Approximation Algorithms

Lecture 8: Approximation Schemes and the Knapsack Problem

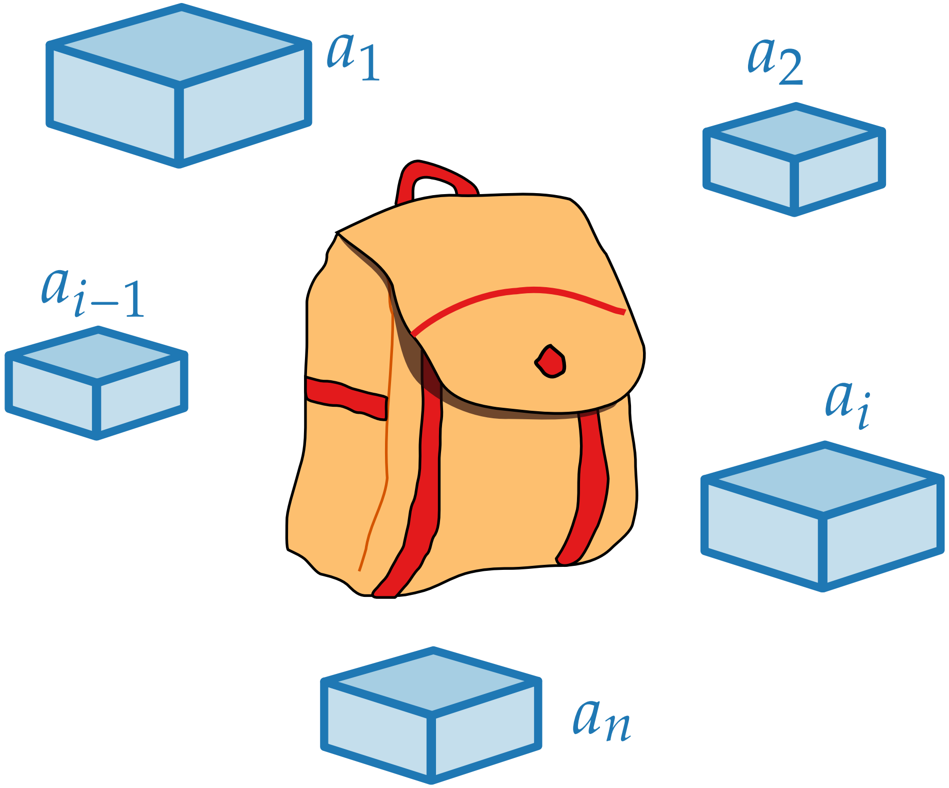
Part I: Knapsack

Joachim Spoerhase

Winter 2021/22

Knapsack

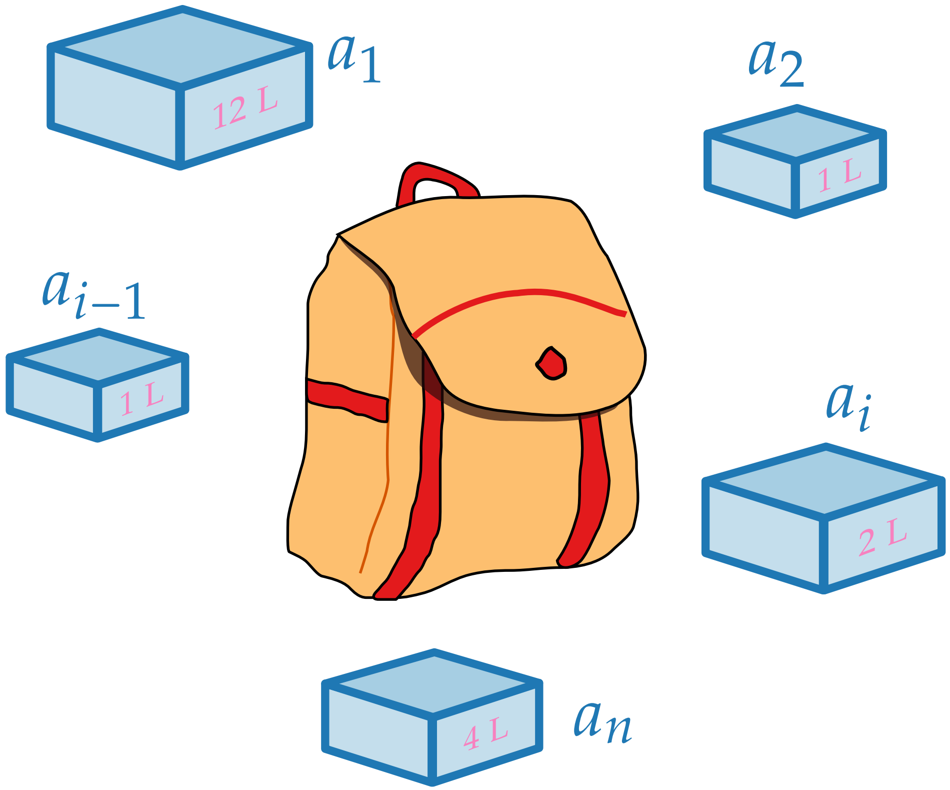
Given: ■ A set $S = \{a_1, \ldots, a_n\}$ of **objects**.



Knapsack

Given:

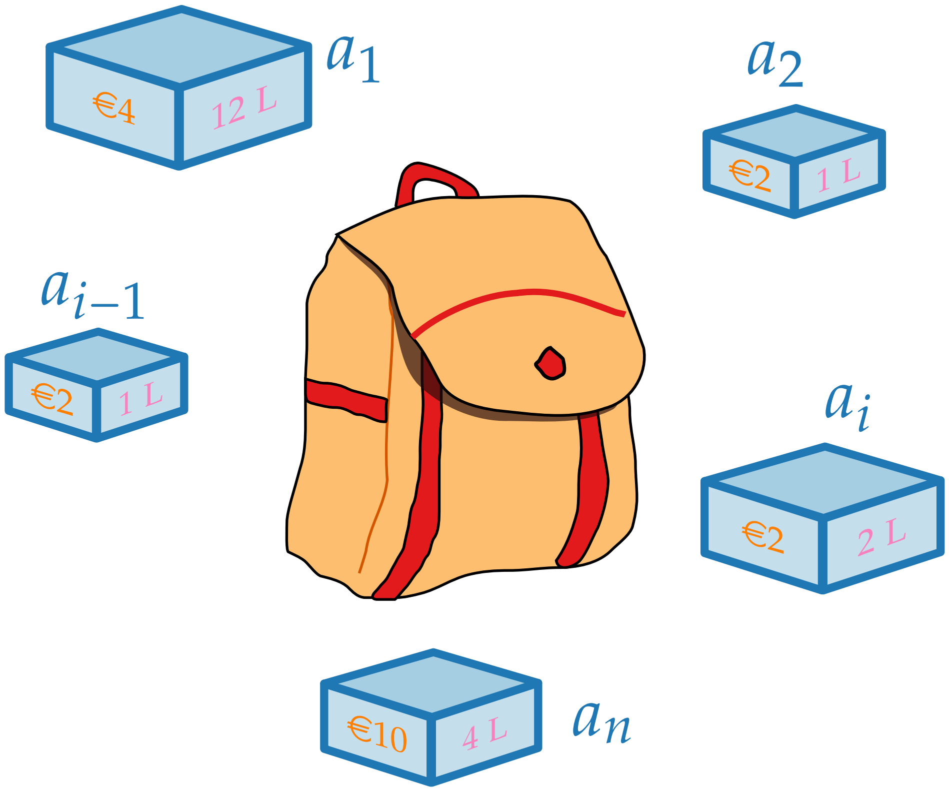
- A set $S = \{a_1, \ldots, a_n\}$ of **objects**.
- For every object a_i a **size** $\mathrm{size}(a_i) \in \mathbb{N}^+$



Knapsack

Given:

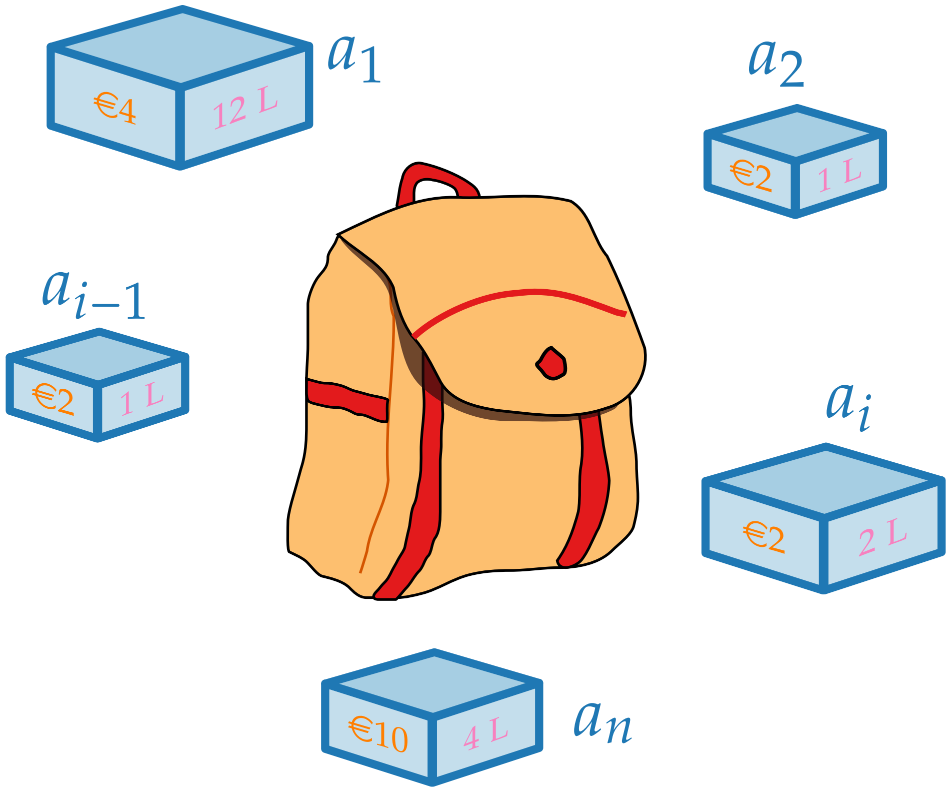
- A set $S = \{a_1, \ldots, a_n\}$ of **objects**.
- For every object a_i a **size** $\text{size}(a_i) \in \mathbb{N}^+$
- For every object a_i a **Profit** $\text{profit}(a_i) \in \mathbb{N}^+$



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** $B \in \mathbb{N}^+$

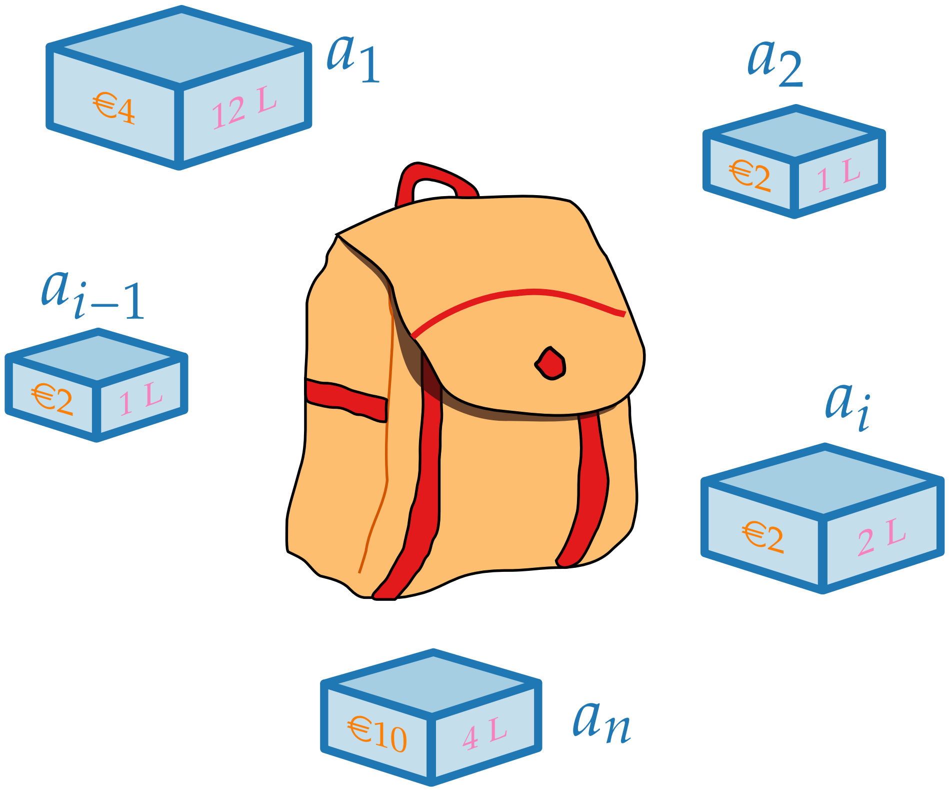


Knapsack

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.

