NP - COMPLETENESS

A PROBLEM IS GENERALLY CONSIDERED TO BE TRACTABLE IF THERE EXISTS AN ALGORITHM TO SOLVE IT WHOSE RUNTIME IS IN THE CLASS $O(n^k)$ FOR SOME CONSTANT k.

MOST (ALL?) OF THE PROBLEMS WE'VE DISCUSSED HERE BEEN SOLVABLE IN POLYNOMIAL TIME.

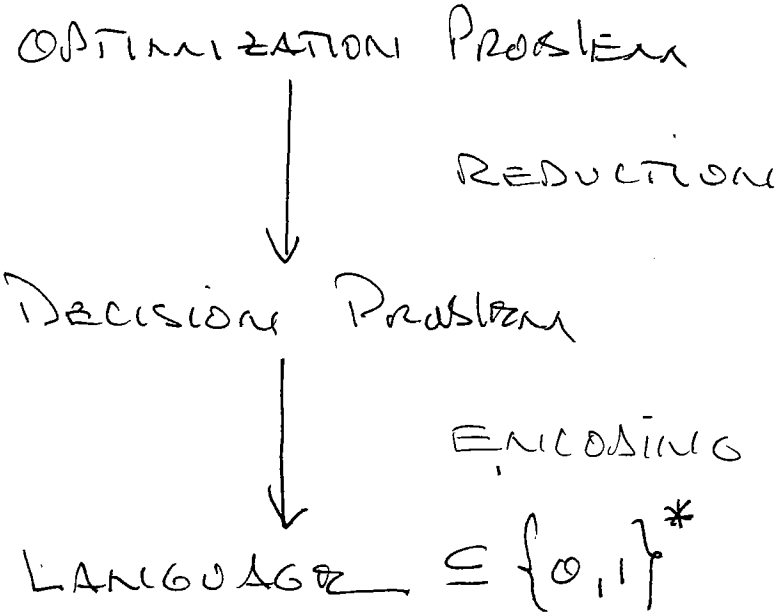
SOME PROBLEMS ARE NOT SOLVABLE NO MATTER HOW MUCH TIME IS PROVIDED (TURING'S HALTING PROBLEM.)

PROBLEMS THAT ARE SOLVABLE, BUT ONLY IN SUPER POLYNOMIAL TIME (e.g. EXPONENTIAL b^n, FACTORIAL $n!$, ETC..) ARE USUALLY CALLED INTRACTABLE.

AN INTERESTING CLASS OF PROBLEMS ARE THE SO CALLED NP COMPLETE PROBLEMS WHICH ARE IN A PRECISE SENSE, AS HARD OR HARDER THAN ANY POLYNOMIAL

TIME SOLVABLE PROBLEM, BUT BUT WHOSE STATUS IS UNKNOWN; NO POLYNOMIAL TIME ALGORITHM IS KNOWN, BUT NEITHER HAS IT BEEN PROVED THAT NO POLY-TIME ALGORITHM EXISTS. THIS IS THE SO-CALLED P≠NP PROBLEM.

TO MAKE ALL THIS CLEAR WE NEED SOME UNIFORM NOTION OF WHAT IS MEANT BY A PROBLEM.



MANY PROBLEMS OF INTEREST ARE OPTIMIZATION PROBLEMS.

Ex. SHORTEST-PATH
INPUT: $G, u, v \in V(G)$
OUTPUT: A SHORTEST u-v PATH IN G

THE THEORY OF NP-COMPLETENESS HOWEVER APPLIES TO DECISION PROBLEMS, i.e. PROBLEMS WHOSE OUTPUT IS yes/no, true/false, or 1/0. FORTUNATELY EVERY OPTIMIZATION PROBLEM HAS A RELATED DECISION PROBLEM.

Ex. PATH
INPUT: $G, u, v \in V(G), k \in \mathbb{Z}$
OUTPUT: yes (1) IF THERE EXIST A u-v PATH IN G OF length AT MOST k, NO (0) OTHERWISE.

NOTE THAT THE RELATED DECISION PROBLEM IS "NO HARDER THAN" THE OPTIMIZATION PROBLEM.

Ex. ANY ALGORITHM THAT SOLVES SHORTEST-PATH CAN BE CONSTRUCTED TO ONE THAT SOLVES PATH BY JUST FINDING A SHORTEST u-v

IN G, THEN RETURN YES IF ITS length is $\leq k$, NO OTHERWISE.

THIS RELATIONSHIP WORKS IN OUR FAVOR WHEN WE ARE TRYING TO SHOW THE OPTIMIZATION PROBLEM TO BE HARD.

NOTICE IN THIS EXAMPLE THAT THE CONVERSION TAKES ONLY POLYNOMIAL (IN FACT CONSTANT) TIME.

TO BE DEFINED

DECISION PROB $\leq$ OPTIMIZATION PROB

↑ SHOW THIS IS HARD

↑ CONCLUDE THIS IS HARD.

