THEORY OF NP-COMPLETENESS

RECALL T.M. HAS RUNNING TIME $T(n)$ IF IT HALTS WITHIN $T(n)$ STEPS ON ALL INPUTS OF LENGTH n, UNDER ALL CIRCUMSTANCES.

POLYNOMIAL TIME $T(n) = O(n^c)$, FOR SOME FIXED c

EXAMPLES

$$\left.\begin{array}{l} O(n^2) \\ O(n \log n) \\ O(n^{2.37}) \end{array}\right\} \text{POLY}$$

$$\left.\begin{array}{l} O(n^{\log n}) \\ O(2^n) \end{array}\right\} \text{NON-POLY.}$$

COMPLEXITY THEORY CLASSIFIES AS TRACTABLE ALL PROBLEMS WITH POLY-TIME ALGORITHMS — CALLING SUCH ALGORITHMS "EFFICIENT".

WHY?

a) ALL GENERAL MODELS OF COMPUTATIONS CAN SIMULATE EACH OTHER IN POLYNOMIAL TIME

E.G. SUPPOSE REAL COMPUTERS CAN SIMULATE T.M. OF RUNNING TIME $T(n)$ IN $O(T(n)^3)$ TIME — AND SUPPOSE $T(n) = O(n^c)$ — THEN REAL MACHINE HAS RUNNING TIME

$$O((n^c)^3) = O(n^{3c})$$

<u>THUS</u> ALL SUCH MODELS DEFINE EXACTLY THE SAME CLASS OF TRACTABLE PROBLEMS

b) AS WE SAW ABOVE, POLYNOMIALS OF POLYNOMIALS ARE STILL POLYNOMIALS — THUS NOTION IS <u>ROBUST</u> EVEN WHEN WE COMBINE ALGORITHMS

c) BOUNDARY BETWEEN POLYNOMIAL & NON-POLY TIMES IS SEVERE AND EASILY FELT IN PRACTICE

<u>E.G.</u> PLOT n^4 AND 2^n AND SEE HOW IT GROWS AS n INCREASES

d) MOST "GOOD" ALGORITHMS KNOWN IN PRACTICE HAVE RUNNING TIME WHICH IS $O(n^c)$ FOR VALUES OF c WHICH ARE TYPICALLY ≤ 3 AND SELDOM ≥ 6.

<u>TIME COMPLEXITY CLASSES.</u>

<u>DEFN</u>

$$P = \left\{ L \;\middle|\; \begin{array}{l} L \text{ ACCEPTED BY} \\ \text{POLY-TIME } \underline{\text{DTM}} \end{array} \right\}$$

$$NP = \left\{ L \;\middle|\; \begin{array}{l} L \text{ ACCEPTED BY} \\ \text{POLY-TIME } \underline{\text{NTM}} \end{array} \right\}$$

<u>REMARK</u>
- BOTH NTM/DTM MUST BE HALTING T.M.
- CAN EASILY EXTEND DEFINITIONS TO ARBITRARY FUNCTION COMPUTATION.

FACT $P \subseteq NP$.

OBSERVE BEING IN P CORRESPONDS TO OUR INTUITION OF BEING EFFICIENTLY SOLVABLE, WHEREAS BEING RECURSIVE MERELY IMPLIES BEING SOLVABLE.

QUESTION WHAT DOES BEING IN NP MEAN?

INFORMALLY BEING IN NP MEANS THAT GIVEN A SOLUTION ITS CORRECTNESS CAN BE EFFICIENTLY VERIFIED

TO SEE THIS LETS CONSIDER SOME SAMPLE PROBLEMS.

SATISFIABILITY

BOOLEAN FORMULA
- OPERATORS AND ($\wedge$), OR ($\vee$), NOT ($\neg$)
- OPERANDS $x_1, x_2, \ldots, x_n$
- FORMULA $F(x_1, x_2, \ldots, x_n)$
- EACH x_i IS EITHER TRUE (1), OR FALSE (0).

SATISFIABILITY PROBLEM. GIVEN A FORMULA $F(x_1, x_2, \ldots x_n)$ DOES THERE EXIST AN ASSIGNMENT OF 0/1 VALUES (TRUTH ASSIGNMENT) TO BOOLEAN VARIABLES $x_1, \ldots x_n$ WHICH SATISFIES F (CAUSES IT TO EVALUATE TO TRUE OR 1)?

