Hongji Yan

Algorithms implemented

1. Next Fit(NF)

   Next Fit is the most basic algorithm to solve the bin-packing problem. It will check if the current item can fit into the bin. If so, item will be placed in that bin. If not, the algorithm will start a new bin. When new item comes next time when algorithm is called, it will starts searching from where it left off, not from where it started.

2. First Fit(FF)

   The first fit is an advanced algorithm for the bin-packing problem. Unlike Next fit, The First fit algorithm always searches for the first bin that large enough to place the item, which means the First fit algorithm will only start a new bin if there's no space to hold the current item in previous bins. In order to implement the algorithm in O(NlogN) running time, I'm using the Ziptree as a helper data structure, I set the key to be the index of each bin, and ziptree node value to be the BRC(biggest remaining capacity), and the RC (remaining capacity of the current bin) Since Ziptree inherits the feature of binary search, So whenever I want to find the first bin that can fit the current item, I first look at the root of zip tree if the BRC is not enough to hold a current item, then the algorithm will simply start a new bin. If the BRC is large enough I will then look that the left subtree can hold. If so, I go to the left subtree, if not, I check if current bin RC fits current item, if so, return the current bin index. If not, I go to the right subtree and repeat this process to find the optimal first bin. This prevents looping over each previous bin which will result in O(N^2) running time for each new item.

3. First Fit Decreasing (FF)

   First Fit Decreasing works the same way as First Fit, except that First Fit Decreasing will first rearrange all items from largest to smallest by item size.

4. Best Fit(BF)

   The best fit is more advanced algorithm for the bin-packing problem. Unlike First fit, The Best fit algorithm always searches for the bin that fits tightest to place the item instead of searching for first bin that can fit the item. In order to implement the algorithm in O(NlogN) running time, I'm using the Ziptree as a helper data structure, I set the key to be the RC (remaining capacity of the current bin) and bin index, and the Ziptree node value to be the BRC(biggest remaining capacity. Since Ziptree inherits the feature of binary search, , So whenever I want to find the bin that can fit the current item tightest, I first look at the root of zip tree if the BRC is not enough to hold a current item, then the algorithm will simply start a new bin. If the BRC is large enough I will then look that the left subtree can hold. If so, I go to the left subtree, if not, I check the if current bin RC fits current item, if so, return the current bin index. If not, I go to the right subtree and repeat this process to find the optimal bin that fits tightest. This also prevents looping over each previous bin which will result in O(N^2) running time for each new item.

5. Best Fit Decreasing (BF)

   Best Fit Decreasing works the same way as Best Fit(BF), except that Best Fit Decreasing will first rearrange all items from largest to smallest by item size.

