Complexity Theory

NP-Complete Problems

Daniel Borchmann, Markus Krötzsch

Computational Logic

2015-11-24

Review

Further NP-complete Problems

Towards More NP-Complete Problems

Starting with SAT, one can readily show more problems $\mathcal{P}$ to be NP-complete, each time performing two steps:

(1) Show that $\mathcal{P} \in \mathrm{NP}$

(2) Find a known NP-complete problem $\mathcal{P}'$ and reduce $\mathcal{P}' \leq_p \mathcal{P}$

Thousands of problem have now been shown to be NP-complete. (See Garey and Johnson for an early survey)

In this course:

	$\leq_p$ CLIQUE		$\leq_p$ INDEPENDENT SET	
SAT	$\leq_p$ 3-SAT		$\leq_p$ DIR. HAMILTONIAN PATH	
	$\leq_p$ SUBSET SUM		$\leq_p$ KNAPSACK	

NP-Completeness of Directed Hamiltonian Path

Directed Hamiltonian Path

Input: A directed graph G.

Problem: Is there a directed path in G containing every vertex exactly once?

Theorem 9.1

Directed Hamiltonian Path *is* NP*-complete.*

NP-Completeness of Directed Hamiltonian Path

Directed Hamiltonian Path

Input: A directed graph G.

Problem: Is there a directed path in G containing every vertex exactly once?

Theorem 9.1

Directed Hamiltonian Path *is* NP*-complete.*

Proof.

- Directed Hamiltonian Path $\in \mathrm{NP}$:
  Take the path to be the certificate.

Digression: How to design reductions

Task: Show that problem $\mathcal{P}$ (DIR. HAMILTONIAN PATH) is NP-hard.

- Arguably, the most important part is to decide where to start from.

  That is, which problem to reduce to DIRECTED HAMILTONIAN PATH?

- Considerations:
  - Is there an NP-complete problem similar to $\mathcal{P}$?
    (for example, CLIQUE and INDEPENDENT SET)
  - It is not always beneficial to choose a problem of the same type

    (for example, reducing a graph problem to a graph problem)
    - For instance, CLIQUE, INDEPENDENT SET are “local” problems
      (is there a set of vertices inducing some structure)
    - Hamiltonian Path is a global problem
      (find a structure – the Hamiltonian path – containing all vertices)

Digression: How to design reductions

Task: Show that problem $\mathcal{P}$ (DIR. HAMILTONIAN PATH) is NP-hard.

- Arguably, the most important part is to decide where to start from.

  That is, which problem to reduce to DIRECTED HAMILTONIAN PATH?

- Considerations:
  - Is there an NP-complete problem similar to $\mathcal{P}$?
    (for example, CLIQUE and INDEPENDENT SET)
  - It is not always beneficial to choose a problem of the same type

    (for example, reducing a graph problem to a graph problem)
    - For instance, CLIQUE, INDEPENDENT SET are "local" problems
      (is there a set of vertices inducing some structure)
    - Hamiltonian Path is a global problem
      (find a structure – the Hamiltonian path – containing all vertices)

- How to design the reduction:
  - Does your problem come from an optimisation problem?
    If so: a maximisation problem? a minimisation problem?
  - Learn from examples, have good ideas.

NP-Completeness of Directed Hamiltonian Path

Directed Hamiltonian Path

Input: A directed graph G.

Problem: Is there a directed path in G containing every vertex exactly once?

Theorem 9.1

Directed Hamiltonian Path *is* NP*-complete.*

Proof.

- Directed Hamiltonian Path $\in \mathrm{NP}$:
  Take the path to be the certificate.

NP-Completeness of Directed Hamiltonian Path

Directed Hamiltonian Path

Input: A directed graph G.

Problem: Is there a directed path in G containing every vertex exactly once?

Theorem 9.1

Directed Hamiltonian Path *is* NP*-complete.*

Proof.