EXAMPLE $F(x_1, x_2, x_3) = (x_1 \wedge x_2 \wedge \neg x_3) \vee (\neg x_1 \wedge x_2 \wedge x_3)$

SATISFIABLE BY T. A. $x_1 \leftarrow 1,\ x_2 \leftarrow 1,\ x_3 \leftarrow 0$.

EXAMPLE $F(x_1, x_2, x_3) = (x_1 \wedge x_2 \wedge \neg x_3) \wedge (\neg x_1 \wedge x_2 \wedge x_3)$

NOT SATISFIABLE

THEOREM SATISFIABILITY IS IN NP.

PROOF
- FIRST WE MOST CONVERT TO A LANGUAGE PROBLEM.
- FOR THIS MUST ENCODE FORMULAS AS STRINGS.
  - $\Sigma = \{ \wedge, \vee, \neg, (,), x, 0, 1 \}$
  - VARIABLE x_i X <i IN BINARY>

    e.g. X_5 ENCODED AS X101
  - CLEARLY $F(x_1, \dots x_n)$ CAN NOW BE ENCODED AS A STRING OVER Σ.
- LANGUAGE $L_{SAT} = \left\{ \omega \;\middle|\; \begin{array}{l} \omega \text{ ENCODES A FORMULA } F \\ \text{WHICH IS SATISFIABLE} \end{array} \right\}$

GOAL SHOW $L_{SAT} \in NP$.

POLY-TIME NTM N

STEP 1 "GUESS" TRUTH ASSIGNMENT FOR $x_1, \dots, x_n$

STEP 2 EVALUATE F ON TRUTH ASSIGNMENT AND VERIFY RESULT IS TRUE

<u>CLEARLY</u> F SATISFIABLE $\iff \exists$ SATISFYING T.A.

$\iff$ N HAS ACCEPTING EXECUTION.

<u>RUNNING TIME</u> $\begin{cases} \text{STEP 1} — O(n) \\ \text{STEP 2} — O(n^2) \text{ (VERIFY)} \end{cases}$

<u>DONE</u>

<u>REMARK 1</u> ALL SUCH DECISION PROBLEMS CAN BE EASILY CONVERTED TO LANGUAGE PROBLEMS VIA AN ENCODING OF INPUT INTO STRINGS — THIS IS USUALLY SO OBVIOUS THAT WE WILL OMIT THE DETAILS.

<u>REMARK 2</u> AT THIS POINT IT IS UNCLEAR WHETHER $L_{SAT} \in P$. OBSERVE THAT CONVERTING <u>NTM</u> N INTO A <u>DTM</u> DOESN'T WORK, SINCE AS WE SAW EARLIER THE SIMULATING DTM WILL HAVE RUNNING TIME AT LEAST $2^{T(n)} = 2^{n^2}$ WHICH IS NOT POLYNOMIAL TIME

<u>OBSERVE</u> UNDER THE STANDARD NTM$\to$DTM CONVERSION, DTM WILL EFFECTIVELY HAVE TO TRY ALL POSSIBLE T.A. (AND THERE ARE 2^n OF THEM).

<u>OPEN QUESTION</u> $L_{SAT} \in P$?

CONJUNCTIVE NORMAL FORM.

- NOTATION $\begin{cases} + \text{ IS } \vee \text{ (OR)} \\ \cdot \text{ IS } \wedge \text{ (AND)} \end{cases}$

- LITERAL $\quad X_i$ OR $\overline{X_i}$ (VARIABLE OR ITS NEGATION)

- CLAUSE $\quad$ SUM | OR OF LITERALS

$$C_j = x_1 + \overline{x_2} + x_3$$

- CNF FORMULA $\quad$ "AND" OF CLAUSES

$$F(x_1, x_2, x_3) = (x_2 + \overline{x_3}) \cdot (x_1 + x_2 + x_3)$$

- THUS $\quad F = \prod_{j=1}^{m} C_j$, EACH $C_j = \sum_{i=1}^{t} x_i$

CSAT PROBLEM $\quad$ INPUT $\quad$ ~~CNF~~ FORMULA F

PROBLEM $\quad$ DECIDE IF F IS SATISFIABLE

CLEARLY $\quad$ CSAT $\in$ NP $\quad$ (SPECIAL CASE OF SAT)

