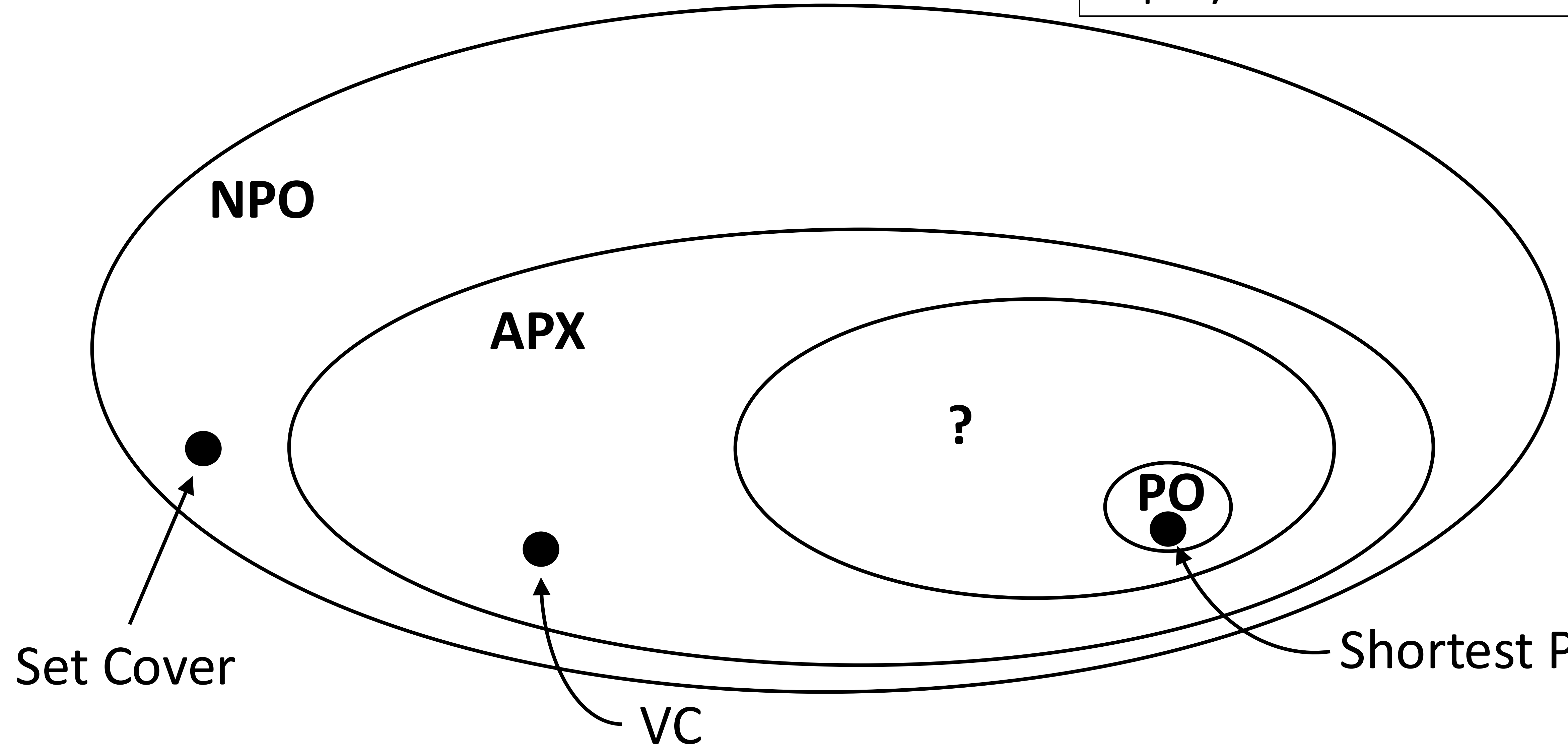
(F)PTAS
CSCI 532

Final Presentation Scheduling

- 12/02, 12/03, 12/11
- Email me if you have restrictions on which days you can present.

Approximability Hierarchy

PO: Optimization problems that can be optimally solved in polynomial time.



Knapsack

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Example:

$W = 11$

Item	Value	Weight
1	1	1
2	6	2
3	18	5
4	22	6
5	28	7

Knapsack

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Example:

$W = 11$

Item	Value	Weight
1	1	1
2	6	2
3	18	5
4	22	6
5	28	7

$\{1, 2, 5\}$: weight = 10, value = 35 ✓

$\{3, 4\}$: weight = 11, value = 40

$\{4, 5\}$: weight = 13, value = 50 ✗

Knapsack

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

How could we approach this?

Greedy?

ILP?

Flow Network?

Dynamic Program?

Item	Value	Weight
1	11	6
2	11	6
3	20	10

$W = 12$

Knapsack – Dynamic Programming

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

Is there optimal substructure?

I.e. If I have an optimal solution to some instance, does that imply an optimal solution to a different instance?

Knapsack – Dynamic Programming

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

Is there optimal substructure?

I.e. If I have an optimal solution to some instance, does that imply an optimal solution to a different instance?

