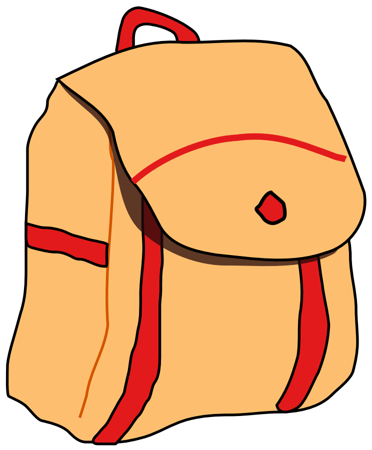
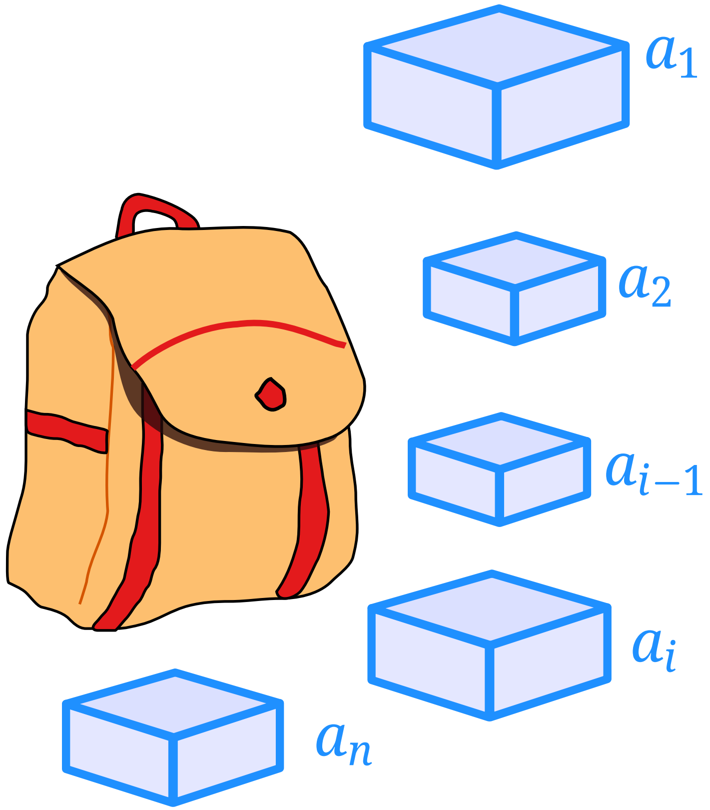
Approximation Schemes and the KNAPSACK Problem



KNAPSACK

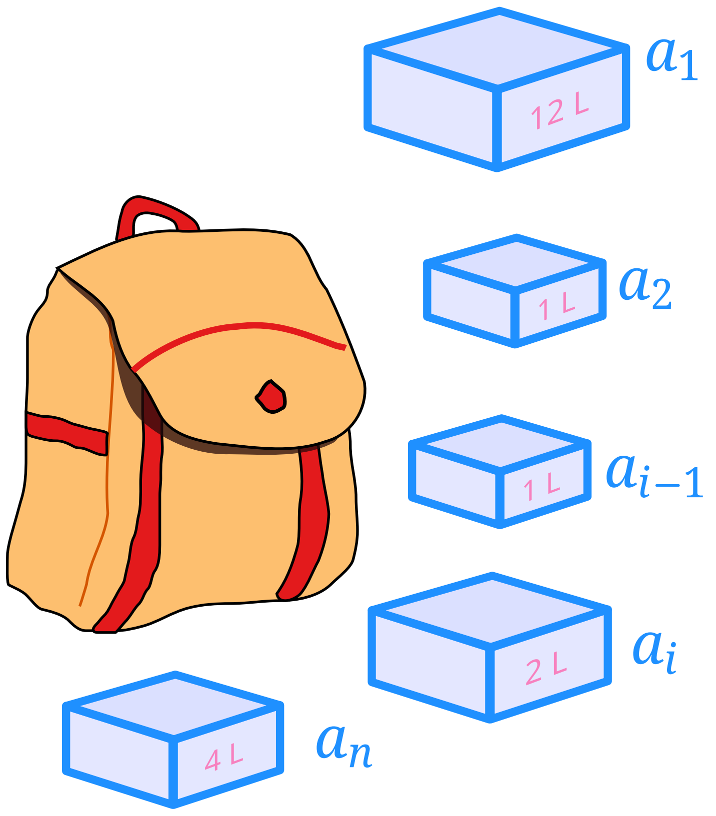
Given: ■ A set $S = \{a_1, \ldots, a_n\}$ of **objects**.



Knapsack

Given:

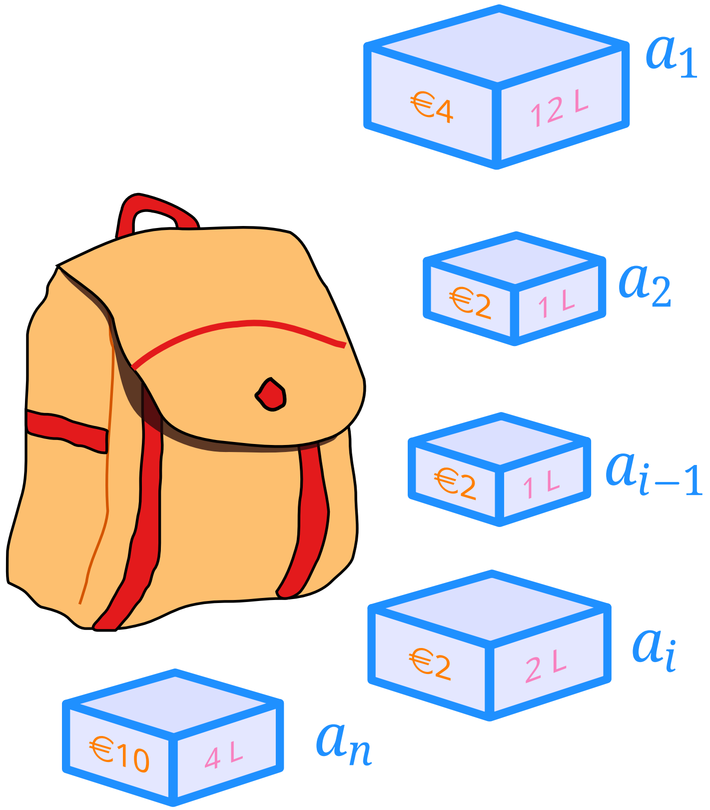
- A set $S = \{a_1, \ldots, a_n\}$ of **objects**.
- For every object a_i a **size** $\mathrm{size}(a_i) \in \mathbb{N}^+$



Knapsack

Given:

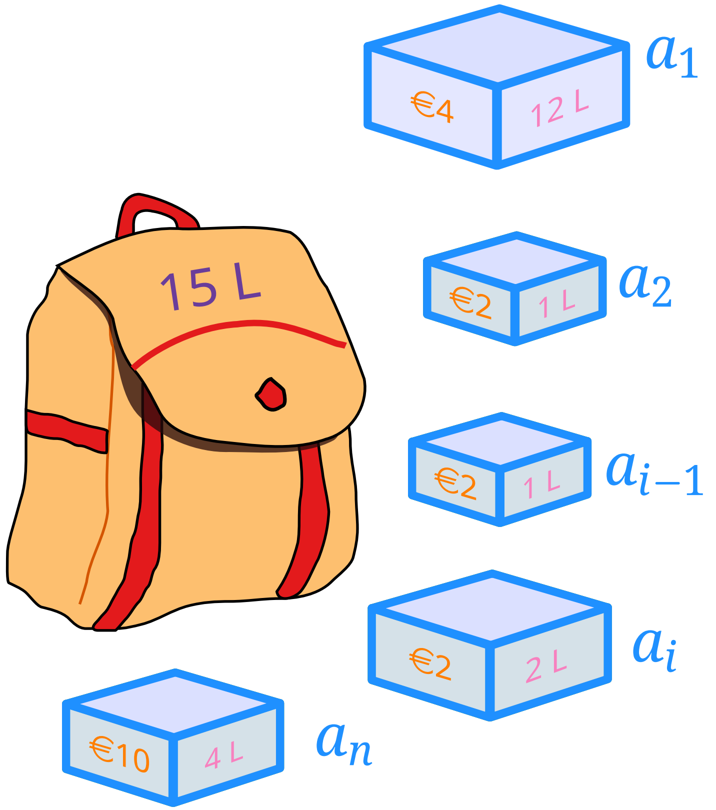
- A set $S = \{a_1, \dots, a_n\}$ of **objects**.
- For every object a_i a **size** $\mathrm{size}(a_i) \in \mathbb{N}^+$
- For every object a_i a **profit** $\mathrm{profit}(a_i) \in \mathbb{N}^+$



Knapsack

Given:

- A set $S = \{a_1, \ldots, a_n\}$ of **objects**.
- For every object a_i a **size** $\text{size}(a_i) \in \mathbb{N}^+$
- For every object a_i a **profit** $\text{profit}(a_i) \in \mathbb{N}^+$
- A knapsack **capacity** $B \in \mathbb{N}^+$

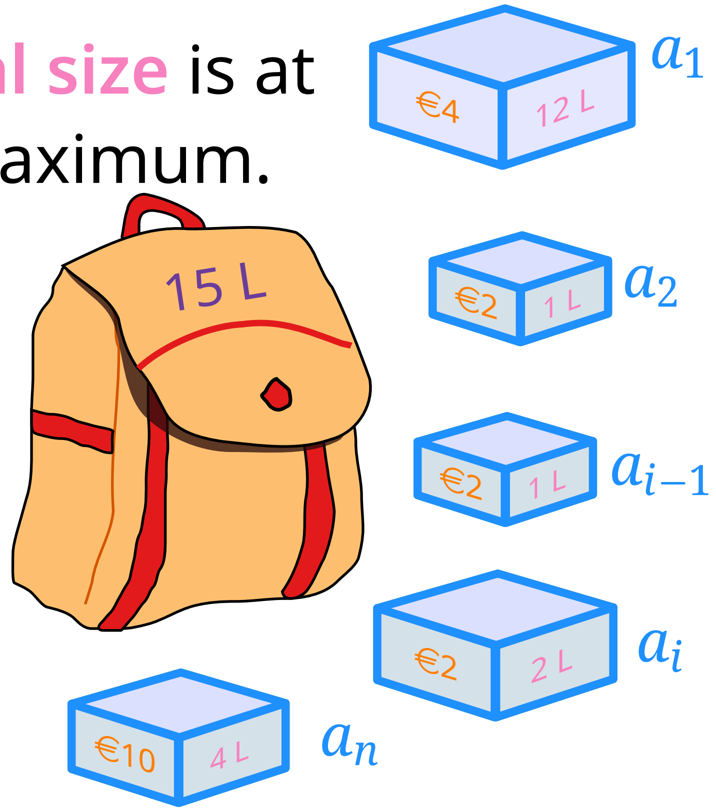


Knapsack

Given:

- A set $S = \{a_1, \ldots, a_n\}$ of **objects**.
- For every object a_i a **size** $\mathrm{size}(a_i) \in \mathbb{N}^+$
- For every object a_i a **profit** $\mathrm{profit}(a_i) \in \mathbb{N}^+$
- A knapsack **capacity** $B \in \mathbb{N}^+$

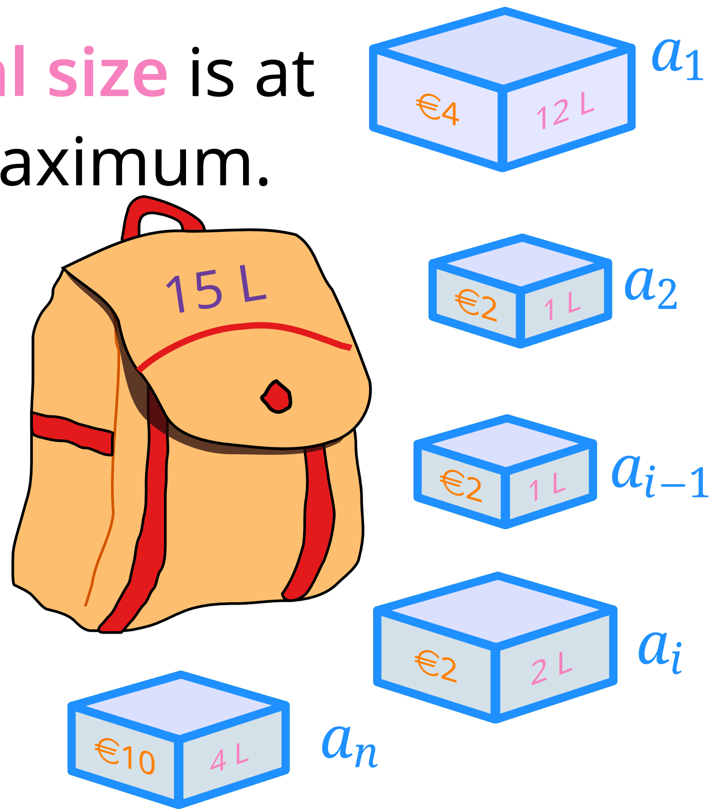
Task: Find a subset of objects whose **total size** is at most B and whose **total profit** is maximum.



Knapsack

Given:
- A set $S = \{a_1, \ldots, a_n\}$ of **objects**.
- For every object a_i a **size** $\mathrm{size}(a_i) \in \mathbb{N}^+$
- For every object a_i a **profit** $\mathrm{profit}(a_i) \in \mathbb{N}^+$
- A knapsack **capacity** $B \in \mathbb{N}^+$

Task: Find a subset of objects whose **total size** is at most B and whose **total profit** is maximum.



Which subset is optimal in the example?

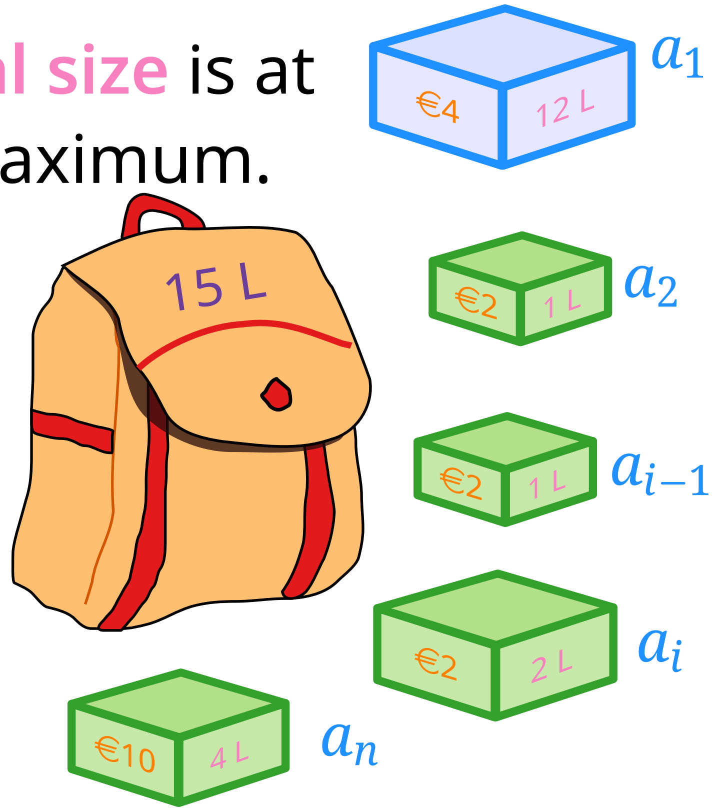
Knapsack

Given:

- A set $S = \{a_1, \ldots, a_n\}$ of **objects**.
- For every object a_i a **size** $\mathrm{size}(a_i) \in \mathbb{N}^+$
- For every object a_i a **profit** $\mathrm{profit}(a_i) \in \mathbb{N}^+$
- A knapsack **capacity** $B \in \mathbb{N}^+$

Task: Find a subset of objects whose **total size** is at most B and whose **total profit** is maximum.

Which subset is optimal in the example?



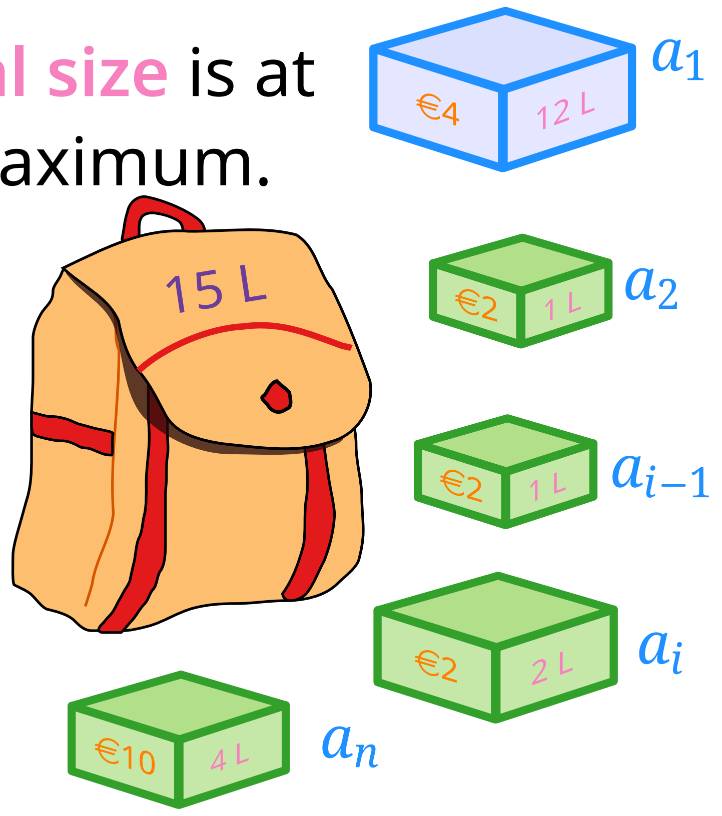
Knapsack

Given:

- A set $S = \{a_1, \ldots, a_n\}$ of **objects**.
- For every object a_i a **size** $\mathrm{size}(a_i) \in \mathbb{N}^+$
- For every object a_i a **profit** $\mathrm{profit}(a_i) \in \mathbb{N}^+$
- A knapsack **capacity** $B \in \mathbb{N}^+$

Task: Find a subset of objects whose **total size** is at most B and whose **total profit** is maximum.

