Today’s objectives

- Renting servers in the cloud
- Online bin packing with advice
Renting Servers in the Cloud

Buying vs. Renting

When you buy servers, the goal is to minimize the total number of opened (purchased) servers.
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.
Renting Servers in the Cloud

First Fit Algorithm

- Apply First Fit algorithm to place items.
Renting Servers in the Cloud

First Fit Algorithm

- Apply First Fit algorithm to place items.
- **Release** the bin when all items depart.

\[
\begin{align*}
& a = (0.3, 1, 7), & b = (0.4, 2, 7), & c = (0.4, 3, 7), & d = (0.5, 4, 7), & e = (0.3, 5, 20), & f = (0.1, 6, 20), \ldots \\
& \text{Time: 0}
\end{align*}
\]
Renting Servers in the Cloud

First Fit Algorithm

- Apply First Fit algorithm to place items.
- Release the bin when all items depart.

< \(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: 1

\(a\)

\(0.3\)
Renting Servers in the Cloud

First Fit Algorithm

- Apply First Fit algorithm to place items.
- Release the bin when all items depart.

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

Time: $2$
Renting Servers in the Cloud

First Fit Algorithm

- Apply First Fit algorithm to place items.
- **Release** the bin when all items depart.

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

Time: 3
Renting Servers in the Cloud

First Fit Algorithm

- Apply First Fit algorithm to place items.
- **Release** the bin when all items depart.

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

<table>
<thead>
<tr>
<th>Time: 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>a 0.3</td>
</tr>
<tr>
<td>b 0.4</td>
</tr>
<tr>
<td>c 0.4</td>
</tr>
<tr>
<td>d 0.5</td>
</tr>
</tbody>
</table>
Renting Servers in the Cloud

First Fit Algorithm

- Apply First Fit algorithm to place items.
- **Release** the bin when all items depart.

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

Time: 5

COMP 7720 - Online Algorithms
Renting Servers in the Cloud

First Fit Algorithm

- Apply First Fit algorithm to place items.
- **Release** the bin when all items depart.

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

Time: 6
Renting Servers in the Cloud

First Fit Algorithm

- Apply First Fit algorithm to place items.
- **Release** the bin when all items depart.

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

Time: 7
Renting Servers in the Cloud

First Fit Algorithm

- Apply First Fit algorithm to place items.
- **Release** the bin when all items depart.
- The cost of First Fit is $18 + 20$ (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
Renting Servers in the Cloud

First Fit Algorithm

- Apply First Fit algorithm to place items.
- **Release** the bin when all items depart.
- The cost of First Fit is 18 + 20 (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
Renting Servers in the Cloud

First Fit Algorithm

- Apply First Fit algorithm to place items.
- **Release** the bin when all items depart.
- The cost of First Fit is $18 + 20$ (assuming no other item arrives till time 20).

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

Time: 7

![Diagram of First Fit Algorithm with items placed in a bin.]

COMP 7720 - Online Algorithms

Online Bin Packing
No Any-Fit algorithm can be better than $\mu$ competitive.

- $\mu$ is the ratio between the length of the largest and the smallest item \cite{LiTang14}.

First Fit is at most $2\mu + 13$-competitive \cite{LiTang14}.

Best Fit is not competitive.
Renting Servers in the Cloud

Next Fit Algorithm

- Apply Next Fit algorithm to place items.

The cost of Next Fit is $7 + 5 + 15 = 27$ (assuming no other item arrives till time 20).
Renting Servers in the Cloud

Next Fit Algorithm

- Apply Next Fit algorithm to place items.
- **Release** the bin when all items depart.

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

Time: 0
Renting Servers in the Cloud

Next Fit Algorithm

- Apply Next Fit algorithm to place items.
- **Release** the bin when all items depart.

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

Time: 1
Renting Servers in the Cloud

Next Fit Algorithm

- Apply Next Fit algorithm to place items.
- Release the bin when all items depart.

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

Time: 2
Renting Servers in the Cloud

Next Fit Algorithm

- Apply Next Fit algorithm to place items.
- **Release** the bin when all items depart.