Yes, removing some item must give an optimal selection for the remaining weight.

E.g. If {item_1, item_3, item_7} is optimal for weight of 10, and item_3 weighs 3, {item_1, item_7} ***must*** be optimal for a weight of 7.

Knapsack – Dynamic Programming

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Define $\text{opt}(i, w) =$ maximum value achievable for items $\{1, \dots, i\}$ and knapsack of capacity w.

Knapsack – Dynamic Programming

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Define $\text{opt}(i, w) =$ maximum value achievable for items $\{1, \dots, i\}$ and knapsack of capacity w.

Case 1: If i **is not** in the optimal solution for $\{1, \dots, i\}$:

$$\text{opt}(i, w) = \text{opt}(i - 1, w)$$

Case 2: If i **is** in the optimal solution for $\{1, \dots, i\}$?

$$\text{opt}(i, w) = \text{opt}(i - 1, w - w_i) + v_i$$

Knapsack – Dynamic Programming

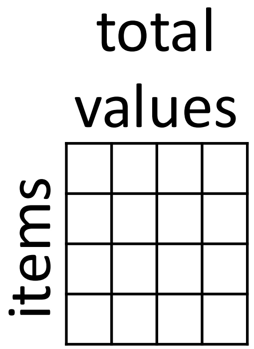
Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Define $\text{opt}(i, V) =$ minimum weight of subset of items $\{1, \dots, i\}$ that gives value at least V.

Knapsack – Dynamic Programming

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Define $\text{opt}(i, V) =$ minimum weight of subset of items $\{1, \dots, i\}$ that gives value at least V.

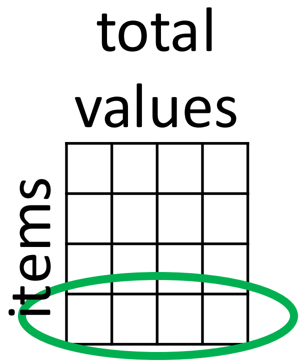


Knapsack – Dynamic Programming

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Define $\mathrm{opt}(i, V) =$ minimum weight of subset of items $\{1, \ldots, i\}$ that gives value at least V.

To find the optimal solution, find largest V such that $\mathrm{opt}(n, V) \leq W$.



Find largest value with compatible weight.

Knapsack – Dynamic Programming

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Define $\text{opt}(i, V) =$ minimum weight of subset of items $\{1, \dots, i\}$ that gives value at least V.

Case 1: If i **is not** in the optimal solution for $\{1, \dots, i\}$:

?

Case 2: If i **is** in the optimal solution for $\{1, \dots, i\}$:

?

Knapsack – Dynamic Programming

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Define $\text{opt}(i, V) =$ minimum weight of subset of items $\{1, \ldots, i\}$ that gives value at least V.

Case 1: If i **is not** in the optimal solution for $\{1, \ldots, i\}$:

$$\text{opt}(i, V) = \text{opt}(i - 1, V)$$

Case 2: If i **is** in the optimal solution for $\{1, \ldots, i\}$:

Knapsack – Dynamic Programming

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Define $\mathrm{opt}(i, V) =$ minimum weight of subset of items $\{1, \dots, i\}$ that gives value at least V.

Case 1: If i **is not** in the optimal solution for $\{1, \dots, i\}$:

$$\mathrm{opt}(i, V) = \mathrm{opt}(i - 1, V)$$

Case 2: If i **is** in the optimal solution for $\{1, \dots, i\}$:

$$\mathrm{opt}(i, V) = \mathrm{opt}(i - 1, V - v_i) + w_i$$

If item i is in the optimal solution, removing it decreases the value by v_i and weight by w_i. What remains must be the minimum weight whose value is at least $V - v_i$.

Knapsack – Dynamic Programming

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Define $\mathrm{opt}(i, V) =$ minimum weight of subset of items $\{1, \ldots, i\}$ that gives value at least V.

$$\mathrm{opt}(i, V) = \begin{cases} 0 & \text{if } V = 0 \\ \infty & \text{if } i = 0 \text{ and } V > 0 \\ \min\begin{pmatrix} \mathrm{opt}(i-1, V), \\ \mathrm{opt}(i-1, V - v_i) + w_i \end{pmatrix} & \text{Otherwise} \end{cases}$$

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

Define $\text{opt}(i, V) =$ minimum weight of subset of items $\{1, \dots, i\}$ that gives value at least V.

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

Define $\text{opt}(i, V) =$ minimum weight of subset of items $\{1, \ldots, i\}$ that gives value at least V.

1. For $0 \leq i \leq n$ and $0 \leq V \leq$? , compute $\text{opt}(i, V)$.

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

Define $\text{opt}(i, V) =$ minimum weight of subset of items $\{1, \ldots, i\}$ that gives value at least V.

1. For $0 \leq i \leq n$ and $0 \leq V \leq n \max_i v_i$, compute $\text{opt}(i, V)$.

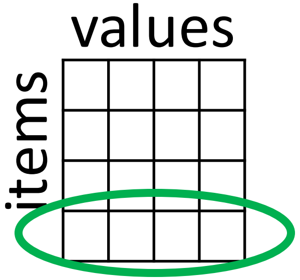
Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

Define $\text{opt}(i, V) =$ minimum weight of subset of items $\{1, \dots, i\}$ that gives value at least V.

