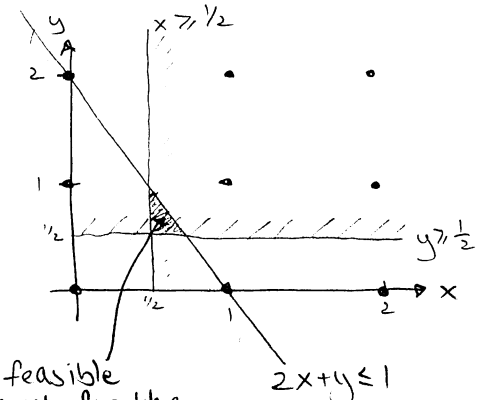
Integer Programming

Linear program = set of linear inequalities with rational coefficients
- feasible if there is a solution to all

Integer program is a linear program, but with the variables required to be integers.

Example: $x \geq 1/2$ $\quad 2x+y \leq 1$

$y \geq 1/2$



Theorem: Linear Programming $\in P$

Theorem: Integer Programming is NP-Complete.

Proof:

Since we can easily check a proposed integer solution, the problem is in NP.

For showing that it is NP hard, reduce from 3-SAT.

For each clause like $(x_1 \vee \bar{x}_2 \vee x_3)$, add inequality

$$x_1 + (1 - x_2) + x_3 \geq 1.$$

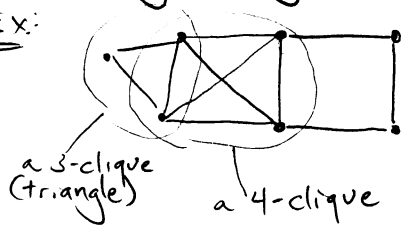
Furthermore, add inequalities $0 \leq x_i \leq 1$ to ensure that each variable has only two possible assignments: 0 (false) or 1 (true). ✓

Note: Just because a problem is NP-complete does not mean every instance is hard.

- E.g., 3-SAT is NP-complete but 2-SAT $\in P$
  and every 2-SAT formula also gives a 3-SAT formula:
  $(x_1 \vee x_2) \wedge (\bar{x}_3 \vee x_1) \ldots \longrightarrow (x_1 \vee x_2 \vee x_2) \wedge (\bar{x}_3 \vee x_1 \vee x_1)$
- Example (Valiant '04)
  - Counting # of satisfying assignments to planar 3-NotAllEqual formulas is in P.
  - $\#_2$Pl-Rtw-Mon-3CNF is NP-hard
  - $\#_7$Pl-Rtw-Mon-3CNF is in P [Valiant '06]
    planar read-twice monotone
- Average-case complexity
  Ajtai '96: a lattice problem in NP is hard on average provided another NP problem is hard on worst case.

Def: CLIQUE = $\{\langle G, k\rangle \mid$ G is an undirected graph with a "k-clique", i.e., a set of k vertices any two of which are connected by an edge$\}$.

Ex:



Theorem: CLIQUE is NP-complete.

Proof:

First, CLIQUE $\in$ NP; here is a poly-time verifier:

V On input $\langle G, k\rangle, c$:

1. Test that c is a set of k nodes of G
2. Test that G includes an edge between every pair of vertices in c.
3. If both pass, accept, else reject.

(To get a nondeterministic Turing machine, just guess c first.)

To show that CLIQUE is NP-hard, we want to show that every NP language L reduces in poly-time to CLIQUE. But since every L already reduces to 3-SAT (3-SAT is NP-hard) it is enough to reduce 3-SAT to CLIQUE.

Reduction Input: a k-clause 3-SAT formula

$$\phi = (a_1 \vee b_1 \vee c_1) \wedge (a_2 \vee b_2 \vee c_2) \wedge \cdots \wedge (a_k \vee b_k \vee c_k)$$

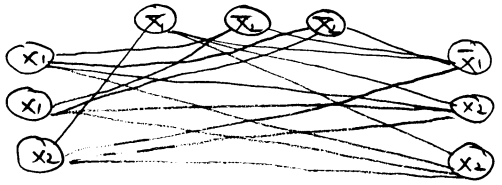
Output: $\langle G, k\rangle$, where G is given as follows

3 vertices for every clause

edges between all pairs of vertices for different clauses, except those for opposite literals, e.g., x_2 and $\overline{x_2}$

Example: $\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$

G:



Claim: ϕ is satisfiable iff G has a k-clique.

Claim: ϕ is satisfiable **if and only if** G has a k-clique.

Proof: Write $\phi(x_1, \ldots, x_m) = C_1 \wedge C_2 \wedge \cdots \wedge C_k$, with $C_i = z_{i1} \vee z_{i2} \vee z_{i3}$

$\Rightarrow$: Let ϕ be satisfiable by truth assignment τ

Choose one true z_{ij} in each clause C_i

and add the corresponding vertex v_{ij} to set S

Clearly $|S| = k$

Claim: S is a clique.

