Review & Plan
Today’s objectives

- Competitive ratio of Next Fit and Worst Fit
- Lower bound for competitive ratio of any algorithm
- Lower bound for competitive ratio of Best Fit and First Fit
(our beautiful) Bin Packing problem
Bin Packing Problem

- The input is a multi-set of items of various sizes in range \((0,1]\).
- The goal is to pack these items into a minimum number of bins of uniform capacity.
- E.g., \(S = \{0.1, 0.2, 0.2, 0.3, 0.3, 0.4, 0.4, 0.5, 0.5, 0.5, 0.6, 0.8, 0.8, 0.9\}\)
Offline Bin Packing

- The problem is NP-Hard
  - reduction from the partition problem
  - **Partition**: decide whether a multiset $S$ of positive integers can be partitioned into two subsets $S_1$ and $S_2$ s.t.
    \[ \text{sum of the numbers in } S_1 = \text{sum of the numbers in } S_2 = X \]
    \[ S = \{3, 1, 3, 2, 3, 2, 3, 4, 1\} \rightarrow S_1 = \{3, 2, 3, 3\} \quad S_2 = \{1, 3, 2, 4, 1\} \]
  - The answer to partition is yes, if the items/integers can be placed in 2 bins of size $X$

- It is NP-hard to see whether a set can be packed in two or three bins!

- There are algorithms that open $(1 + \epsilon) \text{OPT} + 1$ bins
Online Bin Packing

- The input is a sequence of items of various sizes which are revealed in a sequential, online manner.

- Item sizes are in range \((0, 1]\), and the goal is to pack these items into a minimum number of bins of uniform capacity.

- An online algorithm places items into bins, one by one, with no knowledge of future items.

- Decisions of an online algorithm are irrevocable.
Next Fit Algorithm

- Next Fit: Maintain one open bin at any given time.
  - Place an incoming item in the open bin if there is space there; otherwise close the bin and open a new bin.

< 0.9 0.3 0.8 0.5 0.1 0.1 0.3 0.2 0.4 0.2 0.4 0.5 0.5 0.8 0.6 0.4 0.5 >
**First Fit Algorithm**

- **First Fit**: place an incoming item in the first bin which has enough space for the item.
  - Open a new bin if such bin does not exist.

< 0.9  0.3  0.8  0.5  0.1  0.1  0.3  0.2  0.4  0.2  0.4  0.5  0.5  0.8  0.6  0.4  0.5 ... >
Best Fit Algorithm

Place an incoming item in the bin with the highest level (used space) which has enough space for the item.

< 0.9 0.3 0.8 0.5 0.1 0.1 0.3 0.2 0.4 0.2 0.4 0.5 0.5 0.8 0.6 0.4 0.5 ... >
Best Fit vs First Fit

Question: indicate which one is correct:

(a) The cost of First Fit is always better than Best Fit.
(b) The cost of Best Fit is always better than First Fit.
(c) Politicians are honest people.
(d) None of the above.
Harmonic Algorithm

- Harmonic Algorithm classes: \((\frac{1}{2}, 1], (\frac{1}{3}, \frac{1}{2}], \ldots, (\frac{1}{K}, \frac{1}{K-1}], (0, \frac{1}{K}]\).
- Place members of each class separately from others.

\[
\text{Harmonic } K = 4
\]

\[
< 0.9 \ 0.3 \ 0.8 \ 0.5 \ 0.1 \ 0.1 \ 0.3 \ 0.2 \ 0.4 \ 0.2 \ 0.4 \ 0.5 \ 0.5 \ 0.8 \ 0.6 \ 0.4 \ 0.5 \ \ldots >
\]

\[
\begin{align*}
\text{\textcolor{Green}{0.9}} & \quad \text{\textcolor{LightCoral}{0.8}} & \quad \text{\textcolor{LightCoral}{0.8}} & \quad \text{\textcolor{Orange}{0.6}} & \quad x > \frac{1}{2} \\
\text{\textcolor{LightCyan}{0.4}} & \quad \text{\textcolor{LightCyan}{0.5}} & \quad \text{\textcolor{LightCyan}{0.4}} & \quad \text{\textcolor{Green}{0.3}} & \quad \frac{1}{3} < x \leq \frac{1}{2} \\
\text{\textcolor{LightCyan}{0.5}} & \quad \text{\textcolor{LightCyan}{0.4}} & \quad \text{\textcolor{Green}{0.5}} & \quad \text{\textcolor{Green}{0.5}} & \quad \frac{1}{4} < x \leq \frac{1}{3} \\
\text{\textcolor{Green}{0.3}} & \quad \text{\textcolor{Green}{0.3}} & \quad \text{\textcolor{Green}{0.2}} & \quad 0.2 & \quad x \leq \frac{1}{4}
\end{align*}
\]
Analysis Measures

- Competitive ratio of $A$ is the maximum value of $A(\sigma)/\text{OPT}(\sigma)$ among all sequences $\sigma$.
  - We are interested in the asymptomatic competitive ratio where $\text{OPT}(\sigma)$ is arbitrary large.

- For a sequence of items with total size $S$, the cost of $\text{OPT}$ is at least $S$
  - It is equal to $S$ if opt can pack items in a way that all bins are completely full (which is not always possible)
  - For example, consider $\langle 0.51, 0.51, \ldots, 0.51 \rangle$. The cost of opt is $n$, while $S$ is $0.51n$. 
Competitive Analysis of Next Fit

- Competitive ratio of NextFit is at most 2.
  - In the final packing, total size of items in each two consecutive bins is larger than 1.
  - The cost of NextFit for serving $\sigma$ is smaller than $2S(\sigma)$ where $S(\sigma)$ is the total size of items in $\sigma$.
    - Assume $\text{cost}(\text{NextFit}) = k$
    - Each two consecutive bins have total size $> 1 \rightarrow$ total-size $S(\sigma)$ of items in $\sigma$ is more than $k/2 \rightarrow S > k/2 \rightarrow k < 2S$
  - $\text{OPT}(\sigma) \geq S(\sigma)$: Even when $\text{OPT}$ packs items tightly (with no wasted space), $S(\sigma)$ bins are required.

\[\begin{array}{cccccc}
0.9 & 0.3 & 0.8 & 0.1 & 0.1 & 0.2 \\
0.5 & 0.2 & 0.4 & 0.5 & 0.4 & 0.6 \\
& & & & & 0.5
\end{array}\]

- Competitive ratio of NextFit is at most 2.
Competitive Analysis of Next Fit

- Competitive ratio of NextFit is at least 2.
  - Consider sequence $\sigma = \langle 0.5, \epsilon, 0.5, \epsilon, \ldots \rangle$.
  - The cost of NextFit for serving $\sigma$ is roughly $n/2$ ($n$ is the length of $\sigma$).
  - The cost of $\text{OPT}$ is roughly $n/4$.

\[ \begin{array}{cccccc}
\text{NextFit} & \epsilon & \epsilon & \epsilon & \epsilon & \epsilon \\
5 & 5 & 5 & 5 & 5 & 5 \\
\end{array} \quad \begin{array}{ccc}
\text{OPT} & 5 & 5 & 5 \\
5 & 5 & 5 & & & \\
\end{array} \]

