Chapter 9. NP-COMPLETENESS

Topics include:

- Reducibilities among combinatorial problems
- Turing machines
- Feasible computations and the complexity classes P, NP
- Polynomial-time bounded reducibilities
- NP-Completeness and Cook's Theorem
- Further NP-Complete problems.

9.1. Reducibilities Among Combinatorial Problems.

Q. Squaring matrices is a special case of matrix multiplication.
Is squaring easier than multiplication?

Prop. If squaring $n \times n$ matrices can be done in $T(n)$ steps, then multiplying $n \times n$ matrices can be done in $O(T(n))$ steps.

Assumption: $T(2n) \leq 8\,T(n)$.

Proof. Given two $n \times n$ matrices A, B consider

$$\begin{pmatrix} 0 & A \\ B & 0 \end{pmatrix}^2 = \begin{pmatrix} AB & 0 \\ 0 & BA \end{pmatrix} \qquad \square$$

Q. For Boolean matrices how to compare matrix multiplication vs. transitive closure ?

We show their complexities are of same order of magnitude.

Prop. If computing trans. closure of $n \times n$ Boolean matrices can be done in $T(n)$ steps, where $T(3n) \leq 27\,T(n)$, then multiplying $n \times n$ Boolean matrices can be done in $O(T(n))$ steps.

Pf. Given two $n \times n$ Boolean matrices A, B consider the $3n \times 3n$ Boolean matrix

$$X = \begin{pmatrix} 0 & A & 0 \\ 0 & 0 & B \\ 0 & 0 & 0 \end{pmatrix}$$

Since

$$X^2 = \begin{pmatrix} 0 & 0 & AB \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad X^3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} = X^4 = \cdots$$

We have:

$$X^+ = \begin{pmatrix} 0 & A & AB \\ 0 & 0 & B \\ 0 & 0 & 0 \end{pmatrix}$$

□

Prop. If multiplying 2 $n \times n$ Boolean matrices can be done in $M(n)$ steps, where $M(n)$ satisfies $M(2n) \geq 4M(n)$, then computing the transitive closure of an $n \times n$ Boolean matrix can be done in $O(M(n))$ steps.

Pf. Since $X^* = \mathbb{I} + X^+$, we may consider computing X^* instead of X^+.

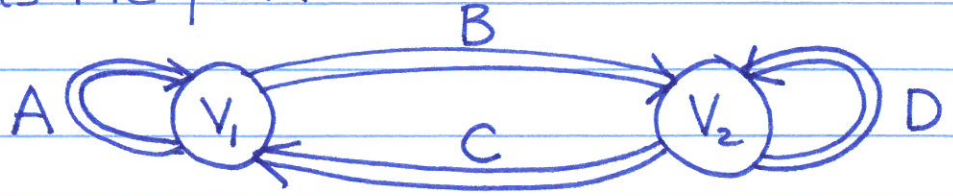
W.l.o.g. let X be an $n \times n$ Boolean matrix and $n = 2^k$ for some k.

We partition X into 4 submatrices of dimension $\frac{n}{2} \times \frac{n}{2}$ (or $2^{k-1} \times 2^{k-1}$):

$$X = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

Let G be the directed graph represented by X where $V = \{1, .., n\}$ is the set of vertices.

Then V can be partitioned as $V = V_1 \dot{\cup} V_2$ where $V_1 = \{1, .., \frac{n}{2}\}$, $V_2 = \{\frac{n}{2}+1, ..., n\}$. And G has the form



Let $X^* = \begin{pmatrix} E & F \\ G & H \end{pmatrix}$

Then $E = (A + \underbrace{B \cdot \underbrace{D^*}_{T_1} \cdot C}_{T_2})^*$

and $F = E \cdot \underbrace{B \cdot D^*}_{T_2}$

$G = \underbrace{D^* \cdot C}_{T_3} \cdot E$

$H = \underbrace{D^*}_{T_1} + \underbrace{\underbrace{D^* \cdot C}_{T_3} \cdot E \cdot \underbrace{B \cdot D^*}_{T_2}}_{G}$.

Now, E, F, G, H can be computed as:

$T_1 = D^* \qquad T_2 = B \cdot T_1 \qquad T_3 = T_1 \cdot C$

$E = (A + T_2 \cdot C)^*$

$F = E \cdot T_2$

$G = T_3 \cdot E$

$H = T_1 + G \cdot T_2$

Let $T(2^k)$ = time to compute X^*. Then

$$T(1) = 1$$

$$T(2^k) \leq 2T(2^{k-1}) + 6 \cdot M(2^{k-1}) + 2 \cdot (2^{k-1})^2$$

addition of $\frac{n}{2} \times \frac{n}{2}$ matrices

Claim. $T(2^k) \leq c \cdot M(2^k)$ for some c.

Pf. (Claim)

First note that from $M(2n) \geq 4M(n)$

we have $M(2^{k-1}) \geq 4 \cdot M(2^{k-2})$

$$\vdots$$

$$\geq 4^{k-1} = 2^{2(k-1)}.$$

Hence,

$$T(2^k) \leq 2T(2^{k-1}) + 6 \cdot M(2^{k-1}) + 2 \cdot (2^{k-1})^2$$

(IH) $$\leq 2c\, M(2^{k-1}) + 6\, M(2^{k-1}) + 2\, M(2^{k-1})$$

$$\leq (2c+8)\, M(2^{k-1})$$

$$= (\tfrac{1}{2}c + 2) \cdot 4 \cdot M(2^{k-1})$$

$$\leq M(2^k)$$

$$\leq c\, M(2^k)$$

for $c \geq 4$ □

Cor. The complexity of computing the trans. closure of an $n \times n$ Boolean matrix is of the same order as the complexity of computing the product of two $n \times n$ Boolean matrices.

Remarks. (1) Squaring and multiplication of matrices are reducible to each other: they have the same order of complexity

(2) Matrix product is reducible to trans. closure: they also have the same order of complexity.

(3) For Boolean matrices these 3 probl. have the same order of complexity.

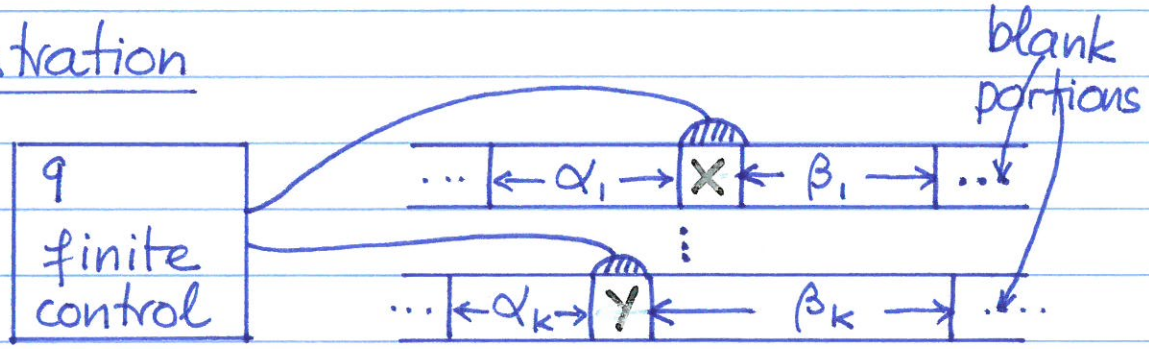
9.2. Turing Machines

A k-tape nondeterministic Turing machine (NTM for short) is a 7-tuple

$$M = (Q, \Sigma, \Gamma, \delta, q_0, F, B) \quad \text{s.t.}$$

- Q = finite set of states
- Σ = input alphabet
- Γ = tape alphabet, $\Gamma \supseteq \Sigma$
- $q_0 \in Q$ is start state
- $F \subseteq Q$ is set of final states
- $B \in \Gamma - \Sigma$ is blank symbol
- $\delta: Q \times \Gamma^k \longrightarrow 2^{Q \times (\Gamma \times \{L,R,S\})^k}$

Illustration



An instantaneous description (ID) of M is a tuple of form

symbols being scanned

$$(q, [\alpha_1, X\beta_1], \ldots, [\alpha_k, Y\beta_k]) \in Q \times (\Gamma^* \times \Gamma^*)^k$$

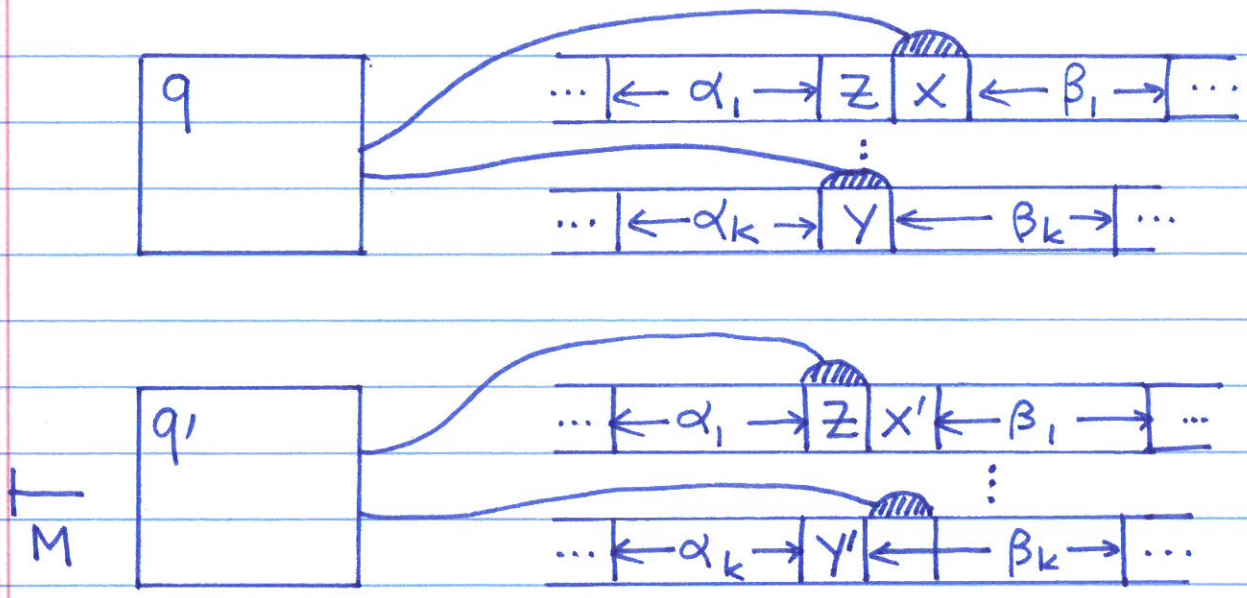
current state; content of tape 1; content of tape k

Suppose there is a transition

$$(q', (X', L), \ldots, (Y', R)) \in \delta(q, (X, \ldots, Y))$$

We write:

$$(q, [\alpha_1, Z, X\beta_1], \ldots, [\alpha_k, Y\beta_k]) \vdash_M (q', [\alpha_1, ZX'\beta_1], \ldots, [\alpha_k Y', \beta_k])$$



$\vdash_M$ is called the yield (one-step) relation of M.