- Directed Hamiltonian Path $\in \mathrm{NP}$:
  Take the path to be the certificate.
- Directed Hamiltonian Path is NP-hard:
  3-Sat $\leq_p$ Directed Hamiltonian Path

NP-Completeness of Directed Hamiltonian Path

Proof idea: (see blackboard for details)

Let $\varphi := \bigwedge_{i=1}^{k} C_i$ and $C_i := (L_{i,1} \vee L_{i,2} \vee L_{i,3})$

- For each variable X occurring in φ, we construct a directed graph ("gadget") that allows only two Hamiltonian paths: "true" and "false"
- Gadgets for each variable are "chained" in a directed fashion, so that all variables must be assigned one value
- Clauses are represented by vertices that are connected to the gadgets in such a way that they can only be visited on a Hamiltonian path that corresponds to an assignment where they are true

Details are also given in [Sipser, Theorem 7.46].

Example 9.2 (see blackboard)

$\varphi := C_1 \wedge C_2$ where $C_1 := (X \vee \neg Y \vee Z)$ and $C_2 := (\neg X \vee Y \vee \neg Z)$

Towards More NP-Complete Problems

Starting with SAT, one can readily show more problems $\mathcal{P}$ to be NP-complete, each time performing two steps:

(1) Show that $\mathcal{P} \in \mathrm{NP}$

(2) Find a known NP-complete problem $\mathcal{P}'$ and reduce $\mathcal{P}' \leq_p \mathcal{P}$

Thousands of problem have now been shown to be NP-complete.
(See Garey and Johnson for an early survey)

In this course:

	$\leq_p$	CLIQUE	$\leq_p$	INDEPENDENT SET
SAT	$\leq_p$	3-SAT	$\leq_p$	DIR. HAMILTONIAN PATH
	$\leq_p$	SUBSET SUM	$\leq_p$	KNAPSACK

NP-Completeness of Subset Sum

Subset Sum

Input: A collection of positive integers $S = \{a_1, \ldots, a_k\}$ and a target integer t.

Problem: Is there a subset $T \subseteq S$ such that $\sum_{a_i \in T} a_i = t$?

Theorem 9.3

Subset Sum *is* NP*-complete.*

Proof.

- Subset Sum $\in \mathrm{NP}$: Take T to be the certificate.
- Subset Sum is NP-hard: Sat $\leq_p$ Subset Sum

Example

$$(X_1 \vee X_2 \vee X_3) \wedge (\neg X_1 \vee \neg X_4) \wedge (X_4 \vee X_5 \vee \neg X_2 \vee \neg X_3)$$

		X_1	X_2	X_3	X_4	X_5	C_1	C_2	C_3
t_1	=	1	0	0	0	0	1	0	0
f_1	=	1	0	0	0	0	0	1	0
t_2	=		1	0	0	0	1	0	0
f_2	=		1	0	0	0	0	0	1
t_3	=			1	0	0	1	0	0
f_3	=			1	0	0	0	0	1
t_4	=				1	0	0	0	1
f_4	=				1	0	0	1	0
t_5	=					1	0	0	1
f_5	=					1	0	0	0
$m_{1,1}$	=						1	0	0
$m_{1,2}$	=						1	0	0
$m_{2,1}$	=						0	1	0
$m_{3,1}$	=						0	0	1
$m_{3,2}$	=						0	0	1
$m_{3,3}$	=						0	0	1
t	=	1	1	1	1	1	3	2	4

SAT $\leq_p$ SUBSET SUM

Given: $\varphi := C_1 \wedge \cdots \wedge C_k$ in conjunctive normal form.

(w.l.o.g. at most 9 literals per clause)

Let $X_1, \ldots, X_n$ be the variables in φ. For each X_i let

$t_i := a_1 \ldots a_n c_1 \ldots c_k$ where $a_j := \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$ and $c_j := \begin{cases} 1 & X_i \text{ occurs in } C_j \\ 0 & \text{otherwise} \end{cases}$

$f_i := a_1 \ldots a_n c_1 \ldots c_k$ where $a_j := \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$ and $c_j := \begin{cases} 1 & \neg X_i \text{ occurs in } C_j \\ 0 & \text{otherwise} \end{cases}$

