Lecture 19
Approximation Algorithms (part four)
Asymptotic Polynomial Time Approximation Schemes

Benjamin Sach
Approximation Algorithms Recap

A polynomial time algorithm $A$ is an $\alpha$-approximation for problem $P$ if, it always outputs a solution $s$ with

$$\frac{\text{Opt}}{\alpha} \leq s \leq \text{Opt} \quad \text{(for a maximisation problem)}$$

$$\text{Opt} \leq s \leq \alpha \cdot \text{Opt} \quad \text{(for a minimisation problem)}$$

A poly-time approximation scheme (PTAS) is a family of algorithms:

For any constant $\epsilon > 0$, $A_\epsilon$ is a $(1 + \epsilon)$-approximation for $P$

- A PTAS can have $A_\epsilon$ which takes $O(n^{1/\epsilon})$ time
  - which is polynomial in $n$ (for any constant $\epsilon$)

- It is a (fully) FPTAS if $A_\epsilon$ takes time polynomial in both $n$ and $1/\epsilon$ i.e. $O((n/\epsilon)^c)$
Approximation Algorithms Recap

A polynomial time algorithm $A$ is an $\alpha$-approximation for problem $P$ if, it always outputs a solution $s$ with

$$\frac{\text{Opt}}{\alpha} \leq s \leq \text{Opt} \quad \text{(for a maximisation problem)}$$

$$\text{Opt} \leq s \leq \alpha \cdot \text{Opt} \quad \text{(for a minimisation problem)}$$

- Various $c$-approximations with constant $c$

- We saw an FPTAS for $\text{SUBSETSUM}$

A poly-time approximation scheme (PTAS) is a family of algorithms:

For any constant $\epsilon > 0$, $A_\epsilon$ is a $(1 + \epsilon)$-approximation for $P$

- A PTAS can have $A_\epsilon$ which takes $O(n^{1/\epsilon})$ time
  - which is polynomial in $n$ (for any constant $\epsilon$)

- It is a (fully) FPTAS if $A_\epsilon$ takes time polynomial in both $n$ and $1/\epsilon$ i.e. $O((n/\epsilon)^c)$
The **SUBSET Sum** problem

\[ t = 12 \]
The **SUBSET**Sum problem

- Let $S$ be a (multi) set of integers and $t$ be a positive integer
  
  here $S = \{4, 2, 4, 7, 2, 3\}$ and $t = 12$
Let $S$ be a (multi) set of integers and $t$ be a positive integer. Here $S = \{4, 2, 4, 7, 2, 3\}$ and $t = 12$.

**Decision Problem** Is there a subset, $S' \subseteq S$ with $\text{SIZE}(S') = t$?
**The SubsetSum problem**

- Let $S$ be a (multi) set of integers and $t$ be a positive integer

> here $S = \{4, 2, 4, 7, 2, 3\}$ and $t = 12$

**Decision Problem** Is there a subset, $S' \subseteq S$ with $\text{SIZE}(S') = t$?

where $\text{SIZE}(S') = \sum_{a \in S'} a$
The **SubsetSum** problem

- Let $S$ be a (multi) set of integers and $t$ be a positive integer
  
  Here $S = \{4, 2, 4, 7, 2, 3\}$ and $t = 12$

**Decision Problem** Is there a subset, $S' \subseteq S$ with $\text{SIZE}(S') = t$?

where $\text{SIZE}(S') = \sum_{a \in S'} a$
The **SubsetSum** problem

- Let $S$ be a (multi) set of integers and $t$ be a positive integer

  $$S = \{4, 2, 4, 7, 2, 3\} \text{ and } t = 12$$

**Decision Problem** Is there a subset, $S' \subseteq S$ with $\text{SIZE}(S') = t$?

where $\text{SIZE}(S') = \sum_{a \in S'} a$
The **SubsetSum** problem

- Let $S$ be a (multi) set of integers and $t$ be a positive integer

  here $S = \{4, 2, 4, 7, 2, 3\}$ and $t = 12$

**Decision Problem** Is there a subset, $S' \subseteq S$ with $\text{SIZE}(S') = t$?

  where $\text{SIZE}(S') = \sum_{a \in S'} a$

The decision version is **NP**-complete

(last lecture we discussed the **NP**-hard optimisation version)
The **PARTITION** problem

- The **PARTITION** problem is a special case of **SUBSET SUM**

![Image of the PARTITION problem with numbers 4, 2, 4, 7, 2, 3]
The **PARTITION** problem

- The **PARTITION** problem is a special case of **SUBSET SUM**
• The **PARTITION** problem is a special case of **SUBSETSUM**

• Let $S$ be a (multi) set of integers
The **PARTITION** problem

- The **PARTITION** problem is a special case of **SUBSET SUM**
- Let $S$ be a (multi) set of integers
  - but $t$ is always half the sum of item sizes
The PARTITION problem

- The PARTITION problem is a special case of SUBSETSUM
- Let $S$ be a (multi) set of integers
  but $t$ is always half the sum of item sizes

$4 \ 2 \ 4 \ 7 \ 2 \ 3$

$t = 11$
The **PARTITION** problem

- The **PARTITION** problem is a special case of **SUBSETSUM**

- Let $S$ be a (multi) set of integers
  
  but $t$ is always half the sum of item sizes

  $$t = \frac{\text{SIZE}(S)}{2} = \sum_{a \in S'} \frac{a}{2}$$
The **PARTITION** problem

- The **PARTITION** problem is a special case of **SUBSET SUM**

- Let $S$ be a (multi) set of integers
  
  but $t$ is always half the sum of item sizes
  
  \[ t = \frac{\text{SIZE}(S)}{2} = \sum_{a \in S'} \frac{a}{2} \]

**Decision Problem** Is there a subset, $S' \subseteq S$ with $\text{SIZE}(S') = \frac{\text{SIZE}(S)}{2}$?
The **PARTITION** problem

- The **PARTITION** problem is a special case of **SUBSETSUM**

- Let $S$ be a (multi) set of integers
  
  but $t$ is always half the sum of item sizes
  
  \[
  t = \frac{\text{SIZE}(S)}{2} = \sum_{a \in S'} \frac{a}{2}
  \]

**Decision Problem** Is there a subset, $S' \subseteq S$ with $\text{SIZE}(S') = \frac{\text{SIZE}(S)}{2}$?
The \textsc{Partition} problem

- The \textsc{Partition} problem is a special case of \textsc{SubsetSum}
- Let $S$ be a (multi) set of integers
  but $t$ is always half the sum of item sizes
  \[ t = \text{SIZE}(S)/2 = \sum_{a \in S'} \frac{a}{2} \]