**Theorem**

*Competitive ratio of NextFit is exactly 2.*
Competitive Analysis Of Other Algorithms

- Competitive ratio of First Fit and Best Fit are both 1.7
  - We see the proof later
- Any **Almost Any Fit** strategy has a competitive ratio of 1.7
  - Any Fit algorithm: algorithm which avoid opening new bin when one of the currently open bins have enough space
  - Almost Any Fit algorithm: an AnyFit algorithm which avoid WorstFit strategy (i.e., avoid placing item in the least full bin)
- Competitive ratio of Harmonic is $1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{42} + \ldots \approx 1.691$
Consider the following input
\[ \sigma = \langle 1/2 - \epsilon, 1/2 - \epsilon, \ldots, 1/2 - \epsilon, 1/2 + \epsilon, 1/2 + \epsilon, \ldots, 1/2 + \epsilon \rangle \]

Consider the sub-sequence formed by the first \( m \) items

- Cost of \( \text{OPT} \) for the sub-sequence is \( m/2 \)
- Cost of Alg is \( \alpha m \) for some \( \alpha \) so that \( 1/2 \leq \alpha \leq 1 \).
- Competitive ratio will be at least \( \frac{\alpha m}{m/2} = 2\alpha \).

Consider the whole sequence \( \sigma \)

- Cost of \( \text{OPT} \) is \( m \)
- Alg has opened \( \alpha m \) bins for the first \( m \) items, out of which \( m - \alpha m \) bins have two items (of size \( 1/2 - \epsilon \)).
- So, \( \alpha m - (m - \alpha m) = 2\alpha m - m \) bins have one item
- Alg has to open \( m - (2\alpha m - m) = 2m - 2\alpha m \) new bins for second half.
- The total cost will be \( \alpha m + 2m - 2\alpha m = 2m - \alpha m \)
- The competitive ratio will be at least \( \frac{2m - \alpha m}{m} = 2 - \alpha \).
An Easy Lower Bound

- Consider the following input
  \[ \sigma = \langle 1/2 - \epsilon, 1/2 - \epsilon, \ldots, 1/2 - \epsilon, 1/2 + \epsilon, 1/2 + \epsilon, \ldots, 1/2 + \epsilon \rangle \]
  \[ \text{m items, m items} \]

- To summarize, any sequence that opens \( \alpha m \) bins for the first half has competitive ratio at least \( \max\{2\alpha, 2 - \alpha\} \)

  - The value of \( \max\{2\alpha, 2 - \alpha\} \) is minimized for \( \alpha = 2/3 \), and the competitive ratio will be at least \( 4/3 \).

Theorem

No online bin packing algorithm (deterministic or randomized) can have a competitive ratio better than \( 4/3 \).
An Easy Lower Bound

- **Input:**
  \[ \sigma = \langle \frac{1}{2} - \epsilon, \frac{1}{2} - \epsilon, \ldots, \frac{1}{2} - \epsilon, \frac{1}{2} + \epsilon, \frac{1}{2} + \epsilon, \ldots, \frac{1}{2} + \epsilon \rangle \]

  - \( m \) items  \( \frac{1}{2} - \epsilon \)
  - \( m \) items  \( \frac{1}{2} + \epsilon \)

- No algorithm can be within a ratio less than \( \frac{4}{3} \) of Opt.

- The worst-case sequence that we formed is **fixed** and not **adversarial**

- What if the online algorithm knows the whole input?
  - The lower bound still holds even if the algorithms knows the whole input
Shortcomings of Online Algorithms

- Online algorithms have two shortcomings against $\text{OPT}$
  
  I) **Online constraint**: online algorithms do not know the future requests/items
     
     - We do not know the future items in bin packing, future points in clustering, etc.
     - Often randomization helps to cope with this!

  II) **sequential constraint** online algorithms have to build their solution sequentially
      
      - They cannot change their previous decisions, e.g., an item placed in a bin, two points placed in the same cluster, etc.
      - Even if the algorithms knows the whole input, adversary just needs to decide where to end the input
      - Randomization does not help!
      - This is the main problem of online bin packing algorithms!
A better lower bound

- \( \sigma_1 = \langle 1/2 - \epsilon, \ldots, 1/2 - \epsilon, 1/2 + \epsilon, \ldots, 1/2 + \epsilon \rangle \rightarrow \)
  - \( m \) items
  - competitive ratio > \( 4/3 \)

- \( \sigma_2 = \langle 1/6 - \epsilon, \ldots, 1/6 - \epsilon, 1/3 - \epsilon, \ldots, 1/3 - \epsilon, 1/2 + 2\epsilon, \ldots, 1/2 + 2\epsilon \rangle \rightarrow \)
  - \( m \) items
  - competitive ratio > \( 3/2 \)

- More complicated sequences results better lower bounds

**Theorem**

*No online bin packing algorithm can have a competitive ratio better than 1.54037.*
Weighting Technique for Bin Packing Upper Bounds
Harmonic Algorithm

- Harmonic Algorithm classes: \( \left( \frac{1}{2}, 1 \right], \left( \frac{1}{3}, \frac{1}{2} \right], \ldots, \left( \frac{1}{K}, \frac{1}{K-1} \right], \left(0, \frac{1}{K} \right] \).
- Place members of each class separately from others.

\[
\begin{array}{c}
\text{Harmonic } K = 4 \\
< 0.9 \ 0.3 \ 0.8 \ 0.5 \ 0.1 \ 0.1 \ 0.3 \ 0.2 \ 0.4 \ 0.2 \ 0.4 \ 0.5 \ 0.5 \ 0.8 \ 0.6 \ 0.4 \ 0.5 \ldots >
\end{array}
\]
Weighting Technique in a Nutshell

- Assume we want to prove an algorithm Alg is competitive
- Define a weight $w(x)$ for each item $x$ based on its size
  - General rule: for an item of size $x$, we should have $w(x) \geq x$
  - Weight should be defined so that total weight of items in any bin $B$ of the algorithm (denoted by $w(B)$) is at least 1
    - By ‘any bin’ we mean all bins except possibly a constant number.
    - Assume algorithm opens $k$ bins; we have $k \cdot 1 \leq W$ where $W$ is the total weight of items in the sequence
    - So, we have $\text{Cost}(\text{Alg}) \leq W$ (ignoring a constant no. of bins)
- Find the maximum weight of items that fit in any bin
  - Let $J$ denote that number
  - $\text{OPT}$ has to place items with total weight of $W$ into bins each taking a weight of at most $J$ out of it
  - So, we have $\text{Cost}(\text{Opt}) \geq W/J$

- The competitive ratio of the algorithm will be at most $J$
Weighting Technique in a Nutshell

- Step I: Define a weight function \( w(x) \) for item sizes.
- Step II: Prove that any bin of the online algorithm has weight 1.
- Step III: Prove that it is not possible to place a total weight more than \( J \) in any empty bin.
- The competitive ratio will be \( J \).
Weighting Technique

- Define a weight for each item based on its size
- The weight of an item in class $i$ is $1/i$ when $i < k$
- The weight of an item of size $x$ in class $k$ is $\frac{k}{k-1} x$

