CS165 Project 2 Report

Huy Minh Tran

May 13 2020

1 Introduction

1.1 Purpose

Since finding the solution for the bin packing algorithm is NP-hard and the (NP-complete for the decision version), it is decided to use the waste of a bin-packing algorithm to experimentally determine the quality of 5 bin packing algorithms. The waste W(A), of a bin-packing algorithm, A, for any given list of items, is the number of bins used by the algorithm A minus the total size of all the items in the list. 5 bin packing algorithms are timed and implemented in C++ are Next Fit, First Fit, First Fit Decreasing, Best Fit and Best Fit Decreasing.

1.2 Experimental Setup

Each bin packing algorithm will receive an items vector that contains random numbers in the range of 0.0 to 0.7. The method to generate random numbers was to use random_device, default_random_engine and uniform_real_distribution which are built in of C++ random library. The size of the input array ranges from 2^3 to 2^{20} . Each bin packing algorithm is timed 5 different times. Each iteration, an input vector is passed to all the bin packing algorithms and the waste is added in total so that in the end, the sum of waste of each algorithm is divided by 5 to get the average waste of each algorithm.

One of the major difficulties of the project was dealing with floating point numbers since the item values are all chosen to be floating point numbers. Floating point numbers are not exact and therefore, doing equality comparison of a float point number against another would not give the correct result. In fact, one solution to this issue is to use the epsilon value (DBL_EPISLON) in the float library and the abs function (std::abs) in the standard library. By taking an absolute of one floating point number subtracting to another, the result can be compared to the epsilon value.

If the result is less than the epsilon value then both numbers are equal, otherwise they are not equal.

1.3 Generating Plot

The plots that are generated for 5 bin packing algorithms are on a loglog scale. The number of wasted bins is plotted as the vertical line (y-axis) and the size of the input array is plotted as the horizontal line (x-axis). Each figure has scatters to indicate the wasted bins correlate to the input size and a regression line.

1.4 Optimization: Balanced Binary Search Tree

Best Fit and First Fit algorithm (and so does Best Fit Decreasing and First Fit Decreasing) run in quadratic time. Therefore, one solution to improve the runtime from quadratic to linearithmic is to use a balanced binary search tree to quickly and efficiently look up the right bin to place the value in. For this project, a Weak AVL Tree (WAVL Tree) was implemented from scratch to enhance the runtime of the algorithms.

In the optimized First Fit algorithm, each node of the tree contains a key, which is the index of the bin and a value, which is the remaining capacity of the node. Each node also has an additional variable called best remaining capacity (brc). A modified version of binary search was used to search for the smallest index that can contain the item that is less than or equal to the brc and less than or equal to the value of the node. The algorithm will stop if it finds the left most node that has enough space to store the item. A recursive algorithm was used to update the brc of the node after each insertion, rebalancing of the tree and after each item is added to a node.

In the optimized Best Fit algorithm, each node of the tree contains a key, which is the remaining capacity (rc) of the node and a value, in this case, is the index of the bin. Search and insertion are performed similar to the version of First Fit. However, in order to update the key, the node that contains the key must be removed first, then reinserted as a new node.

1.5 Content

Each content is organized as follows:

