Fundamental Computer Science

Denis Trystram, inspired by Giorgio Lucarelli

March, 2020

Today

- Deal with numerical problems

SubsetSum $\in$ NP-complete

SubsetSum

Input: a set of positive integers $A = \{a_1, a_2, \ldots, a_k\}$, and a $t \in \mathbb{N}$

Question: is there a set $B \subseteq A$ such that $\sum_{a_i \in B} a_i = t$?

SubsetSum $\in$ NP-complete

SubsetSum

Input: a set of positive integers $A = \{a_1, a_2, \ldots, a_k\}$, and a $t \in \mathbb{N}$

Question: is there a set $B \subseteq A$ such that $\sum_{a_i \in B} a_i = t$?

SubsetSum is in NP

- given the set $B \subseteq A$, create the sum of the elements in B and compare with t

SubsetSum $\in$ NP-complete

SubsetSum

Input: a set of positive integers $A = \{a_1, a_2, \ldots, a_k\}$, and a $t \in \mathbb{N}$

Question: is there a set $B \subseteq A$ such that $\sum_{a_i \in B} a_i = t$?

SubsetSum is in NP

- given the set $B \subseteq A$, create the sum of the elements in B and compare with t

3SAT $\leq_P$ SubsetSum

1. for each variable x_i create two decimal numbers y_i and z_i
   - intuition:
     - select one of y_i, z_i in B
     - if y_i is in B, then $x_i = \text{TRUE}$
     - if z_i is in B, then $x_i = \text{FALSE}$

SubsetSum $\in$ NP-complete

SubsetSum

Input: a set of positive integers $A = \{a_1, a_2, \ldots, a_k\}$, and a $t \in \mathbb{N}$

Question: is there a set $B \subseteq A$ such that $\sum_{a_i \in B} a_i = t$?

SubsetSum is in NP

- given the set $B \subseteq A$, create the sum of the elements in B and compare with t

$3\mathrm{SAT} \leq_{\mathrm{P}}$ SubsetSum

1. for each variable x_i create two decimal numbers y_i and z_i
   - intuition:
     – select one of y_i, z_i in B
     – if y_i is in B, then $x_i = \mathrm{TRUE}$
     – if z_i is in B, then $x_i = \mathrm{FALSE}$
   - each y_i, z_i has two parts:
     – a variable part
     – a clause part

	x_1	x_2	x_3	$\ldots$	x_n	c_1	c_2	$\ldots$	c_m
y_1	1	0	0	$\ldots$	0	1	0	$\ldots$	0
z_1	1	0	0	$\ldots$	0	0	0	$\ldots$	0
y_2		1	0	$\ldots$	0	0	1	$\ldots$	0
z_2		1	0	$\ldots$	0	1	0	$\ldots$	0
y_3			1	$\ldots$	0	1	1	$\ldots$	0
z_3			1	$\ldots$	0	0	0	$\ldots$	1
$\vdots$				$\ddots$	$\vdots$			$\vdots$	
y_n					1	0	0	$\ldots$	1
z_n					1	0	0	$\ldots$	0
g_1						1	0	$\ldots$	0
h_1						1	0	$\ldots$	0
g_2							1	$\ldots$	0
h_2							1	$\ldots$	0
$\vdots$								$\ddots$	
g_m									1
h_m									1
t	1	1	1	$\ldots$	1	3	3	$\ldots$	3

Example

$$(x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee x_3 \vee x_4) \wedge (\bar{x}_1 \vee \bar{x}_3 \vee x_4)$$

	x_1	x_2	x_3	x_4	c_1	c_2	c_3
y_1	1	0	0	0	1	0	0
z_1	1	0	0	0	0	0	1
y_2		1	0	0	0	1	0
z_2		1	0	0	1	0	0
y_3			1	0	1	1	0
z_3			1	0	0	0	1
y_4				1	0	1	1
z_4				1	0	0	0
g_1					1	0	0
h_1					1	0	0
g_2						1	0
h_2						1	0
g_3							1
h_3							1
W	1	1	1	1	3	3	3

