BIN BACKING ALGORITHM

The Bin-Packing Problem (BPP) can also be described,using the terminology of knapsack problems, as follows. Given n items and n knapsacks (or bins), with

assign each item to one bin so that the total weight of the items in each bin does not exceed c and the number of bins used is a minimum.

Next Fit algorithm

The simplest approximate approach to the bin packing problem is the Next-Fit (NF) algorithm which is explained later in this article. The first item is assigned to bin 1. Items 2,... ,n are then considered by increasing indices : each item is assigned to the current bin, if it fits; otherwise, it is assigned to a new bin, which becomes the current one.

Visual Representation

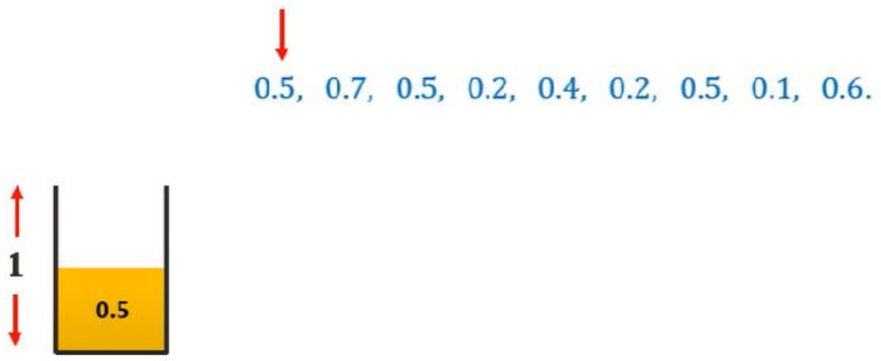
Let us consider the same example as used above and bins of size 1

Assuming the sizes of the items be {0.5, 0.7, 0.5, 0.2, 0.4, 0.2, 0.5, 0.1, 0.6}.

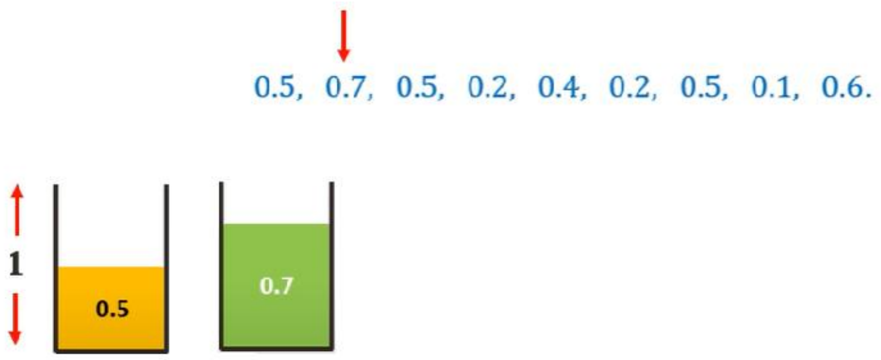
The minimum number of bins required would be Ceil ((Total Weight) / (Bin Capacity))= Celi(3.7/1) = 4 bins.

The Next fit solution (NF(I))for this instance I would be-

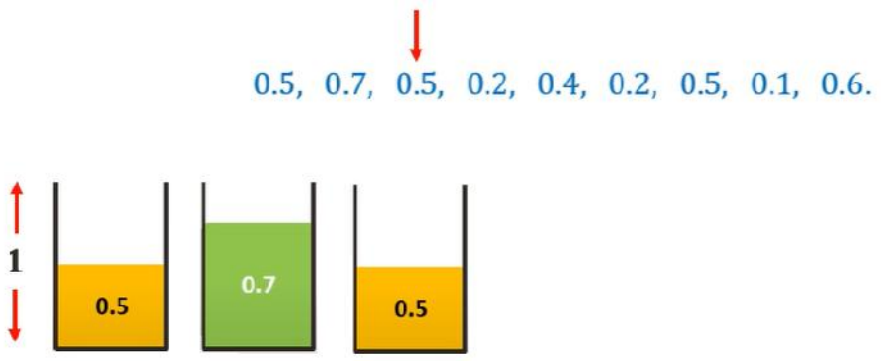
Considering 0.5 sized item first, we can place it in the first bin