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

Time: 3
Renting Servers in the Cloud

Next Fit Algorithm

- Apply Next Fit algorithm to place items.
- **Release** the bin when all items depart.

\[ \text{Time: 4} \]

\[ \langle a = (0.3, 1, 7), b = (0.4, 2, 7), c = (0.4, 3, 7), d = (0.5, 4, 7), e = (0.3, 5, 20), f = (0.1, 6, 20), \ldots \rangle \]
Renting Servers in the Cloud

Next Fit Algorithm

- Apply Next Fit algorithm to place items.
- Release the bin when all items depart.

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

Time: 5
Renting Servers in the Cloud

Next Fit Algorithm

- Apply Next Fit algorithm to place items.
- **Release** the bin when all items depart.

< \( 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: 6
Renting Servers in the Cloud

Next Fit Algorithm

- Apply Next Fit algorithm to place items.
- **Release** the bin when all items depart.

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

Time: 7
Renting Servers in the Cloud

Next Fit Algorithm

- **Apply** Next Fit algorithm to place items.
- **Release** the bin when all items depart.
- The cost of Next Fit is \(7 + 5 + 15 = 27\) (assuming no other item arrives till time 20).

\[
\begin{align*}
\langle a &= (0.3, 1, 7), b = (0.4, 2, 7), c = (0.4, 3, 7), d = (0.5, 4, 7), e = (0.3, 5, 20), f = (0.1, 6, 20), \ldots \rangle \\
\text{Time: } &7
\end{align*}
\]
Renting Servers in the Cloud

Next Fit Algorithm

- Apply Next Fit algorithm to place items.
- **Release** the bin when all items depart.
- The cost of Next Fit is $7 + 5 + 15 = 27$ (assuming no other item arrives till time 20).

\[
\begin{align*}
\langle a = (0.3, 1, 7), b = (0.4, 2, 7), c = (0.4, 3, 7), d = (0.5, 4, 7), e = (0.3, 5, 20), f = (0.1, 6, 20), \ldots \rangle \\
\text{Time: } 7
\end{align*}
\]
Renting Servers in the Cloud

Next Fit Algorithm

- Apply Next Fit algorithm to place items.
- **Release** the bin when all items depart.
- The cost of Next Fit is $7 + 5 + 15 = 27$ (assuming no other item arrives till time 20).

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

Time: 7

\[ e \quad 0.3 \]
\[ f \quad 0.1 \]
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.
Renting Servers in the Cloud

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.
Renting Servers in the Cloud

Move To Front Algorithm

- We introduce a new algorithm Move To Front.
Renting Servers in the Cloud

Move To Front Algorithm

- 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), \ldots > \\

Time: 0
Renting Servers in the Cloud

**Move To Front Algorithm**

- 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.

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

Time: 1
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.

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

Time: 2
Renting Servers in the Cloud

Move To Front Algorithm

- 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: 3
Renting Servers in the Cloud

Move To Front Algorithm

- 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.

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

Time: 3
Renting Servers in the Cloud

Move To Front Algorithm

- 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.

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

Time: 4
Renting Servers in the Cloud

Move To Front Algorithm

- 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: 5
Renting Servers in the Cloud

Move To Front Algorithm

- 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: 5
Renting Servers in the Cloud

---

**Move To Front Algorithm**

- 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.

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

Time: 6
Renting Servers in the Cloud

Move To Front Algorithm

- 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
Renting Servers in the Cloud

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.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>$T = 1,000$</th>
<th>$T = 10,000$</th>
<th>$T = 100,000$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Next Fit</td>
<td>$1.4011$</td>
<td>$1.7186$</td>
<td>$1.9539$</td>
</tr>
<tr>
<td>Modified Next Fit</td>
<td>$1.4392$</td>
<td>$1.8069$</td>
<td>$1.9762$</td>
</tr>
<tr>
<td>First Fit</td>
<td>$1.5647$</td>
<td>$1.7485$</td>
<td>$1.9544$</td>
</tr>
<tr>
<td>Modified First Fit</td>
<td>$1.6666$</td>
<td>$1.8362$</td>
<td>$1.9726$</td>
</tr>
<tr>
<td>Harmonic Best Fit</td>
<td>$1.7598$</td>
<td>$1.9555$</td>
<td>$1.9946$</td>
</tr>
<tr>
<td>Move To Front</td>
<td>$1.4113$</td>
<td>$1.7134$</td>
<td>$1.9536$</td>
</tr>
</tbody>
</table>

For $\mu = 2$:

- Next Fit: $1.4618$, $1.6564$, $1.8721$
- Modified Next Fit: $1.4820$, $1.8061$, $1.9738$
- First Fit: $1.4561$, $1.6635$, $1.9292$
- Modified First Fit: $1.5335$, $1.5288$, $1.8842$
- Harmonic: $1.5848$, $1.5148$, $1.8195$
- Best Fit: $1.3094$, $1.5148$, $1.8195$
- Move To Front: $1.3921$, $1.6323$, $1.8689$
Advice Complexity of Bin Packing

Bin Packing with Advice
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).
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?
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.
- For each item, we require \( O(\log n) \) 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 is it optimal? The resulting packing is similar to \( \text{Opt} \).
Advice Complexity of Bin Packing

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.
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?

\[ O(n \log k) \]
Advice Complexity of Bin Packing

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 $O(\log n)$ bits to encode the target bin. In total, $O(n \log k)$ bits suffice.
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 $O(\log n)$ bits to encode the target bin. In total, $O(n \log k)$ bits suffice.

- How the algorithm work with the given 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 $O(\log n)$ bits to encode the target bin. In total, $O(n \log k)$ bits suffice.

How the algorithm work with the given advice?
- It packs each item in the same bin as $\text{OPT}$ does.
Advice Complexity of Bin Packing

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 $O(\log n)$ bits to encode the target bin. In total, $O(n \log k)$ bits suffice.

- How the algorithm work 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 $\text{OPT}$.
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.

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

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.

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

- Each of the first \( n-k+1 \) items can be packed in any of the \( k \) bins.
- The summation of all of them is less than 1/2.
- There will be \( k^{n-k+1} \) 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 \( k^{n-k+1} \) different packings!
Advice Complexity of Bin Packing

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 \( k^{n-k+1} \) different packings!
- For each packing the last items (\( u_i \)'s) fill the empty space for each bin
Advice Complexity of Bin Packing

Optimal Solution with Advice

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

- There will be \( k^{n-k+1} \) different packings!
- For each packing the last items \((u_i)'s\) fill the empty space for each bin

\[ \sigma_1 = \left\langle 1/4, 1/8, 1/16, 1 - 1/4, 1 - 1/8, 1 - 1/16 \right\rangle \]
\[ \sigma_2 = \left\langle 1/4, 1/8, 1/16, 1 - 1/4 - 1/16, 1 - 1/8, 1 - 1/16 \right\rangle \]
\[ \sigma_3 = \left\langle 1/4, 1/8, 1/16, 1 - 1/4 - 1/8, 1 - 1/16, 1 \right\rangle \]
\[ \sigma_4 = \left\langle 1/4, 1/8, 1/16, 1 - 1/4 - 1/8 - 1/16, 1, 1 \right\rangle \]
Advice Complexity of Bin Packing

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 \( k^{n-k+1} \) different sequences!
  - All start with the same prefix of length \( n - k + 1 \)
- For each sequence, an optimal algorithm should pack the first \( n - k + 1 \) items differently from others.
  - Each sequence requires an advice tailored for itself
- \( k^{n-k+1} \) different advice strings are required
  - \( \Omega(\log k^{n-k+1}) \approx \Omega(n \log k) \) advice bits are required.
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.
Advice Complexity of Bin Packing

**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!
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.*
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!
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
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!
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 $\epsilon$. 

Theorem

$To achieve an optimal solution least \Omega(m \log n)$ bits of advice are required.$
Advice Complexity of Bin 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 in $m - 2$ different bin types.
    - $X$ will be at least $\left(1 + \frac{n}{(m-1)(m-2)}\right)^{m-3}$ (we skip the proof here)
Advice Complexity of Bin Packing

The Idea Behind the Lower Bound

For each partial packing, complete the sequence with items which fill the empty spaces

Example:

\( n = 30, m = 6 \)
(bin capacities scaled up by 12)

sequence: \(<1^{(15)} \ldots\)
Advice Complexity of Bin Packing

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 $\lg X$ is required.
- At least $(1 + \frac{n}{(m-1)(m-2)})^{m-3} = \Omega(m \log n)$ bits are required.

Example:

\[ n = 30, \ m = 6 \]

(bin capacities scaled up by 12)

sequence: \( <1^{(15)}\ 11\ 11\ 11\ 10\ 10\ 9\ 8\ 12^{(7)} > \)

sequence: \( <1^{(15)}\ 11\ 11\ 11\ 11\ 11\ 11\ 11\ 11\ 8\ 8\ 12^{(6)} > \ldots \)
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.
Advice Complexity of Bin Packing

Breaking the Lower Bound

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
- All online algorithms have a competitive ratio of at least 1.54.
Advice Complexity of Bin Packing

Breaking the Lower Bound

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
  - All online algorithms have a competitive ratio of at least 1.54.

- Receive the number of items in range $(1/2, 2/3]$.
  - Reserve a space of size $2/3$ for each, apply FF for other items.
Advice Complexity of Bin Packing

Breaking the Lower Bound

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
  - All online algorithms have a competitive ratio of at least 1.54.
- Receive the number of items in range $(1/2, 2/3]$.
  - Reserve a space of size $2/3$ for each, apply FF for other items.

< 9 3 6 5 1 1 3 2 4 2 4 5 6 8 6 4 5 >
Advice Complexity of Bin Packing

Breaking the Lower Bound

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
  - All online algorithms have a competitive ratio of at least 1.54.
- Receive the number of items in range $(1/2, 2/3]$.
  - Reserve a space of size $2/3$ for each, apply FF for other items.

\[
<9, 3, 6, 5, 1, 1, 3, 2, 4, 2, 4, 5, 6, 8, 6, 4, 5>
\]
Advice Complexity of Bin Packing

## Breaking the Lower Bound

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
  - All online algorithms have a competitive ratio of at least 1.54.

- Receive the number of items in range $(1/2, 2/3]$.
  - Reserve a space of size $2/3$ for each, apply FF for other items.

```
< 9 3 6 5 1 1 3 2 4 2 4 5 6 8 6 4 5 >
```

![Diagram](image.png)
\( O(\log n) \) bits of advice is sufficient to achieve competitive ratio 1.5.

- All online algorithms have a competitive ratio of at least 1.54.

Receive the number of items in range \((1/2, 2/3]\).

- Reserve a space of size \(2/3\) for each, apply FF for other items.

\[
< 9 \ 3 \ 6 \ 5 \ 1 \ 1 \ 3 \ 2 \ 4 \ 2 \ 4 \ 5 \ 6 \ 8 \ 6 \ 4 \ 5 >
\]
Advice Complexity of Bin Packing

Breaking the Lower Bound

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
  - All online algorithms have a competitive ratio of at least 1.54.

- Receive the number of items in range $(1/2, 2/3]$.
  - Reserve a space of size $2/3$ for each, apply FF for other items.

```
< 9 3 6 5 1 1 3 2 4 2 4 5 6 8 6 4 5 >
```

![Diagram](image)
**Advice Complexity of Bin Packing**

**Breaking the Lower Bound**

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
  - All online algorithms have a competitive ratio of at least 1.54.
- Receive the number of items in range $(1/2, 2/3]$.
  - Reserve a space of size $2/3$ for each, apply FF for other items.

```
< 9 3 6 5 1 1 3 2 4 2 4 5 6 8 6 4 5 >
```

![Diagram showing bin packing](image)
Advice Complexity of Bin Packing

Breaking the Lower Bound

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
  - All online algorithms have a competitive ratio of at least 1.54.
- Receive the number of items in range $(1/2, 2/3]$.
  - Reserve a space of size $2/3$ for each, apply FF for other items.
Advice Complexity of Bin Packing

Breaking the Lower Bound

- \( O(\log n) \) bits of advice is sufficient to achieve competitive ratio 1.5.
  - All online algorithms have a competitive ratio of at least 1.54.
- Receive the number of items in range \((1/2, 2/3]\).
  - Reserve a space of size 2/3 for each, apply FF for other items.

\[
< 9 3 6 5 1 1 3 2 4 2 4 5 6 8 6 4 5 >
\]
Advice Complexity of Bin Packing

Breaking the Lower Bound

- \(O(\log n)\) bits of advice is sufficient to achieve competitive ratio 1.5.
  - All online algorithms have a competitive ratio of at least 1.54.
- Receive the number of items in range \((1/2, 2/3]\).
  - Reserve a space of size 2/3 for each, apply FF for other items.
Advice Complexity of Bin Packing

Breaking the Lower Bound

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
  - All online algorithms have a competitive ratio of at least 1.54.
- Receive the number of items in range $(1/2, 2/3]$.
  - Reserve a space of size $2/3$ for each, apply FF for other items.

```plaintext
< 9 3 6 5 1 1 3 2 4 2 4 5 6 8 6 4 5 >
```

```
1
3
6

2
1
3

1
9

2
5
```
Advice Complexity of Bin Packing

Breaking the Lower Bound

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
  - All online algorithms have a competitive ratio of at least 1.54.
- Receive the number of items in range $(1/2, 2/3]$.
  - Reserve a space of size $2/3$ for each, apply FF for other items.
Advice Complexity of Bin Packing

Breaking the Lower Bound

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
  - All online algorithms have a competitive ratio of at least 1.54.
- Receive the number of items in range $(1/2, 2/3]$.
  - Reserve a space of size $2/3$ for each, apply FF for other items.

< 9 3 6 5 1 1 3 2 4 2 4 5 6 8 6 4 5 >
Advice Complexity of Bin Packing

**Breaking the Lower Bound**

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
  - All online algorithms have a competitive ratio of at least 1.54.
- Receive the number of items in range $(1/2, 2/3]$.
  - Reserve a space of size $2/3$ for each, apply FF for other items.

< 9 3 6 5 1 1 3 2 4 2 4 5 6 8 6 4 5 >
Advice Complexity of Bin Packing

Breaking the Lower Bound

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
  - All online algorithms have a competitive ratio of at least 1.54.
- Receive the number of items in range $(1/2, 2/3]$.
  - Reserve a space of size $2/3$ for each, apply FF for other items.

< 9 3 6 5 1 1 3 2 4 2 4 5 6 8 6 4 5 >

![Diagram showing bin packing with reserved space and items placed accordingly.]
Advice Complexity of Bin Packing

Breaking the Lower Bound

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
  - All online algorithms have a competitive ratio of at least 1.54.
- Receive the number of items in range $(1/2, 2/3]$.
  - Reserve a space of size $2/3$ for each, apply FF for other items.

< 9 3 6 5 1 1 3 2 4 2 4 5 6 8 6 4 5 >
Advice Complexity of Bin Packing

Breaking the Lower Bound

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
- All online algorithms have a competitive ratio of at least 1.54.
- Receive the number of items in range $(1/2, 2/3]$.
  - Reserve a space of size $2/3$ for each, apply FF for other items.

\[ \langle 9 \ 3 \ 6 \ 5 \ 1 \ 1 \ 3 \ 2 \ 4 \ 2 \ 4 \ 5 \ 6 \ 8 \ 6 \ 4 \ 5 \rangle \]
Advice Complexity of Bin Packing

**Breaking the Lower Bound**

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
  - All online algorithms have a competitive ratio of at least 1.54.
- Receive the number of items in range $(1/2, 2/3]$.
  - Reserve a space of size $2/3$ for each, apply FF for other items.

```
< 9 3 6 5 1 1 3 2 4 2 4 5 6 8 6 4 5 >
```
Advice Complexity of Bin Packing

Breaking the Lower Bound

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
- All online algorithms have a competitive ratio of at least 1.54.
- Receive the number of items in range $(1/2, 2/3]$.
- Reserve a space of size $2/3$ for each, apply FF for other items.

< 9 3 6 5 1 1 3 2 4 2 4 5 6 8 6 4 5 >
**Advice Complexity of Bin Packing**

**Breaking the Lower Bound**

- $O(\log n)$ bits of advice is sufficient to achieve competitive ratio 1.5.
- All online algorithms have a competitive ratio of at least 1.54.

- Receive the number of items in range ($1/2, 2/3$).
- Reserve a space of size $2/3$ for each, apply FF for other items.

\[
\text{< 9 3 6 5 1 1 3 2 4 2 4 5 6 8 6 4 5 >}
\]

```
< 9 3 6 5 1 1 3 2 4 2 4 5 6 8 6 4 5 >
```

We see the analysis of the algorithm in the next class!