\textbf{Decision Problem} Is there a subset, $S' \subseteq S$ with $\text{SIZE}(S') = \text{SIZE}(S)/2$?
The **PARTITION** problem

- The **PARTITION** problem is a special case of **SubsetSum**

- Let $S$ be a (multi) set of integers
  
  but $t$ is always half the sum of item sizes
  
  \[ t = \text{SIZE}(S)/2 = \sum_{a \in S'} \frac{a}{2} \]

**Decision Problem** Is there a subset, $S' \subseteq S$ with $\text{SIZE}(S') = \text{SIZE}(S)/2$?
The **PARTITION** problem

- The **PARTITION** problem is a special case of **SUBSETSUM**

- Let $S$ be a (multi) set of integers
  
  but $t$ is always half the sum of item sizes

  $$ t = \frac{\text{SIZE}(S)}{2} = \sum_{a \in S'} \frac{a}{2} $$

**Decision Problem** Is there a subset, $S' \subseteq S$ with $\text{SIZE}(S') = \frac{\text{SIZE}(S)}{2}$?
The **PARTITION** problem

- The **PARTITION** problem is a special case of **SUBSETSUM**

- Let $S$ be a (multi) set of integers
  
  but $t$ is always half the sum of item sizes
  
  $$t = \text{SIZE}(S)/2 = \sum_{a \in S'} \frac{a}{2}$$

**Decision Problem** Is there a subset, $S' \subseteq S$ with $\text{SIZE}(S') = \text{SIZE}(S)/2$?

**Alternatively...** Can we pack $S$ into two bins of size $\text{SIZE}(S)/2$?
The **PARTITION** problem

- The **PARTITION** problem is a special case of **SUBSETSUM**

- Let $S$ be a (multi) set of integers

  but $t$ is always half the sum of item sizes

  $$t = \text{SIZE}(S)/2 = \sum_{a \in S'} \frac{a}{2}$$

**Decision Problem** Is there a subset, $S' \subseteq S$ with $\text{SIZE}(S') = \text{SIZE}(S)/2$?

**Alternatively...** Can we pack $S$ into two bins of size $\text{SIZE}(S)/2$?
The **PARTITION** problem

- The **PARTITION** problem is a special case of **SUBSETSUM**

- Let $S$ be a (multi) set of integers
  but $t$ is always half the sum of item sizes
  \[ t = \frac{\text{SIZE}(S)}{2} = \sum_{a \in S'} \frac{a}{2} \]

**Decision Problem** Is there a subset, $S' \subseteq S$ with $\text{SIZE}(S') = \frac{\text{SIZE}(S)}{2}$?

**Alternatively...** Can we pack $S$ into two bins of size $\frac{\text{SIZE}(S)}{2}$?
The **PARTITION** problem

- The **PARTITION** problem is a special case of **SUBSETSUM**

- Let \( S \) be a (multi) set of integers
  but \( t \) is always half the sum of item sizes

\[
t = \frac{\text{SIZE}(S)}{2} = \sum_{a \in S'} \frac{a}{2}
\]

**Decision Problem** Is there a subset, \( S' \subseteq S \) with \( \text{SIZE}(S') = \frac{\text{SIZE}(S)}{2} \)?

**Alternatively...** Can we pack \( S \) into two bins of size \( \frac{\text{SIZE}(S)}{2} \)?

The **PARTITION** problem is also **NP**-complete.
PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING
PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING

4 2 4 7 2 3
**Key Idea** Solve the **PARTITION** problem by approximating **BINPACKING**

- Convert the **PARTITION** instance into a **BINPACKING** problem by dividing all item and bin sizes by $t = \text{SIZE}(S)/2$
**PARTITION** and **BinPacking**

**Key Idea** Solve the **PARTITION** problem by approximating **BinPacking**

- Convert the **PARTITION** instance into a **BinPacking** problem by dividing all item and bin sizes by \( t = \frac{\text{SIZE}(S)}{2} \)
**PARTITION** and **BinPacking**

**Key Idea** Solve the **PARTITION** problem by approximating **BinPacking**

- Convert the **PARTITION** instance into a **BinPacking** problem by dividing all item and bin sizes by $t = \text{SIZE}(S)/2$
- Now all items are have size at most 1 and the bins have size 1
**PARTITION and BINPACKING**

**Key Idea** Solve the PARTITION problem by approximating BINPACKING

- Convert the PARTITION instance into a BINPACKING problem by dividing all item and bin sizes by $t = \text{SIZE}(S)/2$
- Now all items are have size at most 1 and the bins have size 1
- The optimal number of bins $\text{Opt}_b$ is 2 iff the answer to the PARTITION instance is ‘yes’
**PARTITION** and **BinPacking**

**Key Idea** Solve the **PARTITION** problem by approximating **BinPacking**

- Convert the **PARTITION** instance into a **BinPacking** problem by dividing all item and bin sizes by \( t = \text{SIZE}(S)/2 \)
- Now all items are have size at most 1 and the bins have size 1
- The optimal number of bins \( \text{Opt}_b \) is 2 iff the answer to the **PARTITION** instance is ‘yes’
- What does this tell us about approximating **BinPacking**?
**PARTITION and BINPACKING**

**Key Idea** Solve the **PARTITION** problem by approximating **BINPACKING**

- Assume $A$ is an $\alpha$-approximation for **BINPACKING** with $\alpha < 3/2$
**PARTITION and BINPACKING**

**Key Idea** Solve the PARTITION problem by approximating BINPACKING

- Assume $A$ is an $\alpha$-approximation for BINPACKING with $\alpha < \frac{3}{2}$
  
  $A$ outputs a solution with $\text{Opt}_b \leq s \leq \alpha \cdot \text{Opt}_b$
PARTITION and BINPACKING

Key Idea Solve the PARTITION problem by approximating BINPACKING

- Assume $A$ is an $\alpha$-approximation for BINPACKING with $\alpha < 3/2$
  - $A$ outputs a solution with $\text{Opt}_b \leq s \leq \alpha \cdot \text{Opt}_b$
- If $\text{Opt}_b > 2$ then $s > 2$
**PARTITION** and **BinPacking**

**Key Idea** Solve the **PARTITION** problem by approximating **BinPacking**

- Assume $A$ is an $\alpha$-approximation for **BinPacking** with $\alpha < \frac{3}{2}$
  
  $A$ outputs a solution with $\text{Opt}_b \leq s \leq \alpha \cdot \text{Opt}_b$

- If $\text{Opt}_b > 2$ then $s > 2$

- If $\text{Opt}_b = 2$ then $2 \leq s \leq \alpha \cdot \text{Opt}_b < \left(\frac{3}{2}\right) \cdot \text{Opt}_b = 3$
PARTITION and BINPACKING