---

k-CNF FORMULA $\quad$ EACH CLAUSE HAS EXACTLY k LITERALS

k-SAT PROBLEM $\quad$ SATISFIABILITY OF k-CNF ~~PROP~~ FORMULA

1-SAT $\quad (X_1) \cdot (\overline{X_2}) \cdot (X_3)$

2-SAT $\quad (X_1 + \overline{X_2}) \cdot (X_2 + X_3) \cdot (\overline{X_1} + X_3)$

3-SAT $\quad (X_1 + \overline{X_2} + X_3) \cdot (\overline{X_1} + X_2 + X_3)$

$\vdots$

__OBSERVE__ SATISFIABILITY IS AN IMPORTANT PROBLEM ARISING IN A NUMBER OF CRITICAL APPLICATIONS — A.I., THEOREM PROVING, CIRCUIT VERIFICATION, ...

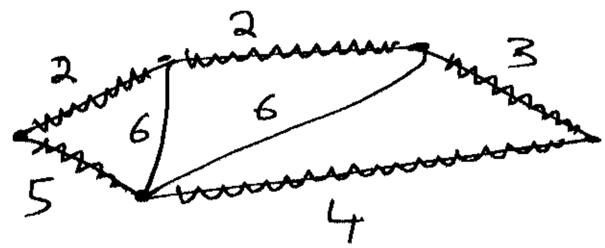
__k-SAT__

- 1-SAT $\in$ P (EASY)
- 2-SAT $\in$ P (HARD)
- __OPEN__ 3-SAT $\in$ P?

__TRAVELING SALESMAN PROBLEM.__

__INPUT__ GRAPH $G(V,E)$, EDGE LENGTHS $\ell(u,v)$, AND INTEGER k.

__PROBLEM__ DOES G HAVE TOUR (VISITING EACH VERTEX EXACTLY ONCE) OF TOTAL LENGTH $\leq k$?



MINIMUM TOUR LENGTH = 16.

__THEOREM__ TSP $\in$ NP (__ACTUALLY__ $L_{TSP} \in NP$)

__PROOF__ __NTM N__ $\begin{cases} \text{STEP 1} & \text{"GUESS" SOME TOUR} \\ \text{STEP 2} & \text{VERIFY LENGTH IS} \leq k \end{cases}$

__CLEARLY__ NTM N RUNS IN $O(n^c)$ TIME.

__QUESTION__ IS TSP $\in$ P?

__PROBLEM__ CAN'T SEE HOW TO DO MUCH BETTER THAN TRYING ALL $n!$ POSSIBLE TOURS

$$n! = O\left(\left(\tfrac{n}{e}\right)^n\right) \text{ IS NOT POLYNOMIAL}$$

<u>PATTERN</u> WHILE ATTEMPTING TO DESIGN EFFICIENT ALGORITHMS FOR PROBLEMS ARISING IN PRACTICE, PEOPLE BEGAN TO NOTICE SOME PATTERNS

- <u>MATRICES</u>
  - DETERMINANT $\in P$
  - PERMANENT — NO POLY-TIME ALGORITHM SEEMS POSSIBLE
- <u>GRAPHS</u>
  - SHORTEST PATH $\in P$
  - LONGEST PATH — NO POLY-TIME ALGO. IS KNOWN
- <u>BOOLEAN FORMULA</u>
  - 1-SAT, 2-SAT $\in P$
  - 3-SAT — NO POLY-TIME ALGO. IS KNOWN.

<u>HOWEVER</u> FOR ALL THE SEEMINGLY INTRACTABLE PROBLEMS, IT IS ESTABLISHED RATHER EASILY THAT THEY BELONG TO NP VIA THE FOLLOWING GENERIC NTM

<u>STEP 1</u> "GUESS" SOME SOLUTION S

<u>STEP 2</u> VERIFY THAT S IS A VALID SOLUTION

<u>OBSERVE</u> THAT FOR NP PROBLEMS, STEP 2 REQUIRES THAT VERIFYING CORRECTNESS OF SOLUTION CAN BE DONE IN POLY-TIME (DETERMINISTICALLY).