1. For $0 \leq i \leq n$ and $0 \leq V \leq n \max_i v_i$, compute $\text{opt}(i, V)$.
2. Return $V^* = \max\{V : \text{opt}(i, V) \leq W\}$.



Find largest value with compatible weight.

Knapsack – Algorithm

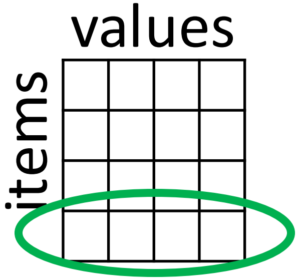
Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

Define $\text{opt}(i, V) =$ minimum weight of subset of items $\{1, \ldots, i\}$ that gives value at least V.

1. For $0 \leq i \leq n$ and $0 \leq V \leq n \max_i v_i$, compute $\text{opt}(i, V)$.
2. Return $V^* = \max\{V \colon \text{opt}(i, V) \leq W\}$.

Running Time = ?



Find largest value with compatible weight.

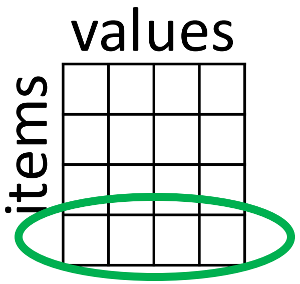
Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

Define $\text{opt}(i, V) =$ minimum weight of subset of items $\{1, \dots, i\}$ that gives value at least V.

1. For $0 \leq i \leq n$ and $0 \leq V \leq n \max_i v_i$, compute $\text{opt}(i, V)$.
2. Return $V^* = \max\{V : \text{opt}(i, V) \leq W\}$.



Find largest value with compatible weight.

Running Time = $O(n^2 \max_i v_i)$

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

Define $\text{opt}(i, V) =$ minimum weight of subset of items $\{1, \dots, i\}$ that gives value at least V.

1. For $0 \leq i \leq n$ and $0 \leq V \leq n \max_i v_i$, compute $\text{opt}(i, V)$.
2. Return $V^* = \max \{V: \text{opt}(i, V) \leq W\}$.

Running Time = $O(n^2 \max_i v_i)$

Optimal algorithm that does not run in polynomial time WRT input size.

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

Define $\text{opt}(i, V) =$ minimum weight of subset of items $\{1, \ldots, i\}$ that gives value at least V.

1. For $0 \leq i \leq n$ and $0 \leq V \leq n \max_i v_i$, compute $\text{opt}(i, V)$.
2. Return $V^* = \max\{V : \text{opt}(i, V) \leq W\}$.

$$v_i \begin{cases} 100000 \\ 200000 \\ 500000 \end{cases}$$

Running Time = $O(n^2 \max_i v_i)$

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

Define $\text{opt}(i, V) =$ minimum weight of subset of items $\{1, \ldots, i\}$ that gives value at least V.

1. For $0 \leq i \leq n$ and $0 \leq V \leq n \max_i v_i$, compute $\text{opt}(i, V)$.
2. Return $V^* = \max\{V\colon \text{opt}(i, V) \leq W\}$.

$$v_i \begin{cases} 100000 & & 1 \\ 200000 & \longrightarrow & 2 \\ 500000 & & 5 \end{cases}$$

Running Time = $O(n^2 \max_i v_i)$

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

Define $\mathrm{opt}(i, V) =$ minimum weight of subset of items $\{1, \ldots, i\}$ that gives value at least V.

1. For $0 \leq i \leq n$ and $0 \leq V \leq n \max_i v_i$, compute $\mathrm{opt}(i, V)$.
2. Return $V^* = \max\{V : \mathrm{opt}(i, V) \leq W\}$.

$$v_i \begin{cases} 100000 \\ 200000 \\ 500000 \end{cases} \longrightarrow \begin{matrix} 1 \\ 2 \\ 5 \end{matrix}$$

Running Time = $O(n^2 \max_i v_i)$

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

Define $\text{opt}(i, V) =$ minimum weight of subset of items $\{1, \ldots, i\}$ that gives value at least V.

1. For $0 \leq i \leq n$ and $0 \leq V \leq n \max_i v_i$, compute $\text{opt}(i, V)$.
2. Return $V^* = \max \{V \colon \text{opt}(i, V) \leq W\}$.

$$v_i \left\{ \begin{matrix} 163882 \\ 254301 \\ 582242 \end{matrix} \right. \longrightarrow \begin{matrix} 1 \\ 2 \\ 5 \end{matrix}$$

Running Time = $O(n^2 \max_i v_i)$

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

Define $\text{opt}(i, V) =$ minimum weight of subset of i
$\{1, \ldots, i\}$ that gives value at lea

1. For $0 \leq i \leq n$ and $0 \leq V \leq n \max_i v_i$, comput
2. Return $V^* = \max\{V \colon \text{opt}(i, V) \leq W\}$.