Initial ID is $(q_0, [\varepsilon, w], [\varepsilon, \varepsilon], \ldots, [\varepsilon, \varepsilon])$

A final ID is $(q_f, [\alpha_1, \beta_1], \ldots, [\alpha_k, \beta_k])$ where $q_f \in F$.

$\vdash_M^*$ denote the reflexive and trans closure of $\vdash_M$.

The language accepted by M is

$$L(M) = \{ w \in \Sigma^* \mid C_0 \vdash_M^* C_f \}$$

where C_0 = initial ID with input w

C_f = a final ID.

An (N)TM M is of time complexity $T(n)$ if for every $w \in L(M)$ there is an accepting computation of length $\leq T(|w|)$.

Prop. If L is accepted by a k-tape (N)TM of time complexity $T(n)$, then L is accept. by a 1-tape (N)TM of time compl. $O(T(n)^2)$.

Prop. Let $T(n) \geq n$ be a fctn. Then every $T(n)$ time-bounded 1-tape NTM has an an equiv. $2^{O(T(n))}$ time-bounded 1-tape DTM.

Pf. Let N be a $T(n)$-time-bounded 1-tape NTM. We construct a 1-tape DTM D that simulates N by searching N's computation tree of depth $T(n)$. □

9.3. Feasible Computations & the Class P

Fact. All reasonable deterministic models of computation are polynomially equiv.

Def. P is the class of languages that can be accepted by polynomial-time bounded 1-tape DTMs

A D/N-TM is polynomial-time bounded if it is $p(n)$ time bounded for some polynomial $p(n)$.

Remark. The class P is important since:

(1) P is invariant for all computational models that are poly. equiv. to 1-tape DTMs.

(2) P corresponds to the class of problems that are practically solvable on a computer.

Examples: (1) Given a directed graph $G=(V,E)$ and $s,t \in V$, determined whether $s \rightsquigarrow t$.

(2) Determine if a given binary integer is prime.

(3) Given a directed graph $G=(V,E)$ and a weight function $w: E \to \mathbb{R}^+$ and $s,t \in V$, $x \in \mathbb{R}^+$, determine if $\exists$ a path $s \rightsquigarrow t$ with cost $\leq x$.

(4) Given 2 bin. integers determine if they are relatively prime

Computational vs. Decision Problems

Example (3) is a decision problem.

The related computational (optimization) problem is the shortest path proplem.

Note there is a close relationship between decision problems and optimization problems: the decision problem can be solved efficiently $\iff$ the computational problem can be solved efficiently.

We confine ourselves with decision problems in the theory of NP-completeness.

Def. NP is the class of languages accepted by polynomial-time bounded 1-tape NTMs.

Verification Algorithms

A verification algor A is a 2-argument alg. where:

- First argument is input string x
- Second arg. is a string y called certificate

A verifies an input x if $\exists$ certificate y s.t. $A(x,y) = 1$ (= Yes)

The language verified by a verif. alg. A is

$$L = \{ x \in \{0,1\}^* \mid \exists y \in \{0,1\}^* : A(x,y) = 1 \}$$

Ex. (Hamiltonian cycles)

A Hamiltonian cycle in an undirected graph $G = (V,E)$ is a simple cycle containing every vertex $v \in V$.

The Hamiltonian cycle problem is the probl. of determining for a given graph $G = (V,E)$ whether it contains a Hamiltonian cycle.

This probl. formulated as a language is

$$\text{HAM-CYCLE} = \{ \langle G \rangle \mid G \text{ has a Hamiltonian cycle} \}$$

$\langle G \rangle$ = bin. encoding of graph G.

A verification algor for HAM-CYCLE has as input $(\langle G \rangle, \langle L \rangle)$ where

- $\langle G \rangle$ = bin. encoding of graph $G=(V,E)$
- $\langle L \rangle$ = bin. encoding of a list of vertices.

The algor. verifies $\langle G \rangle$ by checking if L forms a simple cycle containing every vertex $v \in V$.

Prop. NP = class of languages that can be verified in polynomial time. □

Prop. $P \subseteq NP$ □

Major Question. $P \subsetneq NP$?

Ex. Problems in NP include:

(1) Hamiltonian Cycle

(2) Clique: A complete subgraph of an undirected graph is called a clique (every 2 nodes are connected by an edge).

The clique problem is to determine for an undirected graph $G=(V,E)$ and k whether G contains a clique of size k.

(3) Subset Sum: Given a set of positive integers $\{x_1, \ldots, x_k\}$ and an integer t, determine if there is a subset $\{y_1, \ldots, y_\ell\} \subseteq \{x_1, \ldots, x_k\}$ s.t. $\sum_{i=1}^{\ell} y_i = t$.

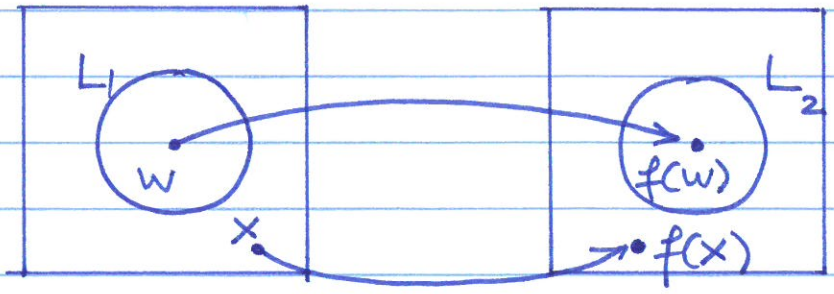
(Note that we allow multisets, i.e., an element may occur more than once.)

9.4. Polynomial-Time-Bounded Reducibilities

Def. A function $f: \{0,1\}^* \longrightarrow \{0,1\}^*$ is *poly-time computable* if there is a polynomial-time-bounded DTM M that on input w outputs $f(w)$

Def. Let $L_1, L_2 \subseteq \{0,1\}^*$ be languages.
L_1 is *polynomial-time reducible to* L_2, written $L_1 \leq_p L_2$ if there exists a polynomial-time computable function $f: \{0,1\}^* \longrightarrow \{0,1\}^*$ such that

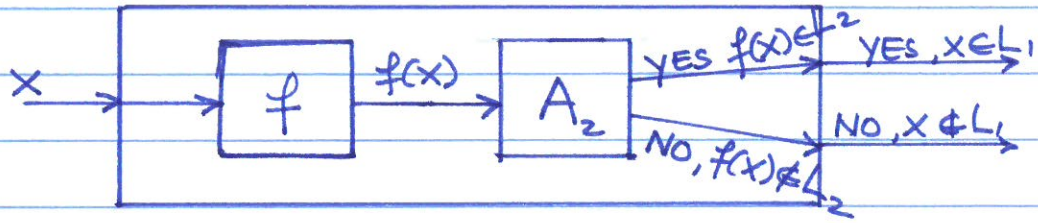
$$w \in L_1 \iff f(w) \in L_2$$



We say: f *reduces* L_1 to L_2
An algor. that computes f is called a *reduction algorithm*

Prop. If $L_1 \leq_p L_2$ and $L_2 \in P$, then $L_1 \in P$

Pf. Let A_2 be a poly. time algorithm for L_2. Then we obtain a poly. time algorithm for L_1 as follows:



□

Prop. If $L_1 \leq_p L_2$ and $L_2 \in NP$, then $L_1 \in NP$ □

Fact. $\leq_p$ is reflexive and transitive.

Pf. Clearly $L \leq_p L$. Hence $\leq_p$ is reflexive.

Now let $L_1 \leq_p L_2$ via reduction f_1
and $L_2 \leq_p L_3$ via reduction f_2

Then $f_2 \circ f_1$ reduces L_1 to L_3 □

Satisfiability

Boolean formulas are expressions involving

- Boolean constants 0, 1
- Boolean variables $x_1, x_2, \ldots$
- Boolean operators $\wedge$, $\vee$, $\neg$

Let $F(x_1,..,x_n)$ be a Boolean formula with variables $x_1,..,x_n$. An assignment for $x_1,..,x_n$ is a function $\varphi: \{x_1,...,x_n\} \to \{0,1\}$

An assignment φ satisfies F if substituting values $\varphi(x_i)$ for x_i, $i=1,..,n$, F evaluates to 1.

F is said to be satisfiable if there is an assignment that satisfies F

Def. Satisfiability is the problem of deciding whether a given formula is satisfiable:

Input. A Boolean formula $F(x_1,..,x_n)$

Quest. Is there an assignment for $x_1,..,x_n$ that satisfies F ?

