CS 5321: Advanced Algorithms – NP-completeness of Some Number Theory Problems

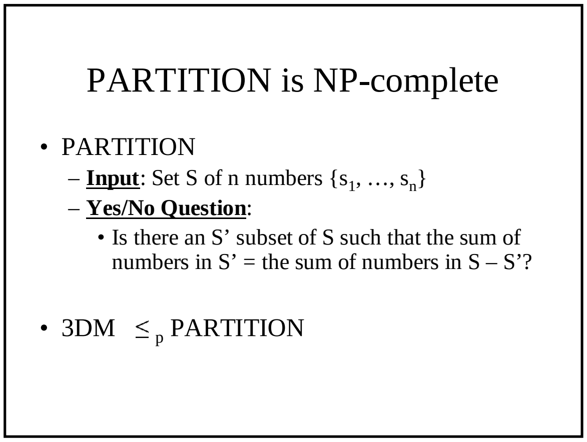
Ali Ebnenasir
Department of Computer Science
Michigan Technological University

Acknowledgement

- Abdulhossein Esfahanian
- Hector A. Villa-Martine

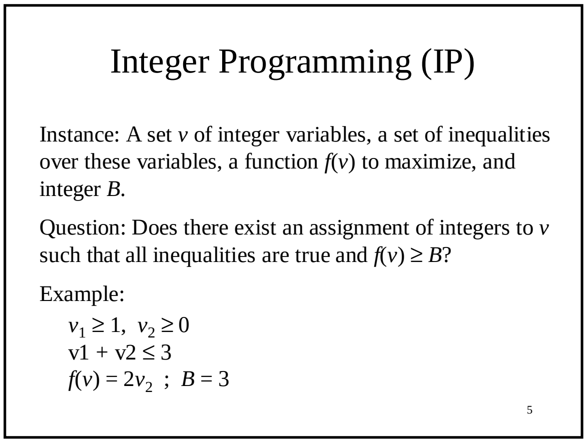
Number Problems

- Problems where the inputs are numbers
  - Prime number problem:
    - Input: Integer n
    - Yes/No Question: Is n prime?
  - Partition problem
    - Input: Set S of n numbers {s_1, …, s_n}
    - Yes/No Question: Is there an S’ subset of S such that the sum of numbers in S’ = the sum of numbers in S – S’.
- What is the input size for these problems?



PARTITION is NP-complete

- PARTITION
  - **Input**: Set S of n numbers $\{s_1, \ldots, s_n\}$
  - **Yes/No Question**:
    - Is there an S' subset of S such that the sum of numbers in S' = the sum of numbers in S – S'?
- 3DM $\leq_p$ PARTITION



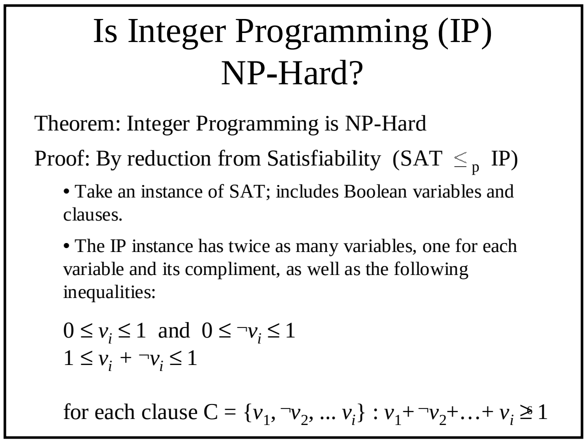
Integer Programming (IP)

Instance: A set *v* of integer variables, a set of inequalities over these variables, a function *f*(*v*) to maximize, and integer *B*.

Question: Does there exist an assignment of integers to *v* such that all inequalities are true and $f(v) \geq B$?

Example:

$v_1 \geq 1$, $v_2 \geq 0$
v1 + v2 ≤ 3
$f(v) = 2v_2$; $B = 3$



Is Integer Programming (IP) NP-Hard?

