

Computational Models - Lecture 12¹

Handout Mode

Iftach Haitner and Yishay Mansour.

Tel Aviv University.

June 11/13, 2012

¹Based on frames by Benny Chor, Tel Aviv University, modifying frames by Maurice Herlihy, Brown University.

Talk Outline

- Reminder - poly-time reductions, NP-completeness and SAT
- SAT is NP-Complete
- $SAT \leq_P 3SAT$
- 2SAT
- Hamiltonian Path Is NP-Complete
- SUBSET-SUM Is NP-Complete
- Partition is NP-Complete
- Bin-Packing is NP-Complete
- Integer-Programming is NP-Complete

- Sipser, 7.4–7.5

Section 1

Reminder

Reminder – Poly-time Reductions

Definition 1

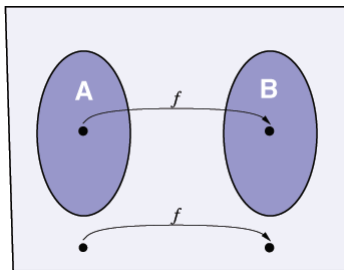
A language A is **poly-time mapping reducible** to B , denoted $A \leq_p B$, if exists poly-time computable function f such that

$$w \in A \iff f(w) \in B.$$

for every $w \in \Sigma^*$.

The function f is called a **polynomial-time reduction** from A to B .

The mapping f **efficiently** converts questions about membership in A to membership in B .



Reminder – NP Completeness

Definition 2 (\mathcal{NP} -complete)

A language B is **NP-complete**, if

- $B \in \mathcal{NP}$, and
- Every $A \in \mathcal{NP}$ is **poly-time** reducible to B

We let \mathcal{NPC} denote the class of all NP-complete languages.

$$A_{\text{NP}} = \{ \langle M, x, 1^n \rangle : M \text{ is a Turing Machine} \\ \wedge \exists c \in \Sigma^* \text{ s.t. } M(x, c) \text{ accepts within } n \text{ steps} \}$$

Theorem 3

$$A_{\text{NP}} \in \mathcal{NPC}$$

Theorem 4

Assume that $B \in \mathcal{NP}$, $A \in \mathcal{NPC}$ and $A \leq_p B$, then $B \in \mathcal{NPC}$.

Boolean Formulas and SAT

A **Boolean** formula is an expression involving Boolean variables and operations.

$$\phi = (\bar{x} \wedge y) \vee (x \wedge \bar{z})$$

Definition 5 (satisfiable formula)

A formula is **satisfiable**, if some assignment of 0s and 1s to the variables makes the formula evaluate to 1.

The formula $\phi = (\bar{x} \wedge y) \vee (x \wedge \bar{z})$ is satisfiable by the assignment

$$x = 0$$

$$y = 1$$

$$z = 0$$

The language of satisfied formulas:

$$\text{SAT} = \{ \langle \phi \rangle : \phi \text{ is a satisfiable Boolean formula} \}$$

Section 2

SAT is NP-Complete

$SAT \in \mathcal{NP}$

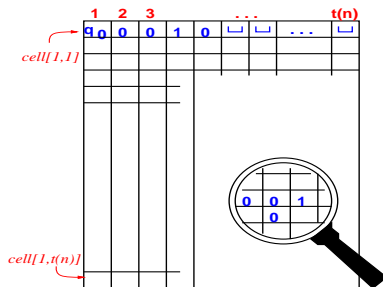
Theorem 6 (Cook-Levin, early 70s)

$SAT \in \mathcal{NP}$.

Proof's idea:

- Suppose $L \in \mathcal{NP}$ and let $M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)$ be an n^c -time **NTM** that accepts L (i.e., $M(w)$ runs in non-deterministic time $|w|^c$).
- We consider the n^c -by- n^c **tableau** that describes the **computation history** of M on input w .
- We create a formula whose satisfying assignments describes **accepting** such tableaux

The Configuration-History Tableau



- Row 1 in tableau represents **initial configuration** of M on input w .
- Row i in tableau represents **i -th configuration** in a computation of M on input w .