Example

$$(X_1 \vee X_2 \vee X_3) \wedge (\neg X_1 \vee \neg X_4) \wedge (X_4 \vee X_5 \vee \neg X_2 \vee \neg X_3)$$

		X_1	X_2	X_3	X_4	X_5	C_1	C_2	C_3
t_1	=	1	0	0	0	0	1	0	0
f_1	=	1	0	0	0	0	0	1	0
t_2	=		1	0	0	0	1	0	0
f_2	=		1	0	0	0	0	0	1
t_3	=			1	0	0	1	0	0
f_3	=			1	0	0	0	0	1
t_4	=				1	0	0	0	1
f_4	=				1	0	0	1	0
t_5	=					1	0	0	1
f_5	=					1	0	0	0
$m_{1,1}$	=						1	0	0
$m_{1,2}$	=						1	0	0
$m_{2,1}$	=						0	1	0
$m_{3,1}$	=						0	0	1
$m_{3,2}$	=						0	0	1
$m_{3,3}$	=						0	0	1
t	=	1	1	1	1	1	3	2	4

SAT $\leq_p$ SUBSET SUM

Further, for each clause C_i take $r := |C_i| - 1$ integers $m_{i,1}, \ldots, m_{i,r}$

where $m_{i,j} := c_i \ldots c_k$ with $c_\ell := \begin{cases} 1 & \ell = i \\ 0 & \ell \neq i \end{cases}$

Definition of S: Let

$$S := \{t_i, f_i \mid 1 \leq i \leq n\} \cup \{m_{i,j} \mid 1 \leq i \leq k, \quad 1 \leq j \leq |C_i| - 1\}$$

Target: Finally, choose as target

$$t := a_1 \ldots a_n c_1 \ldots c_k \text{ where } a_i := 1 \text{ and } c_i := |C_i|$$

Claim: There is $T \subseteq S$ with $\sum_{a_i \in T} a_i = t$ iff φ is satisfiable.

Example

$$(X_1 \vee X_2 \vee X_3) \wedge (\neg X_1 \vee \neg X_4) \wedge (X_4 \vee X_5 \vee \neg X_2 \vee \neg X_3)$$

		X_1	X_2	X_3	X_4	X_5	C_1	C_2	C_3
t_1	=	1	0	0	0	0	1	0	0
f_1	=	1	0	0	0	0	0	1	0
t_2	=		1	0	0	0	1	0	0
f_2	=		1	0	0	0	0	0	1
t_3	=			1	0	0	1	0	0
f_3	=			1	0	0	0	0	1
t_4	=				1	0	0	0	1
f_4	=				1	0	0	1	0
t_5	=					1	0	0	1
f_5	=					1	0	0	0
$m_{1,1}$	=						1	0	0
$m_{1,2}$	=						1	0	0
$m_{2,1}$	=						0	1	0
$m_{3,1}$	=						0	0	1
$m_{3,2}$	=						0	0	1
$m_{3,3}$	=						0	0	1
t	=	1	1	1	1	1	3	2	4

NP-Completeness of SubsetSum

Let $\varphi := \bigwedge C_i$ $\quad$ C_i: clauses

Show: If φ is satisfiable, then there is $T \subseteq S$ with $\sum_{s \in T} s = t$.

Let β be a satisfying assigment for φ

$$\text{Set} \quad T_1 := \begin{array}{ll} \{t_i \mid \beta(X_i) = 1 & 1 \leq i \leq m\} \cup \\ \{f_i \mid \beta(X_i) = 0 & 1 \leq i \leq m\} \end{array}$$

Further, for each clause C_i let r_i be the number of satisfied literals in C_i (with resp. to β).

Set $T_2 := \{m_{i,j} \mid 1 \leq i \leq k, \quad 1 \leq j \leq |C_i| - r_i\}$

and define $T := T_1 \cup T_2$.

It follows: $\sum_{s \in T} s = t$

NP-Completeness of Subset Sum

Show: If there is $T \subseteq S$ with $\sum_{s \in T} s = t$, then φ is satisfiable.