**Key Idea** Solve the PARTITION problem by approximating BINPACKING

- Assume $A$ is an $\alpha$-approximation for BINPACKING with $\alpha < 3/2$
  
  $A$ outputs a solution with $\text{Opt}_b \leq s \leq \alpha \cdot \text{Opt}_b$

- If $\text{Opt}_b > 2$ then $s > 2$

- If $\text{Opt}_b = 2$ then $2 \leq s \leq \alpha \cdot \text{Opt}_b < (3/2) \cdot \text{Opt}_b = 3$
**PARTITION and BINPACKING**

**Key Idea** Solve the PARTITION problem by approximating BINPACKING

- Assume $A$ is an $\alpha$-approximation for BINPACKING with $\alpha < 3/2$

  $A$ outputs a solution with $\text{Opt}_b \leq s \leq \alpha \cdot \text{Opt}_b$

- If $\text{Opt}_b > 2$ then $s > 2$

- If $\text{Opt}_b = 2$ then $2 \leq s \leq \alpha \cdot \text{Opt}_b < (3/2) \cdot \text{Opt}_b = 3$ so $s = 2$
**Key Idea** Solve the **PARTITION** problem by approximating **BINPACKING**

- Assume $A$ is an $\alpha$-approximation for **BINPACKING** with $\alpha < 3/2$

  $A$ outputs a solution with $\text{Opt}_b \leq s \leq \alpha \cdot \text{Opt}_b$

- If $\text{Opt}_b > 2$ then $s > 2$

- If $\text{Opt}_b = 2$ then $2 \leq s \leq \alpha \cdot \text{Opt}_b < (3/2) \cdot \text{Opt}_b = 3$ so $s = 2$

- So $A$ can solve the **NP**-complete **PARTITION** problem in polynomial time!
**PARTITION and BINPACKING**

**Key Idea** Solve the **PARTITION** problem by approximating **BINPACKING**

- Assume $A$ is an $\alpha$-approximation for **BINPACKING** with $\alpha < 3/2$
  
  $A$ outputs a solution with $\text{Opt}_b \leq s \leq \alpha \cdot \text{Opt}_b$

- If $\text{Opt}_b > 2$ then $s > 2$

- If $\text{Opt}_b = 2$ then $2 \leq s \leq \alpha \cdot \text{Opt}_b < (3/2) \cdot \text{Opt}_b = 3$ so $s = 2$

- So $A$ can solve the **NP**-complete **PARTITION** problem in polynomial time!
  which implies that $P = NP$
**Key Idea** Solve the **PARTITION** problem by approximating **BINPACKING**

**Lemma** There is no $\alpha$-approximation for **BINPACKING** with $\alpha < \frac{3}{2}$ unless $P = NP$
PARTITION and BINPACKING

Key Idea: Solve the PARTITION problem by approximating BINPACKING.

Lemma: There is no $\alpha$-approximation for BINPACKING with $\alpha < \frac{3}{2}$ unless $P = NP$.

- We saw that First Fit Decreasing (FFD) is a $\frac{3}{2}$-approximation.
**PARTITION and BINPACKING**

**Key Idea** Solve the **PARTITION** problem by approximating **BINPACKING**

![Diagram showing fractions and bins]

**Lemma** There is no $\alpha$-approximation for **BINPACKING** with $\alpha < \frac{3}{2}$ unless $P = NP$

- We saw that First Fit Decreasing (FFD) is a $\frac{3}{2}$-approximation
  
  *so this is the best we can do?*
**PARTITION and BINPACKING**

**Key Idea** Solve the PARTITION problem by approximating BINPACKING

**Lemma** There is no $\alpha$-approximation for BINPACKING with $\alpha < 3/2$ unless $P = NP$

- We saw that First Fit Decreasing (FFD) is a $3/2$-approximation
  
  *so this is the best we can do?*

- In fact, FFD gives a solution with

$$\text{Opt}_b \leq s \leq \frac{11}{9} \cdot \text{Opt}_b + 1$$
An asymptotic polynomial time approximation scheme (APTAS) for problem $P$ is a family of algorithms:

There is a constant $c$ such that

For any constant $\epsilon > 0$, $A_\epsilon$ runs in poly-time and outputs a solution $s$ with $\text{Opt} \leq s \leq (1 + \epsilon) \cdot \text{Opt} + c$
An asymptotic polynomial time approximation scheme (APTAS) for problem $P$ is a family of algorithms:

There is a constant $c$ such that

For any constant $\epsilon > 0$, $A_\epsilon$ runs in poly-time and outputs a solution $s$ with $\text{Opt} \leq s \leq (1 + \epsilon) \cdot \text{Opt} + c$

- This is the minimisation version of the definition
An asymptotic polynomial time approximation scheme (APTAS) for problem $P$ is a family of algorithms:

There is a constant $c$ such that

For any constant $\epsilon > 0$, $A_\epsilon$ runs in poly-time and outputs a solution $s$ with $\text{Opt} \leq s \leq (1 + \epsilon) \cdot \text{Opt} + c$

- This is the minimisation version of the definition
- An APTAS does not have to have running time polynomial in $1/\epsilon$
An asymptotic polynomial time approximation scheme (APTAS) for problem \( P \) is a family of algorithms:

There is a constant \( c \) such that

For any constant \( \epsilon > 0 \), \( A_\epsilon \) runs in poly-time

and outputs a solution \( s \) with \( \text{Opt} \leq s \leq (1 + \epsilon) \cdot \text{Opt} + c \)

- This is the minimisation version of the definition
- An APTAS does not have to have running time polynomial in \( 1/\epsilon \)
- An asymptotic fully PTAS (AFPTAS) is also polynomial in \( 1/\epsilon \)
An asymptotic polynomial time approximation scheme (APTAS) for problem \( P \) is a family of algorithms: There is a constant \( c \) such that for any constant \( \epsilon > 0 \), \( A_\epsilon \) runs in poly-time and outputs a solution \( s \) with \( \text{Opt} \leq s \leq (1 + \epsilon) \cdot \text{Opt} + c \).

- This is the minimisation version of the definition
- An APTAS does not have to have running time polynomial in \( 1/\epsilon \)
- An asymptotic fully PTAS (AFPTAS) is also polynomial in \( 1/\epsilon \)
- We will see an APTAS for BinPacking
A special case of BinPacking

We will now see a special case of BinPacking which we can solve in polynomial time.
A special case of **BinPacking**

We will now see a special case of **BinPacking** which we can solve in polynomial time.

- We have $n$ items but
A special case of BinPacking

We will now see a special case of BinPacking which we can solve in polynomial time.

