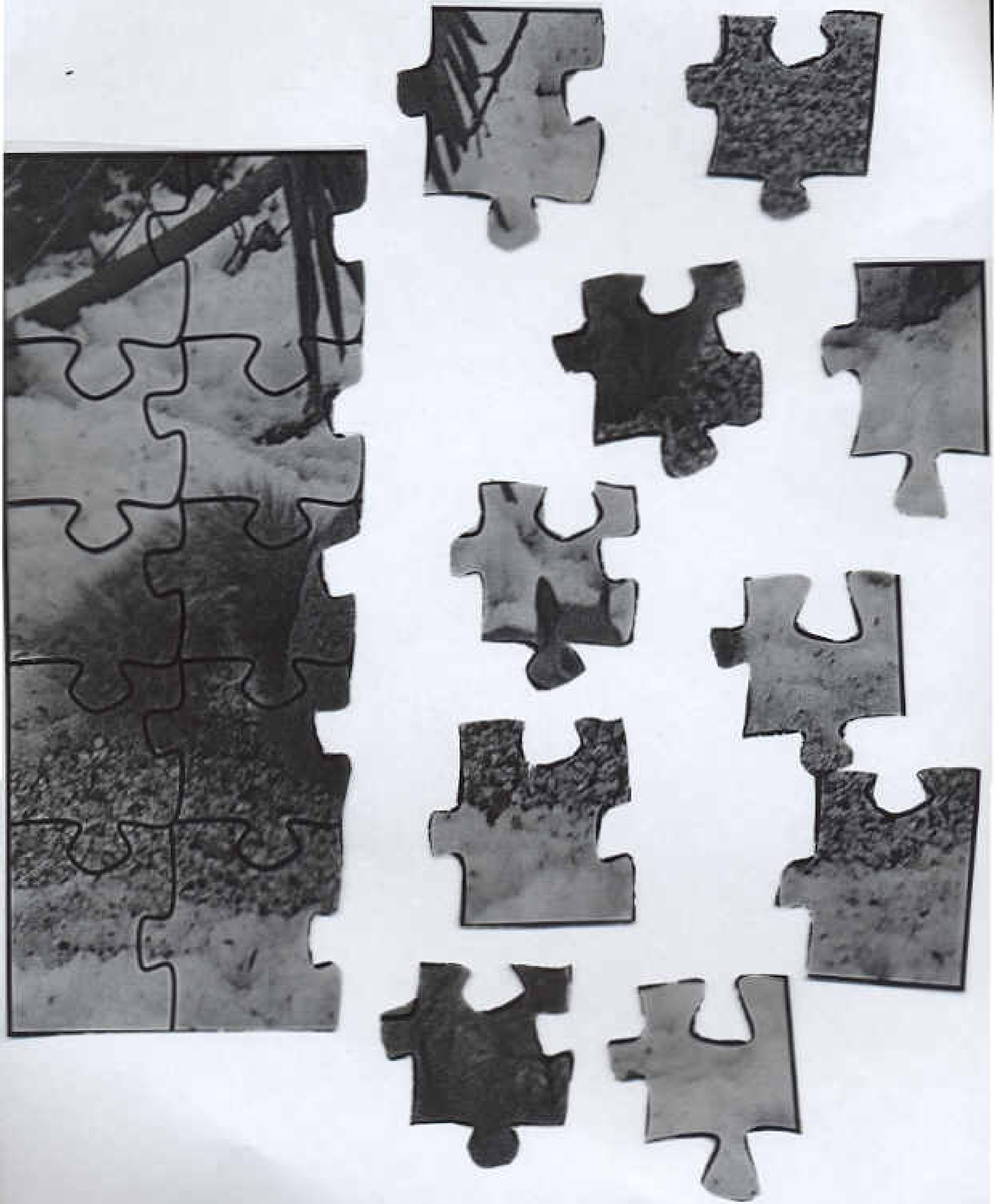