A Formula Simulating the Tableau

- We construct a Boolean CNF formula $\phi_{M,w}$ that “mimics” the tableau.
- Given the string w , it takes $O(n^{2c})$ steps to construct $\phi_{M,w}$.
- The following property holds:
$$\phi_{M,w} \in \text{SAT} \text{ iff } M \text{ accepts } w.$$
- The mapping $w \mapsto \phi_{M,w}$ is a poly-time reduction from L to SAT , establishing $L \leq_P \text{SAT}$.
- We still got a few small details to take care of...

The formula $\phi_{M,w}$

- Let $S = Q \cup \Gamma$ be the alphabet of the configuration history
- $\phi_{M,w}$ uses Boolean variables $\{x_{i,j,s}\}_{i,j \in [n^c], s \in S}$:
- Idea: $x_{i,j,s} = 1$ iff the j -th cell in i 'th configuration contains the symbol s :
 - ▶ Given a satisfying assignment for $\phi_{M,w}$, we construct an accepting configuration history for $M(w)$, by setting the j -th cell in i 'th configuration to s , iff $x_{i,j,s} = 1$.
 - ▶ Given an accepting configuration history for $M(w)$, we construct a satisfying assignment for $\phi_{M,w}$ by $x_{i,j,s} = 1$, iff the j -th cell in the i 'th configuration is s .
- The formula $\phi_{M,w}$ is of the form:

$$\phi_{M,w} = \phi_{\text{Start}(w)} \wedge \phi_{\text{Cell}(M)} \wedge \phi_{\text{Accept}(M)} \wedge \phi_{\text{Move}(M)}$$

The formula $\phi_{\text{Start}(w)}$

$\phi_{\text{Start}(w)}$ guarantees that the first row encodes the initial configuration (i.e., $q_0 w$).

$$\begin{aligned} \phi_{\text{start}(w)} = & X_{1,1,q_0} \wedge X_{1,2,w_1} \wedge X_{1,3,w_2} \wedge \cdots \wedge X_{1,n+1,w_n} \\ & \wedge X_{1,n+2,\sqcup} \wedge \cdots \wedge X_{1,n^c,\sqcup} \end{aligned}$$

The formula $\phi_{\text{Cell}(M)}$

$\phi_{\text{Cell}(M)}$ guarantees that the variables encode **legal** configurations:

- Each cell (i, j) has at least one letter: $\bigvee_{s \in S} x_{i,j,s}$.
- No cell (i, j) has two or more letters $\bigwedge_{s \neq s' \in S} \overline{x_{i,j,s} \wedge x_{i,j,s'}}$.
- Together:

$$\phi_{\text{Cell}(M)} = \bigwedge_{i,j} \left[\left(\bigvee_{s \in S} x_{i,j,s} \right) \wedge \left(\bigwedge_{s \neq s' \in S} \overline{x_{i,j,s} \wedge x_{i,j,s'}} \right) \right]$$

The formula $\phi_{\text{Accept}(M)}$

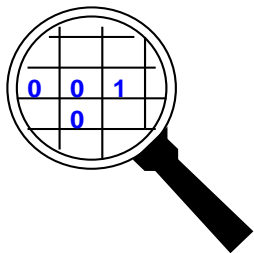
$\phi_{\text{Accept}(M)}$ guarantees that some row encodes an accepting configuration (i.e., $uq_a v$):

$$\phi_{\text{Accept}(M)} = \bigvee_{i,j} x_{i,j,q_a}$$

The formula $\phi_{\text{Move}(M)}$

$\phi_{\text{Move}(M)}$ is the “heart” of $\phi_{M,w}$. To construct it, we employ **locality** of computations.

- To determine contents of tableau entry (i, j) (cell j in configuration i), only the contents of **three** tableau entries (from configuration $i - 1$), $(i - 1, j - 1)$, $(i - 1, j)$, $(i - 1, j + 1)$, and M 's table, are needed.
- If head not in area, **nothing changes**. And and if it is, changes are local and are **determined** by δ .



