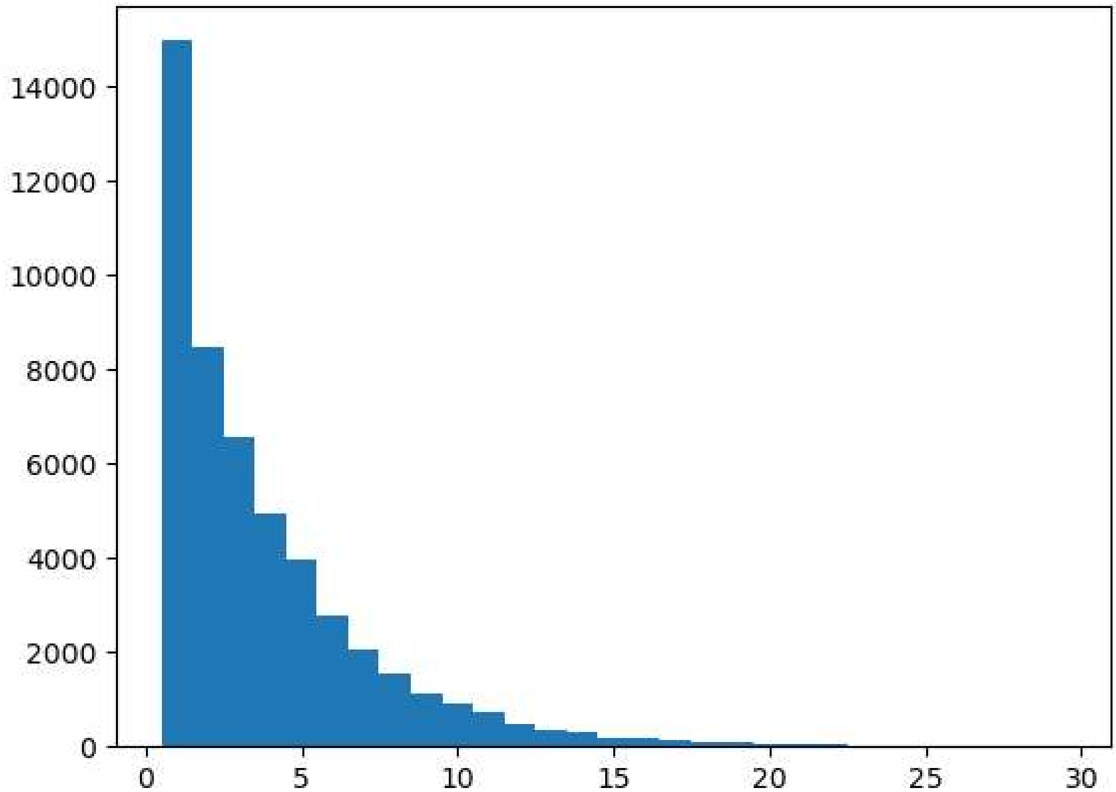
In [ ]:

In [41]:
```
import numpy as np
import matplotlib.pyplot as plt

arrival_durations = []

for _ in range(1, 50000):
    current_position = 2
    position_history = []
    while current_position != 3:
        current_position = np.random.choice([1, 2, 3], p=transition_matrix[current_
        position_history.append(current_position)
    arrival_durations.append(len(position_history))

np.mean(arrival_durations)
plt.hist(arrival_durations, bins=np.arange(0.5, 30, 1))
plt.show()
np.mean(arrival_durations)
```



Out[41]: 3.847076941538831

In [ ]:

In [ ]:

In [ ]: