Bin Packing

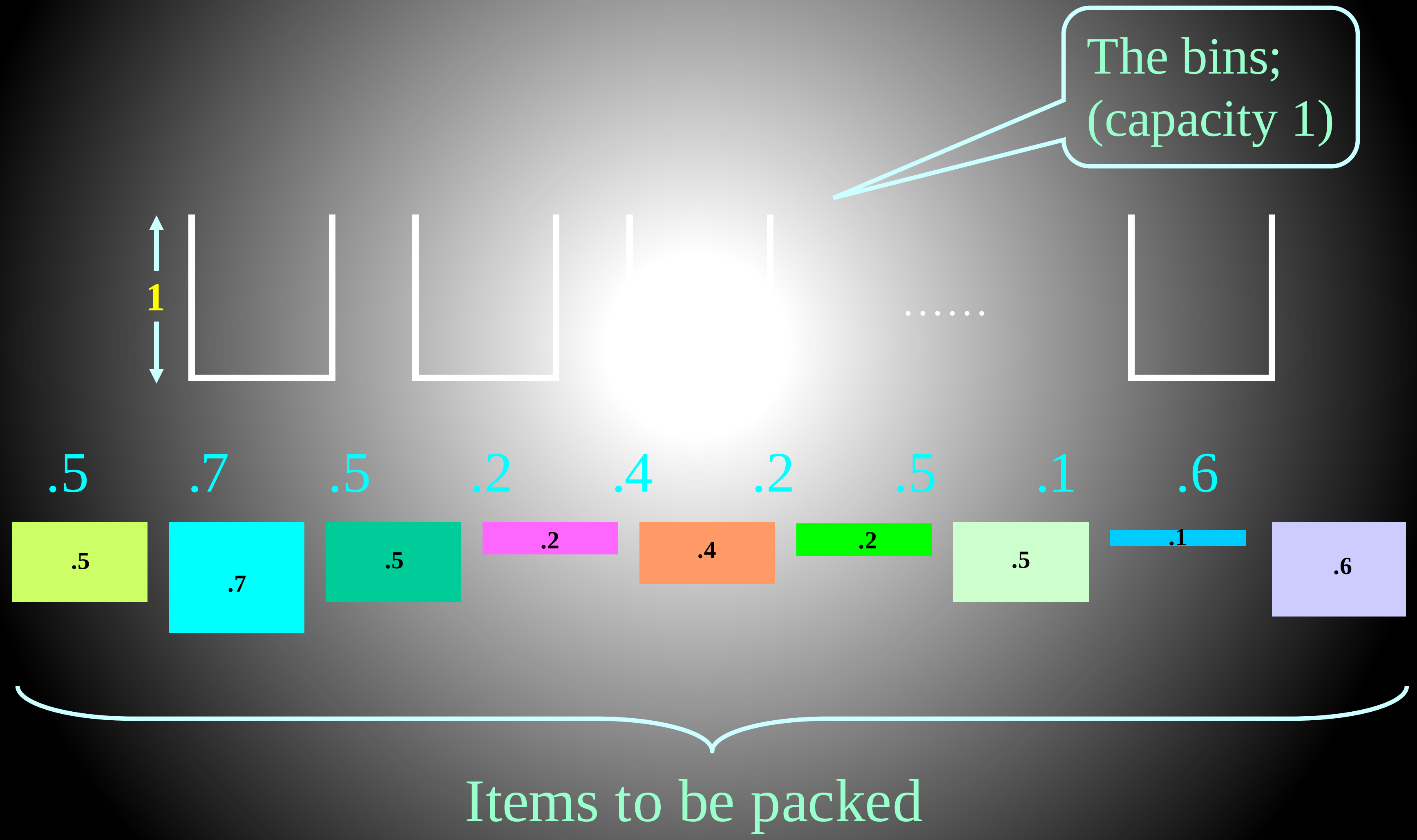
CS 165

Michael T. Goodrich

Some slides adapted from slides from

- Professor C. L. Liu, Tsing Hua University
- Professor Teofilo F. Gonzalez, UCSB

Bin Packing Example



Bin Packing Problem Definition

- Given n items with sizes $s_1, s_2, \ldots, s_n$ such that $0 \leq s_i \leq 1$ for $1 \leq i \leq n$, pack them into the fewest number of unit capacity bins.
- Problem is NP-hard (NP-Complete for the decision version).
- There is no known polynomial time algorithm for its solution, and it is conjectured that none exists.

Example Applications

Filling recycle bins

Loading trucks

Historical Application

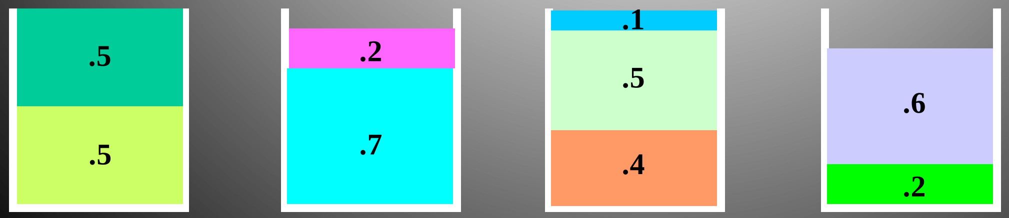
- Mix tapes

Bin Packing Optimal Solution

Bin Packing Problem

.5 .7 .5 .2 .4 .2 .5 .1 .6

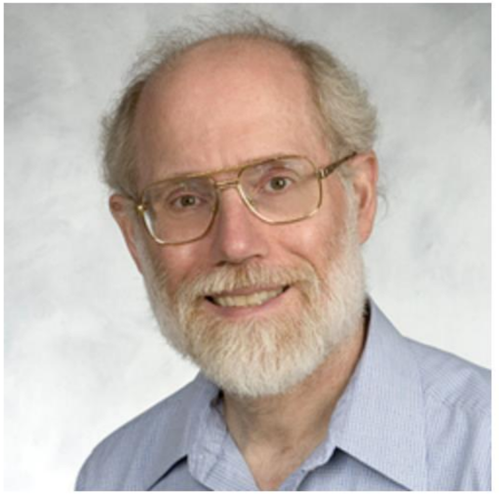
Optimal Packing



$M_{Opt} = 4$

Bin-Packing Heuristics

- Because the Bin-Packing problem is NP-hard, it is **very** unlikely that we can solve it 100% of the time with 100% optimality in polynomial time.
- Fortunately, there are a number of interesting heuristics we can apply to hopefully come close to optimality.



David Johnson

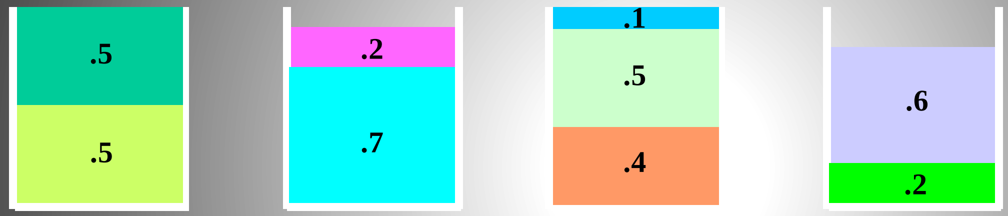
Next-Fit (NF) Algorithm

- Check to see if the current item fits in the current bin. If so, then place it there, otherwise start a new bin.

Next Fit (NF) Packing Algorithm Example

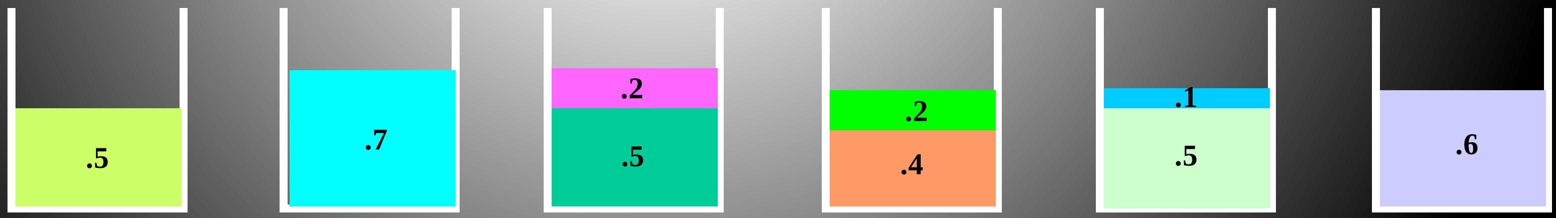
Bin Packing Problem

.5 .7 .5 .2 .4 .2 .5 .1 .6



$M_{Opt} = 4$

Next Fit Packing Algorithm



$M = 6$

Approximation Ratios

- **Approximation Algorithm:**

  – Not an optimal solution, but with some performance ratio guarantee for a given problem instance, I

  (e.g., no worst than *twice the optimal*)

- **Approx. Ratio = Alg(I)/Opt(I)**

Next Fit (NF) Approximation Ratio

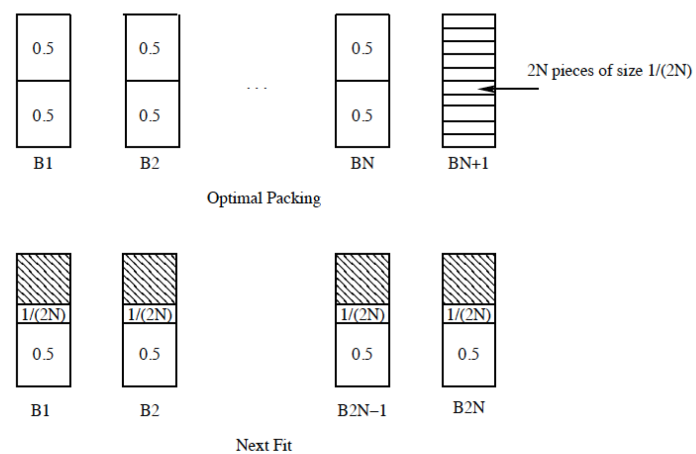
- Theorem: Let M be the number of bins required to pack a list I of items optimally. Next Fit will use at most 2M bins.
- **Proof**:
- Let $s(B_i)$ be the sum of sizes of the items assigned to bin B_i in the Next Fit solution.
- For any two adjacent bins (B_j and B_{j+1}), we know that $s(B_j) + s(B_{j+1}) > 1$.

Next Fit (NF) Approximation Ratio

- Let k be the number of bins used by Next Fit for list I. We prove the case when k is even (odd case is similar).
  - As stated above, $s(B_1) + s(B_2) > 1$, $s(B_3) + s(B_4) > 1$, ..., $s(B_{k-1}) + s(B_k) > 1$.
  - Adding these inequalities we know that $\sum s(B_i) > k/2$.
  - By definition $OPT = M > k/2$.
  - The solution $SOL = k < 2M$.

Next Fit (NF) Lower Bound

- There exist sequences such that Next Fit uses 2M −2 bins, where M is the number of bins in an optimal solution.
- Proof:
  - The odd numbered ones have s_i value $1/2$, and the even number ones have s_i value $1/(2N)$.



- $OPT = N + 1 = M$
- Therefore, $N = M - 1$
- Solution $SOL = 2N = 2M - 2$.

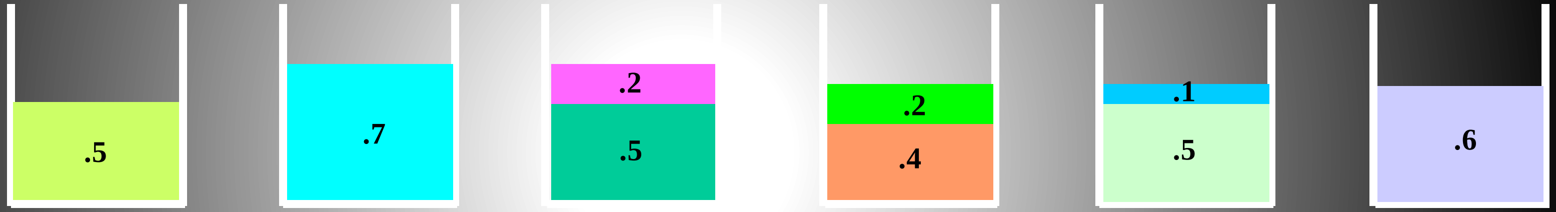
First Fit (FF) Algorithm

- Scan the bins in order and place the new item in the first bin that is large enough to hold it. A new bin is created only when an item does not fit in the previous bins.

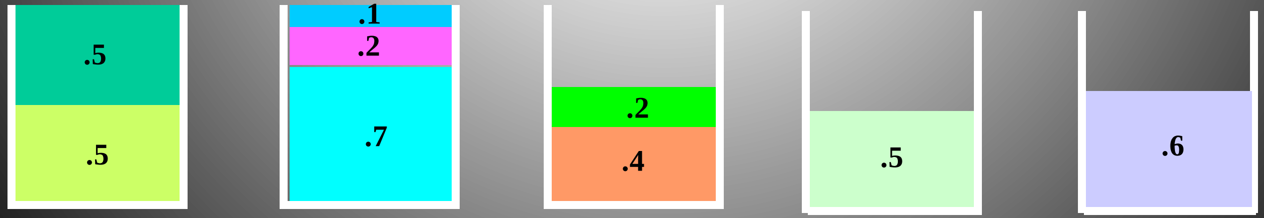
First Fit (FF) Packing Algorithm Example

.5 .7 .5 .2 .4 .2 .5 .1 .6

Next Fit Packing Algorithm



First Fit Packing Algorithm



M = 5

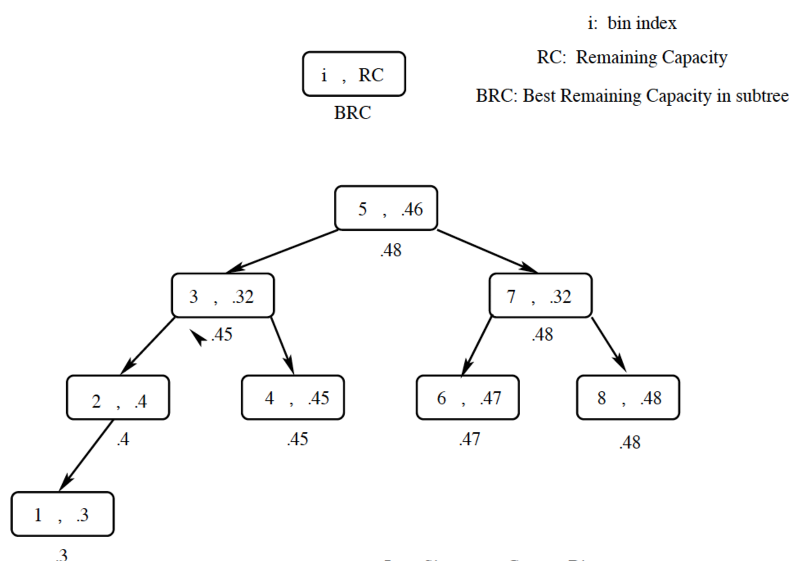
Running Time for First Fit

- Easily implemented in $O(n^2)$ time
- Can be implemented in $O(n \log n)$ time:
  - Idea: Use a balanced search tree with height $O(\log n)$.
  - Each node has three values:
    - index of bin
    - remaining capacity of bin
    - best (largest) in all the bins represented by the subtree rooted at the node.
  - The ordering of the tree is by bin index.

Faster First-Fit (FF) Algorithm

- 8 bins:

Bin	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8
R. Cap.	.3	.4	.32	.45	.46	.47	.32	.48



Item Size	Goes to Bin
s <= .3	B1
.3 < s <= .4	B2
.4 < s <= .45	B4
.45 < s <= .46	B5
.46 < s <= .47	B6
.47 < s <= .48	B8

Zip-Zip Trees

- For Project 2, students are required to implement the faster First-Fit Algorithm using the zip-zip tree data structure, which received the Best Paper Award at WADS 2023.

WADS *Algorithms and Data Structures Symposium*
(formerly *Workshop on Algorithms and Data Structures*)

Michael Goodrich

Ofek Gila

Robert Tarjan

First-Fit (FF) Approx. Ratio

- Let M be the optimal number of bins required to pack a list I of items. Then First Fit never uses more than $\lceil 1.7M \rceil$.
- **Proof**:
  - [omitted]

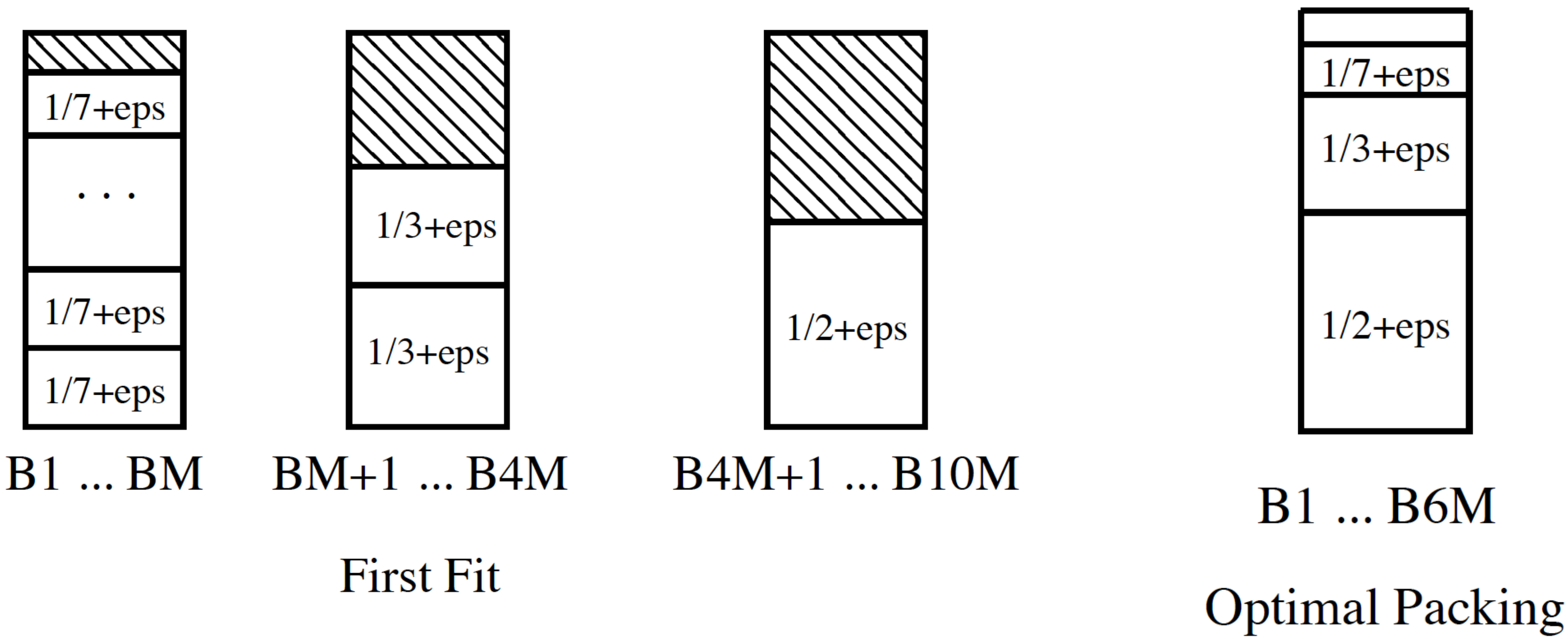
First-Fit (FF) Approx. Ratio

- There exist sequences such that First Fit uses 1.6666…(M) bins.
- Proof:

  - $6M$ items of size $\frac{1}{7} + \epsilon$.
  - $6M$ items of size $\frac{1}{3} + \epsilon$.
  - $6M$ items of size $\frac{1}{2} + \epsilon$.

First-Fit (FF) Lower Bound

- First Fit uses 10M bins, but optimal uses 6M



Best Fit Algorithm (BF)

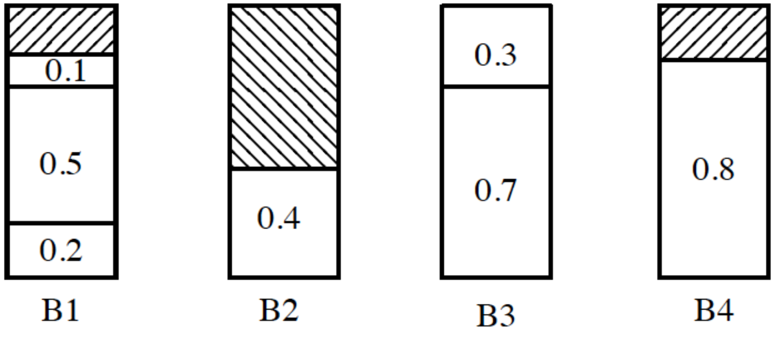
- New item is placed in a bin where it fits the tightest. If it does not fit in any bin, then start a new bin.
- Can be implemented in O(n log n) time, by using a balanced binary tree storing bins ordered by remaining capacity as search key.
- Repeatedly delete the chosen node and re-insert with new remaining capacity.

Zip-Zip Trees

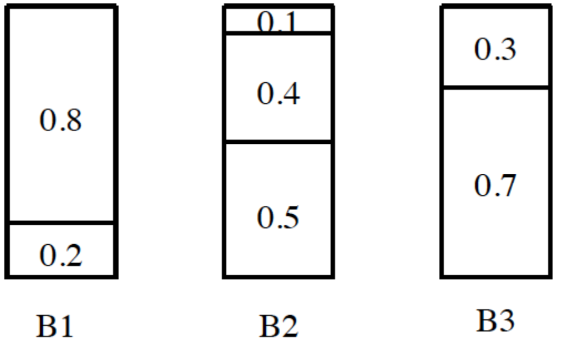
- For Project 2, students are required to implement the Best-Fit Algorithm using the zip-zip tree data structure.

Example for Best Fit (BF)

- I = (0.2, 0.5, 0.4, 0.7, 0.1, 0.3, 0.8)



Best Fit



Optimal Packing

Other Heuristics

- First Fit Decreasing (FFD): First order the items by size, from largest to smallest, then run the First Fit Algorithm.
- Best Fit Decreasing (BFD): First order the items by size, from largest to smallest, then run the Best Fit Algorithm.

Experiments

- It is difficult to experimentally compute approximation ratios.
  - It requires that we know the optimal solution to an NP-hard problem!
- But we can do experiments for a related parameter:
- Define the **waste,** W(A), for a bin-packing algorithm A to be the number of bins that it uses minus the total size of all n items.

Experiments

- We are interested in estimating the waste, W(A), as a function of n and as n grows towards infinity, for random items uniformly distributed in the interval $(0, x)$, for $x < 1$ defined in the assignment and for the following algorithms:
  - A = Next Fit (NF)
  - A = First Fit (FF)
  - A = Best Fit (BF)
  - A = First Fit Decreasing (FFD)
  - A = Best Fit Decreasing (BFD)