Running Time = $O(n^2 \max_i v_i)$

Does this change the optimal value all that much?

$$v_i \begin{cases} 163882 & \longrightarrow 1 \\ 254301 & \longrightarrow 2 \\ 582242 & \longrightarrow 5 \end{cases}$$

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

Define $\text{opt}(i, V) =$ minimum weight of subset of i
$\{1, \dots, i\}$ that gives value at lea

1. For $0 \leq i \leq n$ and $0 \leq V \leq n \max_i v_i$, comput
2. Return $V^* = \max\{V : \text{opt}(i, V) \leq W\}$.

Running Time = $O(n^2 \max_i v_i)$

Does this change the optimal value all that much?

$$v_i \begin{cases} 163882 \\ 254301 \\ 582242 \end{cases} \longrightarrow \begin{matrix} 163 \\ 254 \\ 582 \end{matrix}$$

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

Define $\text{opt}(i, V) =$ minimum weight of subset of i
$\{1, \ldots, i\}$ that gives value at lea

1. For $0 \leq i \leq n$ and $0 \leq V \leq n \max_i v_i$, comput
2. Return $V^* = \max\{V: \text{opt}(i, V) \leq W\}$.

Running Time = $O(n^2 \max_i v_i)$

Does this change the optimal value all that much?

$$v_i \begin{cases} 163882 \\ 254301 \\ 582242 \end{cases} \longrightarrow \begin{matrix} 16388 \\ 25430 \\ 58224 \end{matrix}$$

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

Define $\text{opt}(i, V) =$ minimum weight o
$\{1, \dots, i\}$ that gives

1. For $0 \leq i \leq n$ and $0 \leq V \leq n \max\limits_i$
2. Return $V^* = \max\{V : \text{opt}(i, V) \leq W\}$.

Running Time = $O(n^2 \max\limits_i v_i)$

There seems to be a tradeoff that we can control between accuracy and running time.

$$v_i \begin{cases} 163882 \\ 254301 \\ 582242 \end{cases} \longrightarrow \begin{matrix} 16388 \\ 25430 \\ 58224 \end{matrix}$$

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

1.

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

1. Let $v_{\max} = \max_i v_i$

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

1. Let $v_{\max} = \max_i v_i$
2. For each i, let $v_i' = \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor$

User-supplied approximation factor $\varepsilon > 0$.

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

1. Let $v_{\max} = \max_i v_i$
2. For each i, let $v_i' = \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor$

User-supplied approximation factor $\varepsilon > 0$.

$$\begin{array}{c} 726489 \\ 136212 \\ 384167 \end{array} \xrightarrow{\varepsilon = 0.1} \begin{array}{c} 30 \\ 5 \\ 15 \end{array}$$

$$\begin{array}{c} 726489 \\ 136212 \\ 384167 \end{array} \xrightarrow{\varepsilon = 0.001} \begin{array}{c} 3000 \\ 562 \\ 1586 \end{array}$$

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

1. Let $v_{\max} = \max_i v_i$
2. For each i, let $v_i' = \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor$
3. Run dynamic programming algorithm using $\{v_i'\}, \{w_i\}, W$

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

1. Let $v_{\max} = \max_i v_i$
2. For each i, let $v_i' = \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor$
3. Run dynamic programming algorithm using $\{v_i'\}, \{w_i\}, W$

Running Time = $O(n^2 v'_{\max})$

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

1. Let $v_{\max} = \max_i v_i$
2. For each i, let $v_i' = \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor$
3. Run dynamic programming algorithm using $\{v_i'\}, \{w_i\}, W$

Running Time = $O(n^2 {v'}_{\max}) \in O\left(n^2 \left\lfloor v_{\max} \frac{n}{\varepsilon v_{\max}} \right\rfloor\right)$

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

1. Let $v_{\max} = \max_i v_i$
2. For each i, let $v_i' = \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor$
3. Run dynamic programming algorithm using $\{v_i'\}, \{w_i\}, W$

Running Time = $O(n^2 v'_{\max}) \in O\left(n^2 \left\lfloor v_{\max} \frac{n}{\varepsilon v_{\max}} \right\rfloor\right) \in O\left(\frac{n^3}{\varepsilon}\right)$

Knapsack – Algorithm

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Algorithm:

1. Let $v_{\max} = \max_i v_i$
2. For each i, let $v_i' = \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor$
3. Run dynamic programming algorithm using $\{v_i'\}, \{w_i\}, W$

How does scaling values impact cost?

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

Need to relate $\sum_{i \in S_{\text{OPT}}} v_i$ relate to $\sum_{i \in S_{\text{ALG}}} v_i$?

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

Need to relate $\sum_{i \in S_{\text{OPT}}} v_i$ relate to $\sum_{i \in S_{\text{ALG}}} v_i$?

But the algorithm operates on v_i'.

