In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
```

In [25]:
```
# Load Data
NF = np.genfromtxt('NF.csv', delimiter=',')
FF = np.genfromtxt('FF.csv', delimiter=',')
FFD = np.genfromtxt('FFD.csv', delimiter=',')
BF = np.genfromtxt('BF.csv', delimiter=',')
BFD = np.genfromtxt('BFD.csv', delimiter=',')
```

In [3]:
```
# Print out one set of data for example
print(NF)
```

```
[[3.20000e+01 3.30000e+00]
 [6.40000e+01 5.36000e+00]
 [1.28000e+02 1.12200e+01]
 [2.56000e+02 2.25000e+01]
 [5.12000e+02 4.86900e+01]
 [1.02400e+03 9.59900e+01]
 [2.04800e+03 1.91110e+02]
 [4.09600e+03 3.75340e+02]
 [8.19200e+03 7.51060e+02]
 [1.63840e+04 1.49541e+03]
 [3.27680e+04 3.01292e+03]
 [6.55360e+04 6.01908e+03]
 [1.31072e+05 1.20374e+04]
 [2.62144e+05 2.41060e+04]
 [5.24288e+05 4.83674e+04]]
```

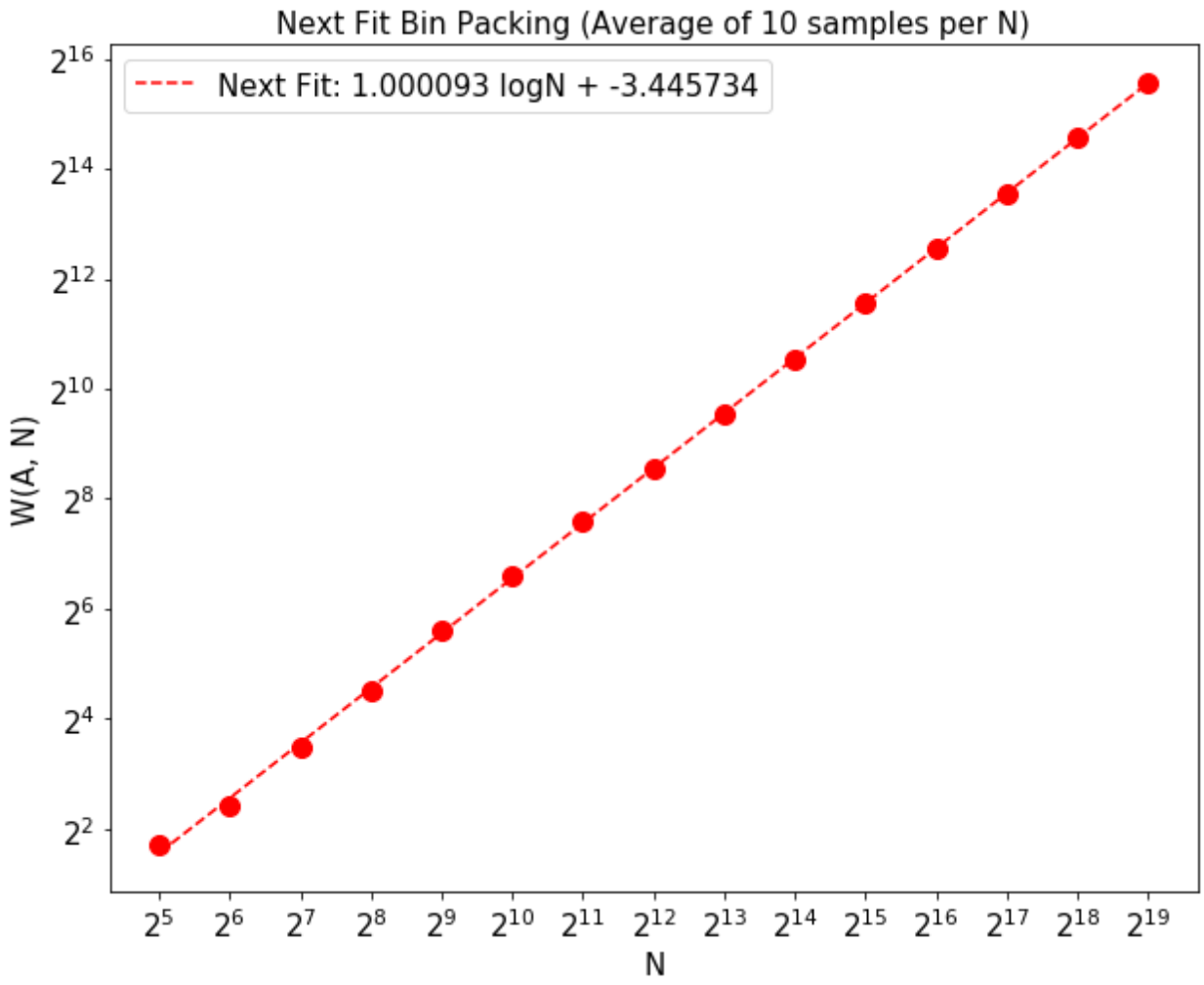
The first columne represents the input size of the data, and the second colume represents the waste of the bin packing algorithm.

Brief Introduction

All of the data above are generated through inputting a list of random double number from a uniform distribution with the value between 0.0 to 0.7. The input sizes are from 2^5 to 2^19. For each algorithm, I conducted ten trials of experiments, which include generation of random double number and running the algorithm, and eventually averaged the waste of the ten results. With these data, I was able to plot them on the following log-log plots. However, there were some issues and findings after I plotted these data. I will address these issues and findings after the plots.

In [4]:
```
# Plotting Next Fit
f, ax = plt.subplots(1,1, figsize=(10,8))