The formula $\phi_{\text{Move}(M)}$, 2

- Assume that $\delta(q_1, a) = \{(q_1, b, R)\}$ and $\delta(q_1, b) = \{(q_2, c, L), (q_2, a, R)\}$.
- Legal** 2×3 configurations:

a	q_1	b
q_2	a	c

a	q_1	b
a	a	q_2

a	a	q_1
a	a	b

a	b	a
a	b	q_2

b	b	b
c	b	b

a	a	b
a	a	b

- Illegal** 2×3 configurations:

a	b	a
a	a	a

a	q_1	b
q_1	a	a

b	q_1	b
q_2	b	q_2

The formula $\phi_{\text{Move}(M)}$, 3

Let C be the set of legal 2×3 configuration tables.

For each entry $(i, j) \in [n^c] \times [n^c]$ and $c \in C$, let $\phi_{\text{Move}, i, j, c}$ be the formula that takes the value 1 iff the 2×3 table of cells in the Tableau whose upper-left corner is (i, j) is c .

For instance, for entry $(1, 1)$ and $c =$

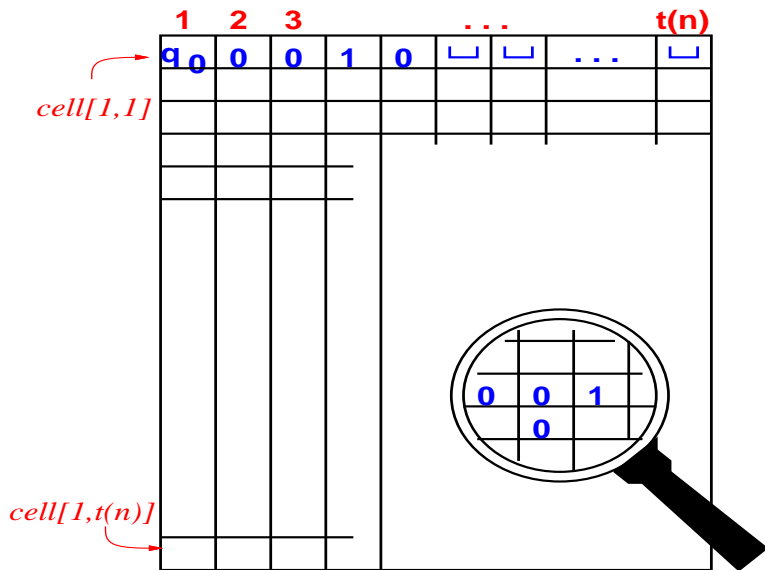
a	q_1	b
q_2	a	c

$$\begin{aligned} \phi_{\text{Move}, 1, 1, c} = & x_{1,1,a} \wedge x_{1,2,q_1} \wedge x_{1,3,b} \wedge \\ & x_{2,1,q_2} \wedge x_{2,2,a} \wedge x_{2,3,c} \end{aligned}$$

Finally,

$$\phi_{\text{Move}(M)} = \bigwedge_{(i,j)} \bigvee_{c \in C} \phi_{\text{Move}, i, j, c}$$

The Configuration History Tableau in Perspective



Correctness of Reduction

- The transformation $w \mapsto \phi_{M,w}$ is computable in time $O(n^{2c})$.
- An assignment satisfying $\phi_{\text{Cell}(M)} \wedge \phi_{\text{Start}(w)} \wedge \phi_{\text{Move}(M)}$ corresponds to a **valid computation** of M on w .
- An assignment satisfying **in addition** $\phi_{\text{Accept}(M)}$, corresponds to an **accepting computation** of M on w .
- Therefore M accepts w iff $\phi_{M,w} \in \text{SAT}$.
- For complete details, consult Sipser chapter 7.4.



