COMP 7720 - Online Algorithms

Online Bin Packing

Shahin Kamali

Lecture 15 - Oct. 25, 2018

University of Manitoba
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., \(S = \{0.1, 0.2, 0.2, 0.3, 0.3, 0.4, 0.4, 0.4, 0.5, 0.5, 0.5, 0.5, 0.6, 0.8, 0.8, 0.9\}\)
The problem is NP-Hard

- reduction from the partition problem
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.
    - sum of the numbers in $S_1 = \text{sum of the numbers in } S_2 = X$
    - $S = \{3, 1, 3, 2, 3, 2, 3, 3, 4, 1\} \rightarrow S_1 = \{3, 2, 3, 3\} \quad S_2 = \{1, 3, 2, 4, 1\}$

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

There are algorithms that open $(1 + \epsilon) \cdot \text{Opt} + 1$ bins assuming there is a constant number of item sizes, it is possible to pack them optimally in polynomial time
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.
    - sum of the numbers in $S_1 = \sum$ of the numbers in $S_2 = X$
    - $S = \{3, 1, 3, 2, 3, 2, 3, 3, 4, 1\} \rightarrow S_1 = \{3, 2, 3, 3\} \; S_2 = \{1, 3, 2, 4, 1\}$
    - The answer to partition is yes, if the items/ integers can be placed in 2 bins of size $X$
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.
    - sum of the numbers in $S_1 = \text{sum of the numbers in } S_2 = X$
    - $S = \{3, 1, 3, 2, 3, 2, 3, 3, 4, 1\} \rightarrow S_1 = \{3, 2, 3, 3\} \quad S_2 = \{1, 3, 2, 4, 1\}$
    - The answer to partition is yes, if the items/integers can be placed in 2 bins of size $X$
Offline Bin Packing

- The problem is NP-Hard
  - reduction from the partition problem
  - It is NP-hard to see whether a set can be packed in two or three bins!

- There are algorithms that open \((1 + \epsilon) \OPT + 1\) bins
- Assuming there is a constant number of item sizes, it is possible to pack them optimally in polynomial time
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.
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.

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

```
< 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 >
```

- 0.9
- 0.8
- 0.3
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>

![Bar chart illustrating Next Fit algorithm with items of different sizes placed into bins.](chart)
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 >
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 >
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 >
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 >
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.

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

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

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

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

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 ... >
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.

Harmonic \quad K = 4

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

Harmonic \(K = 4\)

\(< 0.9 0.3 0.8 0.5 0.1 0.1 0.3 0.2 0.4 0.2 0.4 0.5 0.5 0.8 0.6 0.4 0.5 \ldots >\)

\(x > \frac{1}{2}\)

\(\frac{1}{3} < x \leq \frac{1}{2}\)

\(\frac{1}{4} < x \leq \frac{1}{3}\)

\(x \leq \frac{1}{4}\)
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.

\[
\begin{array}{c}
\text{Harmonic} \quad K = 4 \\
\downarrow \\
< 0.9 \ 0.3 \ 0.8 \ 0.5 \ 0.1 \ 0.1 \ 0.3 \ 0.2 \ 0.4 \ 0.2 \ 0.4 \ 0.5 \ 0.5 \ 0.8 \ 0.6 \ 0.4 \ 0.5 \ldots > \\
\end{array}
\]
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.

Harmonic \(K = 4\)

\[
\begin{array}{c}
\downarrow \\
< 0.9 0.3 0.8 0.5 0.1 0.1 0.3 0.2 0.4 0.2 0.4 0.5 0.5 0.8 0.6 0.4 0.5 \ldots > \\
\end{array}
\]

\[
\begin{array}{ccc}
0.9 & 0.8 \\
x > \frac{1}{2} \\
\frac{1}{3} < x \leq \frac{1}{2} \\
\frac{1}{4} < x \leq \frac{1}{3} \\
x \leq \frac{1}{4} \\
0.3
\end{array}
\]
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.

Harmonic \[ K = 4 \]

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

Harmonic \(K = 4\)

\(< 0.9\ 0.3\ 0.8\ 0.5\ 0.1\ 0.1\ 0.3\ 0.2\ 0.4\ 0.2\ 0.4\ 0.5\ 0.5\ 0.8\ 0.6\ 0.4\ 0.5 \ldots >\)

\[\begin{align*}
0.9 & \quad 0.8 & \quad 0.5 & \quad 0.3 & \quad 0.1 \\
\frac{1}{2} & \quad \frac{1}{2} & \quad \frac{1}{3} & \quad \frac{1}{3} & \quad \frac{1}{4}
\end{align*}\]
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.