Why? Consider any $v_{ij}, v_{kl} \in S$

- $i \neq k$ since only one vertex per clause is in S
- $z_{ij} \neq \overline{z}_{kl}$ as both z_{ij} and z_{kl} are true

$\Rightarrow$ edge (v_{ij}, v_{kl}) is in G. ✓

$\Leftarrow$: Let S be a clique of size k in G.

Define Truth assignment τ by z_{ij} = TRUE if $v_{ij} \in S$

(any unspecified variables may be set arbitrarily)

Claim 1: τ is a valid truth assignment.

Claim 2: ϕ is satisfied by τ.

Proof of Claim 1:

$v_{ij}, v_{kl} \in S \Rightarrow$ edge (v_{ij}, v_{kl}) in G

$\Rightarrow z_{ij} \neq \overline{z}_{kl}$

$\Rightarrow$ a variable and its negation cannot both be set TRUE (no contradictions) □

Proof of Claim 2:

- No two vertices of S are in same clause

$\& \; |S| = \#$ of clauses $= k$

$\Rightarrow$ Each clause has one variable in S

$\Rightarrow$ Each clause has ≥ 1 true literal

$\Rightarrow$ Each clause is satisfied ✓ □

DONE ✓

Independent Set (I.S.)

- Graph $G = (V, E)$
- $S \subseteq V$ is independent if for all $u, v \in S$ the edge (u, v) is not in G.

I.S Instance: Graph $G = (V, E)$, integer k

Problem: Does G have I.S. of size $\geq k$?

Theorem: Independent Set is NP-complete.

Proof:

1) I.S. $\in$ NP — Guess S and verify ✓
2) I.S. is NP-hard

$$\text{CLIQUE} \leq_{\text{poly}} \text{I.S.}$$

reduction f:

Input $G = (V, E)$, integer k

Output $\overline{G}$ complement of G, integer k

edge (u, v) is in G
$\Leftrightarrow$ edge (u, v) is not in G

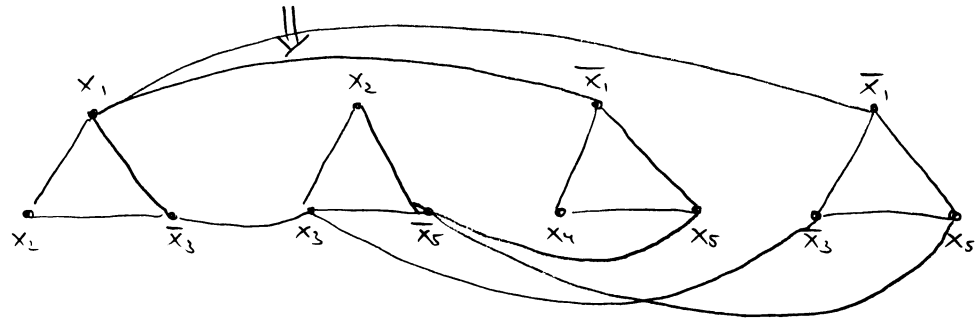
Clearly Reduction f is poly-time computable.

Lemma: G has k-clique $\Leftrightarrow$ $\overline{G}$ has k-I.S. ✓

☑

Of course, following CLIQUE, we can directly reduce 3-SAT $\leq_{\text{poly}}$ I.S., eg,

$$\phi = (x_1 \vee x_2 \vee \overline{x}_3) \wedge (x_2 \vee x_3 \vee \overline{x}_5) \wedge (\overline{x}_1 \vee x_4 \vee x_5) \wedge (\overline{x}_1 \vee \overline{x}_3 \vee x_5)$$



Alternatively, observe that the complement of an independent set is a vertex cover, so we could have used reductions

$$\text{3-SAT} \leq_{\text{poly}} \text{VERTEX-COVER} \leq_{\text{poly}} \text{I.S.} \leq_{\text{poly}} \text{CLIQUE}.$$

Graph coloring problems

Def: A coloring of a graph is an assignment of colors to its nodes so that no two adjacent nodes are assigned the same color.

Theorem: Let

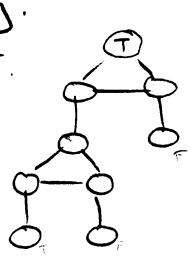
$$3COLOR = \{ \langle G \rangle \mid \text{the nodes of } G \text{ can be colored with three colors so no two nodes joined by an edge have the same color} \}$$

Then 3COLOR is an NP-complete problem.

Proof sketch: $3COLOR \in NP$ clearly.

For NP-hardness, reduce from 3SAT.

Use the gadgets



✓

Theorem (Blum & Kagar '97): Assuming G is a 3-colorable graph, there is a polynomial-time algorithm that colors it using at most

$$n^{3/14} \cdot (\log n)^{O(1)}$$

colors.

Theorem (Khanna, Linial, Safra '93): If $P \neq NP$, there is no poly-time algorithm that colors 3-colorable graphs using 4 colors.