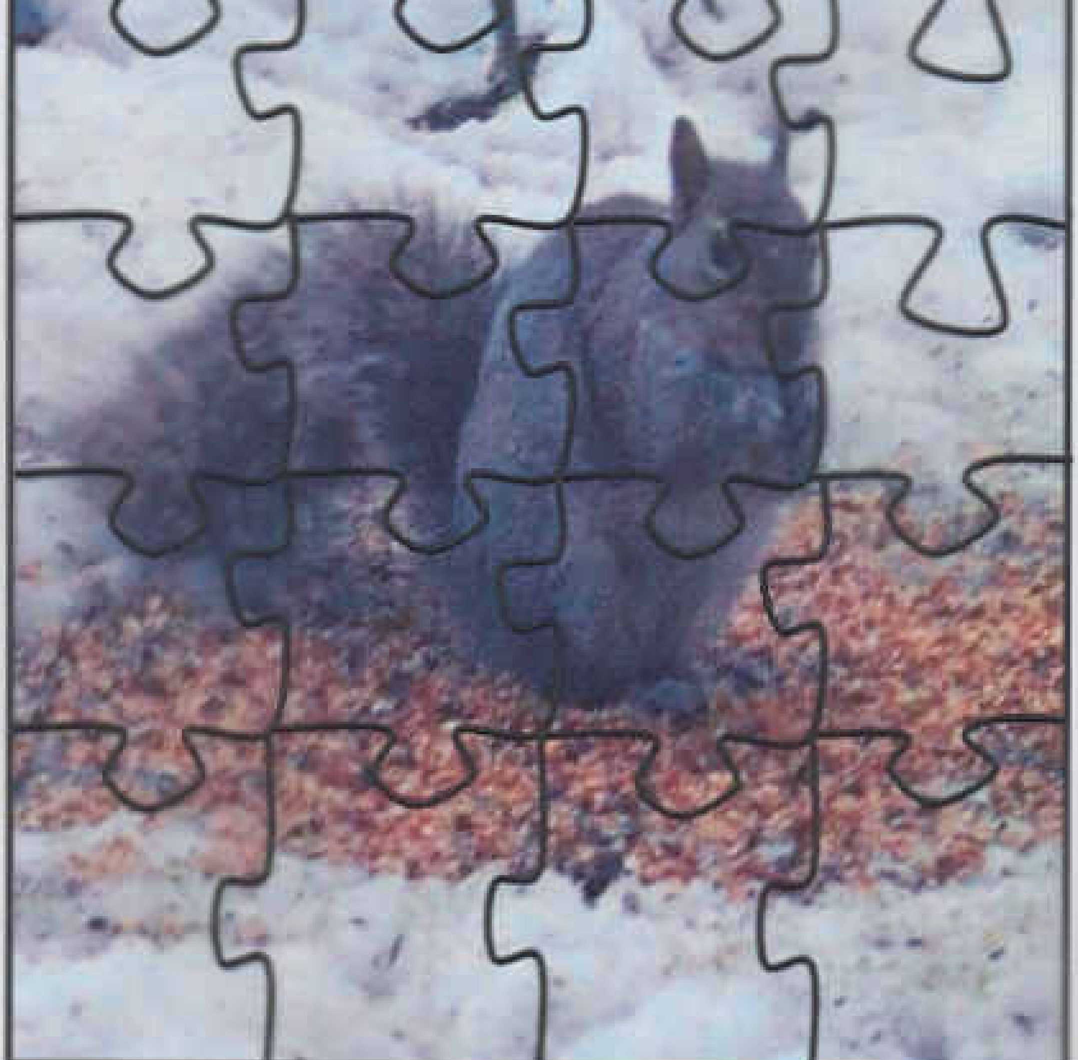