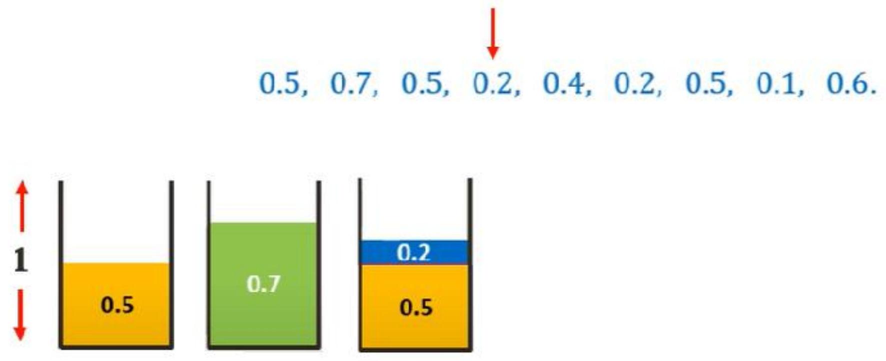
Moving on to the 0.7 sized item, we cannot place it in the first bin. Hence we place it in a new bin.



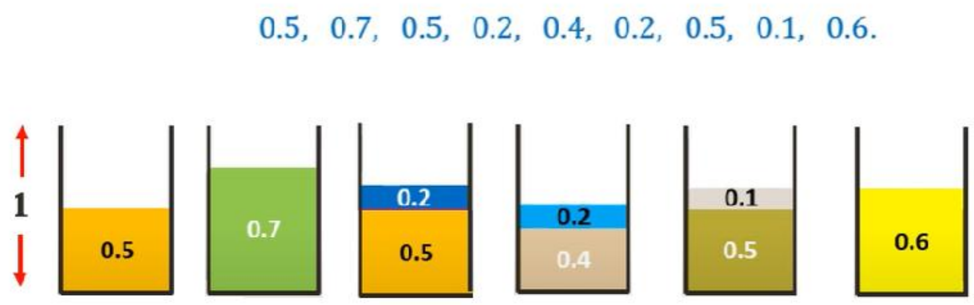
Moving on to the 0.5 sized item, we cannot place it in the current bin. Hence we place it in a new bin.



Moving on to the 0.2 sized item, we can place it in the current (third bin)



Similarly, placing all the other items following the Next-Fit algorithm we get-



Thus we need 6 bins as opposed to the 4 bins of the optimal solution. Thus we can see that this algorithm is not very efficient.

First Fit algorithm

A better algorithm, First-Fit (FF), considers the items according to increasing indices and assigns each item to the lowest indexed initialized bin into which it fits; only when the current item cannot fit into any initialized bin, is a new bin introduced

Visual Representation

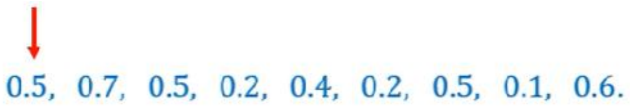
Let us consider the same example as used above and bins of size 1

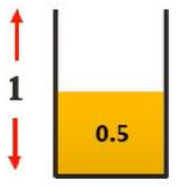
Assuming the sizes of the items be {0.5, 0.7, 0.5, 0.2, 0.4, 0.2, 0.5, 0.1, 0.6}.

The minimum number of bins required would be Ceil ((Total Weight) / (Bin Capacity))= Celi(3.7/1) = 4 bins.