- We have $n$ items but
  - There are $c_s$ (a constant) number of different item sizes
A special case of **BinPacking**

We will now see a special case of **BinPacking** which we can solve in polynomial time.

- We have $n$ items but
  - There are $c_s$ (a constant) number of different item sizes
  - At most $c_b$ (another constant) items fit in each bin
A special case of BinPacking

We will now see a special case of BinPacking which we can solve in polynomial time.

We have $n$ items but

- There are $c_s$ (a constant) number of different item sizes
- At most $c_b$ (another constant) items fit in each bin

Here $c_s = 3$ and $c_b = 2$.
A special case of BinPacking

We will now see a special case of BinPacking which we can solve in polynomial time

Here \( c_s = 3 \) and \( c_b = 2 \)

- We have \( n \) items but
  - There are \( c_s \) (a constant) number of different item sizes
  - At most \( c_b \) (another constant) items fit in each bin

How many different ways are there to fill a bin?
A special case of BinPacking

- We can describe any packing of items into a single bin by its *type*.

<table>
<thead>
<tr>
<th>Type:</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>3/8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3/8</td>
<td></td>
</tr>
<tr>
<td>4/8</td>
<td></td>
<td></td>
<td>7/8</td>
<td></td>
<td>3/8</td>
<td></td>
<td>4/8</td>
</tr>
<tr>
<td>4/8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3/8</td>
<td></td>
<td>4/8</td>
</tr>
</tbody>
</table>
A special case of BinPacking

- We can describe any packing of items into a single bin by its *type*.
- The *type* of a packed bin determines how many of each item is packed.
We can describe any packing of items into a single bin by its type. The type of a packed bin determines how many of each item is packed.

We ignore rearrangement of items.
A special case of **BINPACKING**

- We can describe any packing of items into a single bin by its **type**
- The **type** of a packed bin determines how many of each item is packed

We ignore rearrangement of items
A special case of BinPacking

- We can describe any packing of items into a single bin by its type.
- The type of a packed bin determines how many of each item is packed. We ignore rearrangement of items.
We can describe any packing of items into a single bin by its type.

The type of a packed bin determines how many of each item is packed.

We ignore rearrangement of items.

How many types can there be?
We can describe any packing of items into a single bin by its type. The type of a packed bin determines how many of each item is packed. We ignore rearrangement of items.

How many types can there be?

There are between 0 and $c_b$ items of any one size.
We can describe any packing of items into a single bin by its type. The type of a packed bin determines how many of each item is packed, and we ignore rearrangement of items. How many types can there be? There are between 0 and \( c_b \) items of any one size. There are \( c_s \) different sizes.
A special case of **BINPACKING**

- We can describe any packing of items into a single bin by its *type*.
- The *type* of a packed bin determines how many of each item is packed.
  
  *we ignore rearrangement of items*

- How many types can there be?

  There are between 0 and $c_b$ items of any one size.
  
  There are $c_s$ different sizes.
  
  The number of types $\leq$
We can describe any packing of items into a single bin by its \textit{type}.

The \textit{type} of a packed bin determines how many of each item is packed.

\textit{We ignore rearrangement of items.}

How many types can there be?

There are between 0 and $c_b$ items of any one size.

There are $c_s$ different sizes.

The number of types $\leq (c_b + 1) \times (c_b + 1) \times \ldots \times (c_b + 1)$.
We can describe any packing of items into a single bin by its type.
The type of a packed bin determines how many of each item is packed.

How many types can there be?

There are between 0 and $c_b$ items of any one size.

There are $c_s$ different sizes.

The number of types $\leq (c_b + 1) \times (c_b + 1) \times \ldots \times (c_b + 1) = (c_b + 1)^{c_s}$
A special case of BinPacking

- We can describe any packing $S$ into $b \leq n$ bins by the number of bins of each type.
We can describe any packing $S$ into $b \leq n$ bins by the number of bins of each type.
A special case of \textsc{BinPacking}

\begin{itemize}
\item We can describe any packing $S$ into $b \leq n$ bins by the number of bins of each type.
\end{itemize}

we ignore rearrangement of bins
A special case of **BinPacking**

- We can describe any packing $S$ into $b \leq n$ bins by the number of bins of each type.

  - We ignore rearrangement of bins.

- $c_b$ items fit in each bin

- $c_s$ diff. item sizes

- Type 1: 0
- Type 2: 0
- Type 3: 1
- Type 4: 2
- Type 5: 3
- Type 6: 1
- Type 7: 0
A special case of **BinPacking**

- We can describe any packing $S$ into $b \leq n$ bins by the number of bins of each type.

- we ignore rearrangement of bins
A special case of **BinPacking**

- We can describe any packing $S$ into $b \leq n$ bins by the number of bins of each type.
  
- How *different* packings can there be?
A special case of **BinPacking**

- We can describe any packing $S$ into $b \leq n$ bins by the number of bins of each type.

  - **we ignore rearrangement of bins**

- How *different* packings can there be?

  There are between 0 and $n$ bins of any type.
A special case of **BinPacking**

- We can describe any packing $S$ into $b \leq n$ bins by the number of bins of each type. *We ignore rearrangement of bins.*

- How different packings can there be?

  There are between 0 and $n$ bins of any type.
  There are at most $(c_b + 1)^{c_s}$ different types.
A special case of **BINPACKING**

- We can describe any packing $S$ into $b \leq n$ bins by the number of bins of each type.
- We ignore rearrangement of bins.

- How different packings can there be?
  - There are between 0 and $n$ bins of any type.
  - There are at most $(c_b + 1)^{c_s}$ different types.
  - Number of packings $\leq$
We can describe any packing $S$ into $b \leq n$ bins by the number of bins of each type we ignore rearrangement of bins.

How different packings can there be?

There are between 0 and $n$ bins of any type.
There are at most $(c_b + 1)^{c_s}$ different types.

Number of packings $\leq (n + 1) \times (n + 1) \times \ldots \times (n + 1)\overline{(c_b + 1)^{c_s}}$
A special case of **BINPACKING**

- **$c_b$** items fit in each bin
- **$c_s$** diff. item sizes

- We can describe any packing $S$ into $b \leq n$ bins by the number of bins of each type.
  
  *we ignore rearrangement of bins*

- How *different* packings can there be?
  
  There are between 0 and $n$ bins of any type.
  
  There are at most $(c_b + 1)^{c_s}$ different types.

  \[
  \text{number of packings} \leq (n + 1) \times (n + 1) \times \ldots \times (n + 1) = (n+1)^{(c_b+1)^{c_s}}
  \]

\[
= (c_b + 1)^{c_s}
\]
A special case of **BinPacking**

- We can describe any packing $S$ into $b \leq n$ bins by the number of bins of each type.