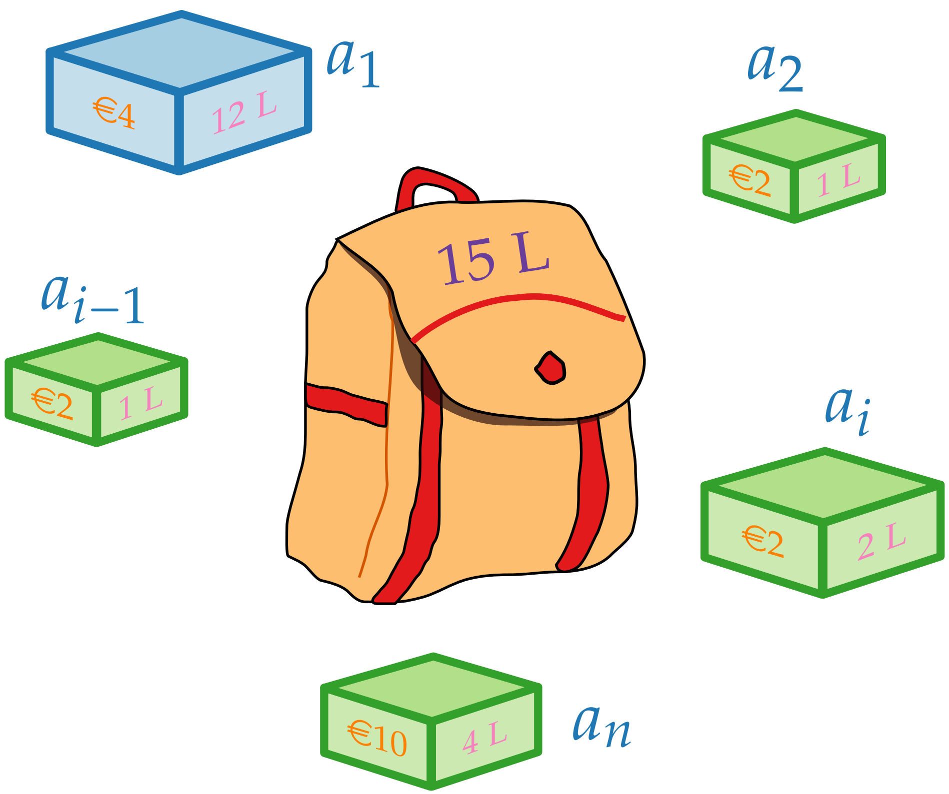


Knapsack

Given:

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

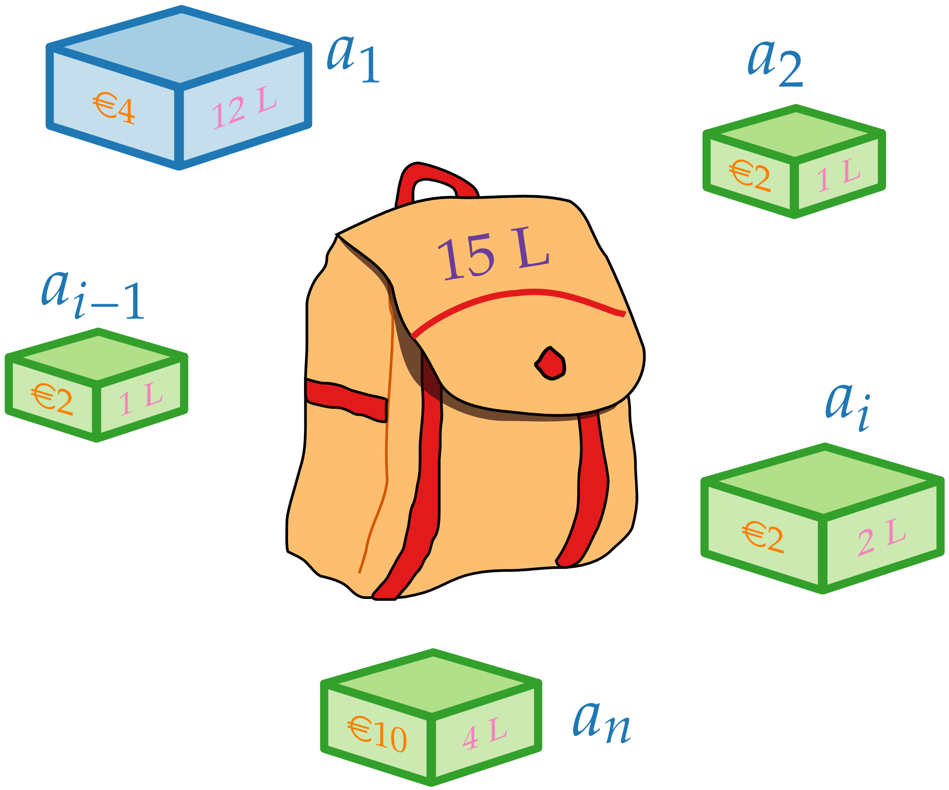


Knapsack

Given:

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

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



NP-hard

Approximation Algorithms

Lecture 8:
Approximation Schemes and the KNAPSACK Problem

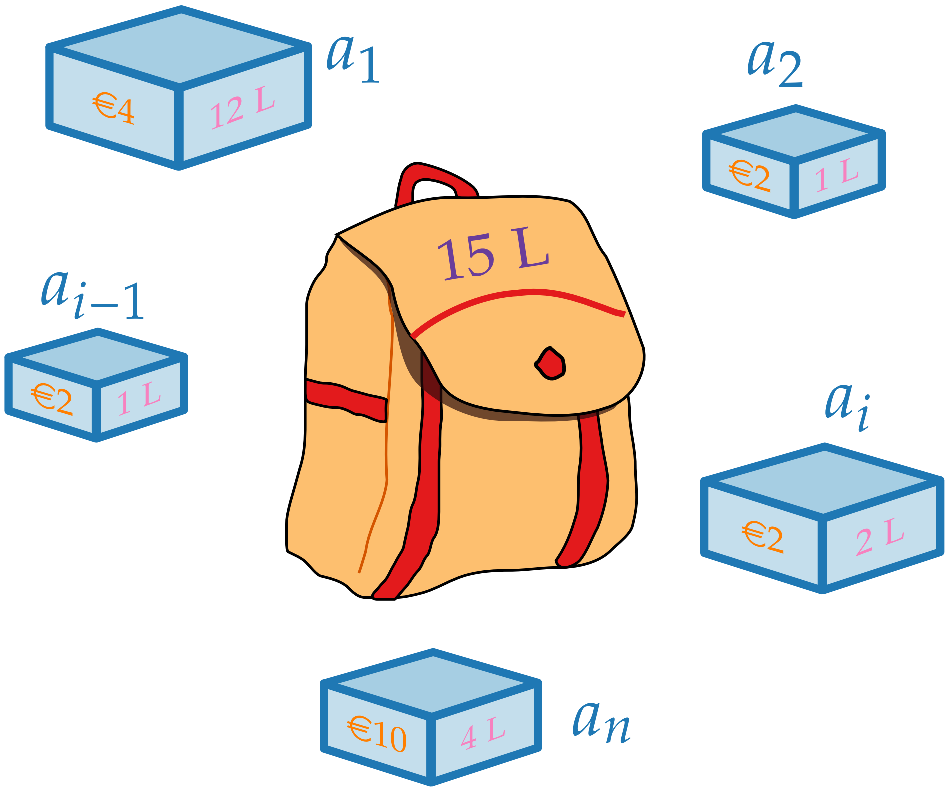
Part II:
Pseudo-Polynomial Algorithms and Strong NP-Hardness

Joachim Spoerhase

Winter 2021/22

Pseudo-Polynomial Algorithms

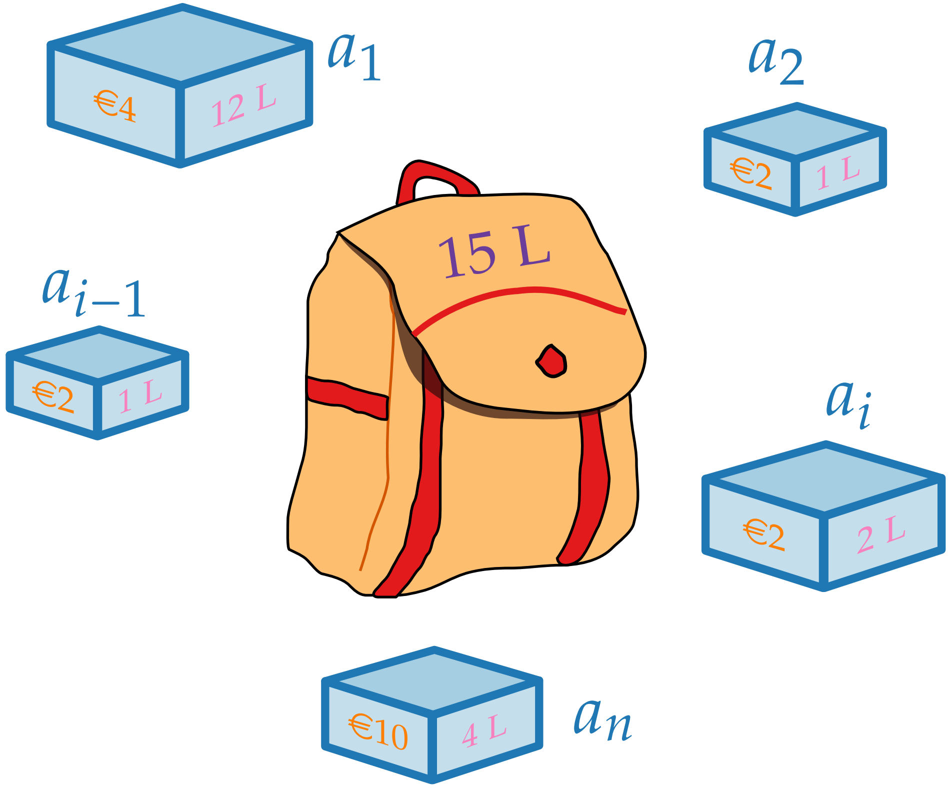
Let Π be an optimization problem whose instances can be represented by **objects** (such as sets, elements, edges, nodes) and **numbers** (such as costs, weights, profits).



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).

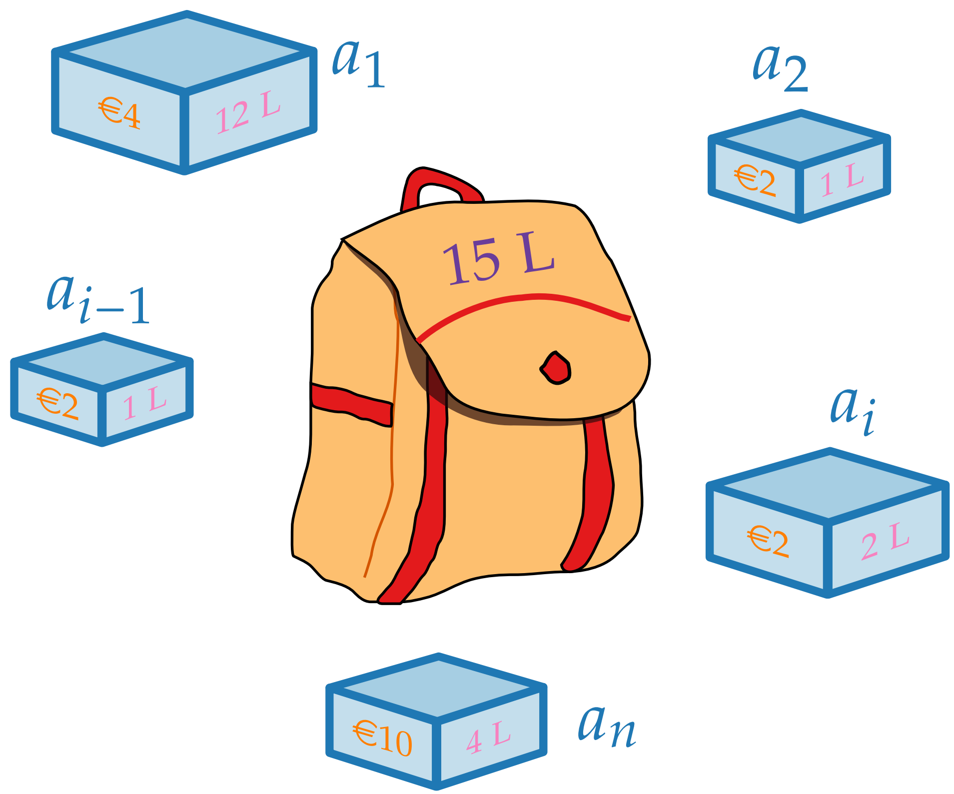
$|I|$: The size of an instance $I \in D_\Pi$, where all numbers in I are encoded in **binary**.



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**. $(5 \mathrel{\hat{=}} 101 \Rightarrow |I| = 3)$

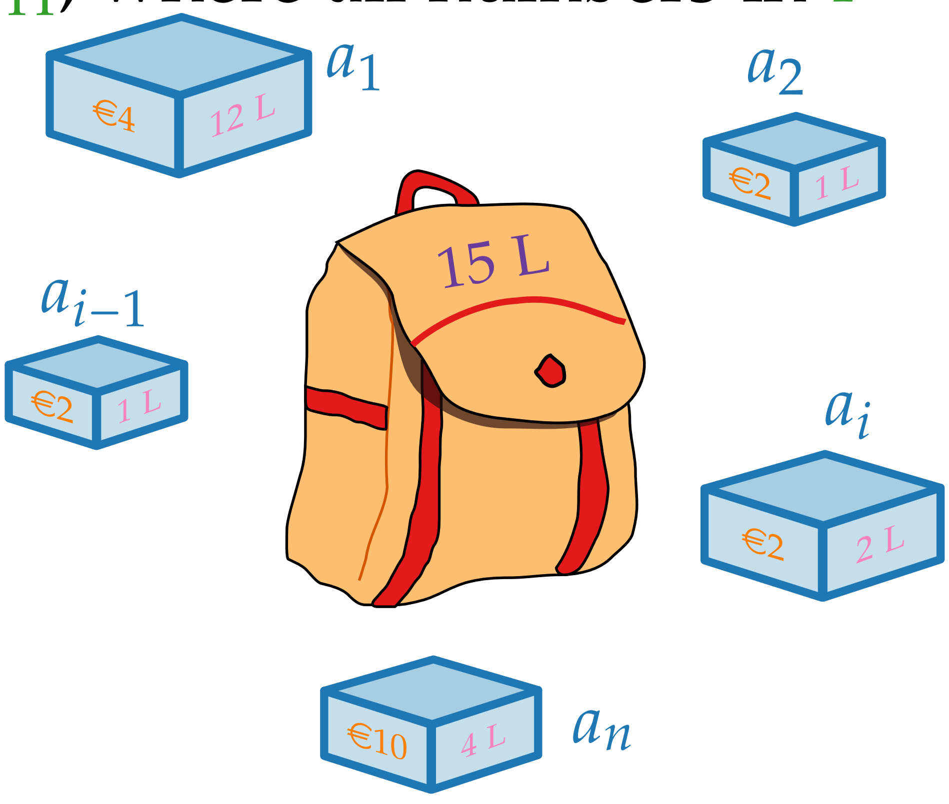


Pseudo-Polynomial Algorithms

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**. $(5 \triangleq 101 \Rightarrow |I| = 3)$

$|I|_\mathrm{u}$: 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**. $(5 \mathrel{\hat{=}} 101 \Rightarrow |I| = 3)$

$|I|_\mathrm{u}$: The size of an instance $I \in D_\Pi$, where all numbers in I are encoded in **unary**. $(5 \mathrel{\hat{=}} 11111 \Rightarrow |I|_\mathrm{u} = 5)$

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**. $(5 \mathrel{\hat{=}} 101 \Rightarrow |I| = 3)$

$|I|_\mathrm{u}$: The size of an instance $I \in D_\Pi$, where all numbers in I are encoded in **unary**. $(5 \mathrel{\hat{=}} 11111 \Rightarrow |I|_\mathrm{u} = 5)$

The running time of a polynomial 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**. $(5 \mathrel{\hat{=}} 101 \Rightarrow |I| = 3)$

$|I|_\mathrm{u}$: The size of an instance $I \in D_\Pi$, where all numbers in I are encoded in **unary**. $(5 \mathrel{\hat{=}} 11111 \Rightarrow |I|_\mathrm{u} = 5)$

The running time of a polynomial 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**. $(5 \triangleq 101 \Rightarrow |I| = 3)$

$|I|_u$: The size of an instance $I \in D_\Pi$, where all numbers in I are encoded in **unary**. $(5 \triangleq 11111 \Rightarrow |I|_u = 5)$

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

The running time of a **pseudo-polynomial algorithm** is polynomial in $|I|_u$.

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 has 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 has a pseudo-polynomial algorithm.

Theorem. A strongly NP-hard problem has no pseudo-polynomial algorithm unless P = NP.

Approximation Algorithms

Lecture 8: Approximation Schemes and the Knapsack Problem

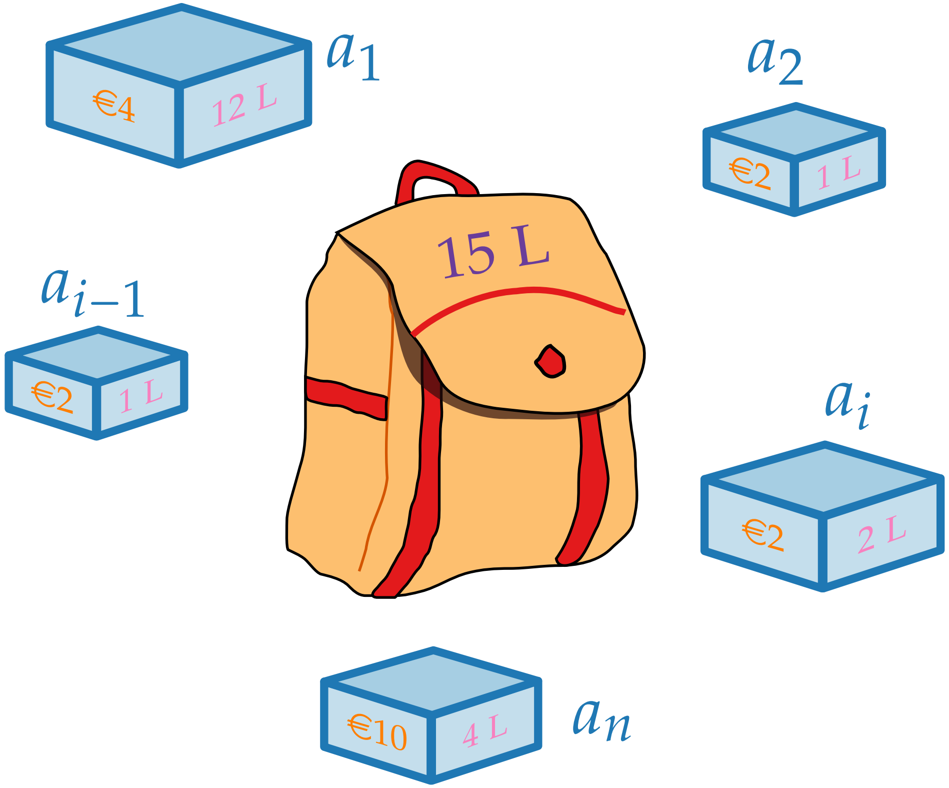
Part III: Pseudo-Polynomial Algorithm for Knapsack

Joachim Spoerhase

Winter 2021/22

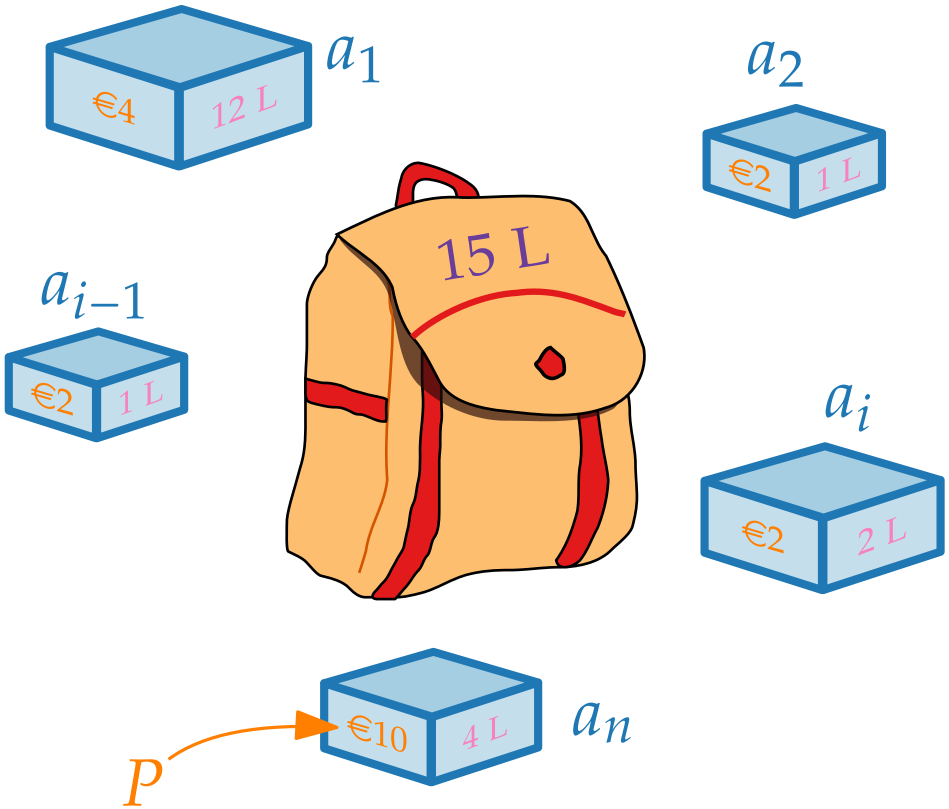
Pseudo-Polynomial Alg. for KNAPSACK

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



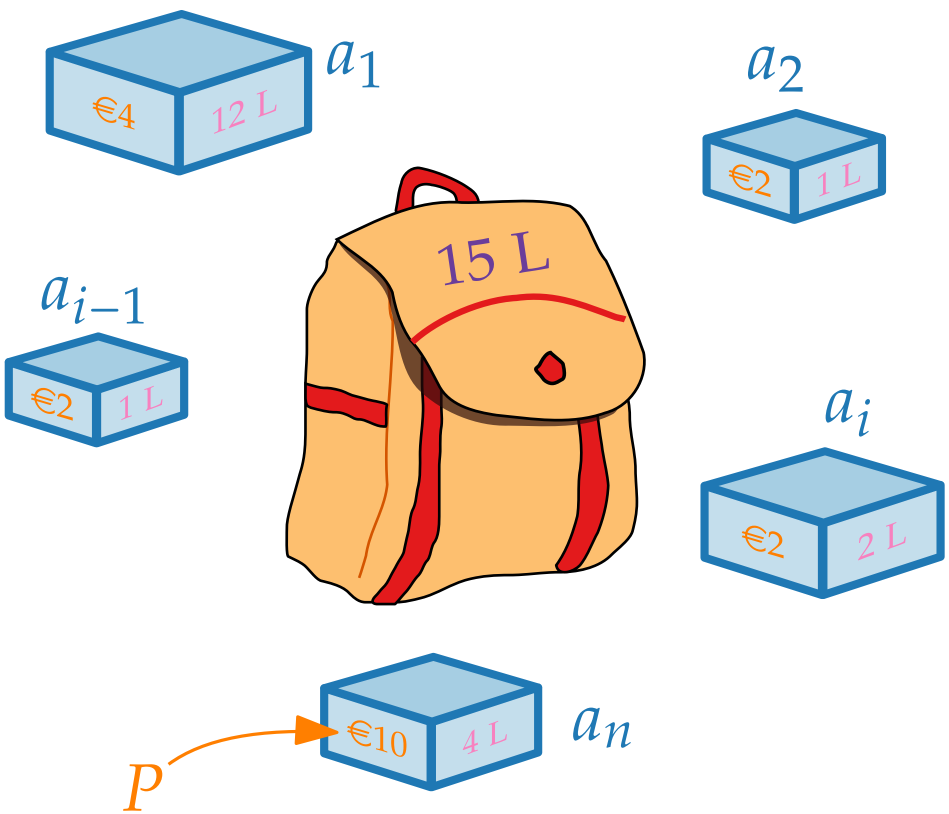
Pseudo-Polynomial Alg. for KNAPSACK

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



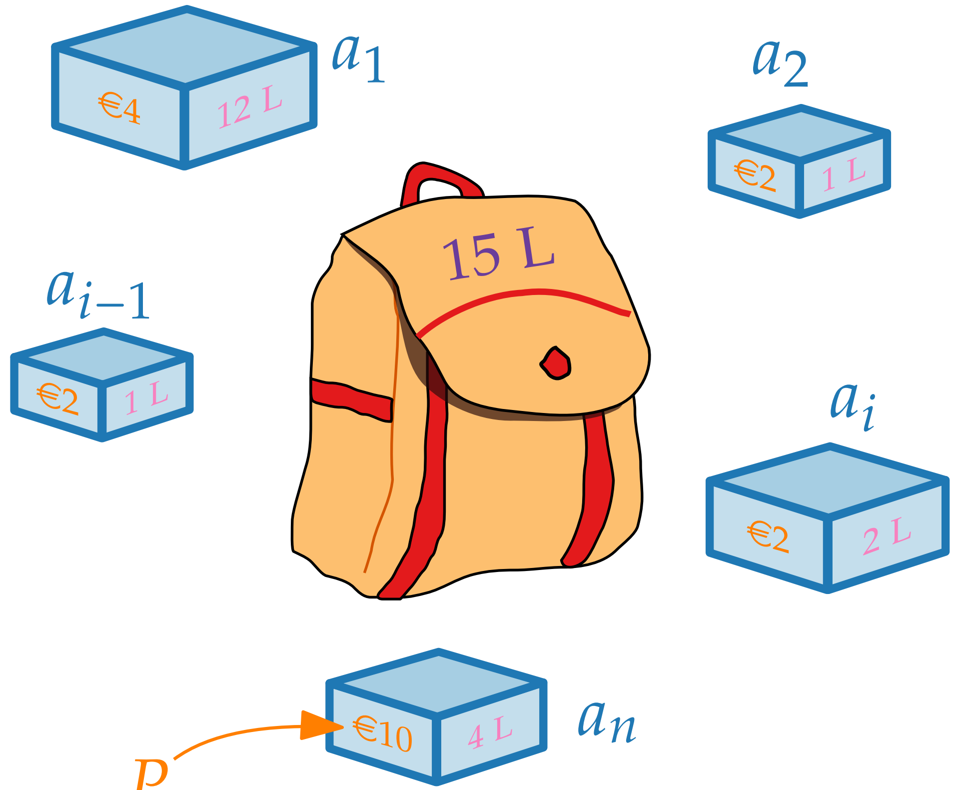
Pseudo-Polynomial Alg. for Knapsack

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



Pseudo-Polynomial Alg. for KNAPSACK

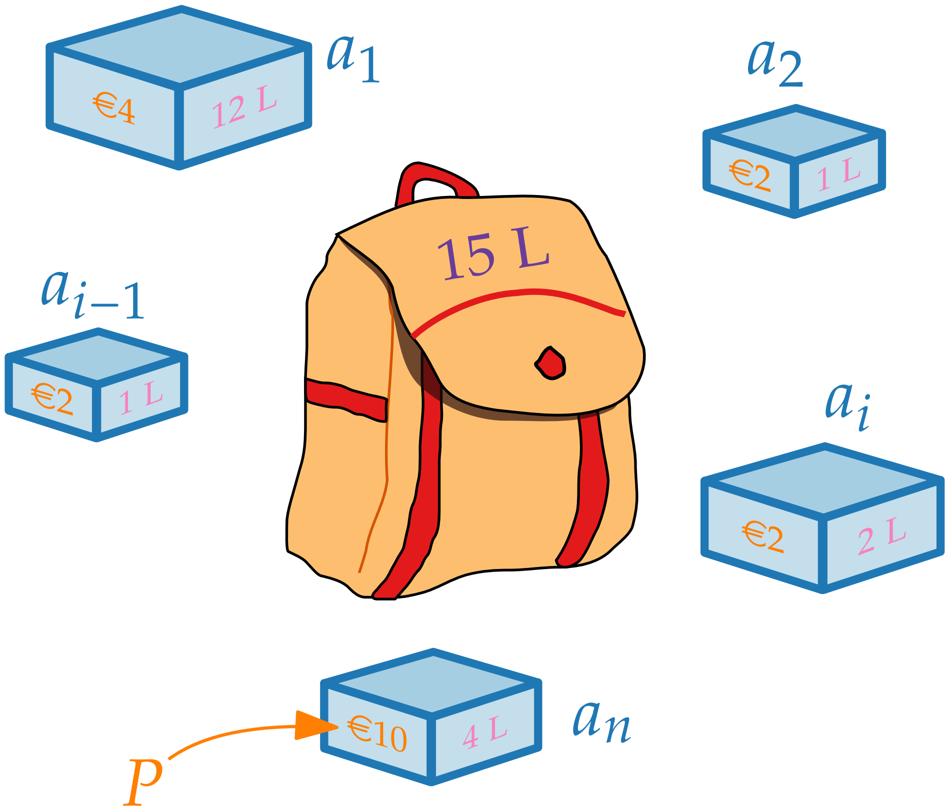
Let $P := \max_i \mathrm{profit}(a_i) \quad \Rightarrow \qquad P \leq \mathrm{OPT} \leq nP$



Pseudo-Polynomial Alg. for KNAPSACK

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

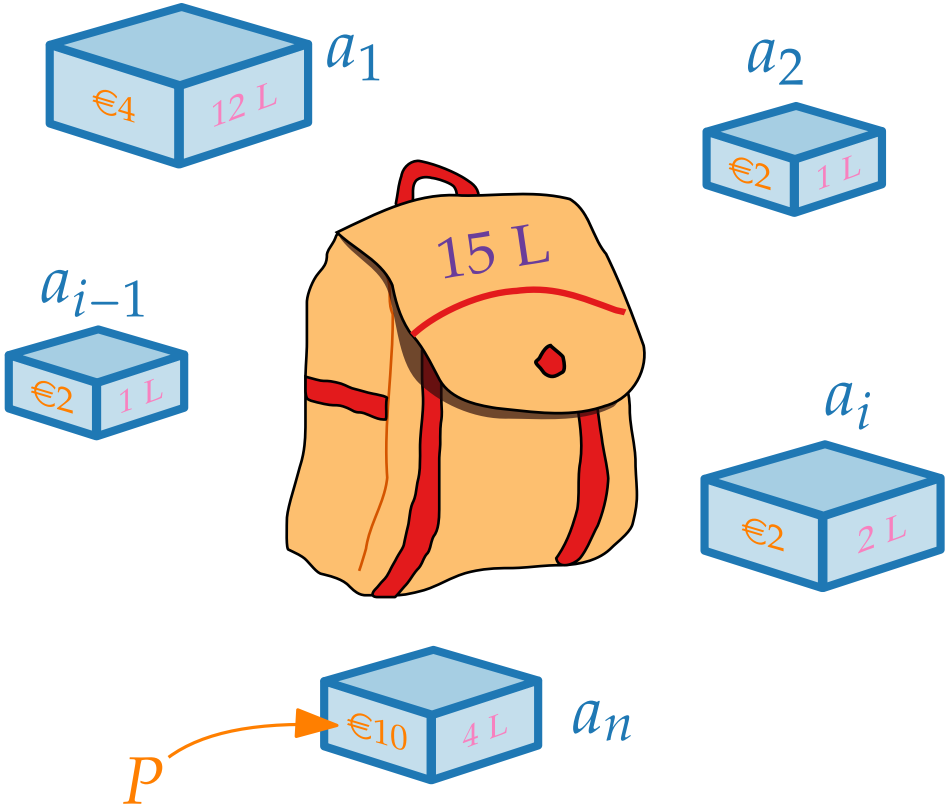
For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$,