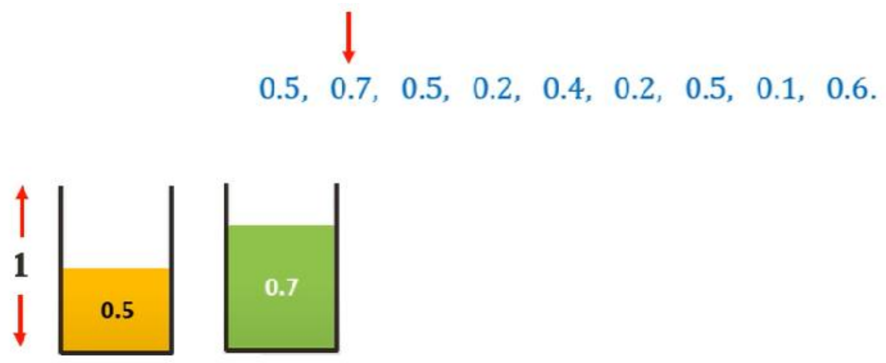
The First fit solution (FF(I))for this instance I would be-

Considering 0.5 sized item first, we can place it in the first bin

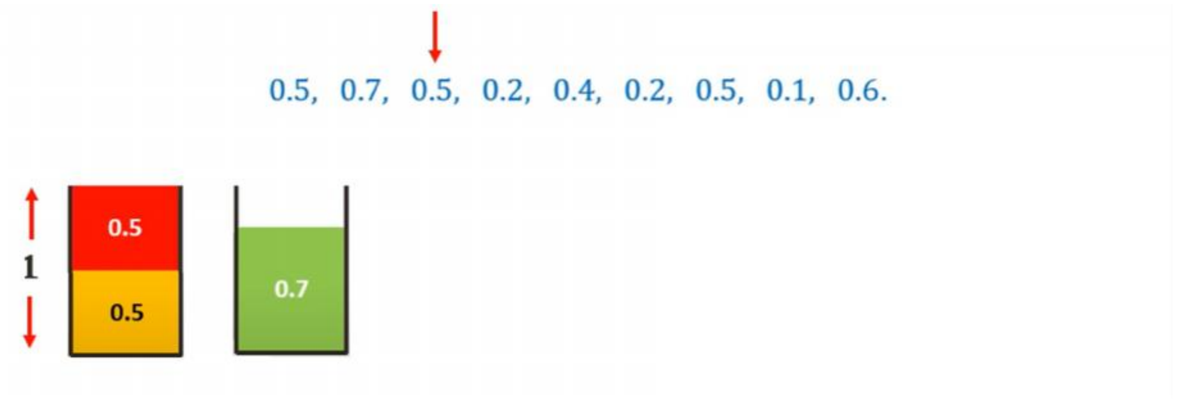




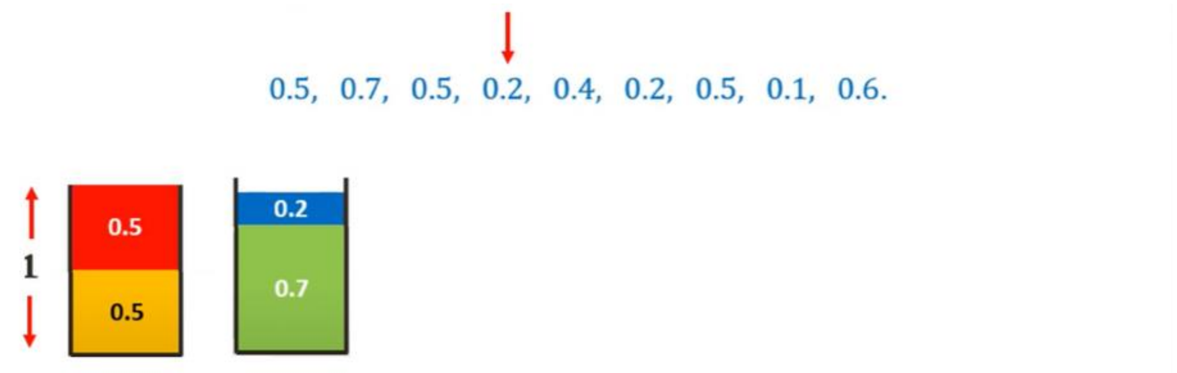
Moving on to the 0.7 sized item, we cannot place it in the first bin. Hence we place it in a new bin.



