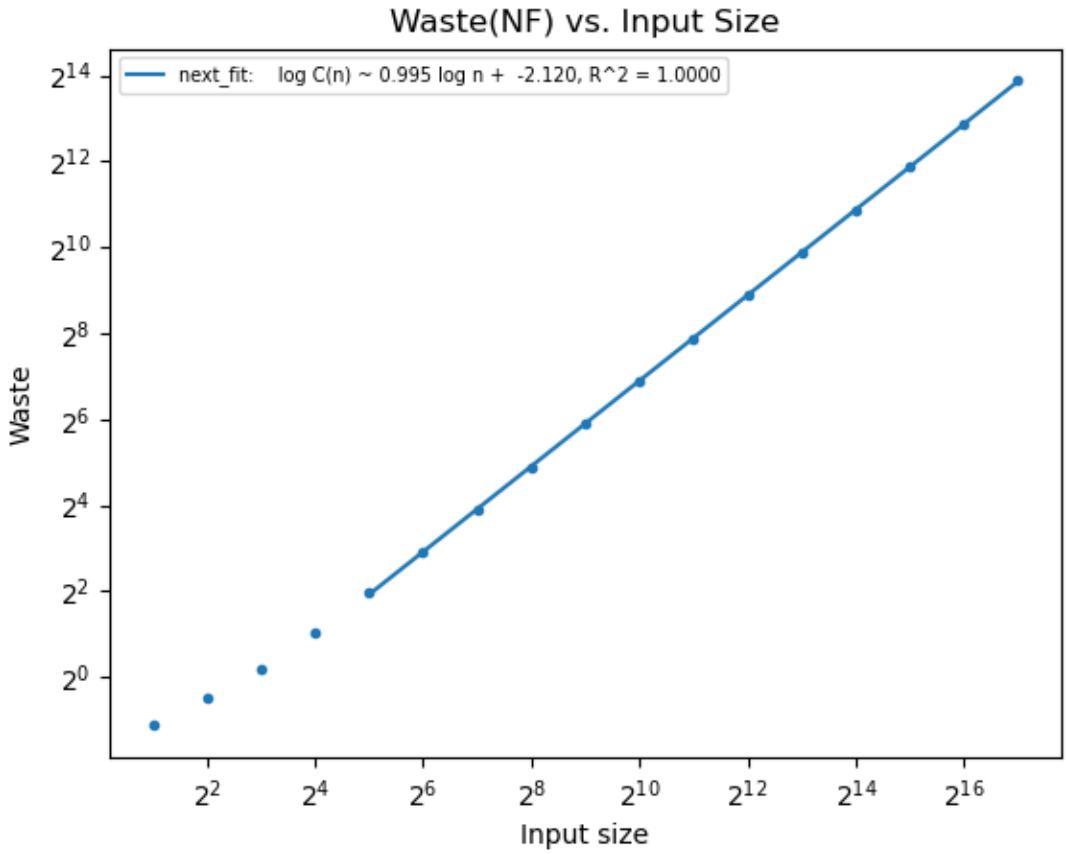
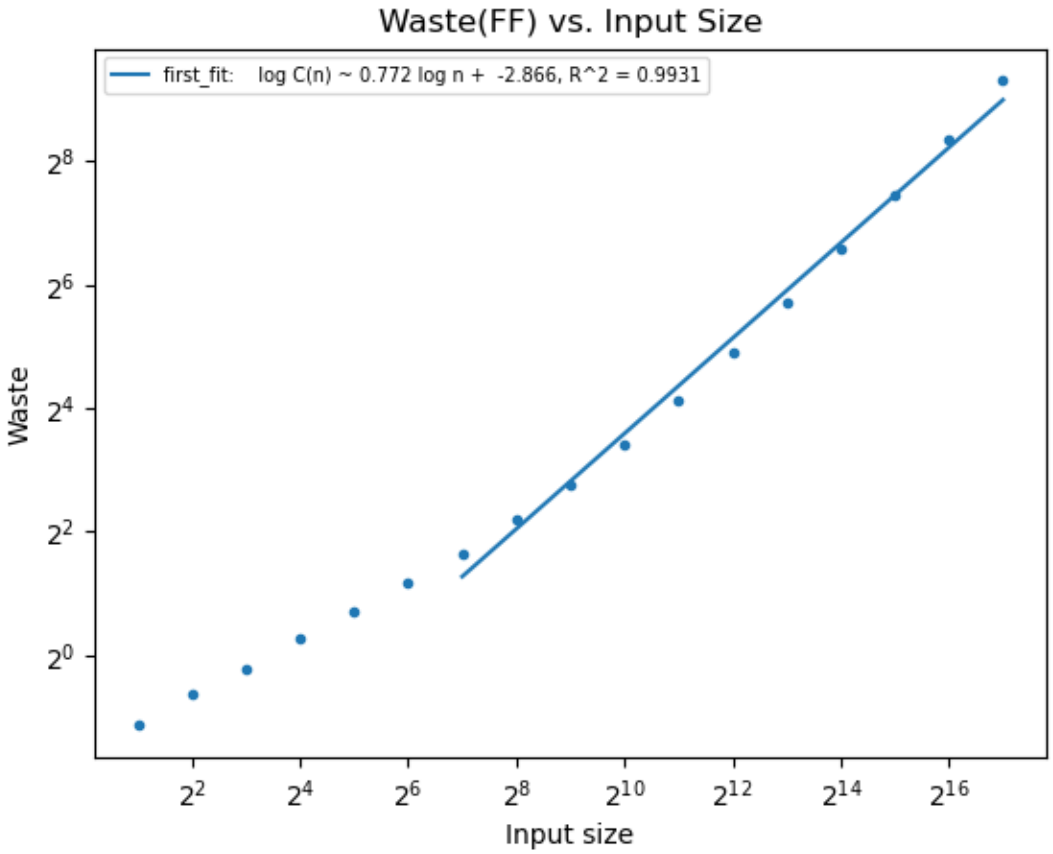
Plotting the waste for the NF algorithm on a log-log plot and determining the slope of an asymptotic best-fit line to determine its waste experimentally as a function of n.