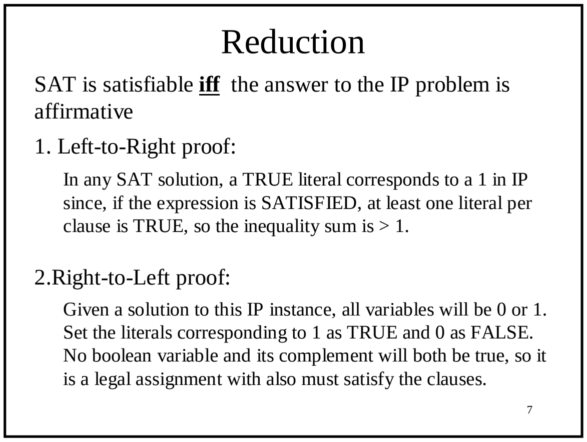
Theorem: Integer Programming is NP-Hard

Proof: By reduction from Satisfiability (SAT $\leq_p$ IP)

- Take an instance of SAT; includes Boolean variables and clauses.
- The IP instance has twice as many variables, one for each variable and its compliment, as well as the following inequalities:

$0 \leq v_i \leq 1$ and $0 \leq \neg v_i \leq 1$
$1 \leq v_i + \neg v_i \leq 1$

for each clause C = $\{v_1, \neg v_2, \ldots v_i\}$: $v_1 + \neg v_2 + \ldots + v_i \geq 1$



Reduction

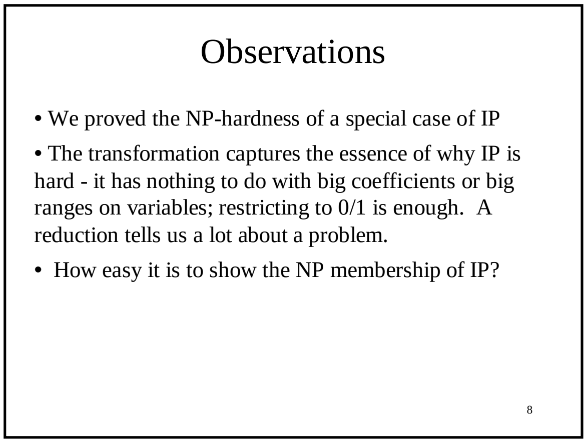
SAT is satisfiable **iff** the answer to the IP problem is affirmative

1. Left-to-Right proof:

In any SAT solution, a TRUE literal corresponds to a 1 in IP since, if the expression is SATISFIED, at least one literal per clause is TRUE, so the inequality sum is > 1.

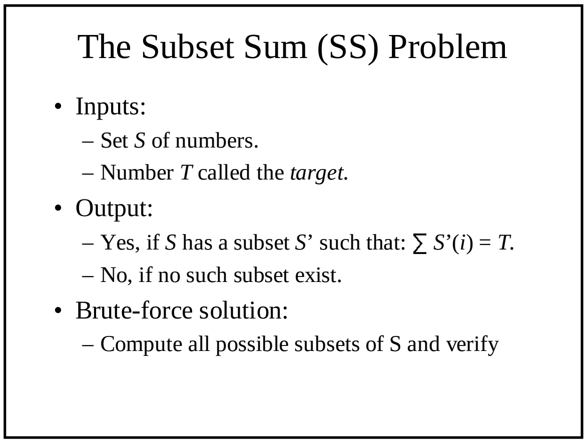
2.Right-to-Left proof:

Given a solution to this IP instance, all variables will be 0 or 1. Set the literals corresponding to 1 as TRUE and 0 as FALSE. No boolean variable and its complement will both be true, so it is a legal assignment with also must satisfy the clauses.



Observations

- We proved the NP-hardness of a special case of IP
- The transformation captures the essence of why IP is hard - it has nothing to do with big coefficients or big ranges on variables; restricting to 0/1 is enough. A reduction tells us a lot about a problem.
- How easy it is to show the NP membership of IP?



The Subset Sum (SS) Problem

- Inputs:
  - – Set *S* of numbers.
  - – Number *T* called the *target*.
- Output:
  - – Yes, if *S* has a subset *S*’ such that: $\sum S'(i) = T$.
  - – No, if no such subset exist.
- Brute-force solution:
  - – Compute all possible subsets of S and verify

Example

- Inputs:
  - S = {10, 9, 15, 22, 39, 5, 15}
  - T = 42
- Output:
  - Yes, because {15, 22, 5} is a subset of S and 15 + 22 + 5 = 42

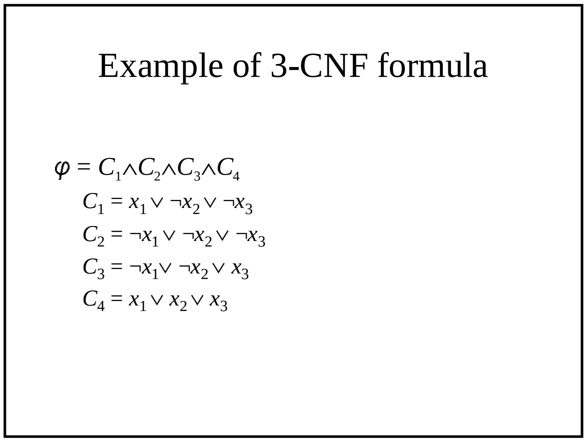
Subset Sum is NP-Complete

- Subset Sum (SS) is in NP
  - What is a certificate?
  - Verifiable in polynomial time?
- 3-SAT $\leq_p$ SS