IN PARTICULAR IF THERE IS NO POLYNOMIAL TIME ALGORITHM FOR THE DECISION PROBLEM, AND THE CONVERSION TAKES POLYNOMIAL TIME, THEN THERE CAN BE NO POLYNOMIAL TIME ALGORITHM FOR THE OPTIMIZATION PROBLEM.

FROM NOW ON WE CONSIDER ONLY DECISION PROBLEMS.

WE CAN USE THIS SAME TECHNIQUE TO COMPARE TWO DECISION PROBLEMS AS FOLLOWS.

SUPPOSE WE HAVE TWO DECISION PROBLEMS A AND B. FURTHER SUPPOSE WE HAVE A POLYNOMIAL TIME ALGORITHM THAT CONVERTS ANY INSTANCE α OF A INTO AN INSTANCE β OF B IN SUCH A WAY SO THAT

THE ANSWER TO α IS YES
IFF
THE ANSWER TO β IS YES

(WE CALL THIS A <u>POLYNOMIAL TIME REDUCTION</u>.)

FINALLY SUPPOSE WE CAN SHOW THAT A HAS NO POLYNOMIAL TIME ALGORITHM.

WE MAY THEN CONCLUDE THAT B HAS NO POLYNOMIAL TIME ALGORITHM EITHER. WHY?

Proof:
ASSUME, TO GET A CONTRADICTION, THAT B HAS A POLYNOMIAL TIME ALGORITHM. TAKE ANY INSTANCE α OF A, CONVERT IT TO AN INSTANCE β OF B IN POLYNOMIAL TIME USING THE REDUCTION ALGORITHM, SOLVE β IN POLYNOMIAL TIME USING OUR ALGORITHM FOR B, THEN RETURN THE ANSWER AS THE ANSWER FOR α.

THIS PROCEDURE GIVES A POLYNOMIAL TIME ALGORITHM FOR A, CONTRARY TO HYPOTHESIS.

$\therefore$ NO SUCH ALGORITHM FOR B CAN EXIST.

//

WE CAN SAY (INFORMALLY) WHAT WAS MEANT BY $\leq$ IN OUR EARLIER DISCUSSION.

DEFN (STILL INFORMAL). GIVEN TWO DECISION PROBLEMS A AND B, WE WRITE

$$A \leq_P B$$

IFF THERE EXISTS A POLYNOMIAL TIME REDUCTION ALGORITHM FROM A TO B. (AS ABOVE 'YES' INSTANCES OF A ARE TRANSFORMED TO 'YES' INSTANCES OF B, RESP. 'NO'.)

WE SAY IN THIS CASE THAT A is <u>POLYNOMIAL TIME REDUCIBLE</u> TO B.

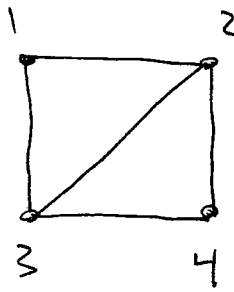
WE STILL MUST FORMALIZE OUR NOTION OF A (DECISION) PROBLEM.

AN <u>ENCODING</u> OF A PROBLEM Q IS A MAPPING e WHICH TRANSFORMS EACH INSTANCE α OF Q INTO A BIT STRING

$$e(\alpha) \in \{0,1\}^*$$

$$\| \quad \{\text{BIT STRINGS OF ANY length}\}$$

<u>EX.</u> AN INSTANCE OF PATH WOULD BE



1 TO 4

$K = 2$

ADJACENCY LIST REPRESENTATION

1 :	2	3	
2 :	1	3	4
3 :	1	2	4
4 :	2	3	

$= \alpha$

$s = 1$

$d = 4$

$K = 2$

↓ ASCII or Unicode

$$e(\alpha) \in \{0,1\}^*$$

WHEN A PROBLEM IS TRANSFORMED VIA SOME ENCODING, INTO A COLLECTION OF BIT STRINGS WE CALL IT A <u>CONCRETE PROBLEM</u>.

USING SOME AN ENCODING WE CAN DEFINE PRECISELY WHAT WE MEAN BY THE SIZE OF AN INSTANCE α OF Q:

$$n = |e(\alpha)| = \#\text{BITS in } e(\alpha).$$

WE AGREE TO MEASURE THE RUN TIME OF AN ALGORITHM FOR Q VIA A FUNCTION

$T(n) = \#$ BASIC OPS. PERFORMED ON INPUT OF SIZE n (DEFINED AS ABOVE.)

A CONCRETE PROBLEM is CALLED <u>POLYNOMIAL TIME SOLVABLE</u> IFF

$$T(n) = O(n^k)$$

FOR SOME CONSTANT k (NOT NECESSARILY AN INTEGER).

WE NOW DEFINE THE COMPLEXITY CLASS P TO BE

$P = \{$ CONCRETE DECISION PROBLEMS THAT ARE POLYNOMIAL TIME SOLVABLE $\}$