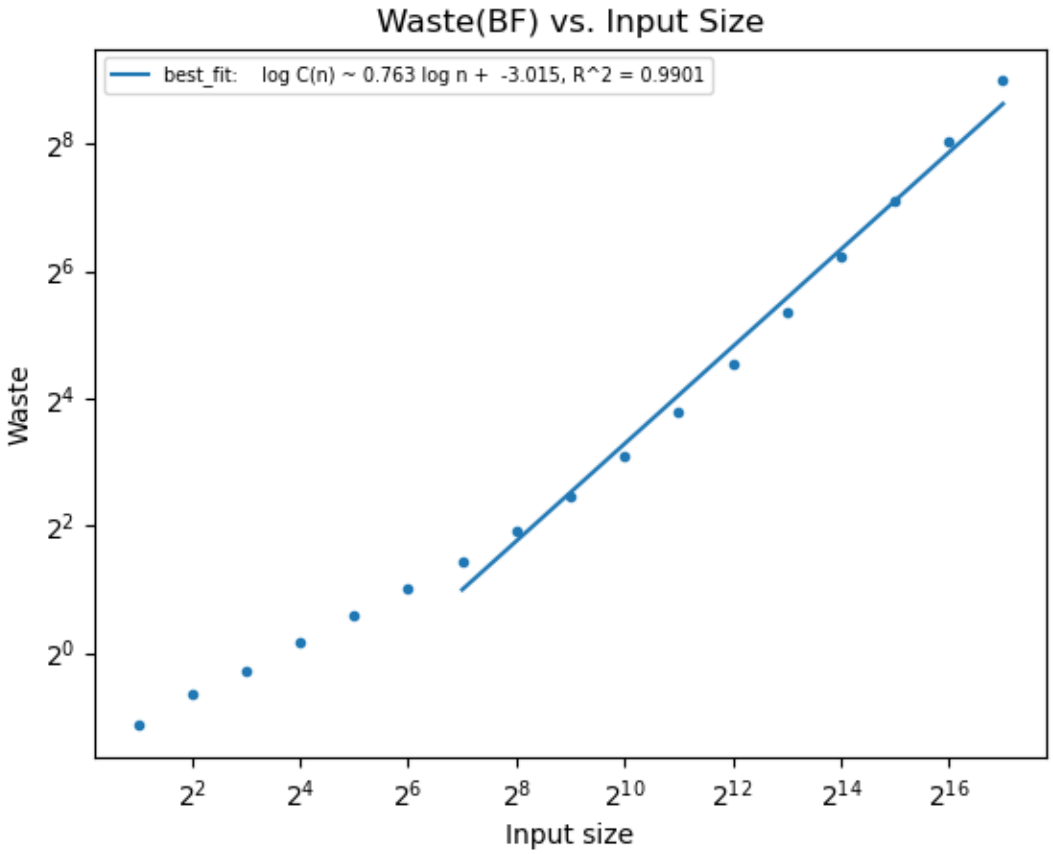
The runtime complexity of the next-fit algorithm is O(n), which has the shortest running time among the rest of the experimental algorithms. However, it has the worst waste since it only cares about the current bin rather than previous non-fully filled bins.