<table>
<thead>
<tr>
<th>Items of Type</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>7</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3/8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3/8</td>
<td>3/8</td>
</tr>
<tr>
<td>3/8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3/8</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4/8</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3/8</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3/8</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3/8</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3/8</td>
</tr>
</tbody>
</table>

$c_b$ items fit in each bin  
$c_s$ diff. item sizes

- $0 \times$ Type 1
- $0 \times$ Type 2
- $1 \times$ Type 3
- $2 \times$ Type 4
- $3 \times$ Type 5
- $1 \times$ Type 6
- $0 \times$ Type 7
A special case of **BinPacking**

- We can describe any packing $S$ into $b \leq n$ bins by the number of bins of each type.
- However, not all these different packings are *valid*.
A special case of **BinPacking**

- We can describe any packing $S$ into $b \leq n$ bins by the number of bins of each type.

- However, not all these different packings are **valid**. Many of them use too few or many items of a size.
A special case of **BinPacking**

- We can describe any packing $S$ into $b \leq n$ bins by the number of bins of each type.
- However, not all these different packings are *valid*. Many of them use too few or many items of a size.
- We check each of the different packings to see if it is valid.
A special case of **BinPacking**

- We can describe any packing $S$ into $b \leq n$ bins by the number of bins of each type.

- However, not all these different packings are **valid**. Many of them use too few or many items of a size.

- We check each of the different packings to see if it is valid and output the valid one which uses the fewest bins.
A special case of \textsc{BinPacking}

- We can describe any packing $S$ into $b \leq n$ bins by the number of bins of each type.

- However, not all these different packings are valid. Many of them use too few or many items of a size.

- We check each of the different packings to see if it is valid and output the valid one which uses the fewest bins.

- This takes $O(n \cdot (n + 1)^{(c_b+1)c_s})$ time and exactly solves \textsc{BinPacking} (it outputs an optimal packing).
Towards an APTAS

The APTAS for BINPACKING will use a three step process:

**Step 1** Remove all the small items

- Only a constant number of the remaining large items will fit into a single bin
- The small items will be packed after (greedily)

**Step 2** Divide the items into groups

- Sizes of items in each group are then rounded up to match the size of the largest member
- This will leave a constant number of item sizes

**Step 3** Use the poly-time algorithm for the remaining special case

the constants will depend on $\epsilon$
Towards an APTAS

The APTAS for B\textsc{inpacking} will use a three step process:

\textbf{Step 1} Remove all the \textit{small} items

\begin{itemize}
  \item Only a constant number of the remaining large items will fit into a single bin
  \item The small items will be packed after (greedily)
\end{itemize}

\textbf{Step 2} Divide the items into groups

\begin{itemize}
  \item Sizes of items in each group are then rounded up to match the size of the largest member
  \item This will leave a constant number of item sizes
\end{itemize}

\textbf{Step 3} Use the poly-time algorithm for the remaining special case

the \textit{constants} will depend on $\epsilon$
Remove the small items

**Lemma** Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon / 2$ into $b$ bins, in *polynomial time* we can find a packing of *all items* in $S$ which either uses:

$$b \text{ bins } \text{ or } (1 + \epsilon)\text{Opt} + 1 \text{ bins}$$
Removing the small items

**Lemma** Let $0 < \varepsilon < 1$. Given a packing of the items $a \in S$ with size $a > \varepsilon/2$ into $b$ bins, in *polynomial time* we can find a packing of *all items* in $S$ which either uses:

- $b$ bins
- $(1 + \varepsilon)\text{Opt} + 1$ bins

- Take the packing of large items ($> \varepsilon/2$)
Removing the small items

**Lemma** Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into $b$ bins, in *polynomial time* we can find a packing of *all items* in $S$ which either uses:

\[ b \text{ bins} \quad \text{or} \quad (1 + \epsilon)\text{Opt} + 1 \text{ bins} \]

- Take the packing of large items ($> \epsilon/2$)
- Pack the small items ($\leq \epsilon/2$) on top of the large items greedily
Removing the small items

**Lemma** Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into $b$ bins, in *polynomial time* we can find a packing of *all items* in $S$ which either uses:

$$b \text{ bins} \quad \text{or} \quad (1 + \epsilon)\text{Opt} + 1 \text{ bins}$$

- Take the packing of large items ($> \epsilon/2$)
- Pack the small items ($\leq \epsilon/2$) on top of the large items greedily
Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into $b$ bins, in *polynomial time* we can find a packing of *all items* in $S$ which either uses:

\[ b \text{ bins} \quad \text{or} \quad (1 + \epsilon)\text{Opt} + 1 \text{ bins} \]

... anywhere you like - just but don’t use an extra bin unless you have to...

- Take the packing of large items ($> \epsilon/2$)
- Pack the small items ($\leq \epsilon/2$) on top of the large items greedily
Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into $b$ bins, in polynomial time we can find a packing of all items in $S$ which either uses:

$$b \text{ bins} \quad \text{or} \quad (1 + \epsilon)\text{Opt} + 1 \text{ bins}$$

- Take the packing of large items ($> \epsilon/2$)
- Pack the small items ($\leq \epsilon/2$) on top of the large items greedily
Removing the small items

**Lemma** Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into $b$ bins, in *polynomial time* we can find a packing of all items in $S$ which either uses:

\[ b \text{ bins} \quad \text{or} \quad (1 + \epsilon)\text{Opt} + 1 \text{ bins} \]

- Take the packing of large items ($> \epsilon/2$)
- Pack the small items ($\leq \epsilon/2$) on top of the large items greedily

Either: We don’t use any extra bins
or every bin (except possibly the last) is $1 - (\epsilon/2)$ full
Removing the small items

**Lemma** Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into $b$ bins, in *polynomial time* we can find a packing of *all items* in $S$ which either uses:

\[ b \text{ bins} \quad \text{or} \quad (1 + \epsilon)\text{Opt} + 1 \text{ bins} \]

- Take the packing of large items ($> \epsilon/2$)
- Pack the small items ($\leq \epsilon/2$) on top of the large items greedily

Either: We don’t use any extra bins

or every bin (except possibly the last) is $1 - (\epsilon/2)$ full
Removing the small items

**Lemma** Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into $b$ bins, in *polynomial time* we can find a packing of *all items* in $S$ which either uses:

- $b$ bins 
- $(1 + \epsilon)\text{Opt} + 1$ bins

- Take the packing of large items ($> \epsilon/2$)
- Pack the small items ($\leq \epsilon/2$) on top of the large items greedily

Either: We don’t use any extra bins

or every bin (except possibly the last) is $1 - (\epsilon/2)$ full
Removing the small items