Section 3

3SAT Is NP-Complete

SAT and 3SAT

- $\text{SAT} = \{\langle \phi \rangle : \phi \text{ is a satisfiable Boolean formula}\}$
- A Boolean formula is in **conjunctive normal form** (CNF) if it consists of **clauses**, connected with \wedge s.
For example: $(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6})$
- A Boolean formula is in **k-CNF form**, if it is a **CNF** formula, and all clauses have **k** literals.
- $\text{3SAT} = \{\langle \phi \rangle : \phi \text{ is satisfiable 3CNF formula}\}$.
- $\text{CSAT} = \{\langle \phi \rangle : \phi \text{ is a satisfiable CNF Boolean formula}\}$
- One can slightly modify the proof of Cook-Levin theorem we just seen, to prove that $\text{CSAT} \in \mathcal{NP}$!
- Hence, to prove that $\text{3SAT} \in \mathcal{NP}$, it suffices to prove that $\text{CSAT} \leq_P \text{3SAT}$.

CSAT \leq_P 3SAT

- The reduction maps CNF formulae to 3CNF ones “clause by clause”.

A clause with d literals is mapped to d clauses, built on the original literals together with $(d - 3)$ new ones.

- For example: $\phi = (x_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee x_4 \vee x_8) \mapsto$
 $\phi_3 = (x_1 \vee \bar{x}_2 \vee y_1) \wedge (\bar{y}_1 \vee \bar{x}_3 \vee y_2) \wedge (\bar{y}_2 \vee x_4 \vee x_8)$

Claim 7

ϕ has a satisfying assignment iff ϕ_3 does.

Proof's idea:

\Leftarrow An assignment satisfying ϕ_3 cannot “rely” on new literals alone – at least one original literal must be satisfied.

\Rightarrow An assignment satisfying ϕ makes at least one literal per clause **happy**. In the “ ϕ_3 clause” of this literal the new variable is under no constraints. This enables propagation to a satisfying assignment that “relies” on new vars alone in rest of ϕ_3 clauses.


SAT \leq_P 3SAT, 2

$$\phi = (x_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee x_4 \vee x_8) \mapsto$$

$$\phi_3 = (x_1 \vee \bar{x}_2 \vee y_1) \wedge (\bar{y}_1 \vee \bar{x}_3 \vee y_2) \wedge (\bar{y}_2 \vee x_4 \vee x_8)$$

Claim 8

ϕ has a satisfying assignment iff ϕ_3 does.

- Doing the above mapping to **each** clause of a **CNF** formula, we get a **3CNF** that is satisfied iff the original one is.
- Since this mapping is polynomial time (**why?**), we get **CSAT \leq_P 3SAT**. 

Section 4

2-SAT

Definition 9 (2CNF)

A Boolean formula is in **2CNF** if it is a **CNF** formula, and all terms have at most **two** literals.

For example

$$(x_1 \vee \bar{x}_2) \wedge (\bar{x}_5 \vee x_6) \wedge (\bar{x}_6 \vee \bar{x}_4)$$

$$2SAT\{\{\phi\} \mid \phi \text{ is satisfiable 2CNF formula}\}$$

Question 10

Is $2SAT \in \mathcal{NPC}$?

Well, turns out $2SAT \in \mathcal{P}$!. For details, though, you'll have to refer to the algorithms (formerly, efficiency of computations) course.