Let $T \subseteq S$ such that $\sum_{s \in T} s = t$

Define $\beta(X_i) = \begin{cases} 1 & \text{if } t_i \in T \\ 0 & \text{if } f_i \in T \end{cases}$

This is well defined as for all i: $t_i \in T$ or $f_i \in T$ but not both.

Further, for each clause, there must be one literal set to 1 as for all i, the $m_{i,j} \in S$ do not sum up to the number of literals in the clause. □

Knapsack and Strong NP-Completeness

Towards More NP-Complete Problems

Starting with SAT, one can readily show more problems $\mathcal{P}$ to be NP-complete, each time performing two steps:

(1) Show that $\mathcal{P} \in \mathrm{NP}$

(2) Find a known NP-complete problem $\mathcal{P}'$ and reduce $\mathcal{P}' \leq_p \mathcal{P}$

Thousands of problem have now been shown to be NP-complete.
(See Garey and Johnson for an early survey)

In this course:

	$\leq_p$	CLIQUE	$\leq_p$	INDEPENDENT SET
SAT	$\leq_p$	3-SAT	$\leq_p$	DIR. HAMILTONIAN PATH
	$\leq_p$	SUBSET SUM	$\leq_p$	KNAPSACK

NP-completeness of KNAPSACK

KNAPSACK

Input: A set $I := \{1, \ldots, n\}$ of items
each of value v_i and weight w_i for $1 \leq i \leq n$,
target value t and weight limit ℓ

Problem: Is there $T \subseteq I$ such that
$\sum_{i \in T} v_i \geq t$ and $\sum_{i \in T} w_i \leq \ell$?

Theorem 9.4

KNAPSACK *is* NP*-complete.*

NP-completeness of Knapsack

Knapsack

Input: A set $I := \{1, \ldots, n\}$ of items each of value v_i and weight w_i for $1 \leq i \leq n$, target value t and weight limit ℓ

Problem: Is there $T \subseteq I$ such that $\sum_{i \in T} v_i \geq t$ and $\sum_{i \in T} w_i \leq \ell$?

Theorem 9.4

Knapsack *is* NP*-complete.*

Proof.

- Knapsack $\in \mathrm{NP}$: Take T to be the certificate.
- Knapsack is NP-hard: Subset Sum $\leq_p$ Knapsack

Subset Sum $\leq_p$ Knapsack

Subset Sum:

Given:	$S := \{a_1, \ldots, a_n\}$	collection of positive integers
	t	target integer

Problem: Is there a subset $T \subseteq S$ such that $\sum_{a_i \in T} a_i = t$?

Subset Sum $\leq_p$ Knapsack

Subset Sum:

Given:	$S := \{a_1, \ldots, a_n\}$	collection of positive integers
	t	target integer

Problem: Is there a subset $T \subseteq S$ such that $\sum_{a_i \in T} a_i = t$?

Reduction: From this input to Subset Sum construct

- set of items $I := \{1, \ldots, n\}$
- weights and values $v_i = w_i = a_i$ for all $1 \leq i \leq n$
- target value $t' := t$ and weight limit $\ell := t$

Subset Sum $\leq_p$ Knapsack

Subset Sum:

Given:	$S := \{a_1, \ldots, a_n\}$	collection of positive integers
	t	target integer

Problem: Is there a subset $T \subseteq S$ such that $\sum_{a_i \in T} a_i = t$?

Reduction: From this input to Subset Sum construct

- set of items $I := \{1, \ldots, n\}$
- weights and values $v_i = w_i = a_i$ for all $1 \leq i \leq n$
- target value $t' := t$ and weight limit $\ell := t$

Clearly: For every $T \subseteq S$

$$\sum_{a_i \in T} a_i = t \quad \text{iff} \quad \begin{array}{ll} \sum_{a_i \in T} v_i \geq t' & = t \\ \sum_{a_i \in T} w_i \leq \ell & = t \end{array}$$

Hence: The reduction is correct and in polynomial time.