Pseudo-Polynomial Alg. for Knapsack

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

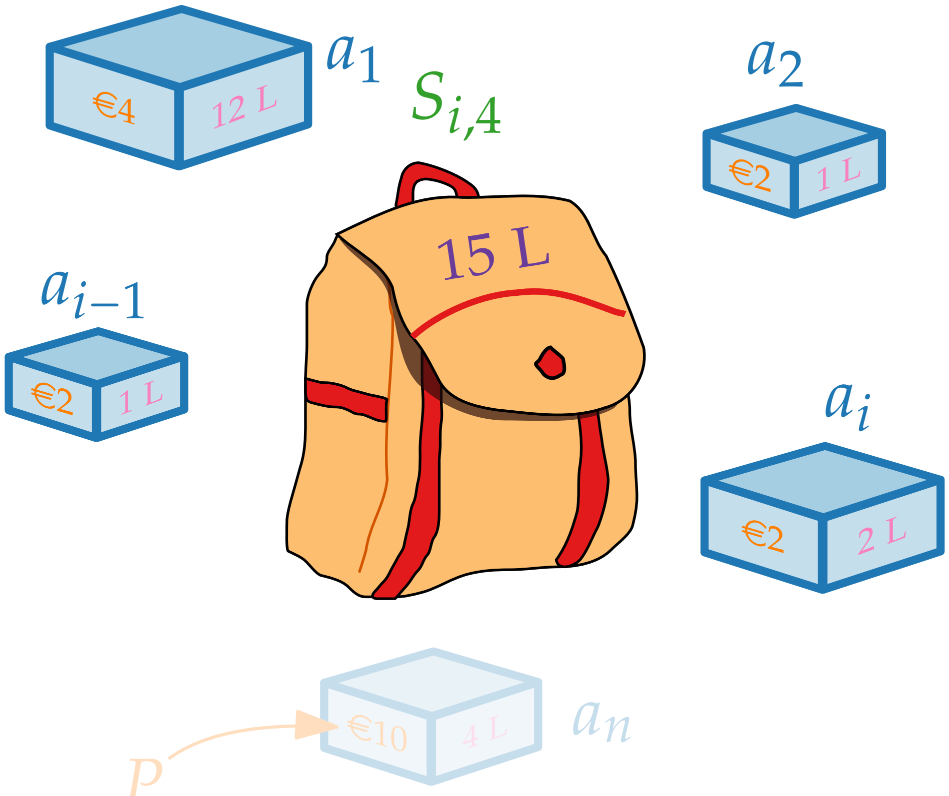
For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$, let $S_{i,p}$ be a subset of $\{a_1, \ldots, a_i\}$ whose total profit is precisely p



Pseudo-Polynomial Alg. for KNAPSACK

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

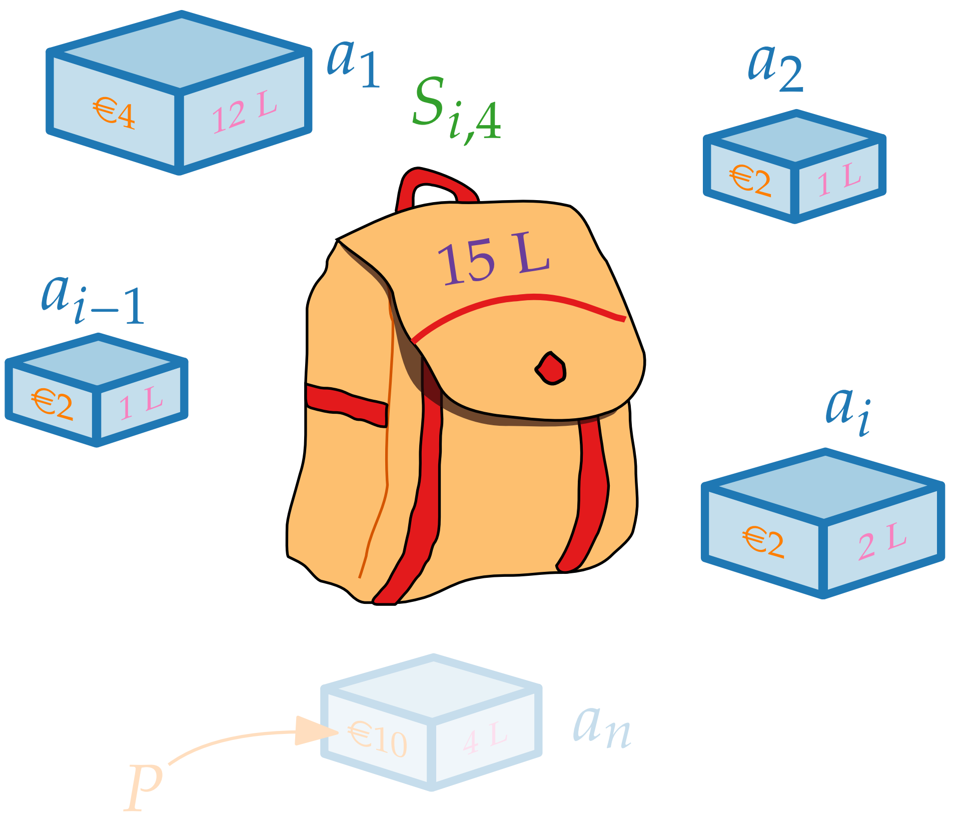
For every $i = 1, \dots, n$ and every $p \in \{1, \dots, nP\}$, let $S_{i,p}$ be a subset of $\{a_1, \dots, a_i\}$ whose total profit is precisely p



Pseudo-Polynomial Alg. for KNAPSACK

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

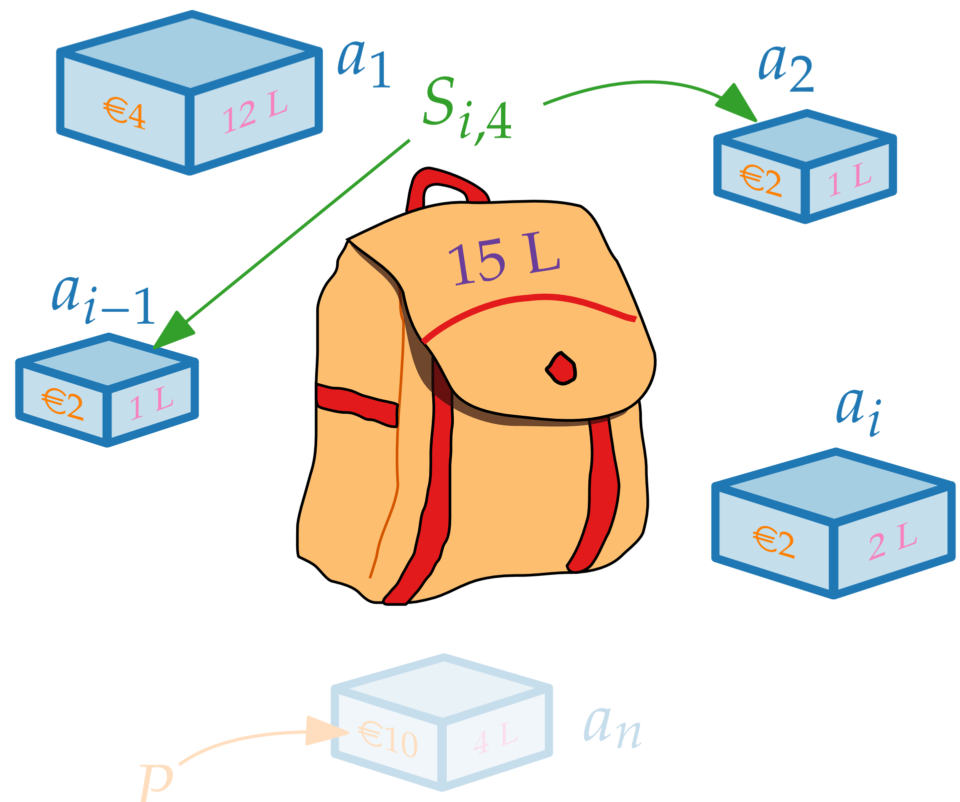
For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$, let $S_{i,p}$ be a subset of $\{a_1, \ldots, a_i\}$ whose total profit is precisely p and whose total size is minimum among all subsets with these properties.



Pseudo-Polynomial Alg. for KNAPSACK

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

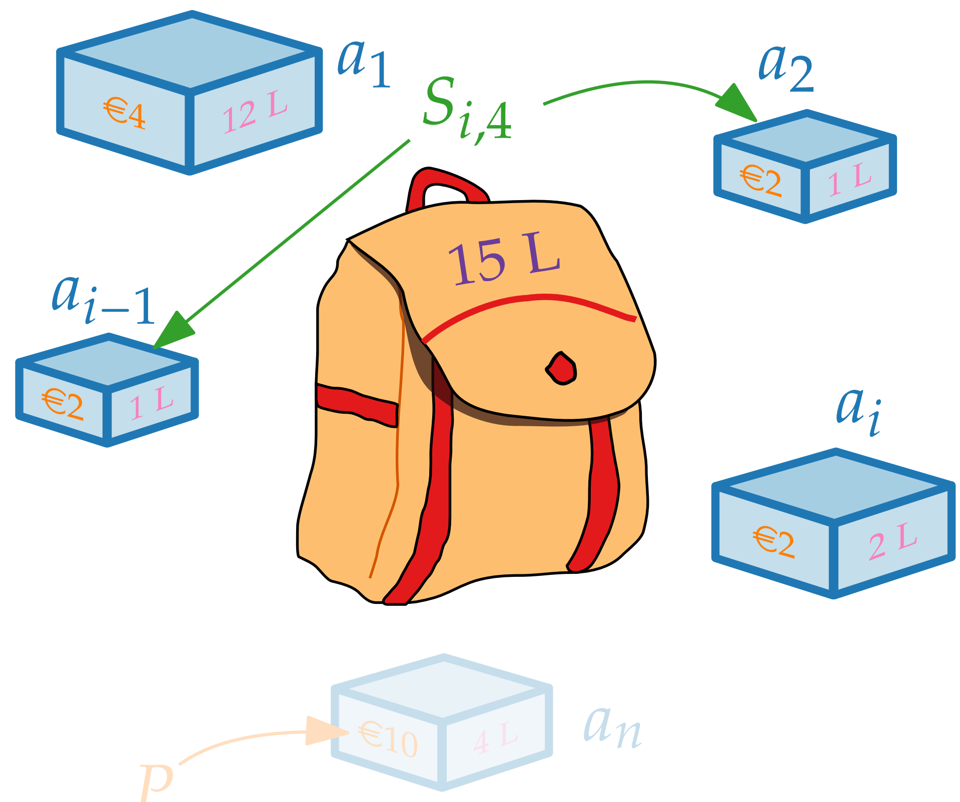
For every $i = 1, \ldots, n$ and every $p \in \{1, \ldots, nP\}$, let $S_{i,p}$ be a subset of $\{a_1, \ldots, a_i\}$ whose total profit is precisely p and whose total size is minimum among all subsets with these properties.



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. Such a set may not exist.

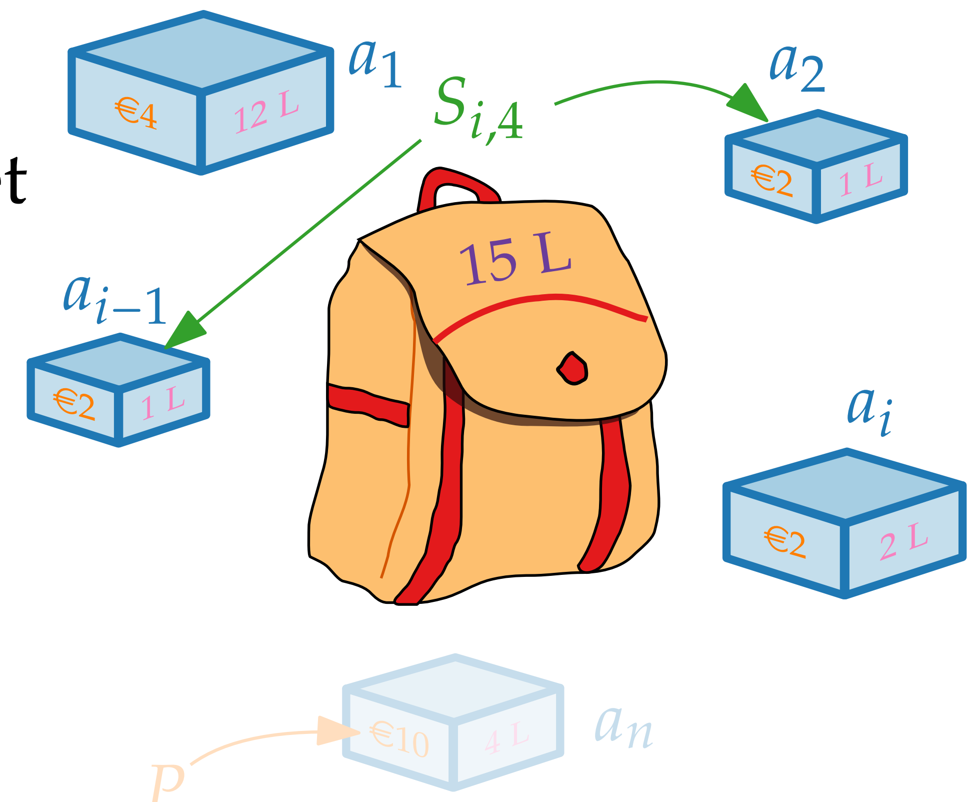


Pseudo-Polynomial Alg. for KNAPSACK

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

For every $i = 1, \dots, n$ and every $p \in \{1, \dots, nP\}$, let $S_{i,p}$ be a subset of $\{a_1, \dots, a_i\}$ 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 set $S_{i,p}$ (set $A[i, p] = \infty$ if no such set exists).