<u>PROBLEM</u> MAKING STEP 1 DETERMINISTIC ENTAILS TRYING ALL POSSIBLE SOLUTION IN A BRUTE-FORCE MANNER — AND NUMBER OF SOLUTIONS IS EXPONENTIAL!

<u>SOON</u> THERE THOUSANDS OF PROBLEMS KNOWN TO BE IN NP — BUT FOR WHICH WE FAILED TO FIND POLY-TIME ALGO.

<u>OBSERVE</u> $P \subseteq NP$.

<u>SUPPOSE</u> $P = NP$, THEN THESE PROBLEMS WOULD BE GUARANTEED TO HAVE EFFICIENT (POLY-TIME) ALGORITHMS & WE COULD FOCUS OUR ENERGIES LOOKING FOR ALGO.

<u>HOWEVER</u> WE CURRENTLY HAVE A STRONG BELIEF THAT

$$P \neq NP \qquad (P \subset NP)$$

<u>NOW</u> ASSUMING $P \neq NP$, HOW DO WE DETERMINE WHICH PROBLEMS OF NP DON'T BELONG TO P, SO WE DON'T WASTE OUR TIME TRYING TO FIND ALGO. WHICH DON'T EXIST ?

<u>NP-COMPLETENESS</u> HELPS US TO DEAL WITH THIS SITUATION.

<u>OPEN</u> IS $P = NP$?

<u>THIS</u> IS ONE OF THE DEEPEST UNSOLVED QUESTION IN MATHEMATICS TODAY!

<u>KEY IDEA</u> WE WILL DEFINE A PROBLEM TO BE <u>NP-COMPLETE</u> IF SHOWING THAT IT IS IN P WOULD IMPLY THAT $P = NP$.

<u>THUS</u> GIVEN THE ASSUMPTION THAT $P \neq NP$, IT IS NOT POSSIBLE FOR AN NP-COMPLETE PROBLEM TO BE IN P.

<u>REMARKABLY</u> ALMOST ALL PROBLEMS NOT KNOWN TO BE IN P TURN OUT TO BE NP-COMPLETE.

<u>POLY-TIME REDUCTION</u> PROBLEM X REDUCES TO PROBLEM Y IN POLY-TIME ($X <_{POLY} Y$) IF THERE IS A FUNCTION f SUCH THAT

(a) $\omega \in L_X \iff f(\omega) \in L_Y$

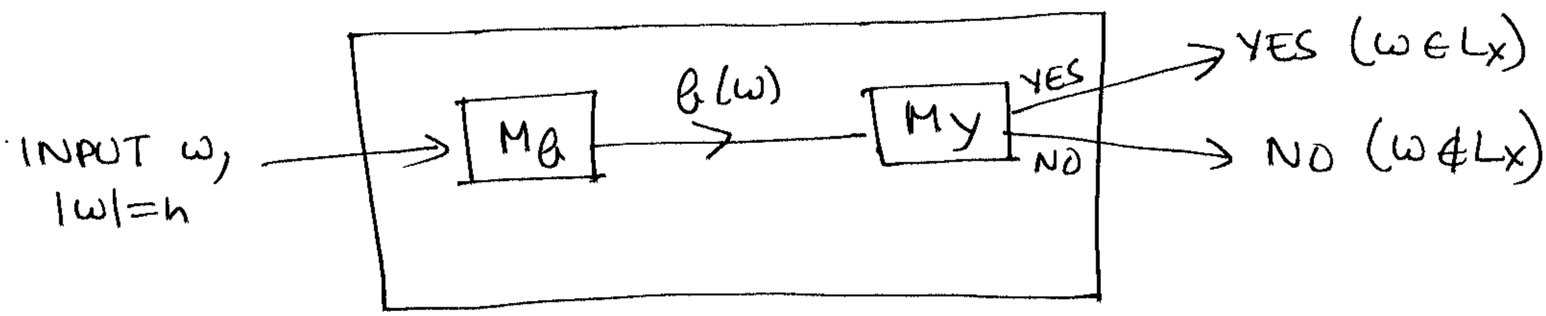
(b) f IS COMPUTABLE BY POLY-TIME DTM.