**Lemma** Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into $b$ bins, in polynomial time we can find a packing of all items in $S$ which either uses:

- $b$ bins  
- $(1 + \epsilon)\text{Opt} + 1$ bins

- Take the packing of large items ($> \epsilon/2$)
- Pack the small items ($\leq \epsilon/2$) on top of the large items greedily

Either: We don’t use any extra bins

or every bin (except possibly the last) is $1 - (\epsilon/2)$ full
Removing the small items

**Lemma** Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into $b$ bins, in *polynomial time* we can find a packing of *all items* in $S$ which either uses:

\[ b \text{ bins} \quad \text{or} \quad (1 + \epsilon) \text{Opt} + 1 \text{ bins} \]

- small items don’t fit in here (so they are very well packed)
- Take the packing of large items ($> \epsilon/2$)
- Pack the small items ($\leq \epsilon/2$) on top of the large items greedily

Either: We don’t use any extra bins

or every bin (except possibly the last) is $1 - (\epsilon/2)$ full
Removing the small items

**Lemma** Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into $b$ bins, in polynomial time we can find a packing of all items in $S$ which either uses:

$b$ bins or $(1 + \epsilon) \text{Opt} + 1$ bins

- Take the packing of large items ($> \epsilon/2$)
- Pack the small items ($\leq \epsilon/2$) on top of the large items greedily

Either: We don’t use any extra bins

or every bin (except possibly the last) is $1 - (\epsilon/2)$ full
Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into $b$ bins, in polynomial time we can find a packing of all items in $S$ which either uses:

\[ b \text{ bins} \quad \text{or} \quad (1 + \epsilon)\text{Opt} + 1 \text{ bins} \]

We can now ignore all the small items and focus on finding a good packing of the large items.
Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into $b$ bins, in polynomial time we can find a packing of all items in $S$ which either uses:

$$b \text{ bins} \quad \text{or} \quad (1 + \epsilon) \text{Opt} + 1 \text{ bins}$$

We can now ignore all the small items and focus on finding a good packing of the large items.

How many large items fit in a single bin?
Removing the small items

Lemma Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into $b$ bins, in polynomial time we can find a packing of all items in $S$ which either uses:

$$b \text{ bins \ or \ } (1 + \epsilon)\text{Opt} + 1 \text{ bins}$$

We can now ignore all the small items and focus on finding a good packing of the large items.

How many large items fit in a single bin?

Each is larger than $\epsilon/2$...
Removing the small items

**Lemma** Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into $b$ bins, in *polynomial time* we can find a packing of all items in $S$ which either uses:

$b$ bins \hspace{1cm} or \hspace{1cm} (1 + \epsilon)\text{Opt} + 1$ bins

We can now ignore all the small items and focus on finding a good packing of the large items

How many large items fit in a single bin?

Each is larger than $\epsilon/2$ . . .

so at most $2/\epsilon$
Removing the small items

**Lemma** Let $0 < \epsilon < 1$. Given a packing of the items $a \in S$ with size $a > \epsilon/2$ into $b$ bins, in *polynomial time* we can find a packing of *all items* in $S$ which either uses:

\[
b \text{ bins} \quad \text{or} \quad (1 + \epsilon) \text{Opt} + 1 \text{ bins}
\]

We can now ignore all the small items and focus on finding a good packing of the large items.

How many large items fit in a single bin?

Each is larger than $\epsilon/2$... so at most $2/\epsilon$ which is a constant :)
Reducing the number of item sizes

- We divide the items by size, into groups of size $k$
  
  the smallest group might contain fewer than $k$ items
Reducing the number of item sizes

- We divide the items by size, into groups of size $k$
  
  the smallest group might contain fewer than $k$ items

- We define a new set of items $S'$ where each item is rounded up
  
  so each item in a group has the same size
Reducing the number of item sizes

- We divide the items by size, into groups of size $k$
  
  the smallest group might contain fewer than $k$ items

- We define a new set of items $S'$ where each item is rounded up
  
  so each item in a group has the same size
Reducing the number of item sizes

- We divide the items by size, into groups of size $k$
  - the smallest group might contain fewer than $k$ items

- We define a new set of items $S'$ where each item is rounded up
  - so each item in a group has the same size
  - and the largest group is removed
Reducing the number of item sizes

- We divide the items by size, into groups of size $k$
  - the smallest group might contain fewer than $k$ items

- We define a new set of items $S'$ where each item is rounded up
  - so each item in a group has the same size
    - and the largest group is removed

- Notice that $S'$ contains only $n/k$ distinct item sizes
Reducing the number of item sizes

- We divide the items by size, into groups of size $k$
  
  the smallest group might contain fewer than $k$ items

- We define a new set of items $S'$ where each item is rounded up
  
  so each item in a group has the same size
  
  and the largest group is removed

- Notice that $S'$ contains only $n/k$ distinct item sizes
  
  ($k$ will be big enough so that $n/k \leq 4/\epsilon^2$ - which is a constant)
Reducing the number of item sizes

**Lemma** Let $S'$ be $S$ after linear grouping (with groups of size $k$).

$$\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.$$  

Further, any packing of $S'$ can be converted into a packing of $S$ in polynomial time by using at most $k$ extra bins.
Reducing the number of item sizes

**Lemma** Let $S'$ be $S$ after linear grouping (with groups of size $k$).

$$\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.$$ 

Further, any packing of $S'$ can be converted into a packing of $S$ in polynomial time by using at most $k$ extra bins.

- Here $\text{Opt}(S)$ is the fewest bins required to pack $S$
- Similarly $\text{Opt}(S')$ is the fewest bins required to pack $S'$
Reducing the number of item sizes

Lemma Let \( S' \) be \( S \) after linear grouping (with groups of size \( k \)).

\[
\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.
\]

Further, any packing of \( S' \) can be converted into a packing of \( S \) in polynomial time by using at most \( k \) extra bins.
Reducing the number of item sizes

**Lemma** Let $S'$ be $S$ after linear grouping (with groups of size $k$).

$$\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.$$
Reducing the number of item sizes

**Lemma** Let $S'$ be $S$ after linear grouping (with groups of size $k$).

$$\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.$$ 

**Proof**
Reducing the number of item sizes

**Lemma** Let \( S' \) be \( S \) after linear grouping (with groups of size \( k \)).

\[
\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k. 
\]

**Proof**

If you can pack \( S \) into \( b \) bins, you can pack \( S' \) into \( b \) bins.
Reducing the number of item sizes

**Lemma** Let \( S' \) be \( S \) after linear grouping (with groups of size \( k \)).

\[
\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.
\]

**Proof**

If you can pack \( S \) into \( b \) bins,
you can pack \( S' \) into \( b \) bins

Take the packing of \( S \)
and replace each item as shown
Reducing the number of item sizes

Lemma Let $S'$ be $S$ after linear grouping (with groups of size $k$).

$$\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.$$ 

Proof

If you can pack $S$ into $b$ bins, you can pack $S'$ into $b$ bins.