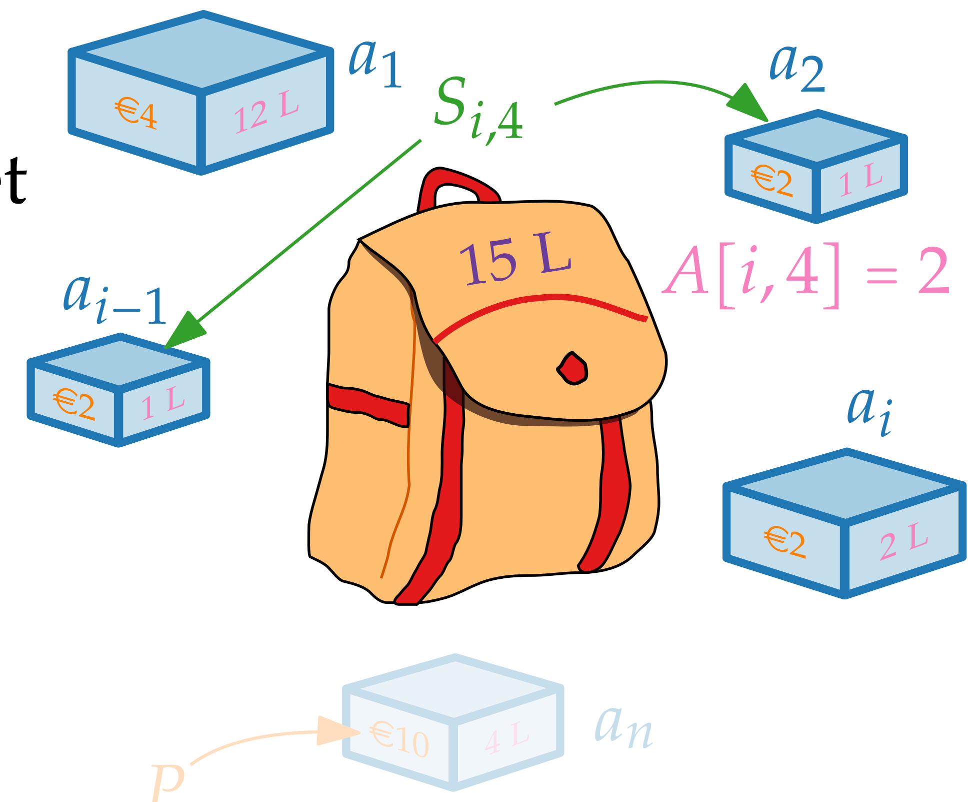


Pseudo-Polynomial Alg. for KNAPSACK

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

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

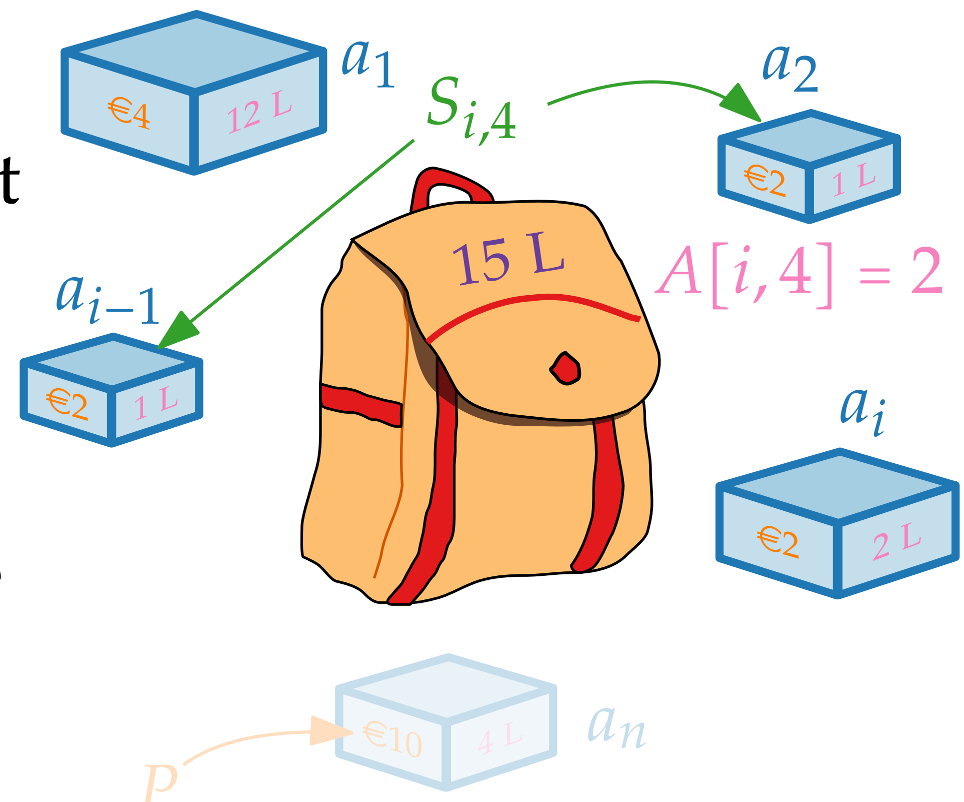
Let $A[i,p]$ be the total size of set $S_{i,p}$ (set $A[i,p] = \infty$ if no such set exists).



Pseudo-Polynomial Alg. for KNAPSACK

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

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 set $S_{i,p}$ (set $A[i,p] = \infty$ if no such set exists).

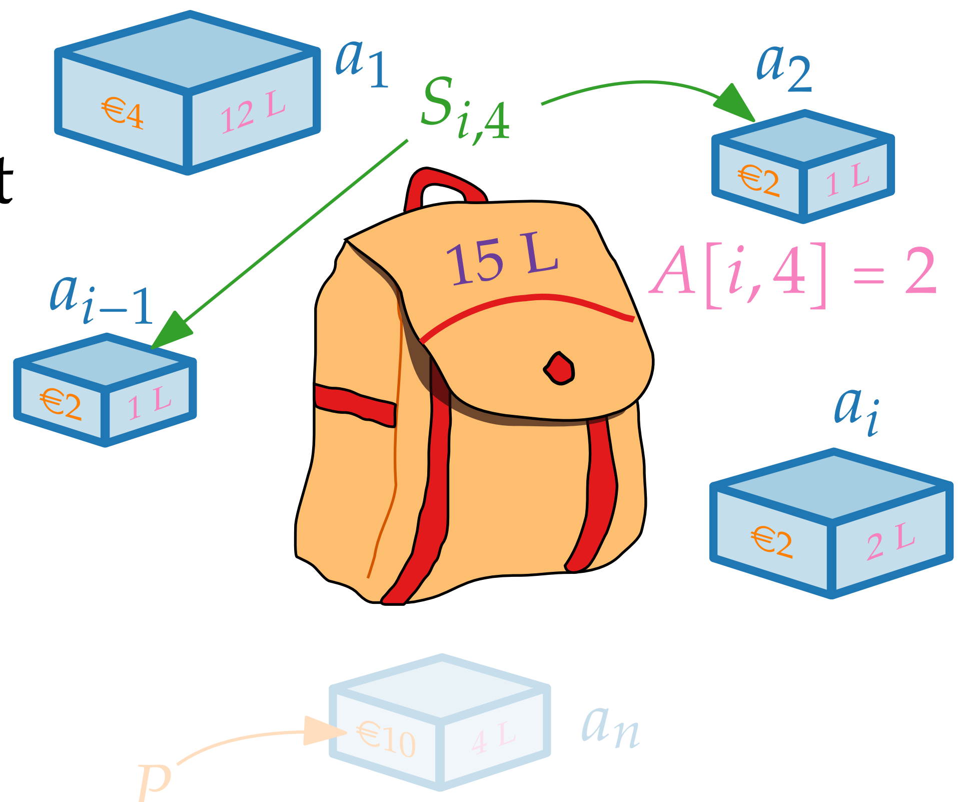
If all $A[i,p]$ are known, then we can compute

$\text{OPT} =$

Pseudo-Polynomial Alg. for KNAPSACK

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

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.

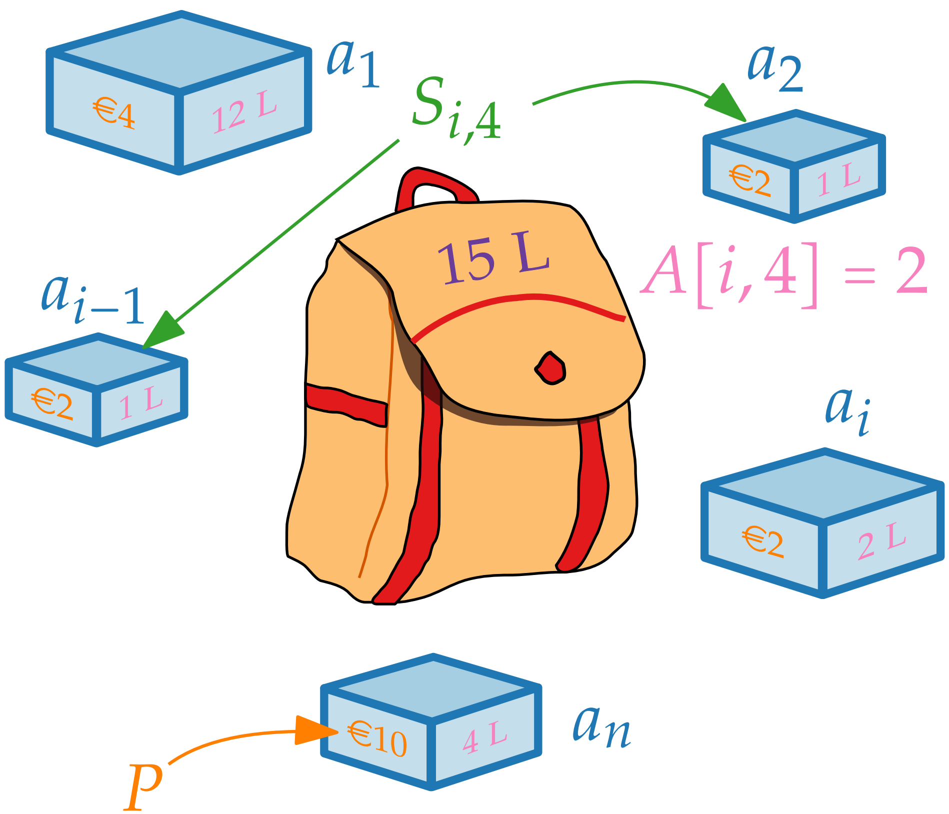


Let $A[i,p]$ be the total size of set $S_{i,p}$ (set $A[i,p] = \infty$ if no such set exists).

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

Pseudo-Polynomial Alg. for KNAPSACK

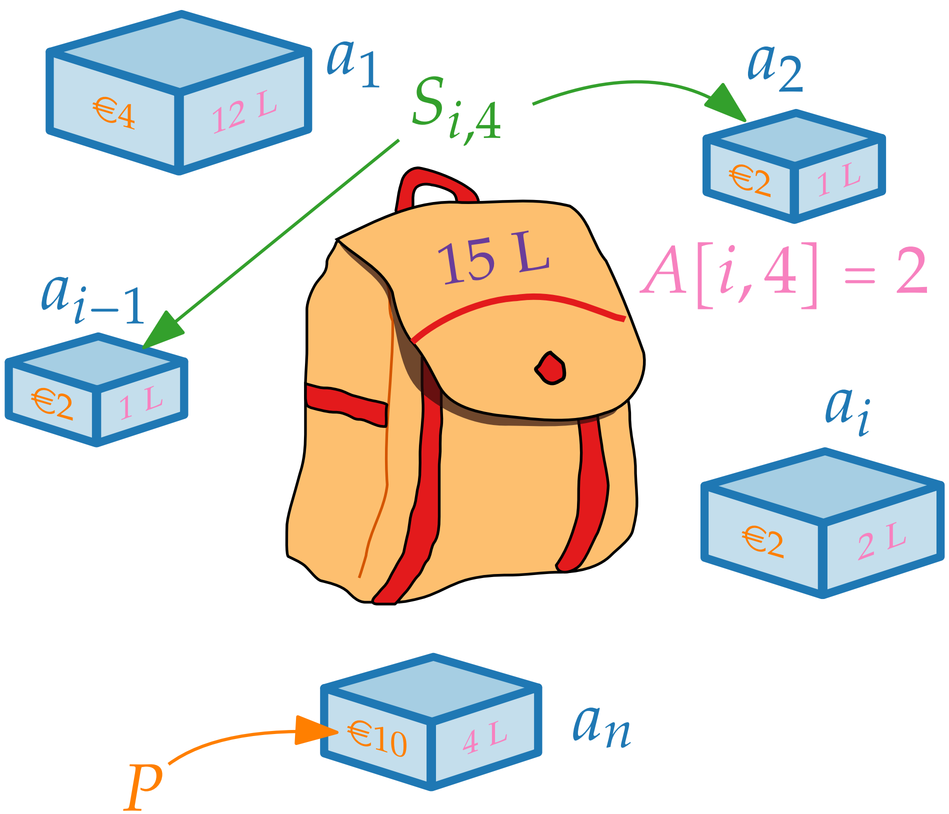
$A[1, p]$ can be computed for all $p \in \{0, \dots, nP\}$.