A Polynomial Time Algorithm for KNAPSACK

KNAPSACK can be solved in time $O(n\ell)$ using dynamic programming

Initialisation:

- Create an $(\ell + 1) \times (n + 1)$ matrix M
- Set $M(w, 0) := 0$ for all $1 \leq w \leq \ell$ and $M(0, i) := 0$ for all $1 \leq i \leq n$

Example

Input $I = \{1, 2, 3, 4\}$ with

Values:	$v_1 = 1$	$v_2 = 3$	$v_3 = 4$	$v_4 = 2$
Weight:	$w_1 = 1$	$w_2 = 1$	$w_3 = 3$	$w_4 = 2$

Weight limit: $\ell = 5$ Target value: $t = 7$

weight limit w	max. total value from first i items				
	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
0					
1					
2					
3					
4					
5					

Set $M(w, 0) := 0$ for all $1 \leq w \leq \ell$ and $M(0, i) := 0$ for all $1 \leq i \leq n$

Example

Input $I = \{1, 2, 3, 4\}$ with

Values: $v_1 = 1$ $v_2 = 3$ $v_3 = 4$ $v_4 = 2$

Weight: $w_1 = 1$ $w_2 = 1$ $w_3 = 3$ $w_4 = 2$

Weight limit: $\ell = 5$ Target value: $t = 7$

weight limit w	max. total value from first i items: $i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
0	0	0	0	0	0
1	0				
2	0				
3	0				
4	0				
5	0				

Set $M(w, 0) := 0$ for all $1 \leq w \leq \ell$ and $M(0, i) := 0$ for all $1 \leq i \leq n$

A Polynomial Time Algorithm for KNAPSACK

KNAPSACK can be solved in time $O(n\ell)$ using dynamic programming

Initialisation:

- Create an $(\ell + 1) \times (n + 1)$ matrix M
- Set $M(w, 0) := 0$ for all $1 \leq w \leq \ell$ and $M(0, i) := 0$ for all $1 \leq i \leq n$

A Polynomial Time Algorithm for KNAPSACK

KNAPSACK can be solved in time $O(n\ell)$ using dynamic programming

Initialisation:

- Create an $(\ell + 1) \times (n + 1)$ matrix M
- Set $M(w, 0) := 0$ for all $1 \leq w \leq \ell$ and $M(0, i) := 0$ for all $1 \leq i \leq n$

Computation: Assign further $M(w, i)$ to be the largest total value obtainable by selecting from the first i items with weight limit w:

For $i = 0, 1, \ldots, n - 1$ set $M(w, i + 1)$ as

$$M(w, i + 1) := \max\left\{M(w, i),\ M(w - w_{i+1}, i) + v_{i+1}\right\}$$

Here, if $w - w_{i+1} < 0$ we always take $M(w, i)$.

Acceptance: If M contains an entry $\geq t$, accept. Otherwise reject.

Example

Input $I = \{1, 2, 3, 4\}$ with

Values: $v_1 = 1$ $v_2 = 3$ $v_3 = 4$ $v_4 = 2$

Weight: $w_1 = 1$ $w_2 = 1$ $w_3 = 3$ $w_4 = 2$

Weight limit: $\ell = 5$ Target value: $t = 7$

weight limit w	max. total value from first i items				
	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
0	0	0	0	0	0
1	0				
2	0				
3	0				
4	0				
5	0				

For $i = 0, 1, \ldots, n-1$ set $M(w, i+1) := \max\left\{M(w, i),\ M(w - w_{i+1}, i) + v_{i+1}\right\}$

Example

Input $I = \{1, 2, 3, 4\}$ with

Values: $v_1 = 1$ $v_2 = 3$ $v_3 = 4$ $v_4 = 2$

Weight: $w_1 = 1$ $w_2 = 1$ $w_3 = 3$ $w_4 = 2$

Weight limit: $\ell = 5$ Target value: $t = 7$

weight limit w	max. total value from first i items				
	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
0	0	0	0	0	0
1	0	1			
2	0	1			
3	0	1			
4	0	1			
5	0	1			