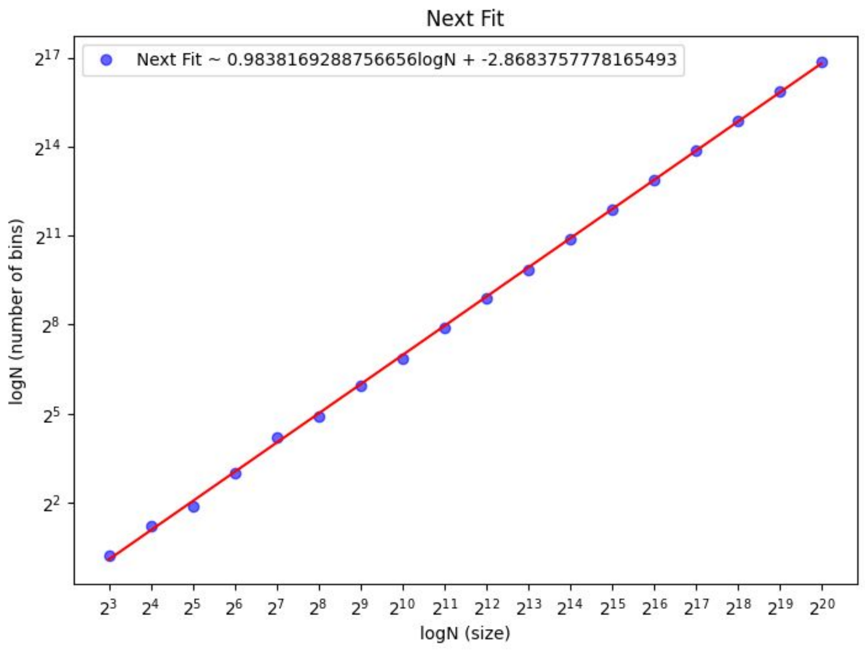
1. Description: pseudocode and explanation of how the bin packing algorithm works.
2. Experimental wasted bin: Plot of the bin packing algorithm.
3. Analysis: the analysis of the algorithm based on the data and the plot.

2 Next Fit

The Next Fit version of bin packing algorithm maintains an index of the last bin in the list of bins that contain the items. For each item, the algorithm checks if the last bin has enough space to store the value of the item. If the value of the item fits, the algorithm will add it to the last bin, otherwise, the algorithm will add another bin to the end of the bin list to contain the new value. Below is the pseudocode of the Next Fit algorithm:

```
void function next_fit(items, assignments, free_space) {
   bins = list()
   bin_index = 1
   for i = 0; i < items.size(); ++i {
     if (bins[bin_index] + items[i] > 1){
        Add new bin to bins
        ++bin_index
     }
   }
   bins[bin_index] += items[i]
   assignment[i] = bin_index
   free_space[bin_index] -= bins[bin_index]
}
```

2.1 Experimental wasted bin



2.2 Analysis

It can be seen that the Next Fit bin packing algorithm produces nearly linear waste. The slope of the line is 0.98, which is very close to 1 and therefore, it shows that the waste function for Next Fit is likely linear. The wasted bin of Next Fit is concluded to be O(n).