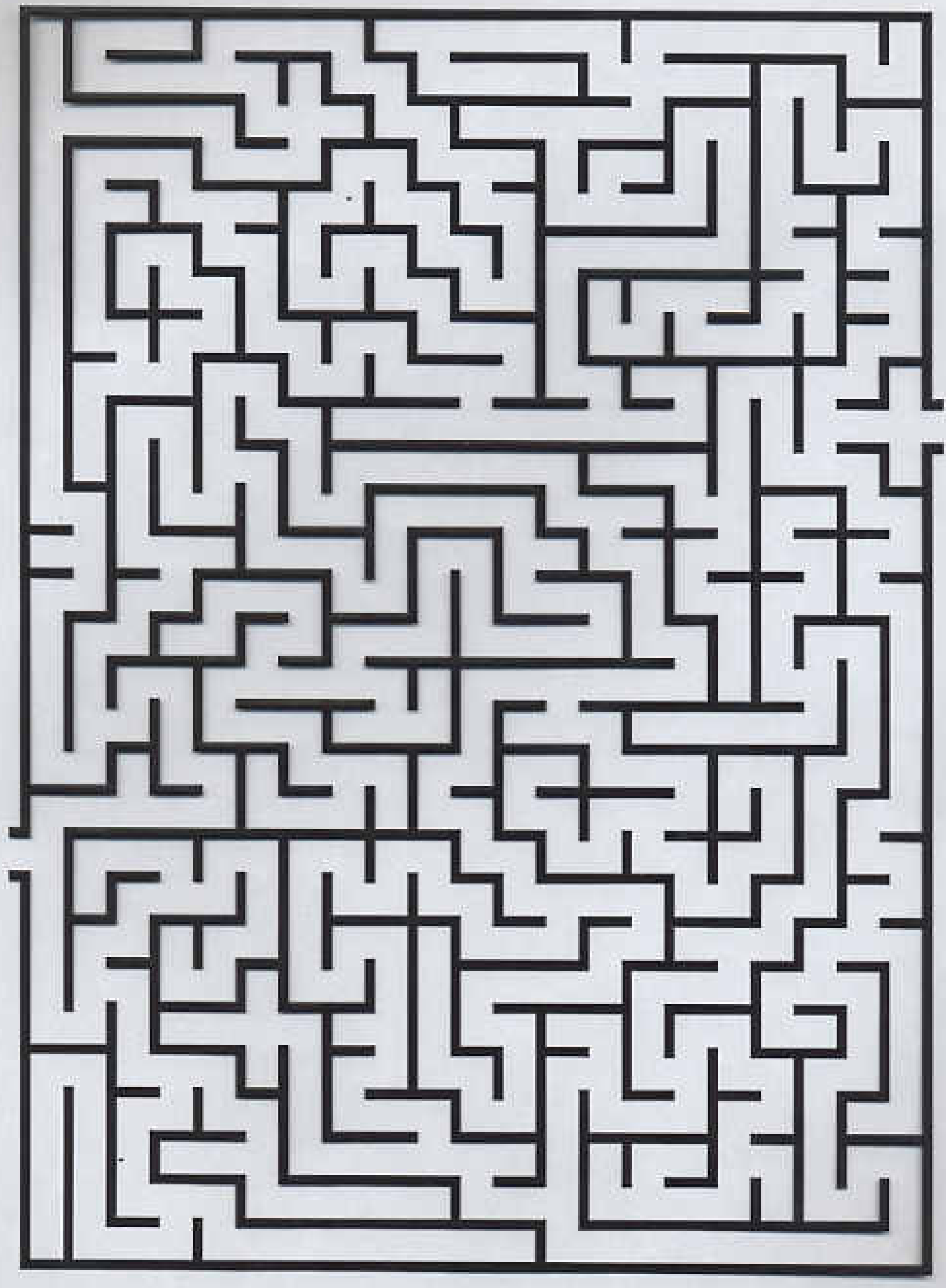
BACKGROUND

- P vs NP
- NP-COMPLETENESS



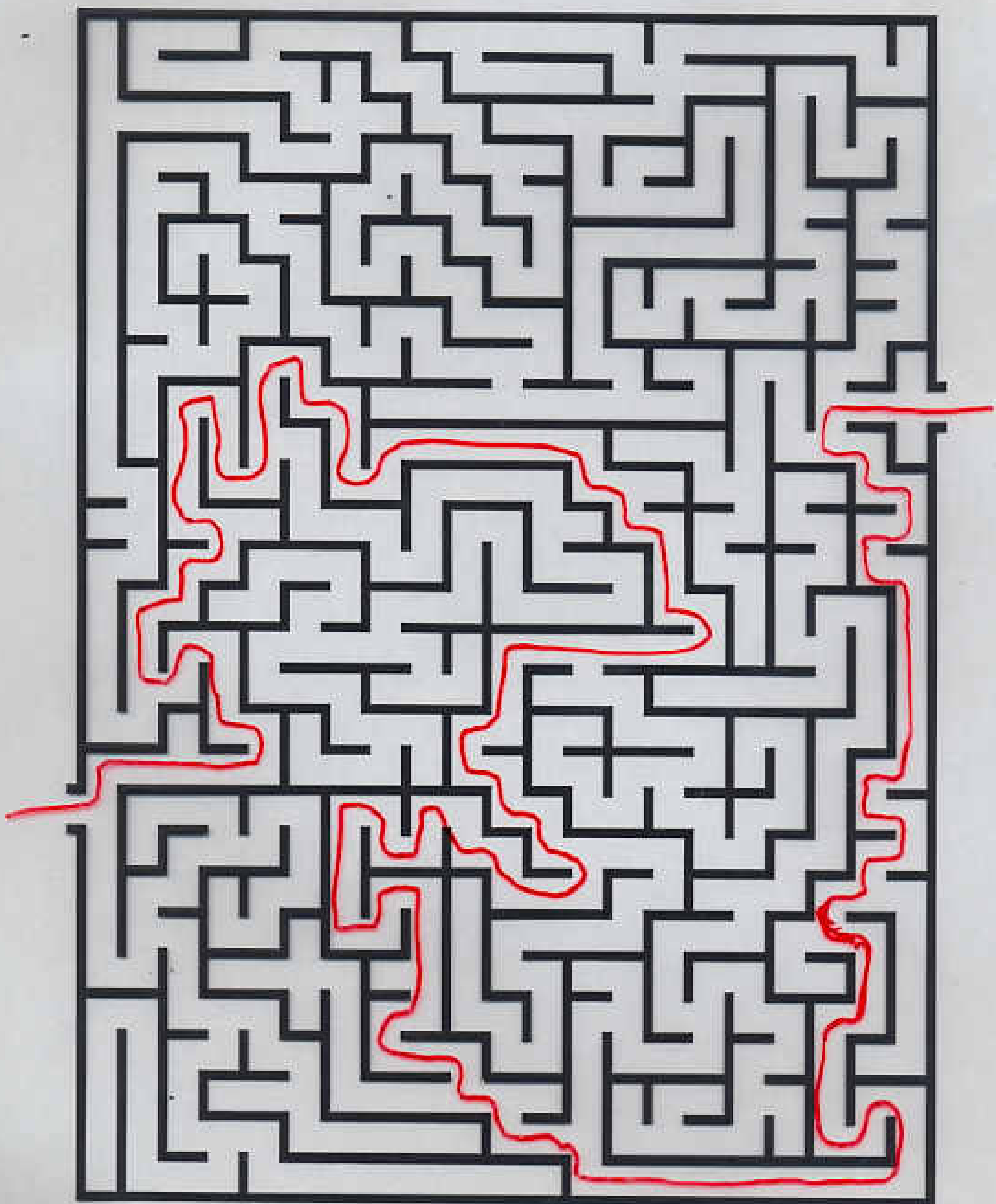


PROBLEM



PROBLEM

SOLUTION



PROBLEM

$$X^2 + 3XY + 7YZ^3 = 1$$

$$X + Y + Z = 4$$

$$XYZ + 2X - Y - Z = -4$$

SOLUTION

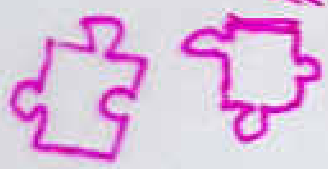
$X = 2 \quad Y = 3 \quad Z = -1$

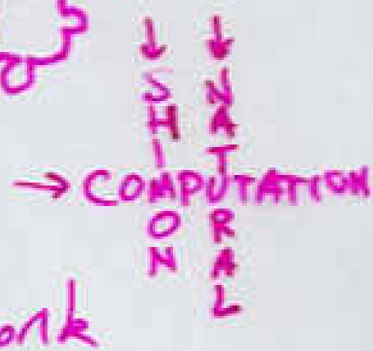
P vs NP: searching vs checking

- Scheduling / Assignment Problems

  e.g. time table (courses, teachers / students, rooms) (times)

  jobs on a single machine (release, deadline, length)

  jigsaw puzzle 

  crossword puzzle 

- Flow Problems

  over a physical/abstract nework

  (e.g., pipes, wires, roads, airlines)

- Routing Problems

  (Short) paths under various constraints

  e.g., finding a labyrinth path 

- Arithmetic/Algebraic problems

  e.g., solving a system of equations

EASY TO CHECK correctness OF SOLUTIONS vs HARD TO FIND correct SOLUTIONS

P vs NP: proving vs verifying

The notion of a PROOF presupposes that verifying validity of proofs is easier than finding them.

(Finding proofs is a search problem for which correct solutions are easy to validate.)

$\Longleftrightarrow$ $P \neq NP$ (RE: TRAD. PROOFS...)

EASY TO VERIFY VALIDITY OF PROOFS

vs

HARD TO FIND CORRECT PROOFS

EFFICIENT VS INFEASIBLE

- (usually) MUST READ THE INPUT