Need to relate $\sum_{i \in S_{\text{OPT}}} v_i$ to $\sum_{i \in S_{\text{OPT}}} v_i'$ to $\sum_{i \in S_{\text{ALG}}} v_i'$ to $\sum_{i \in S_{\text{ALG}}} v_i$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

How does $\sum_{i \in S_{\text{OPT}}} v_i$ relate to $\sum_{i \in S_{\text{OPT}}} v_i'$?

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

How does $\sum_{i \in S_{\text{OPT}}} v_i$ relate to $\sum_{i \in S_{\text{OPT}}} v_i'$?

Algorithm:

1. Let $v_{\max} = \max_i v_i$
2. For each i, let $v_i' = \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor$
3. Run dynamic programming algorithm using $\{v_i'\}, \{w_i\}, W$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

How does $\sum_{i \in S_{\text{OPT}}} v_i$ relate to $\sum_{i \in S_{\text{OPT}}} v_i'$?

$$\sum_{i \in S_{\text{OPT}}} v_i' = \sum_{i \in S_{\text{OPT}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor$$

Algorithm:

1. Let $v_{\max} = \max_i v_i$
2. For each i, let $v_i' = \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor$
3. Run dynamic programming algorithm using $\{v_i'\}, \{w_i\}, W$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

How does $\sum_{i \in S_{\text{OPT}}} v_i$ relate to $\sum_{i \in S_{\text{OPT}}} v_i'$?

$$\begin{aligned}\sum_{i \in S_{\text{OPT}}} v_i' &= \sum_{i \in S_{\text{OPT}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor \\ &\geq \sum_{i \in S_{\text{OPT}}} \left(v_i \frac{n}{\varepsilon v_{\max}} - 1 \right), \text{ because?}\end{aligned}$$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

How does $\sum_{i \in S_{\text{OPT}}} v_i$ relate to $\sum_{i \in S_{\text{OPT}}} v_i'$?

$$\sum_{i \in S_{\text{OPT}}} v_i' = \sum_{i \in S_{\text{OPT}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor$$

$$\geq \sum_{i \in S_{\text{OPT}}} \left(v_i \frac{n}{\varepsilon v_{\max}} - 1 \right), \text{ because floor decreases} < 1$$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

How does $\sum_{i \in S_{\text{OPT}}} v_i$ relate to $\sum_{i \in S_{\text{OPT}}} v_i'$?

$$\begin{aligned}
\sum_{i \in S_{\text{OPT}}} v_i' &= \sum_{i \in S_{\text{OPT}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor \\
&\geq \sum_{i \in S_{\text{OPT}}} \left(v_i \frac{n}{\varepsilon v_{\max}} - 1 \right), \text{ because floor decreases} < 1 \\
&\geq \left(\sum_{i \in S_{\text{OPT}}} v_i \frac{n}{\varepsilon v_{\max}} \right) - n, \text{ because?}
\end{aligned}$$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

How does $\sum_{i \in S_{\text{OPT}}} v_i$ relate to $\sum_{i \in S_{\text{OPT}}} v_i'$?

$$\begin{aligned}
\sum_{i \in S_{\text{OPT}}} v_i' &= \sum_{i \in S_{\text{OPT}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor \\
&\geq \sum_{i \in S_{\text{OPT}}} \left(v_i \frac{n}{\varepsilon v_{\max}} - 1 \right)\text{, because floor decreases} < 1 \\
&\geq \left(\sum_{i \in S_{\text{OPT}}} v_i \frac{n}{\varepsilon v_{\max}} \right) - n\text{, because } |S_{\text{OPT}}| \leq n
\end{aligned}$$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

How does $\sum_{i \in S_{\text{OPT}}} v_i$ relate to $\sum_{i \in S_{\text{OPT}}} v_i'$?

$$\begin{aligned}
\sum_{i \in S_{\text{OPT}}} v_i' &= \sum_{i \in S_{\text{OPT}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor \\
&\geq \sum_{i \in S_{\text{OPT}}} \left(v_i \frac{n}{\varepsilon v_{\max}} - 1 \right), \text{ because floor decreases} < 1 \\
&\geq \left(\sum_{i \in S_{\text{OPT}}} v_i \frac{n}{\varepsilon v_{\max}} \right) - n, \text{ because } |S_{\text{OPT}}| \leq n \\
&= \text{OPT} \frac{n}{\varepsilon v_{\max}} - n, \text{ because?}
\end{aligned}$$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

How does $\sum_{i \in S_{\text{OPT}}} v_i$ relate to $\sum_{i \in S_{\text{OPT}}} v_i'$?

$$\begin{aligned}
\sum_{i \in S_{\text{OPT}}} v_i' &= \sum_{i \in S_{\text{OPT}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor \\
&\geq \sum_{i \in S_{\text{OPT}}} \left(v_i \frac{n}{\varepsilon v_{\max}} - 1 \right), \text{ because floor decreases } < 1 \\
&\geq \left(\sum_{i \in S_{\text{OPT}}} v_i \frac{n}{\varepsilon v_{\max}} \right) - n, \text{ because } |S_{\text{OPT}}| \leq n \\
&= \text{OPT} \frac{n}{\varepsilon v_{\max}} - n, \text{ because } \sum_{i \in S_{\text{OPT}}} v_i = \text{OPT}
\end{aligned}$$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