SubsetSum $\in$ NP-complete

2. Size of the created instance:
   - $|A| = 2n + 2m$
   - each created integer has at most $n + m$ digits (including t)
     $\rightarrow$ integers in the interval $[0, 10^{n+m}]$
     $\rightarrow$ binary representation: at most $\log_2 10^{n+m} = O(n + m)$ bits

SubsetSum $\in$ NP-complete

2. Size of the created instance:
   - $|A| = 2n + 2m$
   - each created integer has at most $n + m$ digits (including t)
     $\rightarrow$ integers in the interval $[0, 10^{n+m}]$
     $\rightarrow$ binary representation: at most $\log_2 10^{n+m} = O(n + m)$ bits

3. $\mathcal{F}$ is satisfiable iff there is a set $B \subseteq A$ with $\sum_{a_i \in B} a_i = t$
   ($\Rightarrow$)
   - assume that $\mathcal{F}$ is satisfiable

SubsetSum $\in$ NP-complete

2. Size of the created instance:
   - $|A| = 2n + 2m$
   - each created integer has at most $n + m$ digits (including t)
     $\rightarrow$ integers in the interval $[0, 10^{n+m}]$
     $\rightarrow$ binary representation: at most $\log_2 10^{n+m} = O(n+m)$ bits

3. $\mathcal{F}$ is satisfiable iff there is a set $B \subseteq A$ with $\sum_{a_i \in B} a_i = t$
   $(\Rightarrow)$
   - assume that $\mathcal{F}$ is satisfiable
   - for each x_i:
     – if $x_i = \text{TRUE}$, then add y_i to B
     – if $x_i = \text{FALSE}$, then add z_i to B
   - for each c_j:
     – if 1 literal is TRUE, then add both g_j and h_j in B
     – if 2 literal are TRUE, then add g_j in B

SubsetSum $\in$ NP-complete

2. Size of the created instance:
   - $|A| = 2n + 2m$
   - each created integer has at most $n + m$ digits (including t)
     $\rightarrow$ integers in the interval $[0, 10^{n+m}]$
     $\rightarrow$ binary representation: at most $\log_2 10^{n+m} = O(n+m)$ bits

3. $\mathcal{F}$ is satisfiable iff there is a set $B \subseteq A$ with $\sum_{a_i \in B} a_i = t$
   $(\Rightarrow)$
   - assume that $\mathcal{F}$ is satisfiable
   - for each x_i:
     – if $x_i = \mathrm{TRUE}$, then add y_i to B
     – if $x_i = \mathrm{FALSE}$, then add z_i to B
   - for each c_j:
     – if 1 literal is TRUE, then add both g_j and h_j in B
     – if 2 literal are TRUE, then add g_j in B
   - B is a SubsetSum
     – left part of t: we select only one of y_i and z_i, for each $1 \leq i \leq n$
     – right part of t: we select g_j and h_j in order to have exactly 3 ones per clause

SubsetSum $\in$ NP-complete

3. $\mathcal{F}$ is satisfiable iff there is a set $B \subseteq A$ with $\sum_{a_i \in B} a_i = t$

   $(\Leftarrow)$

   - assume there is a set B such that $\sum_{a_i \in B} a_i = t$
   - each column contains at most 5 ones $\rightarrow$ there is not a "carry"

SubsetSum $\in$ NP-complete

3. $\mathcal{F}$ is satisfiable iff there is a set $B \subseteq A$ with $\sum_{a_i \in B} a_i = t$
   ($\Leftarrow$)
   - assume there is a set B such that $\sum_{a_i \in B} a_i = t$
   - each column contains at most 5 ones $\rightarrow$ there is not a "carry"
   - there is no other way to have 1 in the variable part of t except from selecting exactly one of each y_i and z_i

SUBSETSUM $\in$ NP-COMPLETE

3. $\mathcal{F}$ is satisfiable iff there is a set $B \subseteq A$ with $\sum_{a_i \in B} a_i = t$