Conjunctive Normal Form

A literal is a var. x or its negation $\overline{x}$

A clause is a disjunction of literals

e.g. $x_1 \vee \overline{x_2} \vee x_3$

A Boolean formula is in conjunctive normal form (CNF) if it's a conjunction of clauses.

e.g. $(x_1 \vee \overline{x}_2 \vee \overline{x}_3) \wedge (x_3 \vee \overline{x}_5 \vee x_6) \wedge (x_3 \vee x_4)$

SAT denotes the satisfiability problem

CNF-SAT denotes the satisfiability problem for Boolean formulas in CNF

A Boolean formula is in 3-CNF if it's in CNF and each clause contains at most 3 literals.

3-SAT denotes the satisfiability problem for Boolean formulas in 3-CNF.

Theorem. 3-SAT $\leq_p$ CLIQUE

Pf. Idea: We construct a poly. time reduction from 3-SAT to CLIQUE

$F = C_1 \wedge \cdots \wedge C_k \xrightarrow{f} (G, k)$ s.t.

Sat. assigments $\longleftrightarrow$ Cliques of size k

Each $C_i = (a_i \vee b_i \vee c_i) \longleftrightarrow$ group of verties a_i, b_i, c_i

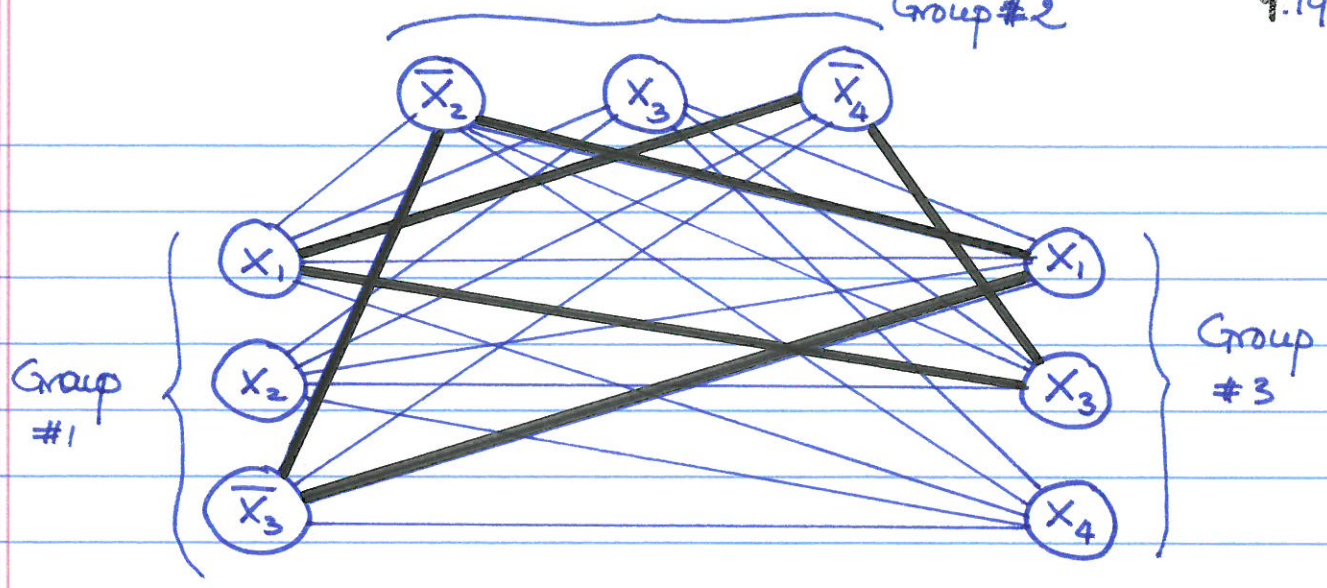
No edges within a group and

No edges between opposite literals

For example consider

$$F = (x_1 \vee x_2 \vee \overline{x}_3) \wedge (\overline{x}_2 \vee x_3 \vee \overline{x}_4) \wedge (x_1 \vee x_3 \vee x_4)$$

Clause #1 Clause #2 Clause #3



Clique $\{\overline{x}_3, \overline{x}_2, x_1\} \leftrightarrow$ Assignment
$\overline{x}_3 = 1, \overline{x}_2 = 1, x_1 = 1$

Clique $\{x_1, \overline{x}_4, x_3\} \leftrightarrow$ Assignment
$x_1 = 1, \overline{x}_4 = 1, x_3 = 1$

From $F = C_1 \wedge \cdots \wedge C_k$

$= (a_1 \vee b_1 \vee c_1) \wedge \cdots \wedge (a_k \vee b_k \vee c_k)$

G is constructed as follows:

- $C_i = (a_i \vee b_i \vee c_i) \leftrightarrow$ Group #i: a_i, b_i, c_i
- There are no edges within a group of vertices
- Between group #i and group #j:
  - No edge between a pair of vertices if they are complementary, i.e., x & $\overline{x}$

Claim. F is satisfiable iff G has a clique of size k.

Pf (Claim). "$\Rightarrow$". Suppose F has a sat. assignment φ s.t. each C_i has a literal $\ell_i \in \{a_i, b_i, c_i\}$ with value true. Since φ is a Boolean assignment, none of the ℓ_i's are complementary literals. Thus, the ℓ_i's form a clique of size k.

"$\Leftarrow$": Conversely, suppose G has a clique of size k. Such a clique has exactly one vertex in each group of vertices. If each literal represented by such a vertex is assigned value true, we obtain a sat. assignment for F. □

9.5. NP-Completeness & Cook's Thm.

Def. A language L is **NP-hard** iff every language $L' \in NP$ is poly-time reducible to L, i.e.,

$\forall L' \in NP : L' \leq_p L$.

L is **NP-complete** iff (1) $L \in NP$ and (2) L is NP-hard.

Prop. Let L be an NP-complete lang. Then $L \in P \iff P = NP$ □

Theorem (Cook). SAT is NP-complete.

Pf. Given a Boolean formula $F(x_1, \ldots, x_n)$ a NTM can nondeterministically guess a Boolean assignment for $x_1, \ldots, x_n$ and check if it satisfies F. Thus, SAT $\in$ NP.

To show SAT is NP-hard we constr. a poly. time reduction from any $L \in NP$ to SAT.

Let $L = L(M)$ for some poly time bounded 1-tape NTM M where (Assume tape is semi-inf.)

$$M = (Q, \{0,1\}, \Gamma, \delta, q_0, F, B)$$

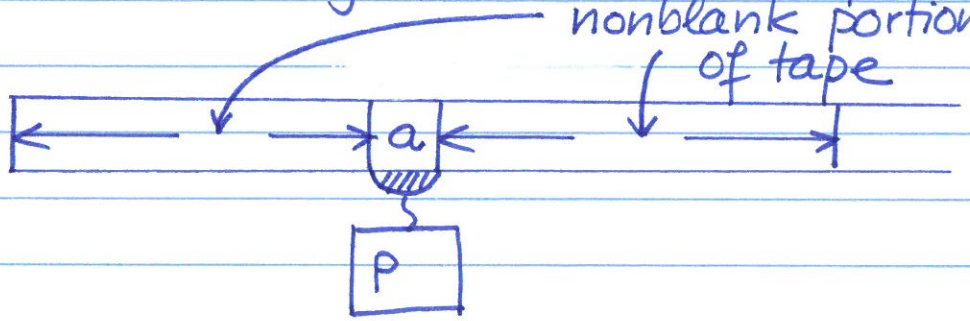
Goal: To constr. a poly. time compt. function $f : \{0,1\}^* \longrightarrow \{0,1\}^*$ s.t

$$w \longmapsto \langle F_w \rangle$$

and $\quad w \in L \iff F_w \in SAT.$

Let NTM M be $p(n)$-time-bounded for some fixed poly. $p(n)$. Let $|w| = n$.

Since M operates in at most $p(n)$ steps, an ID of M is a string β over $Q \times \Gamma$ of length $p(n)$



$$\beta = \underbrace{\quad}_{\Gamma^*} (p,a) \underbrace{\quad}_{\Gamma^*}$$

current head position and state symb. being scanned is a

Allowing a halting conf. to repeat, a computation is a string of form

$$\omega = \beta_0 \# \beta_1 \# \beta_2 \# \cdots \# \beta_{p(n)} \#$$

where β_i is the conf. after the i-th comp. step, $i = 1, 2, \ldots, p(n)$, and β_0 is the initial configuration with input $w = w_1 \ldots w_n$.

Thus, $|\omega| = (p(n)+1)^2$

W.l.o.g. assume that M has at most m choices for its next move, where m is a constant.