Pseudo-Polynomial Alg. for KNAPSACK

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

Set $A[i, p] := \infty$ for $p < 0$

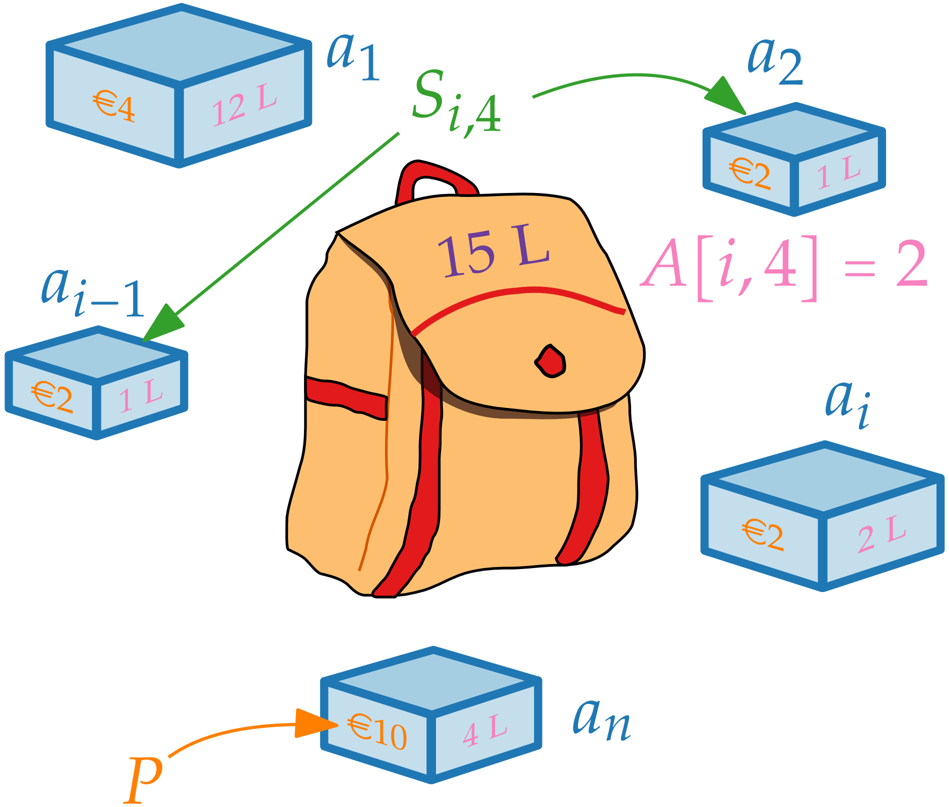


Pseudo-Polynomial Alg. for KNAPSACK

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

Set $A[i, p] := \infty$ for $p < 0$

$A[i + 1, p] =$

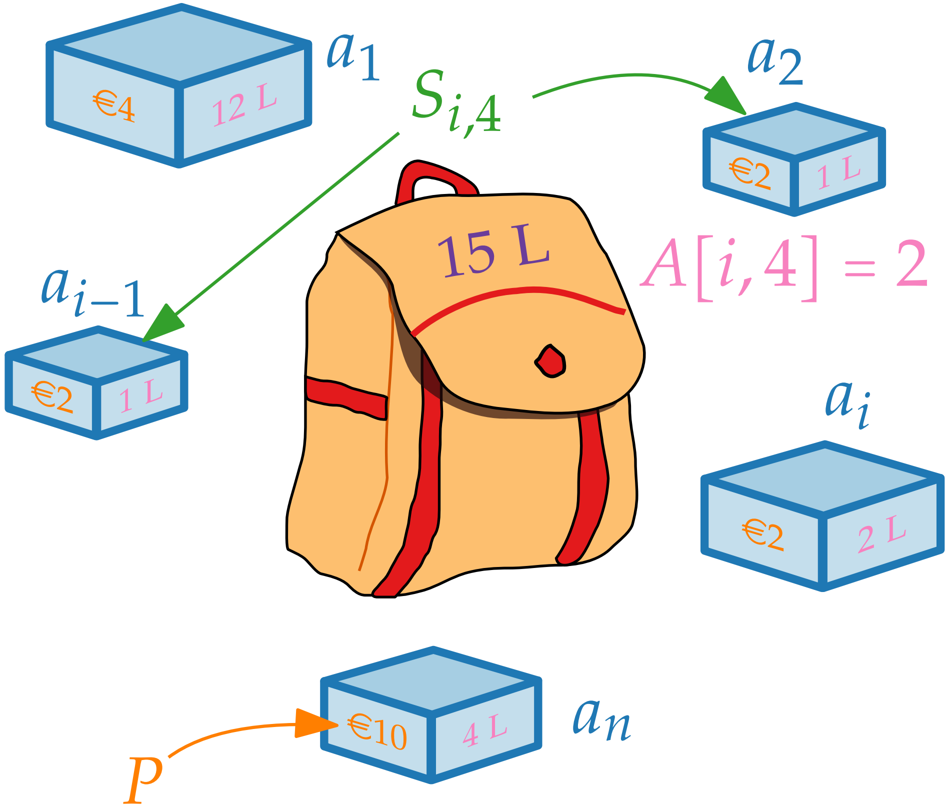


Pseudo-Polynomial Alg. for KNAPSACK

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

Set $A[i,p] := \infty$ for $p < 0$

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

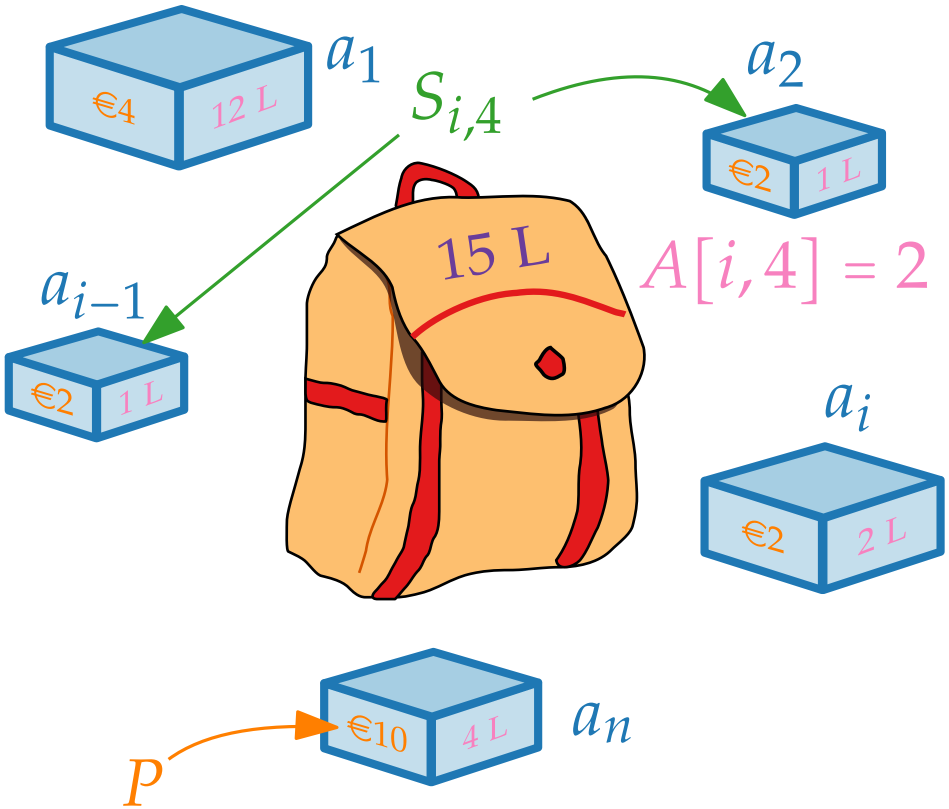


Pseudo-Polynomial Alg. for KNAPSACK

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

Set $A[i, p] := \infty$ for $p < 0$

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

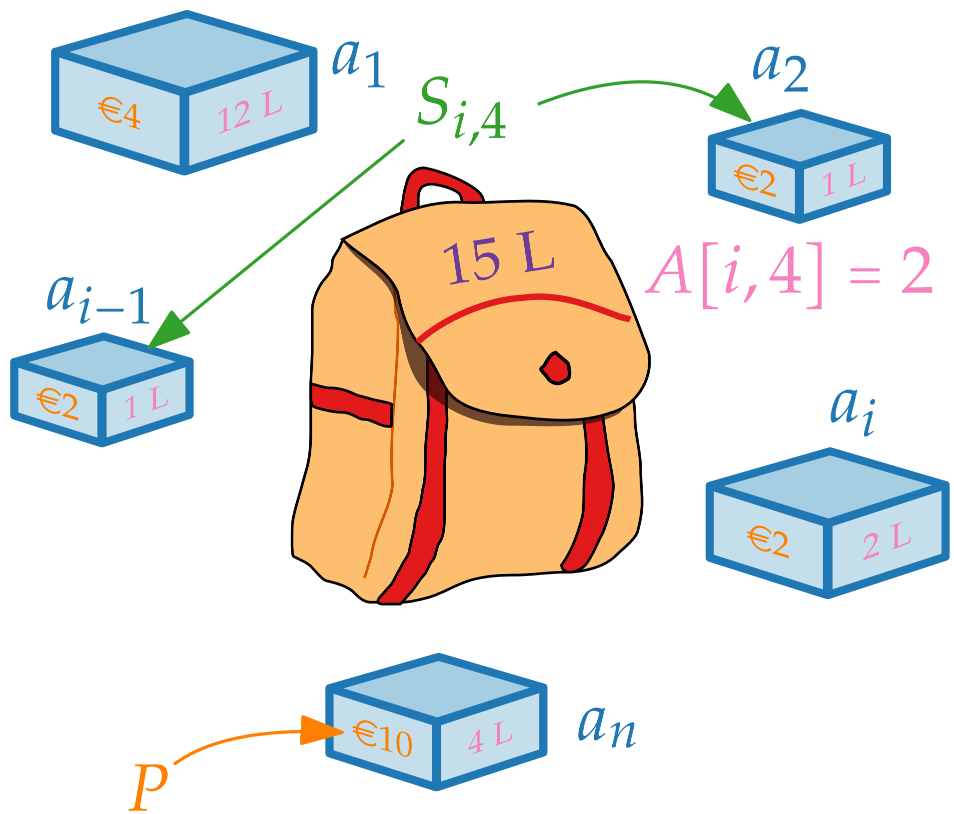


Pseudo-Polynomial Alg. for KNAPSACK

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

Set $A[i,p] := \infty$ for $p < 0$

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

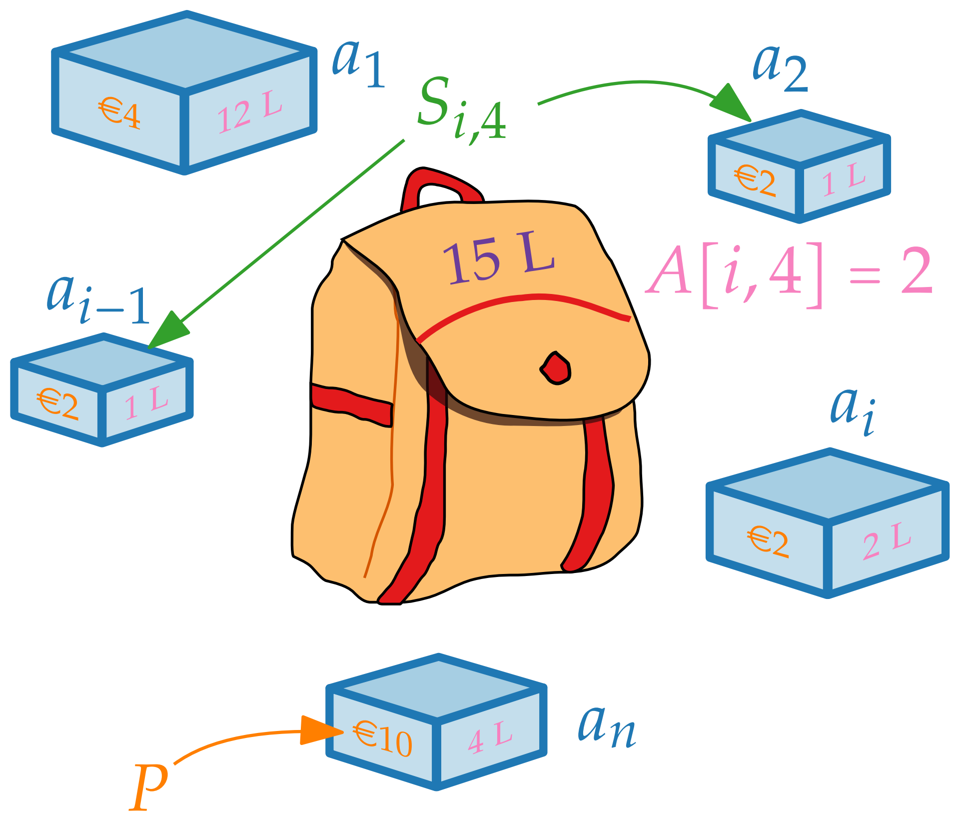


Pseudo-Polynomial Alg. for KNAPSACK

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

Set $A[i, p] := \infty$ for $p < 0$

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



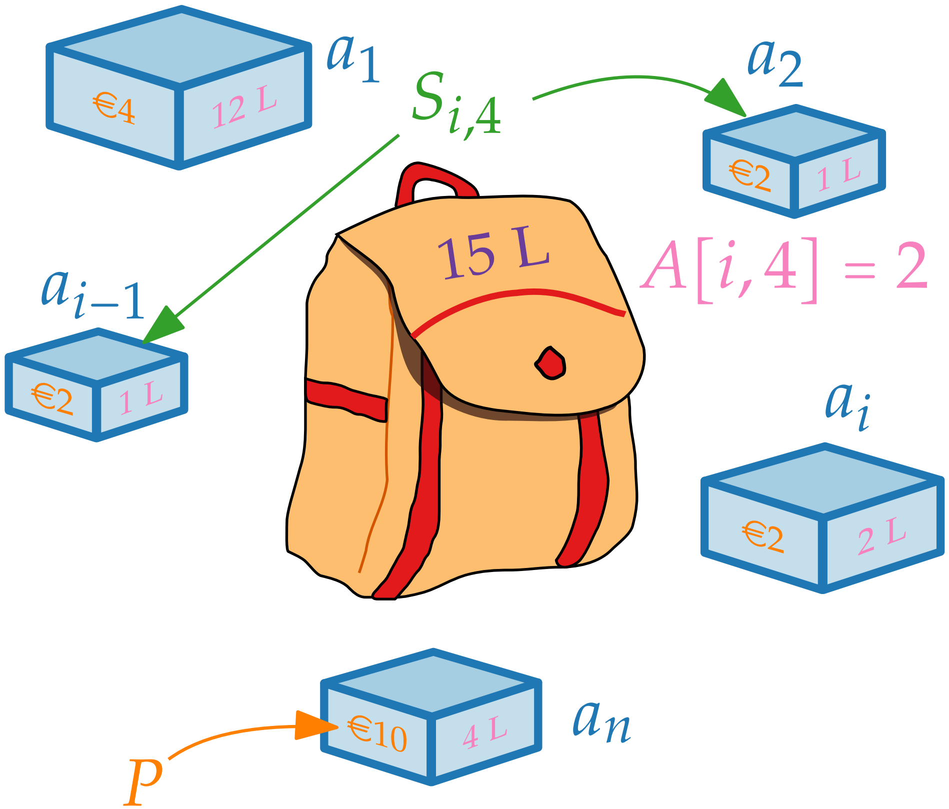
Pseudo-Polynomial Alg. for Knapsack

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

Set $A[i,p] := \infty$ for $p < 0$

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

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



Pseudo-Polynomial Alg. for KNAPSACK

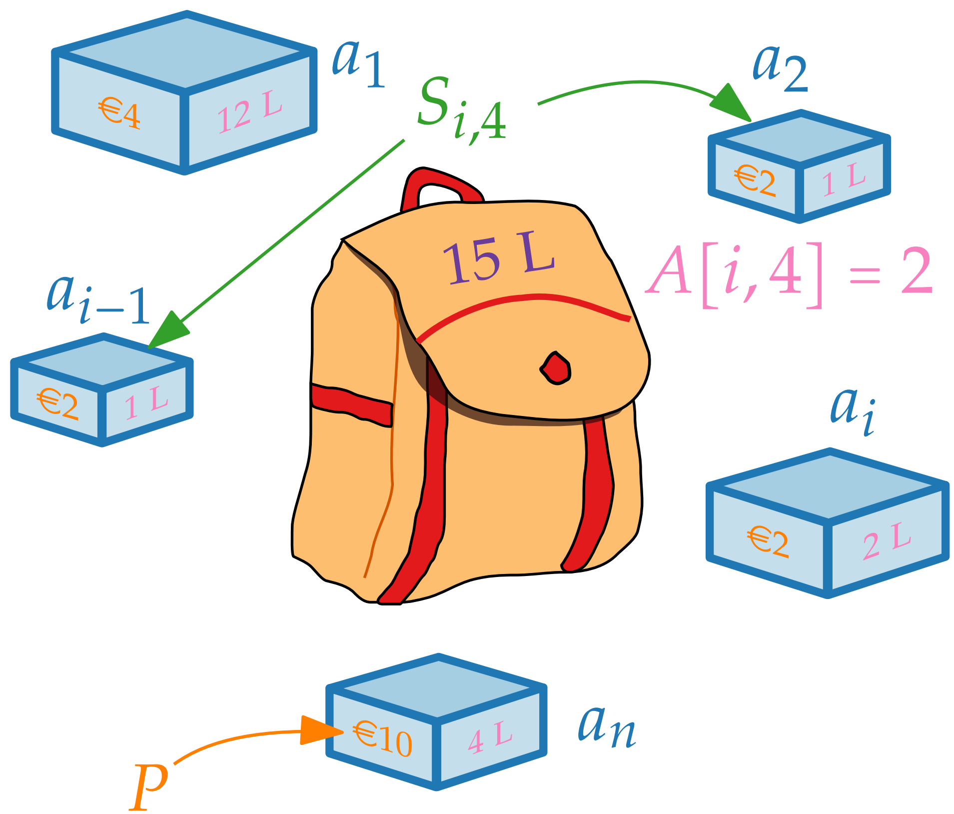
$A[1,p]$ can be computed for all $p \in \{0,\ldots,nP\}$.