Harmonic  $K = 4$

$< 0.9 \ 0.3 \ 0.8 \ 0.5 \ 0.1 \ 0.1 \ 0.3 \ 0.2 \ 0.4 \ 0.2 \ 0.4 \ 0.5 \ 0.8 \ 0.6 \ 0.4 \ 0.5 \ ... >$

$\begin{array}{cccccc}
\text{weight} = 1 & x > \frac{1}{2} \\
0.9 & 0.8 & 0.8 & 0.6 \\
\text{weight} = \frac{1}{2} & \frac{1}{3} < x \leq \frac{1}{2} \\
0.4 & 0.5 & 0.4 \\
\text{weight} = \frac{1}{3} & \frac{1}{4} < x \leq \frac{1}{3} \\
0.5 & 0.5 \\
\text{weight} = \frac{4}{3} x & x \leq \frac{1}{4} \\
0.3 & 0.2 \\
0.2 & 0.1 \\
0.1 & 0.1
\end{array}$
Weighting Technique for Harmonic

- Total weight of items in each bin of Harmonic is at least 1

- Except possibly the current open bin of each class → \( k \) bins
- Bins of type \( i < k \) include \( i \) items, each of weight \( \frac{1}{i} \) → total weight \( i \cdot \frac{1}{i} = 1 \)
- Any bin \( B \) of type \( k \) (except the open bin) has level \( > \frac{k-1}{k} \)
  - Let \( y \) be the first item in the next bin opened → \( y \) did not fit in the level of \( B \) + size of \( y > 1 \) \( \rightarrow \) level of \( B > \frac{k-1}{k} \).

\( (\text{Level of } B) > \frac{k-1}{k} \) \( \rightarrow \) weight of \( x=(k-1)/k \cdot x \) \( \rightarrow \) total weight of items in \( B \) > 1

\[
\begin{array}{cccc}
0.9 & 0.8 & 0.8 & 0.6 \\
\text{x > } & \frac{1}{2} & & \\
\text{weight = 1} & & & \\
\hline
0.5 & 0.4 & 0.5 & 0.4 \\
\frac{1}{3} < x \leq & \frac{1}{2} & & \\
\text{weight = } & \frac{1}{2} & & \\
\hline
0.5 & 0.4 & 0.5 & 0.5 \\
\frac{1}{4} < x \leq & \frac{1}{3} & & \\
\text{weight = } & \frac{1}{3} & & \\
\hline
0.3 & 0.2 & 0.2 & 0.1 \\
x \leq & \frac{1}{4} & & \\
\text{weight = } & \frac{4}{3}x & & \\
\end{array}
\]
Weighting Technique for Harmonic

How much is the maximum total weight of items in a bin of Opt?

- Define density of an item of size \( x \) as \( \frac{w(x)}{x} \)
- Fill the bin with smallest items of classes.
- Use a greedy algorithm that places items with a preference for items of higher density (i.e., larger)!

\[
\begin{align*}
\rho < 2 & \quad \text{weight} = 1 \\
\rho < 3/2 & \quad \text{weight} = \frac{1}{2} \\
\rho < 4/3 & \quad \text{weight} = \frac{1}{3} \\
\rho = k/(k - 1) = 4/3 & \quad \text{weight} = \frac{4}{3}x
\end{align*}
\]
Weighting Technique for Harmonic

How much is the maximum total weight of items in a bin of Opt?

- Next largest item that fits: $1/2 + \epsilon$; weight = 1; size = $1/2 + \epsilon$
- Next item that fits: $1/3 + \epsilon$; weight = $1 + \frac{1}{2}$; size = $1/2 + 1/3 + 2\epsilon = \frac{5}{6} + 2\epsilon$
- Next item that fits: $1/7 + \epsilon$; weight = $1 + \frac{1}{2} + \frac{1}{6}$; size = $\frac{5}{6} + \frac{1}{7} + 3\epsilon = \frac{41}{42} + 3\epsilon$
- Next item that fits: $\frac{1}{43} + \epsilon$; weight = $1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{42}$; size = $\frac{41}{42} + \frac{1}{43} + 4\epsilon$

**Compatibility:**

- $x > \frac{1}{2}$
- $\frac{1}{3} < x \leq \frac{1}{2}$
- $\frac{1}{4} < x \leq \frac{1}{3}$
- $x \leq \frac{1}{4}$

<table>
<thead>
<tr>
<th>weight</th>
<th>$\rho \leq 2$</th>
<th>$\rho \leq 3/2$</th>
<th>$\rho \leq 4/3$</th>
<th>$\rho = (k + 1)/k = 4/3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{4}$</td>
<td>$\frac{4}{3}x$</td>
<td></td>
</tr>
</tbody>
</table>
Weighting Technique for Harmonic

How much is the maximum total weight of items in a bin of Opt?

- So, the greedy approach fills a bin with total weight
  \[ 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{42} + \frac{1}{(42 \cdot 43)} \ldots \approx 1.691 \]
- It turns out that it is not possible to achieve higher weight
  - E.g., if there is no item of class 1, the density and hence total weight will be less than \( \frac{3}{2} \)
  - If there is an item of size \( \frac{1}{2} + \epsilon \) and no item of class 2, there can be at most one item \( \frac{1}{4} + \epsilon \) of class 3, and density of the rest is less than \( \frac{5}{4} \). Weight will be \( 1 + \frac{1}{3} + \frac{5}{4} \cdot \frac{1}{4} \approx 1.64 \) → there is an item of size \( \frac{1}{3} + \epsilon \)

\[
\begin{array}{cccccc}
0.9 & 0.8 & 0.8 & 0.6 \\
\frac{1}{2} \leq x \leq \frac{1}{2} & \frac{1}{4} \leq x \leq \frac{1}{3} & x \leq \frac{1}{4} \\
\text{weight} = 1 & \text{weight} = \frac{1}{2} & \text{weight} = \frac{1}{3} & \text{weight} = \frac{4}{3} x \\
\rho \leq 2 & \rho \leq \frac{3}{2} & \rho \leq \frac{4}{3} & \rho = \frac{k + 1}{k} = \frac{4}{3}
\end{array}
\]
Summary of Weighting Technique for Harmonic

- We define a weight of an item of class $i < k$ to be $1/i$ and the weight of an item of class $k$ to be $\frac{k}{k-1} \cdot x$
- We showed the weight of all bins (except at most $k$ of them) is at least 1 in Harmonic’s packing
- We showed the maximum weight of any bin is at most $J = 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{42} + \ldots$ when $k$ is large enough.
  - We often assume $k$ is a constant around 20.
- The competitive ratio of the algorithm will be at most $J$
Lower Bound: a Nasty Sequence

Consider the following sequence

$$\langle 1/43 + \epsilon, \ldots, 1/43 + \epsilon, 1/7 + \epsilon, \ldots, 1/7 + \epsilon, 1/3 + \epsilon, \ldots, 1/3 + \epsilon, 1/2 + \epsilon, \ldots, 1/2 + \epsilon, \rangle$$

- Harmonic opens $m(1/42 + 1/6 + 1/2 + 1) \approx 1.691m$ bins
- $\text{OPT}$ places one item of each class in each bin $\rightarrow m$ bins
Lower Bound: a Nasty Sequence