A symbol in an ID is from alphabet

$$\Delta = \Gamma \cup \{\#\} \cup \{(q,a,\ell) \mid q \in Q, a \in \Gamma, \text{ and } \ell \leq m\}$$

Now, consider ω and a position i in ω, $1 \leq i \leq (p(n)+1)^2$ and $X \in \Delta$:

The truth of statement

"X is symbol at i-th position in ω"

is described by Boolean variable $c_{i,X}$

Observation. ω is a correct accept. comput. iff:

(1) $\forall i : \exists$ exactly one $X \in \Delta$: $c_{i,X} = \text{true}$

(2) β_0 is init. ID with input w

(3) $\beta_{p(n)}$ is an accepting ID

(4) $\beta_j \vdash_M \beta_{j+1}$ for $j = 0, \ldots, p(n)-1$

To Cond. (1): Condition (1) can be described by:

$$\bigwedge_{1 \le i \le (p(n))+1} \left[\bigvee_{X \in \Delta} \left[c_{i,X} \wedge \neg \left(\bigvee_{Y \in \Delta - \{X\}} c_{i,Y} \right) \right] \right]$$

To Cond. (2): Note that with $w = w_1 \ldots w_n$, $w_i \in \{0,1\}$,

$$\beta_0 = \underbrace{(q_0, w_1, 1)}\, w_2 w_3 \ldots w_n \, BB \ldots B \#$$

$$\in \{ \underbrace{(q_0, w_1, 1)}_{=Y_1}, \ldots, \underbrace{(q_0, w_1, m)}_{=Y_m} \}.$$

So condition (2) can be described by

(i) $c_{p(n)+1, \#}$ ($\hat{=}$ symb at pos. $p(n)+1$ is $\#$)

(ii) $c_{1,Y_1} \vee \ldots \vee c_{1,Y_m}$ ($\hat{=}$ symb at pos. 1 is either Y_1 or ... or Y_m)

(iii) $\bigwedge_{2 \le i \le n} c_{i,w_i}$

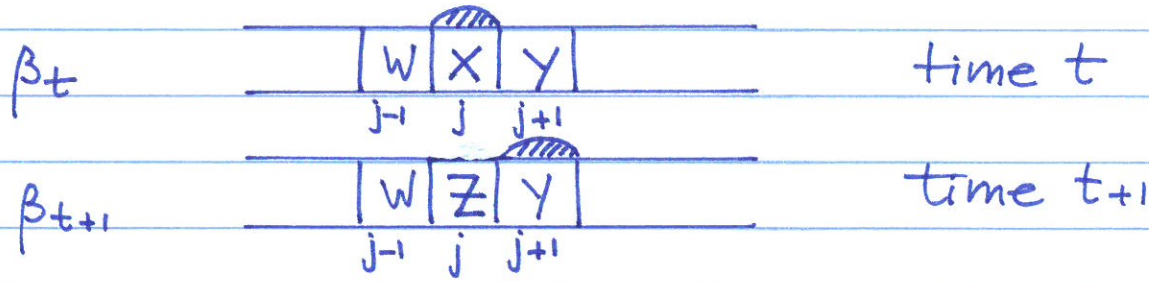
(iv) $\bigwedge_{n+1 \le i \le p(n)} c_{i,B}$

To Cond. (3): Let $\Theta = F \times \Gamma \times [1..m]$

$$\bigvee_{p(n)(p(n)+1) < i < (p(n)+1)^2} \left(\bigvee_{x \in \Theta} c_{i,x} \right)$$

To Cond. (4):

Consider a move of M:



Z depends on X and transition fct δ.

This dependency can be described by a predicate $P(W, X, Y, Z)$

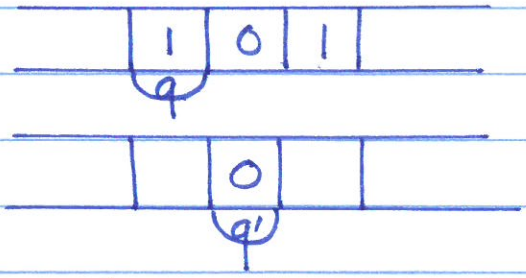
For example, if 2nd choice in $\delta(q,1)$ is $(q', 0, R)$, then

$$0(q,1,2)\,1 \longrightarrow 00(q',1,5)$$

current state; symbol being scanned; 2nd trans. chosen; choice of transition for next move

which means $P(0,(q,1,2),1,0)$ is true

Similarly, $P((q,1,2),0,1,(q',0,4))$ corresponds to

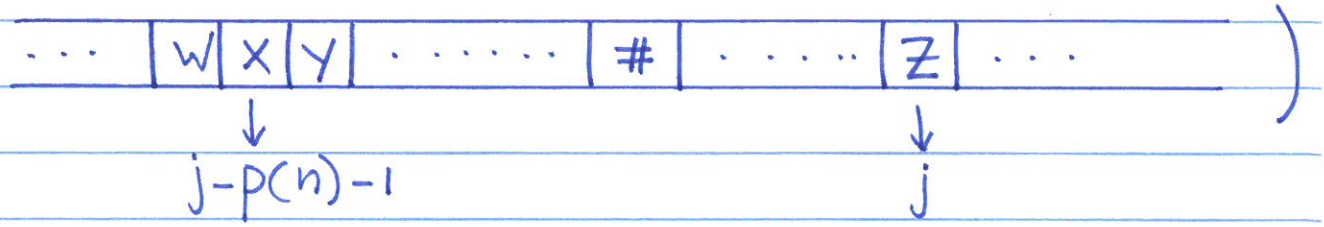


Thus, P is a fixed predicate that is completely specified by M (i.e., δ).

Therefore, Cond. (4) can be described by:

$$\bigwedge_{p(n)<j<(p(n)+1)^2} \bigvee_{\substack{w,x,y,z \\ \text{s.t. } P(w,x,y,z)}} \left(c_{j-p(n)-2,w} \wedge c_{j-p(n)-1,x} \wedge c_{j-p(n),y} \wedge c_{j,z} \right)$$

(Note that due to delimeter #, the subsc. are $j-p(n)-2$, $j-p(n)-1$, $j-p(n)$ rather than $j-p(n)-1$, $j-p(n)$, $j-p(n)+1$:



Define $F_w := \text{Cond}(1) \wedge \cdots \wedge \text{Cond}(4)$

Then $w \in L(M) \iff F_w$ is sat.

Moreover, given w, F_w can be computed in poly. time $\square$

9.6. Further NP-Complete Problems

Thm. CNF-SAT is NP-complete.

Pf. It remains to show NP-hardness.

We construct a poly. time reduction from SAT to CNF-SAT: $SAT \leq_P$ CNF-SAT.

Idea: Transf. a given Boolean formula E into an equiv. formula E' w/o negation ($\neg$), and then E' into an equiv. formula E'' in CNF:

$$E \xrightarrow[\text{Elim. of } \neg]{} E' \xrightarrow[\substack{\text{Normalization} \\ \text{of } \wedge, \vee}]{} E''$$

(1) **Elimination of $\neg$:**

Using De Morgan's laws:

$$\neg(E_1 \wedge E_2) = \neg E_1 \vee \neg E_2$$
$$\neg(E_1 \vee E_2) = \neg E_1 \wedge \neg E_2$$
$$\neg(\neg E_1) = E_1$$

E' can be computed from E in poly. time.

(2) **Normalization of $\wedge, \vee$**

E' is recursively transformed into E''.

Basis. E' is a literal ✓

Ind. Step.

Case 1. $E' = E_1' \wedge E_2'$

By IH, let E_1'' and E_2'' be formulas in CNF that are equiv. to E_1', E_2', resp.

Then, $E'' = E_1'' \wedge E_2''$ is in CNF.

Case 2. $E' = E_1' \vee E_2'$

By IH, let E_1'' and E_2'' be formulas in CNF equiv. to E_1' and E_2'', resp.

Let $E_1'' = (F_1 \wedge \cdots \wedge F_k)$

and $E_2'' = (G_1 \wedge \cdots \wedge G_\ell)$

where $F_1, \ldots, F_k, G_1, \ldots, G_\ell$ are clauses.

Let y be a new variable not in E_1'' and E_2''. Construct

$$E'' = (F_1 \vee y) \wedge \cdots \wedge (F_k \vee y) \wedge (G_1 \vee \bar{y}) \wedge \cdots \wedge (G_\ell \vee \bar{y}).$$

Clearly, E'' is in CNF and equiv. to E'

Moreover, E'' can be constructed from E' in poly. time ⊠

Thm. 3-SAT is NP-complete

Pf. We show CNF_SAT $\leq$ 3-SAT.

Let F be a CNF formula and

$x_1 \vee x_2 \vee \cdots \vee x_n$ ($\rightarrow$ these are literals), $n > 3$

be a clause in F. Let $y_1, \ldots, y_n$ be new var and consider

$$\alpha = (x_1 \vee \bar{y}_1) \wedge (y_1 \vee x_2 \vee \bar{y}_2) \wedge \cdots \wedge (y_{n-1} \vee x_n \vee \bar{y}_n) \wedge y_n.$$

Claim. Let $\psi : \{x_1, \ldots, x_n\} \to \{0,1\}$ be a truth assignment. Then there is an extension $\varphi : \{x_1, \ldots x_n, y_1, \ldots, y_n\} \to \{0,1\}$ s.t. $\varphi(\alpha) = 1 \iff \psi(x_1 \vee \cdots \vee x_n) = 1$.

Pf. (Claim).

"$\Leftarrow$": Assume $\psi(x_1 \vee \cdots \vee x_n) = 1$. Let i_0 be least i s.t. $\psi(x_i) = 1$. Define

$$\varphi(z) = \begin{cases} \psi(x_j) & \text{if } z = x_j \text{ for some } j \\ 0 & \text{if } z = y_j \text{ for some } j < i_0 \\ 1 & \text{otherwise.} \end{cases}$$

Clearly, $\varphi(\alpha) = 1$.

"$\Rightarrow$": Let $\varphi : \{x_1, \ldots, x_n\} \cup \{y_1, \ldots, y_n\} \to \{0,1\}$ be s.t. $\varphi(\alpha) = 1$. We show

$$\varphi(x_1 \vee \cdots \vee x_n) = 1.$$

$C =$ $x_1 \vee x_2 \vee x_3 \vee \cdots \vee x_i \vee \cdots \vee x_n$

$\alpha_C =$

$$(x_1 \vee \overline{y_1}) \wedge (y_1 \vee x_2 \vee \overline{y_2}) \wedge (y_2 \vee x_3 \vee \overline{y_3}) \wedge \cdots \wedge (y_{i-1} \vee x_i \vee \overline{y_i}) \wedge \cdots \wedge (y_{n-1} \vee x_n \vee \overline{y_n}) \wedge y_n$$

where $y_1, \ldots, y_n$ are new variables.