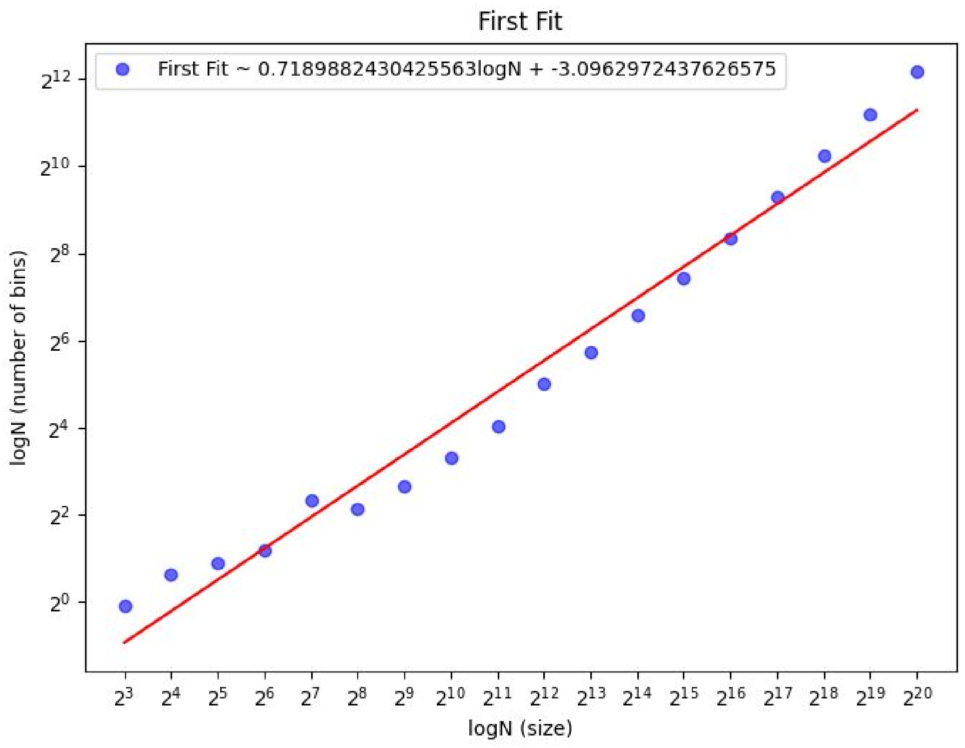
3 First Fit

For the normal version of First Fit, the algorithm loops over all the items in the list, then loops over the list of the bins to find the first bin that has enough space to store the item. If there is a bin that fits the item, the algorithm will place the item in that bin, otherwise, the algorithm will add another bin to the list of bins and add the item to the newly created bin.

For the optimized version of First Fit, the algorithm loops over all the items in the list, if there is no item in the tree, the first item will be inserted along with the first bin index. If the value of the item is greater than the best remaining capacity of the root, the item will be inserted along with the bin index. Otherwise, the tree will add that item into the tree where the add function is responsible for finding the leftmost node that can store the item. The pseudocode for the algorithm is described below:

```
void function first_fit(items, assignments, free_space){
       bin_index = 1
       tree = WAVLTree()
       for int i = 0; i < items.size(); ++i {
              if tree.is_empty()
                     tree.insert(items[i], bin_index)
              else if items[i] > brc_of(tree.root)
                     tree.insert(items[i], tree.size() + 1)
              else
                     tree.add_item(items[i]);
                     bin_index = tree.size()
                     assignment[i] = tree.size()
                     free_space[bin_index] -= items[i]
       }
}
```

3.1 Experimental wasted bin



3.2 Analysis

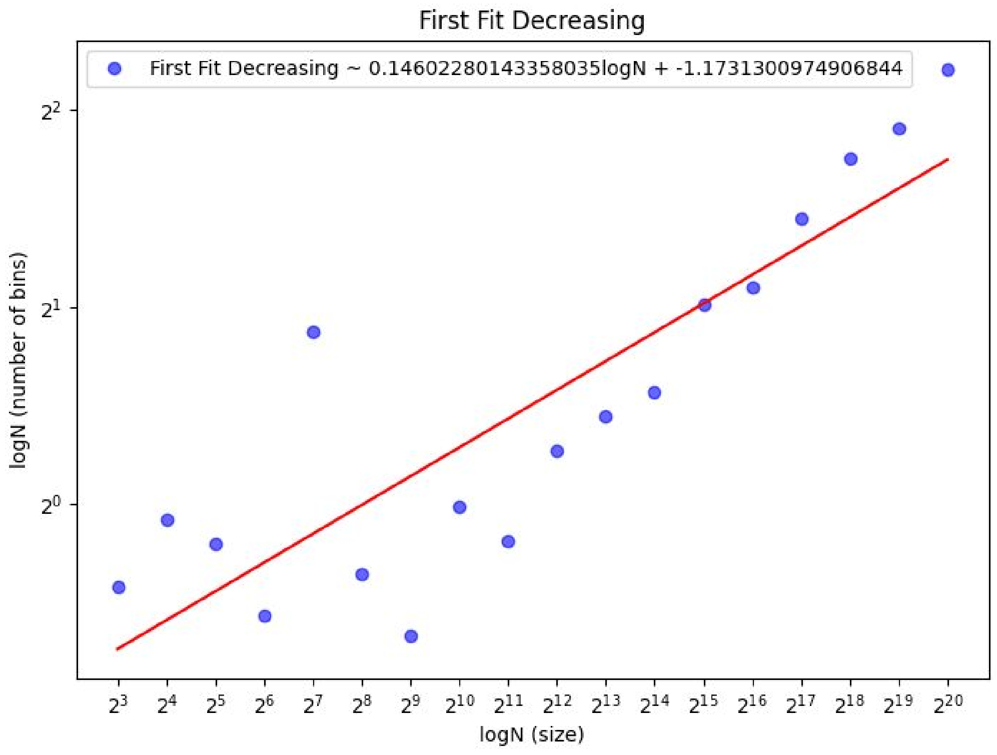
It can be seen the First Fit bin packing algorithm does better than the next fit in terms of the number of wasted bins. The slope of the line is 0.72 which is approximately 7/10. Therefore, the waste of the First Fit algorithm is less than linear and is concluded to be $O(n^{7/10})$