Set $A[i,p] := \infty$ for $p < 0$

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

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

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



Pseudo-Polynomial Alg. for Knapsack

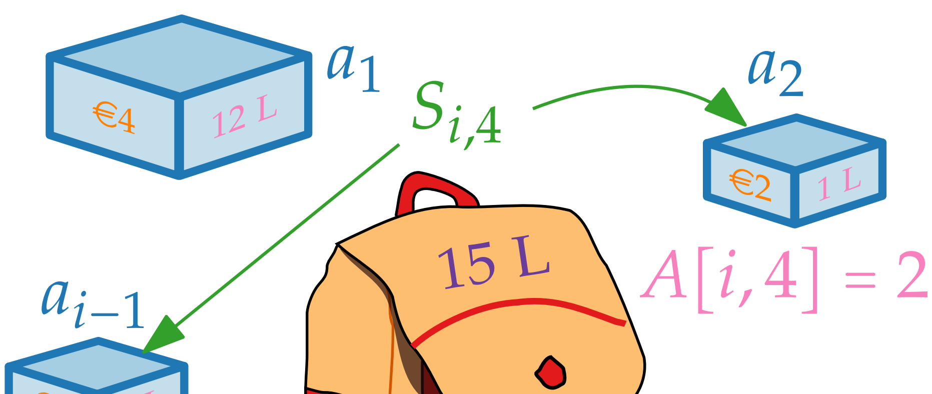
$A[1,p]$ can be computed for all $p \in \{0,\dots,nP\}$.

Set $A[i,p] := \infty$ for $p < 0$

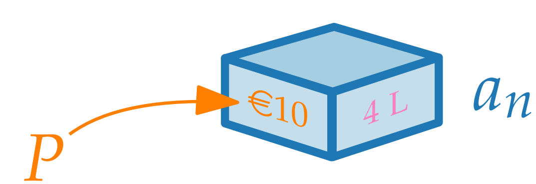
$A[i+1,p] = \min\{A[i,p], \mathrm{size}(a_{i+1}) + A[i, p - \mathrm{profit}(a_{i+1})]\}$

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

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



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



Pseudo-Polynomial Alg. for Knapsack

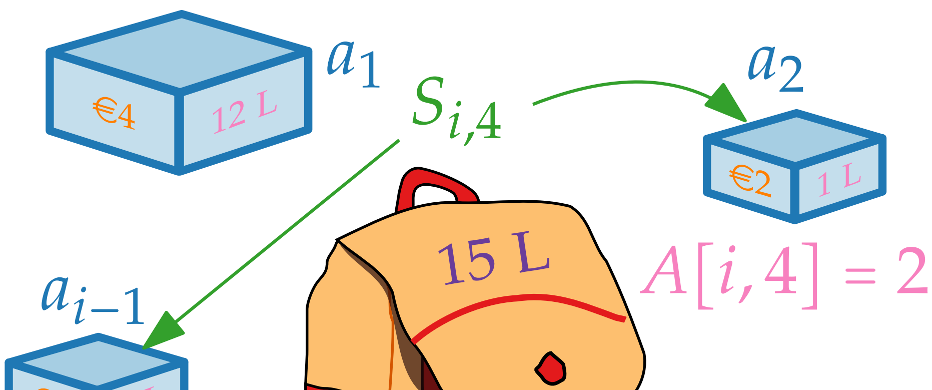
$A[1,p]$ can be computed for all $p \in \{0,\dots,nP\}$.

Set $A[i,p] := \infty$ for $p < 0$

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

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

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

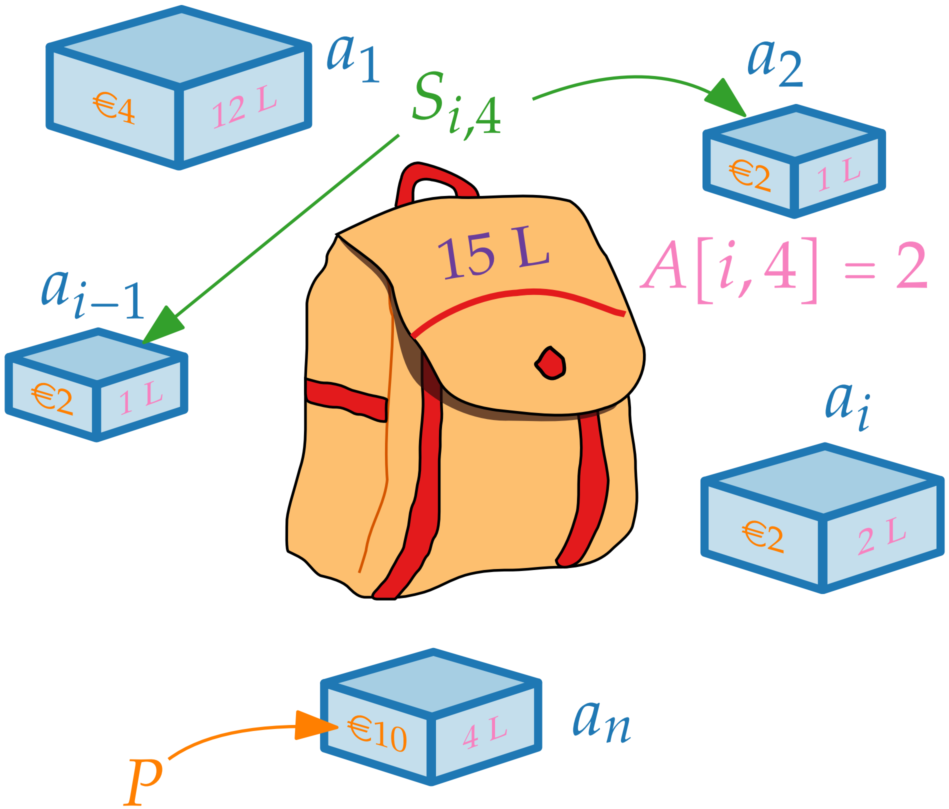


Theorem. Knapsack can be solved optimally in pseudo-polynomial time $O(n^2P)$.

Corollary. Knapsack is weakly NP-hard.

Pseudo-Polynomial Alg. for KNAPSACK

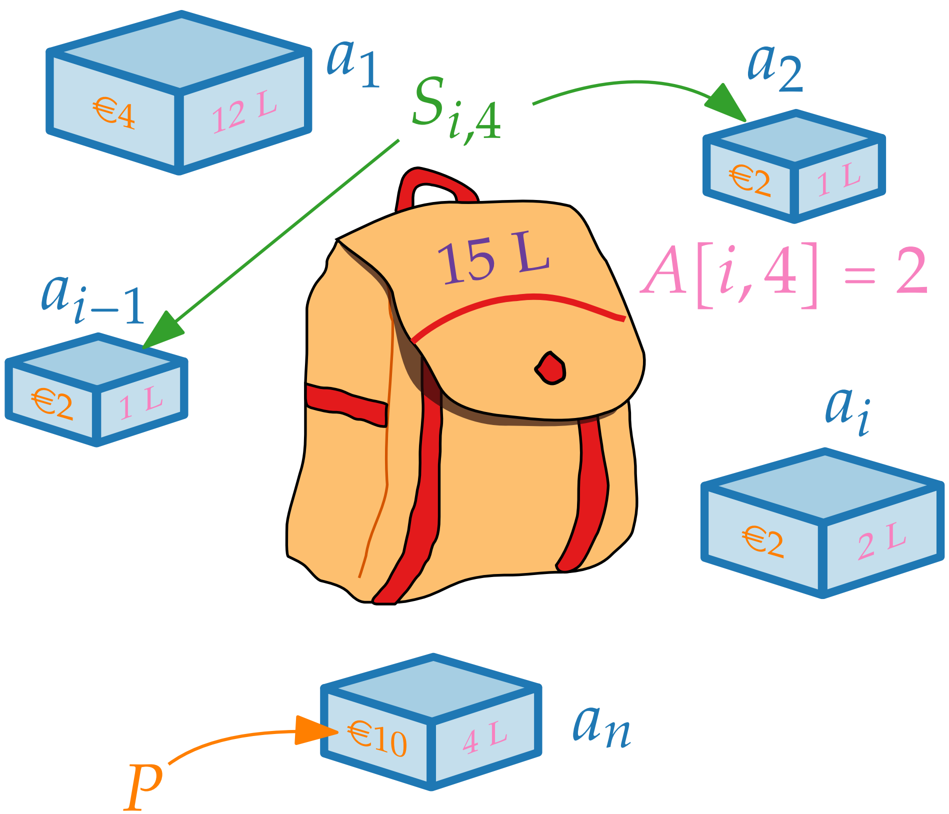
Theorem. KNAPSACK can be solved optimally in pseudo-polynomial time $O(n^2 P)$.



Pseudo-Polynomial Alg. for KNAPSACK

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

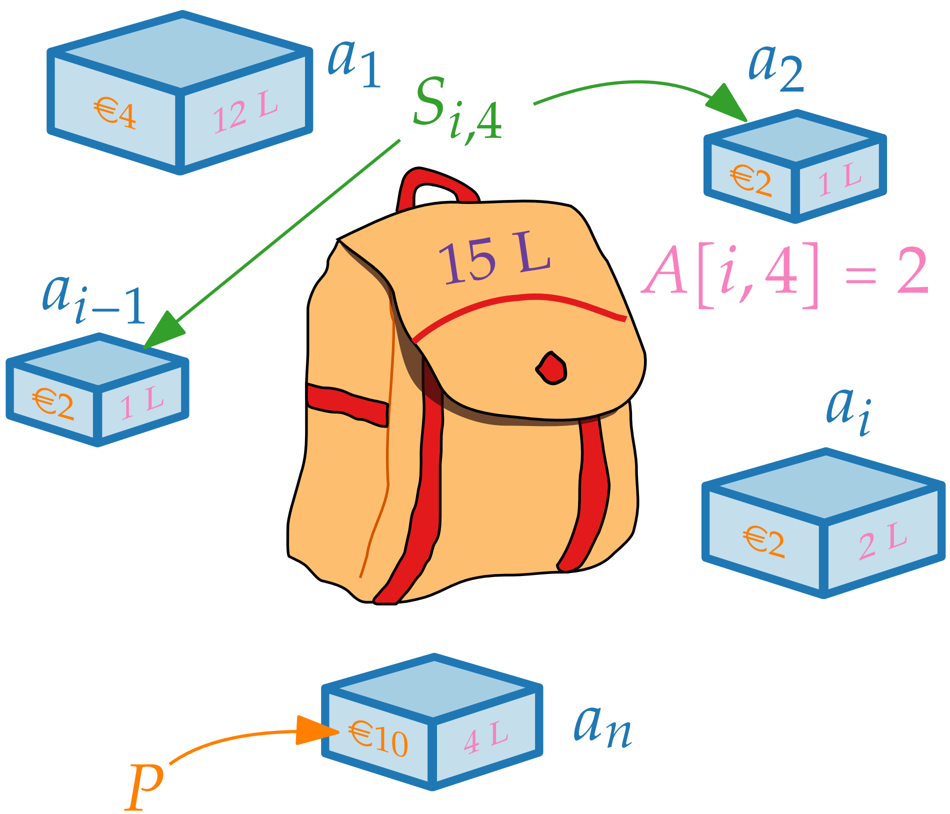
Examples. $P = n^5$



Pseudo-Polynomial Alg. for KNAPSACK

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

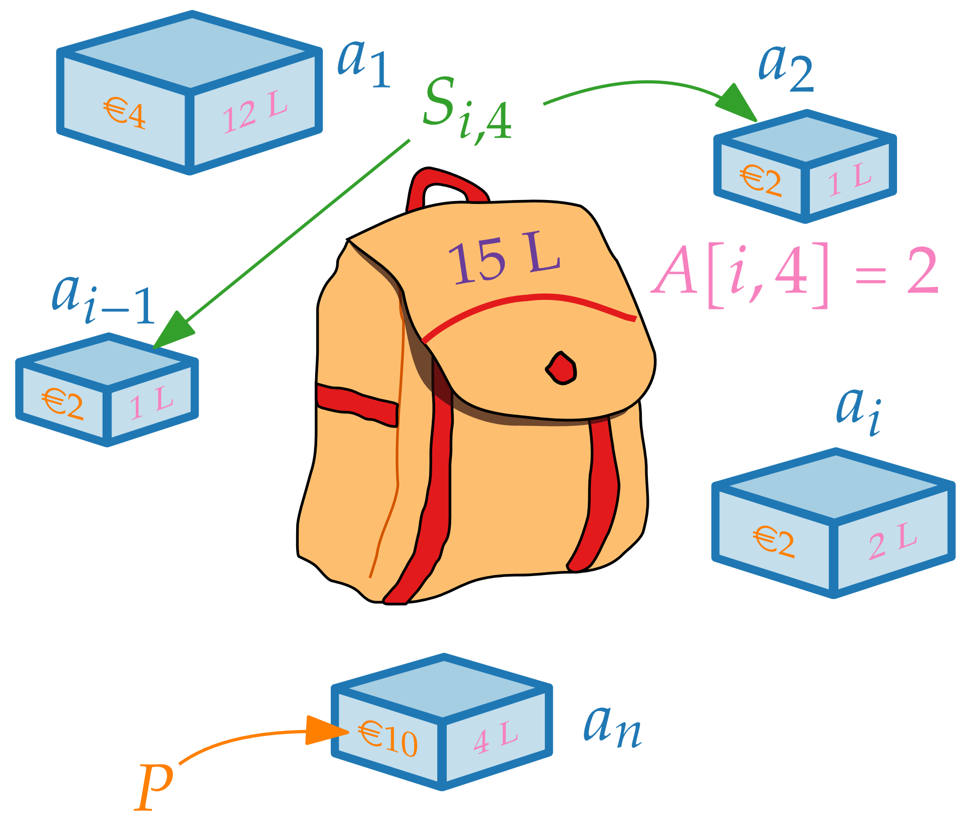
Examples. $P = n^5$ $\Rightarrow$ running time $O(n^7)$