NP-hard

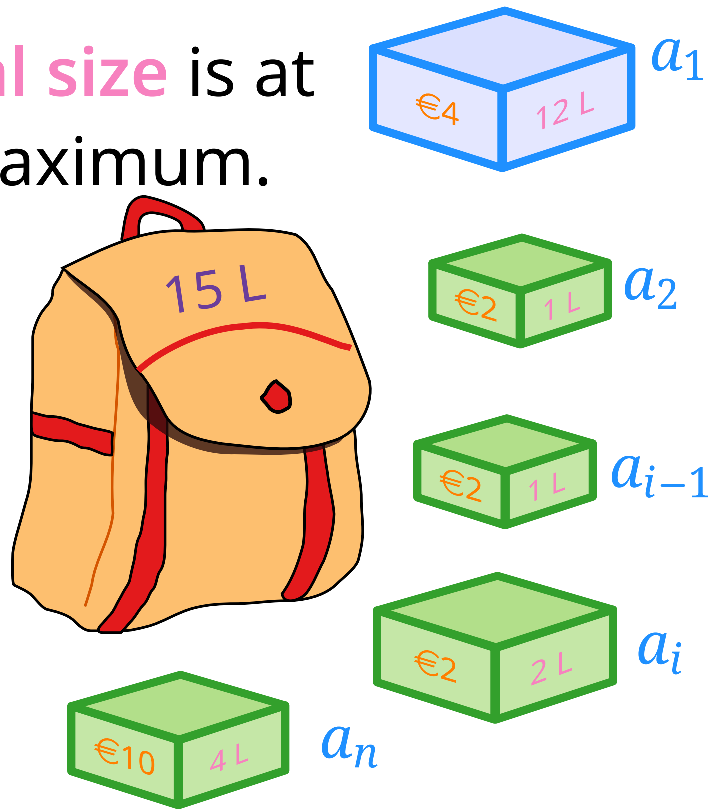


KNAPSACK

Given:

- A set $S = \{a_1, \ldots, a_n\}$ of **objects**.
- For every object a_i a **size** $\mathrm{size}(a_i) \in \mathbb{N}^+$
- For every object a_i a **profit** $\mathrm{profit}(a_i) \in \mathbb{N}^+$
- A knapsack **capacity** $B \in \mathbb{N}^+$

Task: Find a subset of objects whose **total size** is at most B and whose **total profit** is maximum.



Natural greedy approach:

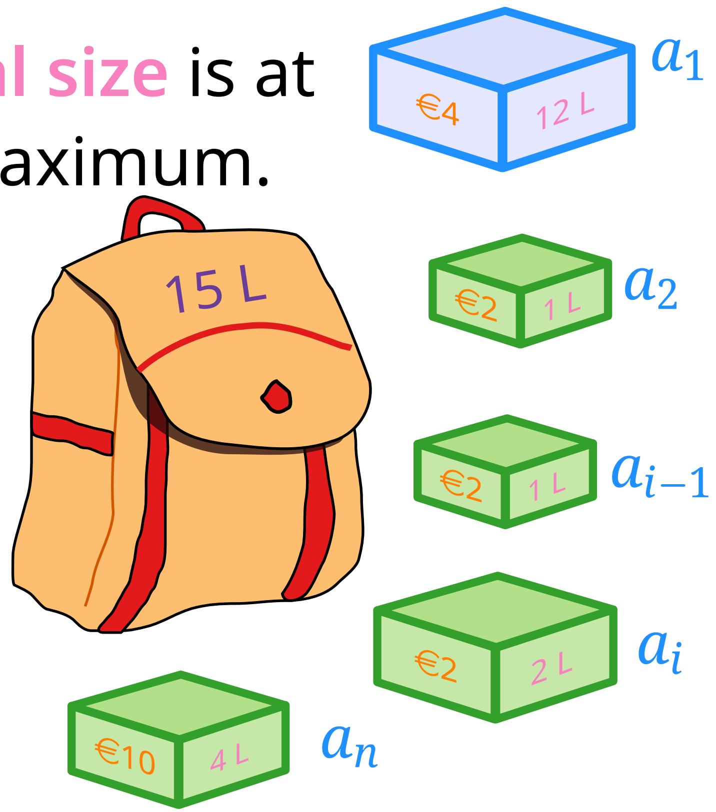
pick elements by decreasing profit/size

Knapsack

Given:

- A set $S = \{a_1, \ldots, a_n\}$ of **objects**.
- For every object a_i a **size** $\mathrm{size}(a_i) \in \mathbb{N}^+$
- For every object a_i a **profit** $\mathrm{profit}(a_i) \in \mathbb{N}^+$
- A knapsack **capacity** $B \in \mathbb{N}^+$

Task: Find a subset of objects whose **total size** is at most B and whose **total profit** is maximum.



Natural greedy approach:

pick elements by decreasing profit/size

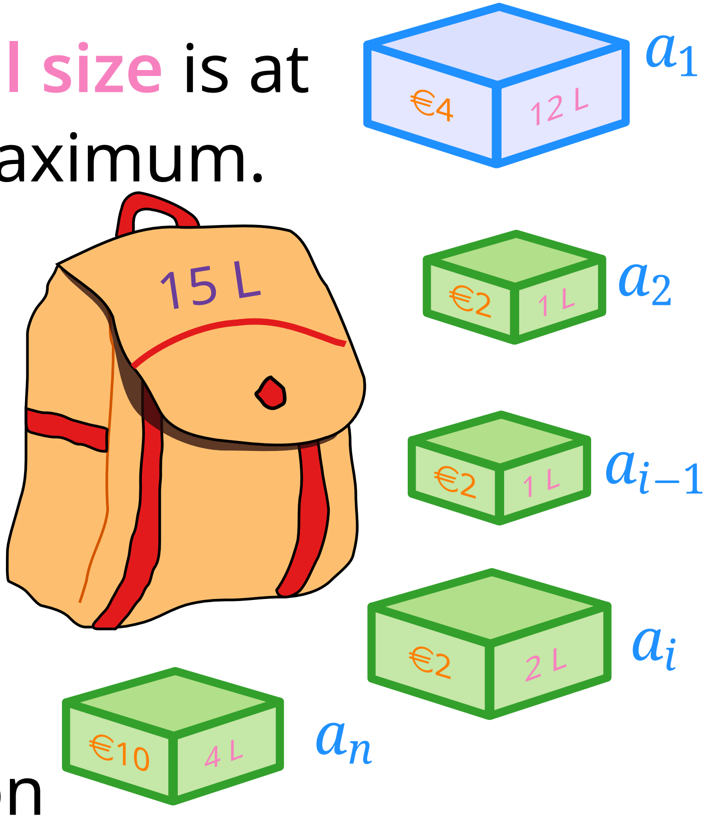
How well does this do?

Knapsack

Given:

- A set $S = \{a_1, \ldots, a_n\}$ of **objects**.
- For every object a_i a **size** $\mathrm{size}(a_i) \in \mathbb{N}^+$
- For every object a_i a **profit** $\mathrm{profit}(a_i) \in \mathbb{N}^+$
- A knapsack **capacity** $B \in \mathbb{N}^+$

Task: Find a subset of objects whose **total size** is at most B and whose **total profit** is maximum.



Natural greedy approach:

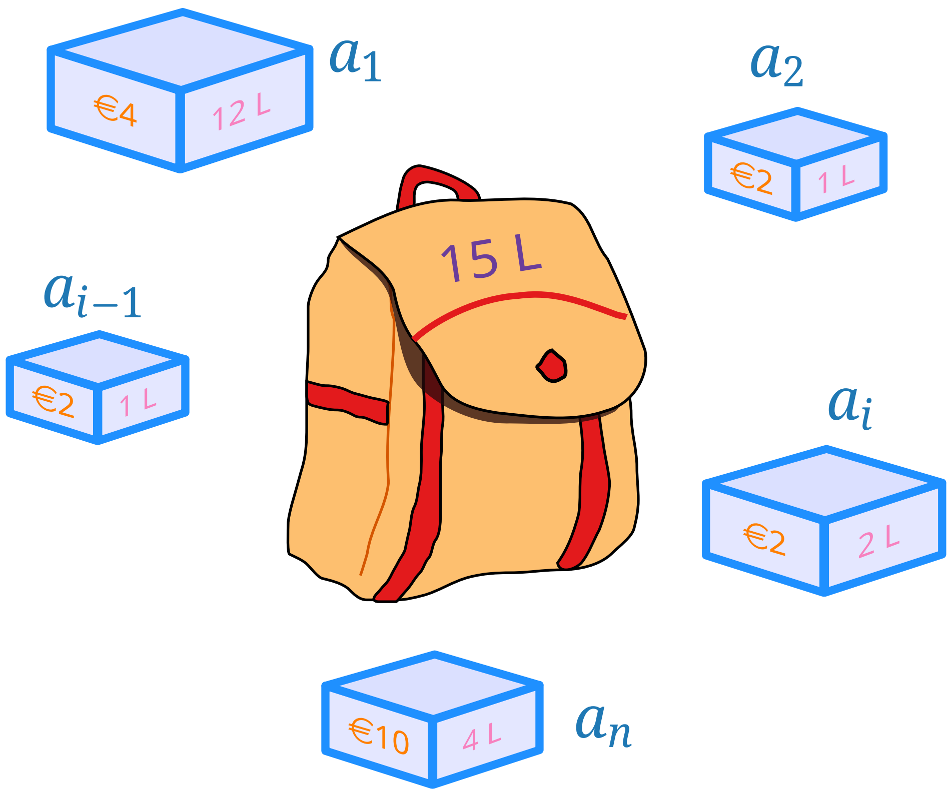
pick elements by decreasing profit/size

How well does this do?

arbitrarily bad!
but picking the max of this and the first element not picked gives a 2-approximation

Pseudo-Polynomial Algorithms

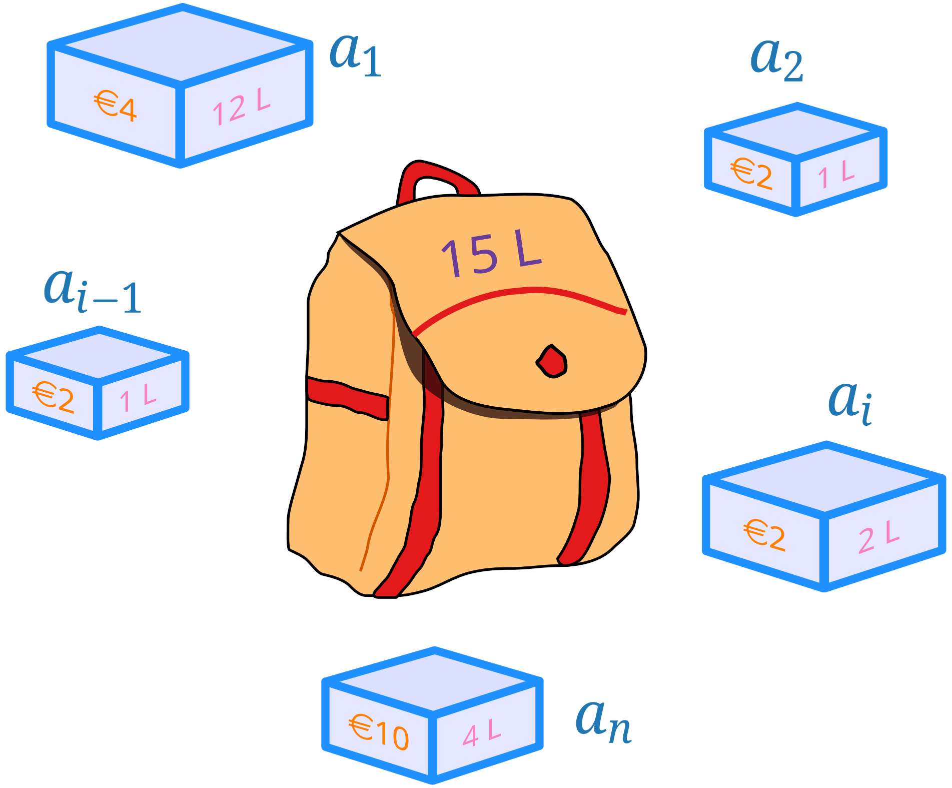
Let Π be an optimization problem whose instances can be represented by **objects** (such as sets, elements, edges, nodes) and **numbers** (such as costs, weights, profits).



Pseudo-Polynomial Algorithms

Let Π be an optimization problem whose instances can be represented by **objects** (such as sets, elements, edges, nodes) and **numbers** (such as costs, weights, profits).

$|I|$: The size of an instance $I \in D_\Pi$, where all numbers in I are encoded in binary.

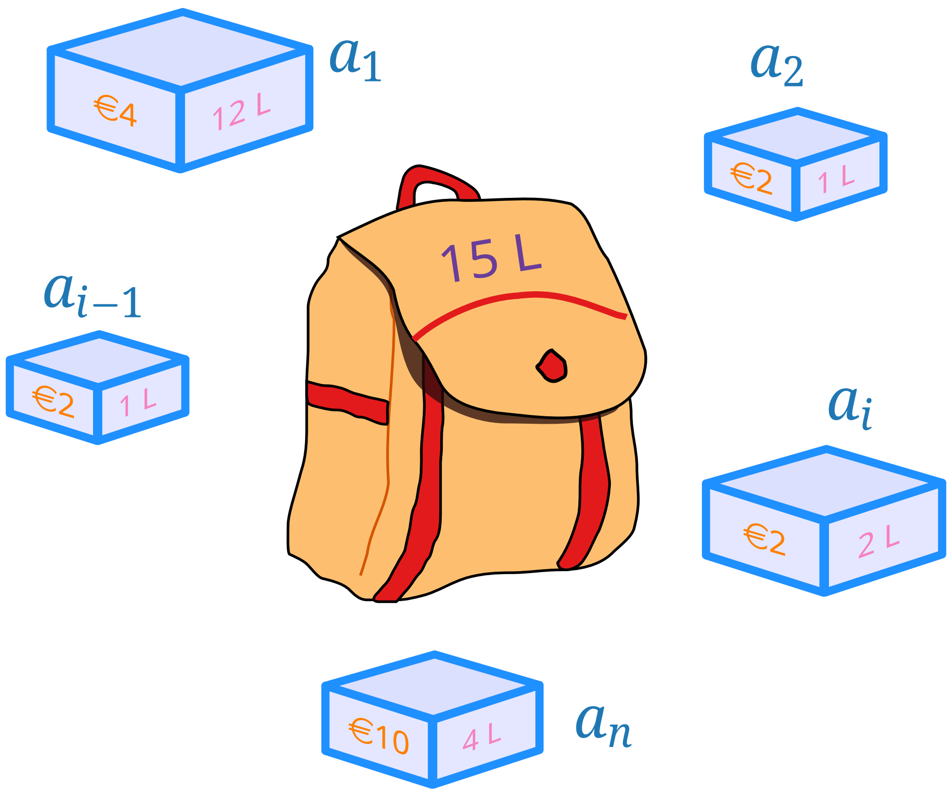


Pseudo-Polynomial Algorithms

Let Π be an optimization problem whose instances can be represented by **objects** (such as sets, elements, edges, nodes) and **numbers** (such as costs, weights, profits).

$|I|$: The size of an instance $I \in D_\Pi$, where all numbers in I are encoded in binary.

$$(5 \triangleq 101_b \Rightarrow |I| = 3 + \ldots)$$



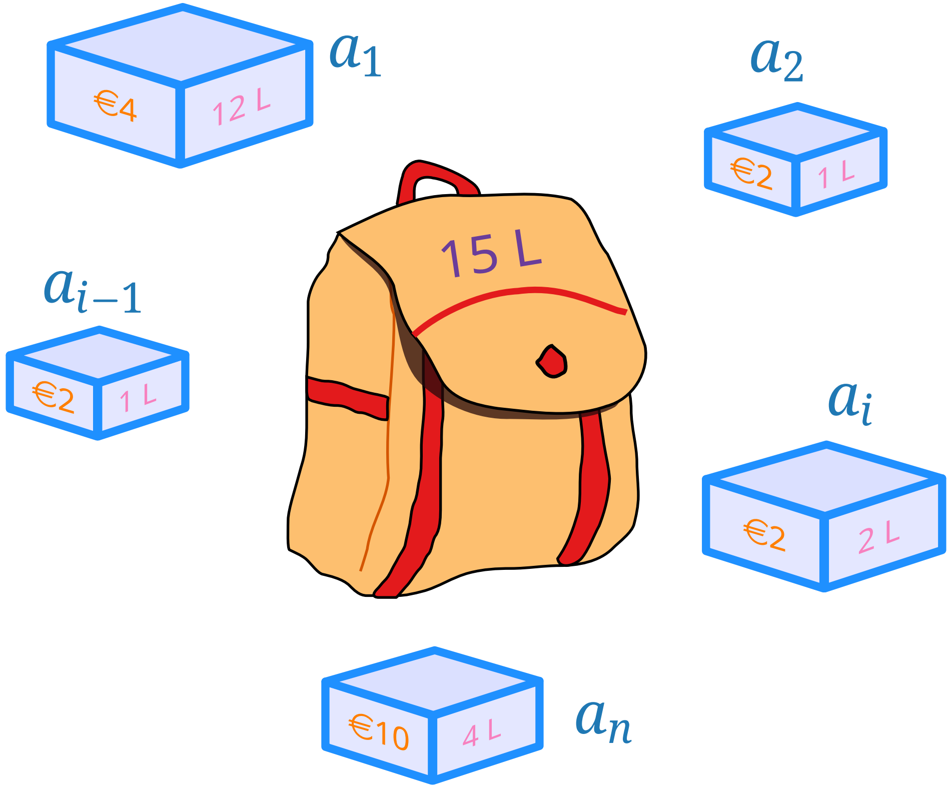
Pseudo-Polynomial Algorithms

Let Π be an optimization problem whose instances can be represented by **objects** (such as sets, elements, edges, nodes) and **numbers** (such as costs, weights, profits).

$|I|$: The size of an instance $I \in D_\Pi$, where all numbers in I are encoded in binary.

$$\left(5 \triangleq 101_b \Rightarrow |I| = 3 + \ldots\right)$$

$|I|_u$: The size of an instance $I \in D_\Pi$, where all numbers in I are encoded in unary.



Pseudo-Polynomial Algorithms

Let Π be an optimization problem whose instances can be represented by **objects** (such as sets, elements, edges, nodes) and **numbers** (such as costs, weights, profits).

$|I|$: The size of an instance $I \in D_\Pi$, where all numbers in I are encoded in binary.

$$\left(5 \triangleq 101_{\mathrm{b}} \Rightarrow |I| = 3 + \ldots\right)$$

$|I|_{\mathrm{u}}$: The size of an instance $I \in D_\Pi$, where all numbers in I are encoded in unary.

$$\left(5 \triangleq 11111_{\mathrm{u}} \Rightarrow |I|_{\mathrm{u}} = 5 + \ldots\right)$$

Pseudo-Polynomial Algorithms

Let Π be an optimization problem whose instances can be represented by **objects** (such as sets, elements, edges, nodes) and **numbers** (such as costs, weights, profits).

$|I|$: The size of an instance $I \in D_\Pi$, where all numbers in I are encoded in binary.

$$(5 \triangleq 101_\mathrm{b} \Rightarrow |I| = 3 + \ldots)$$

$|I|_\mathrm{u}$: The size of an instance $I \in D_\Pi$, where all numbers in I are encoded in unary.

$$(5 \triangleq 11111_\mathrm{u} \Rightarrow |I|_\mathrm{u} = 5 + \ldots)$$

The running time of a polynomial(-time) algorithm for Π is polynomial in $|I|$.

Pseudo-Polynomial Algorithms

Let Π be an optimization problem whose instances can be represented by **objects** (such as sets, elements, edges, nodes) and **numbers** (such as costs, weights, profits).

$|I|$: The size of an instance $I \in D_\Pi$, where all numbers in I are encoded in binary.

$$(5 \triangleq 101_\mathrm{b} \Rightarrow |I| = 3 + \ldots)$$

$|I|_\mathrm{u}$: The size of an instance $I \in D_\Pi$, where all numbers in I are encoded in unary.

$$(5 \triangleq 11111_\mathrm{u} \Rightarrow |I|_\mathrm{u} = 5 + \ldots)$$

The running time of a polynomial(-time) algorithm for Π is polynomial in $|I|$.