m1, b1 = np.polyfit(np.log2(NF[:,0]), np.log2(NF[:,1]), 1)

plt.loglog(NF[:,0], NF[:,1], '.', basex=2, basey=2,
           markersize=20, color='red')

plt.loglog(NF[:,0], 2**(m1*np.log2(NF[:,0])+b1), basex=2, basey=2,
          linestyle='--', label=f'Next Fit: {m1:4f} logN + {b1:4f}', colo
r='red')

plt.xlabel('N', fontsize=15)
```

```
plt.ylabel('W(A, N)', fontsize=15)
plt.xticks(NF[:,0], fontsize=15)
plt.yticks(fontsize=15)
plt.title('Next Fit Bin Packing (Average of 10 samples per N)', fontsize=
15)
plt.legend(fontsize=15)
plt.show()
```

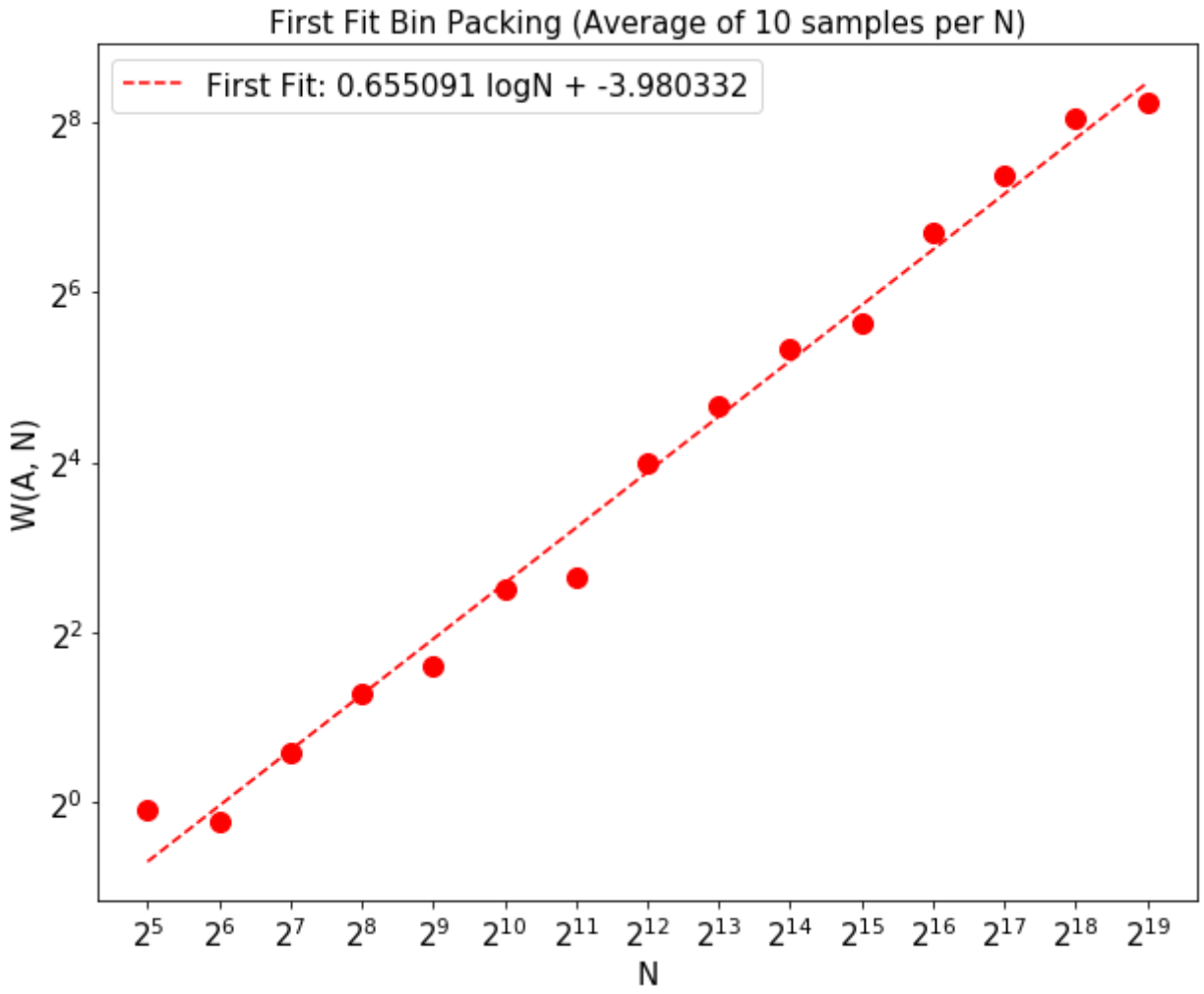


Comments:

The slope of the asymptotic best-fit line to Next Fit bin packing algorithm is about 1. The waste as a function of n is W(A)=O(N).

In [12]:
```
# Plotting First Fit
f, ax = plt.subplots(1,1, figsize=(10,8))