<u>NOTE</u> L_X IS LANGUAGE ENCODING OF A PROBLEM X.

<u>THEOREM</u> $X <_{POLY} Y$ AND $Y \in P \implies X \in P$.

<u>PROOF</u> <u>LET</u> M_f, M_Y BE POLY-TIME DTMs FOR f, L_Y

<u>CONSTRUCT</u> DTM M_X FOR L_X AS FOLLOWS:



M_X's RUNNING TIME?

SUPPOSE
- M_f RUNS IN TIME $T_f(n) \leq n^a$
- M_Y RUNS IN TIME $T_Y(n) \leq n^b$.

<u>THEN</u> LENGTH OF $f(\omega)$ IS $\leq n^a$

<u>THUS</u> M_X RUNS IN TIME

$$\begin{aligned} T_X(n) &\leq n^a + T_Y(T_f(n)) \\ &\leq n^a + (n^a)^b \\ &= O(n^{ab}) \end{aligned}$$

<u>DONE</u>

COROLLARY $X <_{POLY} Y$ AND $X \notin P \implies Y \notin P$.

DEFN PROBLEM Y (OR, LANGUAGE L_Y) IS NP-HARD IF

$\forall X \in NP, \quad X <_{POLY} Y$.

MEANING? Y IS AT LEAST AS HARD AS ANY PROBLEM X IN NP.

THUS FOR NP-HARD Y,

$$Y \in P \implies \forall X \in NP, X \in P$$
$$\implies P = NP.$$

COROLLARY Y IS NP-HARD AND $Y \in P \implies P = NP$.

DEFINITION Y IS NP-COMPLETE IF

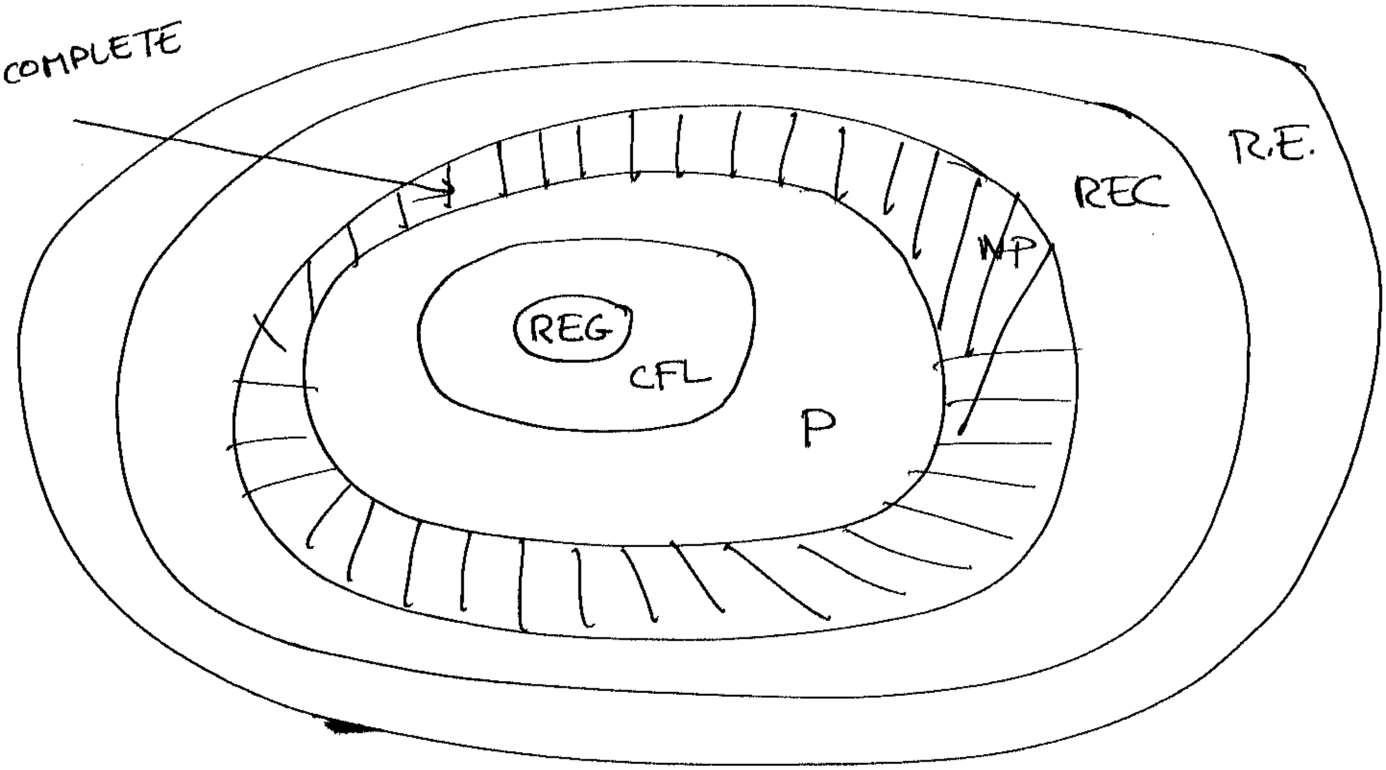
1) $Y \in NP$, AND
2) Y IS NP-HARD.