The running time of a pseudo-polynomial algorithm is polynomial in $|I|_\mathrm{u}$.

Pseudo-Polynomial Algorithms

Let Π be an optimization problem whose instances can be represented by **objects** (such as sets, elements, edges, nodes) and **numbers** (such as costs, weights, profits).

$|I|$: The size of an instance $I \in D_\Pi$, where all numbers in I are encoded in binary.

$$(5 \triangleq 101_\mathrm{b} \Rightarrow |I| = 3 + \ldots)$$

$|I|_\mathrm{u}$: The size of an instance $I \in D_\Pi$, where all numbers in I are encoded in unary.

$$(5 \triangleq 11111_\mathrm{u} \Rightarrow |I|_\mathrm{u} = 5 + \ldots)$$

The running time of a polynomial(-time) algorithm for Π is polynomial in $|I|$.

The running time of a pseudo-polynomial algorithm is polynomial in $|I|_\mathrm{u}$.

The running time of a pseudo-polynomial algorithm may not be polynomial in $|I|$.

Strong NP-Hardness

An optimization problem is called strongly NP-hard if it remains NP-hard under unary encoding.

Strong NP-Hardness

An optimization problem is called strongly NP-hard if it remains NP-hard under unary encoding.

An optimization problem is called weakly NP-hard if it is NP-hard under binary encoding (but may have a pseudo-polynomial algorithm).

Strong NP-Hardness

An optimization problem is called strongly NP-hard if it remains NP-hard under unary encoding.

An optimization problem is called weakly NP-hard if it is NP-hard under binary encoding (but may have a pseudo-polynomial algorithm).

Theorem. A strongly NP-hard problem has no pseudo-polynomial algorithm unless $\mathrm{P} = \mathrm{NP}$.

Strong NP-Hardness

An optimization problem is called strongly NP-hard if it remains NP-hard under unary encoding.

An optimization problem is called weakly NP-hard if it is NP-hard under binary encoding (but may have a pseudo-polynomial algorithm).

Theorem. A strongly NP-hard problem has no pseudo-polynomial algorithm unless $\mathrm{P} = \mathrm{NP}$.

Examples:

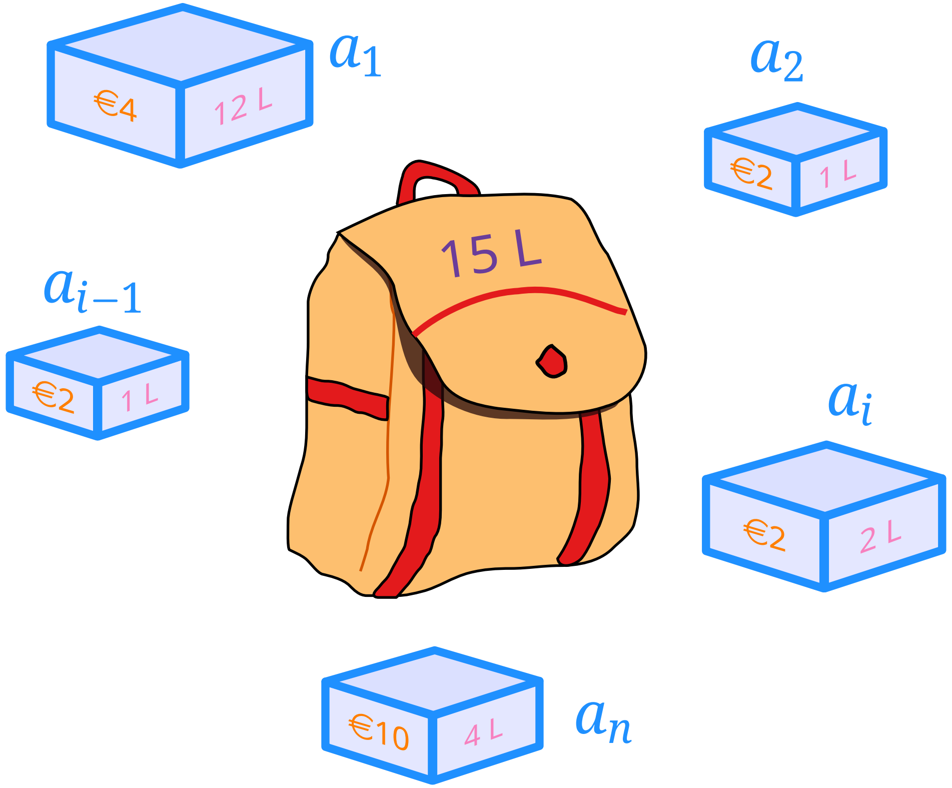
any NP-hard problem without numeric input, such as SAT, Hamilton circuit, ...

3-partition, bin packing, ...

Pseudo-Polynomial Algorithm for KNAPSACK

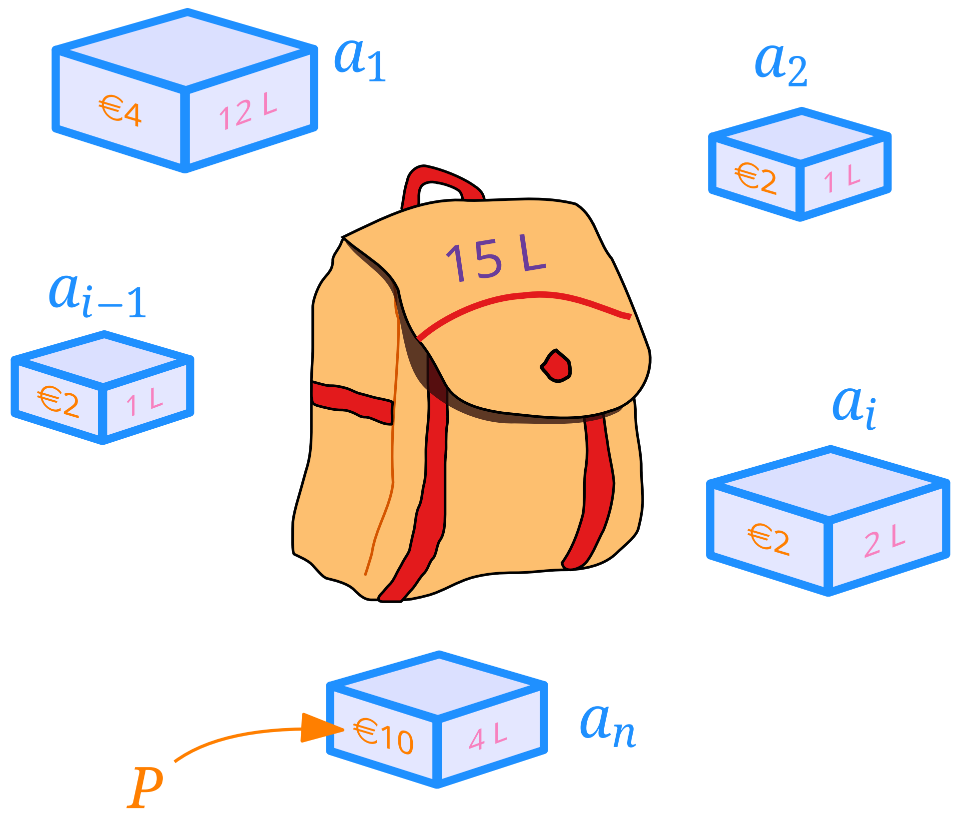
Pseudo-Polynomial Alg. for KNAPSACK

Let $P := \max_i \text{profit}(a_i)$



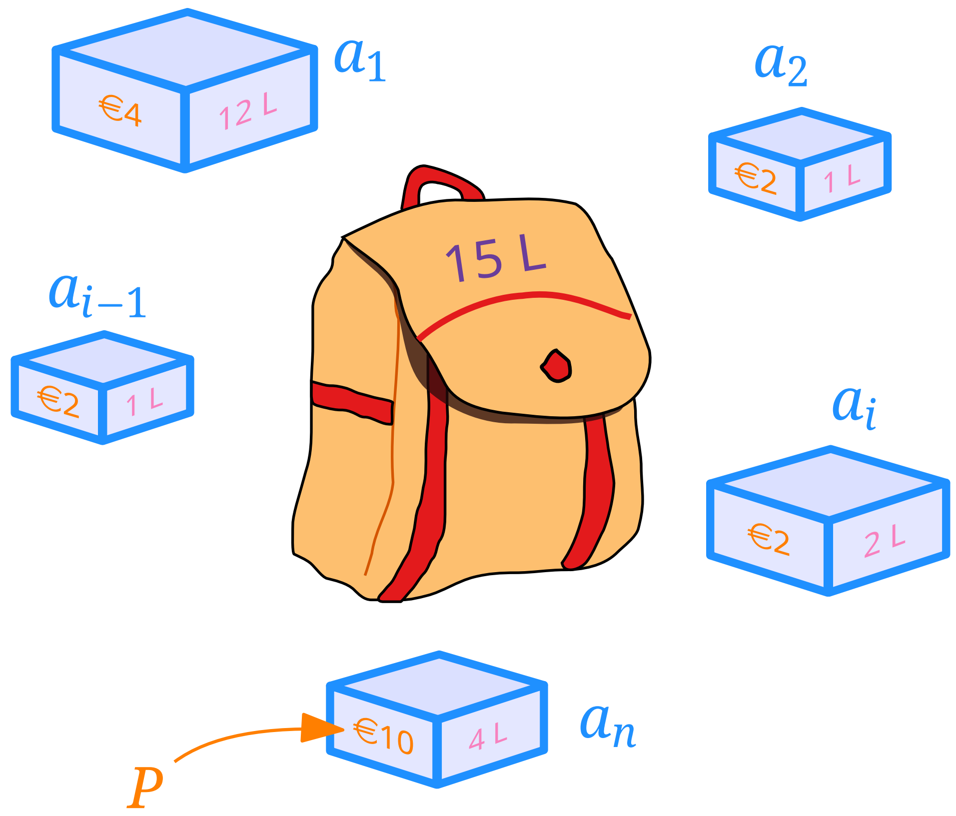
Pseudo-Polynomial Alg. for KNAPSACK

Let $P := \max_i \text{profit}(a_i)$



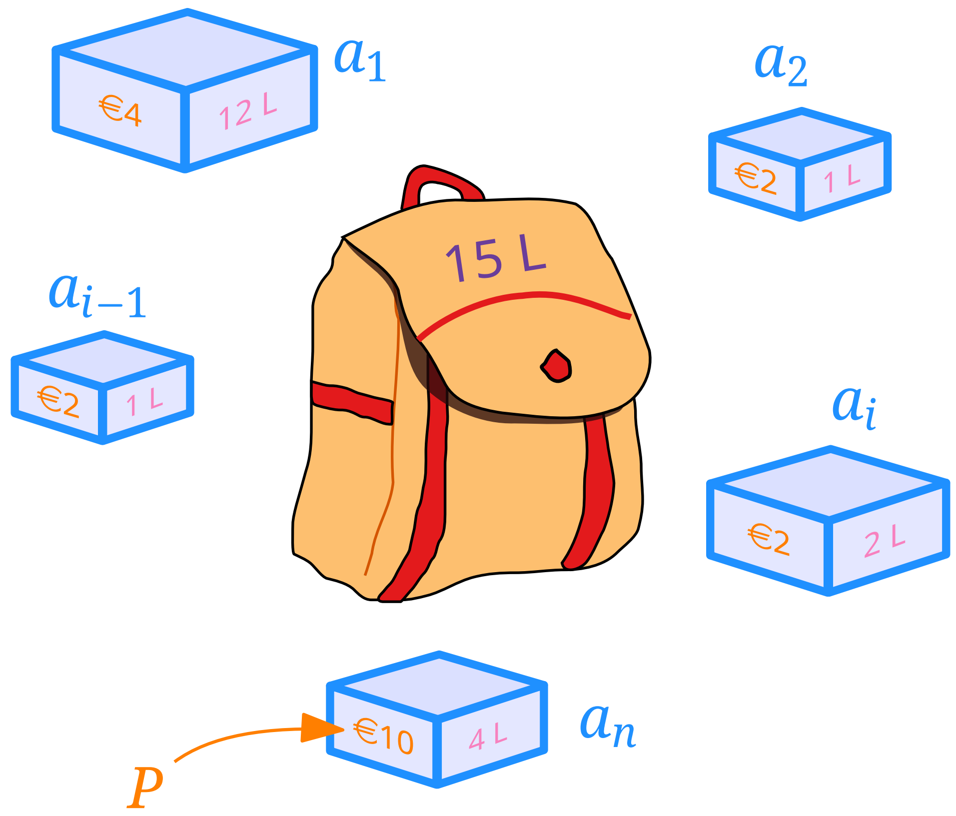
Pseudo-Polynomial Alg. for KNAPSACK

Let $P := \max_i \mathrm{profit}(a_i) \quad \Rightarrow \qquad \leq \mathrm{OPT} \leq$



Pseudo-Polynomial Alg. for KNAPSACK

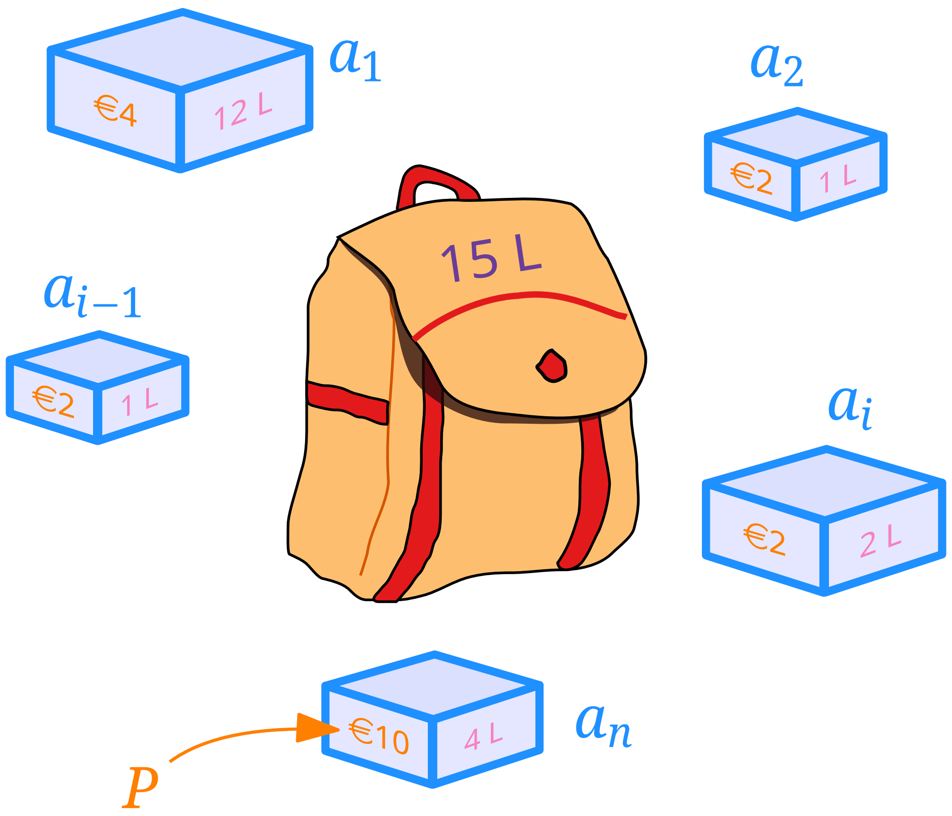
Let $P := \max_i \mathrm{profit}(a_i) \quad \Rightarrow \quad P \le \mathrm{OPT} \le nP$



Pseudo-Polynomial Alg. for Knapsack

Let $P := \max_i \mathrm{profit}(a_i) \quad \Rightarrow \quad P \leq \mathrm{OPT} \leq nP$

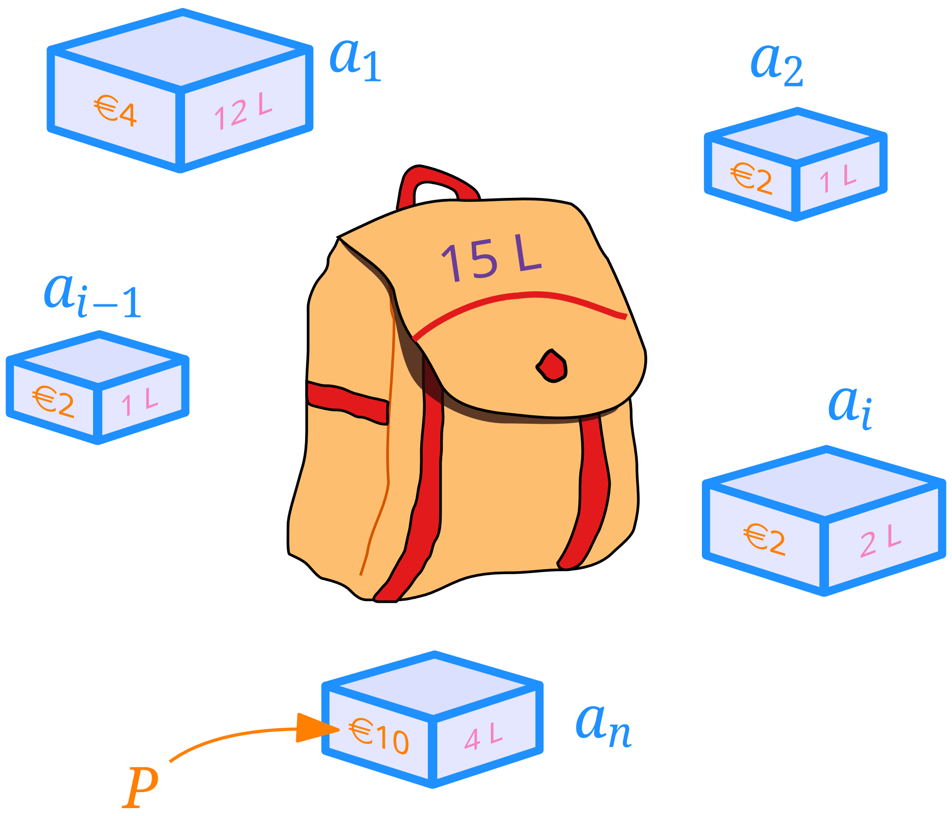
For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$,