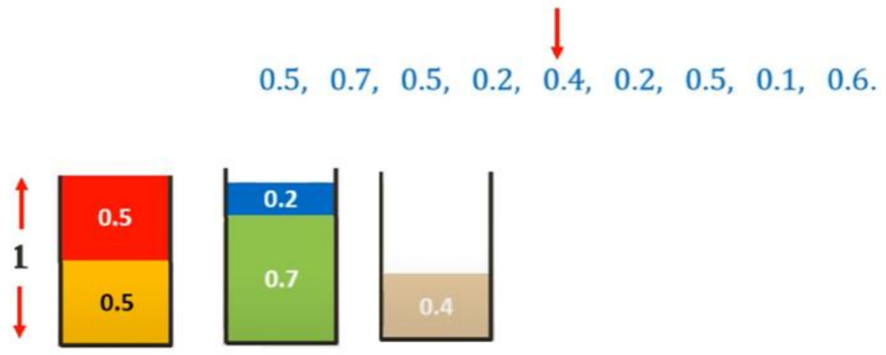
Moving on to the 0.5 sized item, we can place it in the first bin.



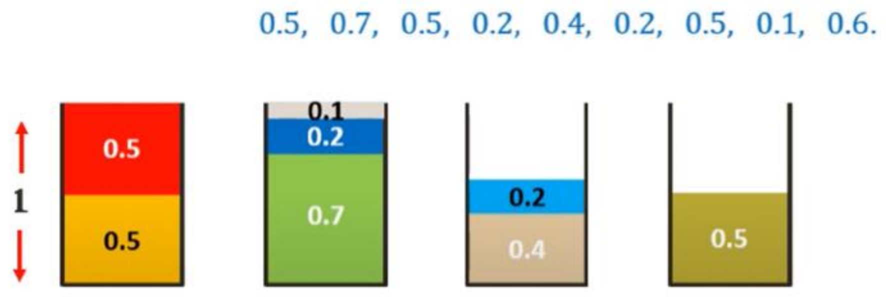
Moving on to the 0.2 sized item, we can place it in the first bin, we check with the second bin and we can place it there.



Moving on to the 0.4 sized item, we cannot place it in any existing bin. Hence we place it in a new bin.



Similarly, placing all the other items following the First-Fit algorithm we get-



Thus we need 5 bins as opposed to the 4 bins of the optimal solution but is much more efficient than Next-Fit algorithm.

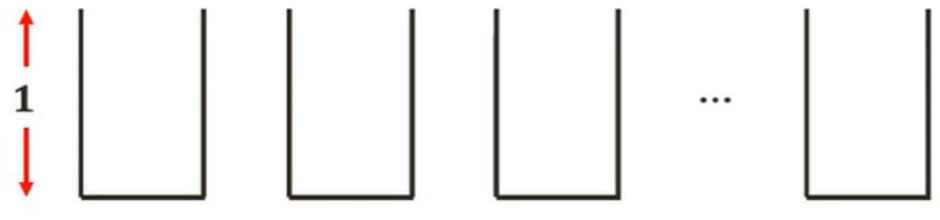
Best Fit Algorithm

The next algorithm, Best-Fit (BF), is obtained from FF by assigning the current item to the feasible bin (if any) having the smallest residual capacity (breaking ties in favor of the lowest indexed bin).

Simply put,the idea is to places the next item in the *tightest* spot. That is, put it in the bin so that the smallest empty space is left.