m1, b1 = np.polyfit(np.log2(FF[:,0]), np.log2(FF[:,1]), 1)
```

```
plt.loglog(FF[:,0], FF[:,1], '.', basex=2, basey=2,
           markersize=20, color='red')

plt.loglog(FF[:,0], 2**(m1*np.log2(FF[:,0])+b1), basex=2, basey=2,
          linestyle='--', label=f'First Fit: {m1:4f} logN + {b1:4f}', col
or='red')

plt.xlabel('N', fontsize=15)
plt.ylabel('W(A, N)', fontsize=15)
plt.xticks(FF[:,0], fontsize=15)
plt.yticks(fontsize=15)
plt.title('First Fit Bin Packing (Average of 10 samples per N)', fontsize
=15)
plt.legend(fontsize=15)
plt.show()
```

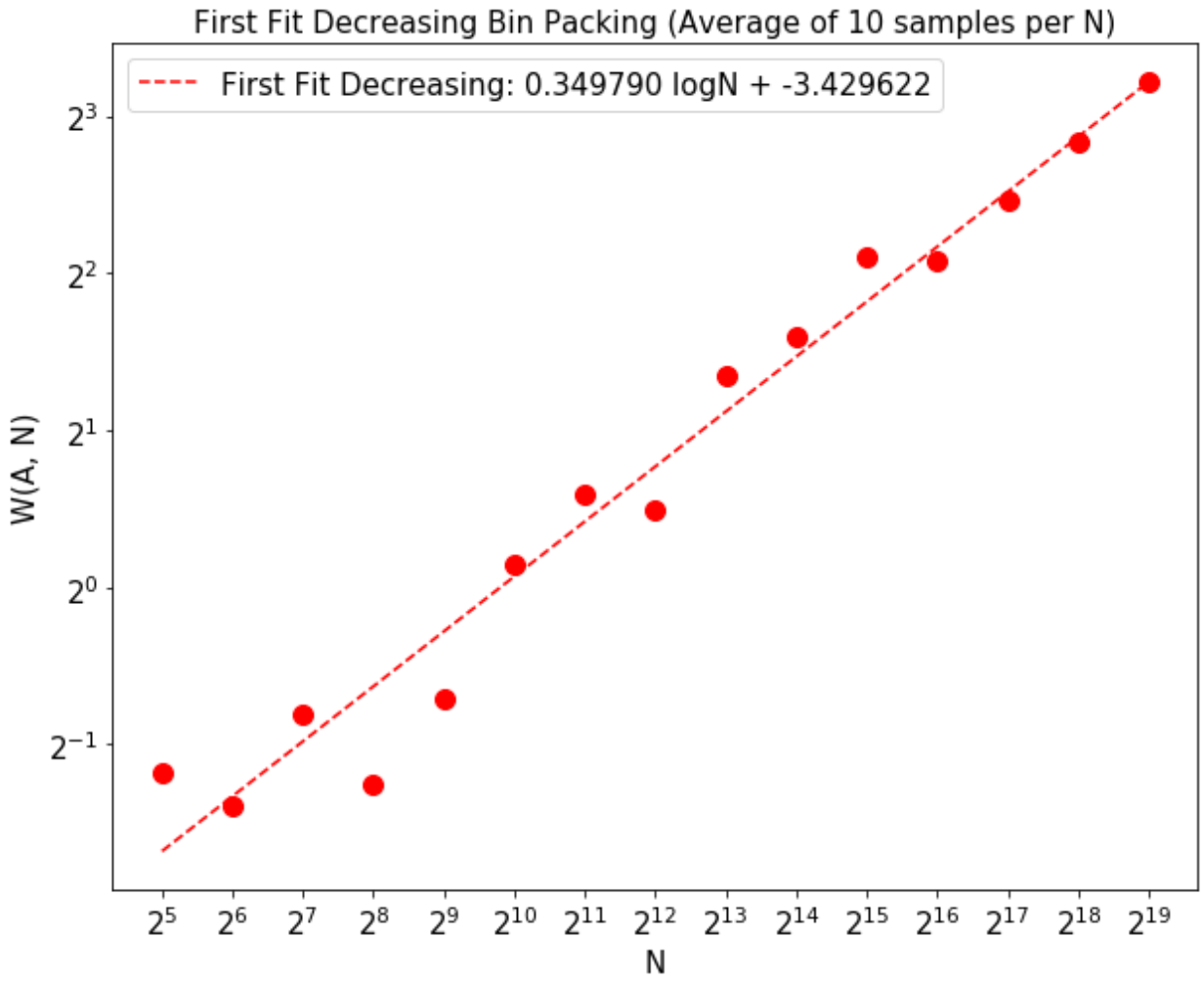


Comments:

The slope of the asymptotic best-fit line to First Fit bin packing algorithm is about 0.655. The waste as a function of n is W(A)=O(N^0.655) which is about O(N^(2/3)).

In [16]:
```
# Plotting First Fit Decreasing
f, ax = plt.subplots(1,1, figsize=(10,8))
```