For $i = 0, 1, \ldots, n-1$ set $M(w, i+1) := \max\left\{M(w, i),\ M(w - w_{i+1}, i) + v_{i+1}\right\}$

Example

Input $I = \{1, 2, 3, 4\}$ with

Values: $v_1 = 1$ $v_2 = 3$ $v_3 = 4$ $v_4 = 2$

Weight: $w_1 = 1$ $w_2 = 1$ $w_3 = 3$ $w_4 = 2$

Weight limit: $\ell = 5$ Target value: $t = 7$

weight limit w	max. total value from first i items				
	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
0	0	0	0	0	0
1	0	1	3		
2	0	1			
3	0	1			
4	0	1			
5	0	1			

For $i = 0, 1, \ldots, n-1$ set $M(w, i+1) := \max\left\{M(w, i),\ M(w - w_{i+1}, i) + v_{i+1}\right\}$

Example

Input $I = \{1, 2, 3, 4\}$ with

Values: $v_1 = 1$ $v_2 = 3$ $v_3 = 4$ $v_4 = 2$

Weight: $w_1 = 1$ $w_2 = 1$ $w_3 = 3$ $w_4 = 2$

Weight limit: $\ell = 5$ Target value: $t = 7$

weight limit w	max. total value from first i items				
	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
0	0	0	0	0	0
1	0	1	3		
2	0	1	4		
3	0	1			
4	0	1			
5	0	1			

For $i = 0, 1, \ldots, n-1$ set $M(w, i+1) := \max\left\{M(w, i),\ M(w - w_{i+1}, i) + v_{i+1}\right\}$

Example

Input $I = \{1, 2, 3, 4\}$ with

Values: $v_1 = 1$ $v_2 = 3$ $v_3 = 4$ $v_4 = 2$

Weight: $w_1 = 1$ $w_2 = 1$ $w_3 = 3$ $w_4 = 2$

Weight limit: $\ell = 5$ Target value: $t = 7$

weight limit w	max. total value from first i items				
	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
0	0	0	0	0	0
1	0	1	3		
2	0	1	4		
3	0	1	4		
4	0	1			
5	0	1			

For $i = 0, 1, \ldots, n-1$ set $M(w, i+1) := \max\left\{M(w, i),\ M(w - w_{i+1}, i) + v_{i+1}\right\}$

Example

Input $I = \{1, 2, 3, 4\}$ with

Values: $v_1 = 1$ $v_2 = 3$ $v_3 = 4$ $v_4 = 2$

Weight: $w_1 = 1$ $w_2 = 1$ $w_3 = 3$ $w_4 = 2$

Weight limit: $\ell = 5$ Target value: $t = 7$

weight limit w	max. total value from first i items				
	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
0	0	0	0	0	0
1	0	1	3		
2	0	1	4		
3	0	1	4		
4	0	1	4		
5	0	1			

For $i = 0, 1, \ldots, n-1$ set $M(w, i+1) := \max\left\{M(w, i),\ M(w - w_{i+1}, i) + v_{i+1}\right\}$

Example

Input $I = \{1, 2, 3, 4\}$ with

Values: $v_1 = 1$ $v_2 = 3$ $v_3 = 4$ $v_4 = 2$

Weight: $w_1 = 1$ $w_2 = 1$ $w_3 = 3$ $w_4 = 2$

Weight limit: $\ell = 5$ Target value: $t = 7$

weight limit w	max. total value from first i items				
	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
0	0	0	0	0	0
1	0	1	3		
2	0	1	4		
3	0	1	4		
4	0	1	4		
5	0	1	4		

For $i = 0, 1, \ldots, n-1$ set $M(w, i+1) := \max\left\{M(w, i),\ M(w - w_{i+1}, i) + v_{i+1}\right\}$

Example

Input $I = \{1, 2, 3, 4\}$ with

Values: $v_1 = 1$ $v_2 = 3$ $v_3 = 4$ $v_4 = 2$

Weight: $w_1 = 1$ $w_2 = 1$ $w_3 = 3$ $w_4 = 2$

Weight limit: $\ell = 5$ Target value: $t = 7$

weight limit w	max. total value from first i items				
	$i = 0$	$i = 1$	$i = 2$	$i = 3$	$i = 4$
0	0	0	0	0	0
1	0	1	3	3	3
2	0	1	4	4	4
3	0	1	4	4	5
4	0	1	4	7	7
5	0	1	4	8	8

For $i = 0, 1, \ldots, n-1$ set $M(w, i+1) := \max\left\{M(w, i),\ M(w - w_{i+1}, i) + v_{i+1}\right\}$

Did we prove $\mathrm{P} = \mathrm{NP}$?