Plotting the waste for the FF algorithm on a log-log plot and determining the slope of an asymptotic best-fit line to determine its waste experimentally as a function of n.



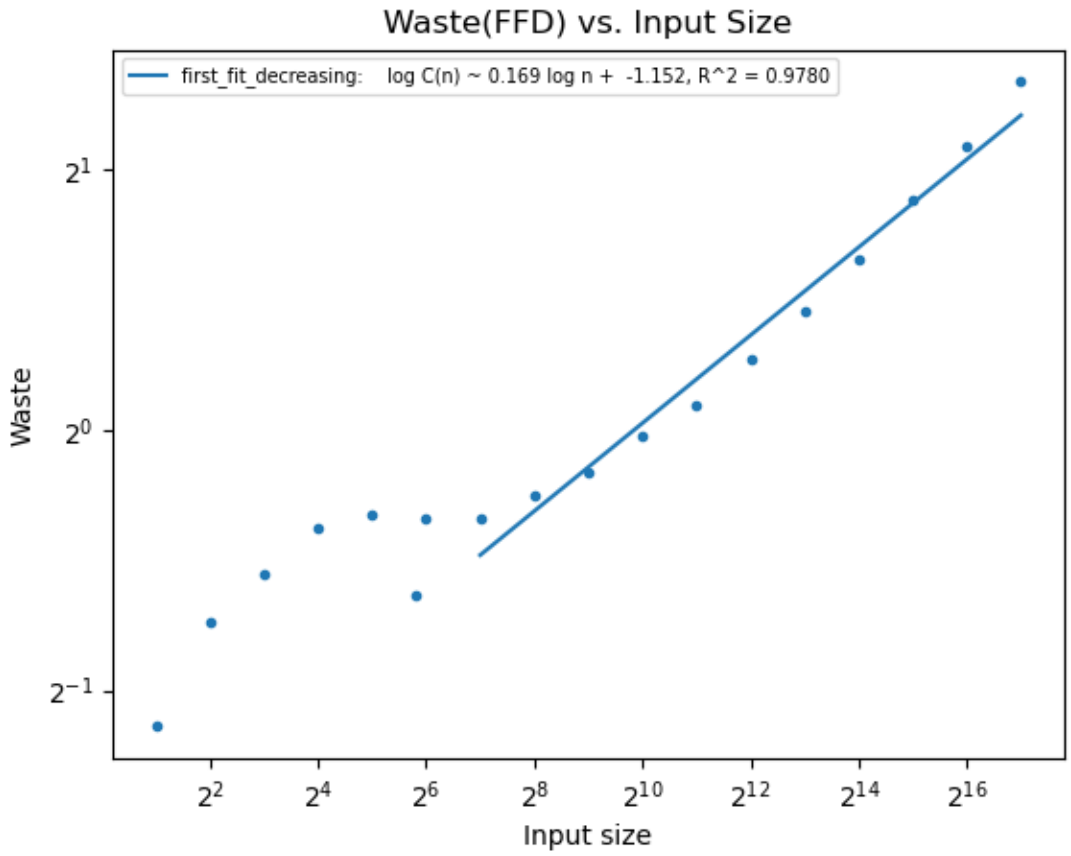
The runtime complexity of the first-fit algorithm is O(nlog(n)) by using the Zipzip Tree. Although the running time cannot catch up with the next-fit algorithm, it has better performance than the next-fit algorithm in terms of waste (please see the figure on page 6 for reference).

Plotting the waste for the BF algorithm on a log-log plot and determining the slope of an asymptotic best-fit line to determine its waste experimentally as a function of n.