```
m1, b1 = np.polyfit(np.log2(FFD[:,0]), np.log2(FFD[:,1]), 1)

plt.loglog(FFD[:,0], FFD[:,1], '.', basex=2, basey=2,
           markersize=20, color='red')

plt.loglog(FFD[:,0], 2**(m1*np.log2(FFD[:,0])+b1), basex=2, basey=2,
          linestyle='--', label=f'First Fit Decreasing: {m1:4f} logN + {b
1:4f}', color='red')

plt.xlabel('N', fontsize=15)
plt.ylabel('W(A, N)', fontsize=15)
plt.xticks(FFD[:,0], fontsize=15)
plt.yticks(fontsize=15)
plt.title('First Fit Decreasing Bin Packing (Average of 10 samples per
 N)', fontsize=15)
plt.legend(fontsize=15)
plt.show()
```

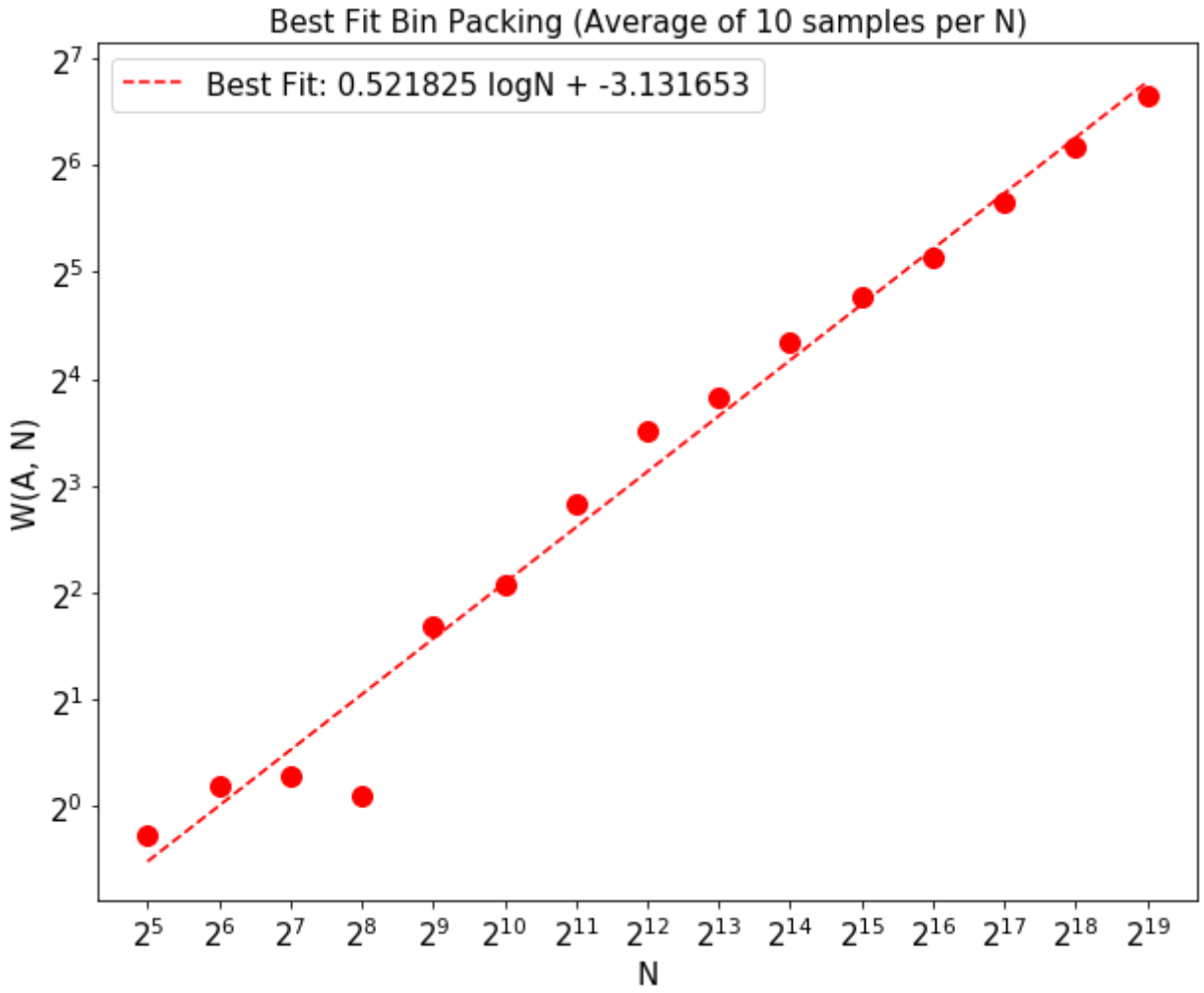


Comments:

The slope of the asymptotic best-fit line to First Fit Decreasing bin packing algorithm is about 0.35. The waste as a function of n is W(A)=O(N^0.35) which is about O(N^(1/3)).

In [23]:
```
# Plotting Best Fit
f, ax = plt.subplots(1,1, figsize=(10,8))
```