Pseudo-Polynomial Alg. for KNAPSACK

Let $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

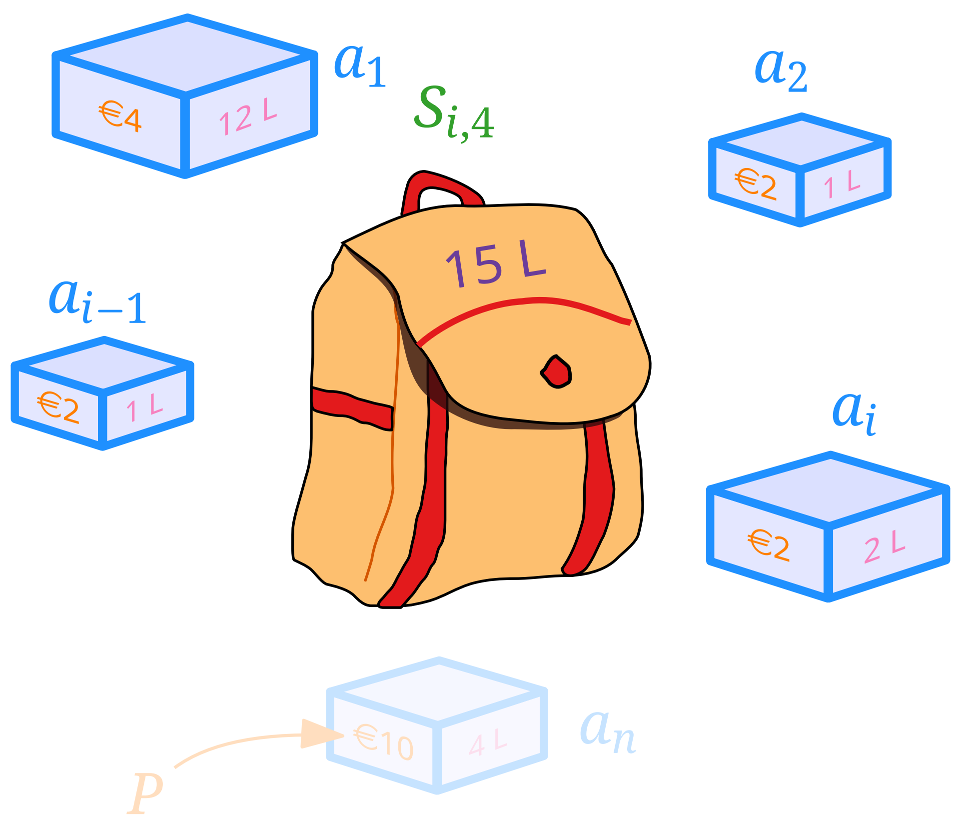
For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$,
let $S_{i,p}$ be a subset of $\{a_1, \ldots, a_i\}$ whose total profit is precisely p



Pseudo-Polynomial Alg. for KNAPSACK

Let $P := \max_i \mathrm{profit}(a_i) \quad \Rightarrow \quad P \le \mathrm{OPT} \le nP$

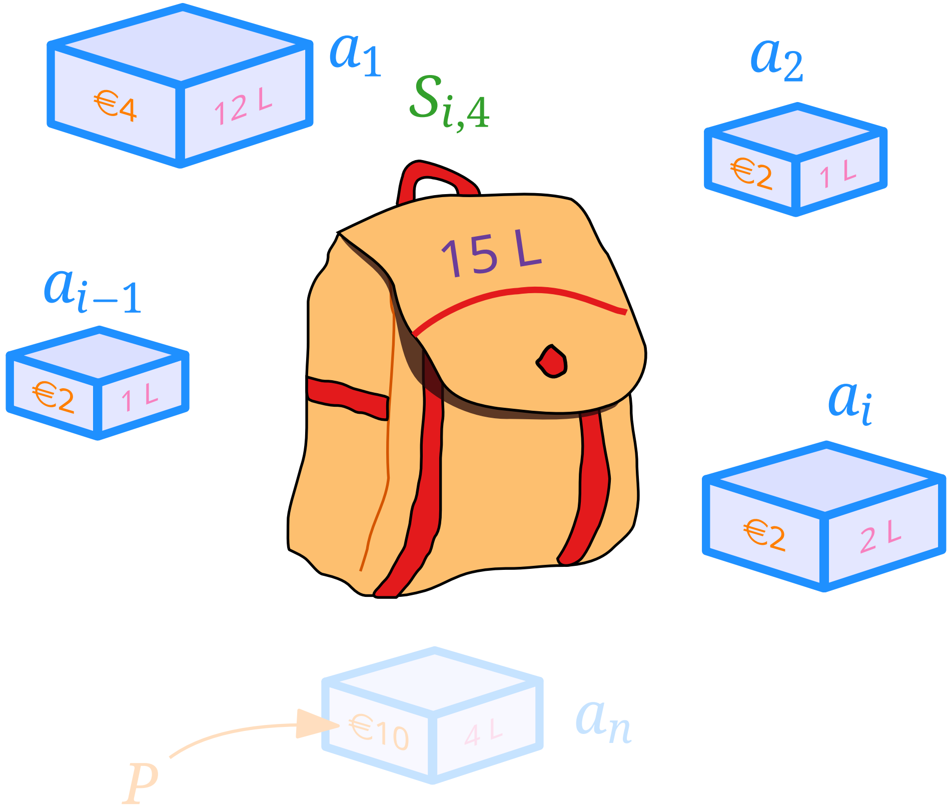
For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$,
let $S_{i,p}$ be a subset of $\{a_1, \ldots, a_i\}$ whose total profit is precisely p



Pseudo-Polynomial Alg. for KNAPSACK

Let $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

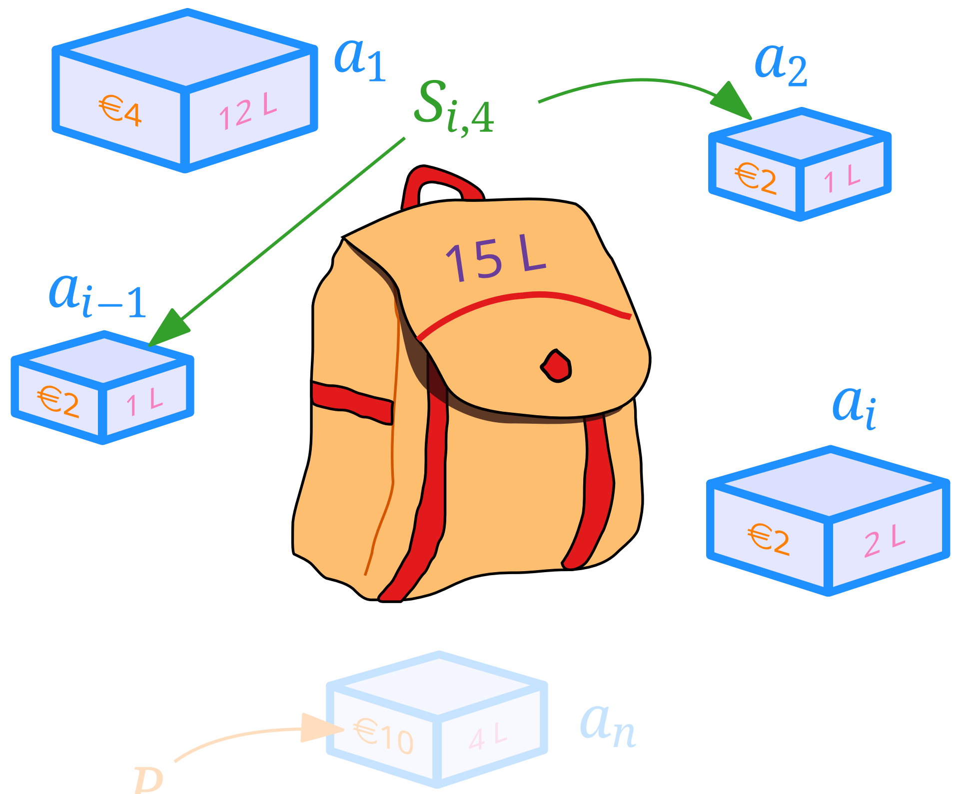
For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$,
let $S_{i,p}$ be a subset of $\{a_1, \ldots, a_i\}$ whose total profit is precisely p and whose total size is minimum among all subsets with these properties.



Pseudo-Polynomial Alg. for KNAPSACK

Let $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

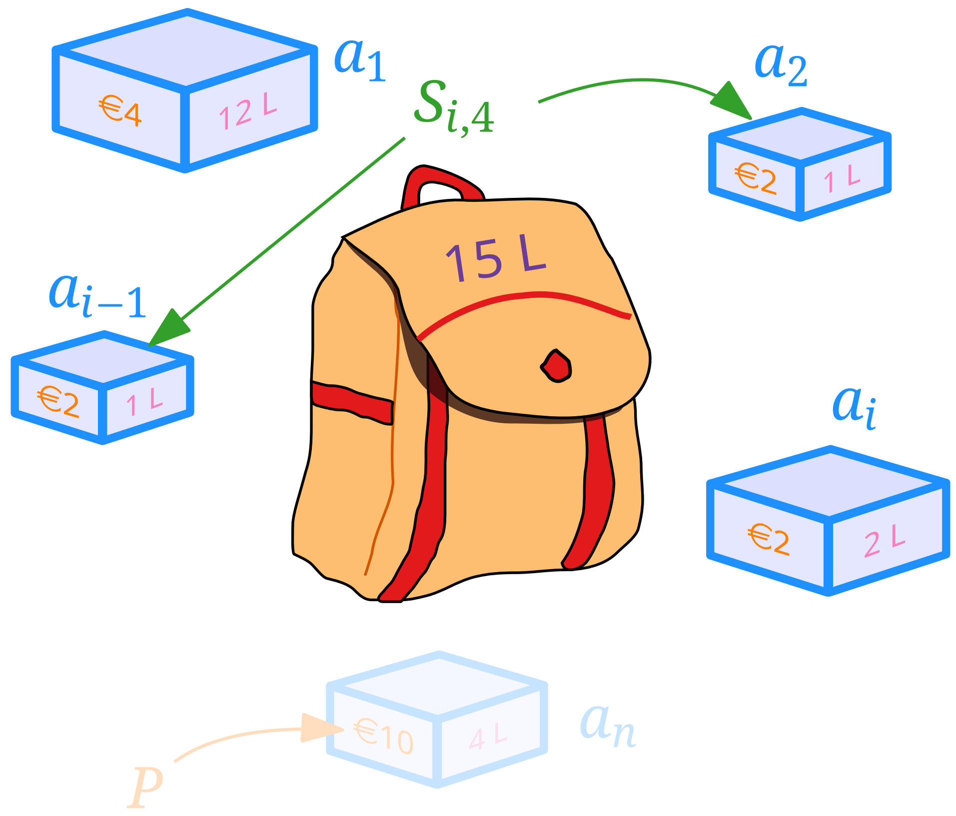
For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$,
let $S_{i,p}$ be a subset of $\{a_1, \ldots, a_i\}$ whose total profit is precisely p and whose total size is minimum among all subsets with these properties.



Pseudo-Polynomial Alg. for Knapsack

Let $P := \max_i \mathrm{profit}(a_i) \quad \Rightarrow \quad P \leq \mathrm{OPT} \leq nP$

For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$,
let $S_{i,p}$ be a subset of $\{a_1, \ldots, a_i\}$ whose total profit is precisely p and whose total size is minimum among all subsets with these properties. Such a set may not exist.

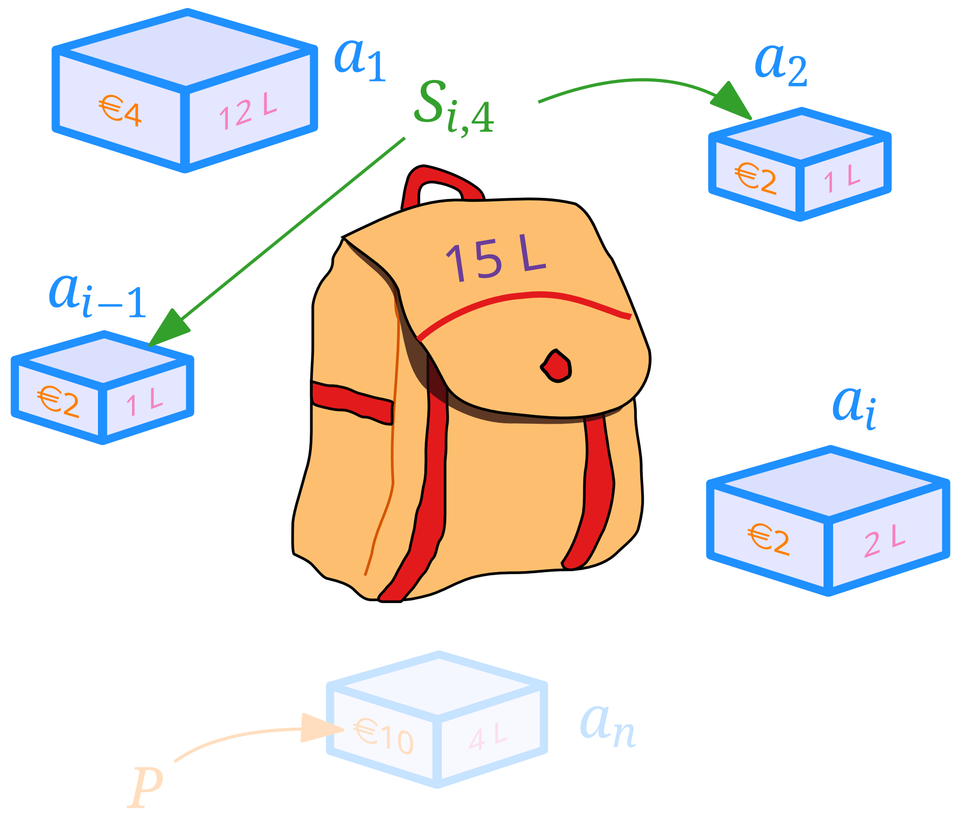


Pseudo-Polynomial Alg. for KNAPSACK

Let $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$,
let $S_{i,p}$ be a subset of $\{a_1, \ldots, a_i\}$ whose total profit is precisely p and whose total size is minimum among all subsets with these properties. Such a set may not exist.

Let $A[i,p]$ be the total size of $S_{i,p}$
(set $A[i,p] = \infty$ if no such set exists).

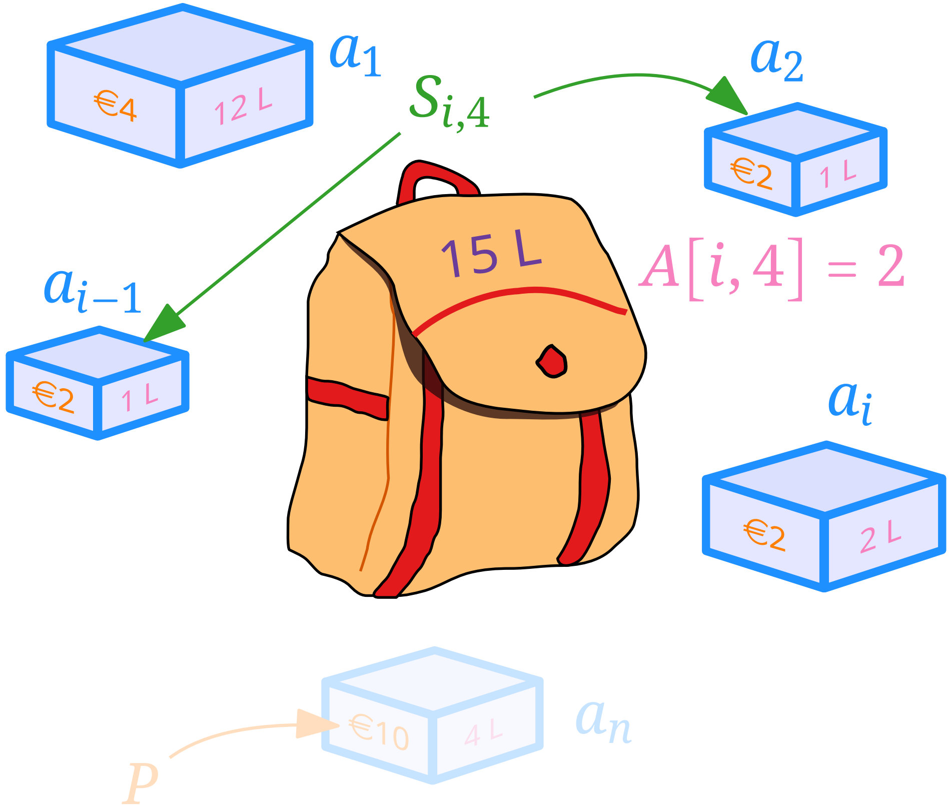


Pseudo-Polynomial Alg. for KNAPSACK

Let $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$,
let $S_{i,p}$ be a subset of $\{a_1, \ldots, a_i\}$ whose total profit is precisely p and whose total size is minimum among all subsets with these properties. Such a set may not exist.

Let $A[i, p]$ be the total size of $S_{i,p}$ (set $A[i, p] = \infty$ if no such set exists).

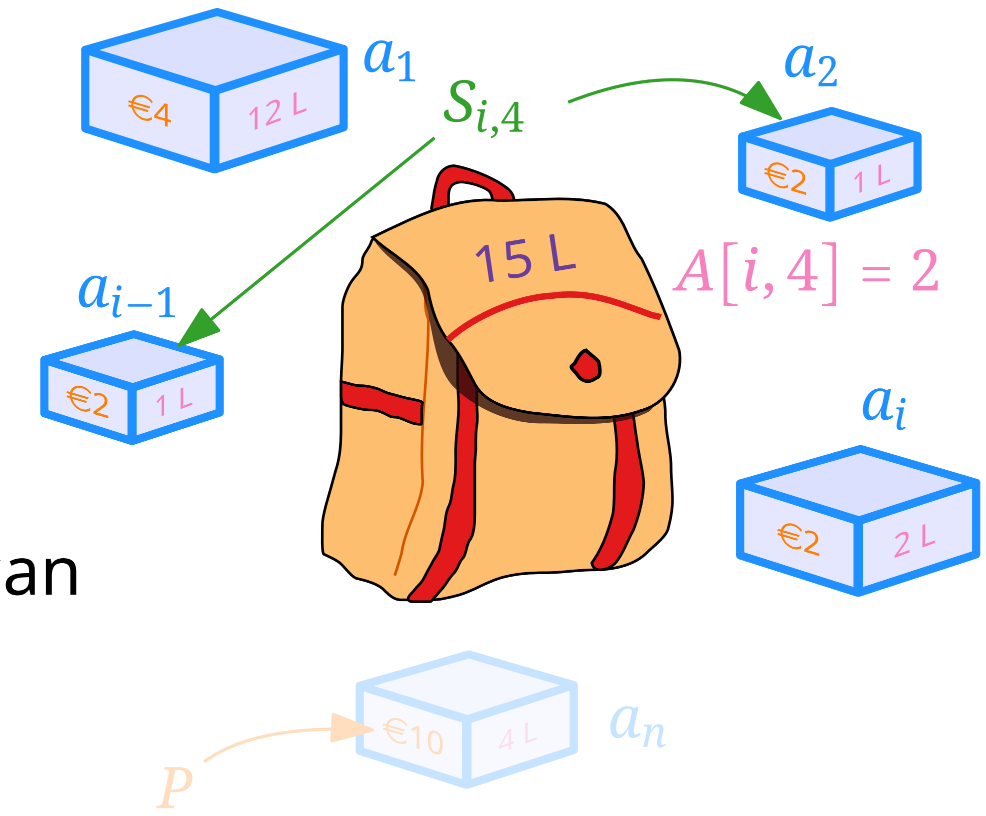


Pseudo-Polynomial Alg. for KNAPSACK

Let $P := \max_i \text{profit}(a_i) \quad \Rightarrow \quad P \leq \text{OPT} \leq nP$

For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$,
let $S_{i,p}$ be a subset of $\{a_1, \ldots, a_i\}$ whose total profit is precisely p and whose total size is minimum among all subsets with these properties. Such a set may not exist.