- Consider the following sequence

\[
\langle \frac{1}{43} + \epsilon, \ldots, \frac{1}{43} + \epsilon, \frac{1}{7} + \epsilon, \ldots, \frac{1}{7} + \epsilon, \frac{1}{3} + \epsilon, \ldots, \frac{1}{3} + \epsilon, \frac{1}{2} + \epsilon, \ldots, \frac{1}{2} + \epsilon, \rangle
\]

- What about First Fit and Best Fit?
- Both create the same packing as Harmonic!
Summary of Weighting Technique for Harmonic

- We define a weight of an item of class $i < k$ to be $1/i$ and the weight of an item of class $k$ to be $\frac{k}{k-1} \cdot x$

- We showed the the weight of all bins (except at most $k$ of them) is at least 1 in Harmonic’s packing

- We showed the the maximum weight of any bin is at most $J = 1 + \frac{1}{2} + \frac{1}{6} + \frac{1}{42} + \ldots \approx 1.691$ when $k$ is large enough.
  - We often assume $k$ is a constant around 20.

- The competitive ratio of the algorithm will be at most $J$
Competitive Analysis Of First Fit

- Competitive ratio of First Fit is 1.7
  - More precisely, for any sequence $\sigma$, we have $FF(\sigma) \leq \lceil 1.7 \cdot OPT(\sigma) \rceil$.
- Use a weighting method!

\[
W(\alpha) = \begin{cases} 
(6/5) \alpha & \text{for } 0 \leq \alpha \leq 1/6, \\
(9/5) \alpha - 1/10 & \text{for } 1/6 < \alpha \leq 1/3, \\
(6/5) \alpha + 1/10 & \text{for } 1/3 < \alpha \leq 1/2, \\
(6/5) \alpha + 4/10 & \text{for } 1/2 < \alpha \leq 1.
\end{cases}
\]

- Use case analysis to prove:
  - Total weight of all items in a bin of FF is at least 1
  - Total weight of items in any bin is at most 1.7

Diagram from the Bin Packing Survey by Coffman et al.
Any-Fit family of algorithms

- Any **Almost Any Fit** strategy has a competitive ratio of 1.7
  - Any Fit algorithm: algorithm which avoid opening new bin when one of the currently open bins have enough space
  - Almost Any Fit algorithm: an AnyFit algorithm which avoid Worst-Fit strategy (i.e., avoid placing item in the least full bin)

- Proof is similar to First Fit

- Best Fit has a competitive ratio of 1.7.
Analysis Measures
A bin packing algorithm is **bounded-space** if it maintains a constant number of open bins at each given time.

- E.g., Next Fit is bounded-space because it keeps one bin open at each given time.

In many practical applications, you need a constant number of bins opened at each time

- A storage space which packs parcels to trucks with limited parking space

Which one the following algorithms is bounded space:

- (a) First Fit
- (b) Worst Fit
- (c) Best Fit
- (d) Harmonic

Harmonic! recall that it keeps $K$ bins opened at each given time, one from each class.
Analysis Measures

- Compare the performance of an online algorithm $A$ with an optimal offline algorithm $\text{OPT}$:
  - $\text{OPT}$ knows the whole sequence in the beginning.
  - $\text{OPT}$ can change its packing at any time.

- Competitive ratio of $A$ is the maximum value of $A(\sigma)/\text{OPT}(\sigma)$ among all sequences $\sigma$.
  - We are interested in the asymptomatic competitive ratio where $\text{OPT}(\sigma)$ is arbitrary large.

- Average case ratio of $A$ is the expected value of $A(\sigma)/\text{OPT}(\sigma)$.
  - Item sizes are generated randomly and independently, from an identical distribution (typically uniform distribution).

- Expected waste of $A$ is the expected value of $A(\sigma) - \text{OPT}(\sigma)$. 
## Summary of Bin Packing Algorithms

- Average performance ratio, expected waste, and competitive ratios for different bin packing algorithms.
- Competitive ratio of any algorithm is at least 1.54037 BalBek12

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Competitive Ratio</th>
<th>Average Ratio</th>
<th>Expected waste</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Fit (NF)</td>
<td>2</td>
<td>1.3 CoHoSY80</td>
<td>Ω(n)</td>
</tr>
<tr>
<td>Best Fit (BF)</td>
<td>1.7 Johnso73</td>
<td>1 BeJLMM84</td>
<td>Θ(√n log3/4 n) Shor86</td>
</tr>
<tr>
<td>First Fit (FF)</td>
<td>1.7 Johnso73</td>
<td>1 LeiSho89</td>
<td>Θ(n²/3) Shor86 CoJoSW95</td>
</tr>
<tr>
<td>Refined First Fit</td>
<td>1.6 Yao80A</td>
<td>&gt; 1</td>
<td>Ω(n)</td>
</tr>
<tr>
<td>Harmonic (HA)</td>
<td>$T_{\infty} \approx 1.691$ LeeLee85</td>
<td>1.2899 LeeLee85</td>
<td>Ω(n)</td>
</tr>
<tr>
<td>Refined Harmonic</td>
<td>1.635 LeeLee85</td>
<td>1.2824 GuChXu02</td>
<td>Ω(n)</td>
</tr>
<tr>
<td>Modified Harmonic</td>
<td>1.615 BrowLeeLee 89</td>
<td>1.189 RamaTsuga89</td>
<td>Ω(n)</td>
</tr>
<tr>
<td>Harmonic++</td>
<td>1.5888 Seid02</td>
<td>&gt; 1</td>
<td>Ω(n)</td>
</tr>
<tr>
<td>Extreme Harmonic</td>
<td>1.5817 Heydrich, van Stee 15</td>
<td>&gt; 1</td>
<td>Ω(n)</td>
</tr>
<tr>
<td>Advanced Harmonic</td>
<td>1.5783 Balogh et al 18</td>
<td>&gt; 1</td>
<td>Ω(n)</td>
</tr>
</tbody>
</table>
Compromise between Competitive Ratio and Average-case Ratio

Is there an algorithm that performs as well as Best Fit while having better competitive ratio?