```
m1, b1 = np.polyfit(np.log2(BF[:,0]), np.log2(BF[:,1]), 1)

plt.loglog(BF[:,0], BF[:,1], '.', basex=2, basey=2,
           markersize=20, color='red')

plt.loglog(BF[:,0], 2**(m1*np.log2(BF[:,0])+b1), basex=2, basey=2,
          linestyle='--', label=f'Best Fit: {m1:4f} logN + {b1:4f}', colo
r='red')

plt.xlabel('N', fontsize=15)
plt.ylabel('W(A, N)', fontsize=15)
plt.xticks(BF[:,0], fontsize=15)
plt.yticks(fontsize=15)
plt.title('Best Fit Bin Packing (Average of 10 samples per N)', fontsize=
15)
plt.legend(fontsize=15)
plt.show()
```

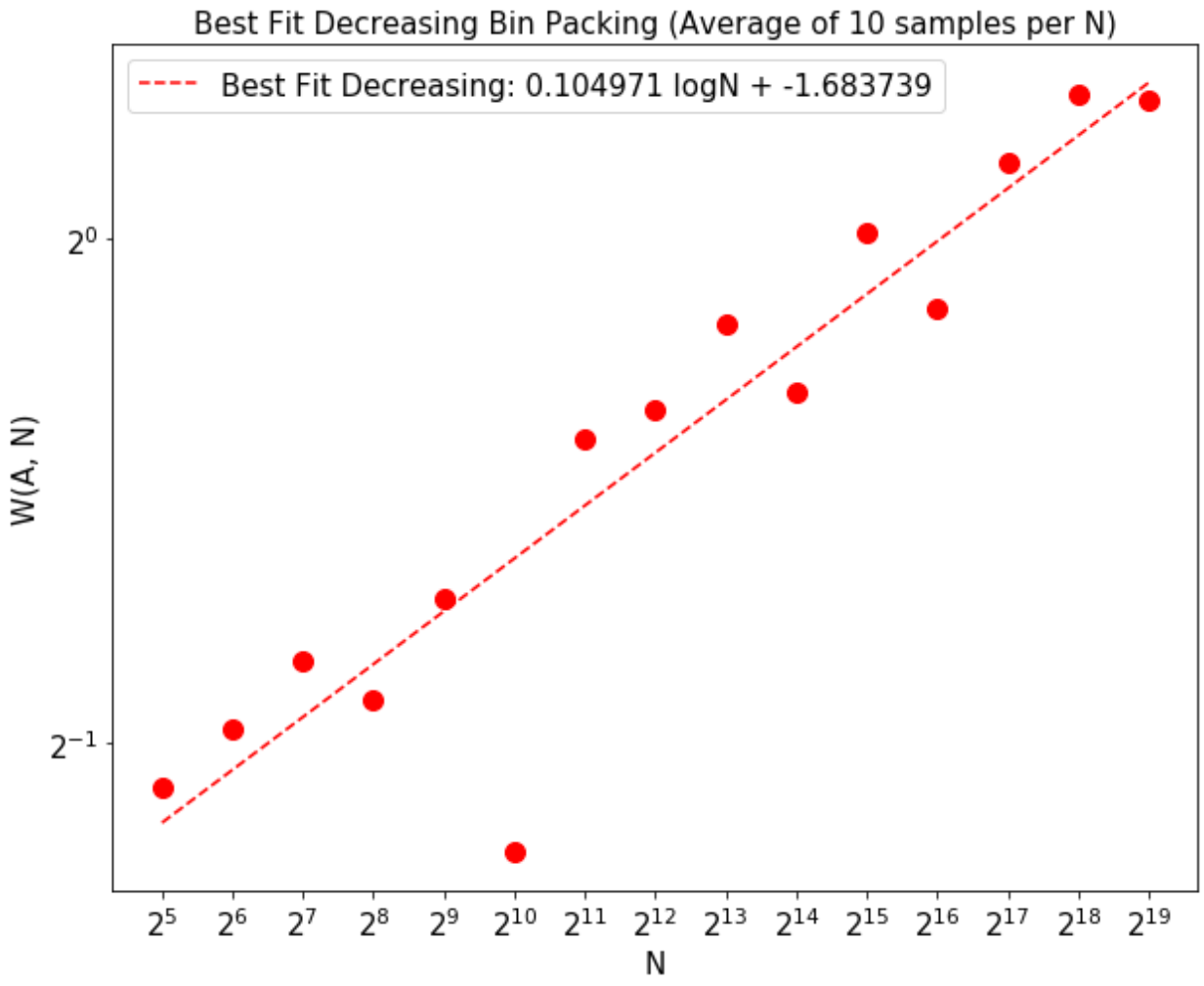


Comments:

The slope of the asymptotic best-fit line to Best Fit bin packing algorithm is about 0.522. The waste as a function of n is W(A)=O(N^0.522) which is about O(N^(1/2)).

In [26]:
```
# Plotting Best Fit Decreasing
f, ax = plt.subplots(1,1, figsize=(10,8))
```