Hongji Yan

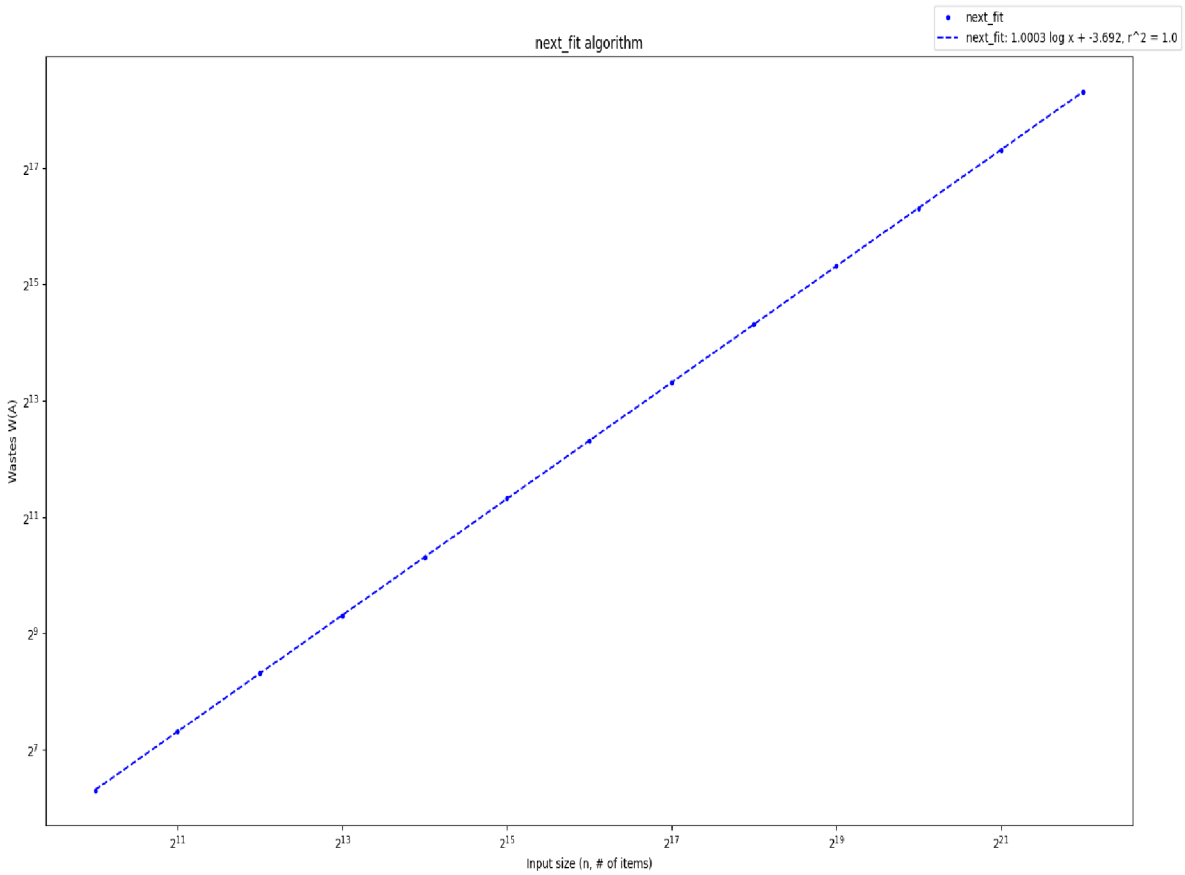
Input data and distribution

Random Uniformly number
I used rand() function from standard library to generate random decimal number between 0.0 and 0.6

Input item size (N)
NF: 2^9 – 2^22
FF: 2^9 – 2^22
BF: 2^9 – 2^22
FFD: 2^13 – 2^24
BBD: 2^10 – 22^5