Pseudo-Polynomial Alg. for KNAPSACK

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

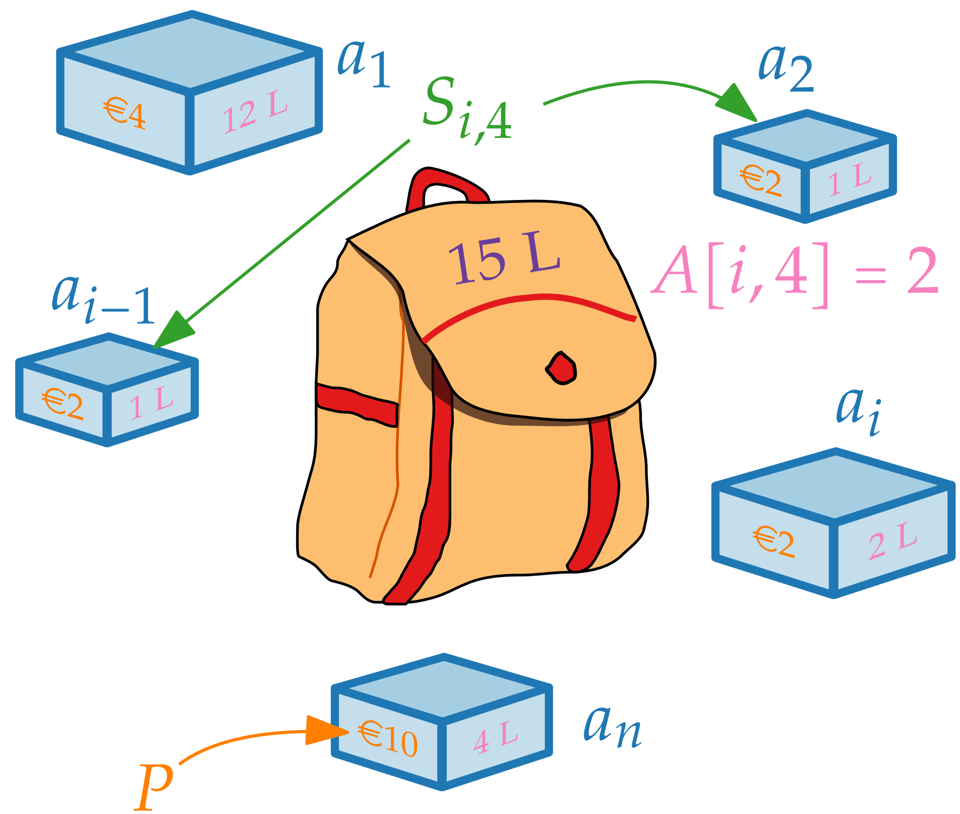
Examples. $P = n^5$ $\Rightarrow$ running time $O(n^7)$
(Bin.) instance size $|I| \geq n \log P = \Omega(n \log n)$



Pseudo-Polynomial Alg. for KNAPSACK

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

Examples. $P = n^5$ $\Rightarrow$ running time $O(n^7)$
(Bin.) instance size $|I| \geq n \log P = \Omega(n \log n)$
$\Rightarrow n \in O(|I| / \log |I|)$



Pseudo-Polynomial Alg. for KNAPSACK

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

Examples. $P = n^5$ $\Rightarrow$ running time $O(n^7)$

(Bin.) instance size $|I| \geq n \log P = \Omega(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$

$\Rightarrow$ running time $O(|I|^7 / \log^7 |I|) = \text{poly}(|I|)$

Pseudo-Polynomial Alg. for KNAPSACK

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

Examples. $P = n^5$ $\Rightarrow$ running time $O(n^7)$

(Bin.) instance size $|I| \geq n \log P = \Omega(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$

$\Rightarrow$ running time $O(|I|^7 / \log^7 |I|) = \text{poly}(|I|)$

$P = 2^n$

Pseudo-Polynomial Alg. for KNAPSACK

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

Examples. $P = n^5$ $\Rightarrow$ running time $O(n^7)$

(Bin.) instance size $|I| \geq n \log P = \Omega(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$

$\Rightarrow$ running time $O(|I|^7 / \log^7 |I|) = \text{poly}(|I|)$

$P = 2^n$ $\Rightarrow$ running time $O(n^2 2^n)$

Pseudo-Polynomial Alg. for Knapsack

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

Examples. $P = n^5$ $\Rightarrow$ running time $O(n^7)$
(Bin.) instance size $|I| \geq n \log P = \Omega(n \log n)$
$\Rightarrow n \in O(|I| / \log |I|)$
$\Rightarrow$ running time $O(|I|^7 / \log^7 |I|) = \mathrm{poly}(|I|)$

$P = 2^n$ $\Rightarrow$ running time $O(n^2 2^n)$
(Bin.) instance size $|I| \leq n \log P = O(n^2)$

Pseudo-Polynomial Alg. for KNAPSACK

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

Examples. $P = n^5$ $\Rightarrow$ running time $O(n^7)$

(Bin.) instance size $|I| \geq n \log P = \Omega(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$

$\Rightarrow$ running time $O(|I|^7 / \log^7 |I|) = \text{poly}(|I|)$

$P = 2^n$ $\Rightarrow$ running time $O(n^2 2^n)$

(Bin.) instance size $|I| \leq n \log P = O(n^2)$

$\Rightarrow$ running time $O(|I| 2^{\sqrt{|I|}}) \neq \text{poly}(|I|)$

Pseudo-Polynomial Alg. for KNAPSACK

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

Examples. $P = n^5$ $\Rightarrow$ running time $O(n^7)$

(Bin.) instance size $|I| \geq n \log P = \Omega(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$

$\Rightarrow$ running time $O(|I|^7 / \log^7 |I|) = \text{poly}(|I|)$

$P = 2^n$ $\Rightarrow$ running time $O(n^2 2^n)$

(Bin.) instance size $|I| \leq n \log P = O(n^2)$

$\Rightarrow$ running time $O(|I| 2^{\sqrt{|I|}}) \neq \text{poly}(|I|)$

(Un.) instance size $|I|_{\text{u}} \leq nP = O(n 2^n)$

Pseudo-Polynomial Alg. for KNAPSACK

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

Examples. $P = n^5$ $\Rightarrow$ running time $O(n^7)$

(Bin.) instance size $|I| \geq n \log P = \Omega(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$

$\Rightarrow$ running time $O(|I|^7 / \log^7 |I|) = \mathrm{poly}(|I|)$

$P = 2^n$ $\Rightarrow$ running time $O(n^2 2^n)$

(Bin.) instance size $|I| \leq n \log P = O(n^2)$

$\Rightarrow$ running time $O(|I| 2^{\sqrt{|I|}}) \neq \mathrm{poly}(|I|)$

(Un.) instance size $|I|_{\mathrm{u}} \leq nP = O(n 2^n)$

$\Rightarrow n \in O(\log |I|_{\mathrm{u}} - \log \log |I|_{\mathrm{u}})$

Pseudo-Polynomial Alg. for KNAPSACK

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

Examples.

$P = n^5$ $\Rightarrow$ running time $O(n^7)$

(Bin.) instance size $|I| \geq n \log P = \Omega(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$

$\Rightarrow$ running time $O(|I|^7 / \log^7 |I|) = \text{poly}(|I|)$

$P = 2^n$ $\Rightarrow$ running time $O(n^2 2^n)$

(Bin.) instance size $|I| \leq n \log P = O(n^2)$

$\Rightarrow$ running time $O(|I| 2^{\sqrt{|I|}}) \neq \text{poly}(|I|)$

(Un.) instance size $|I|_\text{u} \leq nP = O(n 2^n)$

$\Rightarrow n \in O(\log |I|_\text{u} - \log \log |I|_\text{u})$

$\Rightarrow$ running time $O(|I|_\text{u} \log |I|_\text{u}) = \text{poly}(|I|_\text{u})$

Pseudo-Polynomial Alg. for KNAPSACK

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

Examples. $P = n^5$ $\Rightarrow$ running time $O(n^7)$

(Bin.) instance size $|I| \geq n \log P = \Omega(n \log n)$

$\Rightarrow n \in O(|I| / \log |I|)$

$\Rightarrow$ running time $O(|I|^7 / \log^7 |I|) = \mathrm{poly}(|I|)$

$P = 2^n$ $\Rightarrow$ running time $O(n^2 2^n)$

(Bin.) instance size $|I| \leq n \log P = O(n^2)$

$\Rightarrow$ running time $O(|I| 2^{\sqrt{|I|}}) \neq \mathrm{poly}(|I|)$

(Un.) instance size $|I|_{\mathrm{u}} \leq nP = O(n 2^n)$

$\Rightarrow n \in O(\log |I|_{\mathrm{u}} - \log \log |I|_{\mathrm{u}})$

$\Rightarrow$ running time $O(|I|_{\mathrm{u}} \log |I|_{\mathrm{u}}) = \mathrm{poly}(|I|_{\mathrm{u}})$

Observe. Running time $O(n^2 P)$ poly in n if P poly in n.

Approximation Algorithms

Lecture 8:
Approximation Schemes and the Knapsack Problem

Part IV:
Approximation Schemes

Joachim Spoerhase

Winter 2021/22

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), 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 the following holds:

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called a **polynomial-time approximation scheme** (PTAS), 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 the following holds:

- $\text{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \text{OPT}$, if Π min problem,

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called a **polynomial-time approximation scheme** (PTAS), 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 the following holds:

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$, if Π min problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$, if Π max problem.

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called a **polynomial-time approximation scheme** (PTAS), 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 the following holds:

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$, if Π min problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$, if Π max problem.

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), 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 the following holds:

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$, if Π min problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$, if Π max problem.

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), 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 the following holds:

- $\mathrm{obj}_\Pi(I, s) \le (1 + \varepsilon) \cdot \mathrm{OPT}$, if Π min problem,
- $\mathrm{obj}_\Pi(I, s) \ge (1 - \varepsilon) \cdot \mathrm{OPT}$, if Π max problem.

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

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

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called a **polynomial-time approximation scheme** (PTAS), 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 the following holds:

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$, if Π min problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$, if Π max problem.

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

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called a **polynomial-time approximation scheme** (PTAS), 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 the following holds:

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$, if Π min problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$, if Π max problem.

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

Approximation Schemes

Let Π be an optimization problem. An algorithm $\mathcal{A}$ is called a **polynomial-time approximation scheme** (PTAS), 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 the following holds:

- $\mathrm{obj}_\Pi(I, s) \leq (1 + \varepsilon) \cdot \mathrm{OPT}$, if Π min problem,
- $\mathrm{obj}_\Pi(I, s) \geq (1 - \varepsilon) \cdot \mathrm{OPT}$, if Π max problem.

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

Approximation Algorithms

Lecture 8:
Approximation Schemes and the KNAPSACK Problem

Part V:
FPTAS for KNAPSACK

Joachim Spoerhase

Winter 2021/22

FPTAS for KNAPSACK via Scaling

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

FPTAS for Knapsack via Scaling

KnapsackScaling (I, ε)

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

FPTAS for Knapsack via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P / n$

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

FPTAS for Knapsack via Scaling

KnapsackScaling (I, ε)

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

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

FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

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

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

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

FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P / n$ // 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)...

FPTAS for Knapsack via Scaling

KnapsackScaling (I, ε)

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

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

FPTAS for Knapsack via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P / n$ // 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)...

FPTAS for Knapsack via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P/n$ // 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. $\text{profit}(S') \geq (1-\varepsilon) \cdot \text{OPT}$.

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

FPTAS for Knapsack via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P / n$ // 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 $\text{OPT} = \{o_1, \ldots, o_k\}$.

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

FPTAS for Knapsack via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P/n$ // 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 $\mathrm{OPT} = \{o_1, \ldots, o_k\}$.

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

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

FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P/n$ // 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 $\text{OPT} = \{o_1, \ldots, o_k\}$.

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

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

FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P/n$ // 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 $\text{OPT} = \{o_1, \ldots, o_k\}$.

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

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

FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P/n$ // 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 $\text{OPT} = \{o_1, \ldots, o_k\}$.

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

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

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

FPTAS for Knapsack via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P/n$ // 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 $\mathrm{OPT} = \{o_1, \dots, o_k\}$.

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

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

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

FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P/n$ // 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 $\text{OPT} = \{o_1, \ldots, o_k\}$.

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

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

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

FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P/n$ // 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 $\mathrm{OPT} = \{o_1, \ldots, o_k\}$.

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

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

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

FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P/n$ // 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 $\text{OPT} = \{o_1, \ldots, o_k\}$.

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

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

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)…

FPTAS for Knapsack via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P/n$ // 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 $\text{OPT} = \{o_1, \ldots, o_k\}$.

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

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

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)…

FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P/n$ // 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 $\text{OPT} = \{o_1, \ldots, o_k\}$.

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

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

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

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

FPTAS for Knapsack via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P/n$ // 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 $\text{OPT} = \{o_1, \ldots, o_k\}$.

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

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

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

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

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

FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P/n$ // 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 $\mathrm{OPT} = \{o_1, \dots, o_k\}$.

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

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

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)...

FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P / n$ // 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 $\text{OPT} = \{o_1, \ldots, o_k\}$.

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

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

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

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

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

FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P / n$ // 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 $\text{OPT} = \{o_1, \dots, o_k\}$.

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

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

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

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

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

FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P/n$ // 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 $\mathrm{OPT} = \{o_1, \dots, o_k\}$.

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

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

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)...

FPTAS for KNAPSACK via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P/n$ // 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 $\text{OPT} = \{o_1, \ldots, o_k\}$.

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

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

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 $O(\;,\;)$

FPTAS for Knapsack via Scaling

KnapsackScaling (I, ε)

$K \leftarrow \varepsilon P/n$ // 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 $\text{OPT} = \{o_1, \dots, o_k\}$.

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

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

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 $O(n^3/\varepsilon) = O\left(n^2 \cdot \frac{P}{\varepsilon P/n}\right)$.

Approximation Algorithms

Lecture 8: Approximation Schemes and the Knapsack Problem

Part VI: Connections

Joachim Spoerhase

Winter 2021/22

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,

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

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

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

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

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

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

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

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

$\Rightarrow$ 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 P = 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 P = 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 fulfils the restrictions above. If Π is strongly NP-hard, then there is no FPTAS for Π (unless P = NP).