```
m1, b1 = np.polyfit(np.log2(BFD[:,0]), np.log2(BFD[:,1]), 1)

plt.loglog(BFD[:,0], BFD[:,1], '.', basex=2, basey=2,
           markersize=20, color='red')

plt.loglog(BFD[:,0], 2**(m1*np.log2(BFD[:,0])+b1), basex=2, basey=2,
          linestyle='--', label=f'Best Fit Decreasing: {m1:4f} logN + {b
1:4f}', color='red')

plt.xlabel('N', fontsize=15)
plt.ylabel('W(A, N)', fontsize=15)
plt.xticks(BFD[:,0], fontsize=15)
plt.yticks(fontsize=15)
plt.title('Best Fit Decreasing Bin Packing (Average of 10 samples per N)'
, fontsize=15)
plt.legend(fontsize=15)
plt.show()
```



Comments:

The slope of the asymptotic best-fit line to Best Fit Decreasing bin packing algorithm is about 0.1. The waste as a function of n is W(A)=O(N^0.1).

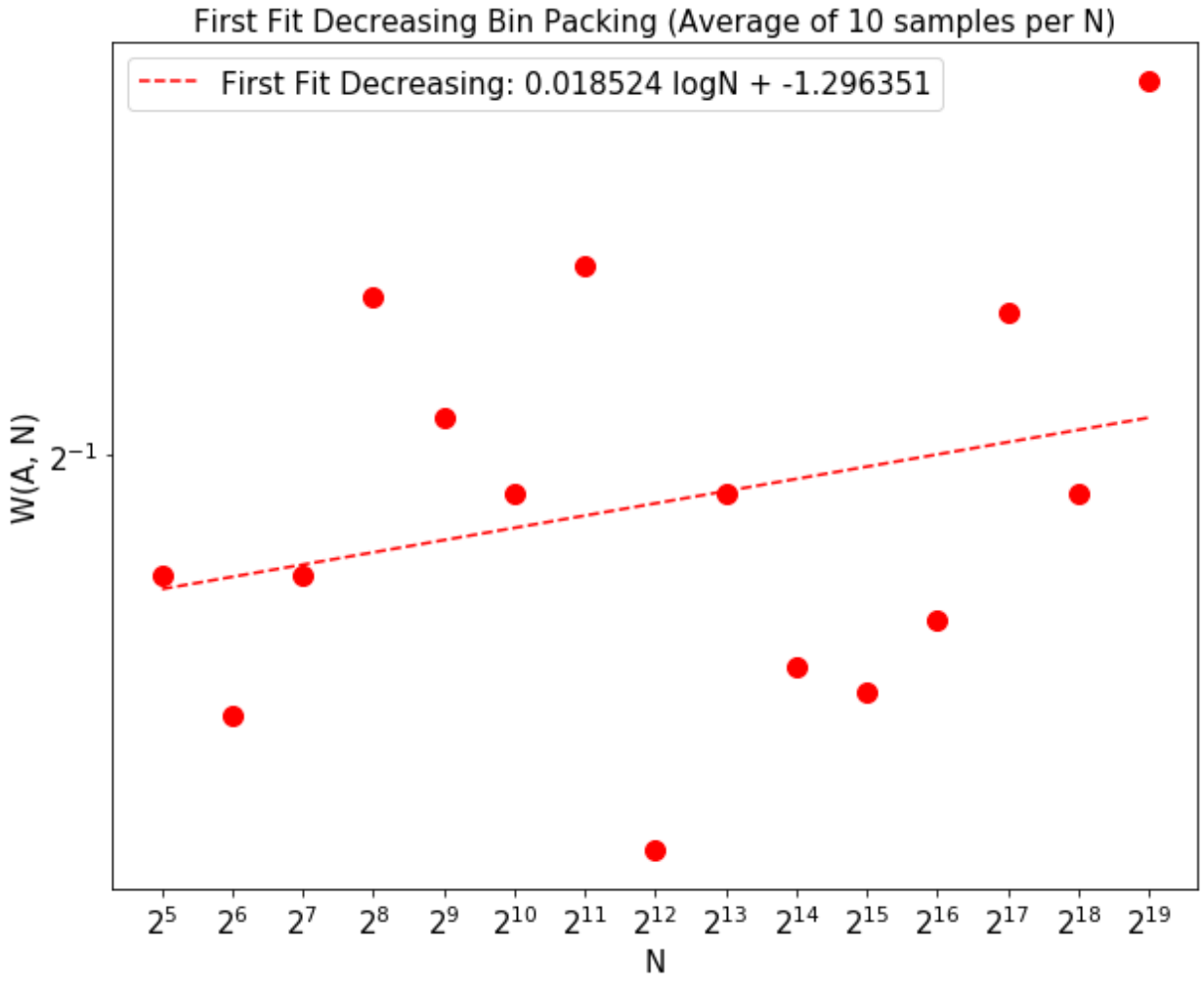