4 First Fit Decreasing

For the normal version of First Fit Decreasing, the list of items is sorted in decreasing order, then the algorithm calls the First Fit algorithm to perform on the sorted list of items.

For the optimized version of First Fit Decreasing, the algorithm performs as above but then it calls the optimized First Fit algorithm to perform on the sorted list of items. Pseudocode is not provided because the algorithm is similar to First Fit but only involves sorting all the items in decreasing order.

4.1 Experimental wasted bin



4.2 Analysis

It can be seen that the First Fit Decreasing algorithm performs way better than Next Fit and pure First Fit. The slope of the line is 0.15, which can be considered as the algorithm wastes very minimal bins. Therefore, the wasted bin of this algorithm is determined to be $O(n^{15/100})$.

5 Best Fit

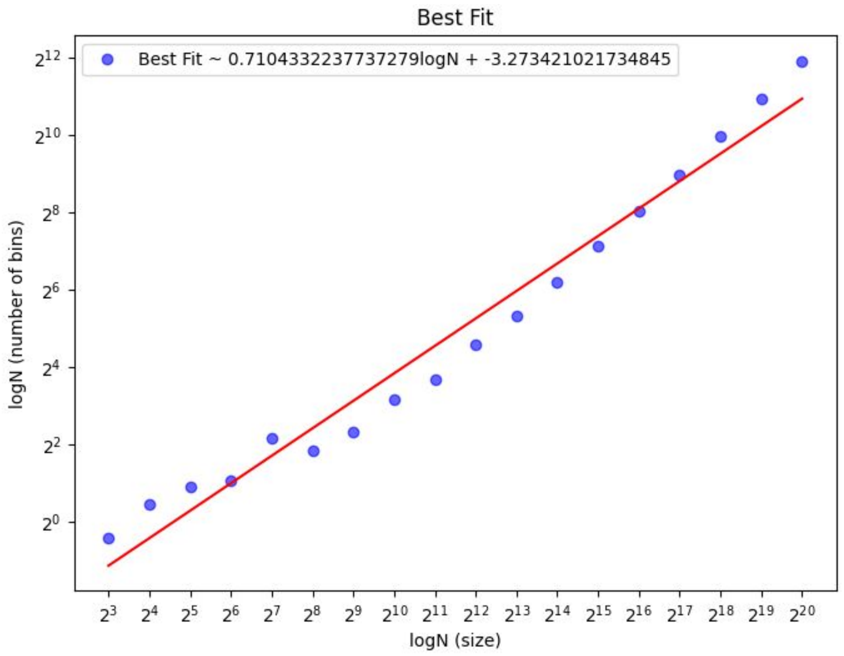
For the normal version of Best Fit, the algorithm loops over the list of items and loops over the list of bins, if there is no bin that can fit the item, a new bin will be added. Otherwise, the algorithm will try to find the bin that fits the item tightest in order to avoid huge waste.