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Competitive Ratio</th>
<th>Average Ratio</th>
<th>Expected waste</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Fit (NF)</td>
<td>2</td>
<td>1.3 CoHoSY80</td>
<td>$\Omega(n)$</td>
</tr>
<tr>
<td>Best Fit (BF)</td>
<td>1.7 Johnso73</td>
<td>1 BeJLMM84</td>
<td>$\Theta(\sqrt{n \log^{3/4} n})$ Shor86</td>
</tr>
<tr>
<td>First Fit (FF)</td>
<td>1.7 Johnso73</td>
<td>1 LeiSho89</td>
<td>$\Theta(n^{2/3})$ Shor86 CoJoSW95</td>
</tr>
<tr>
<td>Refined First Fit</td>
<td>1.6 Yao80A</td>
<td>$&gt;1$</td>
<td>$\Omega(n)$</td>
</tr>
<tr>
<td>Harmonic (HA)</td>
<td>$\rightarrow T_\infty \approx 1.691$</td>
<td>1.2899 LeeLee85</td>
<td>$\Omega(n)$</td>
</tr>
<tr>
<td>Refined Harmonic</td>
<td>1.635 LeeLee85</td>
<td>1.2824 GuChXu02</td>
<td>$\Omega(n)$</td>
</tr>
<tr>
<td>Modified Harmonic</td>
<td>1.615</td>
<td>1.189</td>
<td>$\Omega(n)$</td>
</tr>
<tr>
<td>Harmonic++</td>
<td>1.588 Seiden 02</td>
<td>$&gt;1$</td>
<td>$\Omega(n)$</td>
</tr>
<tr>
<td>Extreme Harmonic</td>
<td>1.5817 van Stee 15</td>
<td>$&gt;1$</td>
<td>$\Omega(n)$</td>
</tr>
<tr>
<td>Harmonic Match</td>
<td>$\rightarrow T_\infty \approx 1.691$</td>
<td>1</td>
<td>$\Theta(\sqrt{n \log^{3/4} n})$</td>
</tr>
<tr>
<td>Refined Harmonic Match</td>
<td>1.635</td>
<td>1</td>
<td>$\Theta(\sqrt{n \log^{3/4} n})$</td>
</tr>
</tbody>
</table>
Harmonic Match

- An extension of the classes of Harmonic algorithm.
- Apply a relaxed variant of Best Fit on items of each class.

\[
\frac{1}{3} < x \leq \frac{1}{2} \\
\frac{1}{4} < x \leq \frac{1}{3} \\
\frac{1}{5} < x \leq \frac{1}{4} \\
\frac{1}{k+1} < x \leq \frac{1}{k}
\]

- \(i = 1\) \(\frac{1}{3} < x \leq \frac{1}{2} \quad \leftrightarrow \quad \frac{1}{2} < x \leq \frac{2}{3}\)
- \(i = 2\) \(\frac{1}{4} < x \leq \frac{1}{3} \quad \leftrightarrow \quad \frac{2}{3} < x \leq \frac{3}{4}\)
- \(i = 3\) \(\frac{1}{5} < x \leq \frac{1}{4} \quad \leftrightarrow \quad \frac{3}{4} < x \leq \frac{4}{5}\)
- \(i = k - 1\) \(\frac{1}{k+1} < x \leq \frac{1}{k} \quad \leftrightarrow \quad \frac{k-1}{k} < x \leq \frac{k}{k+1}\)
- \(i = k\) \(x \leq \frac{1}{k+1} \quad \leftrightarrow \quad x > \frac{k}{k+1}\)
Harmonic Match Algorithm

For placing an item of size $x$:

- If $x > 0.5$, open a new bin.
- If $x \leq 0.5$:
  - Use Best Fit strategy to place $x$ together with an item $y > 0.5$ of the same class.
  - If no such $y$ exists, place $x$ together with items of the same class using Next Fit strategy.

\[
< 0.62 \ 0.28 \ 0.3 \ 0.4 \ 0.79 \ 0.34 \ 0.71 \ 0.21 \ 0.42 \ 0.33 \ 0.22 \ 0.27 \ 0.21 \ 0.24 >
\]
Harmonic Match vs Harmonic

- Packing of Harmonic Match is the same as Harmonic except that some items are ‘removed’ from Harmonic packing.

< 0.62 0.28 0.3 0.4 0.79 0.34 0.71 0.21 0.42 0.33 0.22 0.27 0.21 0.24 >
Competitive Analysis

- Harmonic is a **monotone** algorithm.
  - Removing an item does not increase the number of bins opened by Harmonic.

**Theorem**

*For any sequence, the number of bins opened by Harmonic Match is no more than that of Harmonic.*

- Competitive ratio of Harmonic Match is as good as Harmonic, i.e., $T_\infty \approx 1.691$.
- Unlike Harmonic, First Fit and Best Fit are **anomalous** in the sense that removing items might increase the cost of these algorithms.
Consider **upright matching** problem.

- We are given $n$ points in a $1 \times 1$ coordinate.
- The goal is to match a maximum number of $\ominus$ with $\oplus$ points.
- Each $\ominus$ point can be matched only to $\oplus$ points on its upright position.
- Labels and positions of points are i.i.d. random variables.

**Greedy algorithm:** process $\ominus$ points one by one from top to bottom.

- Match each $\ominus$ item with the left-most unmatched $\oplus$ item on above and right of it.

It is known that Greedy matches all points except an expected number of $\Theta(\sqrt{n} \log^{3/4} n)$ points.
Reduction of bin packing to upright matching

Consider a bin packing sequence of length \( n \) with item sizes randomly distributed in \((0, 1]\).

Create an instance of upright matching:

- Items are mapped to points in the square.
- An item of size \( \alpha > 0.5 \) gets an \( \oplus \) label and \( x \)-coordinate \( 2(1 - \alpha) \).
- An item of size \( \alpha \leq 0.5 \) gets an \( \ominus \) label and \( x \)-coordinate \( 2\alpha \).
- \( y \)-coordinate of the item at index \( i \) is set randomly in \([i/n], [i/n] \)

E.g., \( \sigma = \langle 0.53, 0.69, 0.21, 0.78, 0.4 \rangle \)
Reduction of bin packing to upright matching

- Points receive random labels (with a chance of 0.5 an item is larger than 0.5 (⊕) and with a chance of 0.5 it is ≤ 0.5 (⊖)).
- Points $x$-coordinates are random
  - for an $⊕$ point, item size $x$ is random in $U(0.5, 1]$ and hence $2(1 - x)$ is random in $U[0, 1)$
  - for an $⊖$ point, item size $x$ is random in $U(0, 0.5]$ and hence $2(x)$ is random in $U(0, 1]$
- Points $y$-coordinates are random
  - Exactly one point is distributed randomly in the interval $U[i/n, (i + 1)/n)$ on the $y$-axis.
Reduction of bin packing to upright matching

- So, an instance of bin packing can be reduced to upright matching

- What is the equivalent of greedy algorithm?
  - An $\oplus$ point $y$ appears on the right of $x$ if sum of items $x$ and $y$ is less than 1.
    - $y$ is on right of $x \rightarrow 2(1 - y) \geq 2x \rightarrow x + y \leq 1$

- Greedy matches each $\ominus$ point $p$ (item $x \leq 0.5$) with the leftmost $\oplus$ point (largest item $y$ so that $> 0.5$) that appears above (i.e., $y$ is before $x$ in the sequence) and on the right of $p$ (i.e., $x + y \leq 1$).
Reduction of bin packing to upright matching

Greedy is equivalent to Almost Best Fit:

- If $x > 1/2$, open a new bin for $x$.
- If $x \leq 1/2$, place $x$ with an item $y \geq 0.5$ which best fits $x$ (i.e., largest such $y$ so that $x + y \leq 1$).
- If no such $y$ exists, open a new bin for $x$.

Almost Best Fit is similar to Best Fit except that:

- It closes a bin right after it is opened if the bin is opened by an item of size $\leq 1/2$.
- It closes a bin as soon as two items are placed in it.

For any sequence, the cost of Best Fit is at most equal to Almost-Best-Fit
Average-case analysis of Best Fit

- Number of unmatched point by greedy is expected to be $\Theta(\sqrt{n} \log^{3/4} n)$.