Issues and Findings:

The issues are that I found the wastes of FFD and BFD are a bit unstable (which means the wastes are sometimes low and sometimes high for the

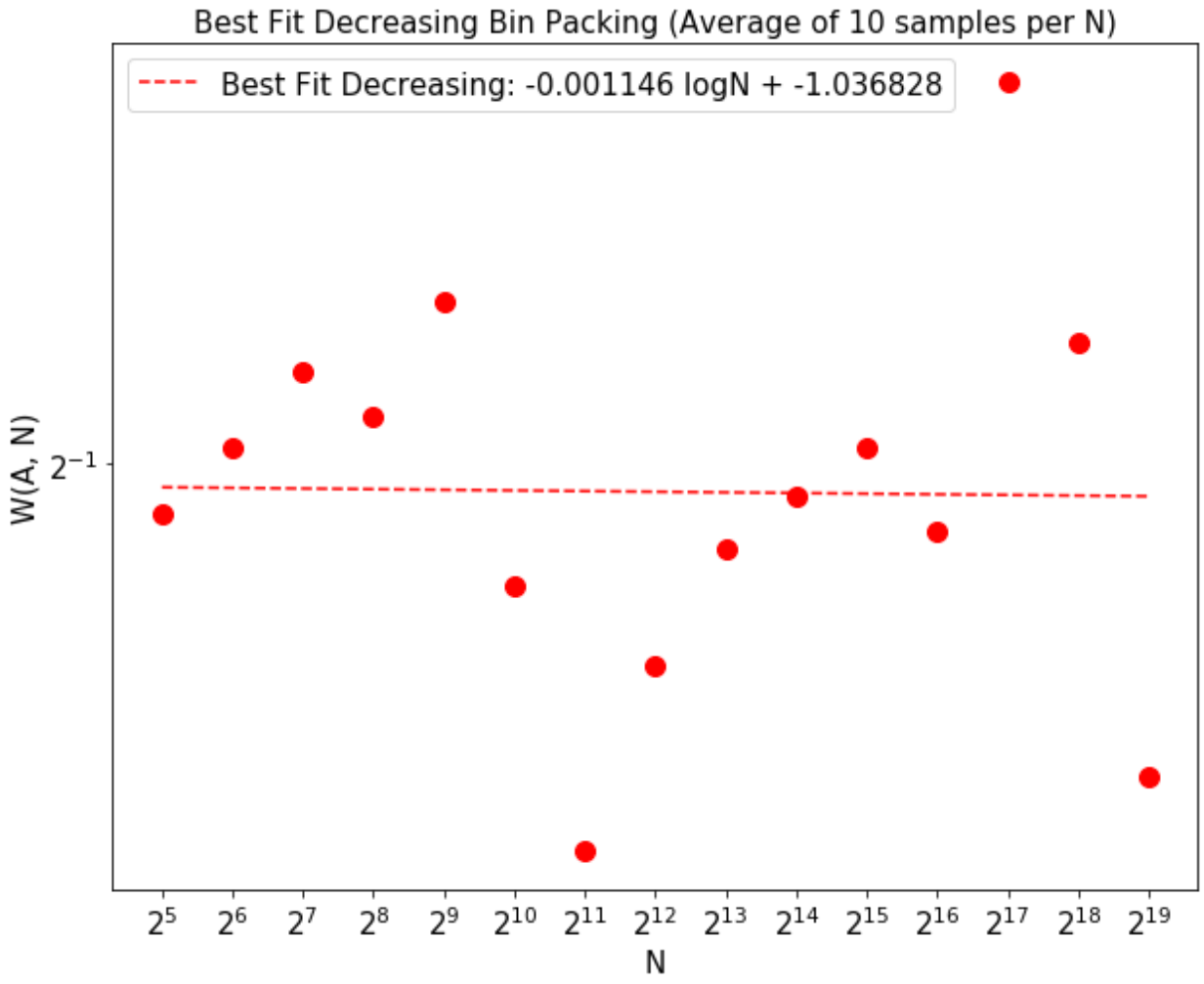
same input size), especially for small input size. I then tried to print out the input items of each trial and found that when most of the items are in the range between 0.0 to 0.5, the waste would be low and not be influenced much by input size. I then proved my hypothesis by restricting the number between 0.0 to 0.5 and plotting the results.

In [31]:
```
# Load Data
FFD_new = np.genfromtxt('FFD_new.csv', delimiter=',')
BFD_new = np.genfromtxt('BFD_new.csv', delimiter=',')
```

In [33]:
```
# Plotting First Fit Decreasing
f, ax = plt.subplots(1,1, figsize=(10,8))