3-SAT

The 3-SAT problem:

- 3-CNF formula φ has:
  - k clauses $C_1, C_2, \ldots, C_k$
  - n propositional variables $x_1, x_2, \ldots, x_n$
- φ has the form $\varphi = C_1 \wedge C_2 \wedge \ldots \wedge C_k$.
  - Each clause C_i has at most three variables $C_i = l_i \vee l_j \vee l_k$, where $i \neq j \neq k$ and l_i is a literal denoting x_i or $\neg x_i$
- x_i and $\neg x_i$ cannot be in the same clause
- Each variable x_i appears in at least one clause.



Example of 3-CNF formula

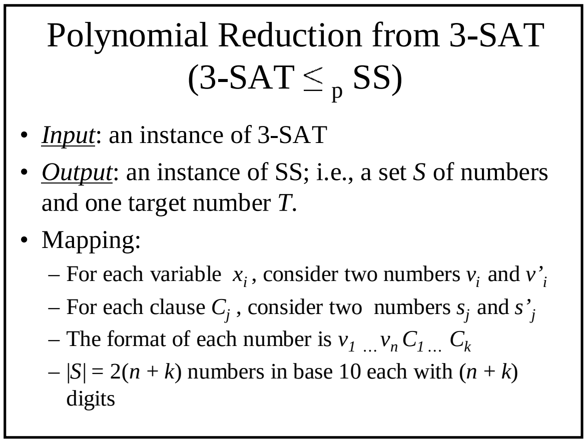
$\varphi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$

$C_1 = x_1 \vee \neg x_2 \vee \neg x_3$

$C_2 = \neg x_1 \vee \neg x_2 \vee \neg x_3$

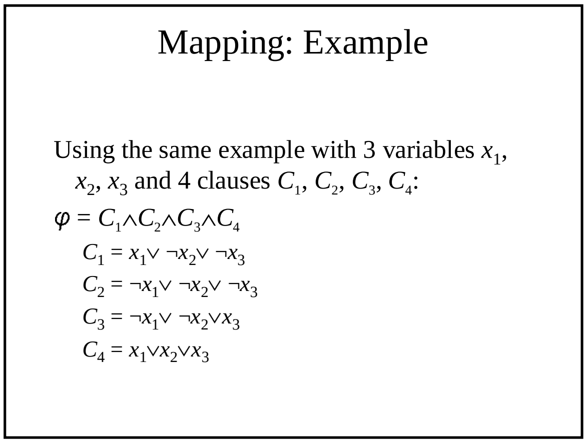
$C_3 = \neg x_1 \vee \neg x_2 \vee x_3$

$C_4 = x_1 \vee x_2 \vee x_3$



Polynomial Reduction from 3-SAT ($3\text{-SAT} \leq_p \text{SS}$)

- *Input*: an instance of 3-SAT
- *Output*: an instance of SS; i.e., a set S of numbers and one target number T.
- Mapping:
  - For each variable x_i, consider two numbers v_i and v'_i
  - For each clause C_j, consider two numbers s_j and s'_j
  - The format of each number is $v_1 \ldots v_n C_1 \ldots C_k$
  - $|S| = 2(n + k)$ numbers in base 10 each with $(n + k)$ digits



Mapping: Example

Using the same example with 3 variables x_1, x_2, x_3 and 4 clauses C_1, C_2, C_3, C_4:

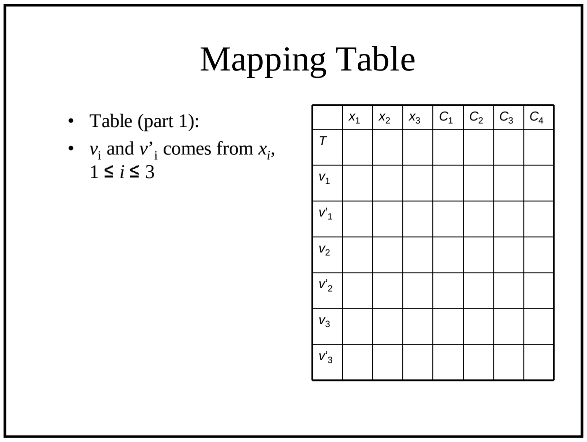
$\varphi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$