MEANING NP-COMPLETE PROBLEMS ARE THE HARDEST PROBLEMS IN NP — IF ONE OF THEM IS IN P, THEN ALL PROBLEMS OF NP ARE IN P.

THUS ASSUMING $P \neq NP$, AN NP-COMPLETE PROBLEM CANNOT BE IN P ~~STRONG~~

WE TREAT NP-COMPLETENESS AS STRONG EVIDENCE OF INTRACTABILITY.

WORLD VIEW



QUESTION HOW DO WE SHOW PROBLEMS ARE NP-COMPLETE?

THEOREM X IS NP-HARD, $X <_{POLY} Y \implies Y$ IS NP-HARD

PROOF

$$NP <_{POLY} X <_{POLY} Y.$$

PROBLEM WE NEED A STARTING POINT — THE FIRST NP-HARD PROBLEM (JUST LIKE L_D IN CASE OF UNDECIDABILITY

COOK'S THEOREM 3-SAT IS NP-HARD.

PROOF WILL SHOW LATER — ACTUALLY HE PROVED THAT CSAT IS NP-HARD, BUT EASY TO MODIFY TO SHOW 3-SAT IS NP-HARD.

INDEPENDENT SET PROBLEM

CONSIDER FOLLOWING EXAM SCHEDULING PROBLEM — WE HAVE n COURSES $C_1, C_2, \ldots, C_n$ FOR WHICH WE WOULD LIKE TO SCHEDULE FINAL EXAMS.

HOWEVER CERTAIN PAIRS OF COURSES ARE IN CONFLICT IN THAT IT IS LIKELY THAT SOME STUDENTS ARE REGISTERED FOR BOTH COURSES.

FIND LARGE SET OF COURSES WHOSE EXAMS CAN BE SCHEDULED AT THE SAME TIME.

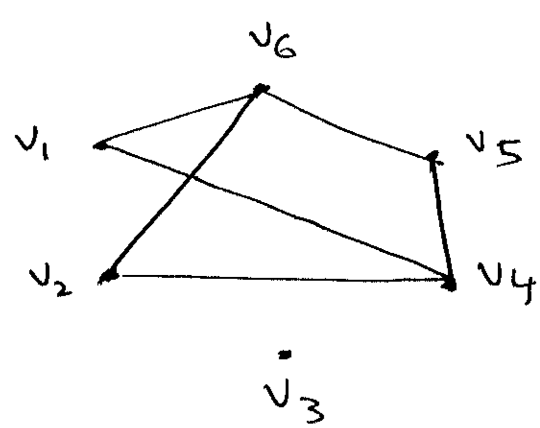
MODEL AS A PROBLEM ON A GRAPH $G(V,E)$

VERTICES $V = \{v_1, v_2, \ldots v_n\}$ — ONE FOR EACH COURSE

EDGES $E = \{ (v_i, v_j) \mid \text{COURSE } C_i \text{ CONFLICTS WITH } C_j \}$

GOAL FIND A LARGE INDEPENDENT SET — WHICH IS A SUBSET OF VERTICES $S \subseteq V$ SUCH THAT NO TWO VERTICES IN S HAVE AN EDGE BETWEEN THEM.