How does $\sum_{i \in S_{\text{OPT}}} v_i$ relate to $\sum_{i \in S_{\text{OPT}}} v_i'$?

$$\begin{aligned}
\sum_{i \in S_{\text{OPT}}} v_i' &= \sum_{i \in S_{\text{OPT}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor \\
&\geq \sum_{i \in S_{\text{OPT}}} \left(v_i \frac{n}{\varepsilon v_{\max}} - 1 \right), \text{ because floor decreases } < 1 \\
&\geq \left(\sum_{i \in S_{\text{OPT}}} v_i \frac{n}{\varepsilon v_{\max}} \right) - n, \text{ because } |S_{\text{OPT}}| \leq n \\
&= \text{OPT} \frac{n}{\varepsilon v_{\max}} - n, \text{ because } \sum_{i \in S_{\text{OPT}}} v_i = \text{OPT}
\end{aligned}$$

So, $\sum_{i \in S_{\text{OPT}}} v_i' \geq \text{OPT} \frac{n}{\varepsilon v_{\max}} - n$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

$$\sum_{i \in S_{\text{OPT}}} v_i' \geq \text{OPT} \, \frac{n}{\varepsilon v_{\max}} - n$$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

$$\sum_{i \in S_{\text{OPT}}} v_i' \geq \text{OPT}\ \frac{n}{\varepsilon v_{\max}} - n$$

How does $\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i$ relate to $\text{OPT} = \sum_{i \in S_{\text{OPT}}} v_i$?

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

$\sum_{i \in S_{\text{OPT}}} v_i' \geq \text{OPT} \frac{n}{\varepsilon v_{\max}} - n$

How does $\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i$ relate to $\text{OPT} = \sum_{i \in S_{\text{OPT}}} v_i$?

$\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

$$\sum_{i \in S_{\text{OPT}}} v_i' \geq \text{OPT} \frac{n}{\varepsilon v_{\max}} - n$$

How does $\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i$ relate to $\text{OPT} = \sum_{i \in S_{\text{OPT}}} v_i$?

$$\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i = \sum_{i \in S_{\text{ALG}}} v_i \frac{n}{\varepsilon v_{\max}} \frac{\varepsilon v_{\max}}{n}$$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

$$\sum_{i \in S_{\text{OPT}}} v_i' \geq \text{OPT}\ \frac{n}{\varepsilon v_{\max}} - n$$

How does $\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i$ relate to $\text{OPT} = \sum_{i \in S_{\text{OPT}}} v_i$?

$$\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i = \sum_{i \in S_{\text{ALG}}} v_i \frac{n}{\varepsilon v_{\max}} \frac{\varepsilon v_{\max}}{n}$$

$$\geq \sum_{i \in S_{\text{ALG}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor \frac{\varepsilon v_{\max}}{n}$$, because?

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

$$\sum_{i \in S_{\text{OPT}}} v_i' \geq \text{OPT} \frac{n}{\varepsilon v_{\max}} - n$$

How does $\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i$ relate to $\text{OPT} = \sum_{i \in S_{\text{OPT}}} v_i$?

$$\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i = \sum_{i \in S_{\text{ALG}}} v_i \frac{n}{\varepsilon v_{\max}} \frac{\varepsilon v_{\max}}{n}$$

$$\geq \sum_{i \in S_{\text{ALG}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor \frac{\varepsilon v_{\max}}{n}$$, because floor function decreases

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

$$\sum_{i \in S_{\text{OPT}}} v_i' \geq \text{OPT}\ \frac{n}{\varepsilon v_{\max}} - n$$

How does $\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i$ relate to $\text{OPT} = \sum_{i \in S_{\text{OPT}}} v_i$?

$$\begin{aligned} \text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i &= \sum_{i \in S_{\text{ALG}}} v_i \frac{n}{\varepsilon v_{\max}} \frac{\varepsilon v_{\max}}{n} \\ &\geq \sum_{i \in S_{\text{ALG}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor \frac{\varepsilon v_{\max}}{n} = \sum_{i \in S_{\text{ALG}}} v_i' \frac{\varepsilon v_{\max}}{n} \end{aligned}$$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

$$\sum_{i \in S_{\text{OPT}}} v_i' \geq \text{OPT} \, \frac{n}{\varepsilon v_{\max}} - n$$

How does $\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i$ relate to $\text{OPT} = \sum_{i \in S_{\text{OPT}}} v_i$?

$$\begin{aligned} \text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i &= \sum_{i \in S_{\text{ALG}}} v_i \frac{n}{\varepsilon v_{\max}} \frac{\varepsilon v_{\max}}{n} \\ &\geq \sum_{i \in S_{\text{ALG}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor \frac{\varepsilon v_{\max}}{n} = \sum_{i \in S_{\text{ALG}}} v_i' \frac{\varepsilon v_{\max}}{n} \\ &\geq \sum_{i \in S_{\text{OPT}}} v_i' \frac{\varepsilon v_{\max}}{n}, \text{ because?} \end{aligned}$$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

$$\sum_{i \in S_{\text{OPT}}} v_i' \geq \text{OPT} \frac{n}{\varepsilon v_{\max}} - n$$