$C_1 = x_1 \vee \neg x_2 \vee \neg x_3$

$C_2 = \neg x_1 \vee \neg x_2 \vee \neg x_3$

$C_3 = \neg x_1 \vee \neg x_2 \vee x_3$

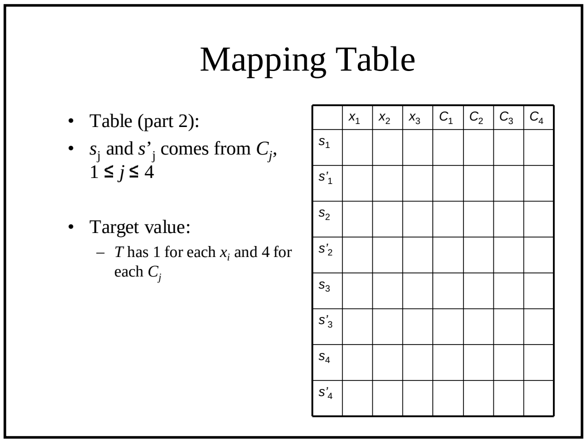
$C_4 = x_1 \vee x_2 \vee x_3$



Mapping Table

- Table (part 1):
- v_i and v'_i comes from x_i, $1 \le i \le 3$

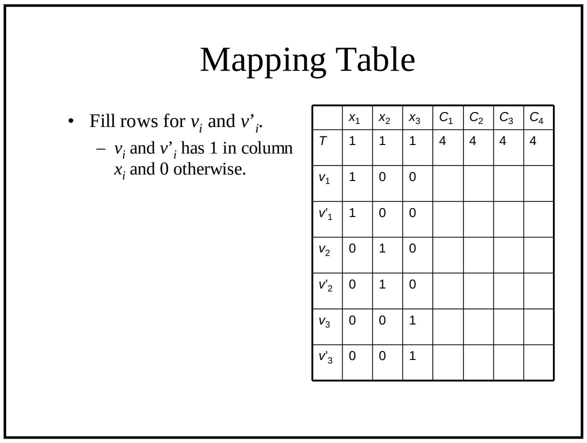
	x_1	x_2	x_3	C_1	C_2	C_3	C_4
T							
v_1							
v'_1							
v_2							
v'_2							
v_3							
v'_3							



Mapping Table

- Table (part 2):
- s_j and s'_j comes from C_j, $1 \le j \le 4$
- Target value:
  - T has 1 for each x_i and 4 for each C_j

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
s_1							
s'_1							
s_2							
s'_2							
s_3							
s'_3							
s_4							
s'_4							



Mapping Table

- Fill rows for v_i and v'_i.
  - v_i and v'_i has 1 in column x_i and 0 otherwise.

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
T	1	1	1	4	4	4	4
v_1	1	0	0				
v'_1	1	0	0				
v_2	0	1	0				
v'_2	0	1	0				
v_3	0	0	1				
v'_3	0	0	1				

Mapping Table

- Corresponding to clauses:
  - v_i has 1 in column C_j if x_i appears in C_j, and 0 otherwise.
  - v'_i has 1 in column C_j if $\neg x_i$ appears in C_j, and 0 otherwise.

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
T	1	1	1	4	4	4	4
v_1	1	0	0	1	0	0	1
v'_1	1	0	0	0	1	1	0
v_2	0	1	0	0	0	0	1
v'_2	0	1	0	1	1	1	0
v_3	0	0	1	0	0	1	1
v'_3	0	0	1	1	1	0	0

$C_1 = x_1 \vee \neg x_2 \vee \neg x_3$

$C_2 = \neg x_1 \vee \neg x_2 \vee \neg x_3$

$C_3 = \neg x_1 \vee \neg x_2 \vee x_3$

$C_4 = x_1 \vee x_2 \vee x_3$

Reduction from SAT (14/15)

- For s_j and s'_j:
- s_j has 1 in column C_j, and 0 otherwise
- s'_j has 2 in column C_j, and 0 otherwise

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
s_1	0	0	0	1	0	0	0
s'_1	0	0	0	2	0	0	0
s_2	0	0	0	0	1	0	0
s'_2	0	0	0	0	2	0	0
s_3	0	0	0	0	0	1	0
s'_3	0	0	0	0	0	2	0
s_4	0	0	0	0	0	0	1
s'_4	0	0	0	0	0	0	2

SS Instance: Example

- Interpret each row as a base 10 integer.
- S = {1001001, 1000110, 100001, 101110, 10011, 11100, 1000, 2000, 100, 200, 10, 20, 1, 2}.
- T = 1114444.