Take the packing of $S$ and replace each item as shown:

Each item from $S$ is replaced with one no larger from $S'$. 

unpack these
Reducing the number of item sizes

**Lemma** Let $S'$ be $S$ after linear grouping (with groups of size $k$).

$$\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.$$ 

**Proof**

If you can pack $S$ into $b$ bins,
you can pack $S'$ into $b$ bins.

Take the packing of $S$
and replace each item as shown.

Each item from $S$ is replaced
with one no larger from $S'$

So the packing is valid and hence
$$\text{Opt}(S') \leq \text{Opt}(S).$$
Reducing the number of item sizes

Lemma Let $S'$ be $S$ after linear grouping (with groups of size $k$).

\[
\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.
\]

Proof

If you can pack $S'$ into $b$ bins,
you can pack $S$ into $b + k$ bins
Reducing the number of item sizes

**Lemma** Let $S'$ be $S$ after linear grouping (with groups of size $k$).

$$\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.$$  

**Proof**

If you can pack $S'$ into $b$ bins,
you can pack $S$ into $b + k$ bins

Take the packing of $S'$
and replace each item as shown
Reducing the number of item sizes

**Lemma** Let $S'$ be $S$ after linear grouping (with groups of size $k$).

$$\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.$$ 

**Proof**

If you can pack $S'$ into $b$ bins,
you can pack $S$ into $b + k$ bins

Take the packing of $S'$
and replace each item as shown
Reducing the number of item sizes

**Lemma** Let \( S' \) be \( S \) after linear grouping (with groups of size \( k \)).

\[
\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.
\]

**Proof**

If you can pack \( S' \) into \( b \) bins,

you can pack \( S \) into \( b + k \) bins

Take the packing of \( S' \)

and replace each item as shown

Each item from \( S' \) is replaced

with one no larger from \( S \)
Reducing the number of item sizes

**Lemma** Let $S'$ be $S$ after linear grouping (with groups of size $k$).

\[ \text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k. \]

**Proof**

If you can pack $S'$ into $b$ bins,
you can pack $S$ into $b + k$ bins

Take the packing of $S'$
and replace each item as shown

Each item from $S'$ is replaced
with one no larger from $S$

The $k$ largest items are
given their own extra bins
Reducing the number of item sizes

**Lemma** Let $S'$ be $S$ after linear grouping (with groups of size $k$).

$$\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.$$ 

**Proof**

If you can pack $S'$ into $b$ bins,
you can pack $S$ into $b + k$ bins
Reducing the number of item sizes

Lemma Let $S'$ be $S$ after linear grouping (with groups of size $k$).

$$\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.$$ 

Proof

If you can pack $S'$ into $b$ bins, you can pack $S$ into $b + k$ bins.

This gives a valid packing of $S$. 

![Diagram of item sizes and packing into bins](image-url)
Reducing the number of item sizes

**Lemma** Let $S'$ be $S$ after linear grouping (with groups of size $k$).

\[
\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.
\]

**Proof**

If you can pack $S'$ into $b$ bins,
you can pack $S$ into $b + k$ bins

This gives a valid packing of $S$

Hence,

\[
\text{Opt}(S) \leq \text{Opt}(S') + k
\]
Reducing the number of item sizes

**Lemma** Let $S'$ be $S$ after linear grouping (with groups of size $k$).

$$\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.$$  

**Proof**

If you can pack $S'$ into $b$ bins, you can pack $S$ into $b + k$ bins.

This gives a valid packing of $S$.

Hence, 

$$\text{Opt}(S) \leq \text{Opt}(S') + k.$$  

Note that both transformations take polynomial time.
Reducing the number of item sizes

**Lemma** Let $S'$ be $S$ after linear grouping (with groups of size $k$).

\[
\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.
\]

Further, any packing of $S'$ can be converted into a packing of $S$ in polynomial time by using at most $k$ extra bins.
Reducing the number of item sizes

**Lemma** Let $S'$ be $S$ after linear grouping (with groups of size $k$).

$$\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.$$  

Further, any packing of $S'$ can be converted into a packing of $S$ in polynomial time by using at most $k$ extra bins.

We set $k = \lfloor n \cdot (\epsilon^2/2) \rfloor$ and let $S$ be the set of large items which implies that...
Reducing the number of item sizes

**Lemma** Let \( S' \) be \( S \) after linear grouping (with groups of size \( k \)).

\[
\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.
\]

Further, any packing of \( S' \) can be converted into a packing of \( S \) in polynomial time by using at most \( k \) extra bins.

We set \( k = \lfloor n \cdot (\epsilon^2/2) \rfloor \) and let \( S \) be the set of large items which implies that...

\[
k \leq \epsilon \cdot \text{Opt}(S)
\]
Reducing the number of item sizes

Lemma Let $S'$ be $S$ after linear grouping (with groups of size $k$).

$$\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.$$ 

Further, any packing of $S'$ can be converted into a packing of $S$ in polynomial time by using at most $k$ extra bins.

We set $k = \lfloor n \cdot (\epsilon^2/2) \rfloor$ and let $S$ be the set of large items which implies that...

$$k \leq \epsilon \cdot \text{Opt}(S') \quad \text{(because each of the $n$ items in $S$ have size at least $\epsilon/2$)}$$
Reducing the number of item sizes

**Lemma** Let $S'$ be $S$ after linear grouping (with groups of size $k$).

$$\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.$$ 

Further, any packing of $S'$ can be converted into a packing of $S$ in polynomial time by using at most $k$ extra bins.

We set $k = \lfloor n \cdot (\epsilon^2/2) \rfloor$ and let $S$ be the set of large items which implies that...

$$k \leq \epsilon \cdot \text{Opt}(S) \quad \text{(because each of the } n \text{ items in } S \text{ have size at least } \epsilon/2)$$

If we can find the optimal packing of $S'$, which uses $\text{Opt}(S')$ bins

we can convert it into a packing of $S$ which uses

$$\text{Opt}(S') + k \leq (1 + \epsilon)\text{Opt}(S) \text{ bins}$$
Reducing the number of item sizes

Lemma Let $S'$ be $S$ after linear grouping (with groups of size $k$).

$$\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.$$  

Further, any packing of $S'$ can be converted into a packing of $S$ in polynomial time by using at most $k$ extra bins.