   ($\Leftarrow$)

   - assume there is a set B such that $\sum_{a_i \in B} a_i = t$
   - each column contains at most 5 ones $\rightarrow$ there is not a "carry"
   - there is no other way to have 1 in the variable part of t except from selecting exactly one of each y_i and z_i
   - then, set:
     – $x_i = \mathrm{TRUE}$, if $y_i \in B$
     – $x_i = \mathrm{FALSE}$, if $z_i \in B$

SubsetSum $\in$ NP-complete

3. $\mathcal{F}$ is satisfiable iff there is a set $B \subseteq A$ with $\sum_{a_i \in B} a_i = t$
   ($\Leftarrow$)
   - assume there is a set B such that $\sum_{a_i \in B} a_i = t$
   - each column contains at most 5 ones $\rightarrow$ there is not a "carry"
   - there is no other way to have 1 in the variable part of t except from selecting exactly one of each y_i and z_i
   - then, set:
     – $x_i = \mathrm{TRUE}$, if $y_i \in B$
     – $x_i = \mathrm{FALSE}$, if $z_i \in B$
   - there is no way to have 3 in the clause part of t by selecting only g_j and h_j

SubsetSum $\in$ NP-complete

3. $\mathcal{F}$ is satisfiable iff there is a set $B \subseteq A$ with $\sum_{a_i \in B} a_i = t$
   ($\Leftarrow$)
   - assume there is a set B such that $\sum_{a_i \in B} a_i = t$
   - each column contains at most 5 ones $\rightarrow$ there is not a "carry"
   - there is no other way to have 1 in the variable part of t except from selecting exactly one of each y_i and z_i
   - then, set:
     – $x_i = \text{TRUE}$, if $y_i \in B$
     – $x_i = \text{FALSE}$, if $z_i \in B$
   - there is no way to have 3 in the clause part of t by selecting only g_j and h_j
   - thus, at least one literal (y_i, z_i) should be one for each clause column
   - therefore, this assignment satisfies $\mathcal{F}$

An algorithm for SubsetSum

- dynamic programming

An algorithm for SubsetSum

- dynamic programming
- consider the integers sorted in non-decreasing order: $a_1 \leq a_2 \leq \ldots \leq a_n$
- $S[i, q] = \begin{cases} \text{True}, & \text{if there is a } \textsc{SubsetSum} \text{ among the } i \text{ first} \\ & \text{integers which sums up exactly to } q \\ \text{False}, & \text{otherwise} \end{cases}$

An algorithm for SUBSETSUM

- dynamic programming
- consider the integers sorted in non-decreasing order: $a_1 \leq a_2 \leq \ldots \leq a_n$
- $S[i, q] = \begin{cases} \text{True}, & \text{if there is a SUBSETSUM among the } i \text{ first} \\ & \text{integers which sums up exactly to } q \\ \text{False}, & \text{otherwise} \end{cases}$

Algorithm

1: Initialization:
- $S[i, 0] = \text{True}$, for any $i \geq 1$
- $S[1, q] = \begin{cases} \text{True}, & \text{if } q = a_1 \\ \text{False}, & \text{otherwise} \end{cases}$

An algorithm for SubsetSum

- dynamic programming
- consider the integers sorted in non-decreasing order: $a_1 \leq a_2 \leq \ldots \leq a_n$
- $S[i,q] = \begin{cases} \text{True}, & \text{if there is a SubsetSum among the } i \text{ first integers which sums up exactly to } q \\ \text{False}, & \text{otherwise} \end{cases}$

Algorithm

1: Initialization:
   – $S[i,0] = \text{True}$, for any $i \geq 1$
   – $S[1,q] = \begin{cases} \text{True}, & \text{if } q = a_1 \\ \text{False}, & \text{otherwise} \end{cases}$
2: **for** $i = 1$ to n **do**
3:     **for** $q = 1$ to t **do**
4:         $S[i,q] = S[i-1,q]$ **or** $S[i-1,q-a_i]$