Reduction Correctness

- Claim:
  - The 3-SAT formula, φ, is satisfiable **if and only if** there is a subset $S' \subseteq S$ whose sum is T

Left-to-Right Proof

- Suppose there is an assignation to x_1, x_2, …, x_n, such that φ evaluates to true
- If x_i = *true*, then include v_i in S'
- If x_i = *false*, then include v'_i in S'

Left-to-Right Proof: Example

- $x_1 = 0$, $x_2 = 0$, $x_3 = 1$
- $S' = \{v_1', v_2', v_3\}$
- Z has the sum of the elements in S'.
- T is the target.
- We need to find how to make $Z = T$.

$\varphi = C_1 \wedge C_2 \wedge C_3 \wedge C_4$

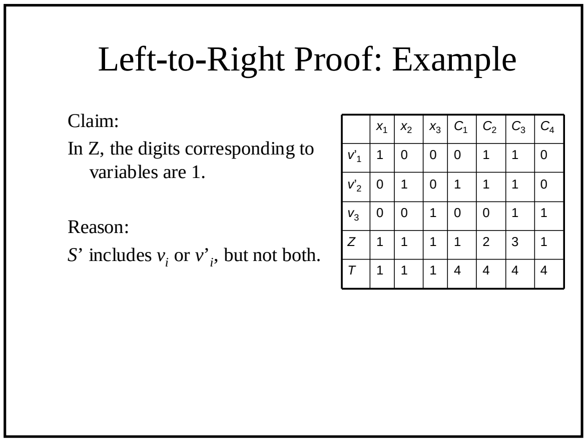
$C_1 = x_1 \vee \neg x_2 \vee \neg x_3$

$C_2 = \neg x_1 \vee \neg x_2 \vee \neg x_3$

$C_3 = \neg x_1 \vee \neg x_2 \vee x_3$

$C_4 = x_1 \vee x_2 \vee x_3$

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
v'_1	1	0	0	0	1	1	0
v'_2	0	1	0	1	1	1	0
v_3	0	0	1	0	0	1	1
Z	1	1	1	1	2	3	1
T	1	1	1	4	4	4	4



Left-to-Right Proof: Example

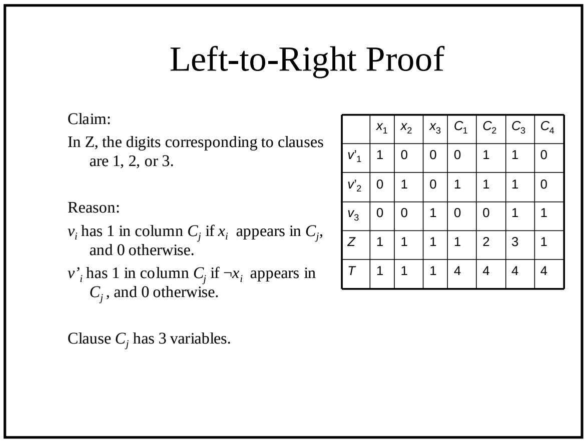
Claim:

In Z, the digits corresponding to variables are 1.

Reason:

S’ includes v_i or v'_i, but not both.

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
v'_1	1	0	0	0	1	1	0
v'_2	0	1	0	1	1	1	0
v_3	0	0	1	0	0	1	1
Z	1	1	1	1	2	3	1
T	1	1	1	4	4	4	4



Left-to-Right Proof

Claim:

In Z, the digits corresponding to clauses are 1, 2, or 3.

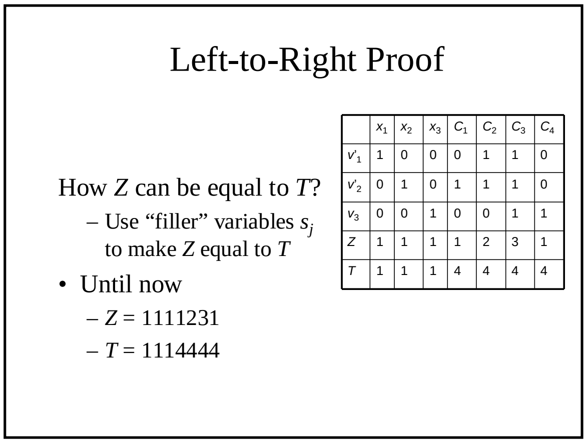
Reason:

v_i has 1 in column C_j if x_i appears in C_j, and 0 otherwise.

v'_i has 1 in column C_j if $\neg x_i$ appears in C_j, and 0 otherwise.