$x_1, x_2, \ldots, x_n$ are literals and $n > 3$

Claim. C is satisfiable $\iff$ α_C is sat.

Proof. "$\Rightarrow$": Let C be sat. and i_0 be least s.t. $x_{i_0} = \text{true} = 1$.

Assign 0/1 to y_j's as follows

$$(x_1 \vee \overline{y_1}) \quad \text{true}$$
$$\wedge (y_1 \vee x_2 \vee \overline{y_2}) \quad \text{true}$$
$$\vdots$$
$$\wedge (y_{i_0-2} \vee x_{i_0-1} \vee \overline{y_{i_0-1}}) \quad \text{true}$$
$$\wedge (y_{i_0-1} \vee x_{i_0} \vee \overline{y_{i_0}}) \quad \text{false}$$
$$\wedge (y_{i_0} \vee x_{i_0+1} \vee \overline{y_{i_0+1}}) \quad \text{true}$$
$$\vdots$$
$$\wedge (y_{n-1} \vee x_n \vee \overline{y_n}) \quad \text{true}$$
$$\wedge y_n \quad \text{true}$$

"$\Leftarrow$": Suppose α_C is sat.

We prove C is sat. by contradiction.

So suppose C is not sat.

Then $x_i = 0$ for all $i = 1, \ldots, n$

Consider

$$\alpha_C$$

$$=$$

$$(x_1 \vee \overline{y_1})$$ must be true

$$\wedge (y_1 \vee x_2 \vee \overline{y_2})$$ must be true

$$\vdots$$

$$\wedge (y_{i-1} \vee x_i \vee \overline{y_i})$$

$$\vdots$$

$$\wedge (y_{n-1} \vee x_n \vee \overline{y_n})$$

$$\wedge \quad y_n$$ → must be true

y_{n-1} must be true & false?

Suppose by way of contradiction that $\varphi(x_i) = 0 \ \forall i$.

We show that $\varphi(y_i) = 0 \ \forall i$ which would imply $\varphi(\alpha) = 0$, a contradiction.

We prove $\varphi(y_i) = 0 \ \forall i$ by ind. on i.

For $i = 1$: Since $\varphi(x_1 \vee \overline{y_1}) = 1$, we must have $\varphi(y_1) = 0$.

Next for $i = 2$: Since $\varphi(y_1 \vee x_2 \vee \overline{y_2}) = 1$, we must have $\varphi(y_2) = 0$.

Similarly $\varphi(y_3) = 0 = \varphi(y_4) = \cdots = \varphi(y_n)$.

To complete the reduction, simply replace any clause with more than 3 literals by a set of clauses as described above. Then in the resulting formula each clause has at most 3 literals. □

Since 3-SAT $\leq_p$ CLIQUE, we have

Thm. CLIQUE is NP-complete

Pf. It is obvious that CLIQUE is in NP. □

We next consider **Integer Programming**

Input. Integer matrix C and integer vector d

Quest. Is there a $(0,1)$-vector c s.t.

$$C \cdot c \geq d \ ?$$

Thm. $(0,1)$-Integer Progr. is NP compl.

Pf. It is obvious that the problem is in NP: Given C, d and a certificate $\bar{c}$, one can verify if $C \cdot \bar{c} \geq d$ in poly. time.

To show NP-hardness we prove

$$\text{CNF-SAT} \leq_p (0,1) \text{ Integer Progr.}$$

Let F be a CNF formula with var. $x_1, \dots, x_n$: $F = F_1 \wedge F_2 \wedge \dots \wedge F_k$.

Def. matrices $C = (c_{ij})$, $1 \leq i \leq k$, $1 \leq j \leq n$ and $d = (d_i)$, $1 \leq i \leq k$ by:

$$c_{ij} = \begin{cases} 1 & \text{if } x_j \text{ occurs in } F_i \\ -1 & \text{if } \overline{x_j} \text{ occurs in } F_i \\ 0 & \text{otherwise} \end{cases}$$

$d_i = 1 - \#$ of var. x with $\bar{x}$ occurring in F_i

Ex. $F = (x_1 \vee \overline{x}_2 \vee x_4) \wedge (\overline{x}_1 \vee x_3 \vee \overline{x}_4) \wedge (x_2 \vee \overline{x}_3 \vee x_4)$

$$C = \begin{pmatrix} 1 & -1 & 0 & 1 \\ -1 & 0 & 1 & -1 \\ 0 & 1 & -1 & 1 \end{pmatrix} \qquad d = \begin{pmatrix} 0 \\ -1 \\ 0 \end{pmatrix}$$

Consider a sat. assignment: $x_1 = 0$, $x_2 = 0$, $x_3 = 1$, $x_4 = 1$ and let $c = (0\ 0\ 1\ 1)$.

Then

$$C \cdot c = \begin{pmatrix} 1 & -1 & 0 & 1 \\ -1 & 0 & 1 & -1 \\ 0 & 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} \geq d$$

$\geq d$ □

Claim. F is sat. iff $\exists$ (0,1) vector c: $C \cdot c \geq d$

Pf. (Claim)

"$\Rightarrow$": Let ψ be an assignment s.t. $\psi(F) = 1$.

Def. $c_j = \psi(x_j)$, $1 \leq j \leq n$.

Then

$$(C \cdot c)_i = \sum_{j=1}^{n} c_{ij} c_j = \sum_{x_j \in F_i} \psi(x_j) - \sum_{\overline{x}_j \in F_i} \psi(x_j)$$

Now, since F_i is sat., there is either $x_j \in F_i$ with $\psi(x_j) = 1$ or $\overline{x}_j \in F_i$ with $\psi(x_j) = 0$,

$$(C \cdot c)_i \geq 1 - \sum_{\overline{x}_j \in F_i} 1 = d_i$$

Consider e.g. $F_i = x_1 \vee \overline{x_2} \vee x_3 \vee \overline{x_4} \vee \overline{x_5}$

and assignment $x_1 = 0$, $x_2 = 1$, $x_3 = 0$, $x_4 = 1$, $x_5 = 1$, i.e., $(0, 1, 0, 1, 1)$:

$$(1 \quad -1 \quad 1 \quad -1 \quad -1) \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = -3 = -\sum_{\overline{x_j} \in F_i} \psi(x_j)$$

This assignment doesn't sat. clause.

Consider a sat. assign. $(1, 1, 0, 1, 1)$

Then

$$(1 \quad -1 \quad 1 \quad -1 \quad -1) \begin{pmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{pmatrix} = 1 - \sum_{\overline{x_j} \in F_i} \psi(x_j)$$

Thus define

$$d_i = - \#\text{ of literals } \overline{x} \text{ in } F_i + 1$$

$$= 1 - \#\text{ of literals } \overline{x} \text{ in } F_i$$

Then assignment c sat. F_i

$$\iff C_i \cdot c \geq d_i$$

i.e. c sat. $F \iff C \cdot c \geq d$.

"$\Leftarrow$": Let c be a $(0,1)$-vector with $C.c \geq d$.

Def. truth assignment ψ by $\psi(x_j) = c_j$

We show that $\psi(F) = 1$.

Suppose otherwise that $\psi(F_i) = 0$ for some i. Then:

- If x_j occurs in F_i, then $\psi(x_j) = c_j = 0$
- If $\overline{x_j}$ " " ", then $\psi(x_j) = c_j = 1$.

Hence,

$$d_i \leq (C.c)_i = \sum_{x_j \in F_i} c_j - \sum_{\overline{x_j} \in F_i} c_j = -\sum_{\overline{x_j} \in F_i} 1 < d_i$$

$\rightsquigarrow$ a contradiction. Thus, $\psi(F) = 1$ □

We next consider <u>VERTEX-Cover</u>

Given a graph $G = (V, E)$, <u>a vertex cover</u> is a subset $C \subseteq V$ of vertices s.t. for every edge $(u,v) \in E$: $u \in C$ or $v \in C$, (or both)

<u>Input</u>. A graph G and k

<u>Quest</u>. Does G have a vertex cover of size k?

Thm. Vertex Cover is NP_complete

Pf. It's obvious that VC is in NP. For NP-hardness we prove: CLIQUE $\leq_p$ VC.

Let $\langle G=(V,E), k\rangle$ be an instance of CLIQUE. Construct an instance of VC by: $\langle \overline{G}, |V|-k\rangle$, where $\overline{G}$ is the complement of G, i.e., $\overline{G}=(V,\overline{E})$ and $\overline{E} = V\times V - E$.

Claim. G has a clique V' of size k $\iff$ V-V' is a VC of size |V|-k.

Pf. (Claim). Suppose V' is a clique in G. We show V-V' is a VC in $\overline{G}$.

Let $(u,v)\in\overline{E}$. Then either $u\notin V'$ or $v\notin V'$. Therefore, $u\in V-V'$ or $v\in V-V'$. Thus, V-V' is a VC in $\overline{G}$.

Conversely, suppose V' is a VC in $\overline{G}$. Then $(u,v)\in\overline{E} \Rightarrow u\in V'$ or $v\in V'$

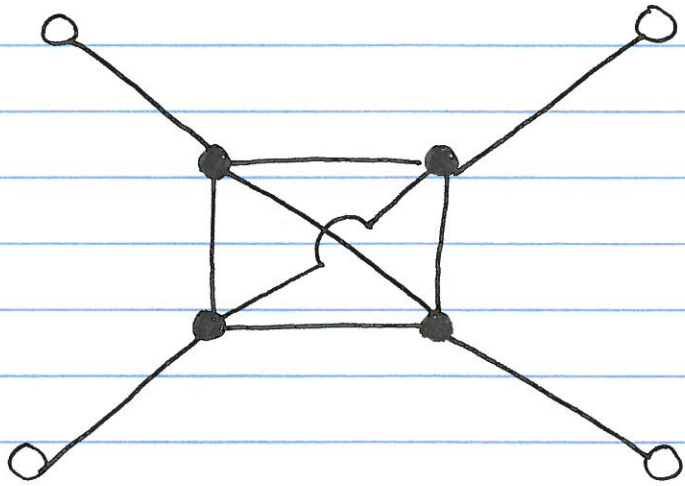
Hence, $u\in V-V' \wedge v\in V-V' \Rightarrow (u,v)\in E$

Thus, V-V' is a clique in G. ⊠

Example.