An algorithm for SUBSETSUM

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

An algorithm for SubsetSum

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q											
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T											
	2	T											
i	3	T											
	4	T											
	5	T											

$S[i, 0] = \text{True}$, for any $i \geq 1$

An algorithm for SubsetSum

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q											
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T											
i	3	T											
	4	T											
	5	T											

$$S[1, q] = \begin{cases} \text{True}, & \text{if } q = a_1 \\ \text{False}, & \text{otherwise} \end{cases}$$

An algorithm for SubsetSum

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q											
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T	F	T									
i	3	T											
	4	T											
	5	T											

$S[i, q] = S[i-1, q]$ **or** $S[i-1, q-a_i]$
$S[2, 2] = S[1, 2]$ **or** $S[1, -1]$

An algorithm for SubsetSum

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q											
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T	F	T	T								
i	3	T											
	4	T											
	5	T											

$S[i, q] = S[i-1, q]$ **or** $S[i-1, q-a_i]$
$S[2, 3] = S[1, 3]$ **or** $S[1, 0]$

An algorithm for SUBSETSUM

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q											
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T	F	T	T	F	T						
i	3	T											
	4	T											
	5	T											

$S[i, q] = S[i-1, q]$ **or** $S[i-1, q-a_i]$
$S[2, 5] = S[1, 5]$ **or** $S[1, 2]$

An algorithm for SubsetSum

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q											
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T	F	T	T	F	T	F	F	F	F	F	F
i	3	T											
	4	T											
	5	T											

$S[i, q] = S[i-1, q]$ **or** $S[i-1, q-a_i]$

An algorithm for SUBSETSUM

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q											
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T	F	T	T	F	T	F	F	F	F	F	F
i	3	T	F	T	T	T							
	4	T											
	5	T											

$S[i, q] = S[i-1, q]$ **or** $S[i-1, q-a_i]$
$S[3, 4] = S[2, 4]$ **or** $S[2, 0]$

An algorithm for SUBSETSUM

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q											
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T	F	T	T	F	T	F	F	F	F	F	F
i	3	T	F	T	T	T	T	T					
	4	T											
	5	T											

$S[i, q] = S[i-1, q]$ **or** $S[i-1, q-a_i]$
$S[3, 6] = S[2, 6]$ **or** $S[2, 2]$

An algorithm for SubsetSum

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

								q					
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T	F	T	T	F	T	F	F	F	F	F	F
i	3	T	F	T	T	T	T	T	T	F	T	F	F
	4	T											
	5	T											

$S[i, q] = S[i-1, q]$ **or** $S[i-1, q-a_i]$

An algorithm for SubsetSum

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q											
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T	F	T	T	F	T	F	F	F	F	F	F
i	3	T	F	T	T	T	T	T	T	F	T	F	F
	4	T	F	T	T	T	T	T	T	T	T	T	T
	5	T											

$S[i, q] = S[i-1, q]$ **or** $S[i-1, q-a_i]$

An algorithm for SubsetSum

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q											
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T	F	T	T	F	T	F	F	F	F	F	F
i	3	T	F	T	T	T	T	T	T	F	T	F	F
	4	T	F	T	T	T	T	T	T	T	T	T	T
	5	T	F	T	T	T	T	T	T	T	T	T	T

- there is a TRUE in column $q = 11$, hence $\langle A, t \rangle \in$ SubsetSum

An algorithm for SubsetSum

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q											
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T	F	T	T	F	T	F	F	F	F	F	F
i	3	T	F	T	T	T	T	T	T	F	T	F	F
	4	T	F	T	T	T	T	T	T	T	T	T	T
	5	T	F	T	T	T	T	T	T	T	T	T	T

- how to construct the set B ?