Clause C_j has 3 variables.

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
v'_1	1	0	0	0	1	1	0
v'_2	0	1	0	1	1	1	0
v_3	0	0	1	0	0	1	1
Z	1	1	1	1	2	3	1
T	1	1	1	4	4	4	4



Left-to-Right Proof

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
v'_1	1	0	0	0	1	1	0
v'_2	0	1	0	1	1	1	0
v_3	0	0	1	0	0	1	1
Z	1	1	1	1	2	3	1
T	1	1	1	4	4	4	4

How *Z* can be equal to *T*?

- Use “filler” variables s_j to make *Z* equal to *T*

- Until now
  - *Z* = 1111231
  - *T* = 1114444

Left-to-Right Proof

- s_j has 0 in digits from variables.
- s_j has 1 or 2 in digits from clauses.
- We can add both s_j and s'_j

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
s_1	0	0	0	1	0	0	0
s'_1	0	0	0	2	0	0	0
s_2	0	0	0	0	1	0	0
s'_2	0	0	0	0	2	0	0
s_3	0	0	0	0	0	1	0
s'_3	0	0	0	0	0	2	0
s_4	0	0	0	0	0	0	1
s'_4	0	0	0	0	0	0	2

Left-to-Right Proof

1. Add V_i to S' which are in the solution to $\boldsymbol{\varphi}$.
2. Use filler variables

S' = {1000110, 101110, 10011, 1000, 2000, 200, 10, 1, 2}

$Z = T$

Q.E.D. (*quod erat demonstrandum*)

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
v'_1	1	0	0	0	1	1	0
v'_2	0	1	0	1	1	1	0
v_3	0	0	1	0	0	1	1
S_1	0	0	0	1	0	0	0
S'_1	0	0	0	2	0	0	0
S'_2	0	0	0	0	2	0	0
S_3	0	0	0	0	0	1	0
S_4	0	0	0	0	0	0	1
S'_4	0	0	0	0	0	0	2
Z	1	1	1	4	4	4	4
T	1	1	1	4	4	4	4

Right-to-Left Proof

- Suppose there is a subset $S' \subseteq S$ that sums to T.
  - T is written as n 1's followed by k 4's
- Claim: There exists a truth-value assignment to x_i that satisfies φ

Right-to-Left Proof

- The n most significant digits of T are 1 (remember T is 1111…4444).
- Thus subset S' includes either v_i or v'_i for i = 1, 2, …, n

	X_{i-1}	X_i	X_{i+1}	C_i	Cj
V_{i-1}	1	0	0		
V'_{i-1}	1	0	0		
V_i	0	1	0		
V'_i	0	1	0		
V_{i+1}	0	0	1		
V'_{i+1}	0	0	1		
T	1	1	1	4	4

Right-to-Left Proof

Truth-value assignment:

- if $v_i \in S'$
  - $x_i = 1$
- If $v'_i \in S'$
  - $x_i = 0$
- Claim:
  - This assignment satisfies φ.

Right-to-Left Proof

In the k least significant digits of T

- The “filler” variables sum up at most to 3
- Thus, the digit corresponding to C_j in T, must include at least one v_i or v'_i.

	x_1	x_2	x_3	C_1	C_2	C_3	C_4
s_1	0	0	0	1	0	0	0
s'_1	0	0	0	2	0	0	0
s_2	0	0	0	0	1	0	0
s'_2	0	0	0	0	2	0	0
s_3	0	0	0	0	0	1	0
s'_3	0	0	0	0	0	2	0
s_4	0	0	0	0	0	0	1
s'_4	0	0	0	0	0	0	2
T	1	1	1	4	4	4	4

Right-to-Left Proof

If $v_i \in S'$ then
$x_i = 1$ and C_j is satisfied since
$C_j = (\ldots \vee x_i \vee \ldots)$

	x_{i-1}	x_i	x_{i+1}	C_{j-1}	C_j	C_{j+1}
v_i	0	1	0	0	1	0
Filler	0	0	0	3	3	3
T	1	1	1	4	4	4

Right-to-Left Proof

If $v'_i \in S'$ then $x_i = 0$ and C_j is satisfied since $C_j = (\ldots \vee \neg x_i \vee \ldots)$

Q.E.D.

	x_{i-1}	x_i	x_{i+1}	C_{j-1}	C_j	C_{j+1}
v'_i	0	1	0	0	1	0
Filler	0	0	0	3	3	3
T	1	1	1	4	4	4

Knapsack Problem

- 0-1 Knapsack optimization problem
  - Input
    - Capacity K
    - n items with weights w_i and values v_i
  - Yes/No Question
    - Find a set of items S such that
      - the sum of weights of items in S is at most K
      - the sum of values of items in S is maximized
- We gave a polynomial-time dynamic programming solution for this problem
- Show that Partition $\leq_p$ Knapsack
- Is this a contradiction?