Summary:

- Theorem 9.4: KNAPSACK is NP-complete
- KNAPSACK can be solved in time $O(n\ell)$ using dynamic programming

What went wrong?

KNAPSACK

Input: A set $I := \{1, \ldots, n\}$ of items
each of value v_i and weight w_i for $1 \leq i \leq n$,
target value t and weight limit ℓ

Problem: Is there $T \subseteq I$ such that
$\sum_{i \in T} v_i \geq t$ and $\sum_{i \in T} w_i \leq \ell$?

Pseudo-Polynomial Time

The previous algorithm is not sufficient to show that KNAPSACK is in P

- The algorithm fills a $(\ell + 1) \times (n + 1)$ matrix M
- The size of the input to KNAPSACK is $O(n \log \ell)$

$\leadsto$ the size of M is not bounded by a polynomial in the length of the input!

Pseudo-Polynomial Time

The previous algorithm is not sufficient to show that KNAPSACK is in P

- The algorithm fills a $(\ell + 1) \times (n + 1)$ matrix M
- The size of the input to KNAPSACK is $O(n \log \ell)$

$\leadsto$ the size of M is not bounded by a polynomial in the length of the input!

Definition 9.5 (Pseudo-Polynomial Time)

Problems decidable in time polynomial in the sum of the input length and the value of numbers occurring in the input.

Equivalently: Problems decidable in polynomial time when using unary encoding for all numbers in the input.

- If KNAPSACK is restricted to instances with $\ell \leq p(n)$ for a polynomial p, then we obtain a problem in P.
- KNAPSACK is in polynomial time for unary encoding of numbers.

Strong NP-completeness

Pseudo Polynomial time: Algorithms polynomial in the maximum of the input length and the value of numbers occurring in the input.

Examples:

- Knapsack
- Subset Sum

Strong NP-completeness: Problems which remain NP-complete even if all numbers are bounded by a polynomial in the input length (equivalently: even for unary coding of numbers).

Examples:

- Clique
- Sat
- Hamiltonian Cycle
- ...

Note: Showing Sat $\leq_p$ Subset Sum required exponentially large numbers.

coNP

The Class coNP

Recall that coNP is the complement class of NP.

Definition 9.6

- For a language $\mathcal{L} \subseteq \Sigma^*$ let $\overline{\mathcal{L}} := \Sigma^* \setminus \mathcal{L}$ be its complement
- For a complexity class C, we define $\mathrm{coC} := \{\mathcal{L} \mid \overline{\mathcal{L}} \in \mathrm{C}\}$
- In particular $\mathrm{coNP} = \{\mathcal{L} \mid \overline{\mathcal{L}} \in \mathrm{NP}\}$

A problem belongs to coNP, if no-instances have short certificates.

Examples:

- No Hamiltonian Path: Does the graph G not have a Hamiltonian path?
- Tautology: Is the propositional logic formula φ a tautology (true under all assignments)?
- ...

coNP-completeness

Definition 9.7

A language $C \in \text{coNP}$ is coNP-complete, if $\mathcal{L} \leq_p C$ for all $\mathcal{L} \in \text{coNP}$.

Theorem 9.8

- $\text{P} = \text{coP}$
- *Hence,* $\text{P} \subseteq \text{NP} \cap \text{coNP}$

Open questions:

- $\text{NP} = \text{coNP}$?

  Most people do not think so.

- $P = \text{NP} \cap \text{coNP}$?

  Again, most people do not think so.