- So, the number of bins with 1 item in Almost Best Fit (ABF) is at most $\Theta(\sqrt{n} \log^{3/4} n)$ on expectation.

- The cost of ABF is less than $n/2 + \Theta(\sqrt{n} \log^{3/4} n)$ for a sequence of length $n$ on expectation.

- The cost of $\text{OPT}$ is expected to be at least $n/2$ (since half items are expected to be larger than 0.5).

- Average case ratio of ABF (and hence BF) is at most

$$\frac{n/2 + \Theta(\sqrt{n} \log^{3/4} n)}{n/2} \approx 1$$

for large values of $n$

- Expected waste of ABF (and hence BF) is at most

$$E(\text{ABF}(\sigma) - \text{OPT}(\sigma)) = n/2 + \Theta(\sqrt{n} \log^{3/4} n) - n/2 = \Theta(\sqrt{n} \log^{3/4} n)$$
The average-case analysis for Harmonic Match is similar to Best Fit; we repeat the same analysis for each class separately.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Competitive Ratio</th>
<th>Average Ratio</th>
<th>Expected waste</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Fit (NF)</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best Fit (BF)</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
<tr>
<td>First Fit (FF)</td>
<td>1.7 Johnso73</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Experimental Evaluation

- Experimental average-case performance of online algorithms for different distributions.

![Bar chart showing performance of online algorithms for different distributions.](chart.png)
Discussion

- In practical scenarios, we should have an eye on both worst-case and average-case performance.
- Harmonic algorithms do well in the worst-case (competitive ratio) but have poor average-case performance.
- Another family of algorithms, e.g., Sum-of-Square algorithm, have a good average-case performance (better than Best Fit) but have a poor competitive ratio.
- There is not necessarily a trade-off between worst-case and average-case performance in bin packing.
- We can devise algorithms that are good in both senses $\rightarrow$ Harmonic-match.
An application of Bin Packing: Fault-tolerant Server Consolidation
Fault-tolerant Bin Packing
(Server Consolidation in the Cloud)

- Bins represent **servers** and items are **clients** (e.g., databases tenant o a movie on NetFlix).
- Server might fail and it should not interrupt the service (clients be should always available).
- Given a sequence of items, place two replicas of each item in different servers
  - Each replica of an item with **load** $x$ has a load of $x/2$.
  - Think of load as the number of people who watch a NetFlix movie; so each replica requires half bandwidth
- In case of a server’s failure, the load of each replica is redirected to the server that hosts its partner.
Valid Solutions

- Consider sequence
  \[ \langle a = 0.6, b = 0.3, c = 0.6, d = 0.8, e = 0.1, f = 0.4 \rangle. \]

- A valid packing:

- An invalid packing:
Mirroring Algorithms

- Consider two types of replicas (blue and red), and apply Best Fit for each type separately
  - Assume a capacity of 1/2 for the bins (why?)
  - The level of a bin should never be more than 0.5 (otherwise there will be an overflow in case of a bin failure)
- Consider sequence
  \(\langle a = 0.6, b = 0.3, c = 0.6, d = 0.8, e = 0.1, f = 0.4 \rangle.\)
Mirroring Algorithm

Question: the competitive ratio of the mirroring algorithm is at least:

- (a) 1.5  (b) 1.7  (c) 2  (d) infinity
Mirroring Algorithm

- Mirroring algorithms are not better than 2-competitive
- Consider sequence \( \langle 2\varepsilon_1, 2\varepsilon_2, \ldots, 2\varepsilon_n \rangle \)
- \( \text{OPT} \) can place all items so that all bins are almost full
  - Each two bin share at most one item!
Horizontal Harmonic (HH) Algorithm

- Like Harmonic, define *classes* for replicas.
  - $\left(\frac{1}{3}, \frac{1}{2}\right], \left(\frac{1}{4}, \frac{1}{3}\right], \ldots, \left(\frac{1}{K}, \frac{1}{K-1}\right], (0, \frac{1}{K}]$ (E.g., $K = 30$).
- Treat members of each class separately.
  - No two bins share more than one replica.
Horizontal Harmonic (HH) Algorithm

- Consider sequence \( \langle a_1, a_2, \ldots, a_m \rangle \) of replicas of the same class (E.g., for class 3, replicas lie in the range \((1/5, 1/4]\)).

- Place \( i \) blue replicas of class \( i < K \) in the same bin.

- Place red replicas whose partners are in the same bin in different bins.
  
  - This ensures a valid packing.
Horizontal Harmonic (HH) in a Nutshell

- Class $i < k$: place $i$ replicas of the range $\left(\frac{1}{i+2}, \frac{1}{i+1}\right]$ in the same bin.
  - No two bins share more than one replica
- Use mirroring for items of class $k$

![Diagram showing bin packing for different intervals](image-url)
Analysis of Horizontal Harmonic

Summary of weighting argument to prove an algorithm has c.r. at most $J$:

- Step I: Define a weight function $w(x) \geq x$ for an item of size $x$
- Step II: Prove that any bin of the online algorithm has weight 1.
- Step III: Prove that it is not possible to place a total weight more than $J$ in any empty bin

1/3 < x ≤ 1/2  
1/4 < x ≤ 1/3  
1/5 < x ≤ 1/4  
x ≤ 1/5
Analysis of Horizontal Harmonic

- Define the weight of an item of class $i$ to be $1/i$.
  - Weight of the bins of that type in HH packing will be 1.
- Define the weight of an item of class $k$ to be $\frac{2(k+1)}{k-1}$.
  - Items of class $k$ are no larger than $1/(k+1)$.
  - Level of these bins is at least $1/2 - 1/(k+1) = \frac{k-1}{2(k+1)}$.
- Steps I and II are done.
Analysis of Horizontal Harmonic

- Step III: how much the weight of an optimal bin can be?
  - Similar to the case of Harmonic, it is better to fill the bin with the larger replicas → They have higher density!
  - The reserved space should be no less than the largest replica!
- Using case analysis we can show the maximum weight for large $k$ is

\[w(1/3+\epsilon)+w(1/4+\epsilon)+w(1/13+\epsilon)+\ldots = 1+1/2+1/11+\ldots \approx 1.597\]
Analysis of Horizontal Harmonic

Theorem

**Competitive ratio of Horizontal Harmonic is at most 1.597 for large values of $K$.**

- This bound is tight!
A ‘real-world’ application of bin packing

Mirroring algorithms used in practice have c.r. of 2

Horizontal Harmonic packs replicas more tightly and has a c.r. of at most 1.597

Real-world implementation of Horizontal-Harmonic shows promising performance.
  - The algorithms works well in both worst-case and average-case.
Application II: Renting Servers in the Cloud
Buying vs. Renting

- When you **buy** servers, the goal is to minimize the total number of opened (purchased) servers.

- When you **rent** servers, the goal is to minimize the total **time** you have rented servers.
  - Each item has an arrival and a departure time.
  - The difference is the **Length** of the item.
First Fit Algorithm

- Apply First Fit algorithm to place items.
- **Release** the bin when all items depart.
- The cost of First Fit is \((20-1) + (20-3) = 36\) (assuming no other item arrives till time 20).