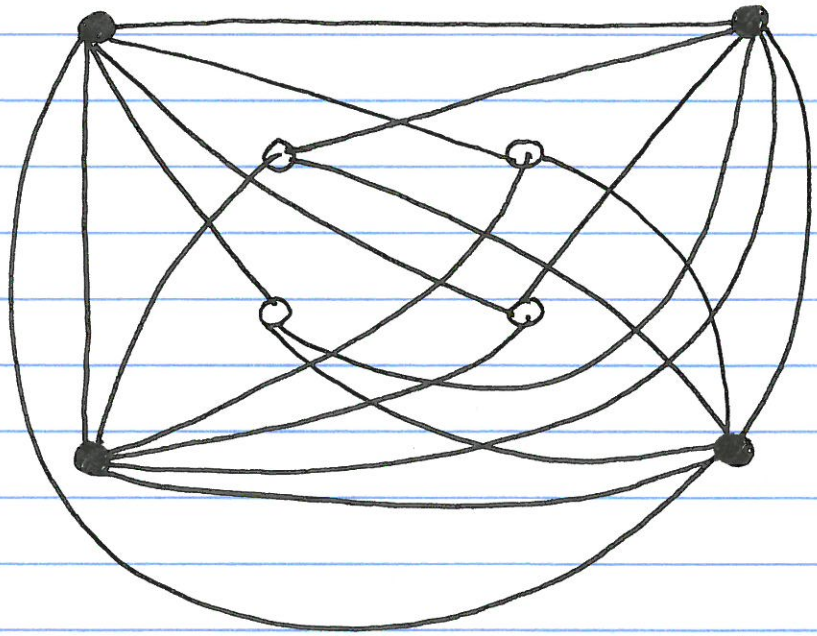
$G =$

$k = 4$



$G' =$

$k' = 4$



Thm. Hamiltonian Cycle is NP-complete

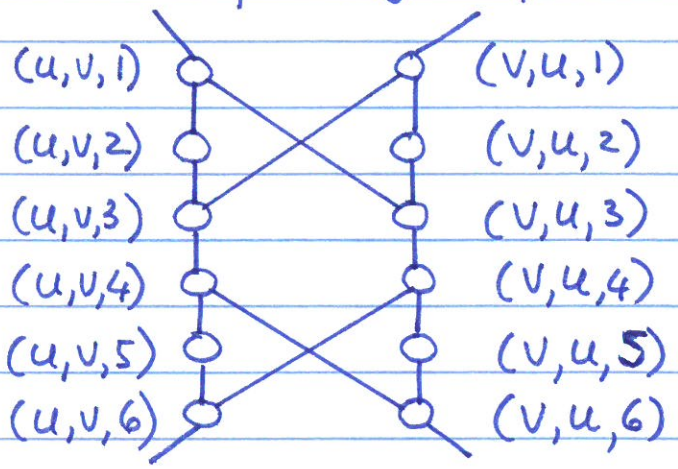
Proof. It's obvious that the problem is in NP.
For NP-hardness we prove

Vertex Cover $\leq_p$ Hamiltonian Cycle.

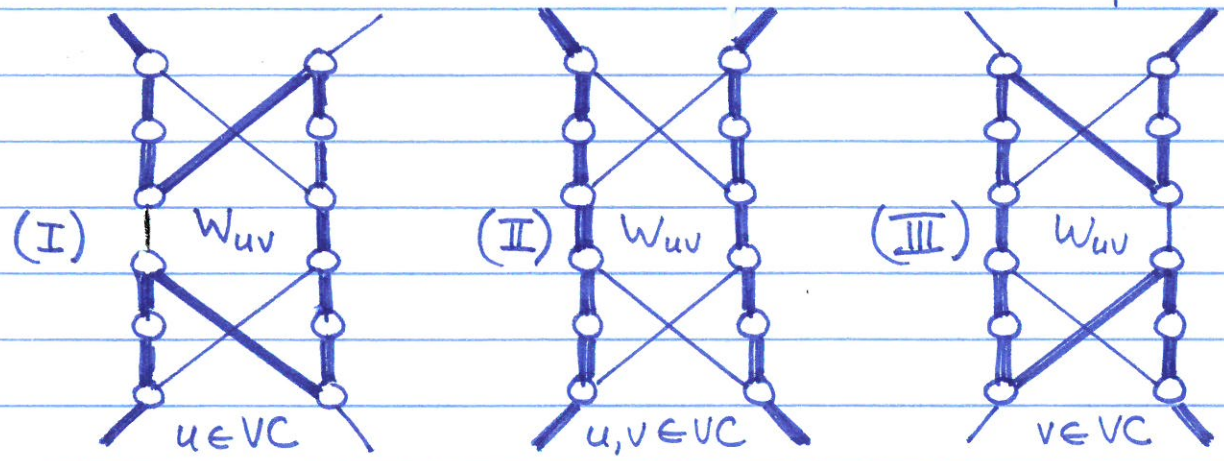
$$G=(V,E), k \longmapsto G'=(V',E')$$

$\exists$ VC of size $k \iff \exists$ Hamiltonian cycle

Construction of widget for $(u,v) \in E$:



Observation. Possible Hamiltonian paths

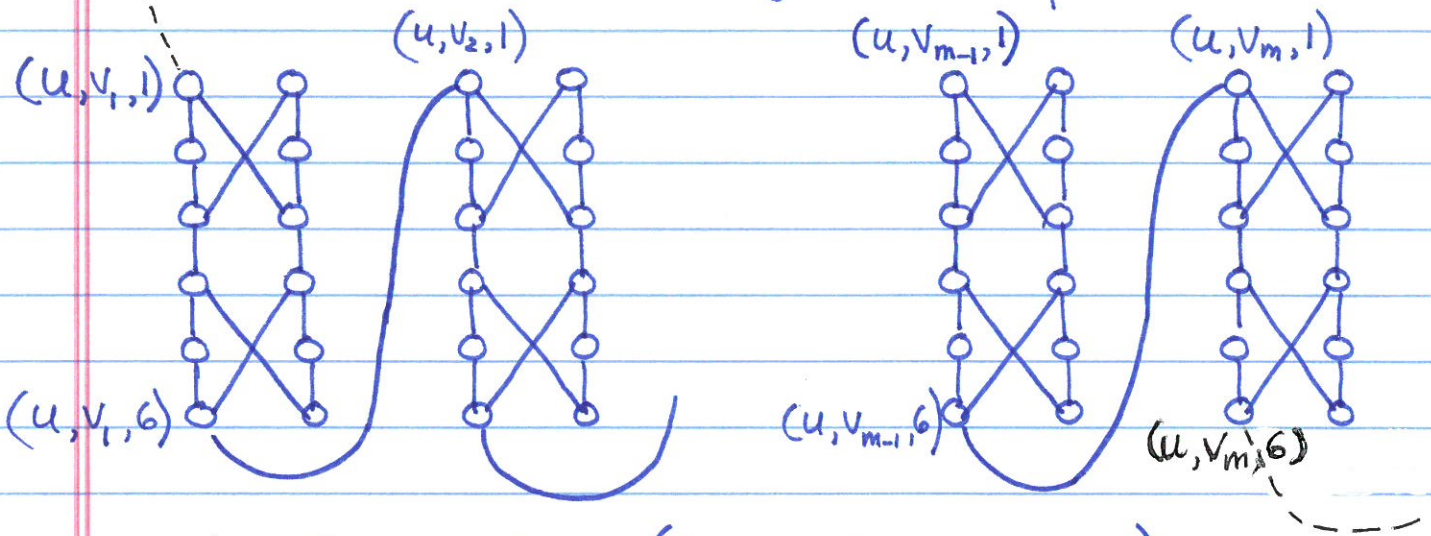


and no other Hamiltonian paths possible!

There are also k selector vertices $s_1, \ldots, s_k$

Connecting the widgets

Let $u \in V$ of degree m, and $(u, v_1), \ldots, (u, v_m)$ be the edges incident on u, where $v_1, \ldots, v_m$ is an arbitrary order of u's neighbors.



We have edges $\big((u, v_i, 6), (u, v_{i+1}, 1)\big)$, $i = 1, \ldots, m-1$.

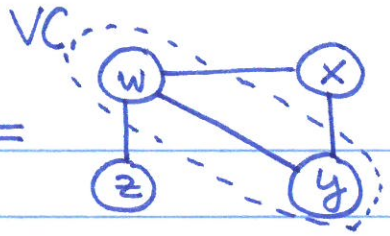
Idea: If $u \in VC$, then its incident edges $(u, v_1), \ldots, (u, v_m)$ correspond to edges $\big((u, v_i, 6), (u, v_{i+1}, 1)\big)$, $1 \le i \le m-1$, on hamiltonian path.