Visual Representation

Let us consider the same example as used above and bins of size 1

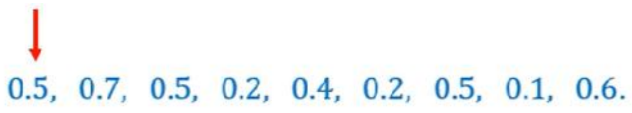


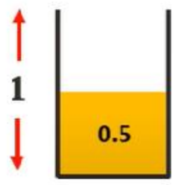
Assuming the sizes of the items be {0.5, 0.7, 0.5, 0.2, 0.4, 0.2, 0.5, 0.1, 0.6}.

The minimum number of bins required would be Ceil ((Total Weight) / (Bin Capacity))= Celi(3.7/1) = 4 bins.

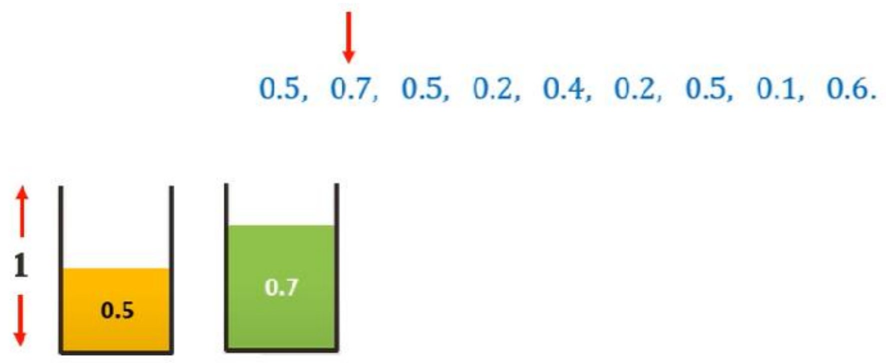
The First fit solution (FF(I))for this instance I would be-