For the optimized version of Best Fit, the algorithm loops over the list of items, if the tree does not have any node, the first item will be added to the tree along with the first bin number. Otherwise, the algorithm will find the bin to fit the item, if there is no such bin, the tree will add the item with the new bin. If there is a bin that fits the item tightest, the algorithm will remove the bin, update the new item value and insert that new item value with the same bin number. Below is the pseudocode of the optimized version:

```
void function best_fit(items, assignments, free_space){
        tree = WAVLTree()
        bin = 1
        for i = 0; i < items.size(); ++i {
                if tree is empty
                        tree.insert(1.0 - items[i], bin)
                else {
                        // find node that can store the items tightest
                        tree.find_node(items[i])

                        if node does not exist
                                tree.insert(1.0 - items[i], tree.size() + 1)
                        else {
                                current_key = node.key
                                bin = node.value
                                new_key = current_key - items[i] // updating new rc
                                tree.remove(current_key)
                                tree.insert(new_key, bin)
                        {
                }
        }
```

5.2 Experimental wasted bin



5.3 Analysis

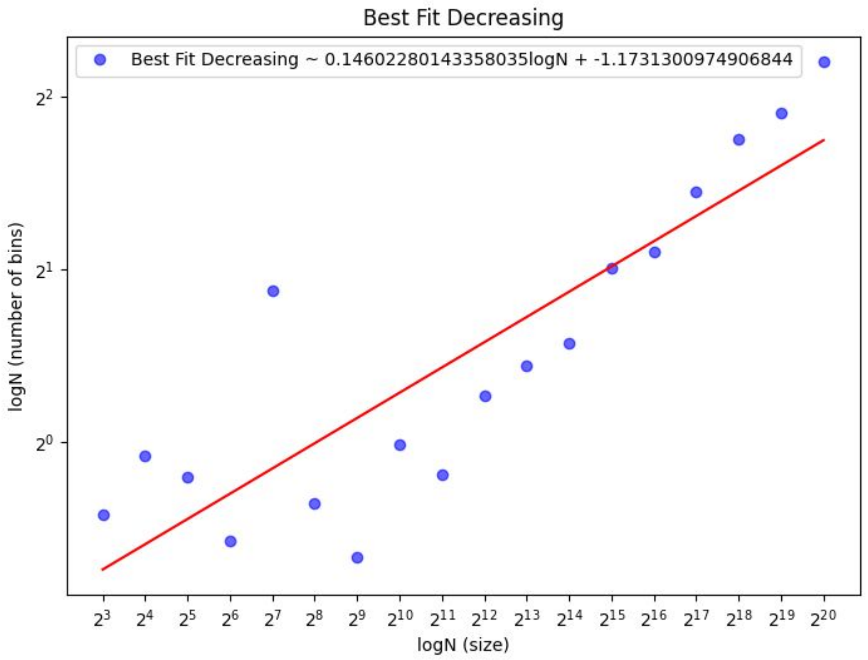
It can be seen that the best fit algorithm performs slightly better than pure First Fit and much better than Next Fit but worse than First Fit Decreasing. The slope of the line is 0.71, which can be considered as the algorithm wastes less bins than Next Fit but more than First Fit Decreasing algorithm. Hence, the wasted bin is determined to be O($n^{7/10}$).

6 Best Fit Decreasing

For the normal version of Best Fit Decreasing, the list of items is sorted in decreasing order, then the algorithm calls the First Fit function to perform on the sorted list of items.

For the optimized version of Best Fit Decreasing, the algorithm performs as above but then it calls the optimized First Fit function to perform on the sorted list of items. The pseudocode is not provided because the algorithm is similar to Best Fit, except it involves sorting the items in decreasing order before calling Best Fit.