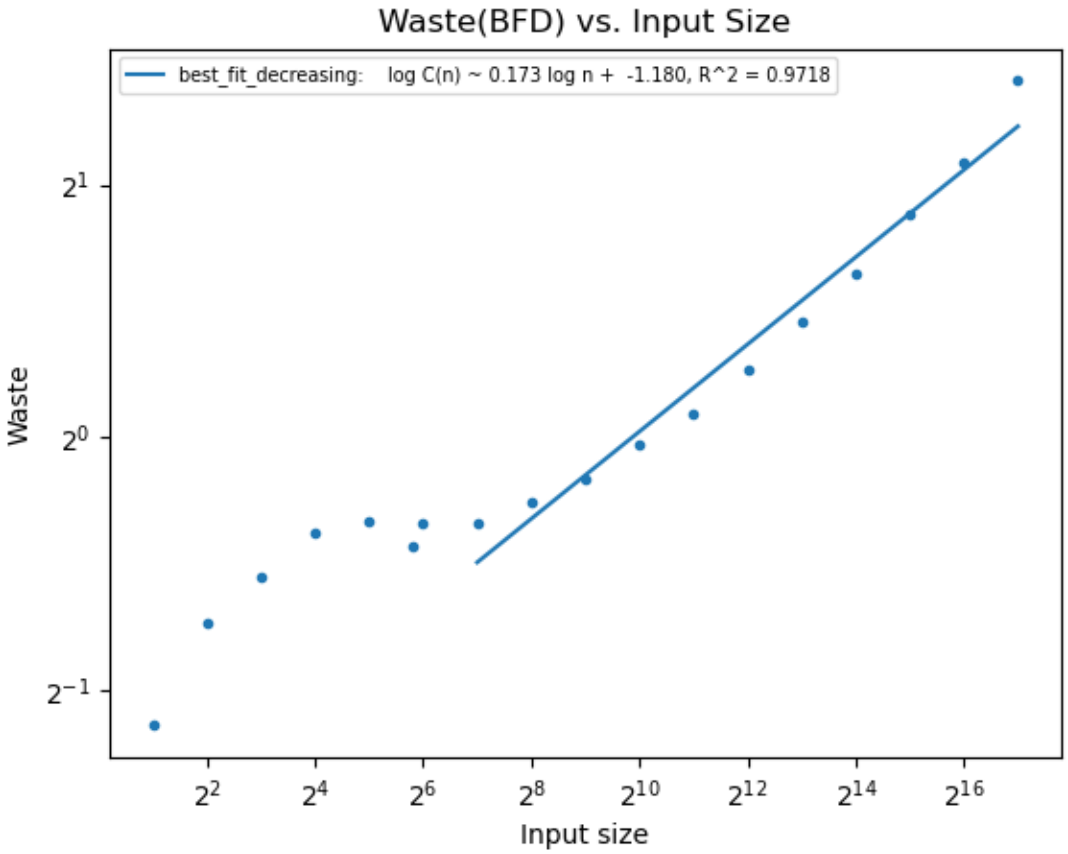
The runtime complexity of the best-fit algorithm is O(nlog(n)) by using the Zipzip Tree. Although the running time cannot catch up with the next-fit algorithm, it has better performance than the next-fit algorithm in terms of waste. Also, through the experiment, we know that it has a slightly better performance than the first-fit algorithm in terms of waste (please see the figure on page 6 for reference).

Plotting the waste for the FFD algorithm on a log-log plot and determining the slope of an asymptotic best-fit line to determine its waste experimentally as a function of n.