Final set of edges in E'

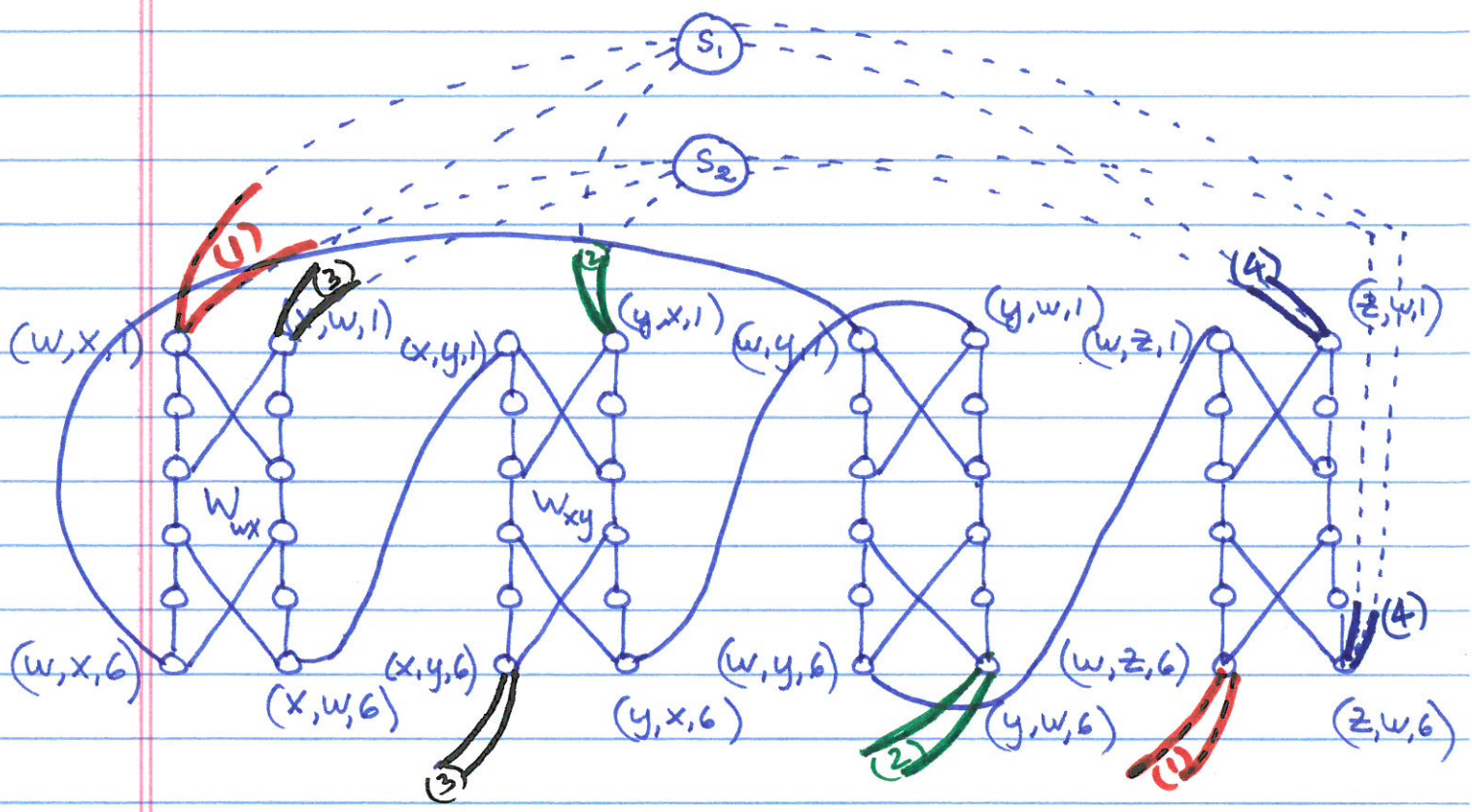
For each $u \in V$ constr. edges

$$\big(s_j, (u, v_1, 1)\big), \big(s_j, (u, v_m, 6)\big) \quad \forall j = 1, \ldots, k$$

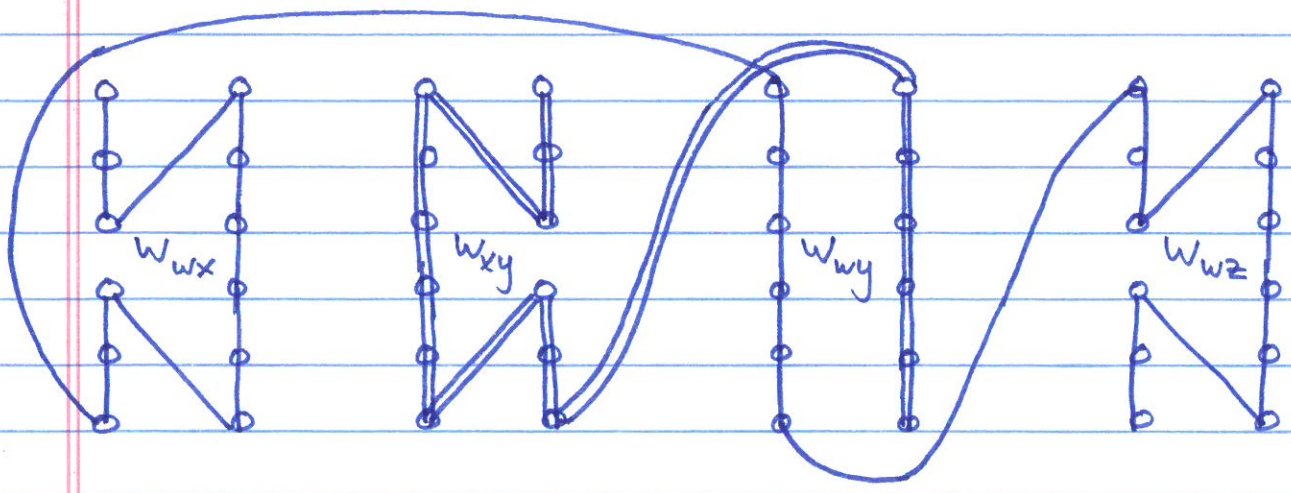
where $m = \text{degree}(u)$.

Ex. $G =$ 

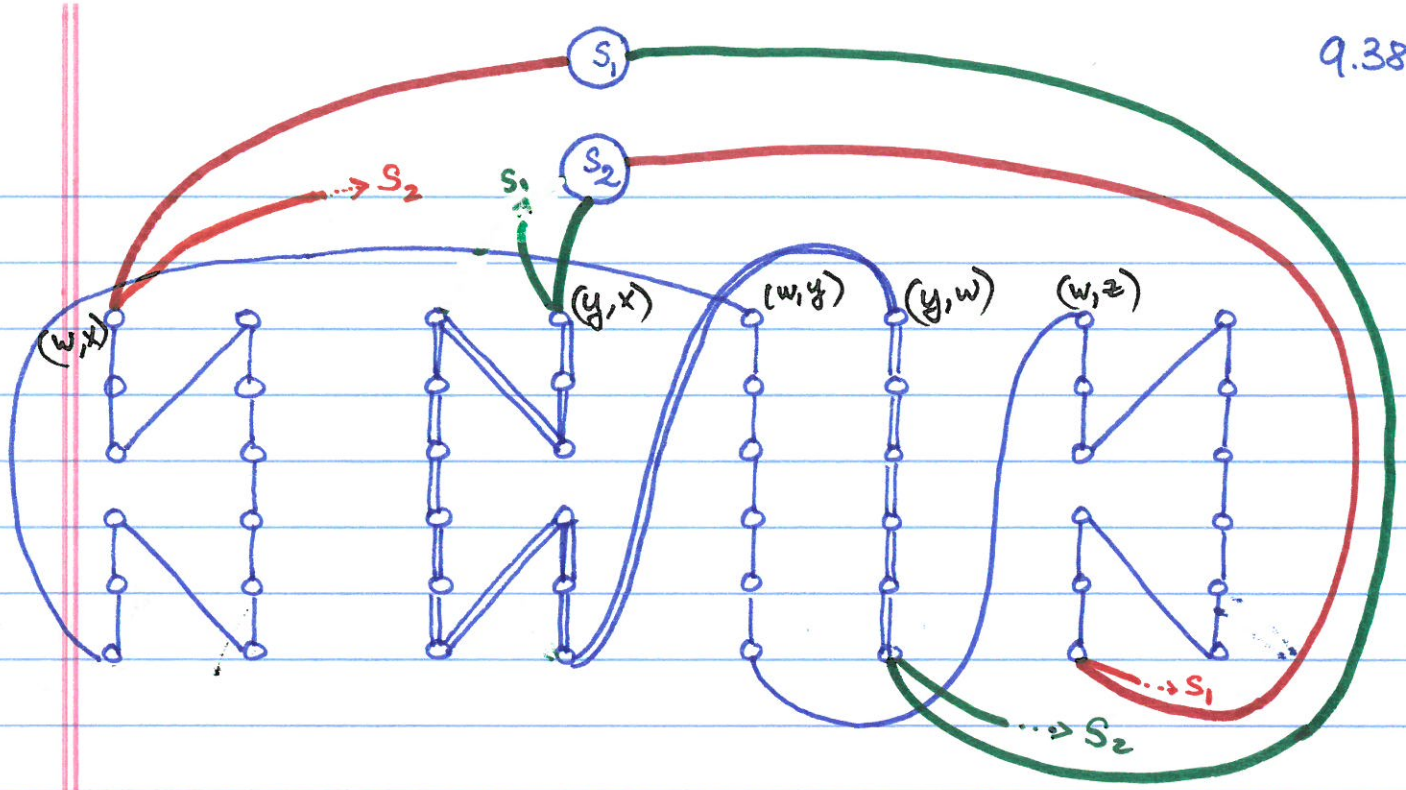
$k = 2$



Construct the final edges by connecting each "unconnected" top/bottom vertex of a widget with s_1 and s_2.



How edges in G are covered by VC.



A Hamiltonian cycle.

We need to show:

(1) Given G, k, the graph $G' = (V', E')$ can be constructed in poly. time

(2) G has a VC of size $k \iff$ G' has a Hamiltonian cycle.