EXAMPLE



INDEPENDET SETS

$\{3, 4, 6\}$

$\{1, 2, 3, 5\}$

I.S. PROBLEM   **INPUT** GRAPH $G(V,E)$ AND INTEGER k.

QUESTION DOES G CONTAIN AN I.S. OF SIZE $\geq k$?

THEOREM I.S. IS NP-COMPLETE

PROOF A) I.S. $\in$ NP

— GUESS $S \subseteq V$, $|S| = k$

— VERIFY ABSENCE OF ALL POSSIBLE $\binom{k}{2}$ EDGES AMONGST PAIRS OF VERTICES IN SET S.

CLEARLY RUNS IN POLY-TIME.

B) I.S. IS NP-HARD   WILL SHOW 3-SAT $<_{POLY}$ I.S.

REDUCTION f

INPUT 3-CNF FORMULA

- $F(x_1, x_2, \ldots, x_n) = C_1 \wedge C_2 \wedge \ldots \wedge C_m$
- $c_i = z_{i1} \vee z_{i2} \vee z_{i3}$ (FOR $1 \leq i \leq m$)
- z_{ij} IS A LITERAL (FOR $1 \leq i \leq m$, $1 \leq j \leq 3$)
- LITERAL — EITHER SOME x_k OR ITS NEGATION $\overline{x_k}$

EXAMPLE

$$F(x_1, x_2, x_3, x_4, x_5) = (x_1 + x_2 + x_3) \cdot (\overline{x_1} + \overline{x_2} + \overline{x_3}) \cdot (\overline{x_1} + x_4 + x_5)$$

B'S OUTPUT GRAPH $G(V,E)$, $k=m$

VERTICES $V = \{ v_{ij} \mid 1 \le i \le m,\ 1 \le j \le 3 \}$

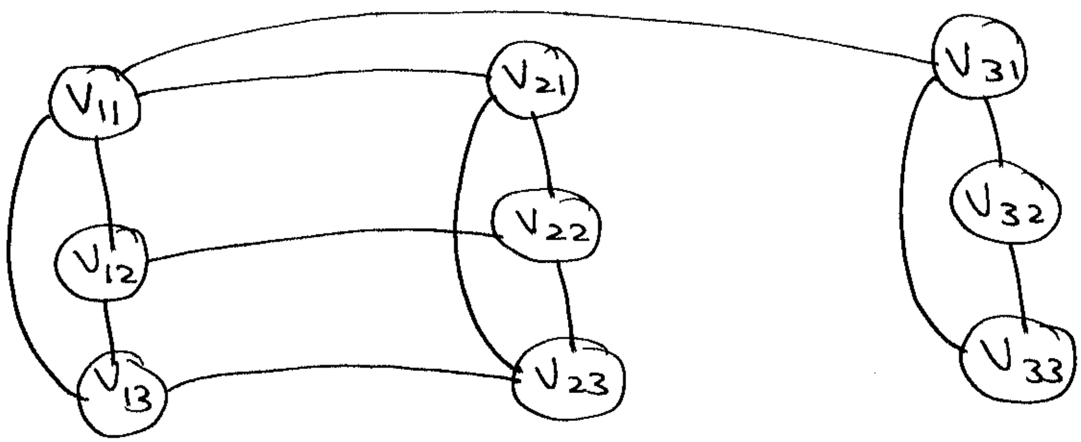
ONE FOR EACH LITERAL z_{ij}

EDGES ARE OF 2 TYPES:

TYPE-A EDGE (v_{ij}, v_{st}) IS PRESENT IF $i=s$

TYPE-B EDGE (v_{ij}, v_{st}) IS PRESENT IF z_{ij} IS NEGATION OF z_{st}.