– huge gap!

Subset Sum Instance Collection of numbers $x_1, \ldots, x_k$ (allowing repetition)
Target number t

Problem Is there a subcollection that adds to t?

$$\text{SUBSET-SUM} = \left\{ \langle S, t \rangle \;\middle|\; S = \{x_1, \ldots, x_k\} \text{ and for some } \{y_1, \ldots, y_\ell\} \subseteq \{x_1, \ldots, x_k\}, \text{ we have } \Sigma y_i = t \right\}.$$

Ex.: $\langle \{4, 11, 16, 21, 27\}, 25 \rangle \in$ SUBSET-SUM $(4 + 21 = 25)$

SUBSET-SUM $\in$ NP clearly It is not obvious that the complement SUBSET-SUM is in NP. Def: coNP = class of languages whose complements are in NP.

Theorem: SUBSET-SUM is NP-complete.

Proof: By a reduction from 3-SAT

Idea: ① Represent variable x_i by two numbers y_i and z_i
Show that exactly one must be in any subcollection summing to the target — encoding x_i's truth value
② Add constraints also for every clause

Let ϕ be a boolean formula with variables $x_1, \ldots, x_\ell$, clauses $c_1, \ldots, c_k$.
eg., $(x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee x_3 \vee \ldots) \wedge \ldots \wedge (\bar{x}_3 \vee \ldots \vee \ldots)$

Becomes

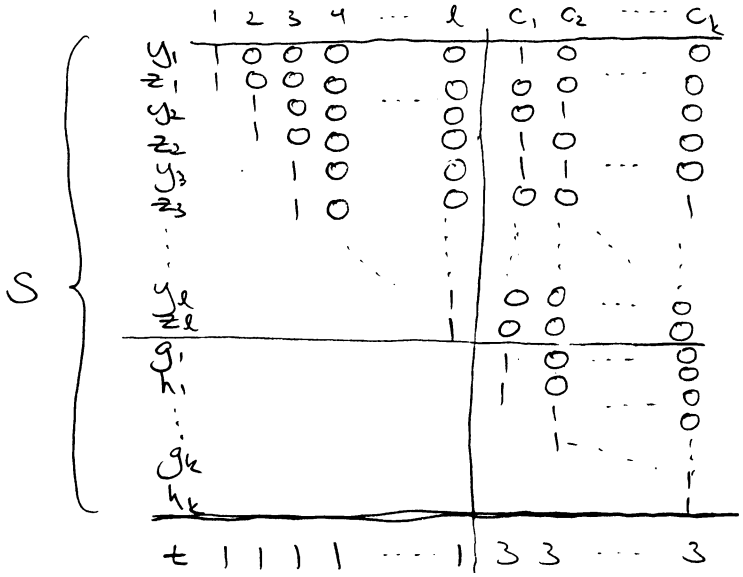


table size is $\sim (k+\ell)^2$, so reduction is computable in poly-time

Claim: ϕ is satisfiable if and only if a subset of S sums to t. ✓

(Observe. Since there are at most 5 ones in any column, "carries" are impossible (base 10).)

Time Hierarchy Theorem

Theorem [simple case]:
There exists a language L that is decidable in time $O(n^{1.5})$ but not in time $O(n)$.

Proof: Let D be the following Turing machine:

D | Input $x \in \{0,1\}^n$

① Run the universal Turing machine U to simulate M_x (Turing machine encoded by x) on input $x = \langle M_x \rangle$ for $n^{1.4}$ steps.

② If M_x outputs an answer, then output the opposite! Otherwise reject.

Let $L = L(D)$.

- L is decidable in time $O(n^{1.5})$ ✓
- Claim: L is not decidable in time $O(n)$.

  Proof: Assume otherwise

  Let M be a Turing machine that decides L (i.e. $M(x) = D(x)$ for all $x \in \{0,1\}^*$) and on inputs of length $n \geq n_0$ runs in time $\leq c \cdot n$.

  Let $\langle M \rangle$ be an encoding of M of length $n \geq n_0$, large enough so $c \cdot n \leq n^{1.4}$.
  (such an encoding exists because every Turing machine is represented by infinitely many strings)

Then $\langle M \rangle \in L \iff D$ accepts M
$\iff M$ rejects $\langle M \rangle$ in $\leq n^{1.4}$ steps
$\iff M$ rejects $\langle M \rangle$ (since time $c \cdot n \leq n^{1.4}$ ✓)
$\iff \langle M \rangle \notin L$ ↯ contradiction! □

More generally,

Theorem: If f, g are time-constructible functions, $f(n) \log f(n) = o(g(n))$,
Then $\text{TIME}(f(n)) \subsetneq \text{TIME}(g(n))$

("Time constructible" means there is a Turing machine that on input 1^n writes $1^{f(n)}$ on its tape in $O(f(n))$ time)