Definining subproblems

- Define P(i,w) to be the problem of choosing a set of objects from the *first* i objects that maximizes value subject to weight constraint of w.
  - – Impose an *arbitrary* ordering on the items
- V(i,w) is the value of this set of items
- Original problem corresponds to V(n, K)

Recurrence Relation/Running Time

- $V(i,w) = \max (V(i-1,w-w_i) + v_i, V(i-1, w))$
  - – A maximal solution for P(i,w) either
    - uses item i (first term in max)
    - or does NOT use item i (second term in max)
- V(0,w) = 0 (no items to choose from)
- V(i,0) = 0 (no weight allowed)
- What is the running time of this solution?
  - – Number of table entries:
  - – Time to fill each entry:

Example

$w_A = 2$ $v_A = \$40$

$w_B = 3$ $v_B = \$50$

$w_C = 1$ $v_C = \$100$

$w_D = 5$ $v_D = \$95$

$w_E = 3$ $v_E = \$30$

Items

Weight

	A	B	C	D	E
1	$0	$0	$100	$100	$100
2	$40	$40	$100	$100	$100
3	$40	$50	$140	$140	$140
4	$40	$50	$150	$150	$150
5	$40	$90	$150	$150	$150
6	$40	$90	$190	$195	$195
7	$40	$90	$190	$195	$195
8	$40	$90	$190	$235	$235
9	$40	$90	$190	$245	$245
10	$40	$90	$190	$245	$245

Weak NP-completeness

- An NP-complete problem is called “weakly NP-complete” if it has in its description one or more integer parameters and the corresponding problem where these parameters are represented in unary is in P.
- An NP-complete problem is strongly NP-complete if the problem is still NP-complete even if integer parameters are encoded in unary

Select the Right Source Problem

3-SAT: The old reliable. When none of the other problems seem to work, this is the one to come back to.

Integer Partition: A good choice for *number* problems.

3-Partition: A good choice for proving “*strong*” NP-completeness for *number* problems.

- *Strongly NP-complete*: A problem that remains NP-complete even if its numerical parameters are bounded by a polynomial in input size

Vertex Cover: A good choice for *selection* problems.

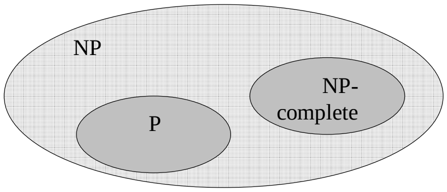
Hamiltonian Path: A good choice for *ordering* problems.

Minimum Set Cover

Problem: Given a family F of subsets $\{S_1, S_2, \ldots, S_m\}$ of a universal set $U = \{u_1, u_2, \ldots, u_n\}$ and an integer k, is it possible to choose only k elements of F such that the union of these elements is U.

- NP membership?
  - given a subset of sets, we can count them, and show that all elements of U are included.
- NP-hardness?
  - What problem should we choose to reduce this time?

State of Knowledge (assuming $P \neq NP$)



If $P \neq NP$ then $NPI = NP - (P \cup NP\text{-complete}) \neq \phi$

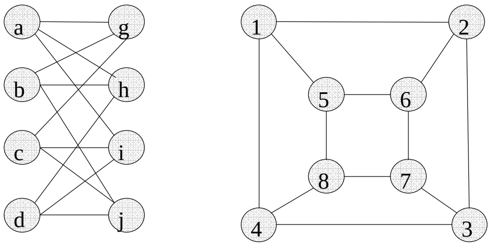
NPI is the class of problems having intermediate complexity

Class of NPI Problems

- Assuming $P \neq NP$
  - Any language in NP that is not known to be in P and there is not NP-hardness proof for it yet
  - Certain problems that have withstood the test of time
    - Graph Isomorphism
    - Composite numbers
    - Linear programming

GRAPH ISOMORPHISM (GI)

- INSTANCE: Graphs $G=(V, E)$ and $G'=(V, E')$
- DECISION:
  - Are G and G' isomorphic?
  - Is there a one-to-one function $f: V \rightarrow V$ such that $(u, v) \in E$ if and only if $(f(u), f(v)) \in E'$?
- NP membership?
- NP-hardness?