EXAMPLE $(x_1+x_2+x_3)(\overline{x_1}+\overline{x_2}+\overline{x_3})(\overline{x_1}+x_4+x_5)$



INTUITION I.S. S SHOULD CORRESPOND TO ONE TRUE LITERAL IN EACH CLAUSE

OBSERVE F IS SATISFIABLE IF & ONLY IF THERE IS A T.A. WHICH MAKES AT LEAST ONE LITERAL TRUE IN EACH CLAUSE

TYPE-A EDGES FORCE US TO CHOOSE EXACTLY ONE LITERAL FROM EACH CLAUSE

TYPE-B EDGES PREVENT A VARIABLE & ITS NEGATION BOTH BEING

CLEARLY f CAN BE COMPUTED IN POLY-TIME.

LEMMA F SATISFIABLE $\Longleftrightarrow$ G HAS AN I.S. OF SIZE $\geq k$

PROOF ($\Longrightarrow$)

SUPPOSE F IS SATISFIED BY SOME T.A. τ

CHOOSE ONE "TRUE" z_{ij} FROM EACH CLAUSE C_i AND ADD CORRESPONDING VERTEX v_{ij} TO S.

CLEARLY $|S| = k = m$.

CLAIM S IS AN I.S.

WHY? CONSIDER ANY v_{ij}, v_{st} IN S AS CHOSEN ABOVE.

CLEARLY $i \neq s$ AS ONLY ONE z_{ij} WAS CHOSEN FROM EACH CLAUSE C_i

$\Longrightarrow$ NO TYPE-A EDGE

FURTHER z_{ij}, z_{st} CANNOT BE EACH OTHERS NEGATION AS BOTH ARE "TRUE" IN T.A. τ

$\Longrightarrow$ NO TYPE-B EDGE.

THUS NO EDGE BETWEEN v_{ij} AND v_{st}.

DONE

PROOF ($\Leftarrow$)

SUPPOSE G HAS AN I.S. S OF SIZE $k=m$.

WE SHOW HOW TO CONSTRUCT A T.A. τ WHICH WILL SATISFY F.

OBSERVE S MUST HAVE EXACTLY ONE VERTEX FROM EACH TRIPLE z_{i1}, z_{i2}, z_{i3}

WHY? SINCE TYPE-A EDGES PREVENT SELECTING MORE THAN ONE, AND TOTAL SIZE OF S IS $k=m$ WHICH IS THE NUMBER OF TRIPLES

THUS IF WE WERE TO SET LITERALS CORRESPONDING TO VERTICES IN S TO BE ALL "TRUE", WE WOULD HAVE AT LEAST ONE TRUE LITERAL IN EACH CLAUSE & WOULD SATISFY F.

PROBLEM WHAT IF TWO VERTICES IN S CORRESPOND TO LITERALS WHICH ARE EACH OTHERS NEGATION — THEN WE CAN'T SET BOTH TO BE "TRUE"

BUT OF COURSE, TYPE-B EDGES PREVENT THIS PROBLEM FROM ARISING.

DONE

EXAMPLE ($\Rightarrow$)

SUPPOSE $\tau = (x_1 = T,\ x_2 = F,\ x_3 = F,\ x_4 = T,\ x_5 = T)$ IS THE SATISFYING T.A.

WE SELECT
$$\begin{cases} \text{FROM } C_1 & z_{11} = x_1 \\ \text{FROM } C_2 & z_{22} = \overline{x_2} \\ \text{FROM } C_3 & z_{32} = \overline{x_4} \end{cases}$$

GET $S = \{v_{11}, v_{22}, v_{32}\}$ AS AN I.S.

EXAMPLE ($\Leftarrow$)

SUPPOSE $S = \{v_{12}, v_{21}, v_{31}\}$ IS THE I.S.

WE SET
$$\begin{cases} z_{12} = x_2 = T \\ z_{21} = \overline{x_1} = T \\ z_{31} = \overline{x_1} = T \end{cases}$$

NOTE THIS DOES NOT SPECIFY VALUES FOR x_3, x_4, x_5 — WE CAN CHOOSE THEM AS WE LIKE, SAY WE SET THEM ALL TO "FALSE"

WE GET T.A. $\tau = (x_1 = F,\ x_2 = T,\ x_3 = F,\ x_4 = F,\ x_5 = F$

VERIFY JUST THE VALUES OF x_1, x_2 ARE ENOUGH TO GUARANTEE THAT ALL CLAUSES IN F ARE SATISFIED