Let $A[i, p]$ be the total size of $S_{i,p}$ (set $A[i, p] = \infty$ if no such set exists).

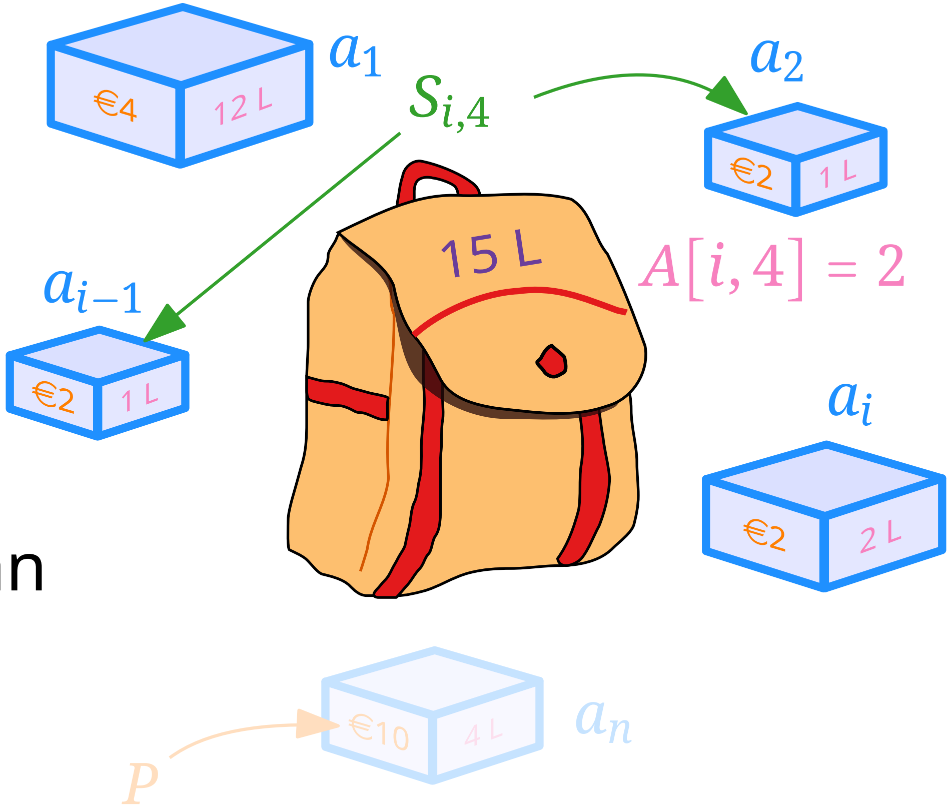


If all $A[i, p]$ are known, then we can compute
$\text{OPT} =$

Pseudo-Polynomial Alg. for KNAPSACK

Let $P := \max_i \operatorname{profit}(a_i) \quad \Rightarrow \quad P \leq \mathrm{OPT} \leq nP$

For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$,
let $S_{i,p}$ be a subset of $\{a_1, \ldots, a_i\}$ whose total profit is precisely p and whose total size is minimum among all subsets with these properties. Such a set may not exist.

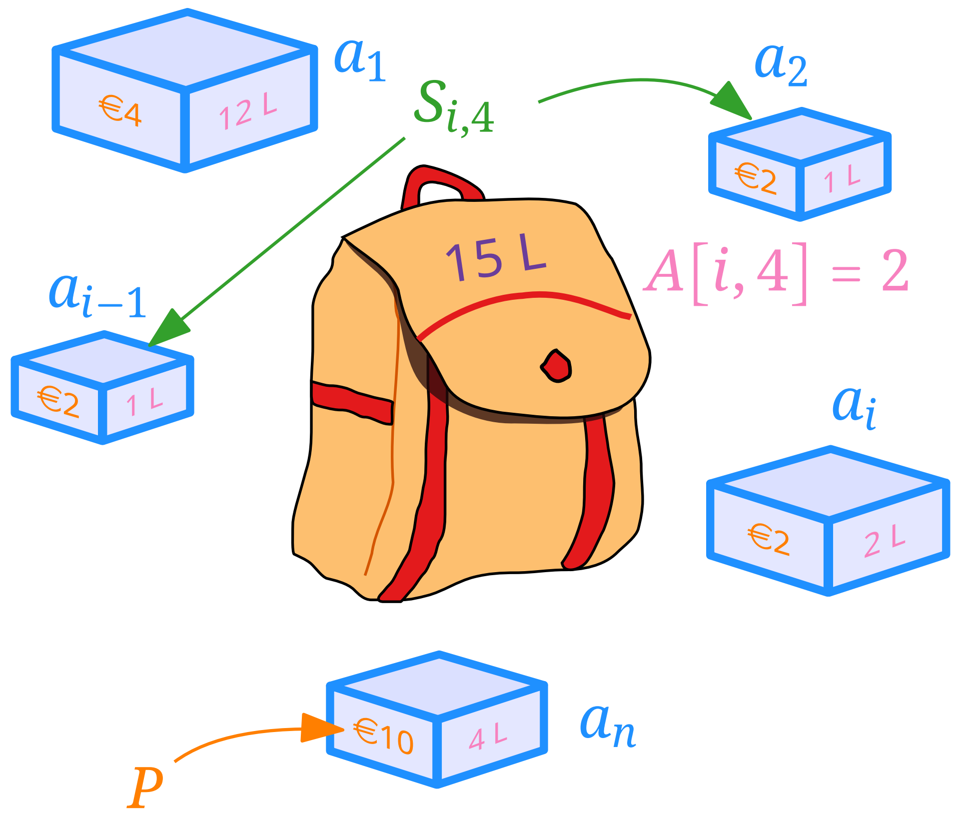


Let $A[i, p]$ be the total size of $S_{i,p}$ (set $A[i, p] = \infty$ if no such set exists).

If all $A[i, p]$ are known, then we can compute
$\mathrm{OPT} = \max\{\, p \mid A[n, p] \leq B \,\}$.

Pseudo-Polynomial Alg. for KNAPSACK

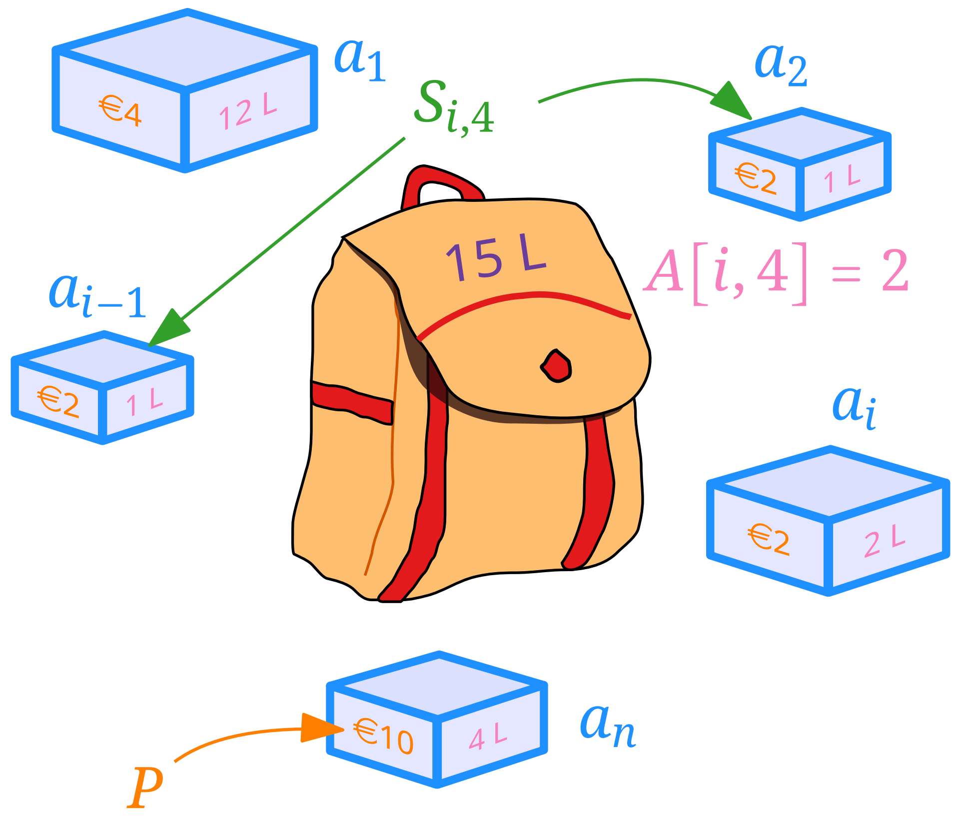
$A[1, p]$ can be computed for all $p \in \{0, \ldots, nP\}$. Note: $A[1, 0] = 0$.



Pseudo-Polynomial Alg. for KNAPSACK

$A[1, p]$ can be computed for all $p \in \{0, \ldots, nP\}$. Note: $A[1, 0] = 0$.

Set $A[i, p] := \infty$ for $p < 0$ (for convenience).

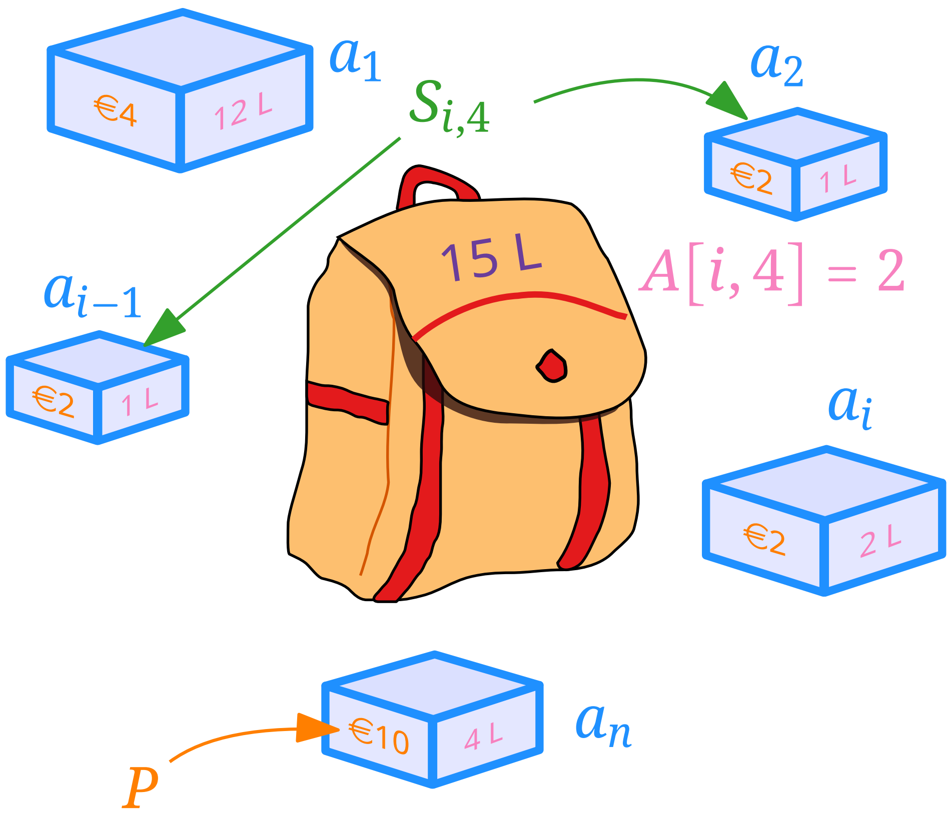


Pseudo-Polynomial Alg. for KNAPSACK

$A[1, p]$ can be computed for all $p \in \{0, \ldots, nP\}$. Note: $A[1, 0] = 0$.

Set $A[i, p] := \infty$ for $p < 0$ (for convenience).

$A[i + 1, p] =$

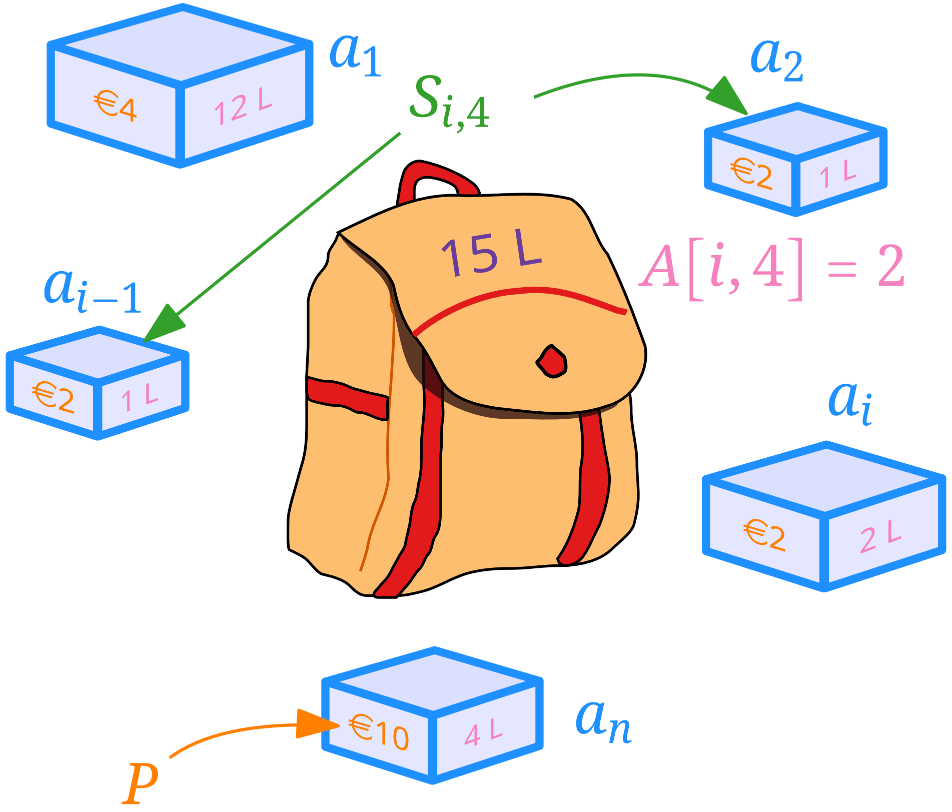


Pseudo-Polynomial Alg. for KNAPSACK

$A[1, p]$ can be computed for all $p \in \{0, \ldots, nP\}$. Note: $A[1, 0] = 0$.

Set $A[i, p] := \infty$ for $p < 0$ (for convenience).

$A[i+1, p] = \min\{ \qquad \}$

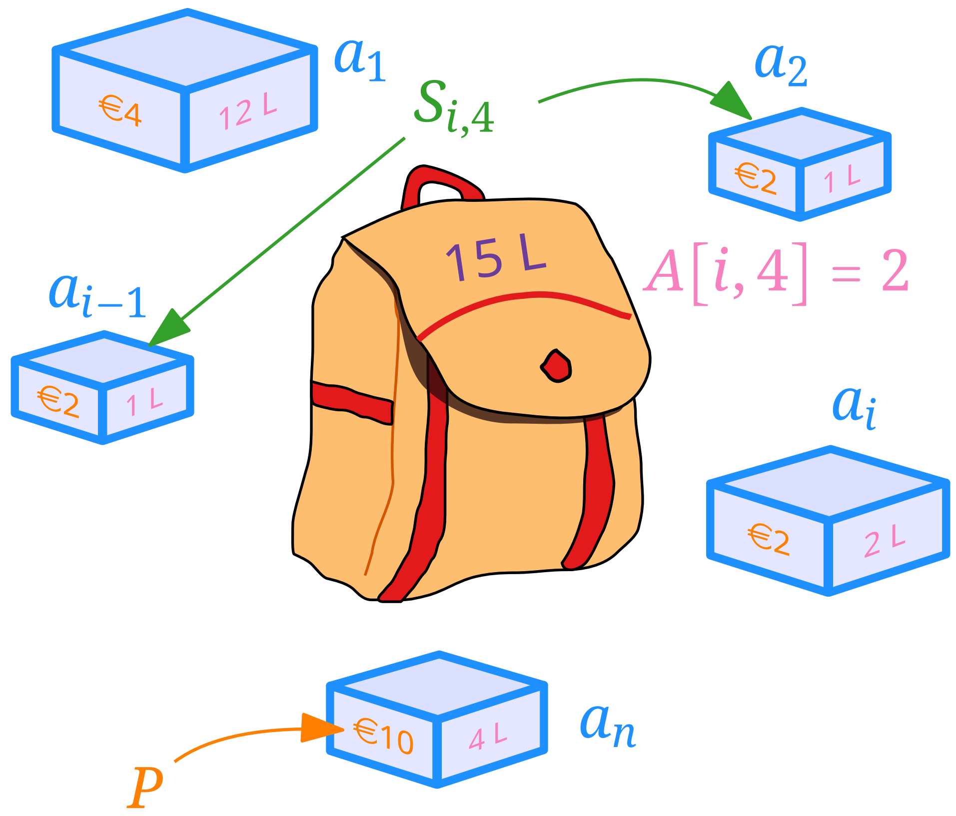


Pseudo-Polynomial Alg. for KNAPSACK

$A[1,p]$ can be computed for all $p \in \{0,\dots,nP\}$. Note: $A[1,0] = 0$.

Set $A[i,p] := \infty$ for $p < 0$ (for convenience).

$A[i+1,p] = \min\{A[i,p], \qquad\qquad\qquad\qquad \}$

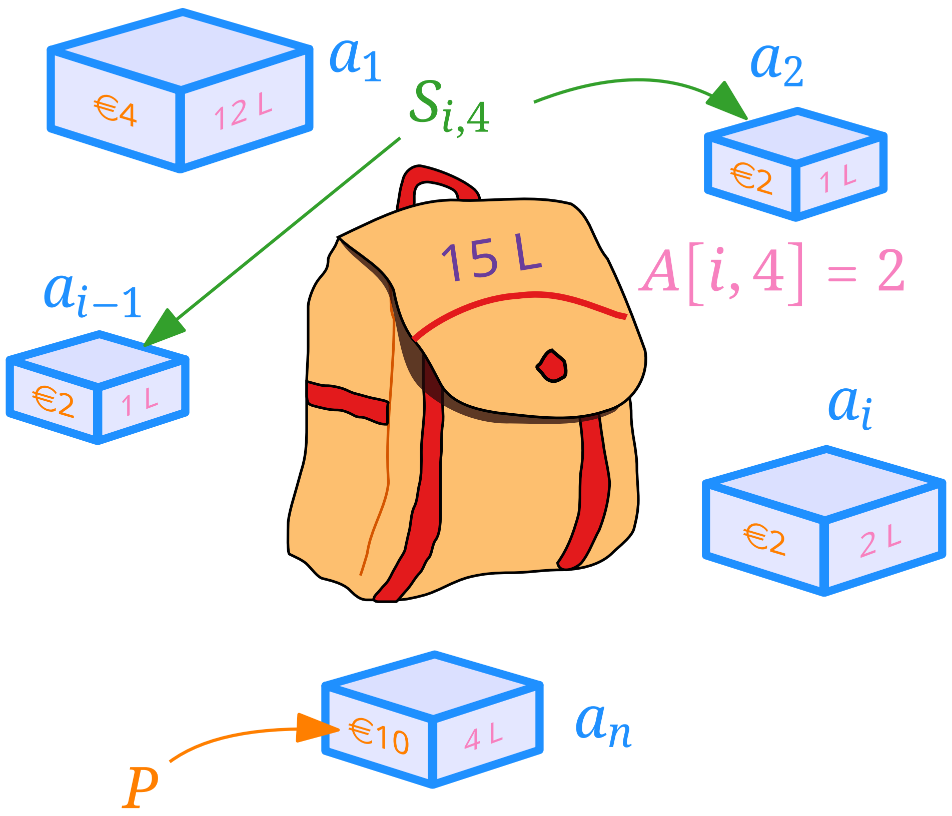


Pseudo-Polynomial Alg. for KNAPSACK

$A[1, p]$ can be computed for all $p \in \{0, \ldots, nP\}$. Note: $A[1, 0] = 0$.