SUB-GRAPH ISOMORPHISM (SGI)

- INSTANCE: Graphs G1=(V1, E1) and G2=(V2, E2)
- DECISION:
  - Does G1 have a sub-graph isomorphic to G2?
- NP membership?
- NP-hardness?
  - CLIQUE is a special case of SGI!
  - Thus SGI is NP-hard!

GRAPH ISOMORPHISM (GI)

- INSTANCE: Graphs G=(V, E) and G'=(V, E')
- DECISION:
  - Are G and G' isomorphic?
  - Is there a one-to-one function f: V →V such that (u, v) ∈ E if and only if (f(u), f(v)) ∈ E'?
- After so many years, we do not yet know whether GI is in P or NP-complete.
- What is it about GI?

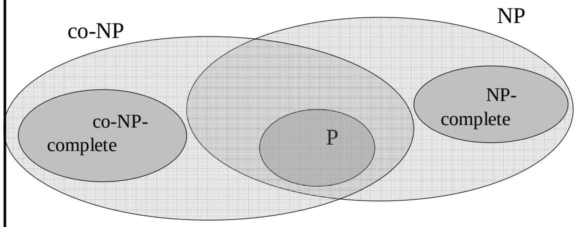
GRAPH ISOMORPHISM (GI)

- GI is much more constrained than already known NP-complete problems
- When reducing a known problem to an unknown problem, we seem to need some redundancy in the target problem
- GI lacks such a redundancy!
    - For example, in SGI, adding new edges to G1 does not affect the fact that G1 includes (or does not include) a subgraph isomorphic to G2
    - We do not have such a leeway with GI!

Complement Problems

- The complement of a problem is the problem with reverse answers to the decision problem
- E.g., what is the complement of SAT, CLIQUE, VERTEX COVER, etc.
- co-NP: class of all problems whose complement is in NP
- co-P: class of all problems whose complement is in P
- What is co-NP-complete?

NP vs. co-NP



co-NP

NP

co-NP-complete

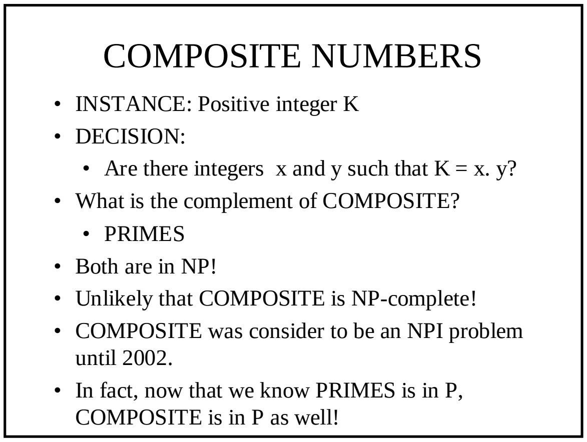
P

NP-complete

Conjecture: co-NP ≠ NP

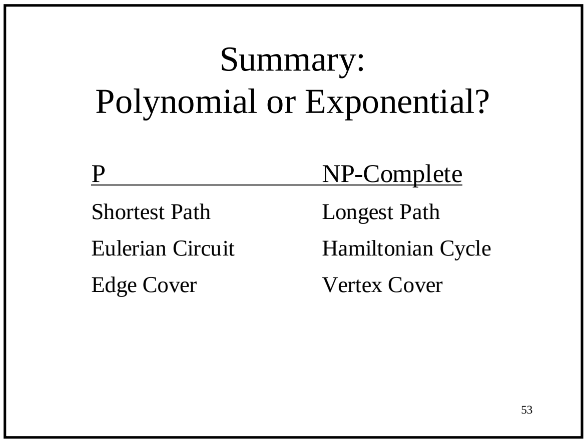
If co-NP ≠ NP then P ≠NP

If the complement of an NP-complete problem is in NP, then co-NP = NP. Why?



COMPOSITE NUMBERS

- INSTANCE: Positive integer K
- DECISION:
  - Are there integers x and y such that K = x. y?
- What is the complement of COMPOSITE?
  - PRIMES
- Both are in NP!
- Unlikely that COMPOSITE is NP-complete!
- COMPOSITE was consider to be an NPI problem until 2002.
- In fact, now that we know PRIMES is in P, COMPOSITE is in P as well!



Summary:
Polynomial or Exponential?

P	NP-Complete
Shortest Path	Longest Path
Eulerian Circuit	Hamiltonian Cycle
Edge Cover	Vertex Cover