How does $\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i$ relate to $\text{OPT} = \sum_{i \in S_{\text{OPT}}} v_i$?

$$\begin{aligned}
\text{ALG} &= \sum_{i \in S_{\text{ALG}}} v_i = \sum_{i \in S_{\text{ALG}}} v_i \frac{n}{\varepsilon v_{\max}} \frac{\varepsilon v_{\max}}{n} \\
&\geq \sum_{i \in S_{\text{ALG}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor \frac{\varepsilon v_{\max}}{n} = \sum_{i \in S_{\text{ALG}}} v_i' \frac{\varepsilon v_{\max}}{n} \\
&\geq \sum_{i \in S_{\text{OPT}}} v_i' \frac{\varepsilon v_{\max}}{n}, \text{ because } \sum_{i \in S_{\text{OPT}}} v_i' \leq \sum_{i \in S_{\text{ALG}}} v_i' \text{ since} \\
&\qquad\qquad S_{\text{ALG}} \text{ is optimal for } v_i'
\end{aligned}$$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

$$\sum_{i \in S_{\text{OPT}}} v_i' \geq \text{OPT}\ \frac{n}{\varepsilon v_{\max}} - n$$

How does $\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i$ relate to $\text{OPT} = \sum_{i \in S_{\text{OPT}}} v_i$?

$$\begin{aligned}
\text{ALG} &= \sum_{i \in S_{\text{ALG}}} v_i = \sum_{i \in S_{\text{ALG}}} v_i \frac{n}{\varepsilon v_{\max}} \frac{\varepsilon v_{\max}}{n} \\
&\geq \sum_{i \in S_{\text{ALG}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor \frac{\varepsilon v_{\max}}{n} = \sum_{i \in S_{\text{ALG}}} v_i' \frac{\varepsilon v_{\max}}{n} \\
&\geq \sum_{i \in S_{\text{OPT}}} v_i' \frac{\varepsilon v_{\max}}{n} \geq \left(\text{OPT}\ \frac{n}{\varepsilon v_{\max}} - n\right) \frac{\varepsilon v_{\max}}{n}
\end{aligned}$$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

$$\sum_{i \in S_{\text{OPT}}} v_i' \geq \text{OPT}\ \frac{n}{\varepsilon v_{\max}} - n$$

How does $\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i$ relate to $\text{OPT} = \sum_{i \in S_{\text{OPT}}} v_i$?

$$\begin{aligned}
\text{ALG} &= \sum_{i \in S_{\text{ALG}}} v_i = \sum_{i \in S_{\text{ALG}}} v_i \frac{n}{\varepsilon v_{\max}} \frac{\varepsilon v_{\max}}{n} \\
&\geq \sum_{i \in S_{\text{ALG}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor \frac{\varepsilon v_{\max}}{n} = \sum_{i \in S_{\text{ALG}}} v_i' \frac{\varepsilon v_{\max}}{n} \\
&\geq \sum_{i \in S_{\text{OPT}}} v_i' \frac{\varepsilon v_{\max}}{n} \geq \left(\text{OPT}\ \frac{n}{\varepsilon v_{\max}} - n\right) \frac{\varepsilon v_{\max}}{n} = \text{OPT} - \varepsilon v_{\max}
\end{aligned}$$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

$$\sum_{i \in S_{\text{OPT}}} v_i' \geq \text{OPT}\ \frac{n}{\varepsilon v_{\max}} - n$$

How does $\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i$ relate to $\text{OPT} = \sum_{i \in S_{\text{OPT}}} v_i$?

$$\begin{aligned}
\text{ALG} &= \sum_{i \in S_{\text{ALG}}} v_i = \sum_{i \in S_{\text{ALG}}} v_i \frac{n}{\varepsilon v_{\max}} \frac{\varepsilon v_{\max}}{n} \\
&\geq \sum_{i \in S_{\text{ALG}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor \frac{\varepsilon v_{\max}}{n} = \sum_{i \in S_{\text{ALG}}} v_i' \frac{\varepsilon v_{\max}}{n} \\
&\geq \sum_{i \in S_{\text{OPT}}} v_i' \frac{\varepsilon v_{\max}}{n} \geq \left(\text{OPT}\ \frac{n}{\varepsilon v_{\max}} - n\right) \frac{\varepsilon v_{\max}}{n} = \text{OPT} - \varepsilon v_{\max} \\
&\geq \text{OPT} - \ \varepsilon\,\text{OPT}, \text{ because?}
\end{aligned}$$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

$$\sum_{i \in S_{\text{OPT}}} v_i' \geq \text{OPT}\ \frac{n}{\varepsilon v_{\max}} - n$$

How does $\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i$ relate to $\text{OPT} = \sum_{i \in S_{\text{OPT}}} v_i$?