An algorithm for SubsetSum

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q											
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T	F	T	T	F	T	F	F	F	F	F	F
i	3	T	F	T	T	T	T	T	T	F	T	F	F
	4	T	F	T	T	T	T	T	T	T	T	T	T
	5	T	F	T	T	T	T	T	T	T	T	T	T

- how to construct the set B ?
- $S[5, 11] = S[4, 11]$ **or** $S[4, 3]$
  - $S[4, 11]$: $a_5 \notin B$
  - $S[4, 3]$: $a_5 \in B$

An algorithm for SUBSETSUM

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q											
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T	F	T	T	F	T	F	F	F	F	F	F
i	3	T	F	T	T	T	T	T	T	F	T	F	F
	4	T	F	T	T	T	T	T	T	T	T	T	T
	5	T	F	T	T	T	T	T	T	T	T	T	T

- how to construct the set B ?
- $S[5, 11] = S[4, 11]$ **or** $S[4, 3]$
  - $S[4, 11]$: $a_5 \notin B$
  - $S[4, 3]$: $a_5 \in B$
- $S[4, 3] = S[3, 3]$ **or** $S[3, -3]$, so $a_4 \notin B$

An algorithm for SubsetSum

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q											
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T	F	T	T	F	T	F	F	F	F	F	F
i	3	T	F	T	T	T	T	T	T	F	T	F	F
	4	T	F	T	T	T	T	T	T	T	T	T	T
	5	T	F	T	T	T	T	T	T	T	T	T	T

- how to construct the set B ?
- $S[5, 11] = S[4, 11]$ **or** $S[4, 3]$
  - $S[4, 11]$: $a_5 \notin B$
  - $S[4, 3]$: $a_5 \in B$
- $S[4, 3] = S[3, 3]$ **or** $S[3, -3]$, so $a_4 \notin B$
- $S[3, 3] = S[2, 3]$ **or** $S[2, -1]$, so $a_3 \notin B$

An algorithm for SubsetSum

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q											
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T	F	T	T	F	T	F	F	F	F	F	F
i	3	T	F	T	T	T	T	T	T	F	T	F	F
	4	T	F	T	T	T	T	T	T	T	T	T	T
	5	T	F	T	T	T	T	T	T	T	T	T	T

- how to construct the set B ?
- $S[5, 11] = S[4, 11]$ **or** $S[4, 3]$
  - $S[4, 11]$: $a_5 \notin B$
  - $S[4, 3]$: $a_5 \in B$
- $S[4, 3] = S[3, 3]$ **or** $S[3, -3]$, so $a_4 \notin B$
- $S[3, 3] = S[2, 3]$ **or** $S[2, -1]$, so $a_3 \notin B$
- $S[2, 3] = S[1, 3]$ **or** $S[1, 0]$, so $a_2 \in B$, $a_1 \notin B$

An algorithm for SUBSETSUM

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q											
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T	F	T	T	F	T	F	F	F	F	F	F
i	3	T	F	T	T	T	T	T	T	F	T	F	F
	4	T	F	T	T	T	T	T	T	T	T	T	T
	5	T	F	T	T	T	T	T	T	T	T	T	T

- Complexity?

An algorithm for SubsetSum

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q 0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T	F	T	T	F	T	F	F	F	F	F	F
i	3	T	F	T	T	T	T	T	T	F	T	F	F
	4	T	F	T	T	T	T	T	T	T	T	T	T
	5	T	F	T	T	T	T	T	T	T	T	T	T

- Complexity?
- $O(n \cdot t)$

An algorithm for SubsetSum

- Example: $A = \{2, 3, 4, 6, 8\}$ and $t = 11$

		q											
		0	1	2	3	4	5	6	7	8	9	10	11
	1	T	F	T	F	F	F	F	F	F	F	F	F
	2	T	F	T	T	F	T	F	F	F	F	F	F
i	3	T	F	T	T	T	T	T	T	F	T	F	F
	4	T	F	T	T	T	T	T	T	T	T	T	T
	5	T	F	T	T	T	T	T	T	T	T	T	T

- Complexity?
- $O(n \cdot t)$
- Is this polynomial?
- if yes, then $\mathrm{P} = \mathrm{NP}$!!!

An algorithm for SubsetSum

- input: $I = \langle A, t \rangle$
- size of the input:

$$|I| = \log_2 t + \sum_{a_i \in A} \log_2 a_i$$

An algorithm for SubsetSum

- input: $I = \langle A, t \rangle$
- size of the input:

$$|I| = \log_2 t + \sum_{a_i \in A} \log_2 a_i = O(\log_2 t)$$

An algorithm for SubsetSum

- input: $I = \langle A, t \rangle$
- size of the input:

$$|I| = \log_2 t + \sum_{a_i \in A} \log_2 a_i = O(\log_2 t)$$

- complexity of the algorithm:

$$O(n \cdot t) = O(n \cdot 2^{|I|})$$

An algorithm for SubsetSum

- input: $I = \langle A, t \rangle$
- size of the input:

$$|I| = \log_2 t + \sum_{a_i \in A} \log_2 a_i = O(\log_2 t)$$

- complexity of the algorithm:

$$O(n \cdot t) = O(n \cdot 2^{|I|})$$

- that is, exponential to the size of the input !

Pseudopolynomial algorithms

- $|I|_1$: the encoding of the input in unary

Pseudopolynomial algorithms

- $|I|_1$: the encoding of the input in unary
- example: $\textsc{SubsetSum}$

$$|I|_1 = t + \sum_{a_i \in A} a_i$$

Pseudopolynomial algorithms

- $|I|_1$: the encoding of the input in unary
- example: SubsetSum

$$|I|_1 = t + \sum_{a_i \in A} a_i$$

then the complexity of the algorithm is polynomial:

$$O(n \cdot t) = O(n \cdot |I|_1)$$

Pseudopolynomial algorithms

- $|I|_1$: the encoding of the input in unary
- example: SUBSETSUM

  $$|I|_1 = t + \sum_{a_i \in A} a_i$$

  then the complexity of the algorithm is polynomial:

  $$O(n \cdot t) = O(n \cdot |I|_1)$$

- Definition: we call an algorithm **pseudopolynomial** if its complexity is polynomial to the size of the input, when this is encoded in unary.

Pseudopolynomial algorithms

- $|I|_1$: the encoding of the input in unary
- example: SubsetSum

$$|I|_1 = t + \sum_{a_i \in A} a_i$$

then the complexity of the algorithm is polynomial:

$$O(n \cdot t) = O(n \cdot |I|_1)$$

- Definition: we call an algorithm **pseudopolynomial** if its complexity is polynomial to the size of the input, when this is encoded in unary.
- Definition: NP-complete problems that admit a pseudopolynomial algorithm are called **weakly** NP-complete.

Pseudopolynomial algorithms

- $|I|_1$: the encoding of the input in unary
- example: SubsetSum

$$|I|_1 = t + \sum_{a_i \in A} a_i$$

then the complexity of the algorithm is polynomial:

$$O(n \cdot t) = O(n \cdot |I|_1)$$

- Definition: we call an algorithm **pseudopolynomial** if its complexity is polynomial to the size of the input, when this is encoded in unary.
- Definition: NP-complete problems that admit a pseudopolynomial algorithm are called **weakly** NP-complete.
- Definition: we call a problem **strong** or **unary** NP-complete if it remains NP-complete even when the input is encoded in unary.

Observations

- where is the problem with the reduction of SubsetSum if the input is encoded in unary?

Observations

- where is the problem with the reduction of SubsetSum if the input is encoded in unary?
  - each created integer has at most $n + m$ digits (including t)
    $\rightarrow$ integers in the interval $[0, 10^{n+m}]$
    $\rightarrow$ unary representation: 10^{n+m} symbols per integer
  - the size of the created input is **not** polynomial with respect to the size of the initial input

Observations

- where is the problem with the reduction of SubsetSum if the input is encoded in unary?
  - each created integer has at most $n+m$ digits (including t)
    $\rightarrow$ integers in the interval $[0, 10^{n+m}]$
    $\rightarrow$ unary representation: 10^{n+m} symbols per integer
  - the size of the created input is **not** polynomial with respect to the size of the initial input
- are there numerical problems that are strong NP-complete?

Observations

- where is the problem with the reduction of SubsetSum if the input is encoded in unary?
  - each created integer has at most $n + m$ digits (including t)
    $\rightarrow$ integers in the interval $[0, 10^{n+m}]$
    $\rightarrow$ unary representation: 10^{n+m} symbols per integer
  - the size of the created input is **not** polynomial with respect to the size of the initial input
- are there numerical problems that are strong NP-complete?
  - YES
  - 3-Partition, Bin-Packing, ...

Observations

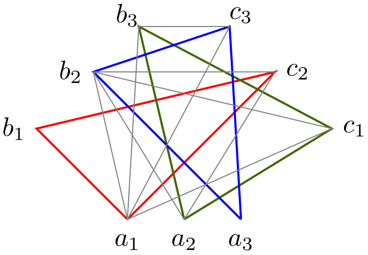
- where is the problem with the reduction of SubsetSum if the input is encoded in unary?
  - each created integer has at most $n + m$ digits (including t)
    $\rightarrow$ integers in the interval $[0, 10^{n+m}]$
    $\rightarrow$ unary representation: 10^{n+m} symbols per integer
  - the size of the created input is **not** polynomial with respect to the size of the initial input
- are there numerical problems that are strong NP-complete?
  - YES
  - 3-Partition, Bin-Packing, ...
- Attention! if $A \leq_{\mathrm{P}} B$ and A is weakly NP-complete, then we only prove that B is weakly NP-complete

The 3-Dimensional Matching problem

3-DM

Input: three sets A, B, C of vertices of the same cardinality $|A| = |B| = |C| = n$, a set $M \subseteq A \times B \times C$ of hyper-edges (triangles)

Question: is there a set $M' \subseteq M$ such that $|M'| = n$ and all vertices appear exactly once in M' ?

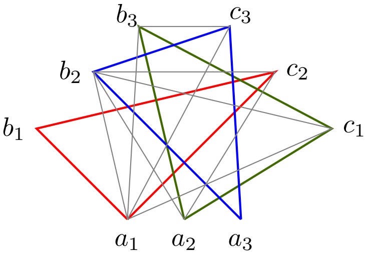


The 3-DIMENSIONAL MATCHING problem

3-DM

Input: three sets A, B, C of vertices of the same cardinality $|A| = |B| = |C| = n$, a set $M \subseteq A \times B \times C$ of hyper-edges (triangles)

Question: is there a set $M' \subseteq M$ such that $|M'| = n$ and all vertices appear exactly once in M' ?



- 3-DM is NP-COMPLETE in the strong sense

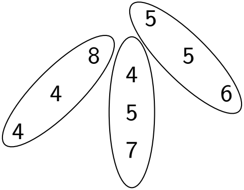
The 3-Partition problem

3-Partition

Input: a set of positive integers $S = \{s_1, s_2, \ldots, s_{3n}\}$, where $\sum_{s_i \in S} = n \cdot t$ and $\frac{t}{4} \leq s_i \leq \frac{t}{2}$ for each $s_i \in A$

Question: can S be partitioned into n disjoint sets $S_1, S_2, \ldots, S_n$ such that $\sum_{s_i \in S_j} s_i = t$, for $1 \leq j \leq n$?

- observation: each S_j should have exactly 3 integers



$t = 16$

- 3-Partition is NP-complete in the strong sense

Another problem

BIN-PACKING

Input: a set of items A, a size $s(a)$ for each $a \in A$, a positive integer capacity C, and a positive integer k

Question: is there a partition of A into disjoint sets $A_1, A_2, \ldots, A_k$ such that the total size of the elements in each set A_j does not exceed the capacity C, i.e., $\sum_{a \in A_j} s(a) \leq C$?

Show that this problem is NP-COMPLETE
Is it strongly or weakly NP-COMPLETE?
(try to give the strongest result)