We set $k = \lfloor n \cdot (\epsilon^2/2) \rfloor$ and let $S$ be the set of large items which implies that...

$$k \leq \epsilon \cdot \text{Opt}(S) \quad \text{(because each of the $n$ items in $S$ have size at least $\epsilon/2$)}$$

If we can find the optimal packing of $S'$, which uses $\text{Opt}(S')$ bins we can convert it into a packing of $S$ which uses

$$\text{Opt}(S') + k \leq (1 + \epsilon)\text{Opt}(S)$$

$S'$ contains $4/\epsilon^2$ distinct item sizes and only $2/\epsilon$ items fit in each bin...
Reducing the number of item sizes

Lemma Let \( S' \) be \( S \) after linear grouping (with groups of size \( k \)).

\[
\text{Opt}(S') \leq \text{Opt}(S) \leq \text{Opt}(S') + k.
\]

Further, any packing of \( S' \) can be converted into a packing of \( S \) in polynomial time by using at most \( k \) extra bins.

We set \( k = \lfloor n \cdot (\epsilon^2/2) \rfloor \) and let \( S \) be the set of large items which implies that.

\[
k \leq \epsilon \cdot \text{Opt}(S)
\]

(because each of the \( n \) items in \( S \) have size at least \( \epsilon/2 \))

If we can find the optimal packing of \( S' \), which uses \( \text{Opt}(S') \) bins

we can convert it into a packing of \( S \) which uses

\[
\text{Opt}(S') + k \leq (1 + \epsilon)\text{Opt}(S)
\]

bins

\( S' \) contains \( 4/\epsilon^2 \) distinct item sizes

and only \( 2/\epsilon \) items fit in each bin.

I.e. we can optimally pack \( S' \) in polynomial time.
The overall APTAS

**Step 1** Remove all the small items (those with size $\leq \epsilon/2$)

- At most $2/\epsilon$ of the remaining large items will fit into a single bin
- The small items will be packed after (using First-Fit)

**Step 2** Divide the items into $k = \lfloor n \cdot (\epsilon^2/2) \rfloor$ groups

- Items in each group are then rounded up and the largest group is removed
- This leaves at most $4/\epsilon^2$ distinct item sizes

**Step 3** Use the poly-time algorithm for the remaining special case

- This takes $O \left( n \cdot (n + 1)^{(4/\epsilon^2+1)^{2/\epsilon}} \right)$ time
  
  as $c_s = 4/\epsilon^2$ and $c_b = 2/\epsilon$. 
The overall APTAS

**Step 1** Remove all the small items (those with size $\leq \epsilon/2$)

- At most $2/\epsilon$ of the remaining large items will fit into a single bin
- The small items will be packed after (using First-Fit)

**Step 2** Divide the items into $k = \lfloor n \cdot (\epsilon^2/2) \rfloor$ groups

- Items in each group are then rounded up and the largest group is removed
- This leaves at most $4/\epsilon^2$ distinct item sizes

**Step 3** Use the poly-time algorithm for the remaining special case

- This takes $O \left( n \cdot (n + 1) \left( \frac{4}{\epsilon^2} + 1 \right)^{2/\epsilon} \right)$ time as $c_s = 4/\epsilon^2$ and $c_b = 2/\epsilon$. 
The overall APTAS

**Step 1** Remove all the small items (those with size $\leq \epsilon/2$)

- At most $2/\epsilon$ of the remaining large items will fit into a single bin
- The small items will be packed after (using First-Fit)

**Step 2** Divide the items into $k = \lfloor n \cdot (\epsilon^2/2) \rfloor$ groups

- Items in each group are then rounded up and the largest group is removed
- This leaves at most $4/\epsilon^2$ distinct item sizes

**Step 3** Use the poly-time algorithm for the remaining special case

- This takes $O \left( n \cdot (n + 1)^{\left(4/\epsilon^2 + 1\right)^{2/\epsilon}} \right)$ time

as $c_s = 4/\epsilon^2$ and $c_b = 2/\epsilon$. 
The overall APTAS

**Step 1** Remove all the small items (those with size \( \leq \frac{\epsilon}{2} \))
- At most \( \frac{2}{\epsilon} \) of the remaining large items will fit into a single bin
- The small items will be packed after (using First-Fit)

**Step 2** Divide the items into \( k = \left\lfloor n \cdot \left( \frac{\epsilon^2}{2} \right) \right\rfloor \) groups
- Items in each group are then rounded up and the largest group is removed
- This leaves at most \( \frac{4}{\epsilon^2} \) distinct item sizes

**Step 3** Use the poly-time algorithm for the remaining special case
- This takes \( O \left( n \cdot (n + 1)^{\left( \frac{4}{\epsilon^2} + 1 \right)^{2/\epsilon}} \right) \) time as \( c_s = \frac{4}{\epsilon^2} \) and \( c_b = \frac{2}{\epsilon} \).

**Theorem** For any \( 0 < \epsilon < 1 \), the algorithm presented runs in polynomial time and returns a packing of any set of items using at most \((1 + \epsilon)\text{Opt} + 1\) bins.
Conclusions

- There is no $\alpha$-approximation for BinPacking with $\alpha < 3/2$ unless $P = NP$.
- We saw an APTAS for BinPacking.
- There is a poly-time algorithm which outputs a solution using at most,
  \[
  \frac{11}{9} \cdot \text{Opt} + 1 \text{ bins}
  \]
- The First Fit Decreasing algorithm uses at most,
  \[
  \boxed{\frac{11}{9} \cdot \text{Opt} + 1} \text{ bins}
  \]
- This in turn implies that there is no PTAS for BinPacking unless $P = NP$.
- There is also an AFPTAS for bin packing.
- There is a poly-time algorithm which outputs a solution using at most,
  \[
  \text{Opt} + O(\log^2 \text{Opt}) = \left(1 + \frac{O(\log^2 \text{Opt})}{\text{Opt}}\right) \cdot \text{Opt} \text{ bins}
  \]
Conclusions

• There is no $\alpha$-approximation for \textsc{BinPacking} with $\alpha < 3/2$ unless $P = NP$

• This in turn implies that there is no PTAS for \textsc{BinPacking} unless $P = NP$

• The First Fit Decreasing algorithm uses at most,

$$\frac{11}{9} \cdot \text{Opt} + 1 \text{ bins}$$

• We saw an APTAS for \textsc{BinPacking}.

  \textit{there is also an AFPTAS for bin packing}

• There is a poly-time algorithm which outputs a solution using at most,

$$\text{Opt} + O(\log^2 \text{Opt}) = \left( 1 + \frac{O(\log^2 \text{Opt})}{\text{Opt}} \right) \cdot \text{Opt} \text{ bins}$$

• A poly-time algorithm which uses at most $\text{Opt} + 1$ bins is open