\(< a = (0.3, 1, 7), b = (0.4, 2, 7), c = (0.4, 3, 7), d = (0.5, 4, 7), e = (0.3, 5, 20), f = (0.1, 6, 20), ...>\)

Time: 7
Previous Results

- No Any-Fit algorithm can be better than $\mu$ competitive
  - $\mu$ is the ratio between the length of the largest and the smallest item LiTang14.
- First Fit is at most $2\mu + 13$-competitive LiTang14.
- Best Fit is not competitive.
Next Fit Algorithm

- Apply Next Fit algorithm to place items.
- **Release** the bin when all items depart.
- The cost of Next Fit is $(7-1) + (7-3) + (20-5) = 25$ (assuming no other item arrives till time 20).

\(< a = (0.3, 1, 7), b = (0.4, 2, 7), c = (0.4, 3, 7), d = (0.5, 4, 7), e = (0.3, 5, 20), f = (0.1, 6, 20), \ldots >\)

Time: 7
New Results for Renting Servers

- No algorithm can be better than $\mu$-competitive.
- Next Fit is at most $2\mu + 1$-competitive.
- If the value of $\mu$ is known, one can achieve a $\mu + 2$-competitive algorithm.
Boosting Average-Case Performance

- On average, Best Fit is still better than Next Fit and First Fit.
- We introduce a new algorithm Move To Front.
  - An Any Fit algorithm that applies after placing an item, moves the bin to the front.
We introduce a new algorithm Move To Front.

An Any Fit algorithm that applies after placing an item, moves the bin to the front.

Intuitively, items that arrive together are more likely to depart at the same time.

< a = (0.3, 1, 7), b = (0.4, 2, 7), c = (0.4, 3, 7), d = (0.5, 4, 7), e = (0.3, 5, 20), f = (0.1, 6, 20), …>

Time: 6
Average-Case Performance of Online Algorithms

- Competitive ratio of Move To Front is at most $6\mu + 7$.
- On average (sequences with uniform size and length), Move To Front outperforms all algorithms.
Advice Model for Online Problems

- Under the advice model, an online algorithm receives $b$ bits of advice from an benevolent offline oracle.
- The advice bits are available since the beginning.
- There is a compromise between the number of advice bits ($b$) and quality of algorithms (e.g., their competitive ratio).
Relevant Questions

- For a fixed sequence of fixed length $n$:
  - How many bits of advice are required (sufficient) to achieve an optimal solution?
  - How many bits of advice are sufficient to outperform all online algorithms?
  - How good the competitive ratio can be with an advice of linear/sublinear size?
Optimal Solution with Advice

**Theorem**

*For any sequence of length* $n$, advice of size $O(n \log k)$ *is sufficient to achieve an optimal solution, where* $k$ *is number of bins in an optimal packing.*

- What advice encodes?
  - For each item, it encodes the bin that it is packed to in an optimal packing

- What is the advice size?
  - For each item, we require $O(\log k)$ bits to encode the target bin. In total, $O(n \log k)$ bits suffice.

- How the algorithm works with the given advice?
  - It packs each item in the same bin as $\text{Opt}$ does.

- Why it is optimal? The resulting packing is similar to $\text{Opt}$. 
Optimal Solution with Advice

**Theorem**

For any sequence of length \( n \), \( \Omega(n \log k) \) bits of advice are required to achieve an optimal solution, where \( k \) is number of bins in an optimal packing.

- Let \( \sigma = \langle \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \ldots, \frac{1}{2^{n-k+1}}, u_1, u_2, \ldots, u_k \rangle \)
- Each of the first \( n - k \) items can be packed in any of the \( k \) bins
  - The summation of all of them is less than 1/2.

**Question:** in how many ways the first \( n - k \) items can be packed?

- (a) \( 2^{n-k} \)
- (b) \( \frac{2^{n-k}}{k!} \)
- (c) \( k^{n-k} \)
- (d) \( \frac{k^{n-k}}{k!} \)

- There will be \( \frac{k^{n-k}}{k!} \) different packings!
Optimal Solution with Advice

\[ \sigma = \left\langle \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \ldots, \frac{1}{2^{n-k+1}}, u_1, u_2, \ldots, u_k \right\rangle \]

- There will be \( \frac{k^{n-k}}{k!} \) different packings!
- For each packing the last \( k \) items (\( u_i \)'s) fill the empty space for each bin

- \( \sigma_1 = \left\langle \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, 1 - \frac{1}{4}, 1 - \frac{1}{8}, 1 - \frac{1}{16} \right\rangle \)
- \( \sigma_2 = \left\langle \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, 1 - \frac{1}{4} - \frac{1}{16}, 1 - \frac{1}{8}, 1 - \frac{1}{16} \right\rangle \)
- \( \sigma_3 = \left\langle \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, 1 - \frac{1}{4} - \frac{1}{8}, 1 - \frac{1}{16}, 1 \right\rangle \)
- \( \sigma_4 = \left\langle \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, 1 - \frac{1}{4} - \frac{1}{8} - \frac{1}{16}, 1, 1 \right\rangle \)
Optimal Solution with Advice

$$\sigma = \langle \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \ldots, \frac{1}{2^{n-k+1}}, u_1, u_2, \ldots, u_k \rangle$$

- There will be $\frac{k^{n-k}}{k!}$ different sequences!
  - All start with the same prefix of length $n - k$
- For each sequence, an optimal algorithm should pack the first $n - k$ items differently from others.
  - Each sequence requires an advice tailored for itself
- $\frac{k^{n-k}}{k!}$ different advice strings are required
  - $\log \frac{k^{n-k}}{k!} \approx (n - 2k) \log k$ advice bits are required.
Optimal Solution with Advice

**Theorem**

To achieve an optimal packing, it is sufficient to receive $n \lceil \log \text{Opt}(\sigma) \rceil$ bits of advice. Moreover, any deterministic online algorithm requires at least $(n - 2 \text{Opt}(\sigma)) \log \text{Opt}(\sigma)$ bits of advice to achieve an optimal packing.
Optimal Solution with Advice

- Assume the sequence is formed by \( m = o(n) \) distinct items which have size larger than a fixed value \( \epsilon \).

**Theorem**

*It is sufficient to read \( O(m \log n) \) bits of advice to achieve an optimal packing.*

- For each item \( x \) encode its frequency in the input sequence!
  - This requires \( O(\log n) \) bits

- The advice encodes the whole multi-set that forms the input in \( O(m \log n) \) bits.

- Given the multi-set, pack it, optimally, using an offline algorithm before starting to serve the input.

- When an item is revealed, place it into its reserved space in the offline packing!
The Idea Behind the Lower Bound

**Theorem**

To achieve an optimal solution least $\Omega(m \log n)$ bits of advice are required.

- Consider a subclass of sequences which start by $n/2$ items of size $\epsilon$.
  - Let $X$ denote the number of ways that these $n/2$ items can be packed
    - $X$ will be at least $\left(1 + \frac{n}{(m-1)(m-2)}\right)^{m-3}$ (we skip the proof here)
The Idea Behind the Lower Bound

- For each partial packing, complete the sequence with items which fill the empty spaces.
- Each sequence requires a distinct advice, and consequently an advice of size $\log X$ is required.
- At least $\log(1 + \frac{n}{(m-1)(m-2)})^{m-3} = m \log n + o(\log n)$ bits are required.