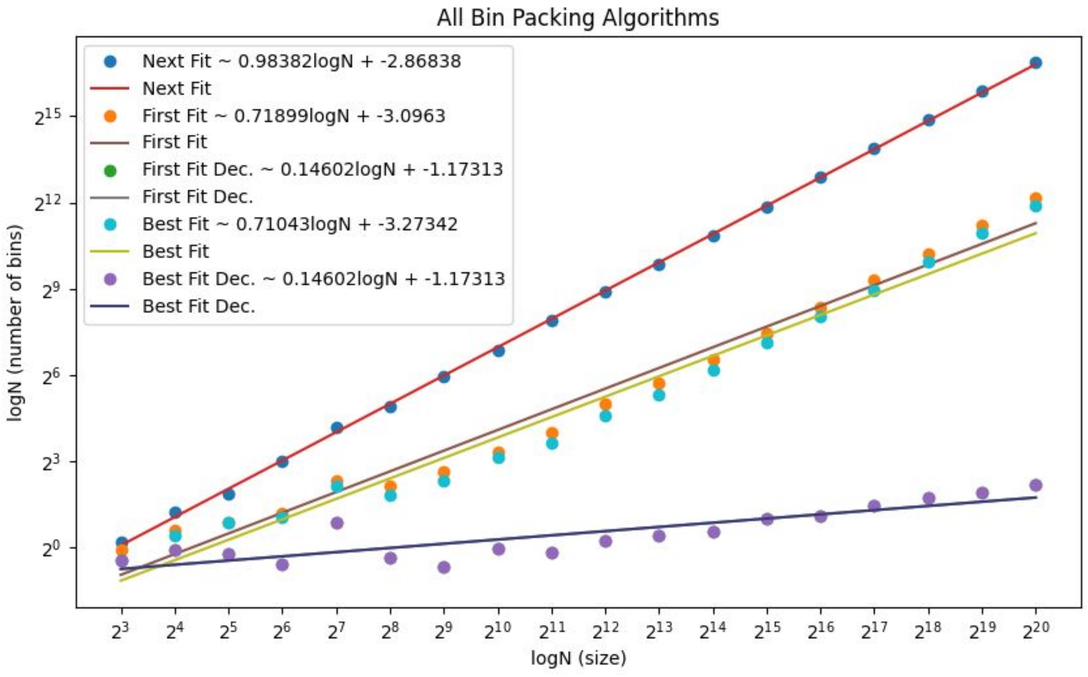
6.2 Experimental wasted bin



6.3 Analysis

It can be seen that the Best Fit Decreasing algorithm wastes minimal bins when the size of the input grows large. The slope of the line is 0.15, which is identical to the First Fit Decreasing algorithm and therefore, its wasted bin is also O(n$^{15/100}$).

7 Comparison of All Algorithms



The most obvious thing that can be seen from the plot is that Next Fit produces wasted bins at the highest rate compared to other bin packing algorithms. The rate of wasted bins produced by the algorithm is mostly linear, which is understandable because the algorithm never looks back at previous bins once it adds the new bin.

Although Best Fit does better than Next Fit, it only produces slightly better wasted bins compared to First Fit. It can be understood that although Best Fit and First Fit both look back at the previous bins, Best Fit looks for the bin that fits the tightest while First Fit would move onto another item once it finds a bin that fits the current item.

On the other hand, it is noticed that Best Fit Decreasing and First Fit Decreasing have identical results in terms of wasted bin and the slope of the line. This can be explained that both algorithms place the bins with the large items in the beginning, therefore, the first bin is also the bin that fits tightest.

8 Conclusion

Given all the information collected by experimentally running all the bin packing algorithms, although First Fit Decreasing and Best Fit Decreasing are tie, First Fit Decreasing is selected to be the best bin packing algorithm because it produces identical wasted bins compared to Best Fit Decreasing. However, the implementation of the optimized version of First Fit Decreasing is simpler than Best Fit Decreasing because implementing the remove function is challenging. Ranked next is Best Fit followed by First Fit because Best Fit has a slightly better slope. Lastly, Next Fit is determined to be the worst bin packing algorithm for producing wasted bin linearly.