Harmonic \(K = 4\)

\(< 0.9 0.3 0.8 0.5 0.1 0.1 0.3 0.2 0.4 0.2 0.4 0.5 0.5 0.8 0.6 0.4 0.5 \ldots >\)
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.

Harmonic  \( K = 4 \)

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

**Harmonic**  
\(K = 4\)

\(< 0.9\ 0.3\ 0.8\ 0.5\ 0.1\ 0.1\ 0.3\ 0.2\ 0.4\ 0.2\ 0.4\ 0.5\ 0.5\ 0.8\ 0.6\ 0.4\ 0.5\ \ldots >\)
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.

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

\[
< 0.9 \ 0.3 \ 0.8 \ 0.5 \ 0.1 \ 0.1 \ 0.3 \ 0.2 \ 0.4 \ 0.2 \ 0.4 \ 0.5 \ 0.5 \ 0.8 \ 0.6 \ 0.4 \ 0.5 \ ... >
\]
Harmonic Algorithm

Harmonic Algorithm classes: \( \left( \frac{1}{2}, 1 \right], \left( \frac{1}{3}, \frac{1}{2} \right], \ldots, \left( \frac{1}{K}, \frac{1}{K-1} \right], (0, \frac{1}{K}] \).

- Place members of each class separately from others.

Harmonic Algorithm

\[ K = 4 \]

\[ < 0.9 \ 0.3 \ 0.8 \ 0.5 \ 0.1 \ 0.1 \ 0.3 \ 0.2 \ 0.4 \ 0.2 \ 0.4 \ 0.5 \ 0.5 \ 0.8 \ 0.6 \ 0.4 \ 0.5 \ ... > \]
Harmonic Algorithm

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

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

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

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

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

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

Harmonic Algorithm

\[
K = 4
\]

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

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

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

- \(x > \frac{1}{2}\)
- \(\frac{1}{3} < x \leq \frac{1}{2}\)
- \(\frac{1}{4} < x \leq \frac{1}{3}\)
- \(x \leq \frac{1}{4}\)
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.

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

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

Harmonic Algorithm

\[ K = 4 \]

\(< 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 \ ... > \]
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.
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 $S$. 
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.
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 $k/2 \rightarrow S > k/2 \rightarrow k < 2S$

![Diagram of Next Fit algorithm](Image)
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.
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 OPT packs items tightly (with no wasted space), $S(\sigma)$ bins are required.
Competitive Analysis of Next Fit

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

- Competitive ratio of NextFit is at least 2.
  - Consider sequence \( \sigma = <0.5, \epsilon, 0.5, \epsilon, \ldots > \).
Competitive Analysis of Next Fit

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

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

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

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
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 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
\[ \sigma = \langle \frac{1}{2} - \varepsilon, \frac{1}{2} - \varepsilon, \ldots, \frac{1}{2} - \varepsilon, \frac{1}{2} + \varepsilon, \frac{1}{2} + \varepsilon, \ldots, \frac{1}{2} + \varepsilon \rangle \]

Consider the sub-sequence formed by the first \( m \) items
An Easy Lower Bound ...

- Consider the following 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 \( m \) items

- Consider the sub-sequence formed by the first \( m \) items
  - Cost of \( \text{OPT} \) for the sub-sequence is \( m/2 \)
Consider the following input

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

\[ m \text{ items} \]

\[ m \text{ items} \]

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 \( 1/2 \leq \alpha \leq 1 \).
An Easy Lower Bound ...

- Consider the following input
  \[ \sigma = \langle \frac{1}{2} - \epsilon, \frac{1}{2} - \epsilon, \ldots, \frac{1}{2} - \epsilon, \frac{1}{2} + \epsilon, \frac{1}{2} + \epsilon, \ldots, \frac{1}{2} + \epsilon \rangle \]
  with \( m \) items.

- 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 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 \( \frac{\alpha m}{m/2} \geq 2\alpha \).

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:
\[ \sigma = \langle 1/2 - \epsilon, 1/2 - \epsilon, \ldots, 1/2 - \epsilon, 1/2 + \epsilon, 1/2 + \epsilon, \ldots, 1/2 + \epsilon \rangle \]

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

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

- 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 \( \alpha = \frac{2}{3} \), and the competitive ratio will be at least \( \frac{4}{3} \).
Consider the following input

$$\sigma = \langle 1/2 - \epsilon, 1/2 - \epsilon, \ldots, 1/2 - \epsilon, 1/2 + \epsilon, 1/2 + \epsilon, \ldots, 1/2 + \epsilon \rangle$$

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

No algorithm can be within a ratio less than 4/3 of OPT.
An Easy Lower Bound

- Input:
  \[ \sigma = \langle 1/2 - \epsilon, 1/2 - \epsilon, \ldots, 1/2 - \epsilon, 1/2 + \epsilon, 1/2 + \epsilon, \ldots, 1/2 + \epsilon \rangle \]

- No algorithm can be within a ratio less than 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
Online algorithms have two shortcomings against $\text{Opt}$

1) **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!
Review & Plan

Shortcomings of Online Algorithms

- Online algorithms have two shortcomings against $\text{Opt}$
  - **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!
  - **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

\( \sigma_1 = \langle \frac{1}{2} - \epsilon, \ldots, \frac{1}{2} - \epsilon, \frac{1}{2} + \epsilon, \ldots, \frac{1}{2} + \epsilon \rangle \rightarrow \)

\( m \) items \( \rightarrow \) competitive ratio \( > \frac{4}{3} \)

More complicated sequences result in better lower bounds.

Theorem

No online bin packing algorithm can have a competitive ratio better than 1.54037.

In the next class, we study the weighting technique to prove that Best Fit and First Fit have competitive ratio 1.7.
A better lower bound

\[ \sigma_1 = \langle \frac{1}{2} - \epsilon, \ldots \frac{1}{2} - \epsilon, \frac{1}{2} + \epsilon, \ldots, \frac{1}{2} + \epsilon \rangle \rightarrow \]

\[ \text{competitive ratio } > \frac{4}{3} \]

\[ \sigma_2 = \langle \frac{1}{6} - \epsilon, \ldots \frac{1}{6} - \epsilon, \frac{1}{3} - \epsilon, \ldots \frac{1}{3} - \epsilon, \frac{1}{2} + 2\epsilon, \ldots, \frac{1}{2} + 2\epsilon \rangle \rightarrow \]

\[ \text{competitive ratio } > \frac{3}{2} \]
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 $m$ 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 $m$ items $m$ items

competitive ratio $> 3/2$

More complicated sequences results better lower bounds
A better lower bound

\[ \sigma_1 = \langle 1/2 - \epsilon, \ldots 1/2 - \epsilon, 1/2 + \epsilon, \ldots, 1/2 + \epsilon \rangle \rightarrow \]

\[ m \text{ items} \quad m \text{ items} \]

Competitive ratio \( > 4/3 \)

\[ \sigma_2 = \langle 1/6 - \epsilon, \ldots 1/6 - \epsilon, 1/3 - \epsilon, \ldots 1/3 - \epsilon, 1/2 + 2\epsilon, \ldots, 1/2 + 2\epsilon \rangle \rightarrow \]

\[ m \text{ items} \quad m \text{ items} \quad m \text{ items} \]

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.*
A better lower bound

\[ \sigma_1 = \langle 1/2 - \epsilon, \ldots 1/2 - \epsilon, 1/2 + \epsilon, \ldots, 1/2 + \epsilon \rangle \rightarrow \]

\begin{align*}
&\text{competitive ratio } > 4/3 \\
\end{align*}

\[ \sigma_2 = \langle 1/6 - \epsilon, \ldots 1/6 - \epsilon, 1/3 - \epsilon, \ldots 1/3 - \epsilon, 1/2 + 2\epsilon, \ldots, 1/2 + 2\epsilon \rangle \rightarrow \]

\begin{align*}
&\text{competitive ratio } > 3/2 \\
\end{align*}

- More complicated sequences results better lower bounds

**Theorem**

No online bin packing algorithm can have a competitive ratio better than 1.54037.

In the next class, we study the **weighting technique** to prove that Best Fit and First Fit have competitive ratio 1.7.