Set $A[i, p] := \infty$ for $p < 0$ (for convenience).

$A[i+1, p] = \min\{A[i, p],\ \mathrm{size}(a_{i+1}) + \qquad\qquad\qquad\qquad \}$

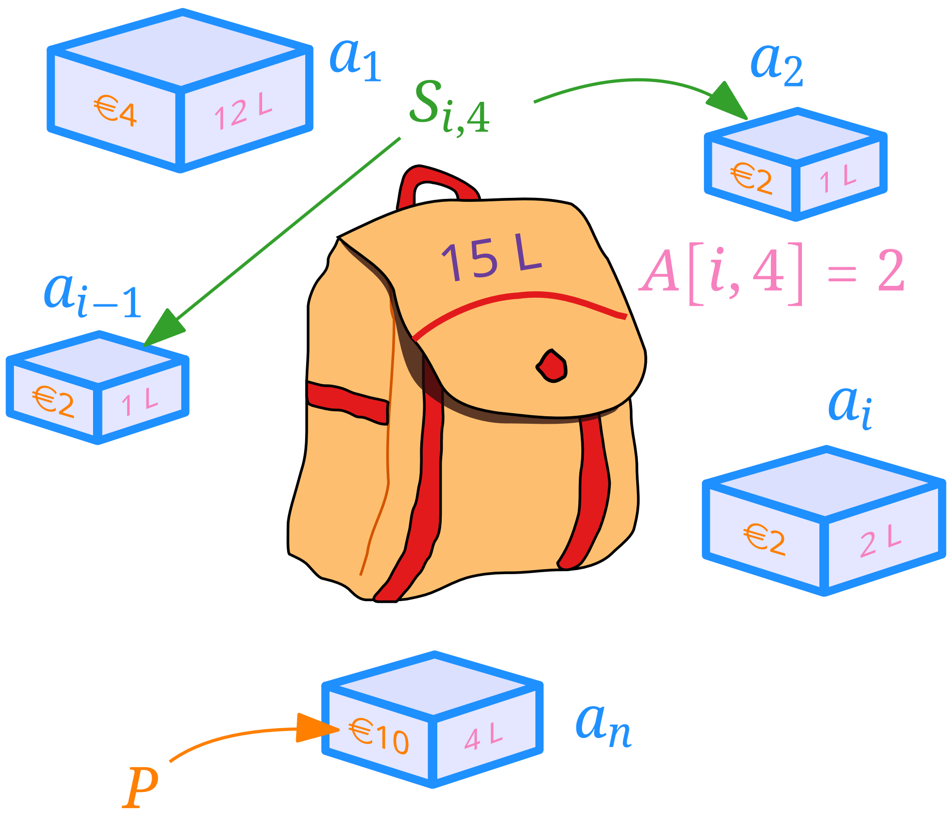


Pseudo-Polynomial Alg. for KNAPSACK

$A[1, p]$ can be computed for all $p \in \{0, \ldots, nP\}$. Note: $A[1, 0] = 0$.

Set $A[i, p] := \infty$ for $p < 0$ (for convenience).

$A[i+1, p] = \min\{A[i, p],\ \mathrm{size}(a_{i+1}) + A[i, p - \mathrm{profit}(a_{i+1})]\}$



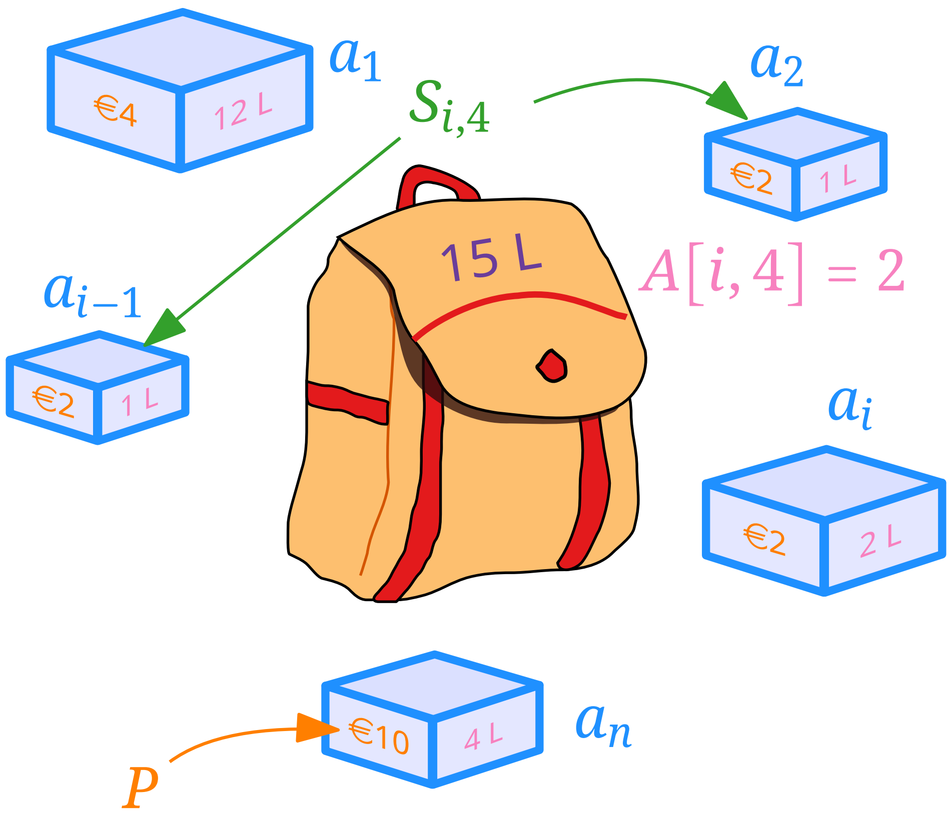
Pseudo-Polynomial Alg. for KNAPSACK

$A[1, p]$ can be computed for all $p \in \{0, \ldots, nP\}$. Note: $A[1, 0] = 0$.

Set $A[i, p] := \infty$ for $p < 0$ (for convenience).

$A[i+1, p] = \min\{A[i, p],\ \mathrm{size}(a_{i+1}) + A[i, p - \mathrm{profit}(a_{i+1})]\}$

$\Rightarrow$ All values $A[i, p]$ can be computed in total time $O(\ \ ?\ \ )$.



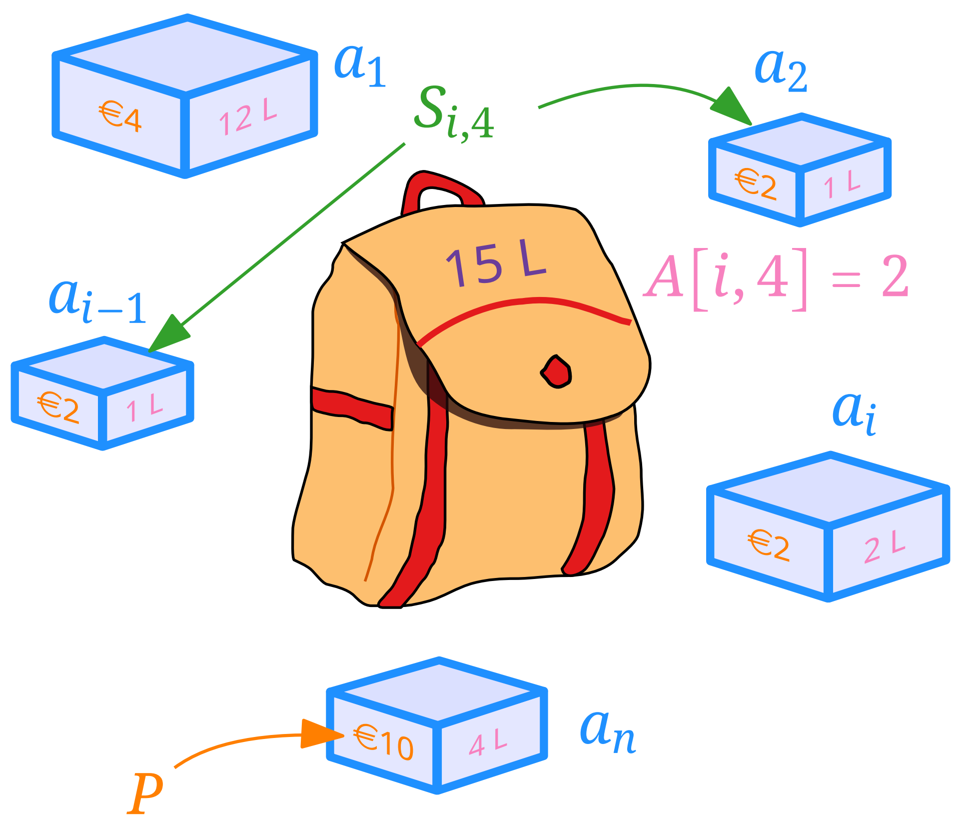
Pseudo-Polynomial Alg. for KNAPSACK

$A[1, p]$ can be computed for all $p \in \{0, \ldots, nP\}$. Note: $A[1, 0] = 0$.

Set $A[i, p] := \infty$ for $p < 0$ (for convenience).

$A[i+1, p] = \min\{A[i, p],\ \mathrm{size}(a_{i+1}) + A[i, p - \mathrm{profit}(a_{i+1})]\}$

$\Rightarrow$ All values $A[i, p]$ can be computed in total time $O(n^2 P)$.



Pseudo-Polynomial Alg. for KNAPSACK

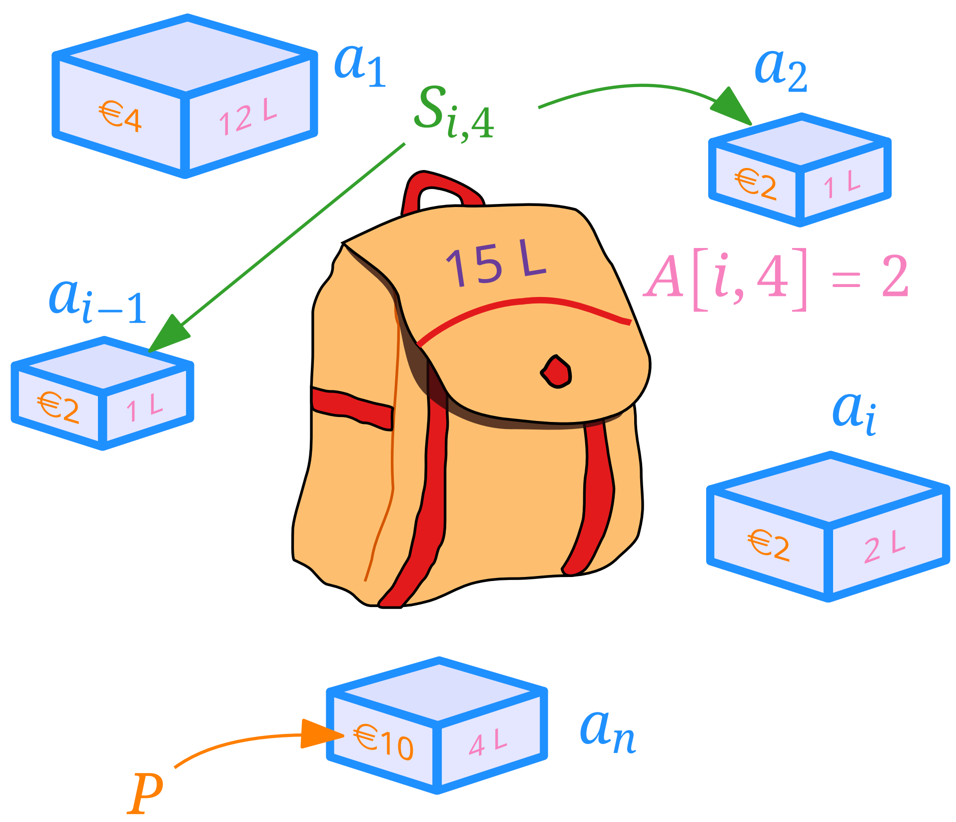
$A[1,p]$ can be computed for all $p \in \{0,\dots,nP\}$. Note: $A[1,0]=0$.

Set $A[i,p] := \infty$ for $p < 0$ (for convenience).

$A[i+1,p] = \min\{A[i,p],\ \mathrm{size}(a_{i+1}) + A[i, p - \mathrm{profit}(a_{i+1})]\}$

$\Rightarrow$ All values $A[i,p]$ can be computed in total time $O(n^2P)$.

$\Rightarrow$ OPT can be computed in $O(n^2P)$ total time.



Pseudo-Polynomial Alg. for KNAPSACK

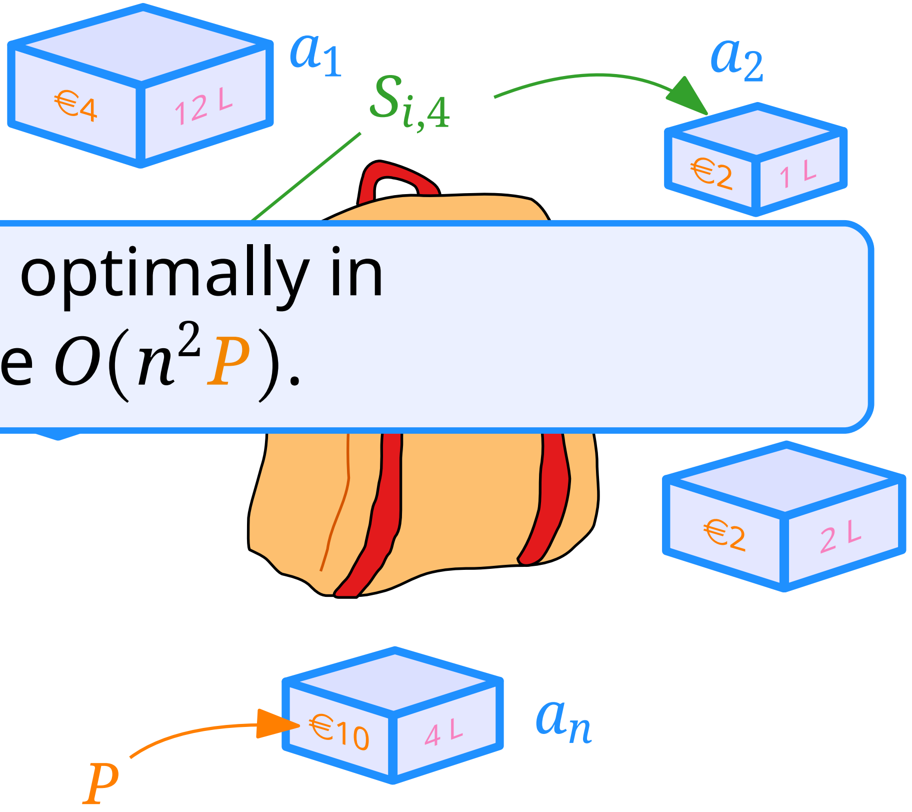
$A[1,p]$ can be computed for all $p \in \{0,\ldots,nP\}$. Note: $A[1,0]=0$.

Set $A[i,p] := \infty$ for $p<0$ (for convenience).

$A[i+1,p] = \min\{A[i,p],\ \mathrm{size}(a_{i+1}) + A[i, p - \mathrm{profit}(a_{i+1})]\}$

$\Rightarrow$ All values $A[i,p]$ can be computed in total time $O(n^2P)$.

$\Rightarrow$ OPT can be computed in $O(n^2P)$ total time.



Theorem. KNAPSACK can be solved optimally in pseudo-polynomial time $O(n^2P)$.

Pseudo-Polynomial Alg. for KNAPSACK

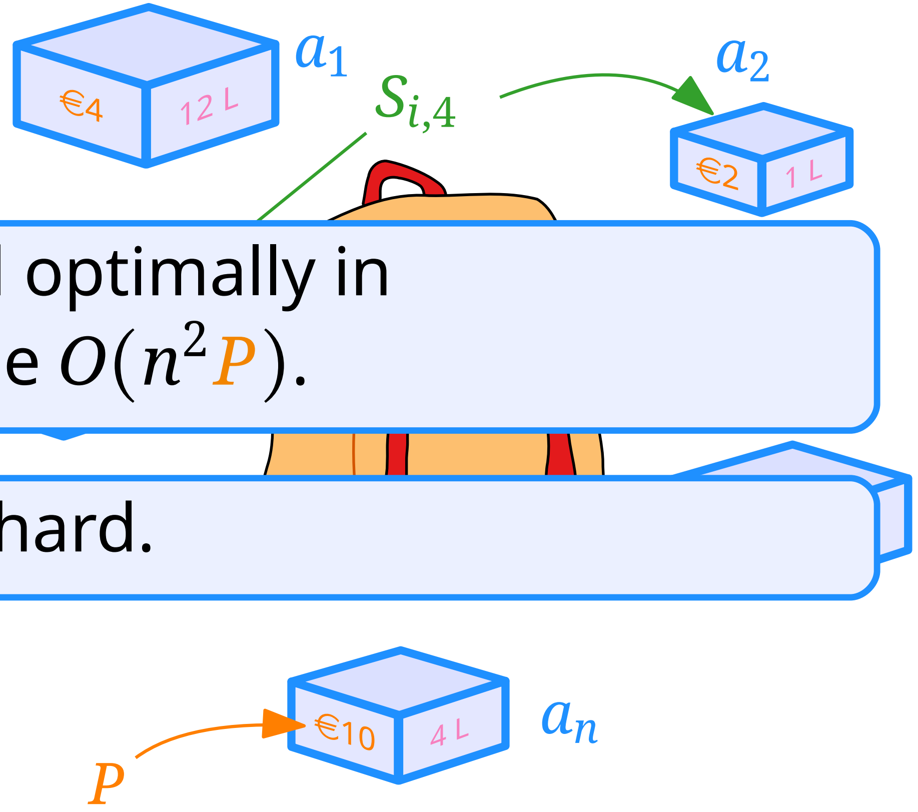
$A[1, p]$ can be computed for all $p \in \{0, \dots, nP\}$. Note: $A[1, 0] = 0$.

Set $A[i, p] := \infty$ for $p < 0$ (for convenience).

$A[i+1, p] = \min\{A[i, p],\ \mathrm{size}(a_{i+1}) + A[i, p - \mathrm{profit}(a_{i+1})]\}$

$\Rightarrow$ All values $A[i, p]$ can be computed in total time $O(n^2 P)$.

$\Rightarrow$ OPT can be computed in $O(n^2 P)$ total time.



Theorem. KNAPSACK can be solved optimally in pseudo-polynomial time $O(n^2 P)$.