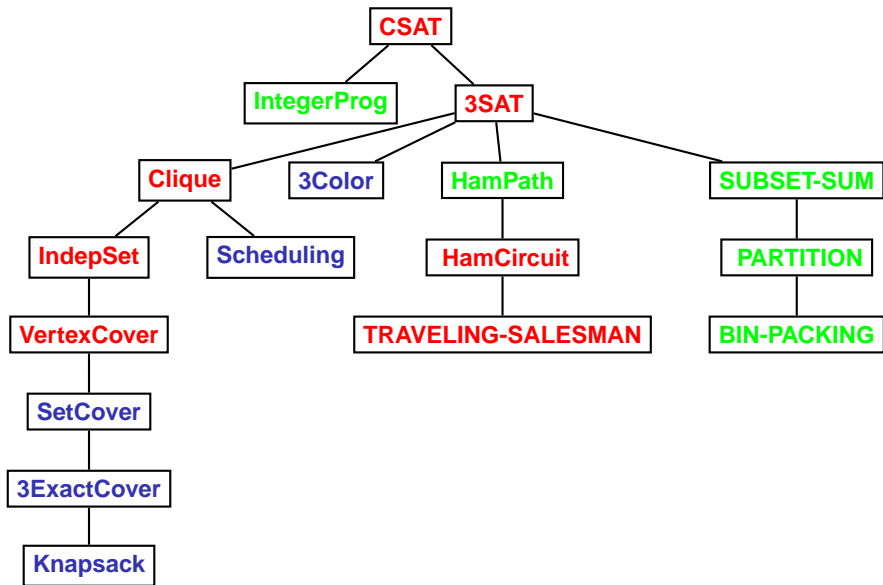
Section 5

Summary so Far

Summary So Far

- We have seen that **SAT** and **CSAT** are NP-complete.
- **CSAT** \leq_P **3SAT**
- **3SAT** \leq_P **CLIQUE**
- **CLIQUE** \leq_P **IND-SET**
- In recitation: **CLIQUE** \leq_P **VC** (Vertex Cover)
- Hence, all of the above languages are NP-complete
- Full list of NP-complete languages contains hundreds or thousands of known NP-complete problems (from combinatorics, operation research, VLSI design, computational geometry, bioinformatics, ...).
- NP-completeness of new and of old problems is still established these days.

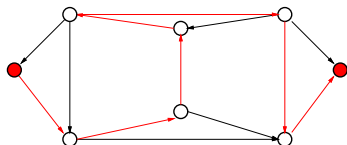
Chains of Reductions: NPC Problems (in red, seen so far, in green today.)



Section 6

Hamiltonian Path Is NP-Complete

(Directed) Hamiltonian Path



A **Hamiltonian path** in a directed G visits each node **exactly** once.

$\text{HAMPATH} = \{ \langle G, s, t \rangle : G \text{ has Hamiltonian path from } s \text{ to } t \}$

- We have seen that $\text{HAMPATH} \leq_P \text{HAMCIRCUIT}$ and $\text{HAMPATH} \leq_P \text{TRAVELING-SALESMAN}$
- Will now show

Theorem 11

$3\text{SAT} \leq_P \text{HAMPATH}$

- thus establishing

Theorem 12

$\text{HAMPATH}, \text{HAMCIRCUIT}, \text{TRAVELING-SALESMAN} \in \mathcal{NPC}$

3SAT \leq_P HAMPATH

For any 3CNF formula ϕ , we construct a directed graph G with vertices s and t , such that ϕ is satisfiable iff there is a directed Hamiltonian path from s to t .

Turn to a separate pdf presentation:

<http://tau-cm.wdfiles.com/local--files/course-schedule/ham-reduction-new.pdf>

Section 7

SUBSET-SUM Is NP-Complete

Reminder - SUBSET-SUM

$$\text{SUBSET-SUM} = \{ \langle S = \{x_1, \dots, x_k\}, t \rangle : \exists \{y_1, \dots, y_\ell\} \subseteq S : \sum_{j=1}^{\ell} y_j = t \}$$

Example 13

- ▶ $\{4, 11, 16, 21, 27\}, 25 \in \text{SUBSET-SUM}$ because $4 + 21 = 25$.
- ▶ $\{4, 11, 16, 21, 27\}, 26 \notin \text{SUBSET-SUM}$

- Will now show

Theorem 14

$3\text{SAT} \leq_P \text{SUBSET-SUM}$.

thus establishing

Theorem 15

$\text{SUBSET-SUM} \in \mathcal{NPC}$.

3SAT \leq_P SUBSET-SUM

Let ϕ be 3-CNF a boolean formula with

- variables x_1, \dots, x_ℓ
- clauses C_1, \dots, C_k .

We “encode” ϕ into a set S containing $2\ell + 2k$ numbers in decimal notation, and a “target” number t , such that

$$\phi \in \text{3SAT} \iff (S, t) \in \text{SUBSET-SUM}$$

Variable Encoding

Each variable x_i is encoded as two $(\ell + k)$ -digit numbers y_i and z_i :
Each decimal representation has two parts.