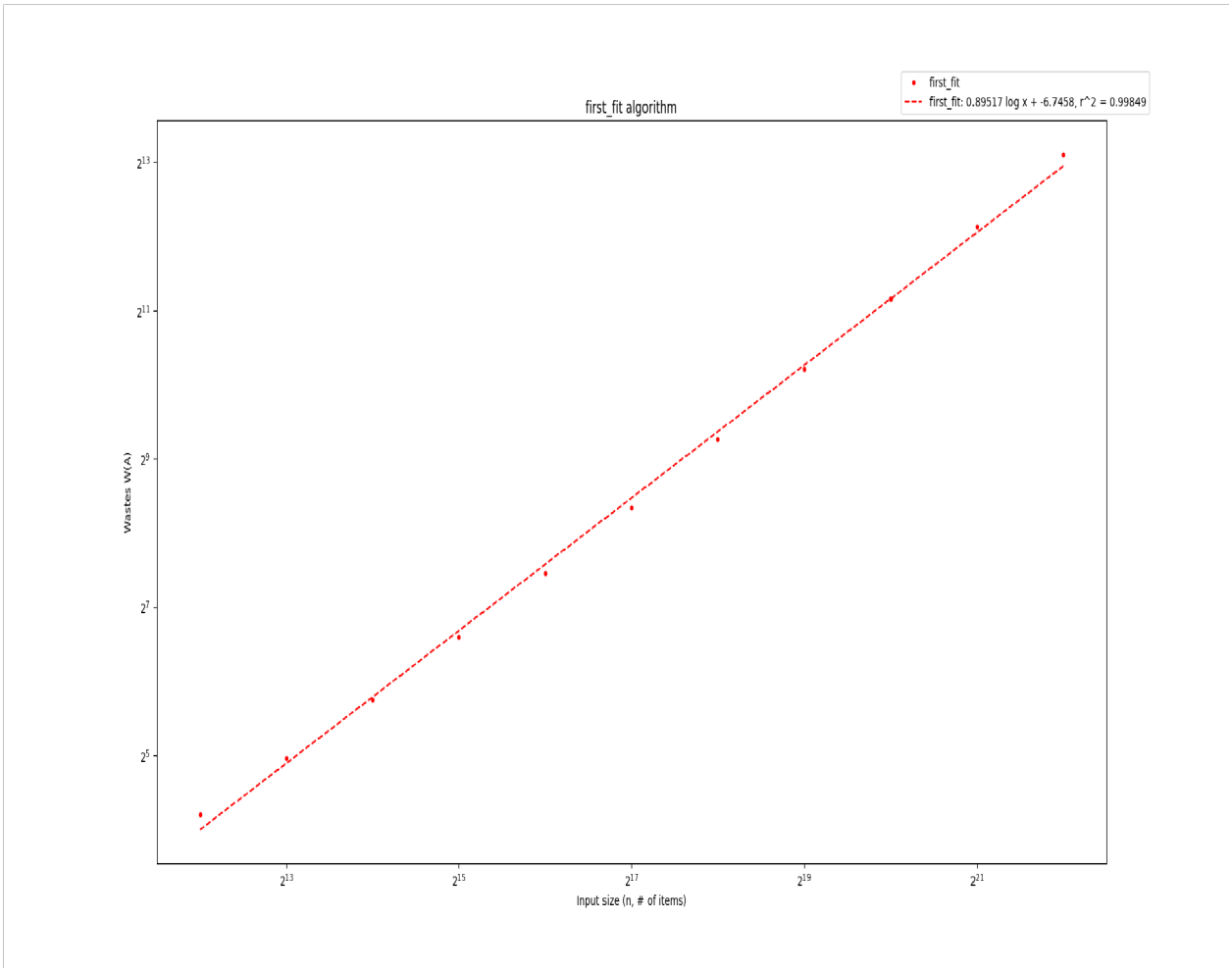
Hongji Yan

Plotting the waste for the NF algorithm



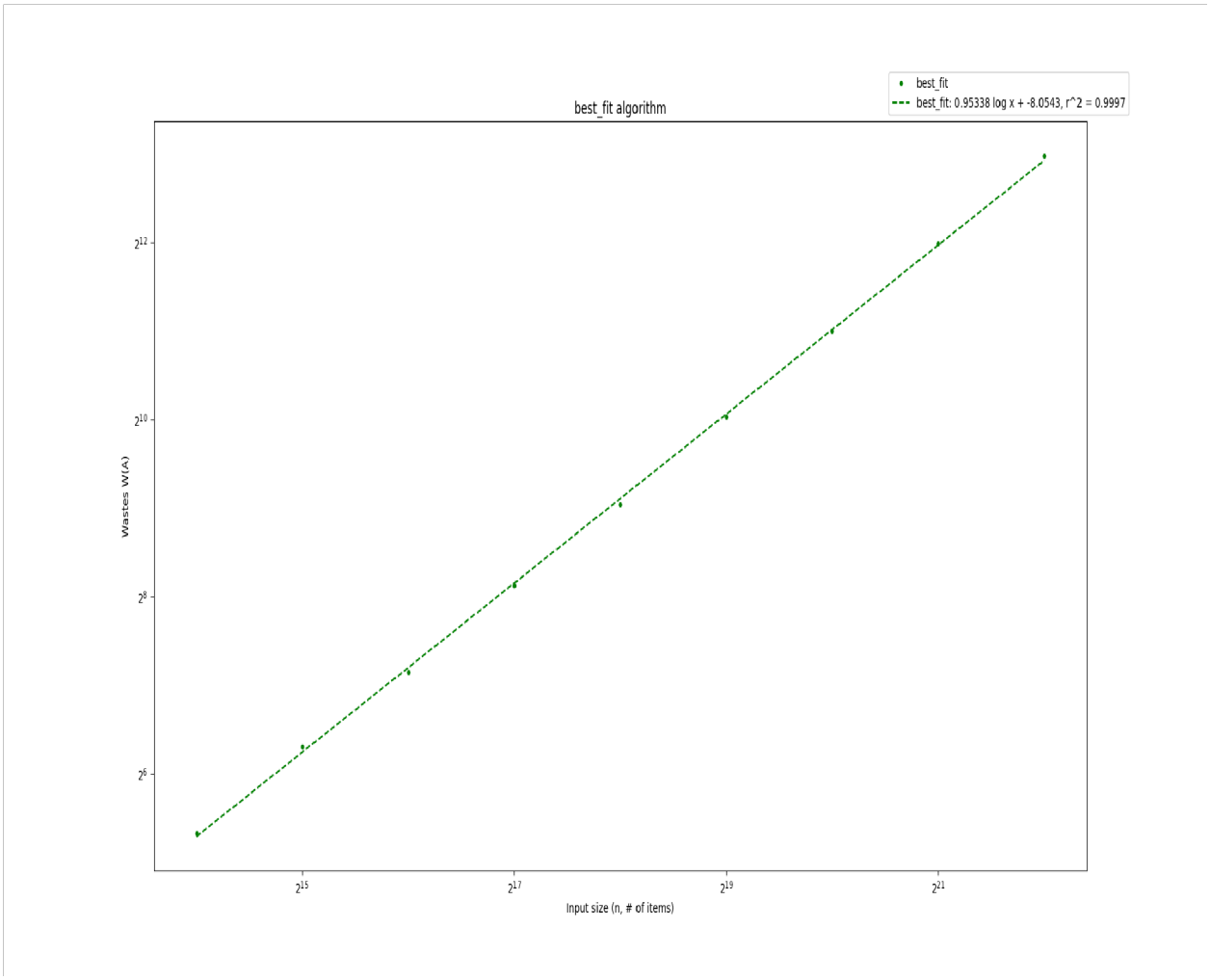
Hongji Yan

Plotting the waste for the FF algorithm



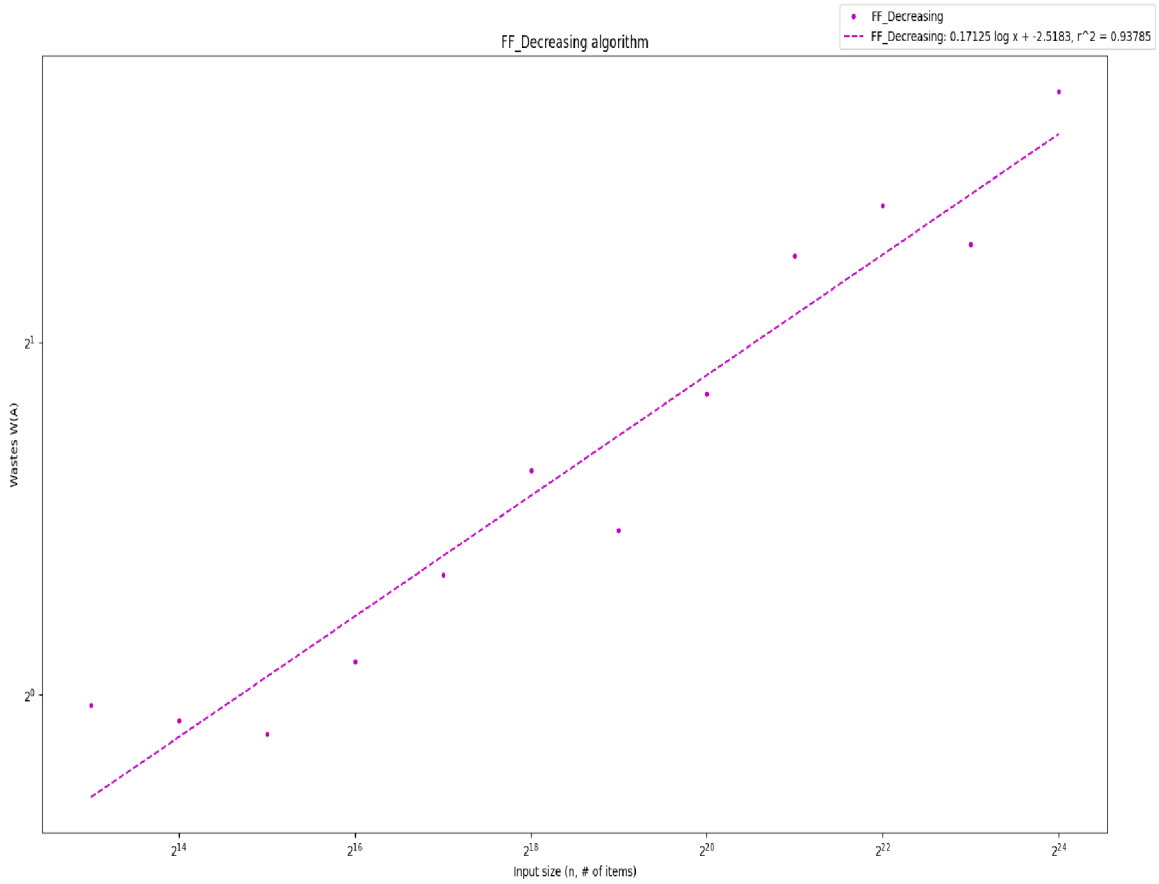
Hongji Yan

Plotting the waste for the BF



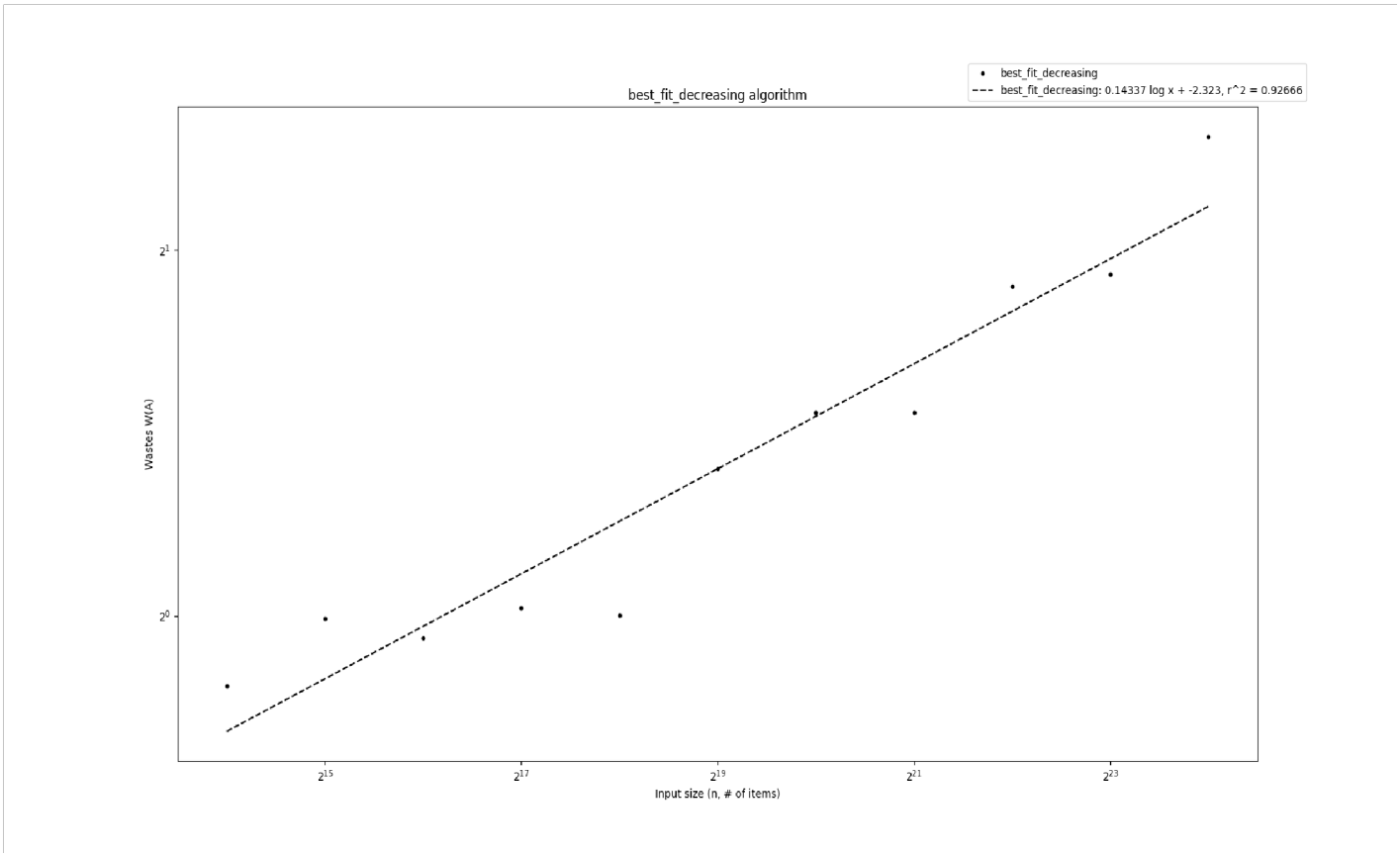
Hongji Yan

Plotting the waste for the FFD



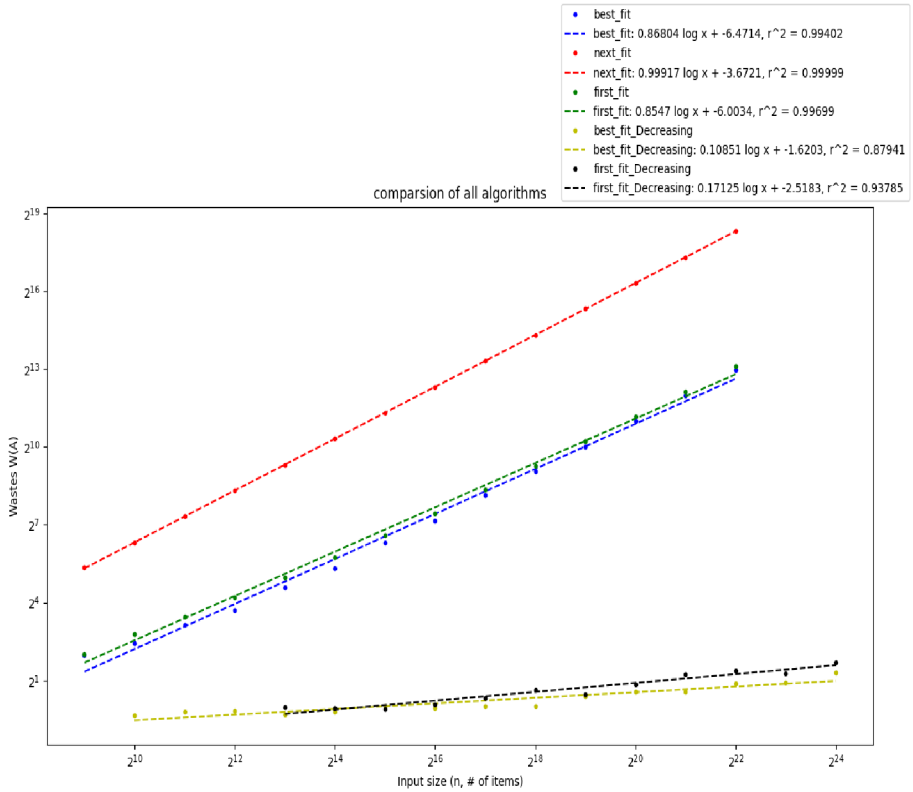
Hongji Yan

Plotting the waste for the BFD



Winner /Suggested Algorithm

Hongji Yan



My winner algorithm is **Best Fit decreasing, according to my experimental data(graph above), Best Fit decreasing has a W(A) about 2^1 with maximum input size N is 2^25,** which is extremely less than any of other algorithms. (example: first fit has W(A) of 2^22 that's is 2^25 – 2^1 = 33554430 more than best fit decreasing). The reason why best-fit decreasing uses least waste space is best fit maximize the available space for new items to place in since best fit always choose the bins that fits tightest, unlike first fit it only selects first bin that can hold the item without caring about optimizing bin available space. In addition, best fit decreasing put the larger items first and then the smaller items, it prevents algorithm start more new bins, which essentially save more space and causes less waste.

However, the winner algorithm Best Fit decreasing is not an algorithm that finds optimal least amount of for bin-packing problem waste, since the problem is Np-hardness, so there's no polynomial time algorithm to find the least amount of waste.