Corollary. KNAPSACK is weakly NP-hard.

Pseudo-Polynomial Alg. for KNAPSACK

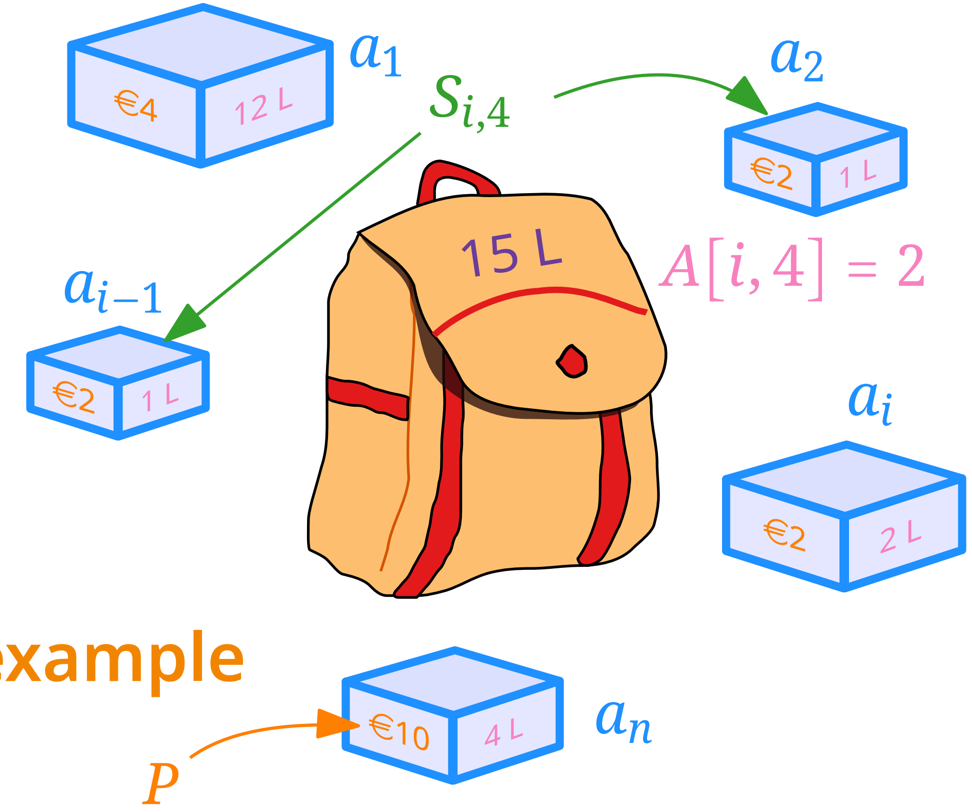
$A[1, p]$ can be computed for all $p \in \{0, \ldots, nP\}$. Note: $A[1, 0] = 0$.

Set $A[i, p] := \infty$ for $p < 0$ (for convenience).

$A[i + 1, p] = \min\{A[i, p],\ \mathrm{size}(a_{i+1}) + A[i, p - \mathrm{profit}(a_{i+1})]\}$

$\Rightarrow$ All values $A[i, p]$ can be computed in total time $O(n^2 P)$.

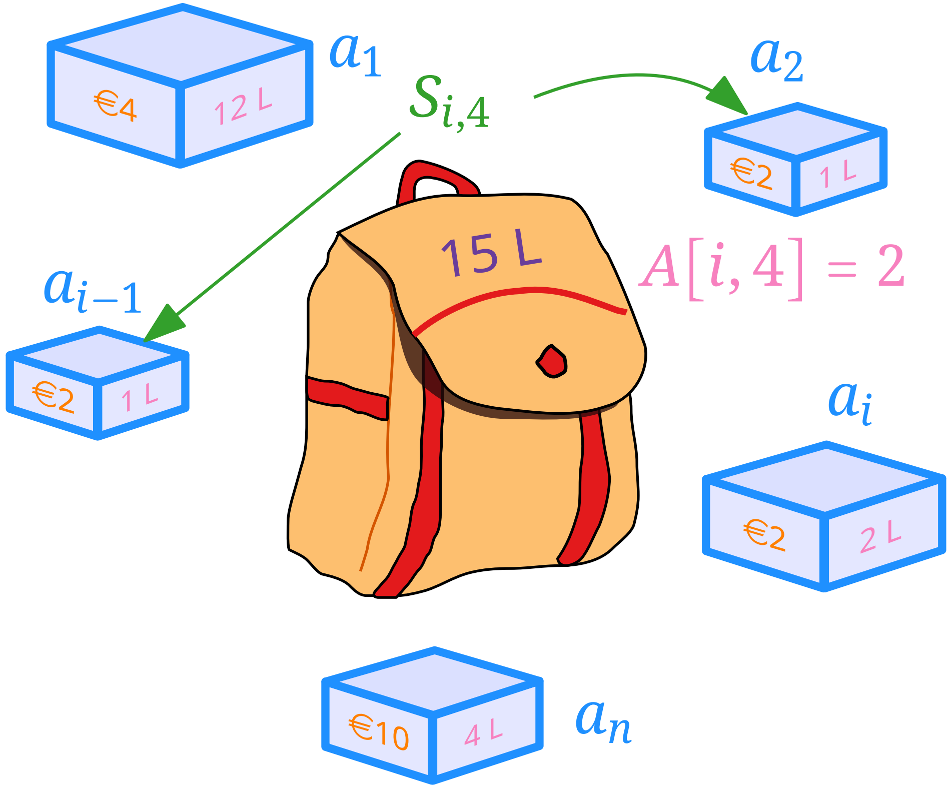
$\Rightarrow$ OPT can be computed in $O(n^2 P)$ total time.



Exercise: Execute algorithm on example

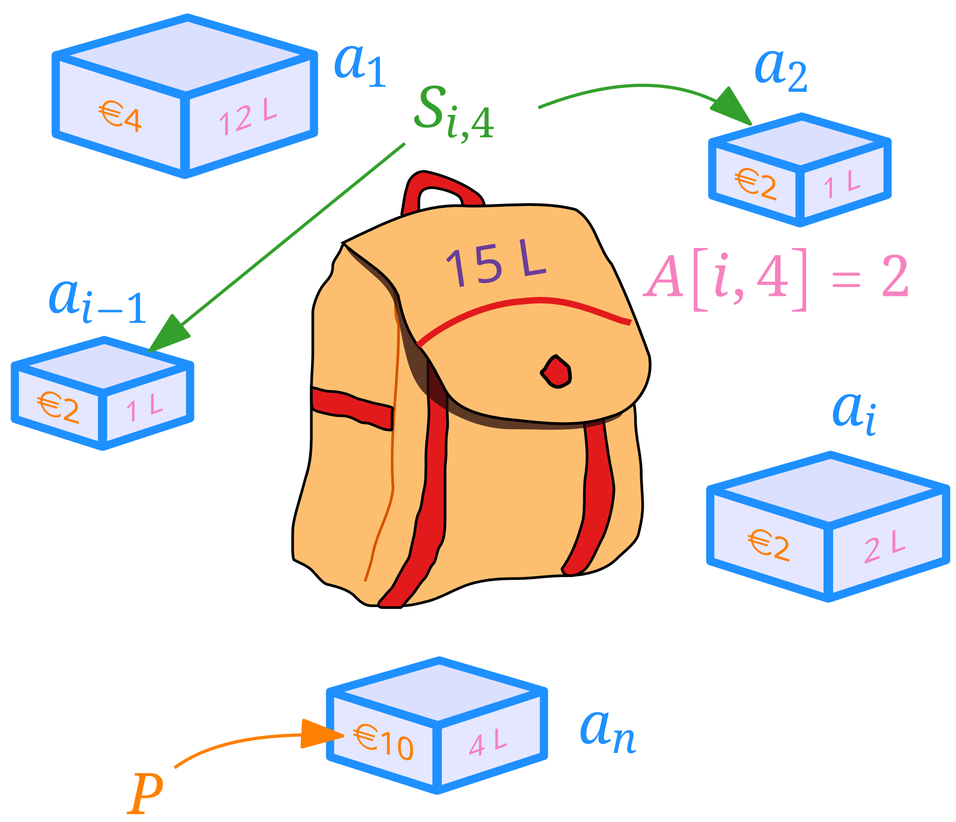
Solution to exercise

$i \setminus p$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0	∞	∞	∞	12	∞	...												
2	0	∞	1	∞	12	∞	13	∞	...										
3	0	∞	1	∞	2	∞	13	∞	14	∞	...								
4	0	∞	1	∞	2	∞	4	∞	14	∞	(16)	∞	...						
5	0	∞	1	∞	2	∞	4	∞	14	∞	4	∞	5	∞	6	∞	8	∞	(18)



Pseudo-Polynomial Alg. for KNAPSACK

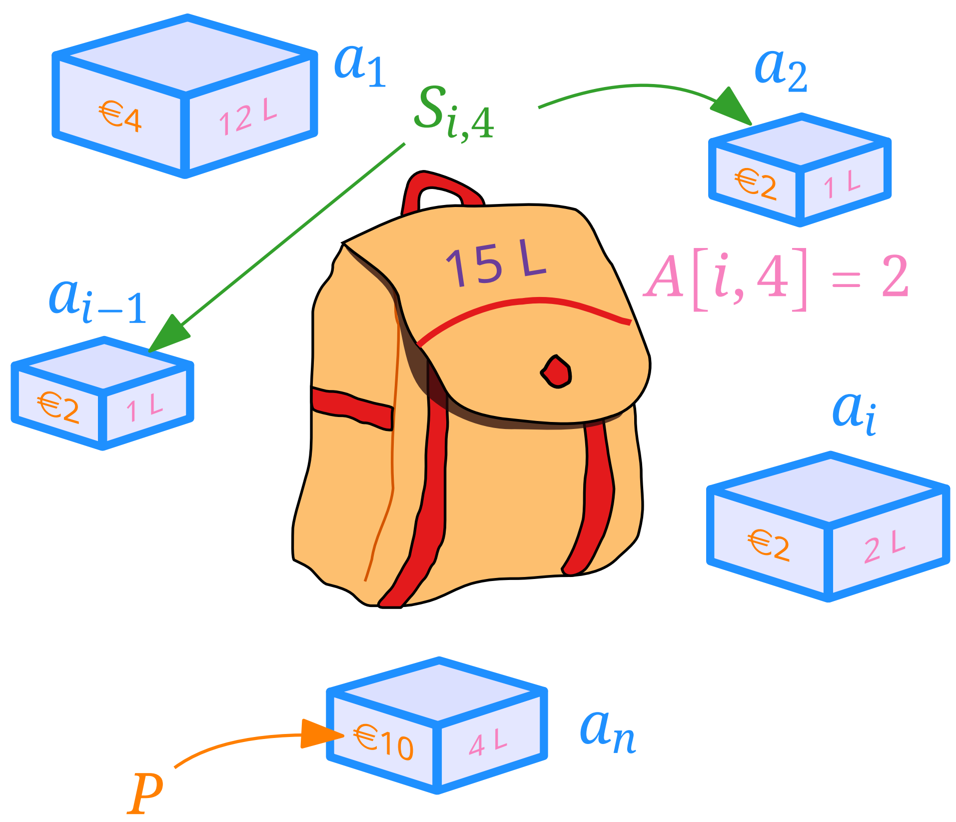
Theorem. KNAPSACK can be solved optimally in pseudo-polynomial time $O(n^2 P)$.



Pseudo-Polynomial Alg. for KNAPSACK

Theorem. KNAPSACK can be solved optimally in pseudo-polynomial time $O(n^2 P)$.

Observe. The running time $O(n^2 P)$ is polynomial in n if P is polynomial in n.



Approximation Schemes

Approximation Schemes

Let Π be an optimization problem.

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called a **polynomial-time approximation scheme** (PTAS) for Π if it outputs, for every input (I, ε) with $I \in D_\Pi$ and $\varepsilon > 0$, a solution $s \in S_\Pi(I)$ such that

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called a **polynomial-time approximation scheme** (PTAS) for Π
if it outputs, for every input (I, ε) with $I \in D_\Pi$ and $\varepsilon > 0$,
a solution $s \in S_\Pi(I)$ such that

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$ if Π is a minimization problem,

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called a **polynomial-time approximation scheme** (PTAS) for Π
if it outputs, for every input (I, ε) with $I \in D_\Pi$ and $\varepsilon > 0$,
a solution $s \in S_\Pi(I)$ such that

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$ if Π is a minimization problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$ if Π is a maximization problem,

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called a **polynomial-time approximation scheme** (PTAS) for Π
if it outputs, for every input (I, ε) with $I \in D_\Pi$ and $\varepsilon > 0$,
a solution $s \in S_\Pi(I)$ such that

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$ if Π is a minimization problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$ if Π is a maximization problem,

and the runtime of $\mathcal{A}$ is polynomial in $|I|$ for **every fixed** $\varepsilon > 0$.

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called a **polynomial-time approximation scheme** (PTAS) for Π if it outputs, for every input (I, ε) with $I \in D_\Pi$ and $\varepsilon > 0$, a solution $s \in S_\Pi(I)$ such that

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$ if Π is a minimization problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$ if Π is a maximization problem,

and the runtime of $\mathcal{A}$ is polynomial in $|I|$ for **every fixed** $\varepsilon > 0$.

$\mathcal{A}$ is called **fully polynomial-time approximation scheme** (FPTAS) if its running time is polynomial in $|I|$ and $1/\varepsilon$.

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called a **polynomial-time approximation scheme** (PTAS) for Π
if it outputs, for every input (I, ε) with $I \in D_\Pi$ and $\varepsilon > 0$,
a solution $s \in S_\Pi(I)$ such that

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$ if Π is a minimization problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$ if Π is a maximization problem,

and the runtime of $\mathcal{A}$ is polynomial in $|I|$ for **every fixed** $\varepsilon > 0$.

$\mathcal{A}$ is called **fully polynomial-time approximation scheme** (FPTAS) if its running time is polynomial in $|I|$ and $1/\varepsilon$.

Example running times

- $O(n^{1/\varepsilon}) \rightsquigarrow$
- $O(n^3/\varepsilon^2) \rightsquigarrow$
- $O(2^{1/\varepsilon} n^4) \rightsquigarrow$

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called a **polynomial-time approximation scheme** (PTAS) for Π
if it outputs, for every input (I, ε) with $I \in D_\Pi$ and $\varepsilon > 0$,
a solution $s \in S_\Pi(I)$ such that

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$ if Π is a minimization problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$ if Π is a maximization problem,

and the runtime of $\mathcal{A}$ is polynomial in $|I|$ for **every fixed** $\varepsilon > 0$.

$\mathcal{A}$ is called **fully polynomial-time approximation scheme** (FPTAS) if its running time is polynomial in $|I|$ and $1/\varepsilon$.

Example running times

- $O(n^{1/\varepsilon}) \rightsquigarrow$ PTAS
- $O(n^3/\varepsilon^2) \rightsquigarrow$
- $O(2^{1/\varepsilon} n^4) \rightsquigarrow$

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called a **polynomial-time approximation scheme** (PTAS) for Π
if it outputs, for every input (I, ε) with $I \in D_\Pi$ and $\varepsilon > 0$,
a solution $s \in S_\Pi(I)$ such that

- $\text{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \text{OPT}$ if Π is a minimization problem,
- $\text{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \text{OPT}$ if Π is a maximization problem,

and the runtime of $\mathcal{A}$ is polynomial in $|I|$ for **every fixed** $\varepsilon > 0$.

$\mathcal{A}$ is called **fully polynomial-time approximation scheme** (FPTAS) if its running time is polynomial in $|I|$ and $1/\varepsilon$.

Example running times

- $O(n^{1/\varepsilon}) \rightsquigarrow$ PTAS
- $O(n^3/\varepsilon^2) \rightsquigarrow$ FPTAS
- $O(2^{1/\varepsilon} n^4) \rightsquigarrow$

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called a **polynomial-time approximation scheme** (PTAS) for Π if it outputs, for every input (I, ε) with $I \in D_\Pi$ and $\varepsilon > 0$, a solution $s \in S_\Pi(I)$ such that

- $\text{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \text{OPT}$ if Π is a minimization problem,
- $\text{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \text{OPT}$ if Π is a maximization problem,

and the runtime of $\mathcal{A}$ is polynomial in $|I|$ for **every fixed** $\varepsilon > 0$.

$\mathcal{A}$ is called **fully polynomial-time approximation scheme** (FPTAS) if its running time is polynomial in $|I|$ and $1/\varepsilon$.

Example running times

- $O(n^{1/\varepsilon}) \rightsquigarrow$ PTAS
- $O(n^3/\varepsilon^2) \rightsquigarrow$ FPTAS
- $O(2^{1/\varepsilon} n^4) \rightsquigarrow$ PTAS

FPTAS for KNAPSACK

An FPTAS for KNAPSACK via Scaling

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε). . .

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor

$\text{profit}'(a_i) =$

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor

$\text{profit}'(a_i) = \lfloor \text{profit}(a_i)/K \rfloor$

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor

$\mathrm{profit}'(a_i) = \lfloor \mathrm{profit}(a_i)/K \rfloor$

Compute optimal solution S' for I w.r.t. $\mathrm{profit}'(\cdot)$.

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for Knapsack via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor

$\text{profit}'(a_i) = \lfloor \text{profit}(a_i) / K \rfloor$

Compute optimal solution S' for I w.r.t. $\text{profit}'(\cdot)$.

return S'

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor
$\mathrm{profit}'(a_i) = \lfloor \mathrm{profit}(a_i)/K \rfloor$
Compute optimal solution S' for I w.r.t. $\mathrm{profit}'(\cdot)$.
return S'

Lemma. $\mathrm{profit}(S') \geq (1 - \varepsilon) \cdot \mathrm{OPT}$.

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor
$\mathrm{profit}'(a_i) = \lfloor \mathrm{profit}(a_i) / K \rfloor$
Compute optimal solution S' for I w.r.t. $\mathrm{profit}'(\cdot)$.
return S'

Lemma. $\mathrm{profit}(S') \geq (1 - \varepsilon) \cdot \mathrm{OPT}$.

Proof. Let $\mathrm{OPT} = \{o_1, \ldots, o_\ell\}$.

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor
$\mathrm{profit}'(a_i) = \lfloor \mathrm{profit}(a_i) / K \rfloor$
Compute optimal solution S' for I w.r.t. $\mathrm{profit}'(\cdot)$.
return S'

Lemma. $\mathrm{profit}(S') \geq (1 - \varepsilon) \cdot \mathrm{OPT}$.

Proof. Let $\mathrm{OPT} = \{o_1, \dots, o_\ell\}$.

Obs. 1. For $i = 1, \dots, \ell$, $\leq K \cdot \mathrm{profit}'(o_i) \leq$

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor
$\mathrm{profit}'(a_i) = \lfloor \mathrm{profit}(a_i)/K \rfloor$
Compute optimal solution S' for I w.r.t. $\mathrm{profit}'(\cdot)$.
return S'

Lemma. $\mathrm{profit}(S') \geq (1-\varepsilon) \cdot \mathrm{OPT}$.

Proof. Let $\mathrm{OPT} = \{o_1, \dots, o_\ell\}$.