  ⇒ MUST SPEND $\underbrace{\text{LINEAR TIME}}_{\text{time linear in the input length}}$.

- LINEAR TIME IS EFFICIENT
- What about quadratic time?
  (e.g. INTEGER MULTIPLICATION via ELEMENTARY School Method)

  ⋮

  ↕ POLYNOMIAL-TIME IS EFFICIENT

  ⋮

- EXPONENTIAL-TIME IS INFEASIBLE
  (e.g., the naive FACTORING ALGORITHM)

BK

EFFICIENT VS INFEASIBLE

- (usually) MUST READ THE INPUT

  $\Longrightarrow$ MUST SPEND LINEAR TIME.

  time linear in the input length

- LINEAR TIME IS EFFICIENT
- What about quadratic time?
  (e.g. INTEGER MULTIPLICATION via ELEMENTARY School Method)

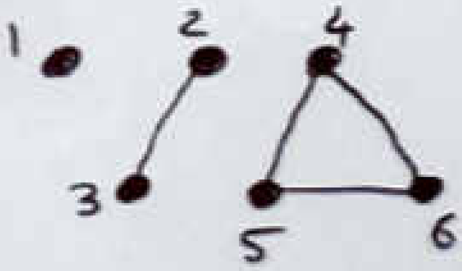
  POLYNOMIAL-TIME IS EFFICIENT

n	10	20	100
n^2	100	400	10,000
2^n	1000	1 M	$\underbrace{1,000,\ldots\ldots000}_{30\text{ ZEROS}}$

- EXPONENTIAL-TIME IS INFEASIBLE
  (e.g., the naive FACTORING ALGORITHM)

GRAPH THEORY

GRAPH $\triangleq$ SET OF POINTS (vertices)
+ SET OF PAIRS OF POINTS (edges).



POINTS = {1,2,3,4,5,6}

EDGES = {(2,3), (4,6), (4,5), (5,6)}

GRAPHS REPRESENT VARIOUS NATURAL OBJECTS

e.g., - Networks (of ROADS, COMMUNICATION, PIPES, etc.)

- (binary) RELATIONS
(e.g., who are related / friends)

SOLVING (real) PROBLEMS BY CONSIDERING
their ABSTRACTION as graph problems.
(e.g., MATCHING, 3-COLORING, HAM. CIRCLE, etc.)