m1, b1 = np.polyfit(np.log2(FFD_new[:,0]), np.log2(FFD_new[:,1]), 1)

plt.loglog(FFD_new[:,0], FFD_new[:,1], '.', basex=2, basey=2,
           markersize=20, color='red')

plt.loglog(FFD_new[:,0], 2**(m1*np.log2(FFD_new[:,0])+b1), basex=2, basey
=2,
          linestyle='--', label=f'First Fit Decreasing: {m1:4f} logN + {b
1:4f}', color='red')

plt.xlabel('N', fontsize=15)
plt.ylabel('W(A, N)', fontsize=15)
plt.xticks(FFD_new[:,0], fontsize=15)
plt.yticks(fontsize=15)
plt.title('First Fit Decreasing Bin Packing (Average of 10 samples per
 N)', fontsize=15)
plt.legend(fontsize=15)
plt.show()
```



In [34]:
```
# Plotting Best Fit Decreasing
f, ax = plt.subplots(1,1, figsize=(10,8))


m1, b1 = np.polyfit(np.log2(BFD_new[:,0]), np.log2(BFD_new[:,1]), 1)

plt.loglog(BFD_new[:,0], BFD_new[:,1], '.', basex=2, basey=2,
           markersize=20, color='red')

plt.loglog(BFD_new[:,0], 2**(m1*np.log2(BFD_new[:,0])+b1), basex=2, basey
=2,
          linestyle='--', label=f'Best Fit Decreasing: {m1:4f} logN + {b
1:4f}', color='red')

plt.xlabel('N', fontsize=15)
plt.ylabel('W(A, N)', fontsize=15)
plt.xticks(BFD_new[:,0], fontsize=15)
plt.yticks(fontsize=15)
plt.title('Best Fit Decreasing Bin Packing (Average of 10 samples per N)'
, fontsize=15)
plt.legend(fontsize=15)
plt.show()
```



Comments:

From these plots, we can see that the slopes of the asymptotic best-fit line are about 0 for both algorithms when I restricted the values of the number between 0.0 to 0.5. This means that the waste as a function of n is W(A)=O(1) in this case.

Conclusion:

Overall, among all these bin packing algorithms, I would say Best Fit Decreasing is the best. This is because Best Fit Decreasing algorithm has the lowest slopes of the asymoptotic best-fit line. This means that the waste will grow slowest among these algorithms. In addition to the slope, looking at the actual value of the waste, BFD has only about 1.2 waste for input size of 2^19. This means that this algorithm can really maximize the usage of the free spaces of the bins. We can also see that sorting the items in a descending order before bin packing can greatly reduce the waste comparing to not sorting the items. However, this can only be done in off-line bin packing, which means we know what the items are before the process of the algorithm. For online bin packing, sorting the items are not feasible. Moreover, although Next Fit algorithm is the worst one considering waste, it has the advantage of saving memory spaces and the best run time.

In [ ]: