Stephen Belden

Meghan Haukaas

Chris Ruiz

**Bin Packing Test Cases**

**Test Cases Description**

We will be building and testing two different bin packing algorithms. Both algorithms respect the rules of bins, both perform in the same time complexity, and both algorithms are expected to fill bins relatively efficiently. As such, most test cases can be shared between both algorithms. Only with randomly generated input data and a few specific edge cases do we expect to see differences in how each algorithm fills bins.

**General Test Case Construction**

1. Input: Nothing  
   Expected Output: 1 bin, 0% full
2. Input: 1 object equal in size to 1 bin  
   Expected Output: 1 bin, 100% full
3. Input: 1 object larger than 1 bin  
   Expected Output: 1 bin, 0% full, and an error
4. Input: n objects of size 1/n bin  
   Examples: 2 objects of size 1/2 bin; 3 objects of size 1/3 bin  
   Expected Output: 1 bin, 100% full
5. Input: n objects of size greater than 1/n bin  
   Example: 3 objects of 2/3 bin  
   Expected output: n bins, each more than 50% full
6. Input: A number of unequal objects adding up to size 1 bin  
   Example: 0.5 \* size, 0.3 \* size, 0.2 \* size  
   Expected output: 1 bin, 100% full
7. Input: A large set of objects that can be fit perfectly into n bins, inserted randomly  
   Expected output for first fit: no more than 2 \* n bins  
   Expected output for best fit: no more than 1.7 \* n bins
8. Input: n objects of size k (all less than the size of a bin), inserted randomly  
   Example: any valid random input  
   Expected output: any valid output (no more than n bins, no bins over 100%)  
   **This test case will be used to verify time complexity. As n changes, completion time should change with O(N log N)**

**Test Cases for Specific Edge Cases**

For these test cases, the capacity of each bin is 1

1. Insert: 0.6, 0.6, 0.4  
   Expected Output: One bin holds 0.6 and 0.4, another bin holds 0.6  
   Reasoning: This test makes sure that both algorithms look back through existing bins before creating a new bin.
2. Insert: 0.5, 0.6, 0.3, 0.4  
   Expected Output for first fit: binA{0.5, 0.3} binB{0.6, 0.4}  
   Expected Output for best fit: binA{0.5, 0.4} binB{0.6, 0.3}  
   Reasoning: This test shows a clear difference between how first fit and best fit insert elements.
3. Insert: 0.5, 0.6, 0.3, 0.1, 0.5  
   Expected Output for first fit: binA{0.5, 0.3, 0.1} binB{0.6} binC{0.5}  
   Expected Output for best fit: binA{0.5, 0.5} binb{0.6, 0.3, 0.1}  
   Reasoning: This test shows a case in which best fit is more space efficient than first fit.
4. Insert: 0.5, 0.6, 0.3, 0.4, 0.2  
   Expected Output for first fit: binA{0.5, 0.3, 0.2} binB{0.6, 0.4}  
   Expected Output for best fit: binA{0.5, 0.4} binB{0.6, 0.3} binC{0.2}  
   Reasoning: This test shows an unusual case in which first fit incidentally creates a more space efficient solution

**Discussion of Test Cases**

All test cases worked as expected. Both First Fit and Best Fit have deterministic behavior, so no deviation from the expected outcomes was anticipated. These results indicate that the algorithms work in every case that we can imagine.

Additionally, both algorithms were tested for robustness with large data sets. Each algorithm was tasked with inserting 1,000,000 elements or more. Both algorithms maintained their stability, correctness, and O(N log N) running time for these large data sets.

**Time Complexity Analysis – First Fit**

First Fit fits perfectly into time class O(N log N), with N inserted objects. Completion time for the algorithm was measured for input sizes 8 – 131072. Using the Least Squares Fitting method on an assumption of time = C\*(N log N), a nearly perfect best-fit curve was found, with a variance (average error squared) of approximately 9.5 \* 10-24.

It’s worth noting that this result is significantly more precise that our recorded data. Completion times were only recorded to 6 significant figures.

For comparison, the best fit variance for C\*(N) was 0.02, and the best fit variance for C\*(N2) was 0.522. Our first fit algorithm fits O(N log N) as closely as could ever be hoped.

The full data set and statistical methods used can be found in FirstFit\_Time.xlsx