To (1): Note that V' contains:

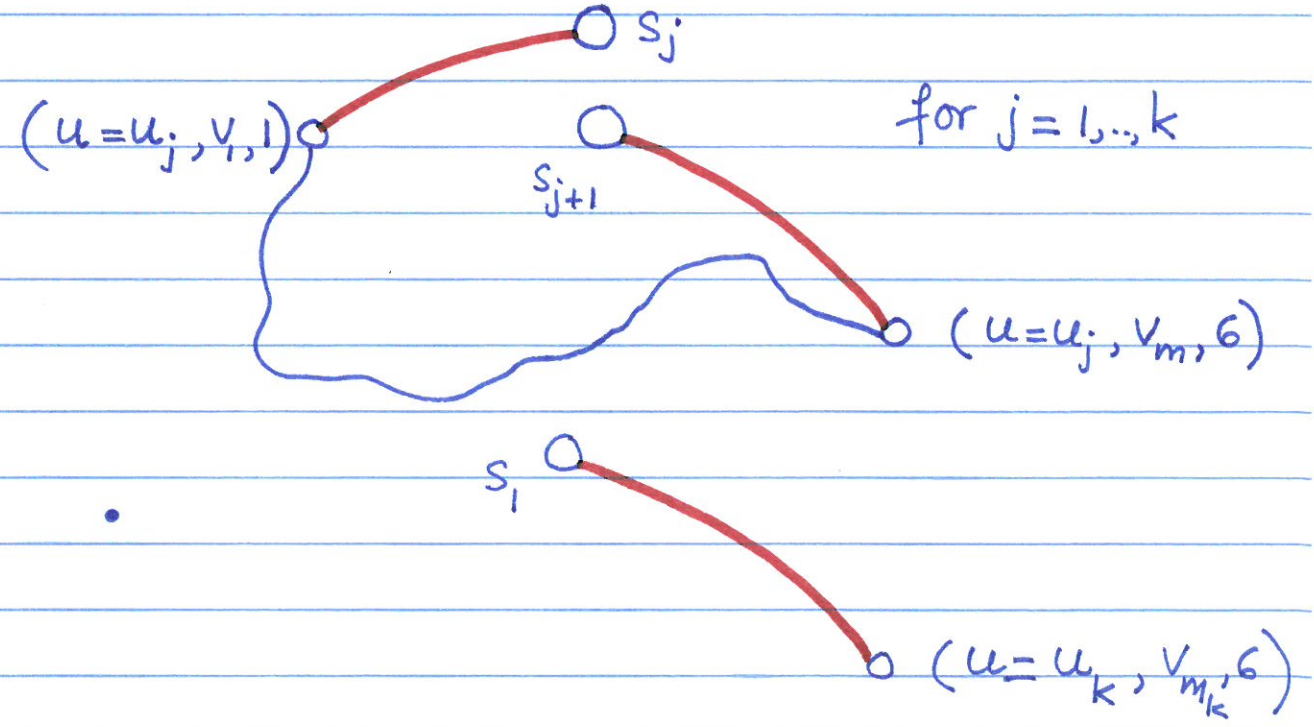
- For each edge $(u,v) \in E$, 12 vertices in widget W_{uv}.
- k Selector vertices

It's obivous that G' can be constructed from G, k in poly. time.

To (2):

"$\Rightarrow$": Given a vertex cover C of size k we form a Hamiltonian cycle as follows:

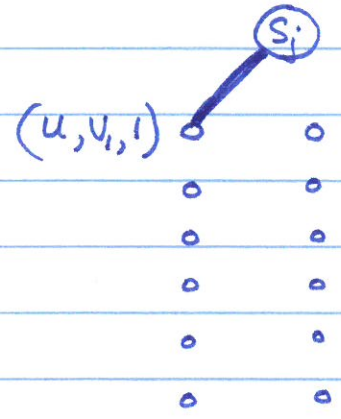
- For each $u \in C$ of degree m with incident edges $(u, v_1), \ldots, (u, v_m)$, choose for (u, v_i) Hamiltonian path (I) if $v_i \notin C$, otherwise choose (II).
- Connect these Hamiltonian paths using $((u, v_1, 6), (u, v_2, 1)), \ldots, ((u, v_{m-1}, 6), (u, v_m, 1))$
- Now suppose that vertices in C are ordered as $u_1, \ldots, u_k$. Then form



"$\Leftarrow$": Conversely, suppose G' has a Hamiltonian cycle $\Omega \subseteq E'$.

Consider edges of Ω that are of form

s_j ——— $(u, v_1, 1)$:



We claim that the set of these k vertices u forms a VC C in G.

. First note that Hamiltonian path through a widget must be of pattern I (or III) or II

. when we start Ω at s_1 we must end at s_1, otherwise Ω is no Ham. cycle.

. Ω can be decomposed into subpaths

$$\Omega = (\underbrace{s_1, \ldots,}_{\text{subpath}} \underbrace{s', \ldots,}_{\text{subpath}} s'', \ldots\ldots, \underbrace{s_1''\ldots\ldots, s_1}_{\text{subpath}})$$

in which no $s \in \{s_1, \ldots, s_k\}$ occurs

. Each subpath traverses through widgets

using pattern I (or III) or II.

. Since all widgets are covered by Ω, every edge (u,v) has $u \in C$ or $v \in C$.

Thus, C is a VC. ⊠

Def. The Traveling-salesman problem (TSP) is def. as follows:

Input. A graph $G=(V,E)$, a distance function $c: V \times V \to \mathbb{Z}$, and $k \in \mathbb{Z}$

Question. Is there a Hamiltonian cycle of cost $\leq k$?

Thm. TSP is NP-complete.

Pf. As TSP is obviously in NP, we show Hamiltonian cycle $\leq_P$ TSP.

Given an instance $G=(V,E)$ of HAM-CYC. construct an instance $G'=(V,E')$, c, k of TSP by:

(1) $E' = V \times V$

(2) $c(i,j) = \begin{cases} 1 & \text{if } (i,j) \in E \\ 2 & \text{otherwise.} \end{cases}$

(3) $k = n$.

□

Def. The Subset Sum (Knapsack) probl. is def. as follows:

Input. A set of numbers $x_1, \dots, x_k \in \mathbb{N}$ and a target number $t \in \mathbb{N}$

Quest. Is there a subset $x_{i_1}, \dots, x_{i_j}$ of $\{x_1, \dots, x_k\}$ s.t. $x_{i_1} + \cdots + x_{i_j} = t$?

Thm. Subset Sum is NP-complete

Pf. We only need to show

$$3\text{-SAT} \leq_p \text{Subset Sum}$$

Consider an instance of 3-SAT:

$$F = C_1 \wedge C_2 \wedge \cdots \wedge C_k$$

that has n variables $x_1, x_2, \dots, x_n$.

We constr. for F an instance $\langle S, t \rangle$ of Subset Sum as follows.

- For each var. x_i we have a pair of numbers y_i ($\leftrightarrow x_i = 1$)
  z_i ($\leftrightarrow x_i = 0 \Leftrightarrow \overline{x_i} = 1$)

- For each clause C_j we have a pair of numbers g_j, h_j

In decimal representation, y_i, z_i, g_j, h_j are constructed as follows:

	i: 1	2	3	4	.....	n	j: c_1	c_2	....	c_k
y_1	1	0	0	0		0	1	0	...	0
z_1	1	0	0	0		0	0	0	...	0
y_2		1	0	0		0	0	1	...	0
z_2		1	0	0		0	1	0	...	0
y_3			1	0		0	1	1	...	0
z_3			1	0		0	0	0	...	1
⋮					⋱	⋮	⋮	⋮		⋮
y_n						1				
z_n						1				
g_1							1	0	...	0
h_1							1	0	...	0
g_2								1	...	0
h_2								1	...	0
⋮								⋮		⋮
g_k										1
h_k										1
t	1	1	1	1	.....	1	3	3	....	3

for $F = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (x_2 \vee x_3 \vee \cdots) \wedge \cdots \wedge (\bar{x}_3 \vee \ldots \vee \ldots)$

- Each y_i, z_i has 2 parts : left / right
  - On the left each y_i, z_i has a 1 at position i followed by $n-i$ 0's.

	1	i	$i+1$	$\dots$	n
y_i =	1	1	0	$\dots$	0
z_i =	1	1	0	$\dots$	0

left part

  - The right part encodes the occurrence of x_i or $\overline{x}_i$ in clause C_j

	C_1	$\dots$	C_j	$\dots$	C_k
y_i =			?		
z_i =			?		

In y_i/z_i row, ? = 1 if $x_i / \overline{x}_i$ occurs in C_j

Observation. Since the left entries of t are 1's, we can select either y_i ($\Leftrightarrow x_i = 1$) or z_i ($\Leftrightarrow \overline{x}_i = 1$) but not both.

For example, if we select z_i and $\overline{x}_i$ occurs in C_k, this would contribute 1 to the corresponding entry of t.

Thus, if F is sat. by some assignment φ, then each clause C_j is sat. by at least one literal. This def. a selection of y_i/z_i

s.t. the entries in the right part in their sum are between 1 and 3 ($\leq$3 literals per clause).

The g_j/h_j numbers are so def. that the target t can be achieved:

g_j/h_j has 1 in position j followed by $k-j$ 0's.

So if in the selection of y_i/z_i the j-th entry in the right part add to

- 1, then we choose both g_j/h_j
- 2, then ——— either g_j or h_j
- 3, then we don't choose g_j or h_j.

Thus, it is easy to see that if F is sat. then there is a choice of y_i/z_i and g_j/h_j that add to target t.

Conversely, suppose there is a selection of y_i/z_i and g_j/h_j that add to target t. Then for each i exactly

one of y_i/z_i is chosen. Moreover, in the sum of the chosen y_i/z_i's, each entry in the right part must be ≥ 1. Hence, each C_j is sat. And therefore, F is sat. □

Summary of this section