Obs. 1. For $i = 1, \dots, \ell$, $\leq K \cdot \mathrm{profit}'(o_i) \leq \mathrm{profit}(o_i)$

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P/n$ // scaling factor
$\mathrm{profit}'(a_i) = \lfloor \mathrm{profit}(a_i)/K \rfloor$
Compute optimal solution S' for I w.r.t. $\mathrm{profit}'(\cdot)$.
return S'

Lemma. $\mathrm{profit}(S') \geq (1-\varepsilon) \cdot \mathrm{OPT}$.

Proof. Let $\mathrm{OPT} = \{o_1, \dots, o_\ell\}$.

Obs. 1. For $i = 1, \dots, \ell$, $\mathrm{profit}(o_i) - K \leq K \cdot \mathrm{profit}'(o_i) \leq \mathrm{profit}(o_i)$

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for Knapsack via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor

$\mathrm{profit}'(a_i) = \lfloor \mathrm{profit}(a_i)/K \rfloor$

Compute optimal solution S' for I w.r.t. $\mathrm{profit}'(\cdot)$.

return S'

Lemma. $\mathrm{profit}(S') \geq (1 - \varepsilon) \cdot \mathrm{OPT}$.

Proof. Let $\mathrm{OPT} = \{o_1, \dots, o_\ell\}$.

Obs. 1. For $i = 1, \dots, \ell$, $\mathrm{profit}(o_i) - K \leq K \cdot \mathrm{profit}'(o_i) \leq \mathrm{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \mathrm{profit}'(o_i) \geq$

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for Knapsack via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor

$\text{profit}'(a_i) = \lfloor \text{profit}(a_i)/K \rfloor$

Compute optimal solution S' for I w.r.t. $\text{profit}'(\cdot)$.

return S'

Lemma. $\text{profit}(S') \geq (1 - \varepsilon) \cdot \text{OPT}$.

Proof. Let $\text{OPT} = \{o_1, \ldots, o_\ell\}$.

Obs. 1. For $i = 1, \ldots, \ell$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - \ell K \geq$

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor
$\mathrm{profit}'(a_i) = \lfloor \mathrm{profit}(a_i)/K \rfloor$
Compute optimal solution S' for I w.r.t. $\mathrm{profit}'(\cdot)$.
return S'

Lemma. $\mathrm{profit}(S') \geq (1 - \varepsilon) \cdot \mathrm{OPT}$.

Proof. Let $\mathrm{OPT} = \{o_1, \dots, o_\ell\}$.

Obs. 1. For $i = 1, \dots, \ell$, $\mathrm{profit}(o_i) - K \leq K \cdot \mathrm{profit}'(o_i) \leq \mathrm{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \mathrm{profit}'(o_i) \geq \mathrm{OPT} - \ell K \geq \mathrm{OPT} - nK =$

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for $\textsc{Knapsack}$ via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor
$\text{profit}'(a_i) = \lfloor \text{profit}(a_i) / K \rfloor$
Compute optimal solution S' for I w.r.t. $\text{profit}'(\cdot)$.
return S'

Lemma. $\text{profit}(S') \geq (1 - \varepsilon) \cdot \text{OPT}$.

Proof. Let $\text{OPT} = \{o_1, \dots, o_\ell\}$.

Obs. 1. For $i = 1, \dots, \ell$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - \ell K \geq \text{OPT} - nK = \text{OPT} - \varepsilon P$.

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor

$\text{profit}'(a_i) = \lfloor \text{profit}(a_i)/K \rfloor$

Compute optimal solution S' for I w.r.t. $\text{profit}'(\cdot)$.

return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Proof. Let $\text{OPT} = \{o_1, \ldots, o_\ell\}$.

Obs. 1. For $i = 1, \ldots, \ell$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - \ell K \geq \text{OPT} - nK = \text{OPT} - \varepsilon P$.

Obs. 2. $\geq K \cdot \sum_i \text{profit}'(o_i)$

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P/n$ // scaling factor
$\text{profit}'(a_i) = \lfloor \text{profit}(a_i)/K \rfloor$
Compute optimal solution S' for I w.r.t. $\text{profit}'(\cdot)$.
return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Proof. Let $\text{OPT} = \{o_1, \dots, o_\ell\}$.

Obs. 1. For $i = 1, \dots, \ell$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - \ell K \geq \text{OPT} - nK = \text{OPT} - \varepsilon P$.

Obs. 2. $\geq K \cdot \text{profit}'(S') \geq K \cdot \sum_i \text{profit}'(o_i)$

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor
$\mathrm{profit}'(a_i) = \lfloor \mathrm{profit}(a_i)/K \rfloor$
Compute optimal solution S' for I w.r.t. $\mathrm{profit}'(\cdot)$.
return S'

Lemma. $\mathrm{profit}(S') \geq (1 - \varepsilon) \cdot \mathrm{OPT}$.

Proof. Let $\mathrm{OPT} = \{o_1, \dots, o_\ell\}$.

Obs. 1. For $i = 1, \dots, \ell$, $\mathrm{profit}(o_i) - K \leq K \cdot \mathrm{profit}'(o_i) \leq \mathrm{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \mathrm{profit}'(o_i) \geq \mathrm{OPT} - \ell K \geq \mathrm{OPT} - nK = \mathrm{OPT} - \varepsilon P$.

Obs. 2. $\mathrm{profit}(S') \geq K \cdot \mathrm{profit}'(S') \geq K \cdot \sum_i \mathrm{profit}'(o_i)$

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor
$\mathrm{profit}'(a_i) = \lfloor \mathrm{profit}(a_i)/K \rfloor$
Compute optimal solution S' for I w.r.t. $\mathrm{profit}'(\cdot)$.
return S'

Lemma. $\mathrm{profit}(S') \geq (1 - \varepsilon) \cdot \mathrm{OPT}$.

Proof. Let $\mathrm{OPT} = \{o_1, \ldots, o_\ell\}$.

Obs. 1. For $i = 1, \ldots, \ell$, $\mathrm{profit}(o_i) - K \leq K \cdot \mathrm{profit}'(o_i) \leq \mathrm{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \mathrm{profit}'(o_i) \geq \mathrm{OPT} - \ell K \geq \mathrm{OPT} - nK = \mathrm{OPT} - \varepsilon P$.

Obs. 2. $\mathrm{profit}(S') \geq K \cdot \mathrm{profit}'(S') \geq K \cdot \sum_i \mathrm{profit}'(o_i)$

$\Rightarrow \mathrm{profit}(S') \geq$

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor
$\mathrm{profit}'(a_i) = \lfloor \mathrm{profit}(a_i)/K \rfloor$
Compute optimal solution S' for I w.r.t. $\mathrm{profit}'(\cdot)$.
return S'

Lemma. $\mathrm{profit}(S') \geq (1 - \varepsilon) \cdot \mathrm{OPT}$.

Proof. Let $\mathrm{OPT} = \{o_1, \ldots, o_\ell\}$.

Obs. 1. For $i = 1, \ldots, \ell$, $\mathrm{profit}(o_i) - K \leq K \cdot \mathrm{profit}'(o_i) \leq \mathrm{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \mathrm{profit}'(o_i) \geq \mathrm{OPT} - \ell K \geq \mathrm{OPT} - nK = \mathrm{OPT} - \varepsilon P$.

Obs. 2. $\mathrm{profit}(S') \geq K \cdot \mathrm{profit}'(S') \geq K \cdot \sum_i \mathrm{profit}'(o_i)$

$\Rightarrow \mathrm{profit}(S') \geq \mathrm{OPT} - \varepsilon P \geq$

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P/n$ // scaling factor
$\mathrm{profit}'(a_i) = \lfloor \mathrm{profit}(a_i)/K \rfloor$
Compute optimal solution S' for I w.r.t. $\mathrm{profit}'(\cdot)$.
return S'

Lemma. $\mathrm{profit}(S') \geq (1 - \varepsilon) \cdot \mathrm{OPT}$.

Proof. Let $\mathrm{OPT} = \{o_1, \dots, o_\ell\}$.

Obs. 1. For $i = 1, \dots, \ell$, $\mathrm{profit}(o_i) - K \leq K \cdot \mathrm{profit}'(o_i) \leq \mathrm{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \mathrm{profit}'(o_i) \geq \mathrm{OPT} - \ell K \geq \mathrm{OPT} - nK = \mathrm{OPT} - \varepsilon P$.

Obs. 2. $\mathrm{profit}(S') \geq K \cdot \mathrm{profit}'(S') \geq K \cdot \sum_i \mathrm{profit}'(o_i)$

$\Rightarrow \mathrm{profit}(S') \geq \mathrm{OPT} - \varepsilon P \geq \mathrm{OPT} - \varepsilon\, \mathrm{OPT} =$

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for Knapsack via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P/n$ // scaling factor

$\mathrm{profit}'(a_i) = \lfloor \mathrm{profit}(a_i)/K \rfloor$

Compute optimal solution S' for I w.r.t. $\mathrm{profit}'(\cdot)$.

return S'

Lemma. $\mathrm{profit}(S') \geq (1-\varepsilon) \cdot \mathrm{OPT}$.

Proof. Let $\mathrm{OPT} = \{o_1, \ldots, o_\ell\}$.

Obs. 1. For $i = 1, \ldots, \ell$, $\mathrm{profit}(o_i) - K \leq K \cdot \mathrm{profit}'(o_i) \leq \mathrm{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \mathrm{profit}'(o_i) \geq \mathrm{OPT} - \ell K \geq \mathrm{OPT} - nK = \mathrm{OPT} - \varepsilon P$.

Obs. 2. $\mathrm{profit}(S') \geq K \cdot \mathrm{profit}'(S') \geq K \cdot \sum_i \mathrm{profit}'(o_i)$

$\Rightarrow \mathrm{profit}(S') \geq \mathrm{OPT} - \varepsilon P \geq \mathrm{OPT} - \varepsilon\,\mathrm{OPT} = (1-\varepsilon) \cdot \mathrm{OPT}$ $\square$

FPTAS idea: **Scale** profits to polynomial size (as required by the error parameter ε)...

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P / n$ // scaling factor
$\text{profit}'(a_i) = \lfloor \text{profit}(a_i)/K \rfloor$
Compute optimal solution S' for I w.r.t. $\text{profit}'(\cdot)$.
return S'

Lemma. $\text{profit}(S') \geq (1 - \varepsilon) \cdot \text{OPT}$.

Proof. Let $\text{OPT} = \{o_1, \dots, o_\ell\}$.

Obs. 1. For $i = 1, \dots, \ell$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - \ell K \geq \text{OPT} - nK = \text{OPT} - \varepsilon P$.

Obs. 2. $\text{profit}(S') \geq K \cdot \text{profit}'(S') \geq K \cdot \sum_i \text{profit}'(o_i)$

$\Rightarrow \text{profit}(S') \geq \text{OPT} - \varepsilon P \geq \text{OPT} - \varepsilon\, \text{OPT} = (1 - \varepsilon) \cdot \text{OPT}$ $\square$

Theorem. KnapsackScaling is an FPTAS for KNAPSACK with running time $O(n^3/\varepsilon)$

An FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K = \varepsilon P/n$ // scaling factor
$\text{profit}'(a_i) = \lfloor \text{profit}(a_i)/K \rfloor$
Compute optimal solution S' for I w.r.t. $\text{profit}'(\cdot)$.
return S'

Lemma. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

Proof. Let $\text{OPT} = \{o_1, \ldots, o_\ell\}$.

Obs. 1. For $i = 1, \ldots, \ell$, $\text{profit}(o_i) - K \leq K \cdot \text{profit}'(o_i) \leq \text{profit}(o_i)$

$\Rightarrow K \cdot \sum_i \text{profit}'(o_i) \geq \text{OPT} - \ell K \geq \text{OPT} - nK = \text{OPT} - \varepsilon P$.

Obs. 2. $\text{profit}(S') \geq K \cdot \text{profit}'(S') \geq K \cdot \sum_i \text{profit}'(o_i)$

$\Rightarrow \text{profit}(S') \geq \text{OPT} - \varepsilon P \geq \text{OPT} - \varepsilon\,\text{OPT} = (1-\varepsilon) \cdot \text{OPT}$ $\square$

Theorem. KnapsackScaling is an FPTAS for KNAPSACK with running time $O(n^3/\varepsilon) = O\left(n^2 \cdot \frac{P}{\varepsilon P/n}\right)$.

FPTAS vs strong NP-hardness

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_\mathrm{u})$ for all instances I of Π.

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS, then

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS, then there is a pseudo-polynomial algorithm for Π.

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_\mathrm{u})$ for all instances I of Π. If Π has an FPTAS, then there is a pseudo-polynomial algorithm for Π.

Proof.

Assuming there is an FPTAS for Π (in $q(|I|, 1/\varepsilon)$ time).

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS, then there is a pseudo-polynomial algorithm for Π.

Proof.

Assuming there is an FPTAS for Π (in $q(|I|, 1/\varepsilon)$ time).

Set $\varepsilon =$

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS, then there is a pseudo-polynomial algorithm for Π.

Proof.

Assuming there is an FPTAS for Π (in $q(|I|, 1/\varepsilon)$ time).

Set $\varepsilon = 1/p(|I|_{\mathrm{u}})$.

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS, then there is a pseudo-polynomial algorithm for Π.

Proof.

Assuming there is an FPTAS for Π (in $q(|I|, 1/\varepsilon)$ time).

Set $\varepsilon = 1/p(|I|_{\mathrm{u}})$.

$\Rightarrow \mathrm{ALG} \leq (1+\varepsilon)\mathrm{OPT} \; <$

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS, then there is a pseudo-polynomial algorithm for Π.

Proof.

Assuming there is an FPTAS for Π (in $q(|I|, 1/\varepsilon)$ time).

Set $\varepsilon = 1/p(|I|_{\mathrm{u}})$.

$\Rightarrow \mathrm{ALG} \leq (1+\varepsilon)\mathrm{OPT} \;<$

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS, then there is a pseudo-polynomial algorithm for Π.

Proof.

Assuming there is an FPTAS for Π (in $q(|I|, 1/\varepsilon)$ time).

Set $\varepsilon = 1/p(|I|_{\mathrm{u}})$.

$\Rightarrow \mathrm{ALG} \leq (1+\varepsilon)\mathrm{OPT} < \quad \mathrm{OPT} + \varepsilon p(|I|_{\mathrm{u}}) =$

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS, then there is a pseudo-polynomial algorithm for Π.

Proof.

Assuming there is an FPTAS for Π (in $q(|I|, 1/\varepsilon)$ time).

Set $\varepsilon = 1/p(|I|_{\mathrm{u}})$.

$\Rightarrow \mathrm{ALG} \leq (1+\varepsilon)\mathrm{OPT} < \mathrm{OPT} + \varepsilon p(|I|_{\mathrm{u}}) =$

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS, then there is a pseudo-polynomial algorithm for Π.

Proof.

Assuming there is an FPTAS for Π (in $q(|I|, 1/\varepsilon)$ time).

Set $\varepsilon = 1/p(|I|_{\mathrm{u}})$.

$\Rightarrow \mathrm{ALG} \leq (1+\varepsilon)\mathrm{OPT} < \mathrm{OPT} + \varepsilon p(|I|_{\mathrm{u}}) = \mathrm{OPT} + 1.$

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS, then there is a pseudo-polynomial algorithm for Π.

Proof.

Assuming there is an FPTAS for Π (in $q(|I|, 1/\varepsilon)$ time).

Set $\varepsilon = 1/p(|I|_{\mathrm{u}})$.

$\Rightarrow \mathrm{ALG} \leq (1+\varepsilon)\mathrm{OPT} < \mathrm{OPT} + \varepsilon p(|I|_{\mathrm{u}}) = \mathrm{OPT} + 1.$

$\Rightarrow \mathrm{ALG} = \mathrm{OPT}.$

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS, then there is a pseudo-polynomial algorithm for Π.

Proof.

Assuming there is an FPTAS for Π (in $q(|I|, 1/\varepsilon)$ time).

Set $\varepsilon = 1/p(|I|_{\mathrm{u}})$.

$\Rightarrow \mathrm{ALG} \leq (1+\varepsilon)\mathrm{OPT} < \mathrm{OPT} + \varepsilon p(|I|_{\mathrm{u}}) = \mathrm{OPT} + 1.$

$\Rightarrow \mathrm{ALG} = \mathrm{OPT}.$

Running time:

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_\mathrm{u})$ for all instances I of Π. If Π has an FPTAS, then there is a pseudo-polynomial algorithm for Π.

Proof.

Assuming there is an FPTAS for Π (in $q(|I|, 1/\varepsilon)$ time).

Set $\varepsilon = 1/p(|I|_\mathrm{u})$.

$\Rightarrow \mathrm{ALG} \le (1+\varepsilon)\mathrm{OPT} < \mathrm{OPT} + \varepsilon p(|I|_\mathrm{u}) = \mathrm{OPT} + 1.$

$\Rightarrow \mathrm{ALG} = \mathrm{OPT}.$

Running time: $q(|I|, p(|I|_\mathrm{u}))$

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS, then there is a pseudo-polynomial algorithm for Π.

Proof.

Assuming there is an FPTAS for Π (in $q(|I|, 1/\varepsilon)$ time).

Set $\varepsilon = 1/p(|I|_{\mathrm{u}})$.

$\Rightarrow \mathrm{ALG} \leq (1+\varepsilon)\mathrm{OPT} \;<\; \mathrm{OPT} + \varepsilon p(|I|_{\mathrm{u}}) = \mathrm{OPT} + 1.$

$\Rightarrow \mathrm{ALG} = \mathrm{OPT}.$

Running time: $q(|I|, p(|I|_{\mathrm{u}}))$, so

FPTAS and Pseudo-Poly. Algorithms

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS, then there is a pseudo-polynomial algorithm for Π.

Proof.

Assuming there is an FPTAS for Π (in $q(|I|, 1/\varepsilon)$ time).

Set $\varepsilon = 1/p(|I|_{\mathrm{u}})$.

$\Rightarrow \mathrm{ALG} \leq (1+\varepsilon)\mathrm{OPT} < \mathrm{OPT} + \varepsilon p(|I|_{\mathrm{u}}) = \mathrm{OPT} + 1.$

$\Rightarrow \mathrm{ALG} = \mathrm{OPT}.$

Running time: $q(|I|, p(|I|_{\mathrm{u}}))$, so $\mathrm{poly}(|I|_{\mathrm{u}})$.

FPTAS and Strong NP-Hardness

Theorem. A strongly NP-hard problem has no pseudo-polynomial algorithm unless $\mathrm{P} = \mathrm{NP}$.

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS, then there is a pseudo-polynomial algorithm for Π.

FPTAS and Strong NP-Hardness

Theorem. A strongly NP-hard problem has no pseudo-polynomial algorithm unless $\mathrm{P} = \mathrm{NP}$.

Theorem. Let p be a polynomial and let Π be an NP-hard minimization problem with integral objective function and $\mathrm{OPT}(I) < p(|I|_{\mathrm{u}})$ for all instances I of Π. If Π has an FPTAS, then there is a pseudo-polynomial algorithm for Π.

Corollary. Let Π be an NP-hard optimization problem that fulfills the restrictions above. If Π is strongly NP-hard, then there is no FPTAS for Π (unless $\mathrm{P} = \mathrm{NP}$).