Example:
$n=30$, $m=6$
(bin capacities scaled up by 12)

sequence: $\langle 1^{(5)} 11 11 11 11 10 10 9 8 12^{(7)} \rangle$

Example:
sequence: $\langle 1^{(5)} 11 11 11 11 11 11 8 8 12^{(6)} \rangle$...
Optimal Solution with Advice

Theorem

It is sufficient to read $O(m \log n)$ bits of advice to achieve an optimal packing.
At least $\Omega(m \log n)$ bits of advice are required to achieve an optimal solution.
Breaking the Lower Bound

- How many bits of advice are sufficient to achieve a competitive ratio better than all online algorithms?
  - We have seen before that no online algorithm can have a competitive ratio better than 1.54.
- \(O(\log n)\) bits of advice is sufficient to achieve competitive ratio 1.5.
- Divide items into four groups based on their sizes:
  - **Huge** items: size larger than \(2/3\)
  - **critical** items: size in range \((1/2, 2/3]\)
  - **mini** item: size in range \((1/3, 1/2]\)
  - **tiny** items: size smaller than \(1/3\)
Breaking the Lower Bound

- Receive number of critical items with $O(\log n)$ bits of advice
- Consider **ReserveCritical** algorithm:
  - At the beginning, reserve a space of size $2/3$ for critical items
  - **Huge** items: open a new bin
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin
  - **tiny** items: apply First-Fit to place in bins with critical or other tiny items
ReserveCritical Algorithm

- At the beginning, reserve a space of size 2/3 for critical items
  - **Huge** items: open a new bin (no other item goes there)
  - **critical** items: place in a reserve space
  - **mini** item: place two of them in the same bin (no other item goes there)
  - **tiny** items: apply FF to place in bins with critical or other tiny items

\[ \sigma = \langle 0.3 \ 0.9 \ 0.6 \ 0.5 \ 0.1 \ 0.1 \ 0.56 \ 0.4 \ 0.3 \ 0.45 \ 0.8 \ 0.51 \ 0.41 \ 0.2 \ 0.1 \ 0.37 \ 0.3 \rangle \]
ReserveCritical algorithm

Theorem

Competitive ratio of ReserveCritical is at most 1.5.

- With $O(\log n)$ bits of advice, one can achieve a competitive ratio of 1.5
- Can we improve this?
RedBlue Algorithm (sketch)

- Instead of receiving the number of critical items in $O(\log n)$ bits, receive the ratio between critical and tiny bins in the final packing of ReserveCritical.

- Treat Huge and Mini items as before.

- Place a critical item in the reserved space of a critical bin; if no reserve space exists, open a new bin and declare it as critical.

- Place a tiny item in non-reserved space of critical bins (using FF).
  - If no such critical bin exists, place it in the available space of a tiny bin (using Next Fit).
  - If no suitable tiny bin exists, open a new bin.
  - Declare the new bin to be a critical or a tiny bin so that the ratio between the number of these bins becomes closer to the ratio received in advice.
RedBlue Algorithm (sketch)

- If the ratio between critical and tiny bins is encoded using $k$ bits of advice, RedBlue algorithm has a competitive ratio of at most $1.5 + \frac{15}{2^{k/2+1}}$

Theorem

*With constant number of bits of advice, one can achieve a competitive ratio of (almost) 1.5.*

- Can we do better?
In fact, with a more complicated argument, we can show that with advice of constant size, one can achieve a competitive ratio of 1.47.

Idea: pack items of size larger than 1/3 separately from the rest.

- How advice can help in packing items of size larger than 1/3?
Power of Advice of Constant Size

- It is often useful to think of algorithms that ‘complement’ each other.
- Assume all items are larger than 1/3:
  - **Sbf:** All small items (< 1/2) are packed according to BestFit, and each large item (≥ 1/2) is placed in a new bin.
  - **Lbf:** All large items are packed according to BestFit, and each small item is placed in a new bin.

\[ \sigma = \langle 0.45, 0.6, 0.75, 0.34, 0.40, 0.56, 0.35, 0.55, 0.50 \rangle \]

An Optimal packing

![Optimal packing diagram](image-url)
Theorem

When all items are larger than 1/3, the better algorithm among Sbf and Lbf has a competitive ratio of 1.39.

- With only one bit of advice, one can achieve a competitive ratio of 1.39 (when all items are larger than 1/3).
- Think of the two algorithms as ‘parallel algorithms’
- This algorithm is used as a subroutin for an algorithm which gets a competitive ratio of 1.47 with constant advice! (details skipped here).
Lower bound

- Advice of size $\Omega(n)$ is required to achieve an algorithm with c.r. $\leq \frac{9}{8}$
- A reduction from binary guessing problem
- Consider

$$\sigma = \langle 0.5 + \epsilon, \ldots, 0.5 + \epsilon, a_1, a_2, a_3, \ldots, a_{2m}, b_1, \ldots, b_m \rangle$$

- $m$ green items
- White items in range $(1/3, 1/2]$ complements of smaller white items

- The algorithm should 'guess' whether each white item is among the larger half of smaller half of white items!
Lower bound

\( \langle 0.51, \ldots, 0.51, 0.42, 0.37, 0.4, 0.39, 0.38, 0.385, 0.388, 0.386, 0.63, 0.62, 0.615, 0.614 \rangle \)

- \( m \) green items
- \( 2m \) white items
- red complements

- Guess if an item is among smaller or larger half of white items
  - open a new bin for smaller half of white items (in anticipation of their complements coming in the future)
  - for the larger half of white items, put them with green items

- The ‘type’ (being in smaller or larger half) of the white item cannot be revealed from knowing types of previous white items

- For any four mistakes in guessing, at least 1 extra bin is opened
Lower bound

**Theorem**

In order to achieve a competitive ratio better than $9/8$, advice of linear size is required.

- This result can be improved to show that for a competitive ratio better than $4 - 2\sqrt{2} \approx 1.172$, a linear number of bits are required.
Bin Packing & Advice: A General Picture

- No advice: best upper and lower bounds by [?] and [?].
- With $\Theta(n \log N)$ bits, one can achieve an optimal solution ($N$ is the cost of $\text{OPT}$) [?].
- With $\Theta(\log n)$ bits, one can achieve a competitive ratio of 1.5 (better than all online algorithms) [?].
- With linear number bits, one can achieve a competitive ratio of 4/3 [?].
- For a competitive ratio better than 9/8, a linear number of bits are required [?].
- With linear number bits, one can achieve a competitive ratio of 1.0 [?].
- With $k \geq 4$ bits, one can get a competitive ratio of $1.5 + \frac{15}{2^{k/2}+1}$ [?].
- With $\Theta(1)$ bits, one can get a competitive ratio of 1.4702 [?].
- For a competitive ratio better than 7/6, a linear number of bits are required [?].
- For a competitive ratio better than $4 - 2\sqrt{2} \approx 1.172$, a linear number of bits are required [?].