Considering 0.5 sized item first, we can place it in the first bin

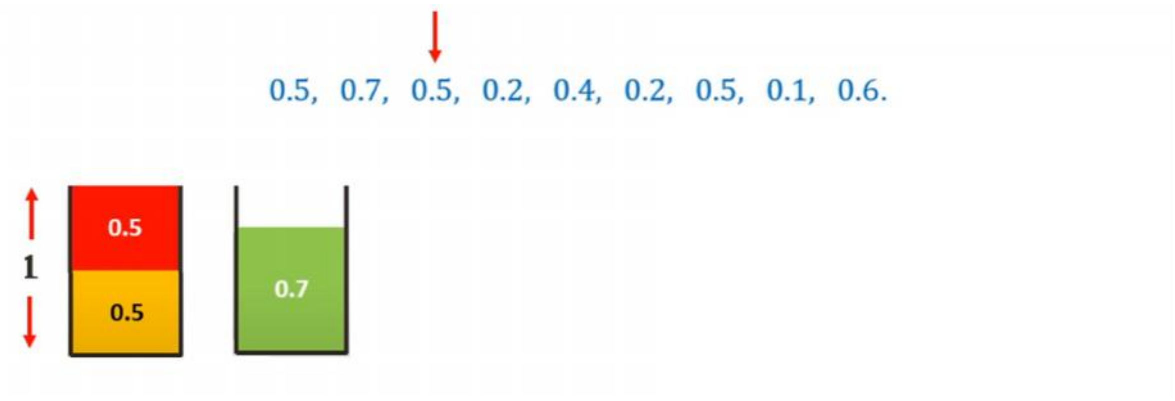




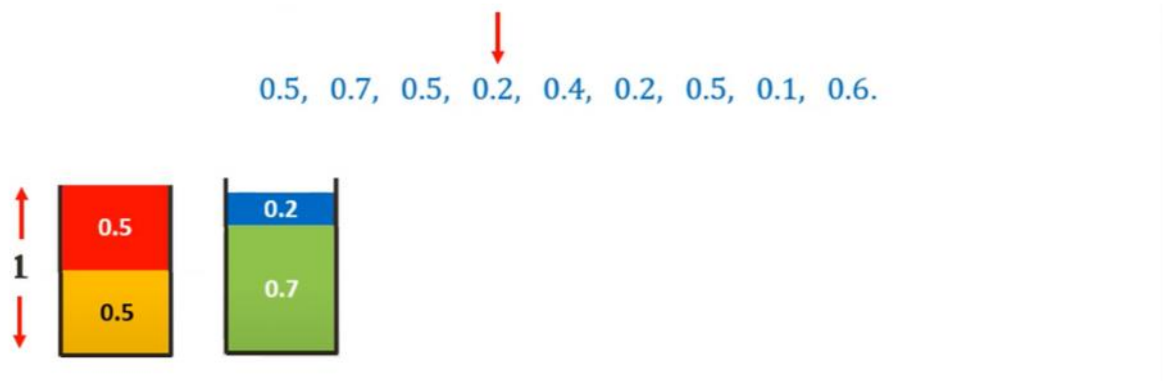
Moving on to the 0.7 sized item, we cannot place it in the first bin. Hence we place it in a new bin.



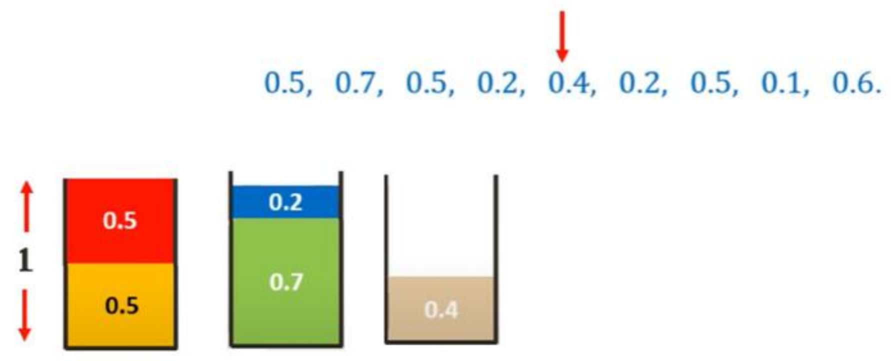
Moving on to the 0.5 sized item, we can place it in the first bin tightly.



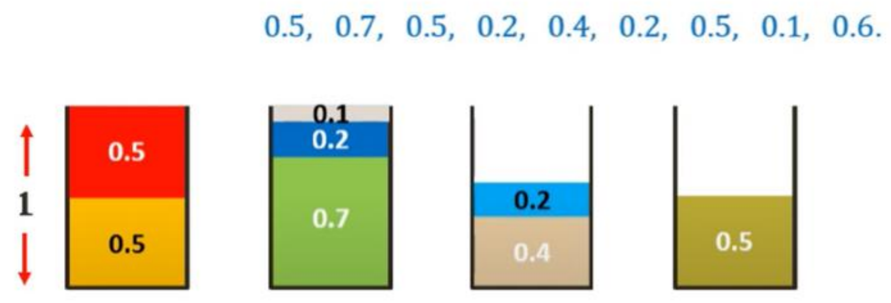
Moving on to the 0.2 sized item, we cannot place it in the first bin but we can place it in second bin tightly.



Moving on to the 0.4 sized item, we cannot place it in any existing bin. Hence we place it in a new bin.



Similarly, placing all the other items following the First-Fit algorithm we get-



Thus we need 5 bins as opposed to the 4 bins of the optimal solution but is

Worst Fit Algorithm

This algorithm involves an idea to places the next item in the least tight spot to even out the bins. In other words, put it in the bin so that most empty space is left.

Analysis Of upper-bound of Worst-Fit algorithm

Worst Fit can also be implemented in O(n Log n) time using Self-Balancing Binary Search Trees.

If z(I) is the optimal number of bins , then Worst Fit never uses more than 2 * z(I)-2 bins. So Worst Fit is same as Next Fit in terms of upper bound on