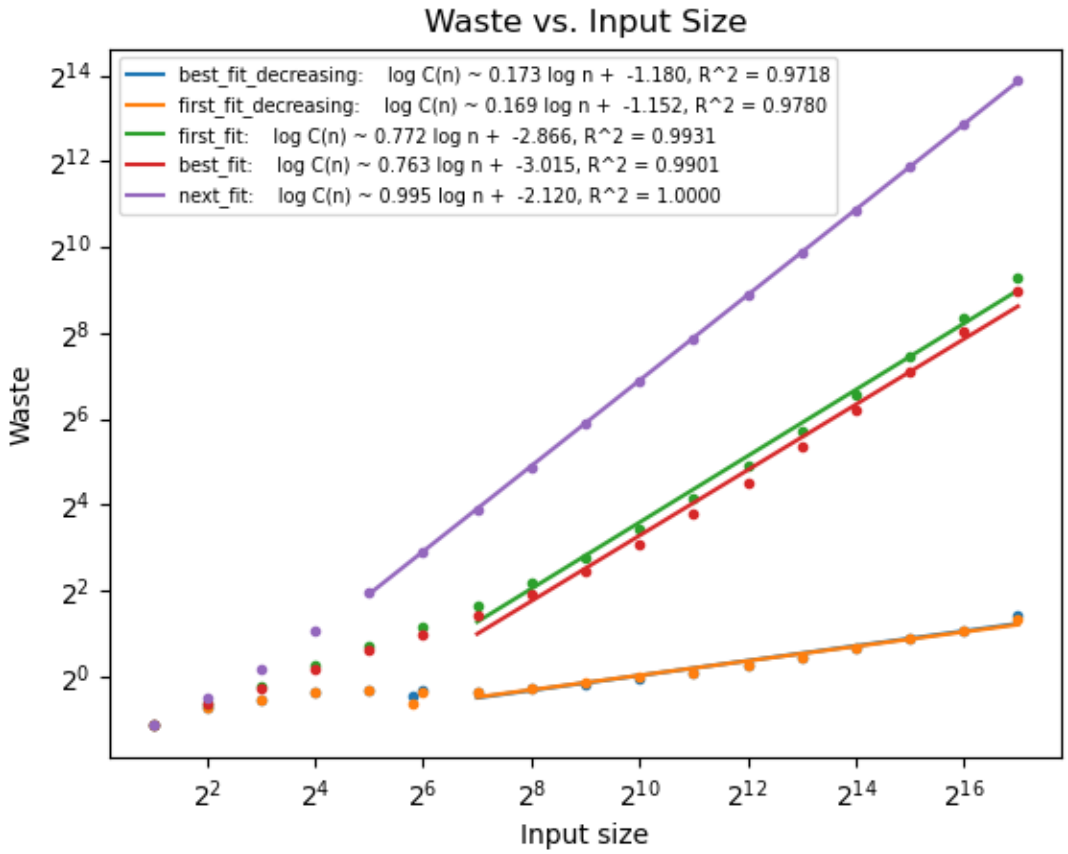
The runtime complexity of the first-fit-decreasing algorithm is O(nlog(n)) by using the Zipzip Tree. Its time complexity is not better than the next-fit, first-fit, and best-fit algorithms since it required an extra step to sort the input in decreasing order. However, it has the best performance on waste among the rest of the experimental algorithms (please see the figure on page 6 for reference). To waste performance, the first-fit-decreasing algorithm has the same performance as the best-fit decreasing order (please see the figure on page 6 for reference).

Plotting the waste for the BFD algorithm on a log-log plot and determining the slope of an asymptotic best-fit line to determine its waste experimentally as a function of n.



The runtime complexity of the best-fit-decreasing algorithm is O(nlog(n)) by using the Zipzip Tree. Its time complexity is not better than the next-fit, first-fit, and best-fit algorithms since it required an extra step to sort the input in decreasing order. However, it has the best performance on waste among the rest of the experimental algorithms (please see the figure on page 6 for reference). To waste performance, the best-fit-decreasing algorithm has the same performance as the first-fit decreasing order (please see the figure on page 6 for reference).

Identifying the algorithm you think is best. Explain why you think this algorithm produces the least amount of waste of all the algorithms you studied.



The best-fit-decreasing algorithm is the best for waste efficiency.

The best-fit-decreasing and first-fit-decreasing algorithms produce the least amount of waste according to the graph. If we want to choose the best one among these two, the best-fit-decreasing algorithm will be a good option.

If we compare the first-fit and best-fit algorithms, the best-fit algorithm seems to save more waste than the first-fit algorithm. This is due to the insertion since the best-fit algorithm inserts each item to its best fit position and checks each bin for its minimum remaining capacity that the current item fits. These processes will take more runtime than the first fit algorithms but save more waste.

Using decreasing order helps matching with complementary pairs easier. E.g. item 0.5 can match item 0.5 earlier as a bin of 1.0 instead of matching item 0.5 with item 0.4 as a bin of 0.9 first, but not existing an item 0.1 in later items, which causes 0.1 space waste.

From the above two points, the advantages of the best-fit algorithm and using decreasing order can make best-fit-decreasing the best for waste efficiency.