- Left-hand part, ℓ digits: for both y_i and z_i , the i 'th digit is 1, rest are 0s
- Right-hand part, k digits: one digit for each clause
 - ▶ j^{th} digit of y_i is 1 if C_j contains x_i
 - ▶ j^{th} digit of z_i is 1 if C_j contains \bar{x}_i
 - ▶ rest are 0s

Example: $\phi = (x_1 \vee \bar{x}_2 \vee \dots) \wedge \dots \wedge (\bar{x}_1 \vee \bar{x}_2 \vee \dots)$

	1	2	...	ℓ	C_1	...	C_k
y_1	1	0	...	0	1	...	0
z_1	1	0	...	0	0	...	1
y_2	0	1	...	0	0	...	0
z_2	0	1	...	0	1	...	1

Clause Encoding

Each clause C_j is encoded into two equal $(\ell + k)$ -digit numbers g_j and h_j :

for both g_j and h_j , the $(\ell + j)$ 'th digit is 1, rest are 0.

	1	...	ℓ	C_1	C_2	...	C_k
\vdots							
g_1	0	...	0	1	0	...	0
h_1	0	...	0	1	0	...	0
g_2	0	...	0	0	1	...	0
h_2	0	...	0	0	1	...	0

Target Encoding

The target t is assigned the following $(\ell + k)$ -digit number

- first ℓ digits are 1
- last k digits are 3

	1	...	ℓ	C_1	C_2	...	C_k
\vdots							
t	1	...	1	3	3	...	3

Overall Encoding

	1	2	...	ℓ	C_1	C_2	...	C_k
y_1	1	0	...	0	1	0	...	0
z_1	1	0	...	0	0	0	...	1
y_2	0	1	...	0	0	0	...	0
z_2	0	1	...	0	1	0	...	1
\vdots								
g_1	0	0	...	0	1	0	...	0
h_1	0	0	...	0	1	0	...	0
g_2	0	0	...	0	0	1	...	0
h_2	0	0	...	0	0	1	...	0
\vdots								
t	1	1	...	1	3	3	...	3

$$S = \{y_1, z_1, y_2, z_2, \dots, y_\ell, z_\ell, g_1, h_1, g_2, h_2, \dots, g_k, h_k\}.$$

Transformation takes $O(n^2)$ time.

ϕ is satisfiable $\implies (S, t) \in \text{SUBSET-SUM}$

Fix a satisfying assignment to ϕ , construct an **initial** solution V to (S, t) as follows: If $x_j = 1$ add y_j to V ; otherwise, add z_j to V .

- Each of the **first** ℓ digits of $\sum_{v \in V} v$ is **1** (as in t)
- For $1 \leq j \leq k$, consider the $(\ell + j)$ 'th digit of $\sum_{v \in V} v$. Note that
 - ▶ **at least 1** and **not more** than **3** literals in C_j are 1
 \implies the $(\ell + j)$ 'th digit is between **1** and **3**.
 - ▶ Add "enough" $\{h_i, g_i\}$ to V to bring this digit up to **3**.
(Doing this does **not** effect the other digits of $(\sum_{v \in V} v)$).

Hence, $\sum_{v \in V} v = t$. ♣

$(S, t) \in \text{SUBSET-SUM} \implies \phi$ is Satisfiable

Fix a solution V to (S, t) .

- 1 Summation causes **no carries**
 - ▶ All digits in the numbers of S are either **0** or **1**.
 - ▶ Since ϕ is in **3CNF**, each "column" contains at most **five 1s** (why?)
- 2 For each $i \in [\ell]$, the subset V contains **one** of y_i or z_i , but **not both**.
- 3 For each $j \in [k]$, the subset V contains **at least one** $\{y_1, z_1, \dots, y_\ell, z_\ell\}$ whose $(\ell + j)$ 'th bit is **1**
 - ▶ Each of final k "columns" sums to **3**.
 - ▶ The set $\{g_j, h_j\}$ contributes at most **2**
 - ▶ So at least one must come from literal taking the value **1**

Here is the satisfying assignment.