$$\begin{aligned}
\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i &= \sum_{i \in S_{\text{ALG}}} v_i \frac{n}{\varepsilon v_{\max}} \frac{\varepsilon v_{\max}}{n} \\
&\geq \sum_{i \in S_{\text{ALG}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor \frac{\varepsilon v_{\max}}{n} = \sum_{i \in S_{\text{ALG}}} v_i' \frac{\varepsilon v_{\max}}{n} \\
&\geq \sum_{i \in S_{\text{OPT}}} v_i' \frac{\varepsilon v_{\max}}{n} \geq \left(\text{OPT}\ \frac{n}{\varepsilon v_{\max}} - n\right) \frac{\varepsilon v_{\max}}{n} = \text{OPT} - \varepsilon v_{\max} \\
&\geq \text{OPT} - \varepsilon\, \text{OPT}, \text{ because } \text{OPT} \geq v_{\max} \text{ (discard overweight items)}
\end{aligned}$$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

$$\sum_{i \in S_{\text{OPT}}} v_i' \geq \text{OPT} \frac{n}{\varepsilon v_{\max}} - n$$

How does $\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i$ relate to $\text{OPT} = \sum_{i \in S_{\text{OPT}}} v_i$?

$$\begin{aligned}
\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i &= \sum_{i \in S_{\text{ALG}}} v_i \frac{n}{\varepsilon v_{\max}} \frac{\varepsilon v_{\max}}{n} \\
&\geq \sum_{i \in S_{\text{ALG}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor \frac{\varepsilon v_{\max}}{n} = \sum_{i \in S_{\text{ALG}}} v_i' \frac{\varepsilon v_{\max}}{n} \\
&\geq \sum_{i \in S_{\text{OPT}}} v_i' \frac{\varepsilon v_{\max}}{n} \geq \left(\text{OPT} \frac{n}{\varepsilon v_{\max}} - n\right) \frac{\varepsilon v_{\max}}{n} = \text{OPT} - \varepsilon v_{\max} \\
&\geq \text{OPT} - \varepsilon \, \text{OPT}, \text{ because } \text{OPT} \geq v_{\max} \text{ (discard overweight items)} \\
&= (1 - \varepsilon) \, \text{OPT}
\end{aligned}$$

Knapsack – Performance

S_{ALG} = Set of algorithm selected items. S_{OPT} = Set of optimal items.

$$\sum_{i \in S_{\text{OPT}}} v_i' \geq \text{OPT} \, \frac{n}{\varepsilon v_{\max}} - n$$

How does $\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i$ relate to $\text{OPT} = \sum_{i \in S_{\text{OPT}}} v_i$?

$$\begin{aligned}
\text{ALG} = \sum_{i \in S_{\text{ALG}}} v_i &= \sum_{i \in S_{\text{ALG}}} v_i \frac{n}{\varepsilon v_{\max}} \frac{\varepsilon v_{\max}}{n} \\
&\geq \sum_{i \in S_{\text{ALG}}} \left\lfloor v_i \frac{n}{\varepsilon v_{\max}} \right\rfloor \frac{\varepsilon v_{\max}}{n} = \sum_{i \in S_{\text{ALG}}} v_i' \frac{\varepsilon v_{\max}}{n} \\
&\geq \sum_{i \in S_{\text{OPT}}} v_i' \frac{\varepsilon v_{\max}}{n} \geq \left(\text{OPT} \, \frac{n}{\varepsilon v_{\max}} - n\right) \frac{\varepsilon v_{\max}}{n} = \text{OPT} - \varepsilon v_{\max} \\
&\geq \text{OPT} - \varepsilon \, \text{OPT}, \text{ because } \text{OPT} \geq v_{\max} \text{ (discard overweight items)} \\
&= (1 - \varepsilon) \, \text{OPT}
\end{aligned}$$

So, $\text{ALG} \geq (1 - \varepsilon) \, \text{OPT}$

Knapsack

Knapsack: Given a set of n items with values $v_1, \ldots, v_n$ and weights $w_1, \ldots, w_n$, select the most valuable combination with total weight $\leq W$.

Performance Guarantee: ALG $\geq (1 - \varepsilon)$ OPT

Running Time: $O\left(\frac{n^3}{\varepsilon}\right)$

Knapsack

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Performance Guarantee: ALG $\geq (1 - \varepsilon)$ OPT

Running Time: $O\left(\frac{n^3}{\varepsilon}\right)$

We can solve Knapsack instances arbitrarily close to optimal in polynomial time!!

Knapsack

Knapsack: Given a set of n items with values $v_1, \dots, v_n$ and weights $w_1, \dots, w_n$, select the most valuable combination with total weight $\leq W$.

Performance Guarantee: ALG $\geq (1 - \varepsilon)$ OPT

Running Time: $O\left(\frac{n^3}{\varepsilon}\right)$

Fully Polynomial-Time Approximation Scheme (FPTAS)

We can solve Knapsack instances arbitrarily close to optimal in polynomial time!!

Approximability Hierarchy

<u>PTAS</u>: Running time polynomial in input size.

<u>FPTAS</u>: Running time polynomial in input size and ε.