NP - completeness

"מעז יצא מתוק"

<u>Conjecture</u>: $P \neq NP$ (widely believed <u>but</u> <u>unproven</u>)

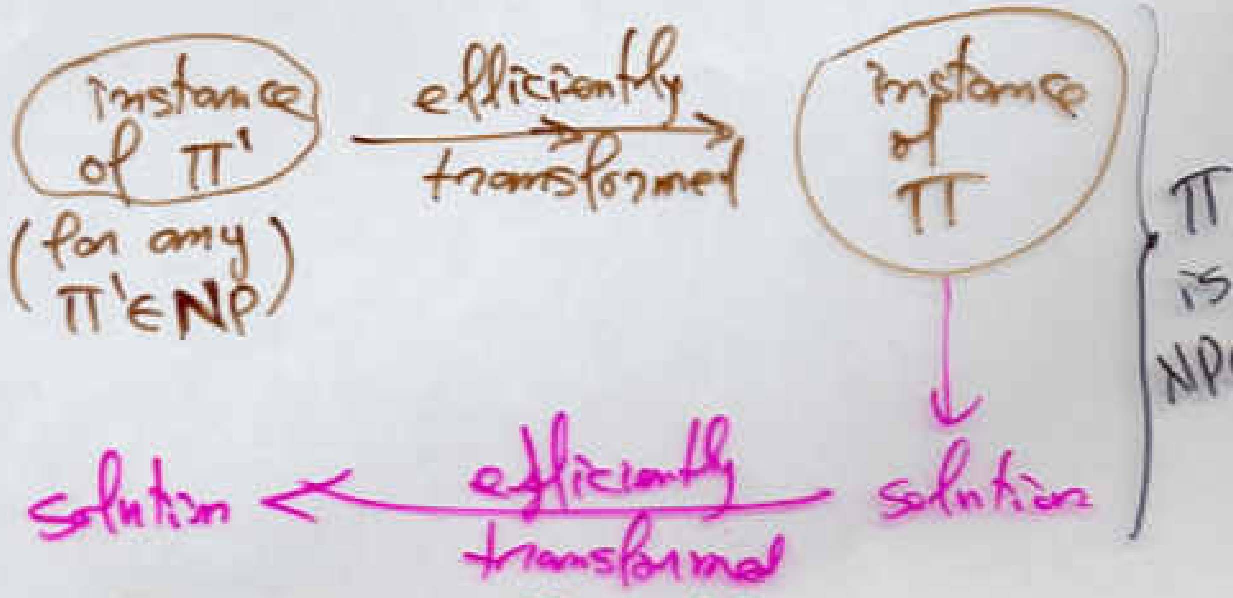
<u>Phenumenon</u>: Natural (many) "NP Problems" have no efficient solvers (i.e. are not in **P**).

For such $\Pi \in NP$, we wish to show $\Pi \notin P$ but ... this would yield $P \neq NP$ ...

Instead, we can hope to show that "if $\Pi \in \mathbf{P}$ then $\mathbf{NP} = \mathbf{P}$".

$\Rightarrow$ $\mathbf{P} \neq \mathbf{NP}$ implies that $\Pi \notin \mathbf{P}$

<u>METHOD</u>: Show that any problem in **NP** can be "reduced" to Π.



$\Rightarrow$ Π "encodes" all problems in NP.

NP-Completeness (cont.)

Solving <u>systems</u> of (quadratic) equations is NP-complete

⟹ Each problem in NP

e.g. – scheduling
– finding short TSP
– factoring integers

CAN BE "ENCODED" (reduced to solving) A SYSTEM OF EQUATIONS.

- TSP ≡ (1) Dist. between pairs of cities
  (2) Need a path that visits all cities (tour)

- SCHEDULING over a single processor/machine a seq. of jobs w. release, length & deadline.

- Factoring: Composite number ⟹ prime factorization