For $i \in [\ell]$: if $y_i \in V$ then $x_i = 1$; otherwise (i.e., $z_i \in V$), $x_i = 0$

The assignment satisfies ϕ because

- ▶ No contradicting assignments (see **Item 2**)
- ▶ All clauses are satisfied (see **Item 3**)



Section 8

Partition is NP-Complete

Partition

An instance of the problem

Set of integers P

Question: Can we partition the set P into set U and $P \setminus U$ such that $\sum_{u \in U} u = \sum_{u \notin U} u$?

$$\text{PARTITION} = \{ \langle P \rangle : \exists U \subseteq P \text{ s.t. } \sum_{u \in U} u = \sum_{u \notin U} u \}$$

Observation

PARTITION is a special case of **SUBSET-SUM** (in particular, $\text{PARTITION} \in \mathcal{NP}$)

SUBSET-SUM \leq_P PARTITION

The reduction:

- Input: Set of integers S and integer t
- Output: $P = S \cup \{b = 2A - t, c = A + t\}$, where $A = \sum_{s \in S} s$.

Note that the sum of numbers in P is $4A$.

Correctness:

- If there exists a solution V for (S, t) , then $V \cup \{b\}$ is a partition of P
- Assume U partitions P .
 - ▶ wlg. $b \in U$ and $c \notin U$
 - ▶ Then $U \setminus \{b\}$ is a solution for (S, t) .

Section 9

Bin-Packing is NP-Complete

Bin-Packing

An instance of the problem

- Set of integers S ; bin size B ; number of bins m

Question: Is there a partition of S to m subsets S_1, \dots, S_m , such that $\sum_{s \in S_j} x \leq B$ for every $j \in [m]$?

$\text{BIN-PACKING} = \{ \langle S, B, m \rangle : \exists \text{ partition } S_1 \dots S_m \text{ s.t.} \\ \forall j \in [m] \sum_{s \in S_j} s \leq B \}$

Theorem 16

$\text{PARTITION} \leq_P \text{BIN-PACKING}$

The reduction:

- Input: Set P
- Output: S , $B = \frac{1}{2} \sum_{s \in S} s$ and $m = 2$.

Section 10

Integer Programming (IP) is NP-Complete

Integer Programming (IP)

Definition 17

A **linear inequality** has the form

$$a_1x_1 + a_2x_2 + \dots + a_nx_n \leq b$$

where a_1, \dots, a_n, b are real **numbers**, and x_1, \dots, x_n are real **variables**.

The Integer Programming (IP) problem]:

Input: A set of linear inequalities with **integer** coefficients in variables

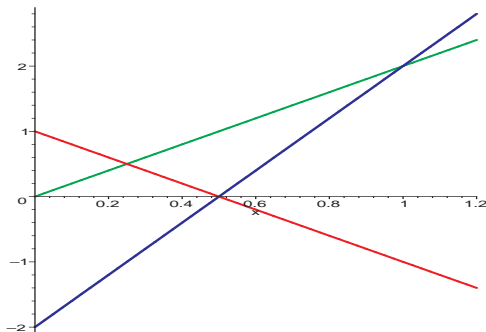
x_1, x_2, \dots, x_n .

Question: Does this set has an **integer solution** – all x_j are **integers**?

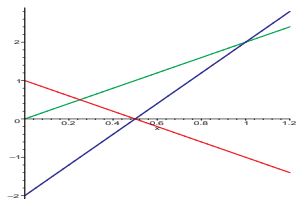
Integer Programming: Example

Consider the following system of linear inequalities

$$\begin{aligned}y &\leq 2x && \text{below green line} \\-2x + 1 &\leq y && \text{above red line} \\4x - 2 &\leq y && \text{above purple line} \\0 &\leq x \leq 1 \\0 &\leq y \leq 2\end{aligned}$$



Integer Programming: Example, 2



This set does have a **unique** solution: the right hand corner of the solid triangle, $(1, 2)$.

But if we change the constraint on y to $0 \leq y \leq 1$, then we'd have no solution with integer coordinates, even though there are many solutions with **rational**, or **real**, coordinates.

The Language IP

The language of integer linear inequalities with integer solution:

$$\text{IP} = \{ \langle e_1, \dots, e_m \rangle : \\ e_1, \dots, e_m \text{ are integer linear inequalities with (joint) integer solution} \}$$

Theorem 18

$\text{IP} \in \mathcal{NPC}$.

Question 19

Do we always have a **small enough** certificate?

Consider the following equations

$$\begin{aligned}x_0 &= 1 \\x_1 &= 2x_0 \\&\vdots \\x_n &= 2x_{n-1}\end{aligned}$$

The solution $x_n = 2^n$!

Question 20

Solving a linear set of n equations with n unknowns: $Ax = b$, where $|a_{i,j}|, |b_i| < M$. How large can $|x_j|$ get?

Answer: Recall Cramer's Rule, using determinants. $|A| < n!M^n$.
Same holds for inequalities.

CSAT = $\{\langle \phi \rangle : \phi \text{ is a satisfiable CNF Boolean formula}\}$

Let ϕ be a CNF formula with m clauses and n variables x_1, \dots, x_n .

We reduce ϕ to an IP instance E with $2n$ variables $x_1, y_1, \dots, x_n, y_n$, $m + 2n$ linear inequalities, and n linear equalities.

- Each x_i in ϕ corresponds to the variable x_i in E .
- Each \bar{x}_i in ϕ corresponds to the variable y_i in E .
- For each i , we add E the inequalities $x_i \geq 0$, $y_i \geq 0$, and the equality $x_i + y_i = 1$ (what do these three express?)
- For each clause C of ϕ , we add E the inequality $\sum_{z \in C} z \geq 1$ (what does this inequality express?)

For example, $(x_1 \vee \bar{x}_2 \vee \bar{x}_3 \vee x_4)$ is translated to $x_1 + y_2 + y_3 + x_4 \geq 1$.

Reduction Example

$\phi = (x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6})$ is translates to

$$x_1 + y_2 + y_3 + x_4 \geq 1$$

$$x_3 + y_5 + x_6 \geq 1$$

$$x_3 + y_6 \geq 1$$

$$x_1 \geq 0, y_1 \geq 0, x_1 + y_1 = 1$$

$$x_2 \geq 0, y_2 \geq 0, x_2 + y_2 = 1$$

$$x_3 \geq 0, y_3 \geq 0, x_3 + y_3 = 1$$

$$x_4 \geq 0, y_4 \geq 0, x_4 + y_4 = 1$$

$$x_5 \geq 0, y_5 \geq 0, x_5 + y_5 = 1$$

$$x_6 \geq 0, y_6 \geq 0, x_6 + y_6 = 1$$

CSAT \leq_P IP: Validity (sketch)

Should show

- 1 Reduction, g , is poly-time computable
 - 2 $\phi \in \text{CSAT} \implies g(\phi) \in \text{IP}$
 - 3 $g(\phi) \in \text{IP} \implies \phi \in \text{CSAT}$.
- 1 Poly time: easy (verify details!).
 - 2 Suppose $\phi \in \text{CSAT}$. Take a satisfying assignment.
If $x_i = 1$, in $g(\phi)$ assign $x_i = 1, y_i = 0$.
If $x_i = 0$, in $g(\phi)$ assign $x_i = 0, y_i = 1$.
 - ▶ "Sanity check" constraints (i.e., $x_i \geq 0, y_i \geq 0, x_i + y_i = 1$) are satisfied.
 - ▶ "Clause constraints" (i.e., $\sum_{z \in C} z \geq 1$) are satisfied due to at least one literal satisfied in each clause.
 - ▶ Implying $g(\phi) \in \text{IP}$.
 - 3 $g(\phi) \in \text{IP} \implies \phi \in \text{CSAT}$, is similar.

